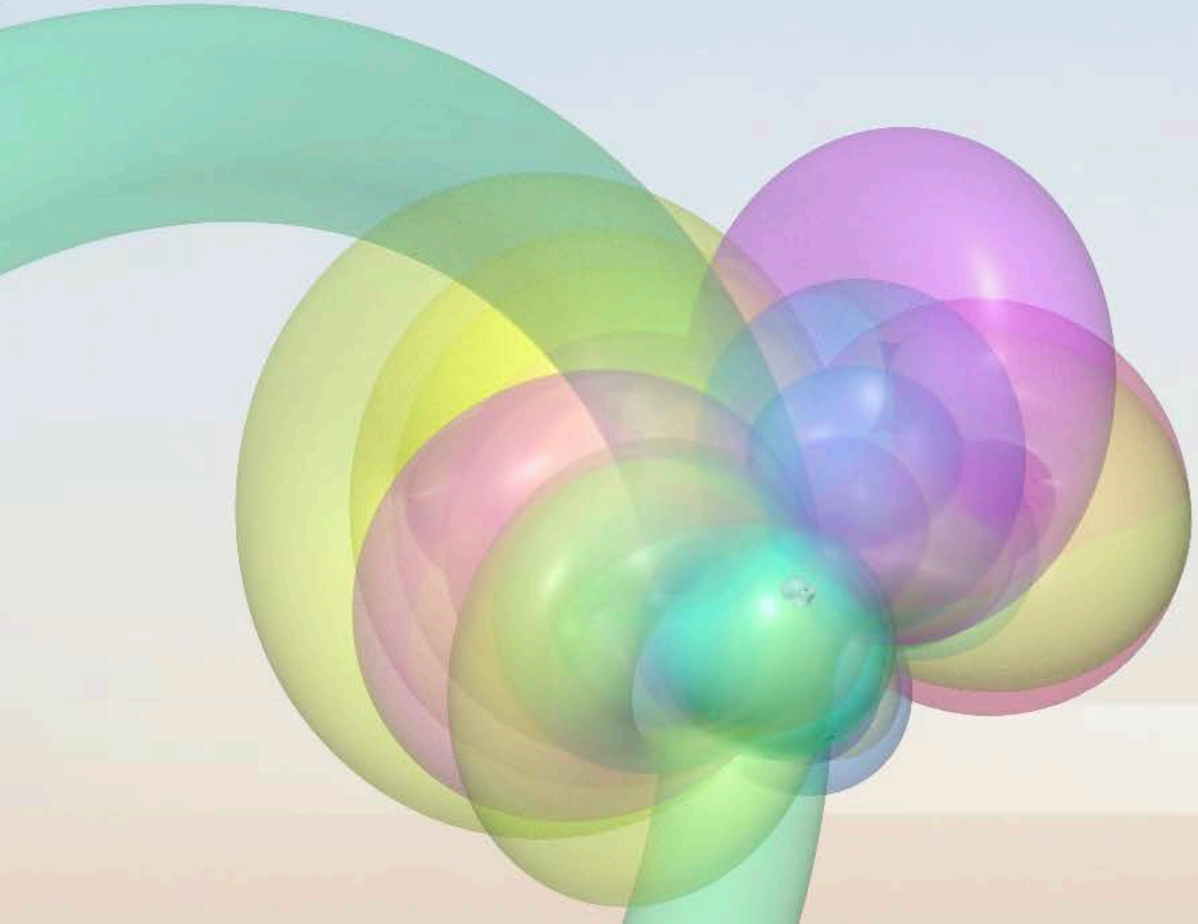
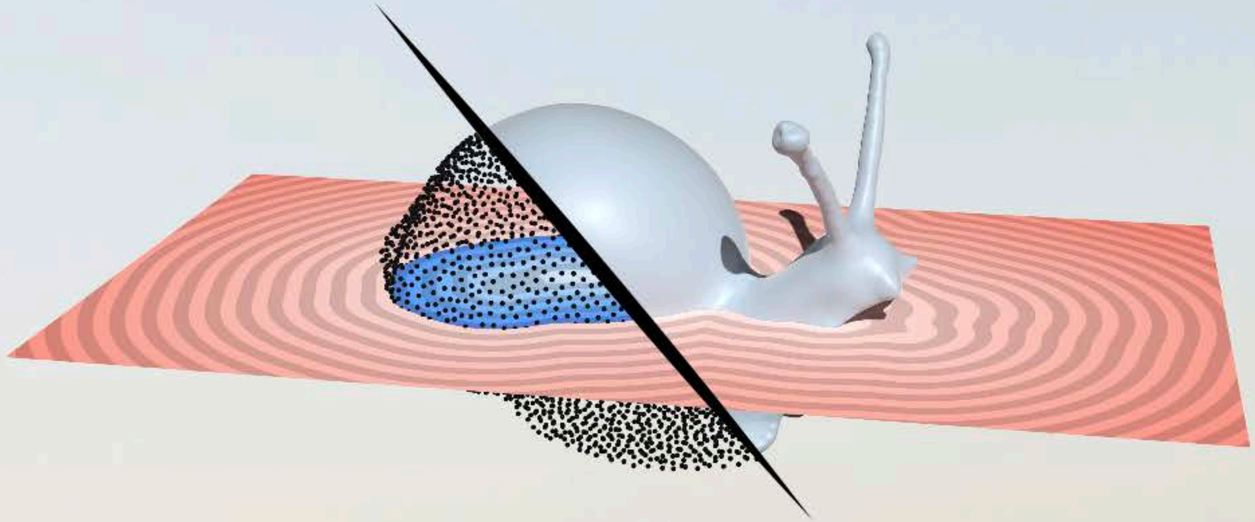
Points as Tori: Fast Pointwise Signed Distance for Point Clouds

Nicole Feng, Ioannis Gkioulekas*, Keenan Crane*

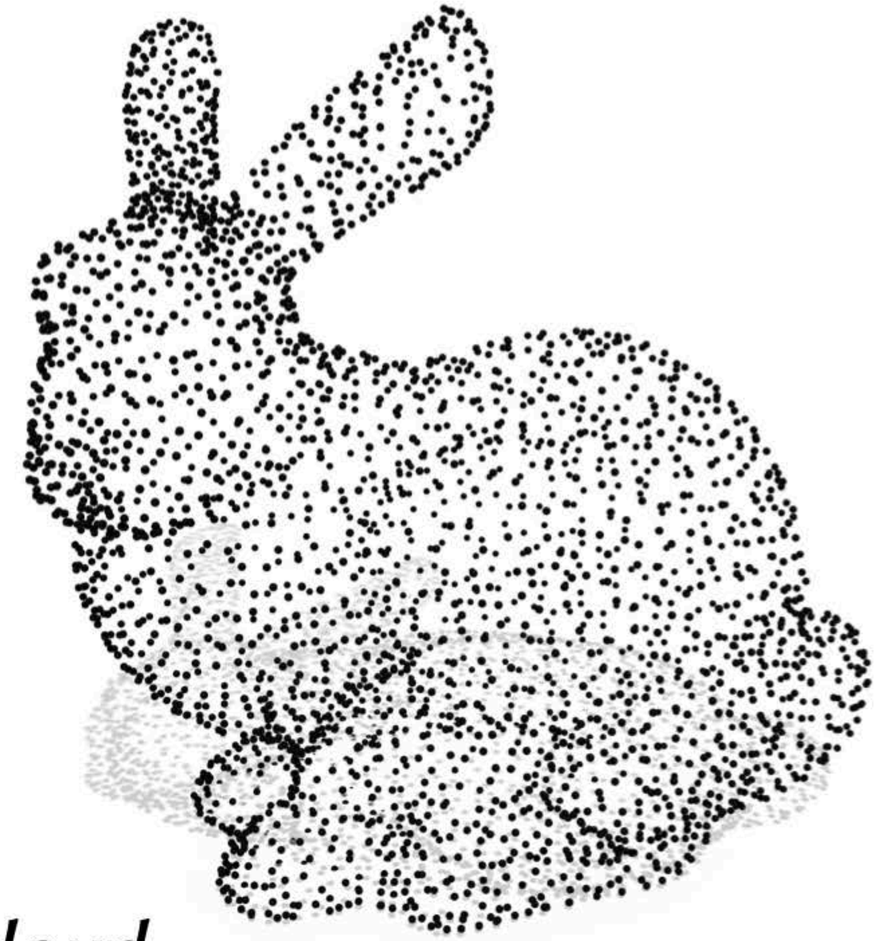
Carnegie Mellon University





The problem

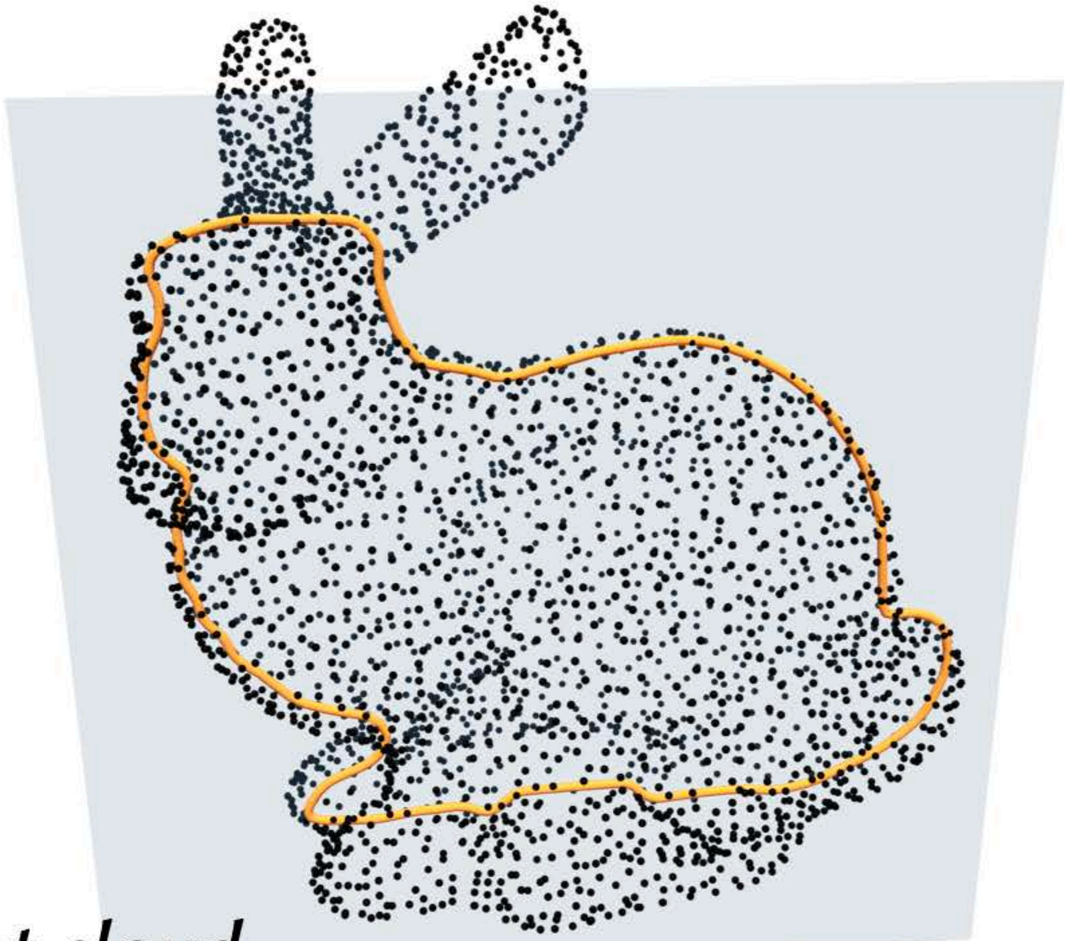
The problem



point cloud

scanning, vision, modeling...

The problem

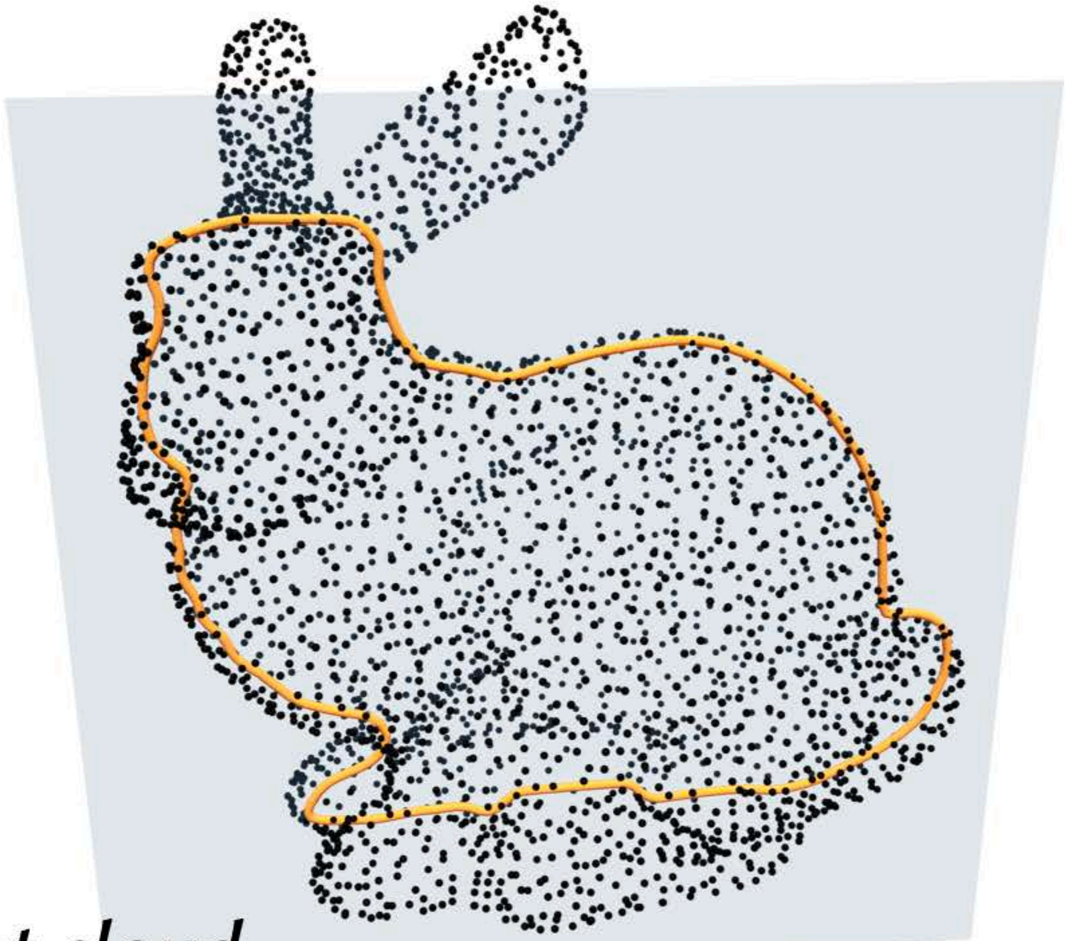


point cloud

scanning, vision, modeling...

signed distance function (SDF)

The problem

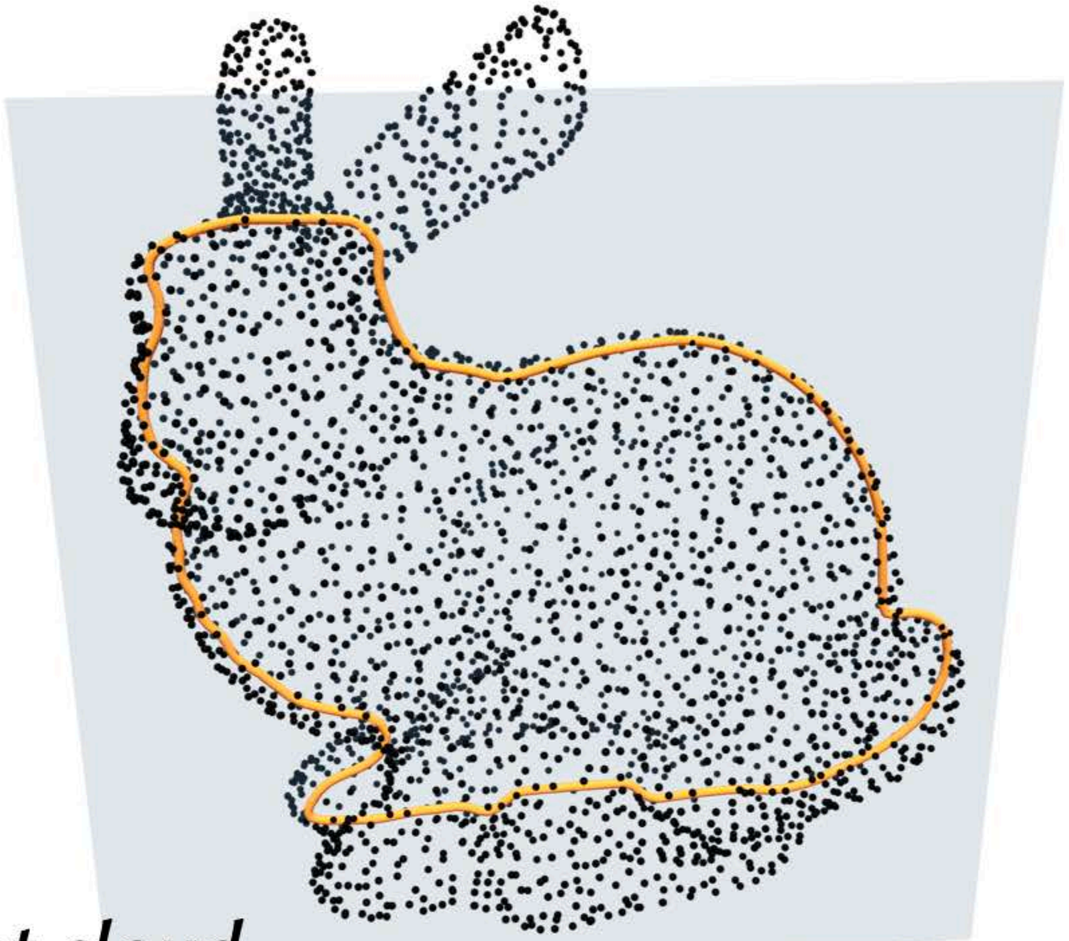


point cloud

scanning, vision, modeling...

signed distance function (SDF)

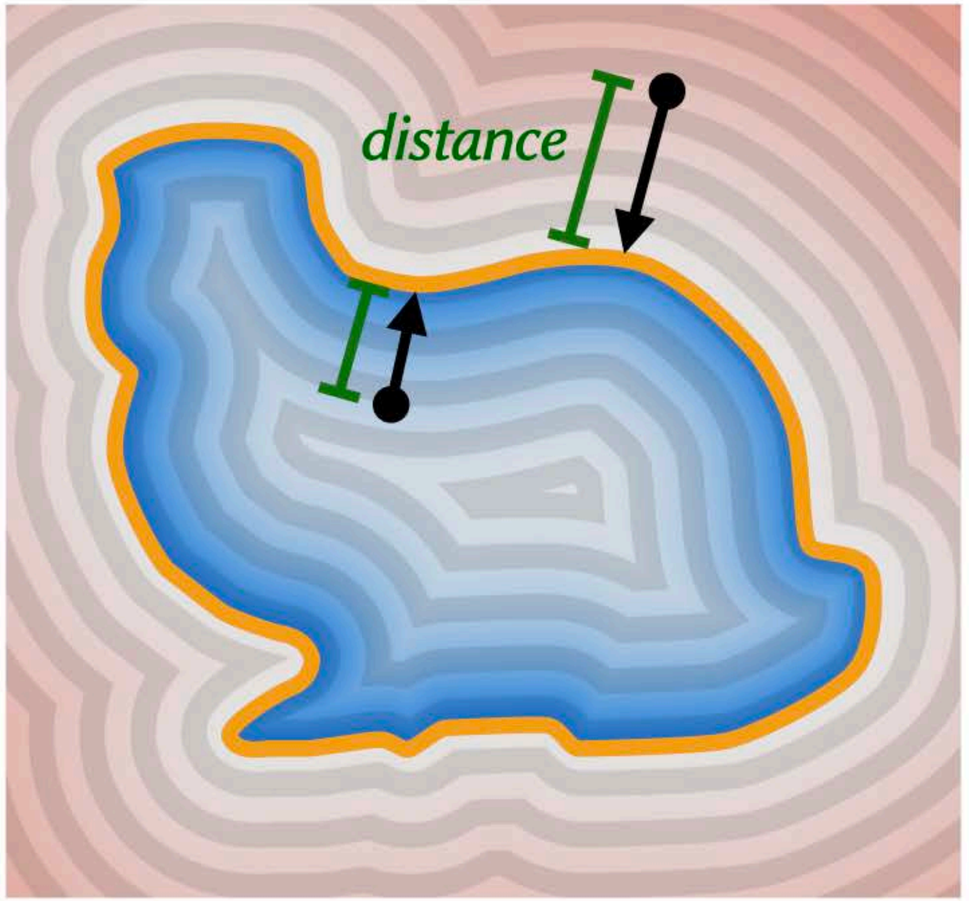
The problem



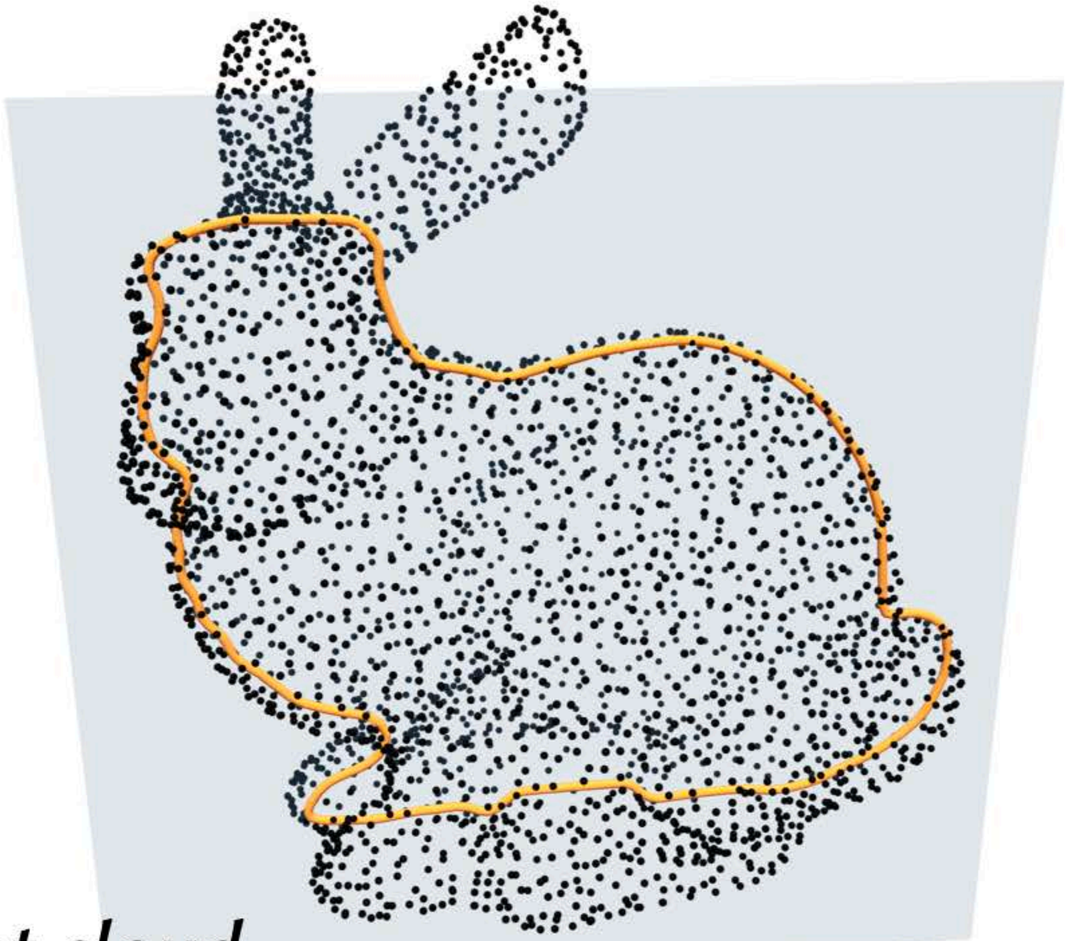
point cloud

scanning, vision, modeling...

signed distance function (SDF)



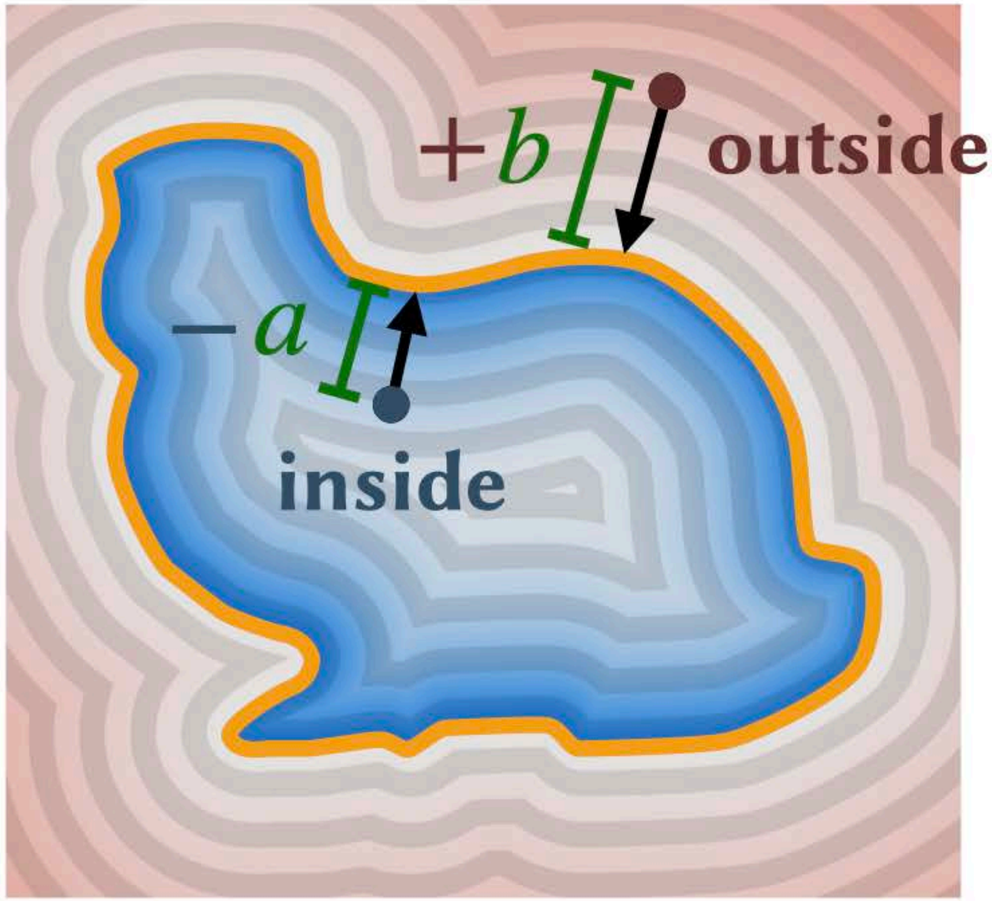
The problem



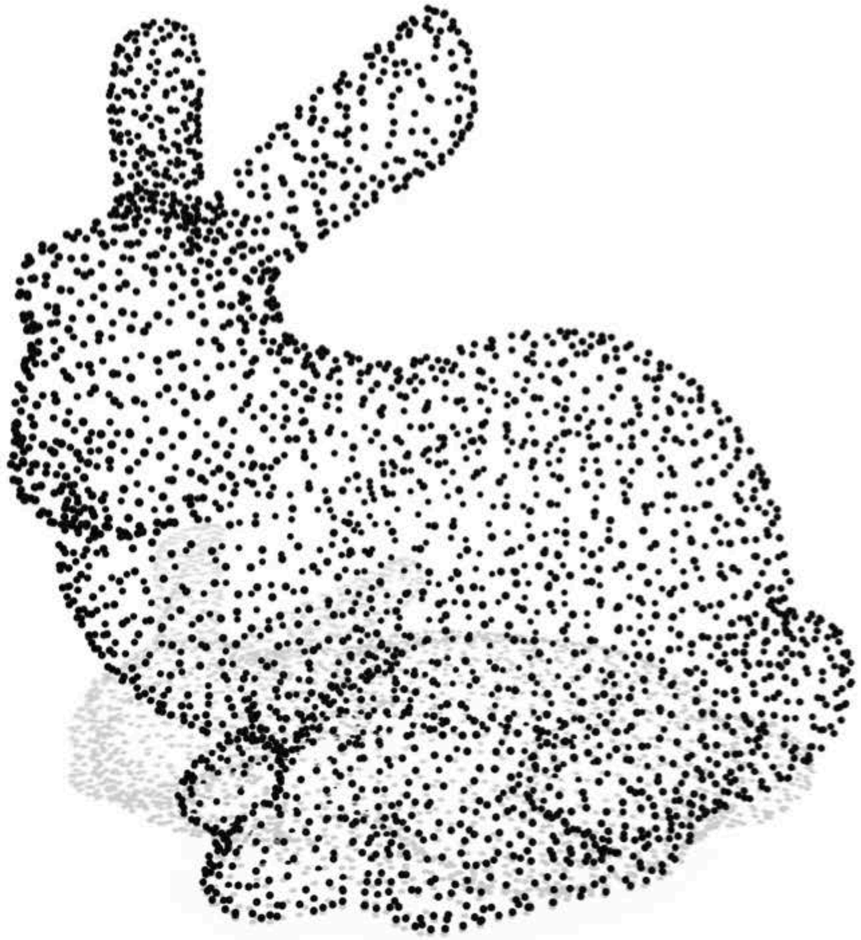
point cloud

scanning, vision, modeling…

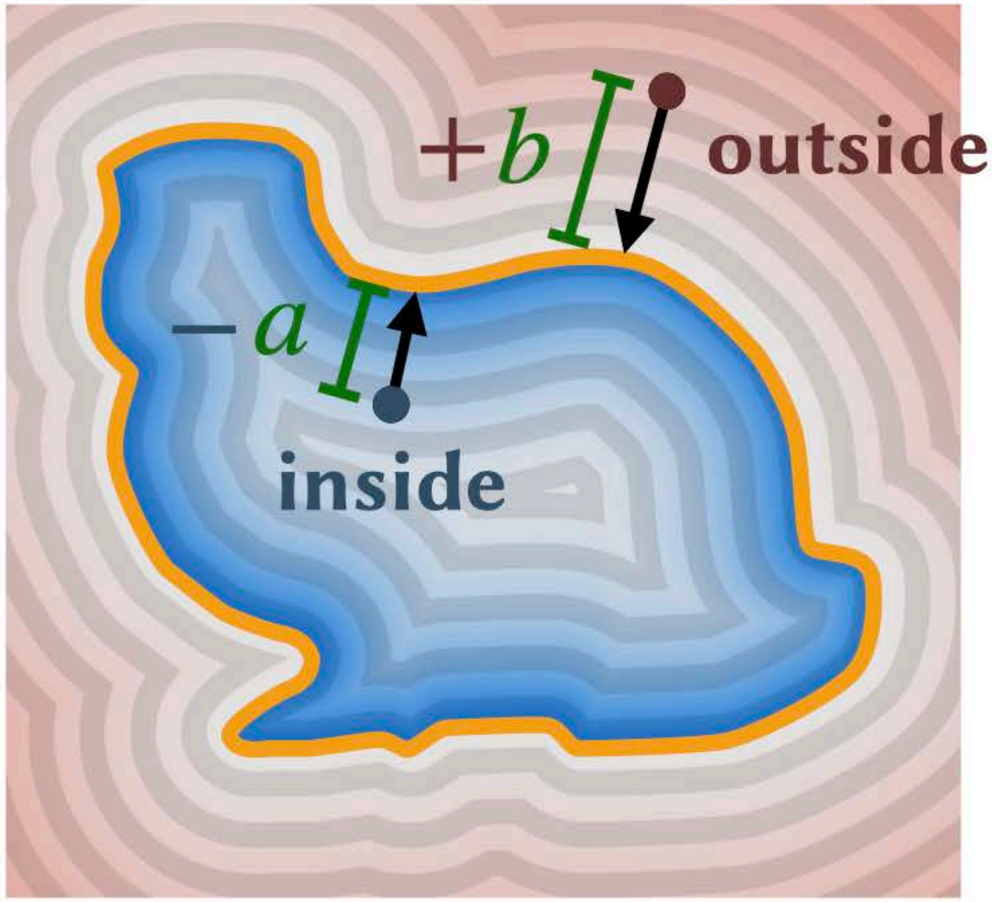
signed distance function (SDF)



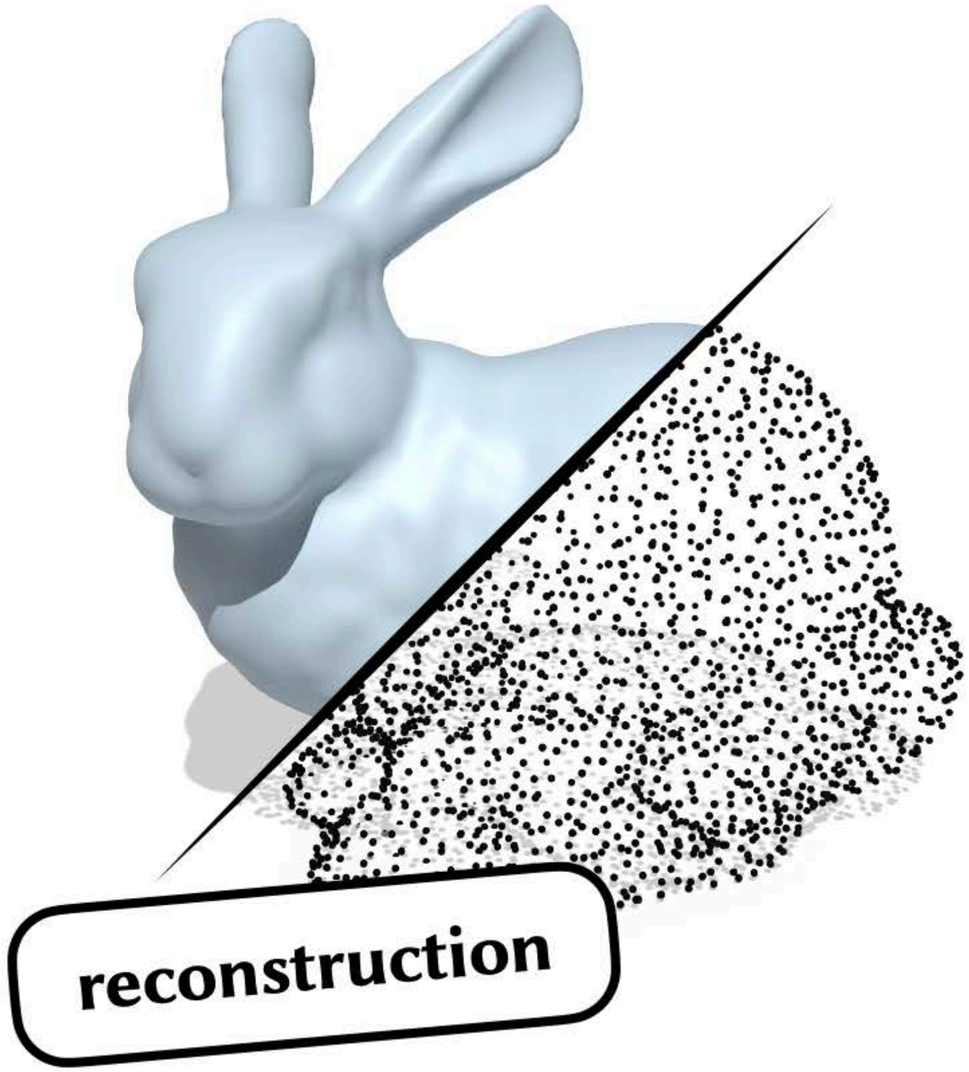
Signed distance requires both reconstruction and distance



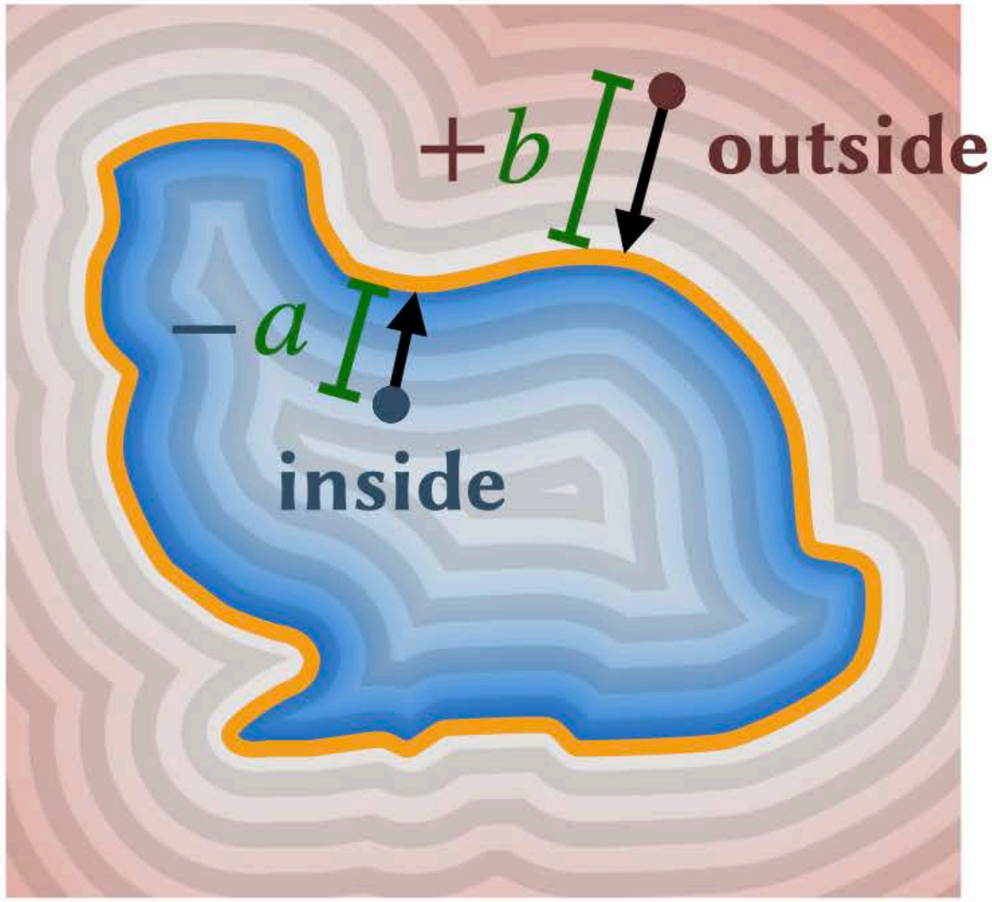
signed distance function (SDF)



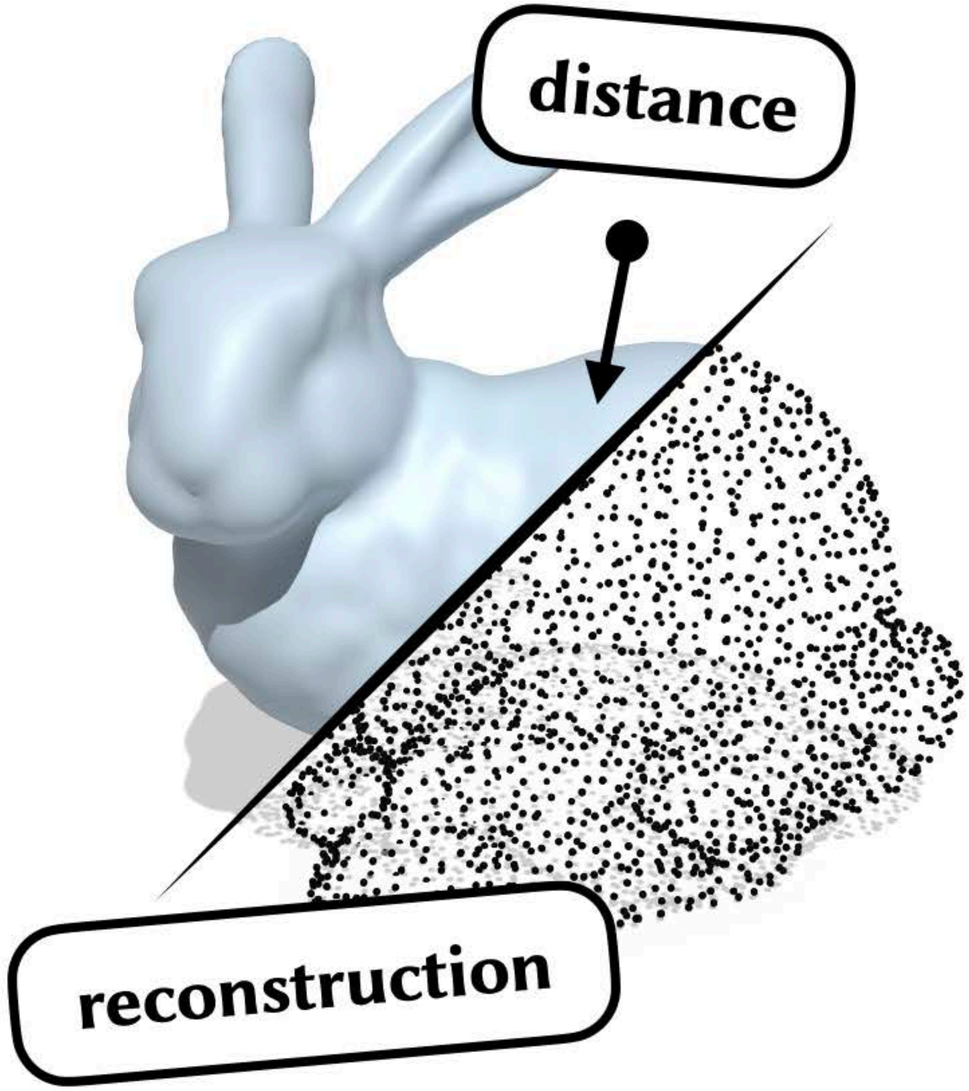
Signed distance requires both reconstruction and distance



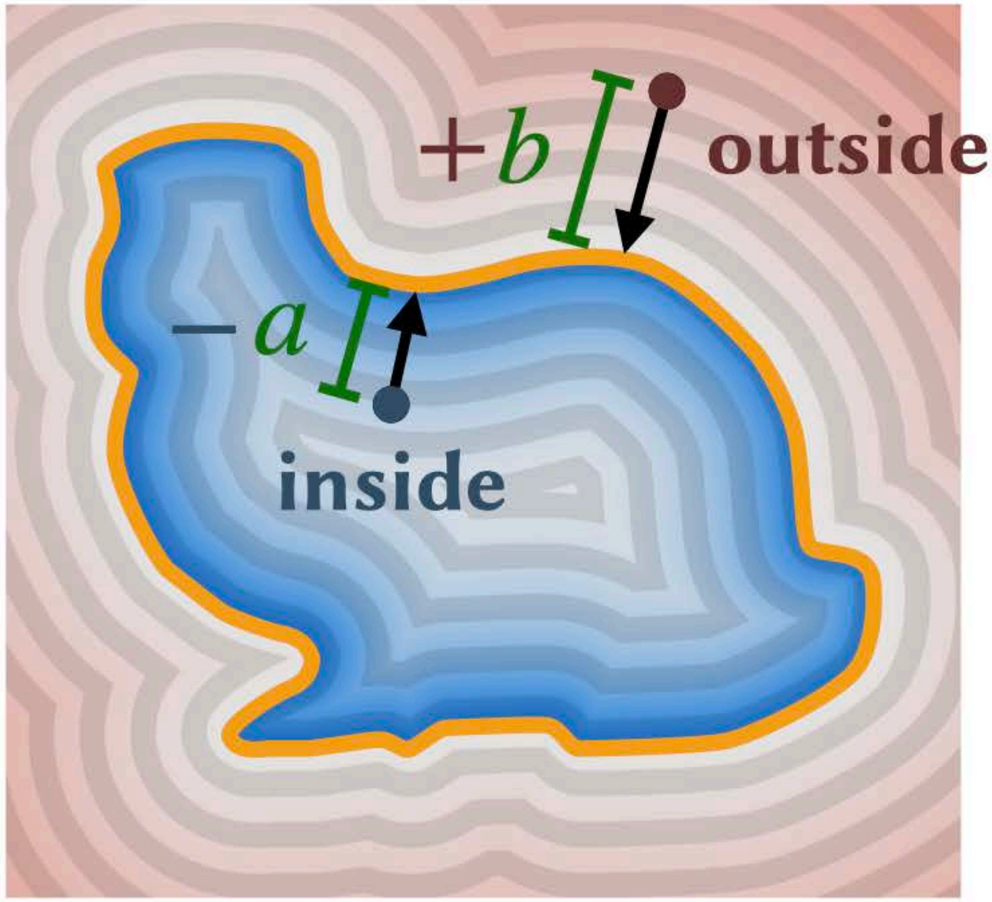
signed distance function (SDF)



Signed distance requires both reconstruction and distance



signed distance function (SDF)



Generalized distance algorithms are expensive

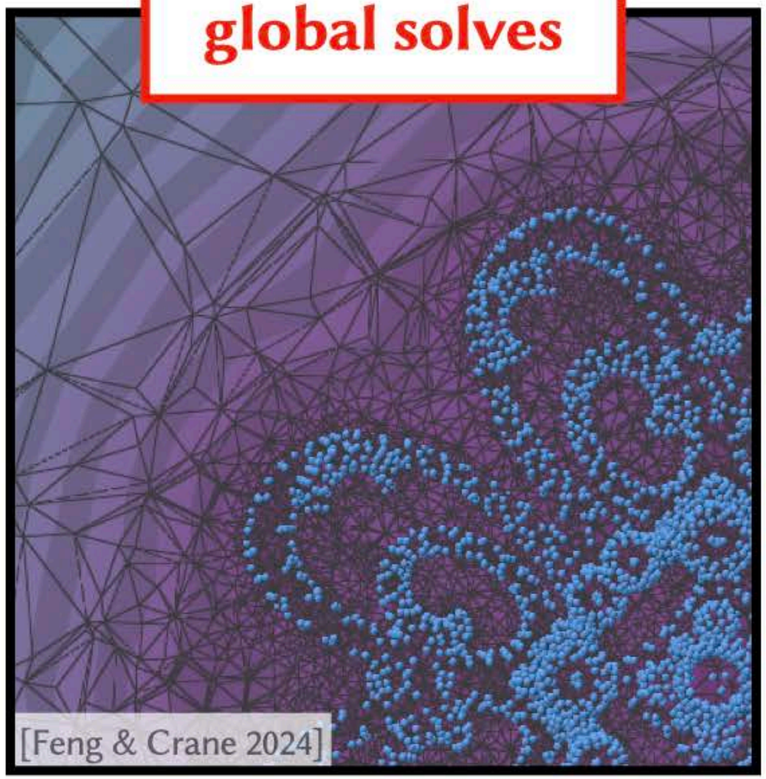
Generalized distance algorithms are expensive

SSD: Smoothed Signed Distance Surface Reconstruction
[Calakli & Taubin 2011]

A Heat Method for Generalized Signed Distance
[Feng & Crane 2024]

etc.…

global solves



[Feng & Crane 2024]

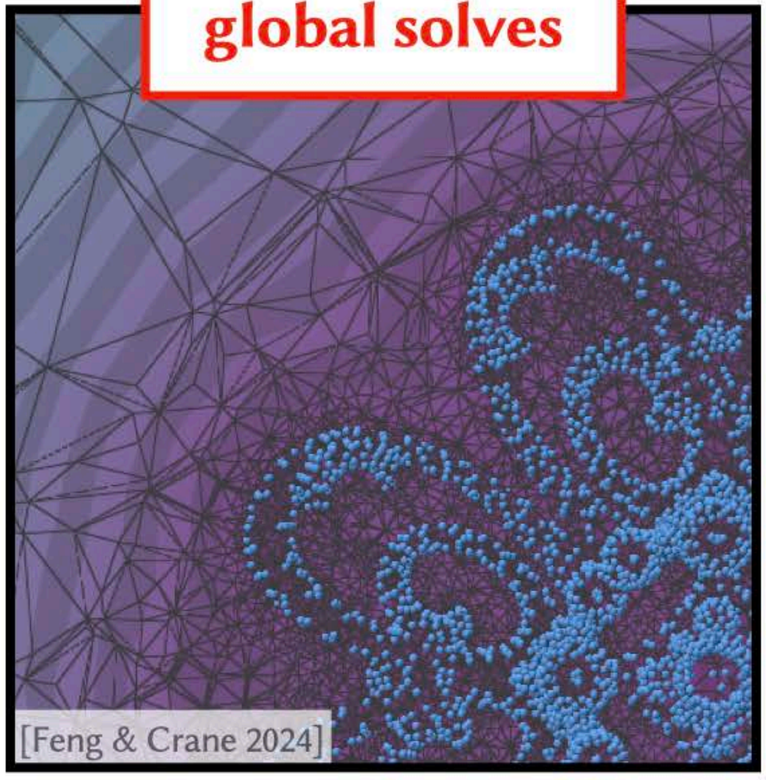
Generalized distance algorithms are expensive

SSD: Smoothed Signed Distance Surface Reconstruction
[Calakli & Taubin 2011]

A Heat Method for Generalized Signed Distance
[Feng & Crane 2024]

etc.…

global solves

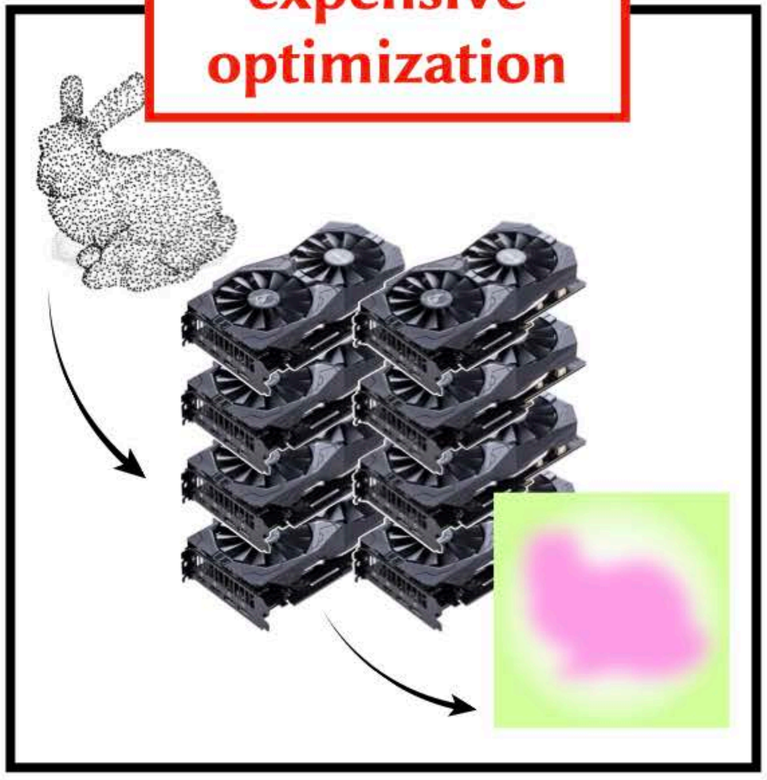


Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

etc.…

expensive optimization



Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

SDFs from Unoriented Point Clouds using Neural Variational Heat Distances
[Weidemaier et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

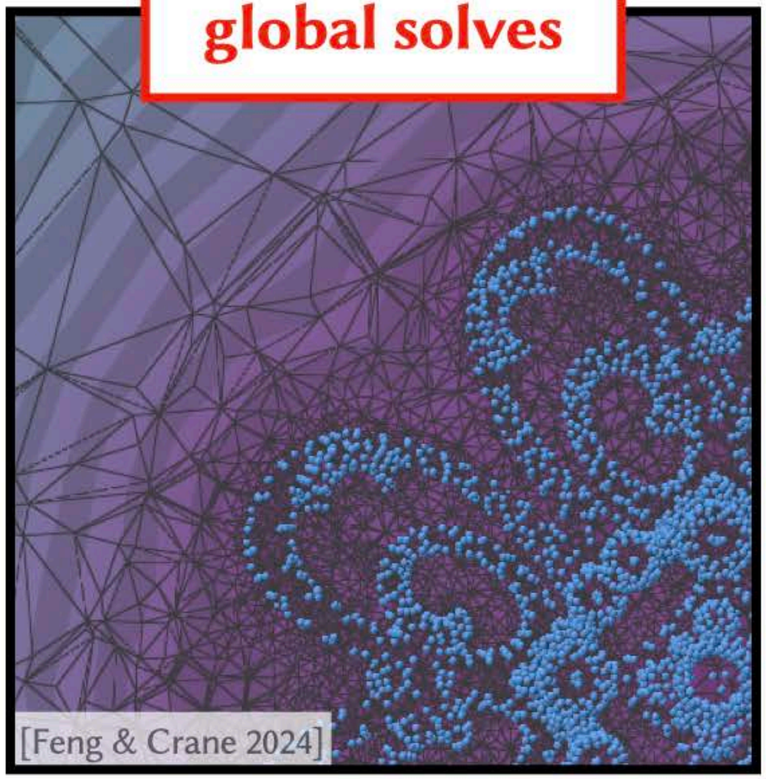
Generalized distance algorithms are expensive

SSD: Smoothed Signed Distance Surface Reconstruction
[Calakli & Taubin 2011]

A Heat Method for Generalized Signed Distance
[Feng & Crane 2024]

etc.…

global solves

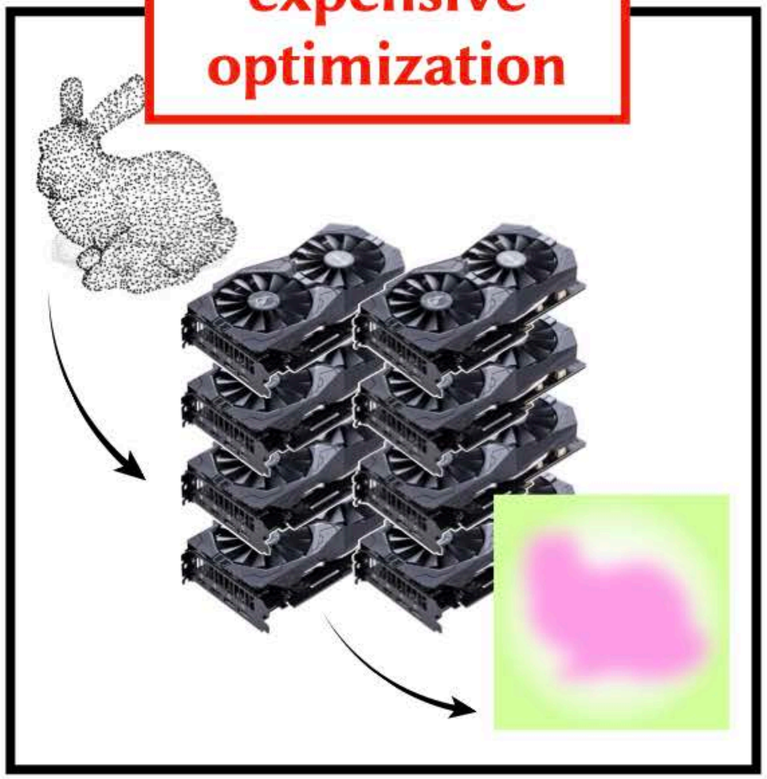


Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

etc.…

expensive optimization

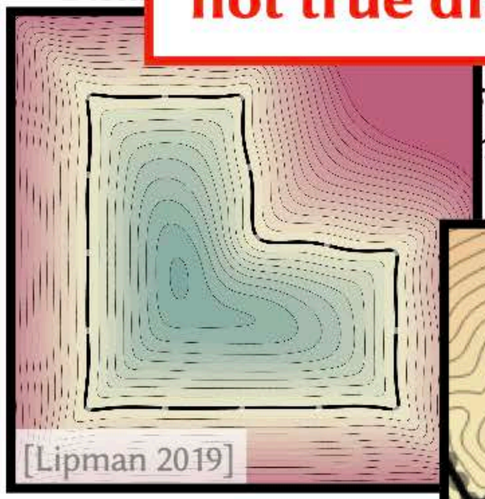


Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

StEik… Neural Signed Dista… presentation

not true distance



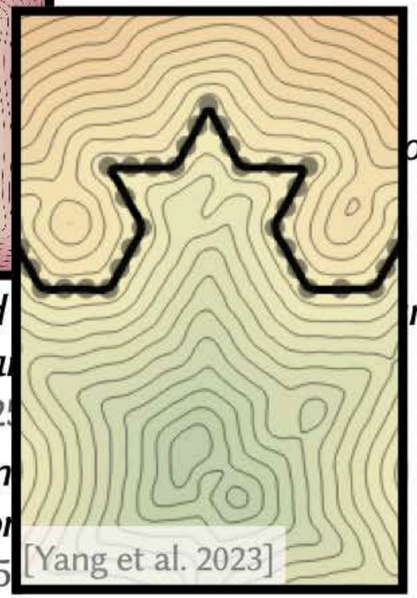
…truction in …t Clouds

…on

SDFs from Unoriented … Variational Heat Dista…
[Weidemaier et al. 202…]

…ral

efunc: An Efficient Fun… without Neural Networ…
[Zhang & Wonka 2025]



Fast reconstruction algorithms don’t compute distance

Fast reconstruction algorithms don't compute distance

meta-balls

Generalization of Algebraic Surface Drawing
[Blinn 1982]

Convolution Surfaces
[Bloomenthal & Shoemake 1991]

Alternate Distance Metrics for Implicit Surface Modeling
[Tigges et al. 1999]

Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions
[Morse et al. 2005]

etc. …

winding numbers

Fast Winding Numbers for Soups and Clouds
[Barill et al. 2018]

3D Reconstruction with Fast Dipole Sums
[Chen et al. 2024]

moving least squares / partition of unity

Surfaces generated by moving least squares methods
[Lancaster & Salkauskas 1981]

The Approximation Power of Moving Least-Squares
[Levin 1998]

Variational Implicit Surfaces
[Turk & O'Brien 1999]

Reconstruction and representation of 3D objects with radial basis functions
[Carr et al. 2001]

Defining point-set surfaces
[Amenta & Kil 2004]

Interpolating and Approximating Implicit Surfaces from Polygon Soup
[Shen et al. 2004]

An adaptive MLS surface for reconstruction with guarantees
[Dey & Sun 2005]

Multi-level partition of unity implicits
[Ohtake et al. 2005]

Anisotropic point set surfaces
[Adamson & Alexa 2006]

Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression
[Öztireli et al. 2009, etc.]

A Curvature-Adaptive Implicit Surface Reconstruction for Irregularly Spaced Points
[Zagorchev & Goshtasby 2012]

Floating Scale Surface Reconstruction
[Fuhrmann & Goesele 2014]

etc. …

convolutional "distance"

Provably good moving least squares
[Kolluri 2008]

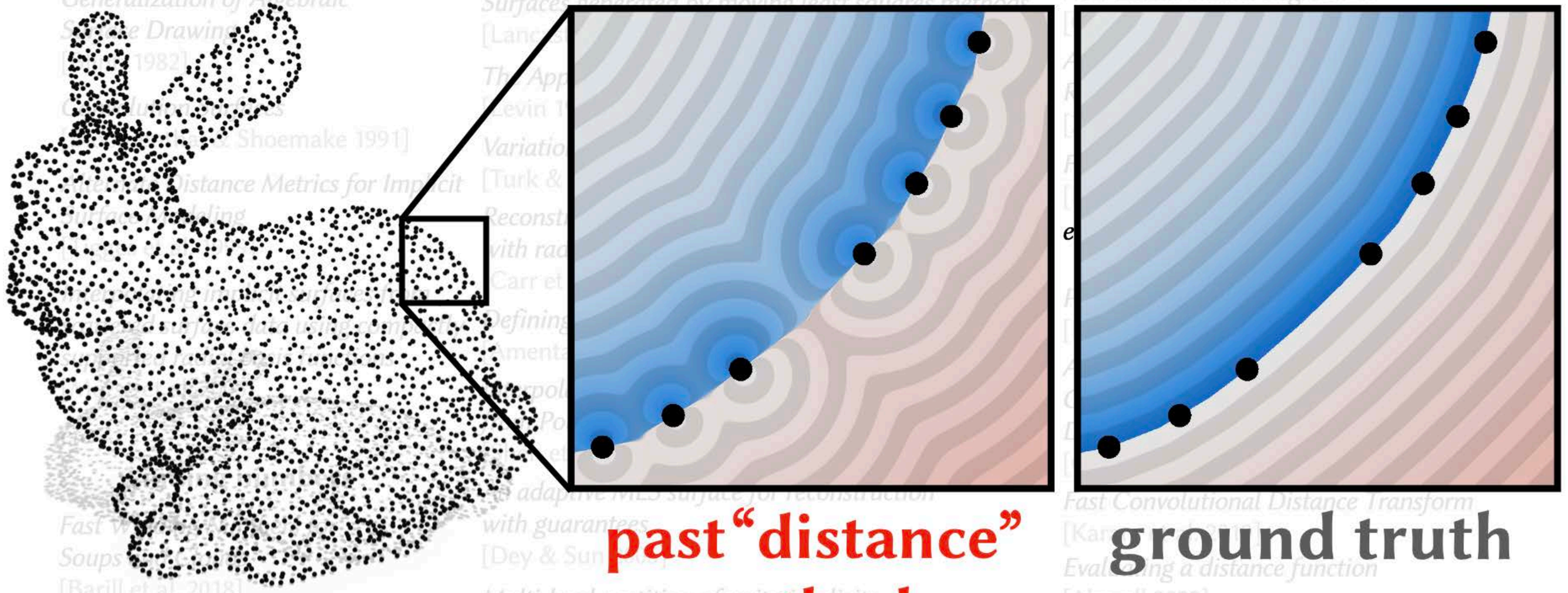
A Schrödinger Equation for the Fast Computation of Approximate Euclidean Distance Functions
[Gurumoorthy & Rangarajan 2009]

Fast Convolutional Distance Transform
[Karam et al. 2019]

Evaluating a distance function
[Abgrall 2022]

Fast evaluation of smooth distance constraints on co-dimensional geometry
[Madan & Levin 2022]

Fast reconstruction algorithms don't compute distance



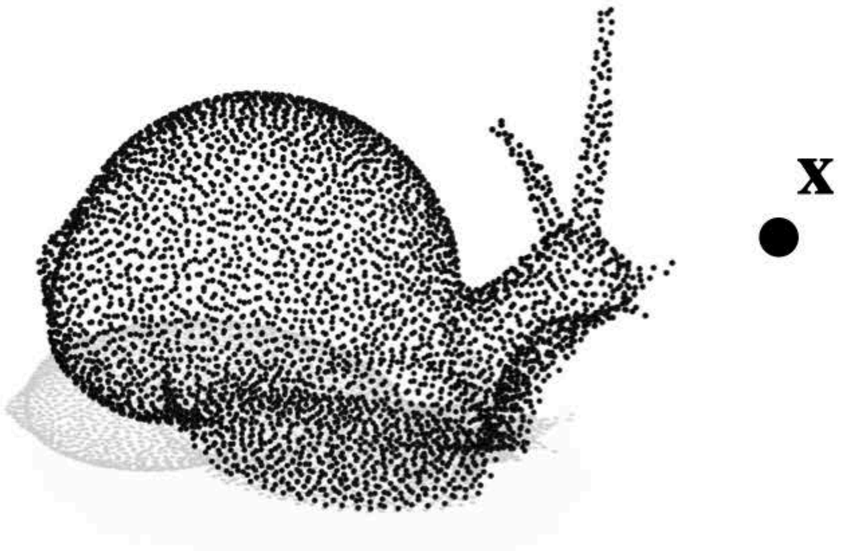
past "distance" methods

ground truth

Points as Tori (PAT)

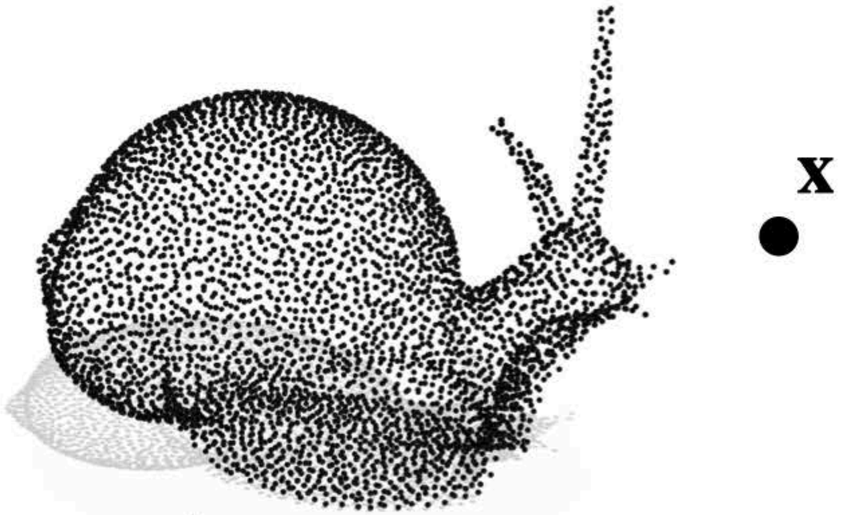
Points as Tori (PAT)

Input: point cloud with normals, and a query point $\mathbf{x} \in \mathbb{R}^3$

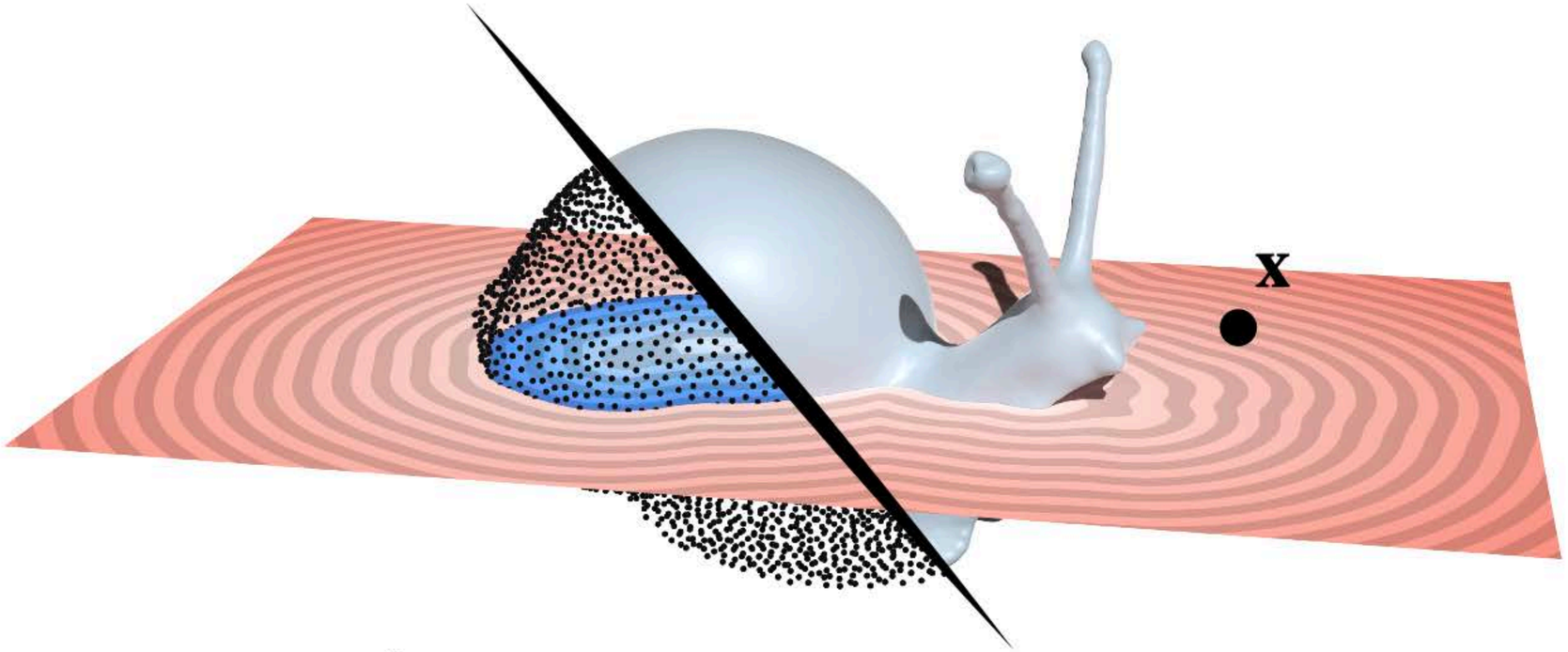


Points as Tori (PAT)

Input: point cloud with normals, and a query point $\mathbf{x} \in \mathbb{R}^3$



Output: signed distance at $\mathbf{x}$ to an approximate underlying surface



Points as Tori (PAT)

Points as Tori (PAT)

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

kernel density estimator

Points as Tori (PAT)

weights

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

kernel density estimator

Points as Tori (PAT)

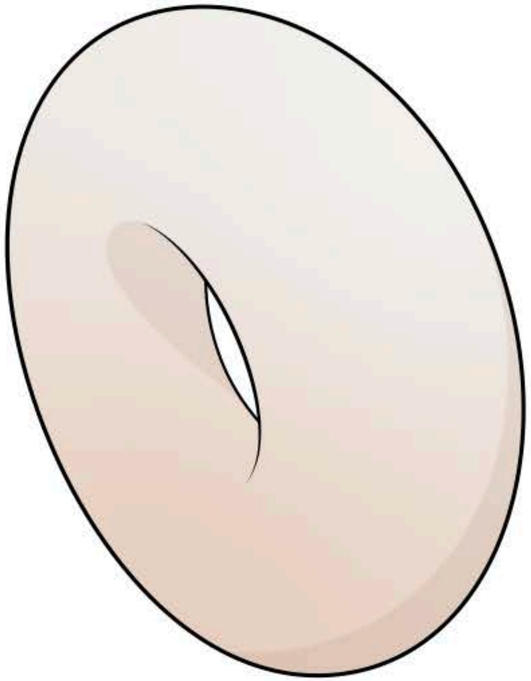
i-th function *weights*

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

kernel density estimator

Points as Tori (PAT)

torus

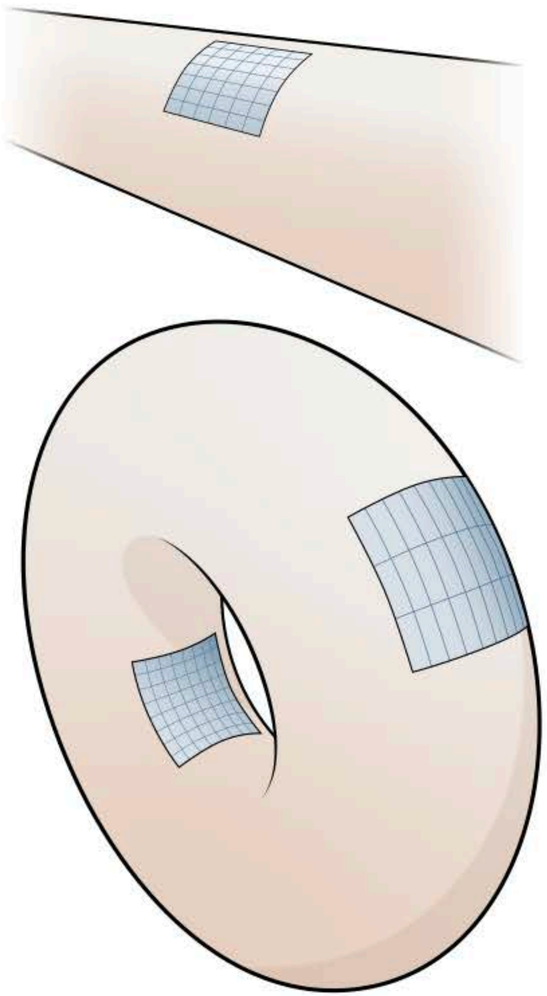


i-th function *weights*

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

kernel density estimator

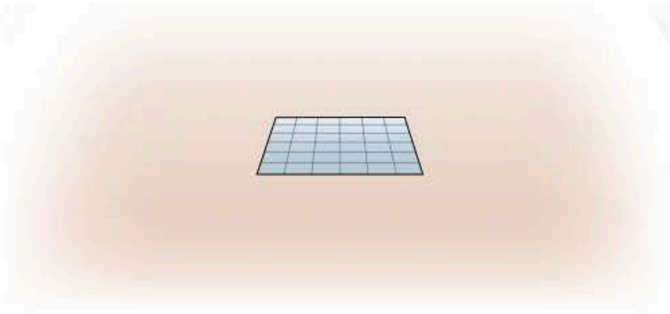
Points as Tori (PAT)



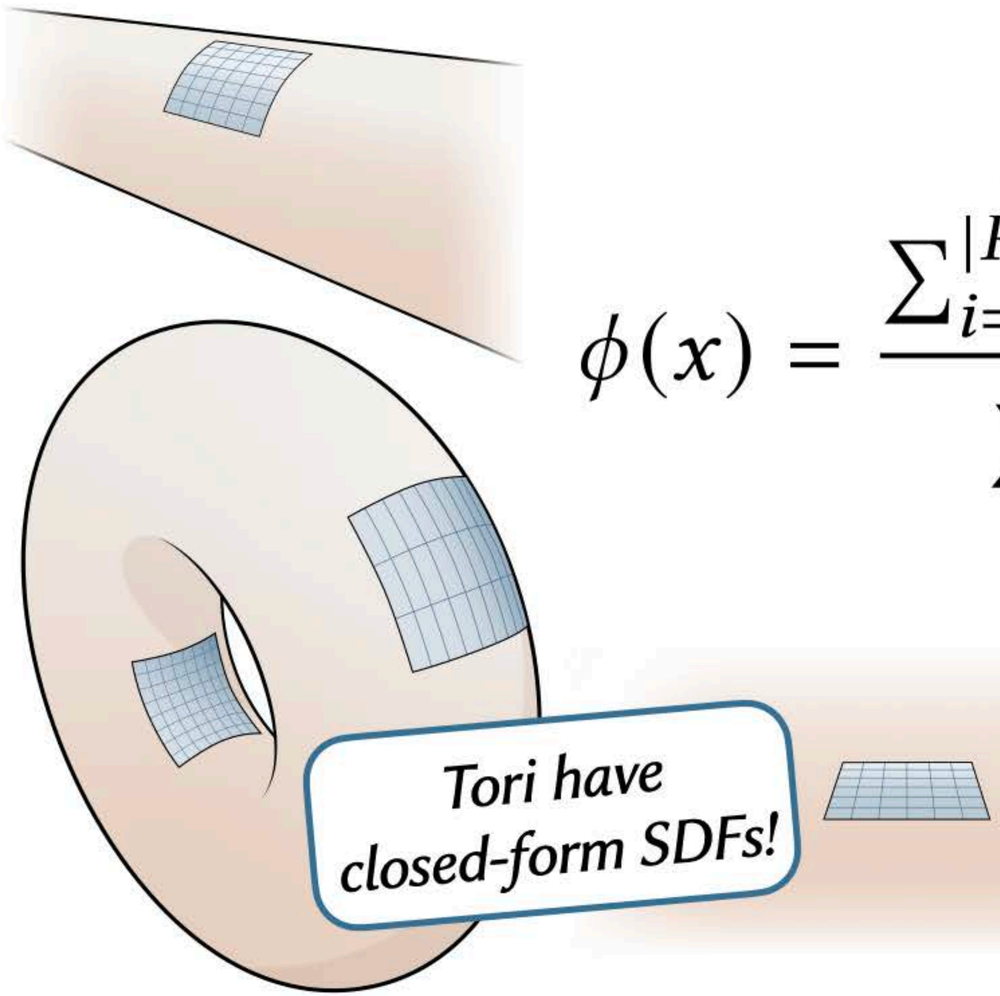
i-th function *weights*

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

kernel density estimator



Points as Tori (PAT)

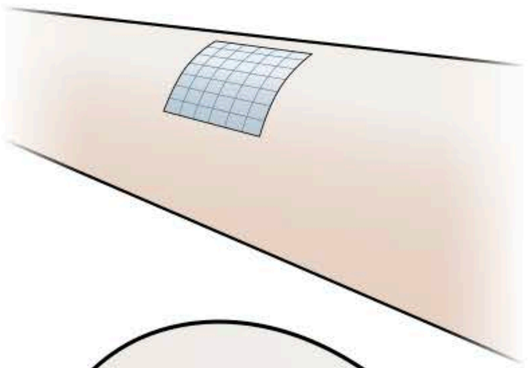


$$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}$$

i-th function *weights*

kernel density estimator

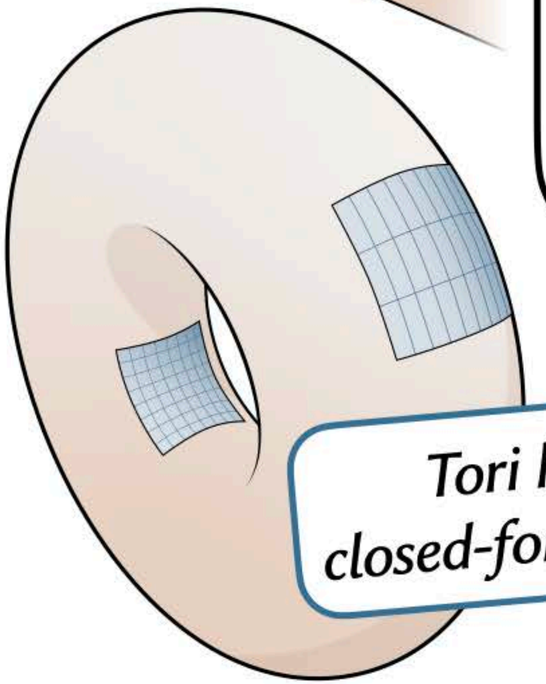
Points as Tori (PAT)



global SDF

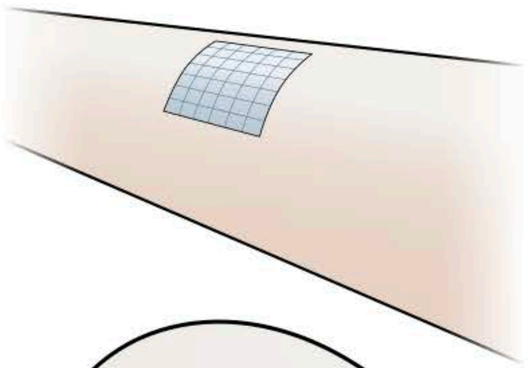
SDF of i-th torus

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



Tori have closed-form SDFs!

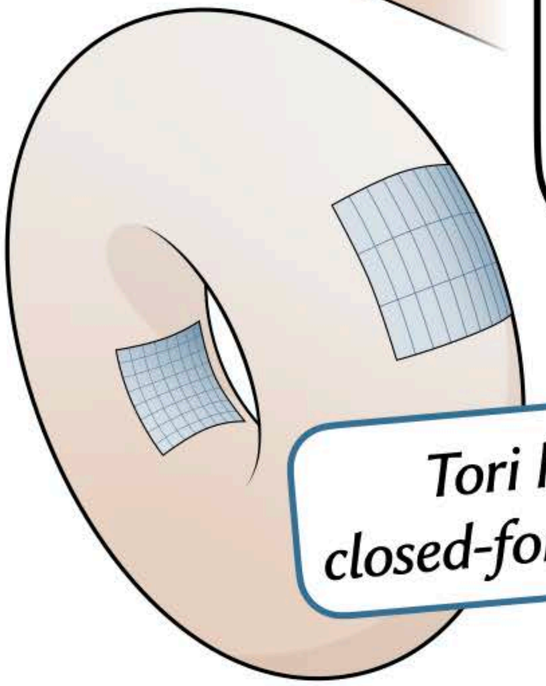
Points as Tori (PAT)



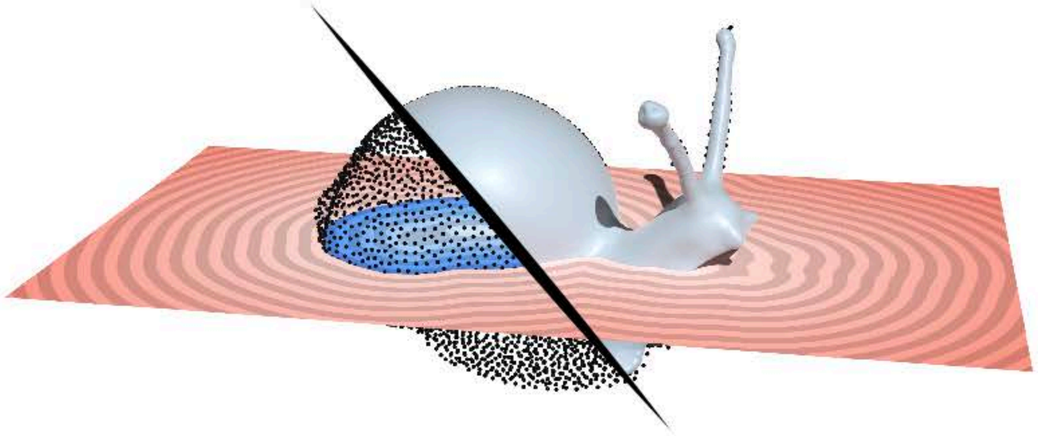
global SDF

SDF of i-th torus

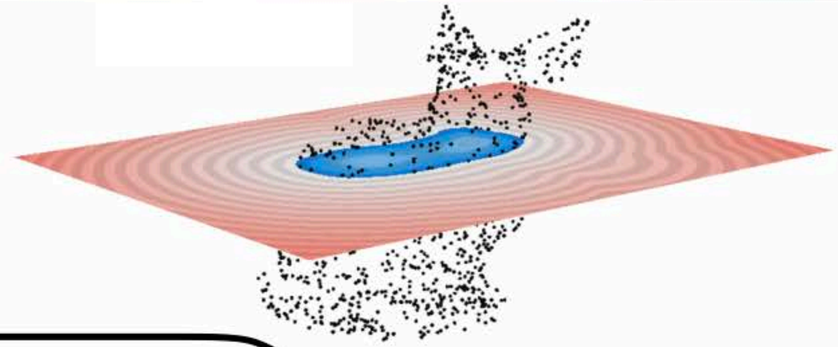
$$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}$$

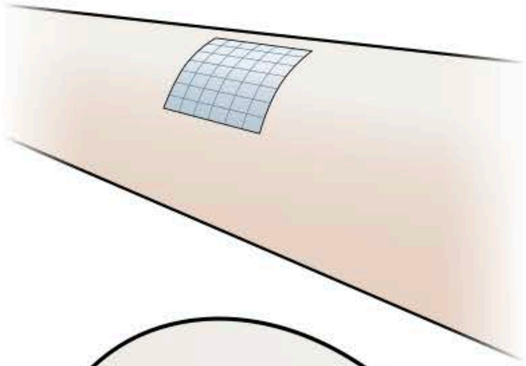


Tori have closed-form SDFs!



Points as Tori (PAT)

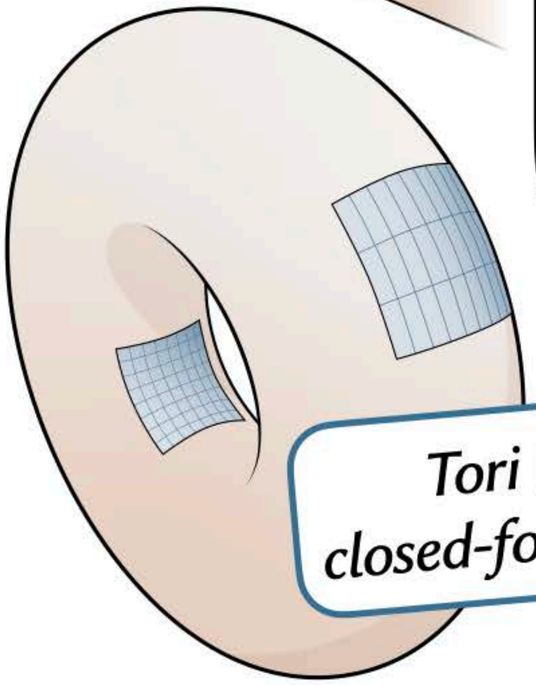




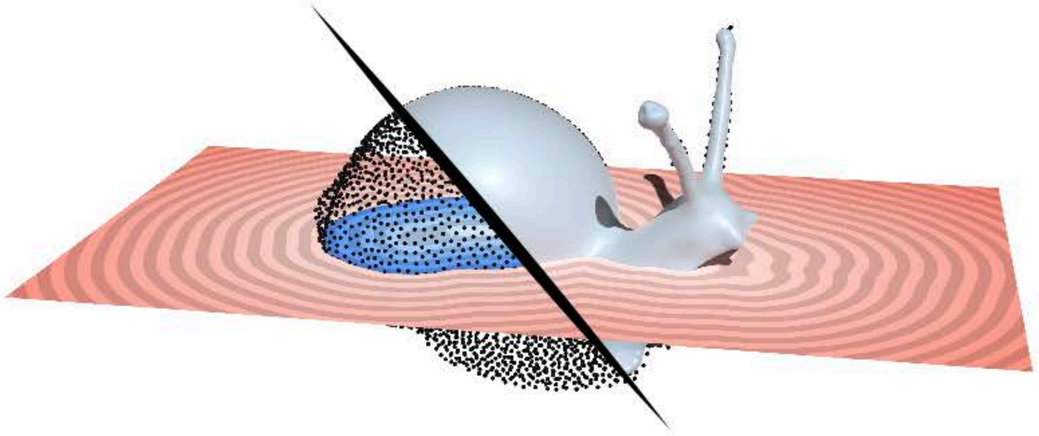
global SDF

SDF of i-th torus

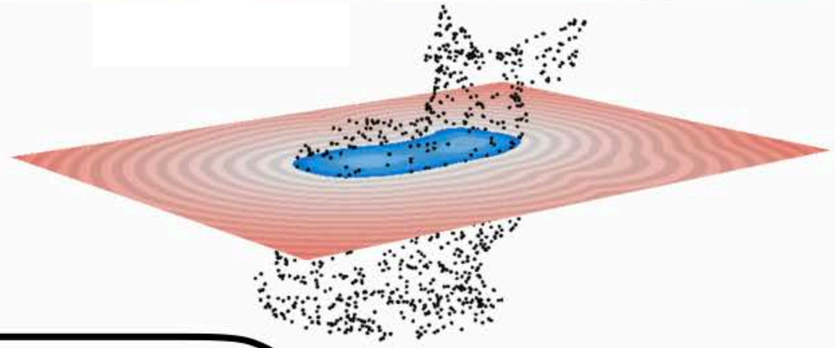
$$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}$$



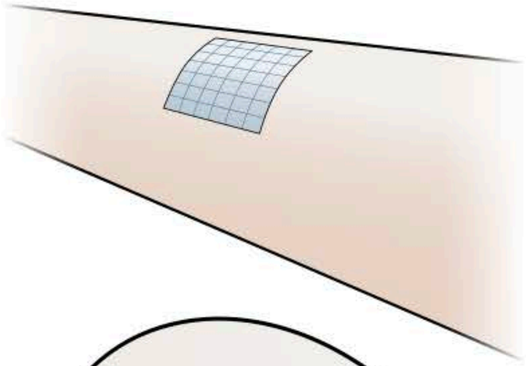
Tori have closed-form SDFs!



Points as Tori (PAT)



no global solves



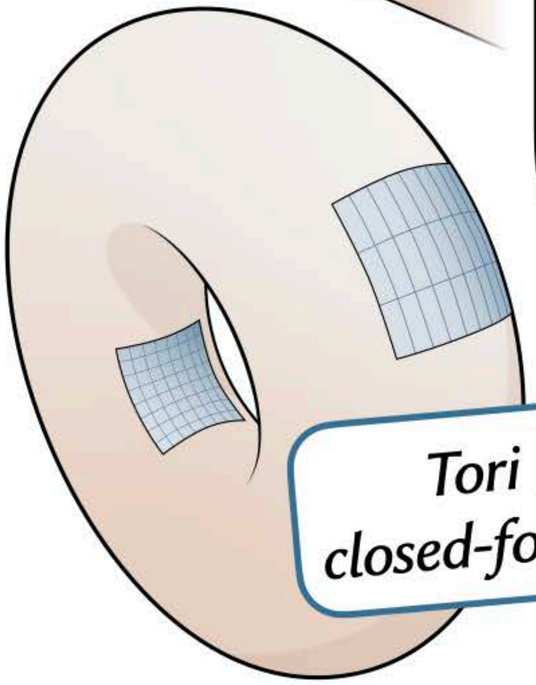
global SDF

SDF of i-th torus

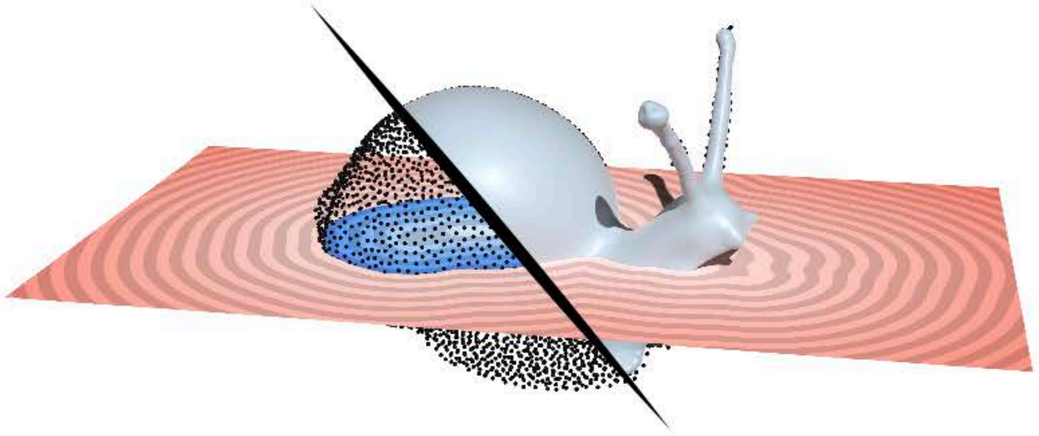
$$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}$$

no discretization

pointwise evaluation



Tori have closed-form SDFs!



Two *convolutional formulas* for distance

Two *convolutional formulas* for distance

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

Formula #2:

$$\widehat{d}(x) := \frac{\int_{\Omega} g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_{\Omega} \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

Two *convolutional formulas* for distance

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

Formula #2:

$$\widehat{d}(x) := \frac{\int_{\Omega} g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_{\Omega} \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

"convolutional distance formulas"

Two *convolutional formulas* for distance

Formula #1:

summation over the shape boundary Ω

$$\widetilde{d}(x) := -\frac{1}{\lambda}\log\left[\int_{\Omega} h(z)\exp\left(-\lambda\|x-z\|\right)\mathrm{d}z\right]$$

kernel

"convolutional distance formulas"

Formula #2:

kernel

$$\widehat{d}(x) := \frac{\int_{\Omega} g(z)\exp\left(-\lambda\|x-z\|\right)\mathrm{d}z}{\int_{\Omega}\exp\left(-\lambda\|x-z\|\right)\mathrm{d}z}$$

summation over Ω

Many special cases

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

Many special cases

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

Examples:

- *Varadhan's formula* [Varadhan 1967]
- *Hopf-Cole transformations* for distance [Belyaev & Fayolle 2015; Lipman 2021]
- *LogSumExp* distance [Madan & Levin 2022]
- *Kreisselmeier-Steinhauser* functions [Kreisselmeier & Steinhauser 1980]
- *Schrödinger distance transform* [Gurumoorthy & Rangarajan 2009]
- other instances [Karam et al. 2019, Abgrall 2022]
- discrete LogSumExp

Many special cases

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

exponential

Examples:

- *Varadhan's formula* [Varadhan 1967]
- *Hopf-Cole transformations* for distance [Belyaev & Fayolle 2015; Lipman 2021]
- *LogSumExp* distance [Madan & Levin 2022]
- *Kreisselmeier-Steinhauser* functions [Kreisselmeier & Steinhauser 1980]
- *Schrödinger distance transform* [Gurumoorthy & Rangarajan 2009]
- other instances [Karam et al. 2019, Abgrall 2022]
- discrete LogSumExp

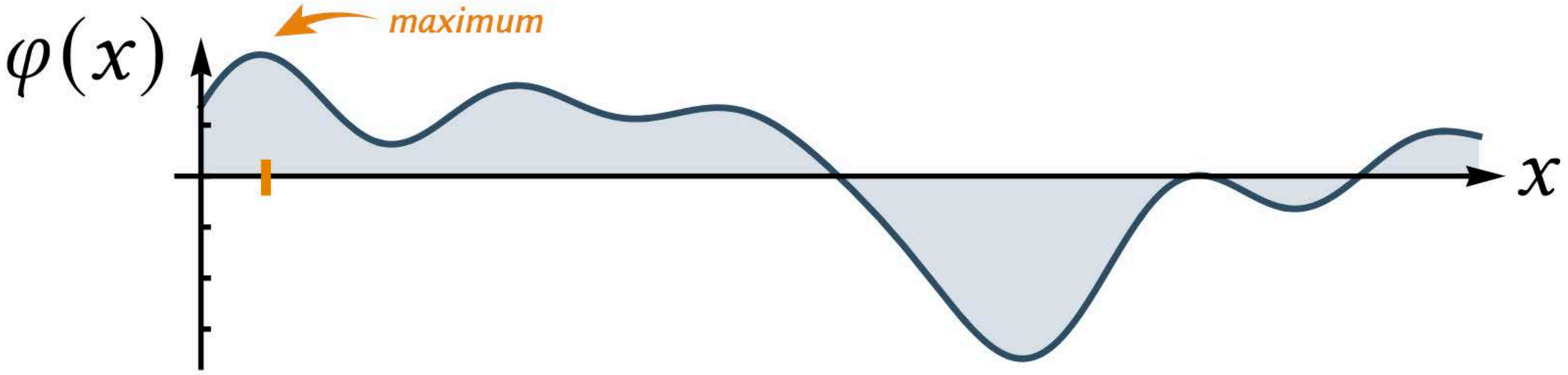
Consider the exponential of any function...

$$\varphi(x)$$

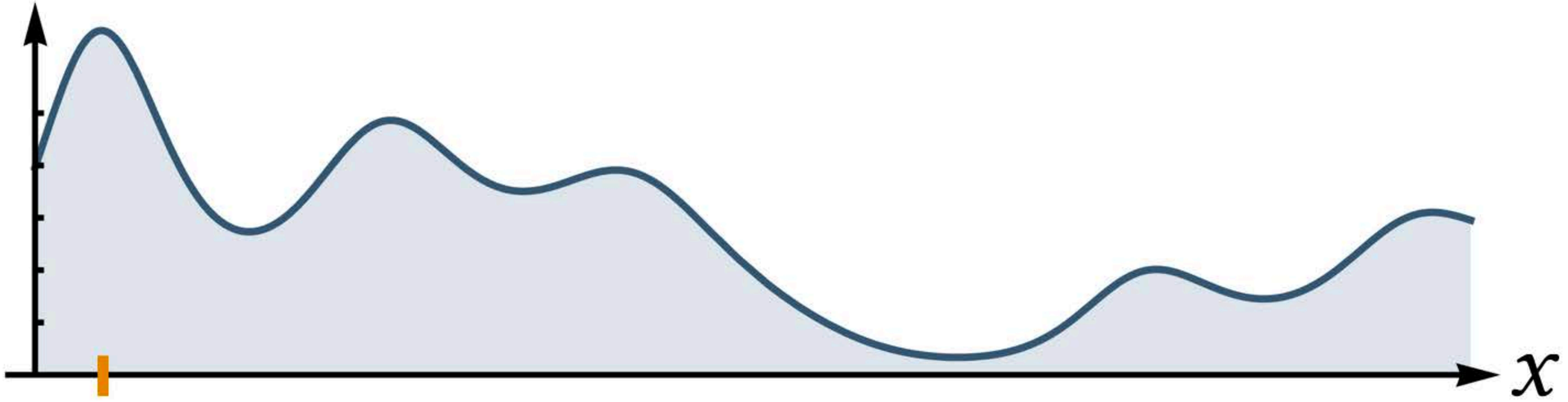
Consider the exponential of any function...

$$\exp\left(\varphi(x)\right)$$

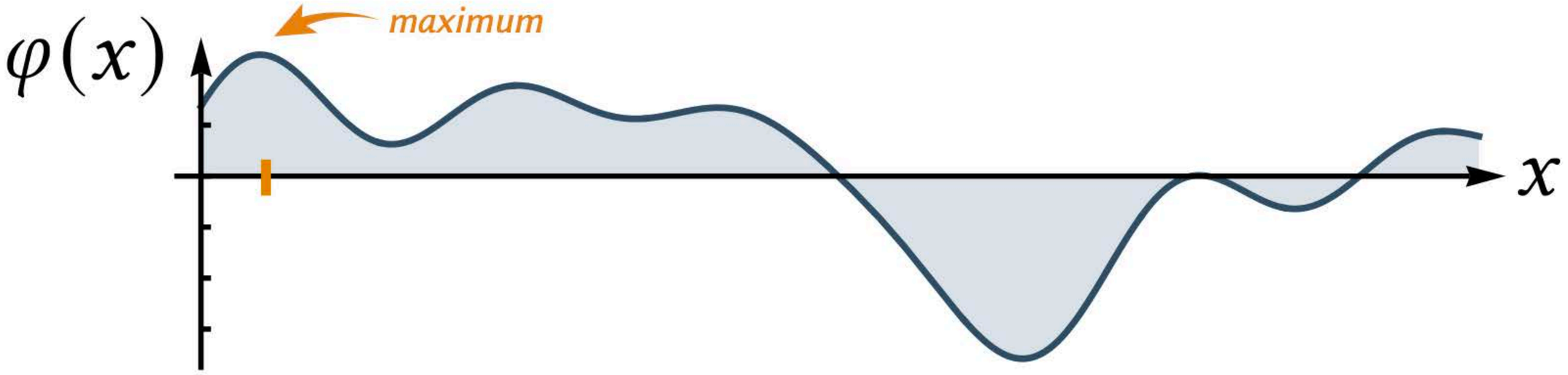
... the extremum value becomes exponentially more extreme



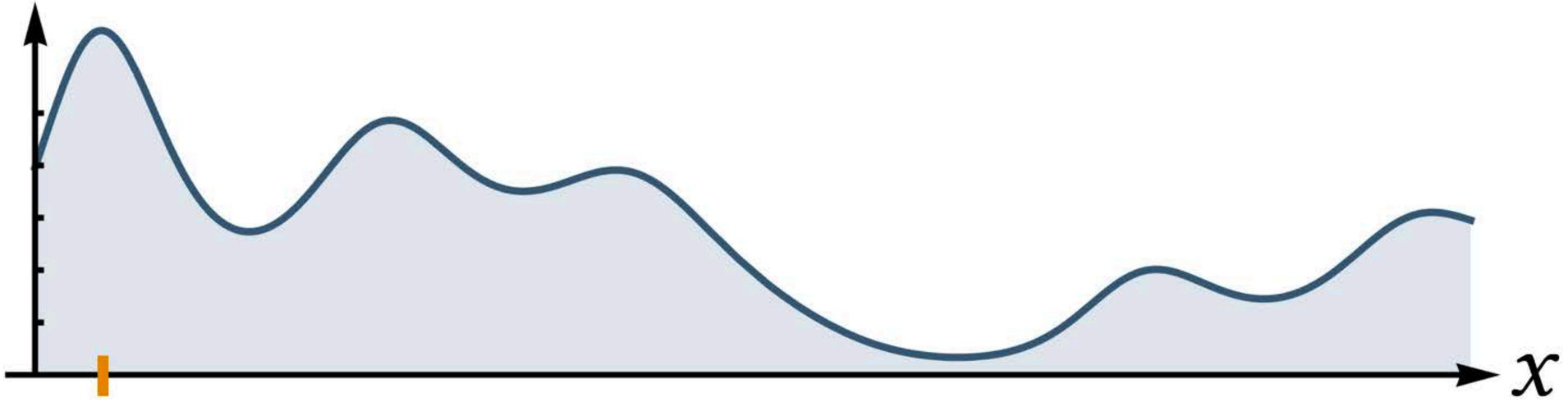
$\exp(\lambda\varphi(x))$ $\lambda = 1.00$



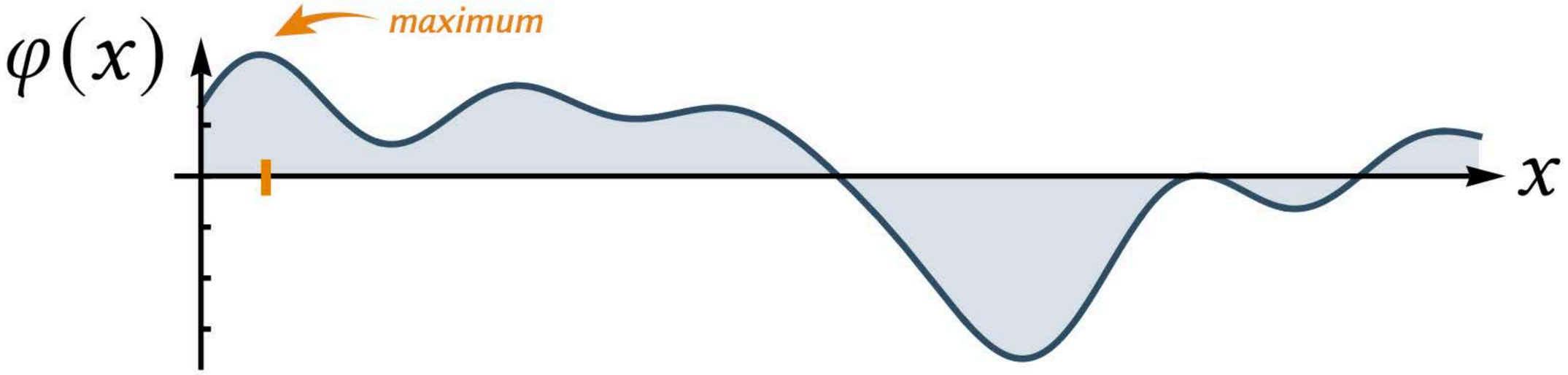
... the extremum value becomes exponentially more extreme



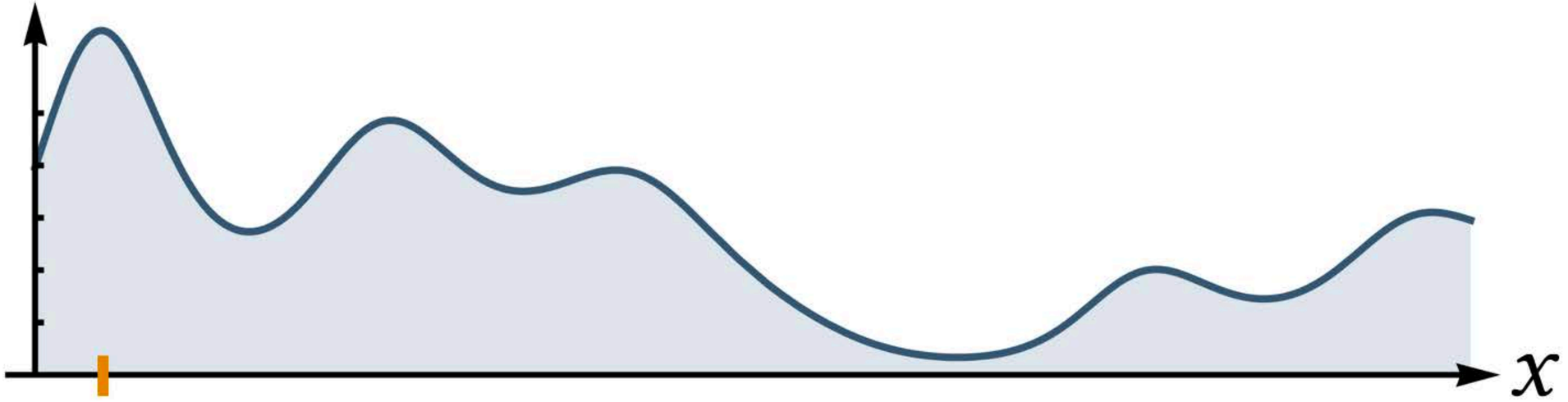
$\exp(\lambda\varphi(x))$ $\lambda = 1.00$



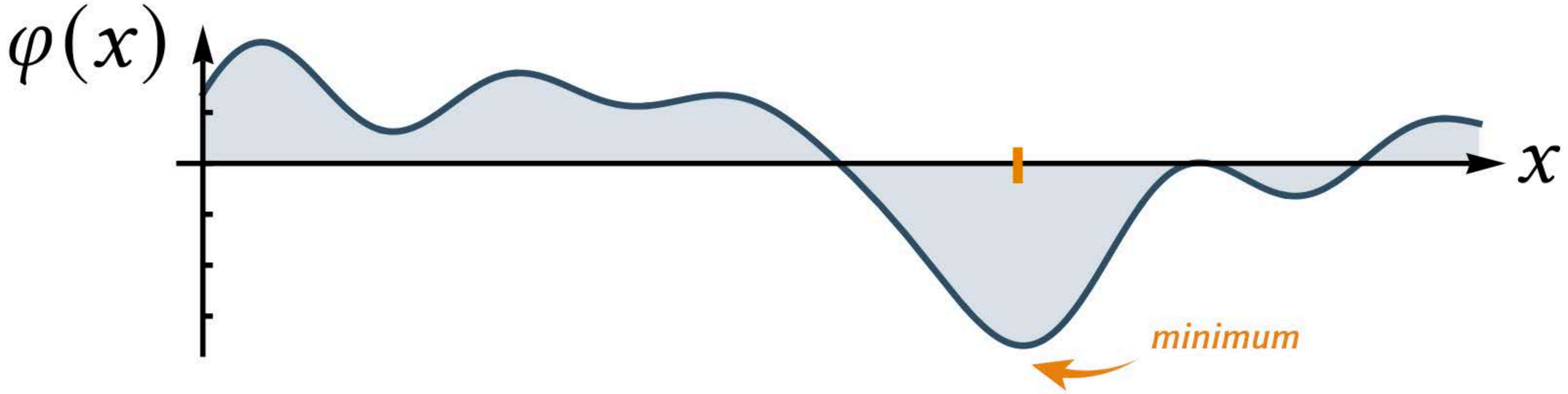
... the extremum value becomes exponentially more extreme



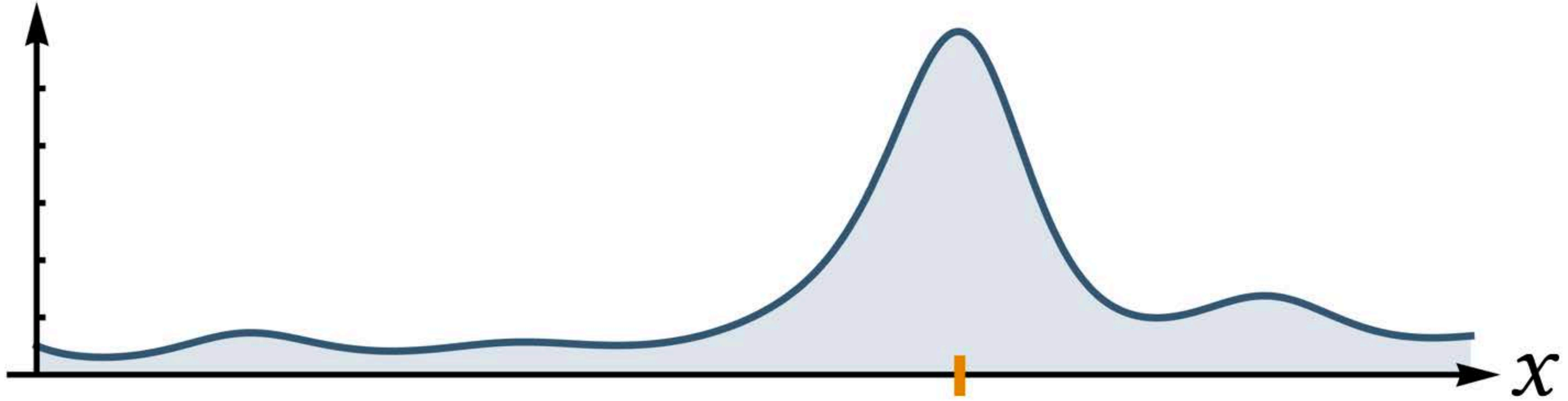
$\exp(\lambda\varphi(x))$ $\lambda = 1.00$



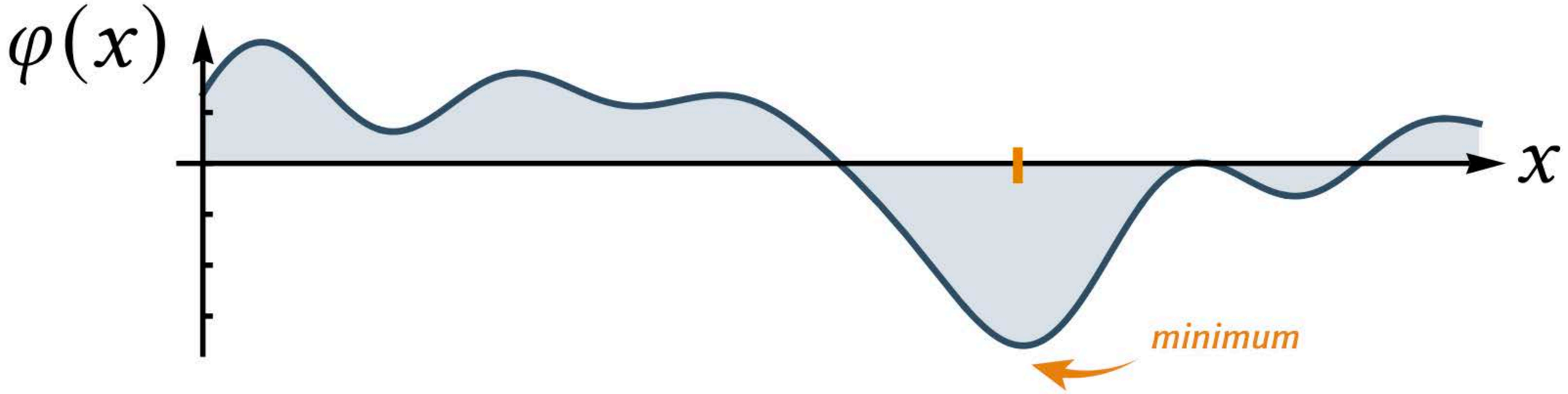
... the extremum value becomes exponentially more extreme



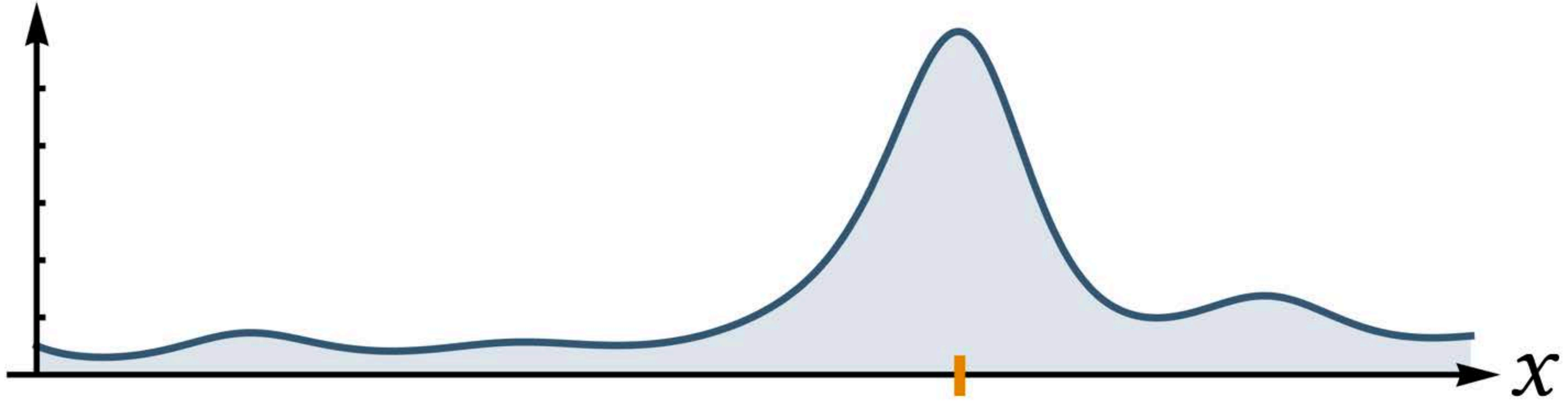
$\exp\left(-\lambda\,\varphi(x)\right)$ $\lambda = 1.00$



... the extremum value becomes exponentially more extreme



$\exp\left(-\lambda\,\varphi(x)\right)$ $\lambda = 1.00$



Asymptotic analysis

$$\int_{\Omega} h(z) \exp\left(-\lambda \varphi(z)\right) \mathrm{d}z$$

Asymptotic analysis

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

Laplace's method says... *... only the contribution at* $x^* = \underset{z\in\Omega}{\operatorname{argmin}}\,\varphi(z)$ *matters.*

The above expression is for twice-differentiable φ.

Asymptotic analysis

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

Laplace's method says... *... only the contribution at* $x^* = \underset{z\in\Omega}{\operatorname{argmin}}\,\varphi(z)$ *matters.*

Applying $-\frac{1}{\lambda}\log(\cdot)$ to both sides,

$$-\frac{1}{\lambda}\log\left[\int_\Omega h(z)\exp\left(-\lambda\varphi(z)\right)\mathrm{d}z\right] \sim \varphi(x^*) + O(\lambda^{-1}) \qquad \text{as } \lambda\to\infty$$

Asymptotic analysis

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

Laplace's method says... *... only the contribution at* $x^* = \underset{z\in\Omega}{\operatorname{argmin}}\,\varphi(z)$ *matters.*

Applying $-\frac{1}{\lambda}\log(\cdot)$ to both sides,

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

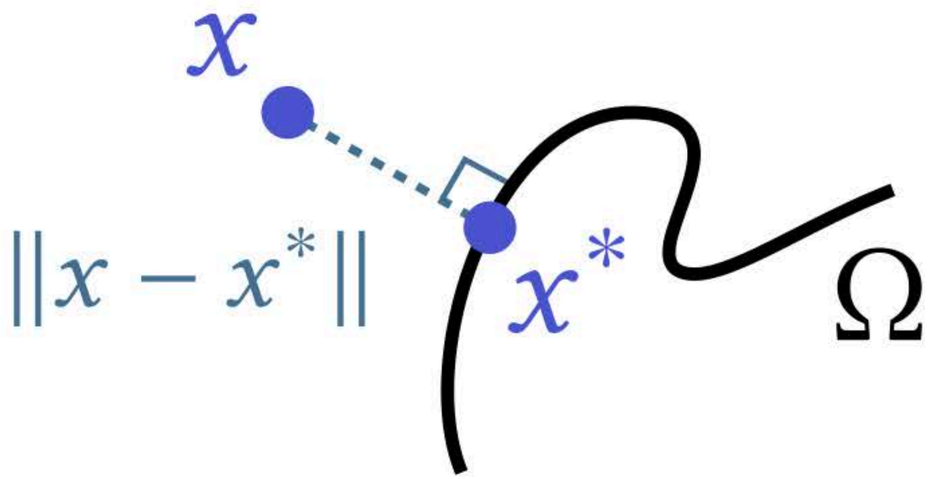
minimum of φ

Deriving convolutional distance

distance

Use $\varphi_x(z) = \|x - z\|$ to get $\|x - x^*\| = \min_{z \in \Omega} \|x - z\|$ as $\lambda \to \infty$.

distance from x to closest point in Ω



$$-\frac{1}{\lambda} \log \left[\int_\Omega h(z) \exp\left(-\lambda \varphi(z)\right) \mathrm{d}z \right] \sim \varphi(x^*) + O(\lambda^{-1}) \qquad \text{as } \lambda \to \infty$$

A general convolutional formula for distance

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \varphi(z)\right) \mathrm{d}z \right]$$

where $\varphi_x(z) = \|x - z\|$.

A general convolutional formula for distance

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \varphi(z)\right) \mathrm{d}z \right]$$

where $\varphi_x(z) = \|x - z\|$.

Examples:

- *Varadhan's formula* [Varadhan 1967]
- *Hopf-Cole transformations* for distance [Belyaev & Fayolle 2015; Lipman 2021]
- *LogSumExp* distance [Madan & Levin 2022]
- *Kreisselmeier-Steinhauser* functions [Kreisselmeier & Steinhauser 1980]
- *Schrödinger distance transform* [Gurumoorthy & Rangarajan 2009]
- other instances [Karam et al. 2019, Abgrall 2022]
- discrete LogSumExp

We unify signed distance with reconstruction

We unify signed distance with reconstruction

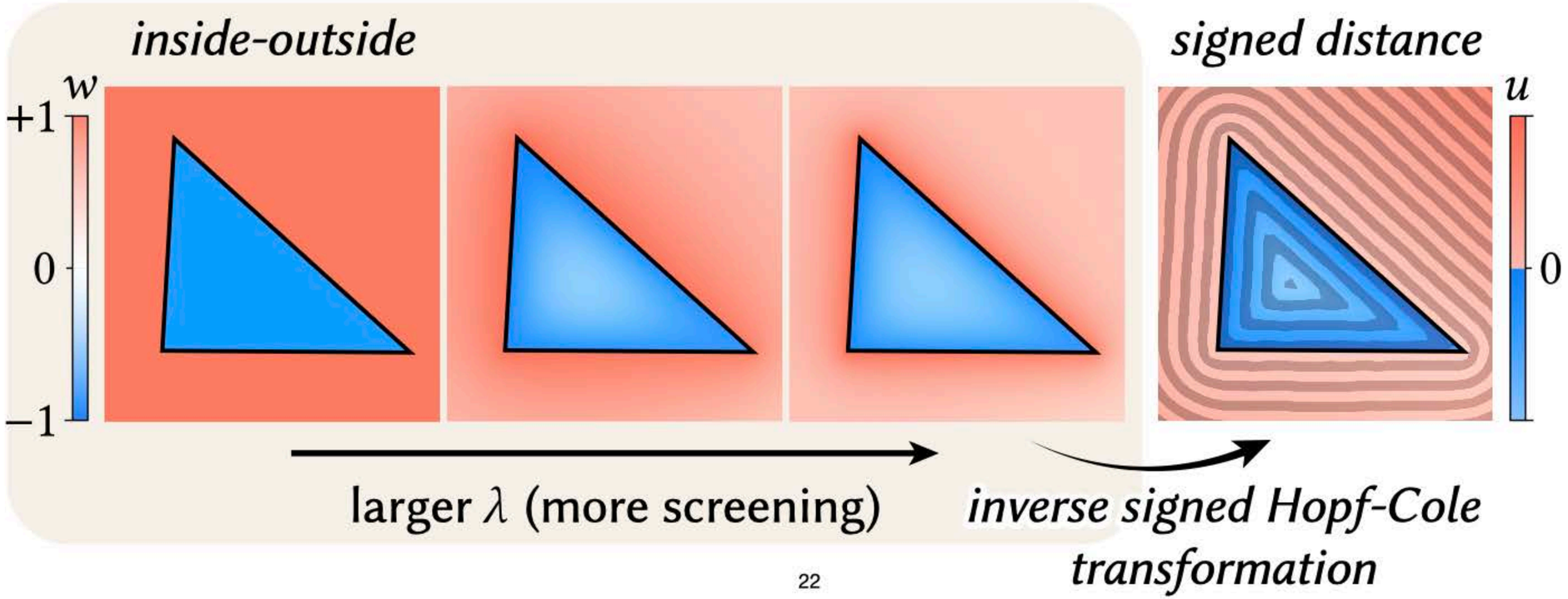
eikonal equation → *jump screened Laplace equation* → *screened Poisson equation* → *boundary integral*

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

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

$$\Delta w(x) - \lambda^2 w(x) = -2(\nabla \cdot n(x))\mu_\Omega(x)$$

$$w(x) = 2\int_\Omega \frac{\partial G^\lambda(x,z)}{\partial n(z)}\,\mathrm{d}z$$



We unify signed distance with reconstruction

eikonal equation → *jump screened Laplace equation* → *screened Poisson equation* → *boundary integral*

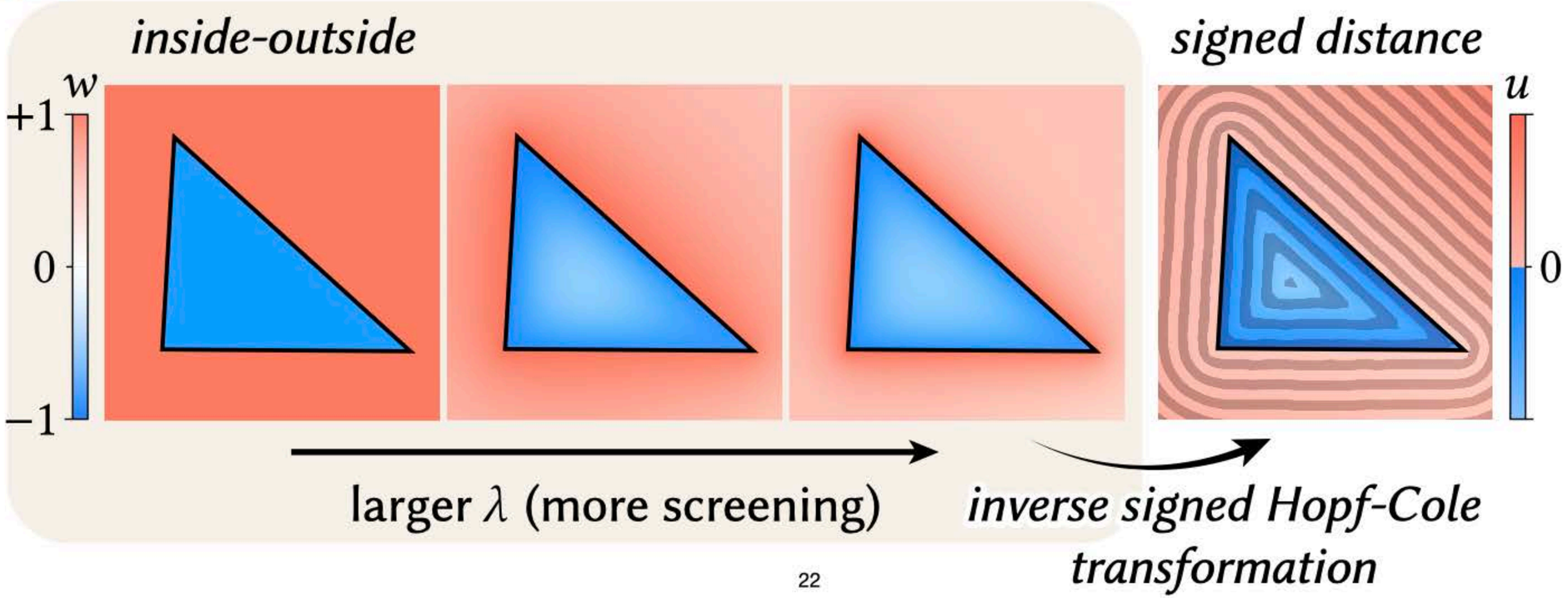
$$\begin{aligned} \|\nabla u(x)\|^2 &= 1, \quad x \notin \Omega \\ u(x) &= 0 \quad x \in \Omega \\ \frac{\partial u}{\partial n}(x) &= 1 \end{aligned}$$

$$\begin{aligned} \Delta w(x) - \lambda^2 w(x) &= 0, \quad x \notin \Omega \\ w^{\pm}(x) &= \quad x \in \Omega \\ & \quad \in \Omega \end{aligned}$$

$$\Delta w(x) - \lambda^2 w(x) = -2(\nabla \cdot n(x))\mu_\Omega(x)$$

$$w(x) = 2\int_\Omega \frac{\partial G^\lambda(x,z)}{\partial n(z)}\,\mathrm{d}z$$

$\lambda \to 0$: **winding numbers** (*jump harmonic functions*) [Feng et al. 2023]



We unify signed distance with reconstruction

eikonal equation → *jump screened Laplace equation* → *Laplace equation* → *boundary integral*

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

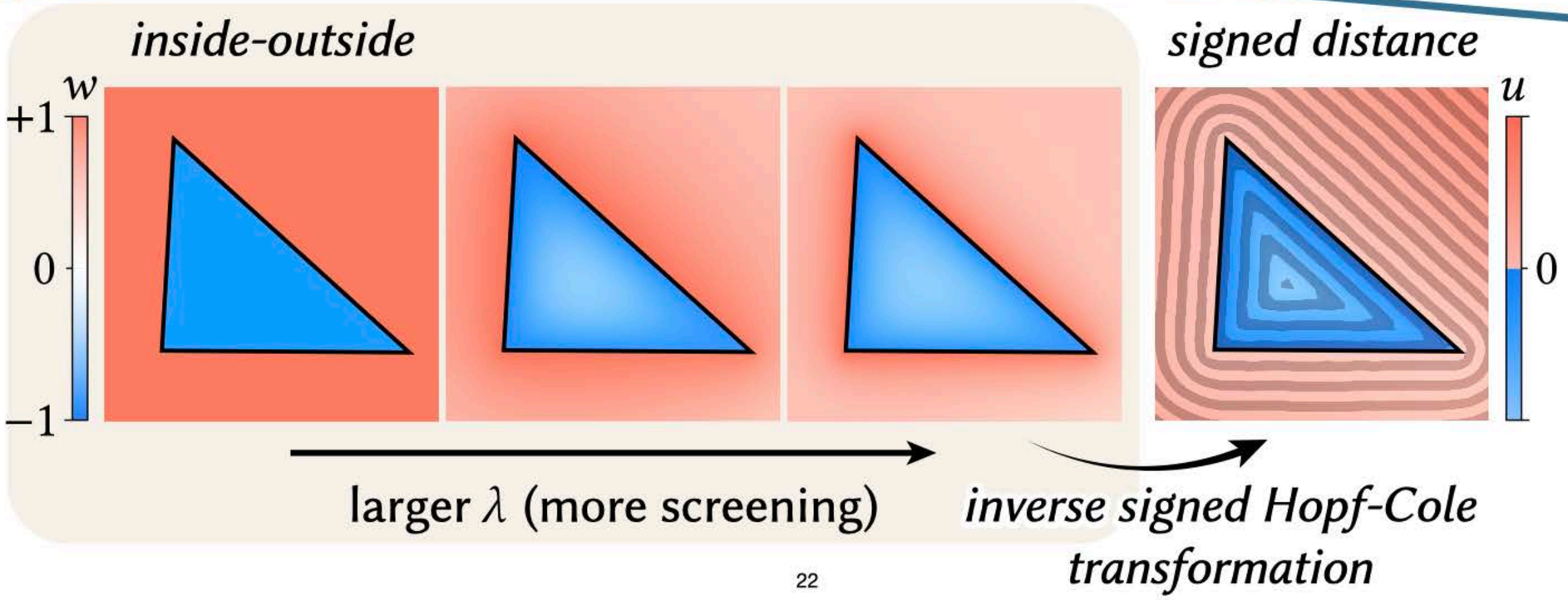
$$\begin{aligned} \Delta w(x) - \lambda^2 w(x) &= 0, \quad x \notin \Omega \\ w^{\pm}(x) &= \pm 1 \quad x \in \Omega \end{aligned}$$

$$\frac{\partial G^{\lambda}(x,z)}{\partial n(z)}\,\mathrm{d}z$$

Poisson Surface Reconstruction
[Kazhdan et al. 2006]

$\lambda \to 0$: **winding numbers**
(*jump harmonic functions*) [Feng et al. 2023]

regularized winding number
[Chen et al. 2024]



We unify signed distance with reconstruction

eikonal equation → *jump screened Laplace equation* → *Laplace equation* → *boundary integral*

$$\begin{aligned} \|\nabla u(x)\|^2 &= 1, \quad x \notin \Omega \\ u(x) &= 0 \quad x \in \Omega \\ \frac{\partial u}{\partial n}(x) &= 1 \end{aligned} \qquad \begin{aligned} \Delta w(x) - \lambda^2 w(x) &= 0, \quad x \notin \Omega \\ w^{\pm}(x) &= \pm 1 \quad x \in \Omega \end{aligned}$$

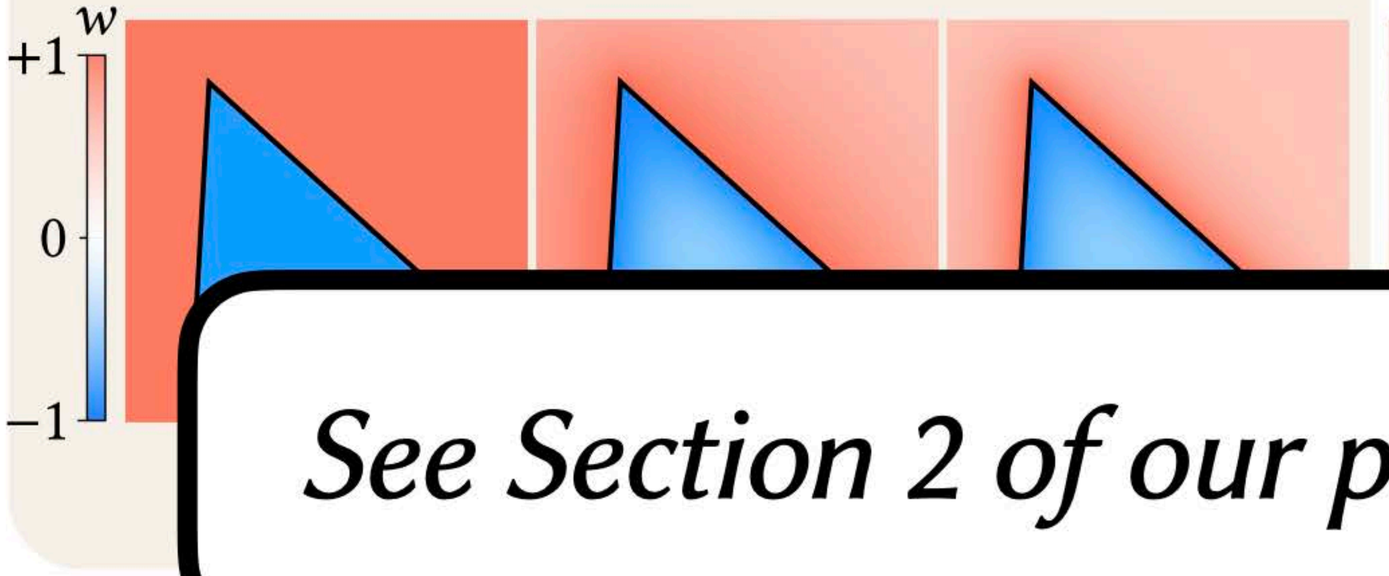
$$\frac{\partial G^{\lambda}(x,z)}{\partial n(z)}\,\mathrm{d}z$$

Poisson Surface Reconstruction
[Kazhdan et al. 2006]

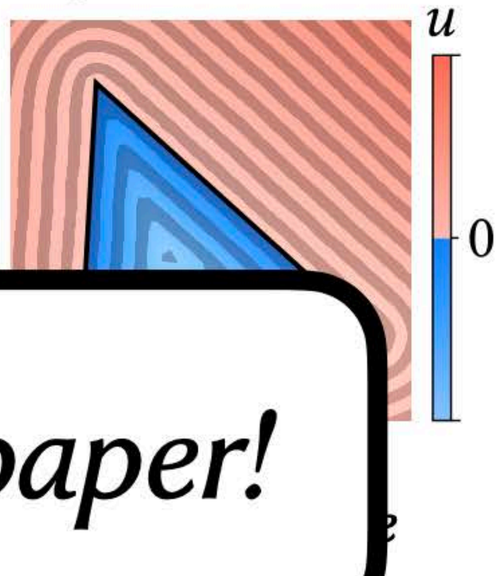
$\lambda \to 0$: **winding numbers**
(*jump harmonic functions*) [Feng et al. 2023]

regularized winding number
[Chen et al. 2024]

inside-outside



signed distance

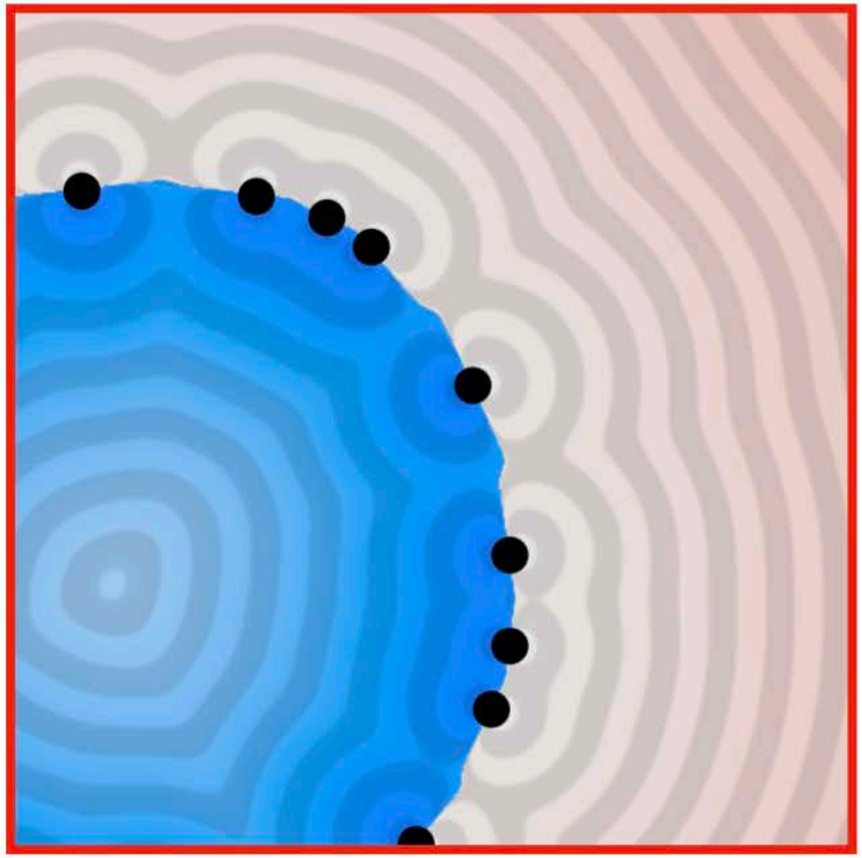


See Section 2 of our paper!

Convolutional formulas break down on real data

input point cloud P

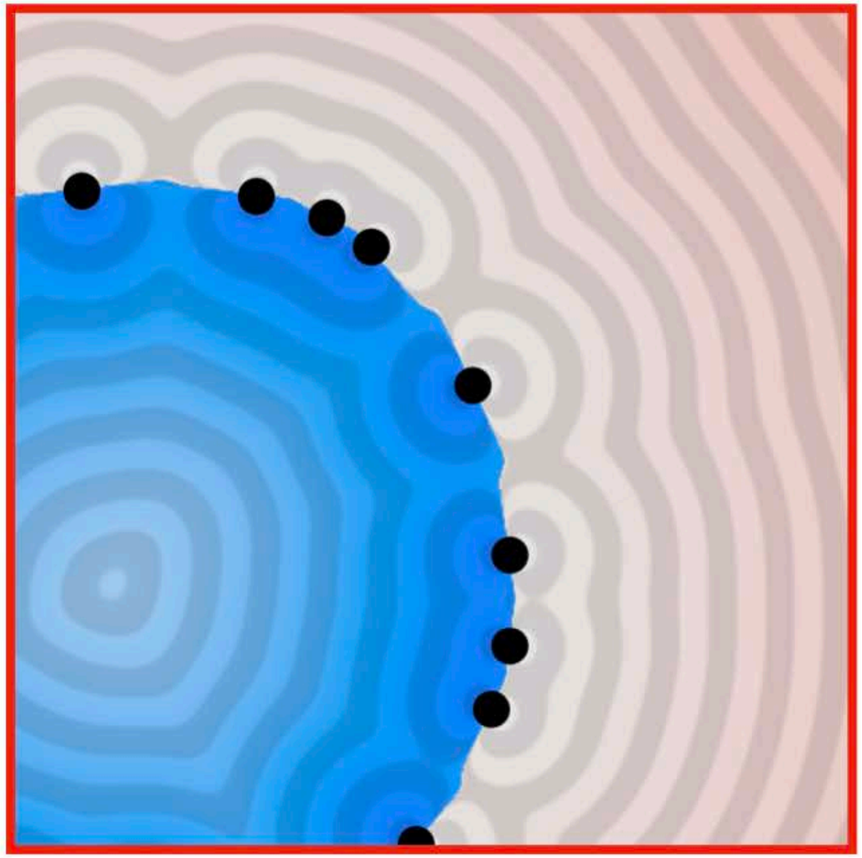
signed distance approximation



Convolutional formulas break down on real data

input point cloud P

signed distance approximation



We obtain distance to the *sampled* geometry, no matter what!

The exponential is (too) strong

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right] \sim \|x - x^*\|$$

$$x^* := \underset{z \in \Omega}{\operatorname{argmin}} \|x - z\|$$

The exponential is (too) strong

essentially a single-point approximation

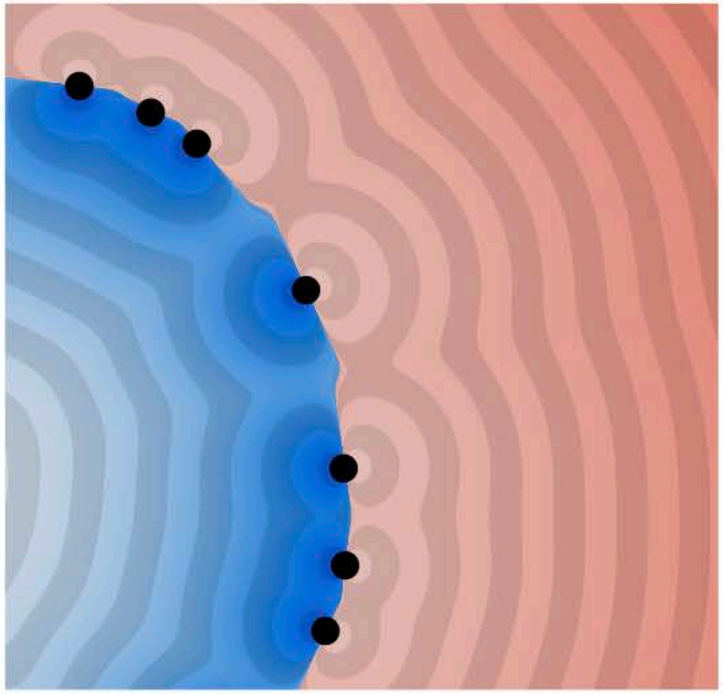
$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right] \sim \|x - x^*\| \text{ as } \lambda \to \infty$$

$$x^* := \operatorname*{argmin}_{z \in \Omega} \|x - z\|$$

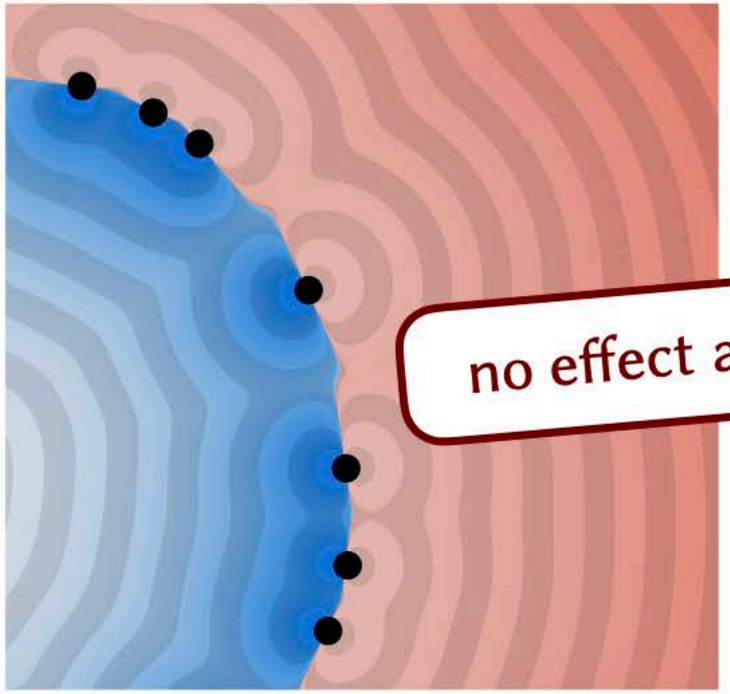
Regularization is futile for distance computation

regularization ≡ choosing h

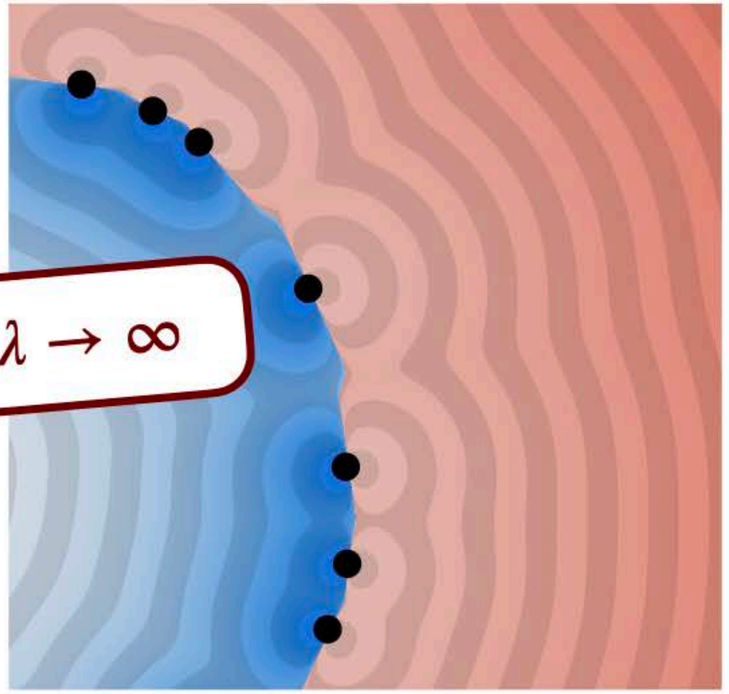
$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_\Omega h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right] \sim \|x - x^*\| \quad \text{as} \quad \lambda \to \infty$$



no regularization

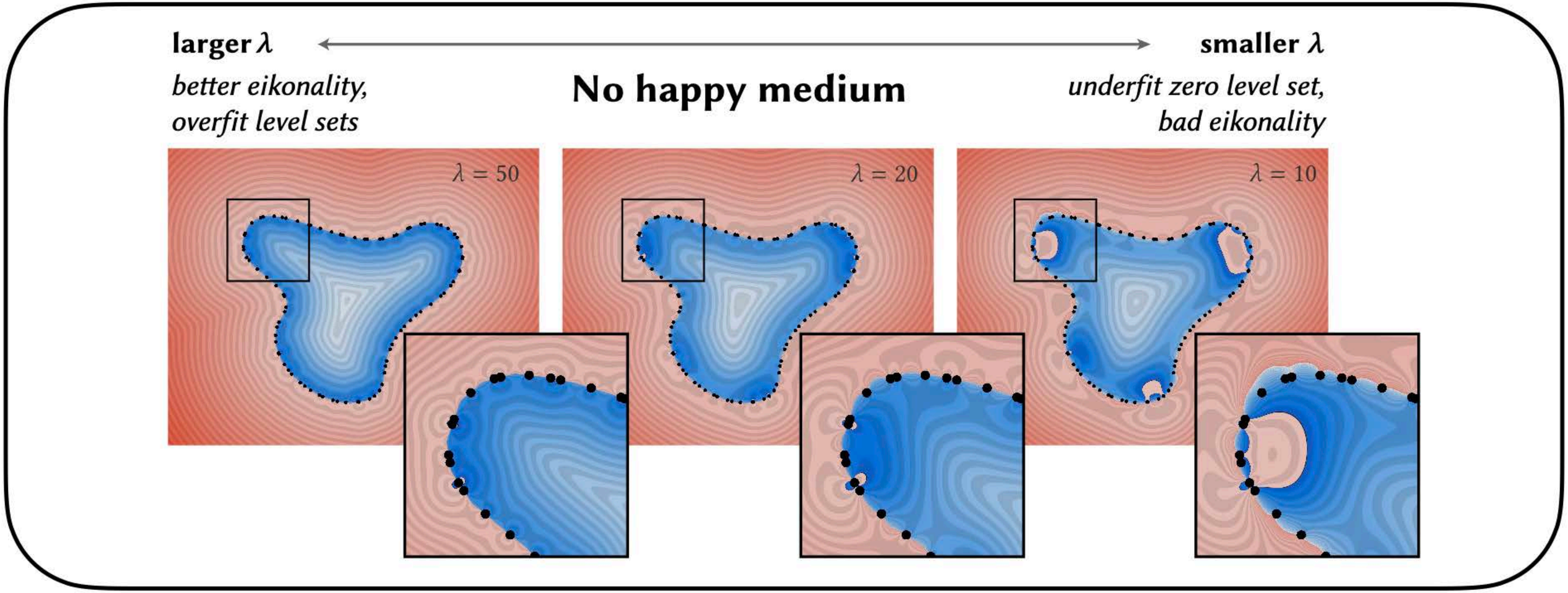


isotropic regularization



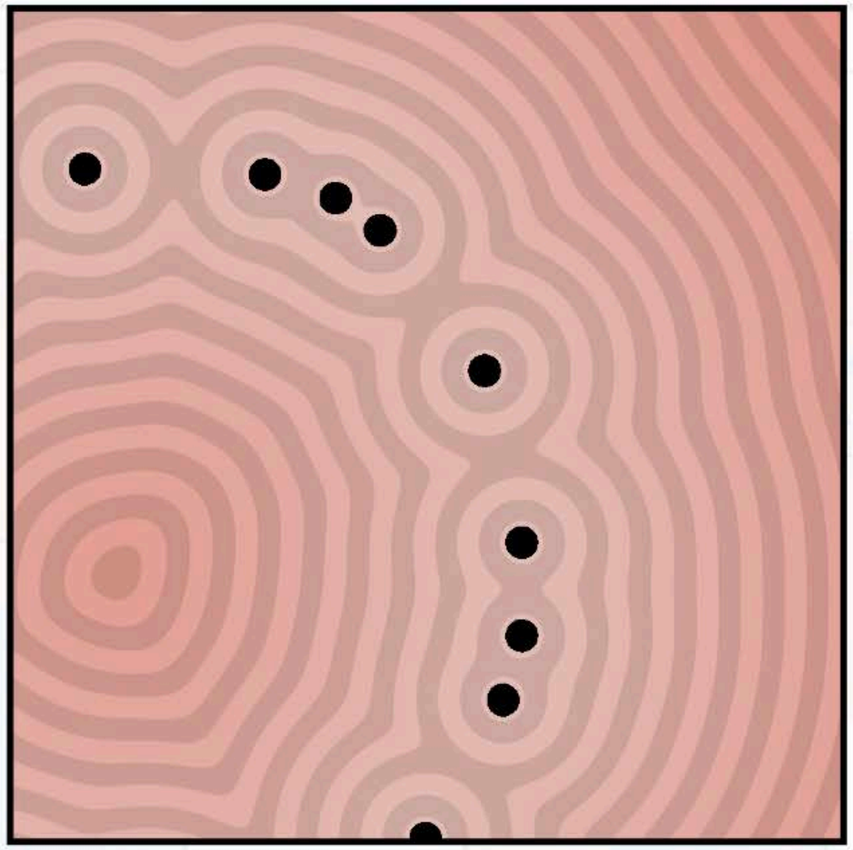
anisotropic regularization

Regression is coupled to distance — can't have both



The parameter λ is coupled to both regression quality and distance accuracy, in opposite directions.

Past distance formulas fail for point clouds



[Varadhan 1967, Kreisselmeier & Steinhauser 1980, Gurumoorthy & Rangarajan 2009, Karam et al. 2019, Abgrall 2022, Madan & Levin 2022]

A second convolutional distance formula

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

A second convolutional distance formula

Formula #1:

$$\widetilde{d}(x) := -\frac{1}{\lambda} \log \left[\int_{\Omega} h(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z \right]$$

Formula #2:

a.k.a. kernel density estimator

$$\widehat{d}(x) := \frac{\int_{\Omega} g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_{\Omega} \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

A second convolutional distance formula

$$\widehat{d}(x) := \frac{\int_\Omega g(z)\, \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z} \qquad \lambda \underset{\sim}{\to} \infty \qquad g(x^*)$$

$$x^* := \operatorname*{argmin}_{z \in \Omega} \|x - z\|$$

A second convolutional distance formula

closest-point interpolation

$$\widehat{d}(x) := \frac{\int_\Omega g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

$$\lambda \xrightarrow[\sim]{} \infty \qquad g(x^*)$$

$$x^* := \operatorname*{argmin}_{z \in \Omega} \|x - z\|$$

A second convolutional distance formula

closest-point interpolation

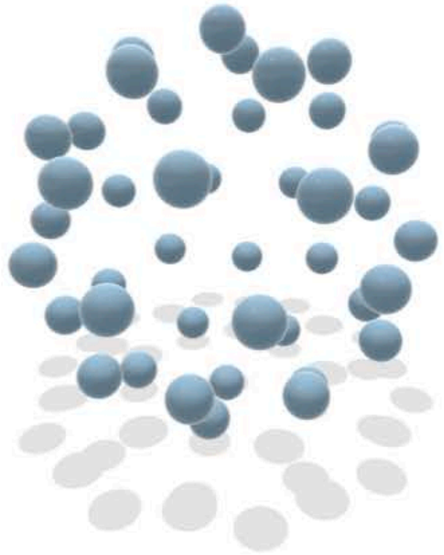
$$\widehat{d}(x) := \frac{\int_\Omega g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z} \qquad \lambda \underset{\sim}{\to} \infty \qquad g(x^*)$$

A second convolutional distance formula

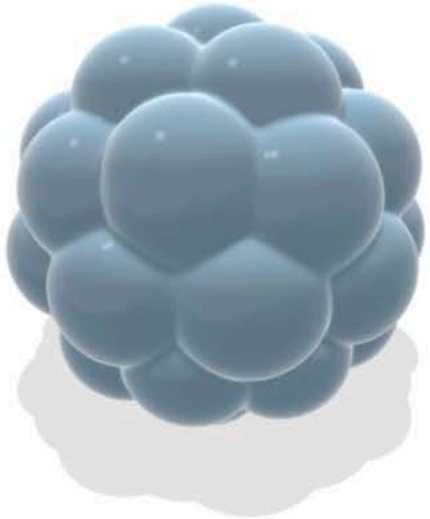
closest-point interpolation

$$\widehat{d}(x) := \frac{\int_\Omega g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z} \quad \lambda \underset{\sim}{\to} \infty \quad g(x^*)$$

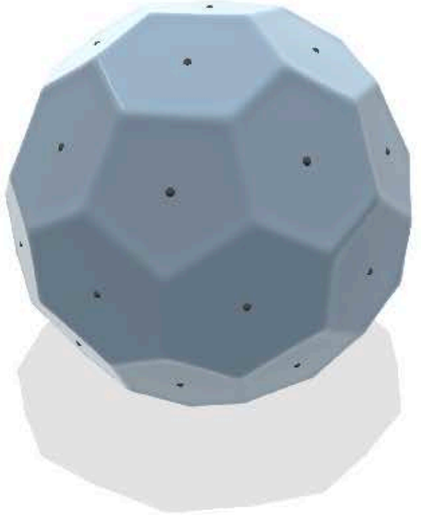
$g(z) = \|x - z\| - 0.1$



$g(z) = \|x - z\| - 1$



$g(z) = \langle x - z, n_z \rangle$



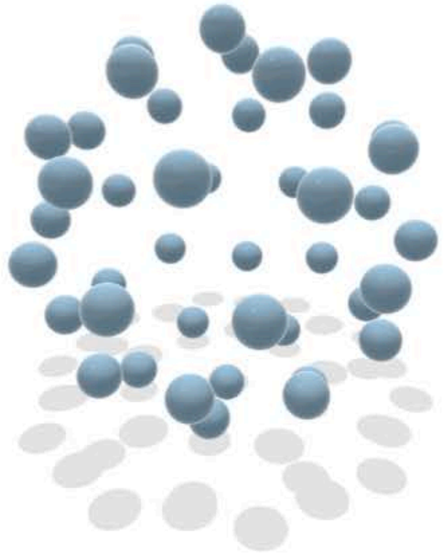
A second convolutional distance formula

a.k.a. kernel density estimator, partition-of-unity methods, softmax function, vector heat method , induced exponential function, …

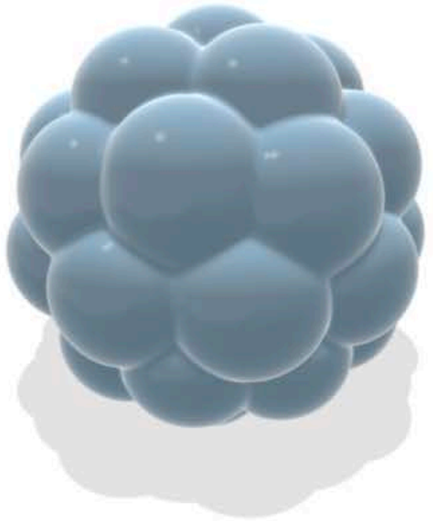
closest-point interpolation

$$\widehat{d}(x) := \frac{\int_\Omega g(z) \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z} \qquad \lambda \xrightarrow[\sim]{} \infty \qquad g(x^*)$$

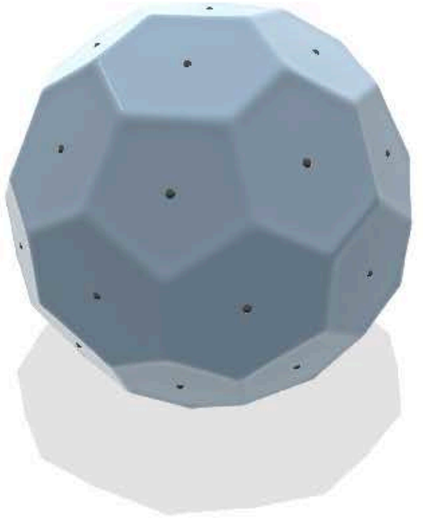
$g(z) = \|x - z\| - 0.1$



$g(z) = \|x - z\| - 1$



$g(z) = \langle x - z, n_z \rangle$



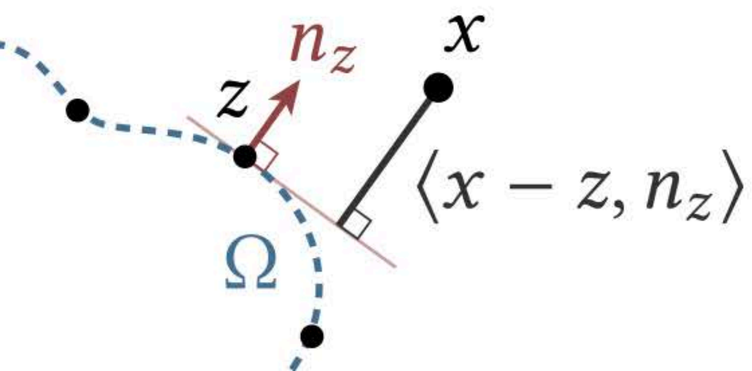
Linear approximations are undesirable...

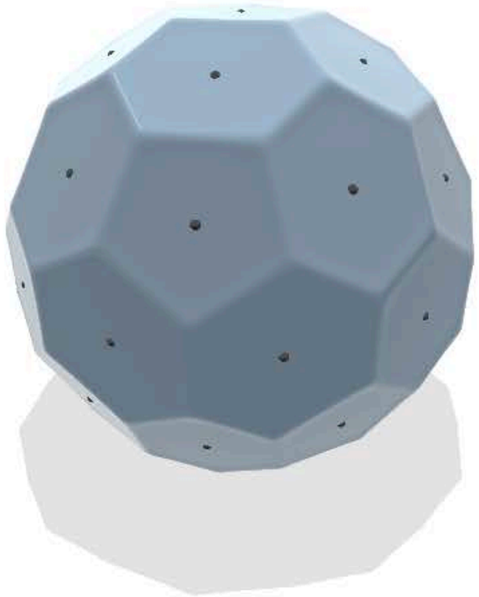
"normal" distance

$$\widehat{d}(x) = \frac{\int_\Omega \langle x - z, n_z \rangle \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

"pseudonormal test"
"tangent plane approximation"
"plane test"

[Alexa et al. 2001; Boissonnat & Cazals 2002; Bærentzen & Aanæs 2005, Kolluri 2008; Öztireli et al. 2009; Yang et al. 2025]





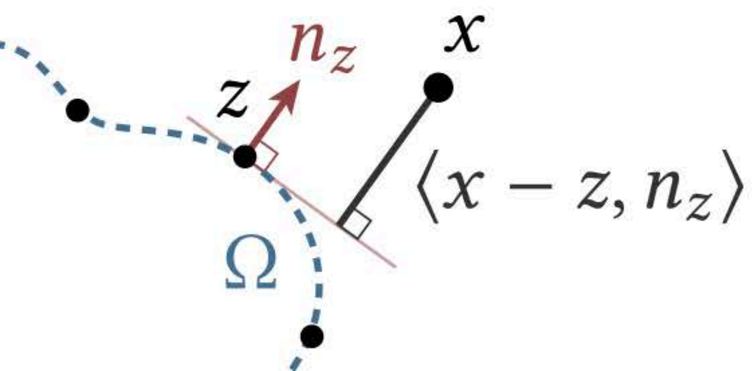
Linear approximations are undesirable...

"normal" distance

$$\widehat{d}(x) = \frac{\int_\Omega \langle x - z, n_z \rangle \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

"pseudonormal test"
"tangent plane approximation"
"plane test"

[Alexa et al. 2001; Boissonnat & Cazals 2002; Bærentzen & Aanæs 2005, Kolluri 2008; Öztireli et al. 2009; Yang et al. 2025]



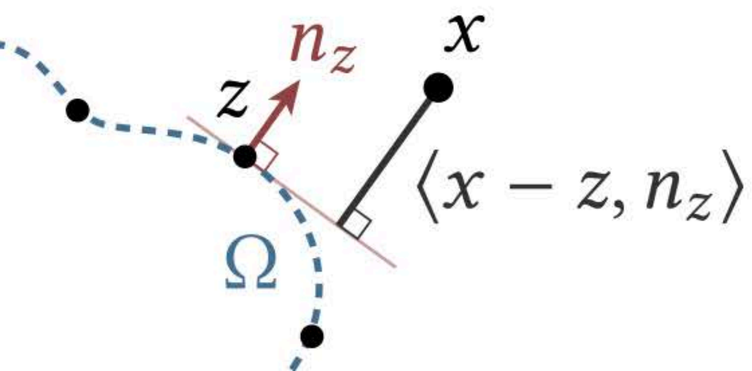
Linear approximations are undesirable...

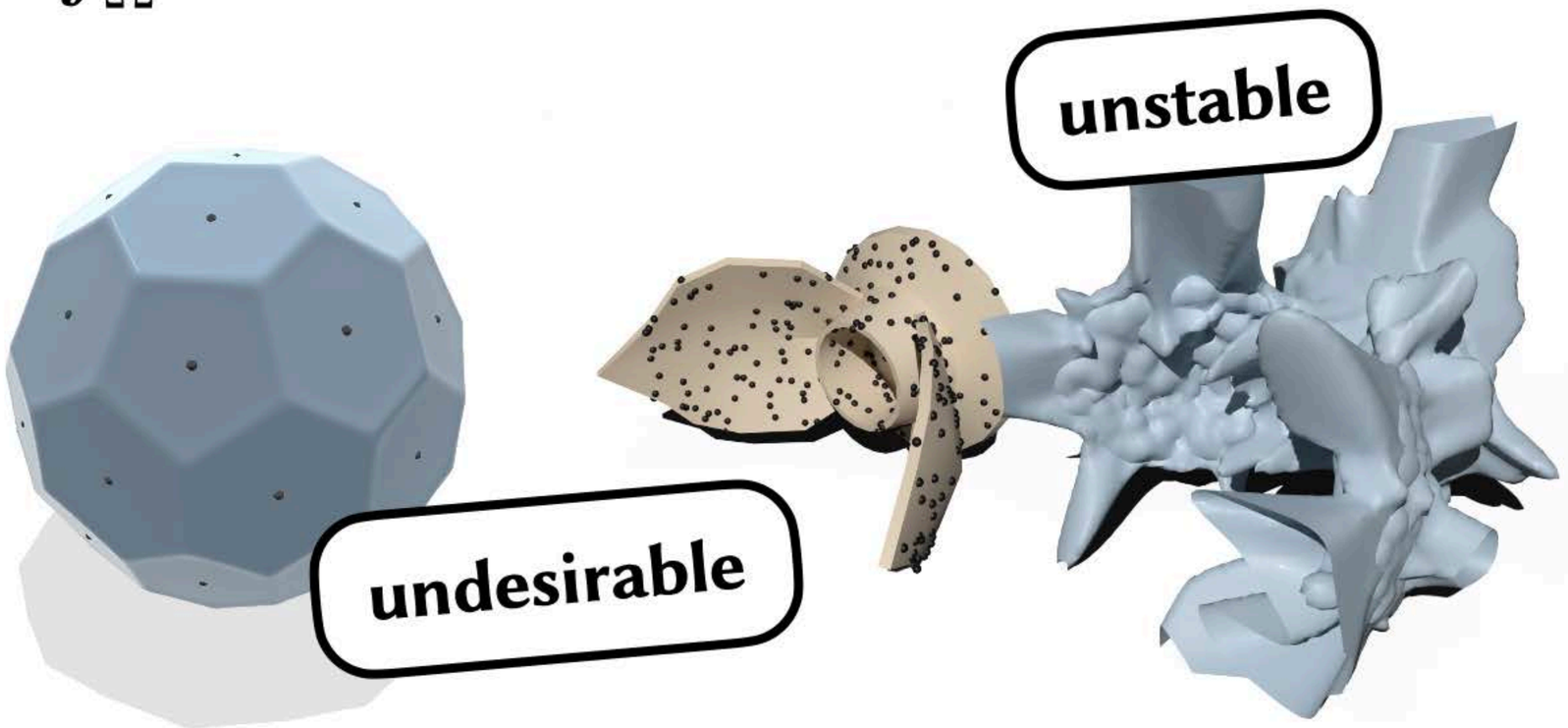
$$\widehat{d}(x) = \frac{\int_\Omega \langle x - z, n_z \rangle \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}{\int_\Omega \exp\left(-\lambda \|x - z\|\right) \mathrm{d}z}$$

"normal" distance

"pseudonormal test"
"tangent plane approximation"
"plane test"

[Alexa et al. 2001; Boissonnat & Cazals 2002; Bærentzen & Aanæs 2005, Kolluri 2008; Öztireli et al. 2009; Yang et al. 2025]





… so let's use second-order surfaces!

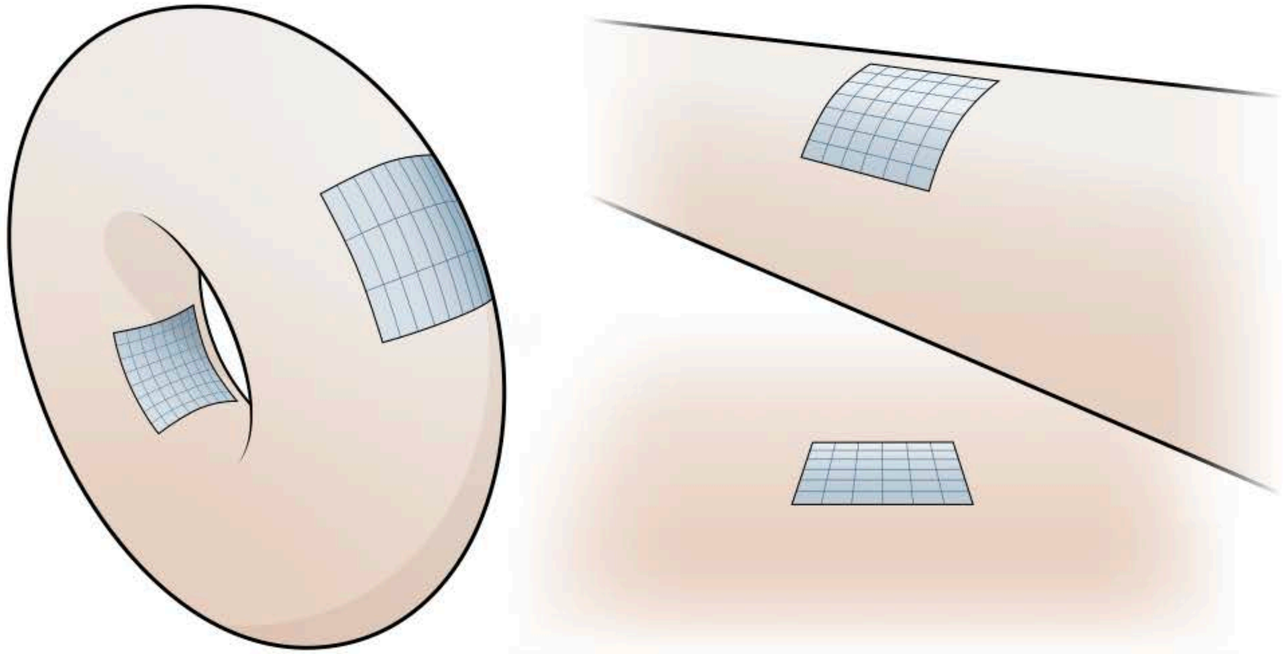
Discrete estimator: $$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}$$

… so let's use second-order surfaces!

torus distance!

Discrete estimator:

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

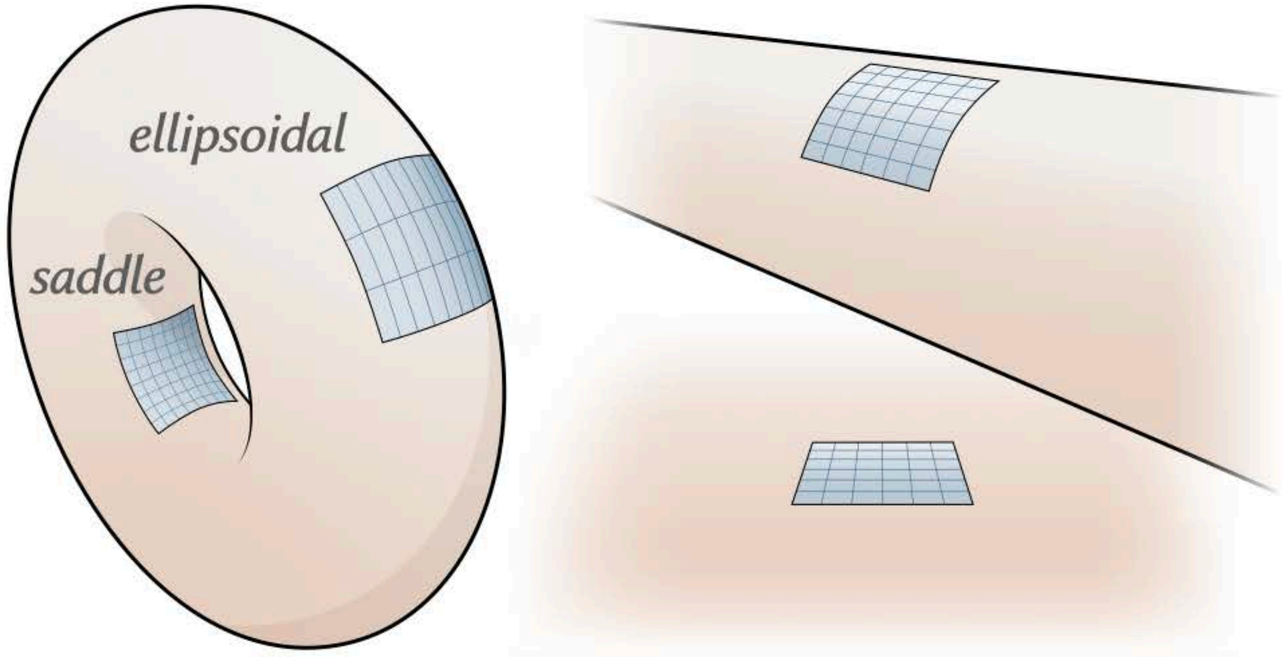


… so let's use second-order surfaces!

torus distance!

Discrete estimator:

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

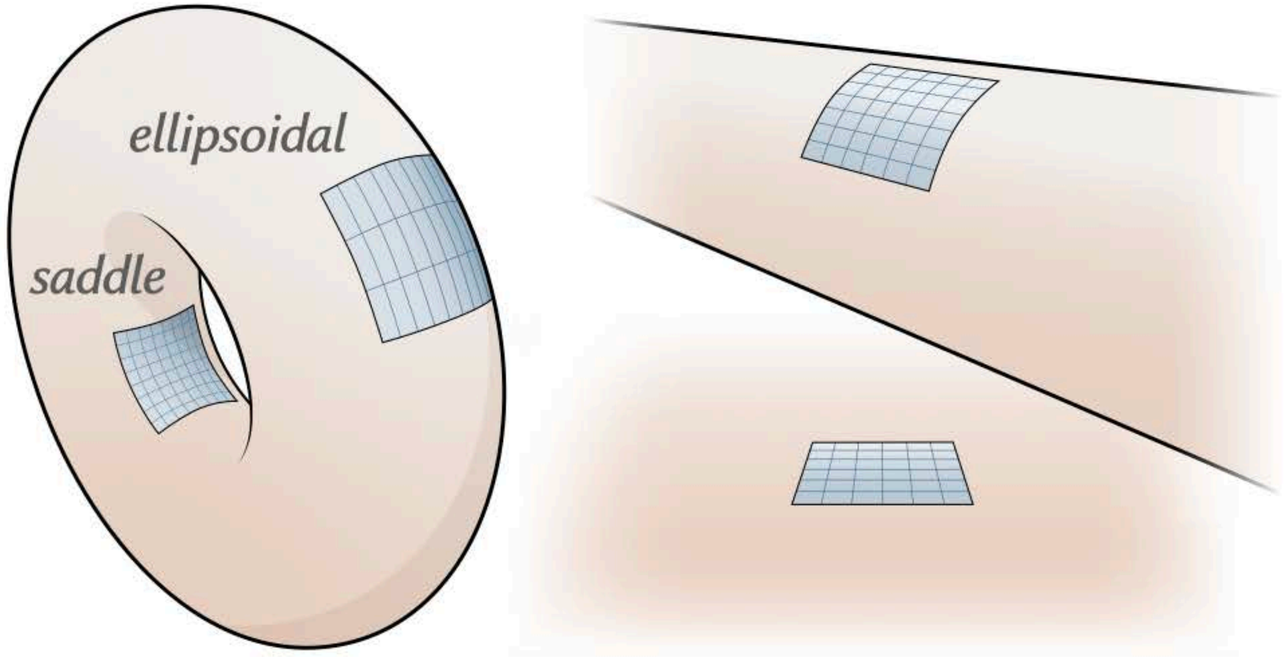


… so let's use second-order surfaces!

torus distance!

Discrete estimator:

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



Fitting geometric proxies to point clouds (planes, cylinders, spheres, quadrics, splines, polynomial patches…)

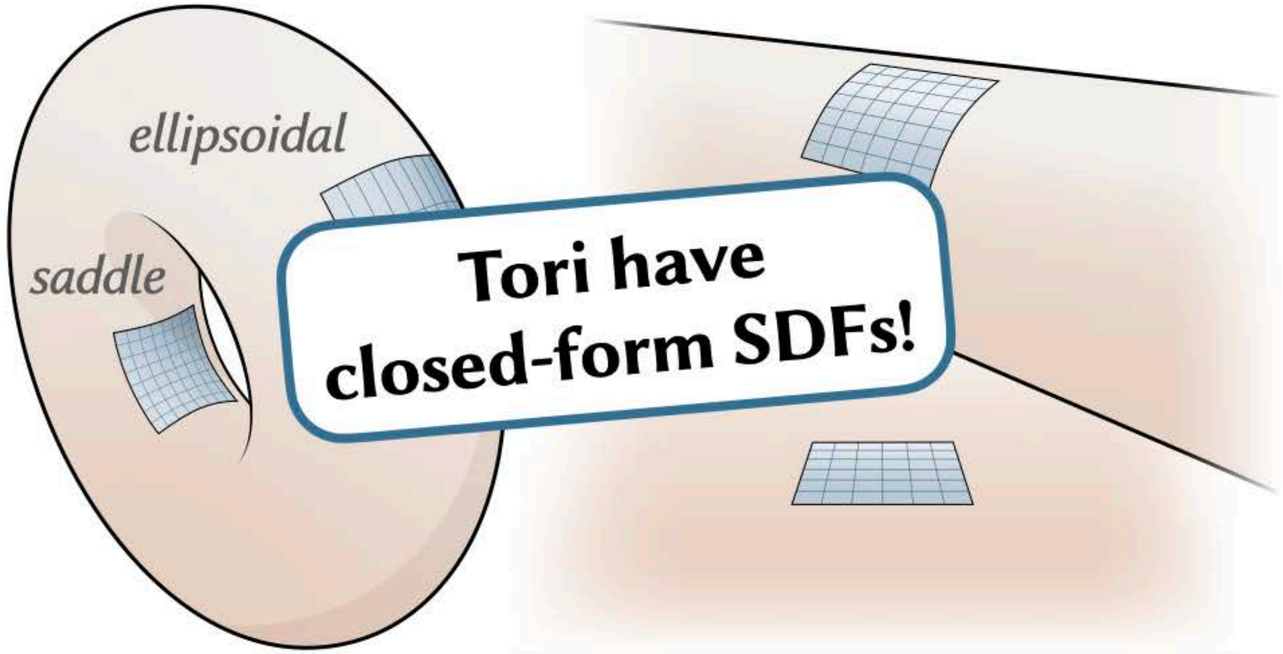
Faugeras 1983; Besl & Jain 1988; Cao et al. 1994; Kaveti et al. 1996; Yang & Lee 1999; Cohen-Steiner et al. 2004; Cazals & Pouget 2005; Wu & Kobbelt 2005; Simari & Singh 2005; Yan et al. 2006; Attene et al. 2006; Xia & Ju 2025…

Tori: Lukács et al. 1998; Liu et al. 2009; Eberly 2020
— but not used to construct SDFs!

… so let's use second-order surfaces!

torus distance!

Discrete estimator: $$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}$$

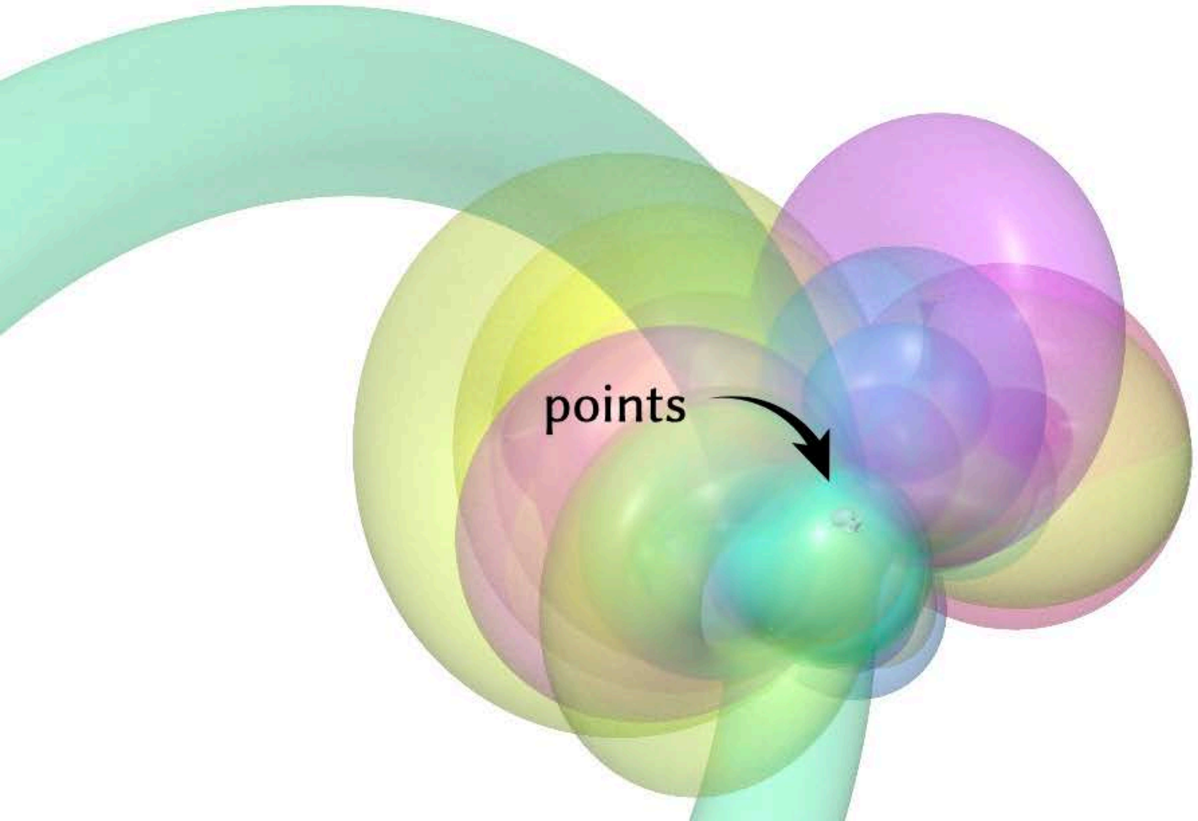


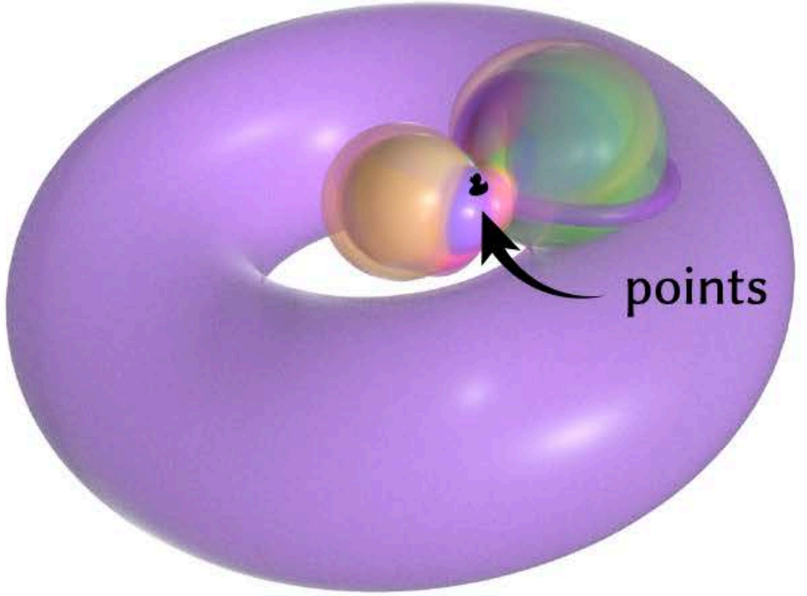
Fitting geometric proxies to point clouds (planes, cylinders, spheres, quadrics, splines, polynomial patches…)

Faugeras 1983; Besl & Jain 1988; Cao et al. 1994; Kaveti et al. 1996; Yang & Lee 1999; Cohen-Steiner et al. 2004; Cazals & Pouget 2005; Wu & Kobbelt 2005; Simari & Singh 2005; Yan et al. 2006; Attene et al. 2006; Xia & Ju 2025…

Tori: Lukács et al. 1998; Liu et al. 2009; Eberly 2020 — but not used to construct SDFs!

The tori look crazy... but they work together

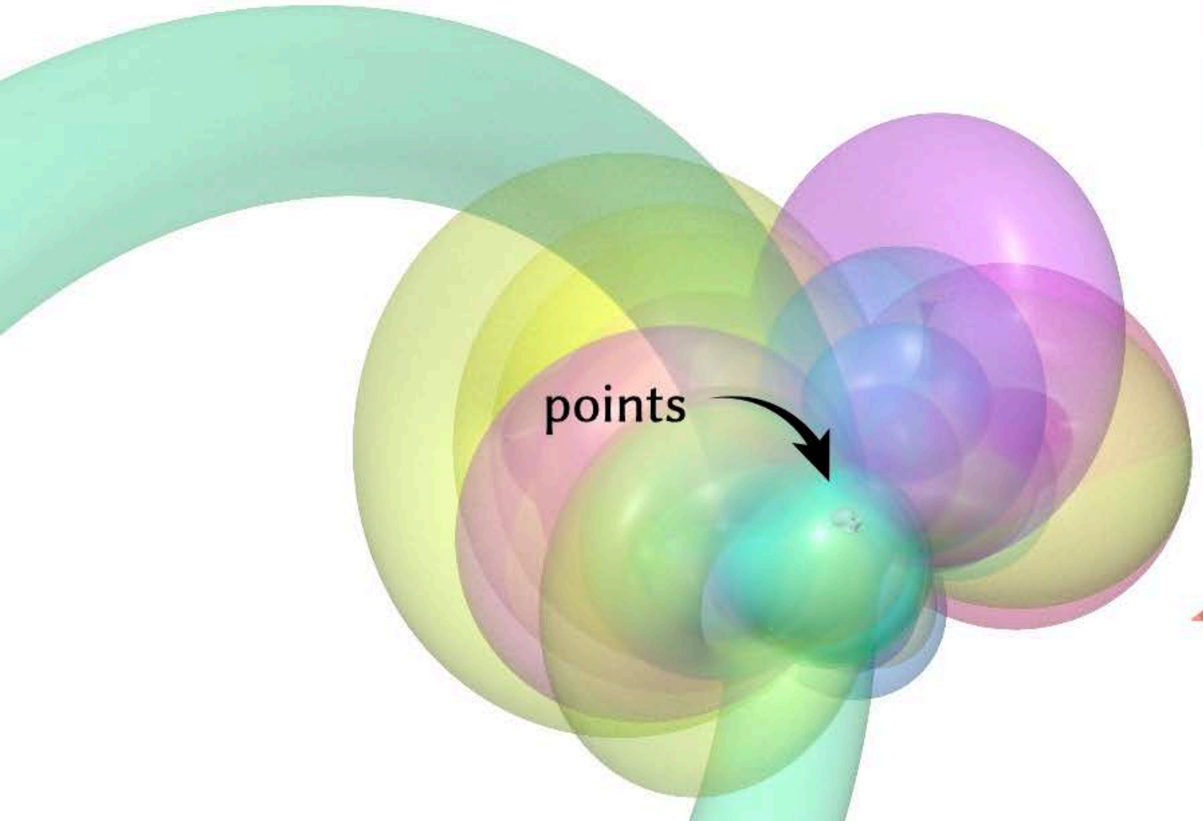


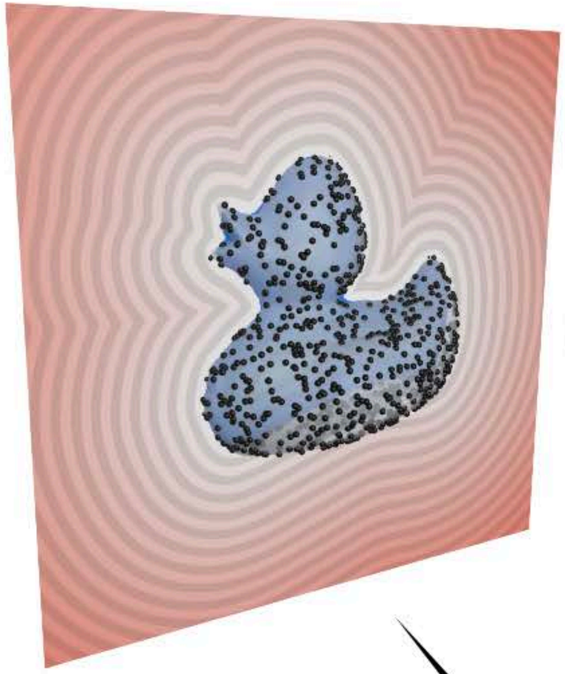


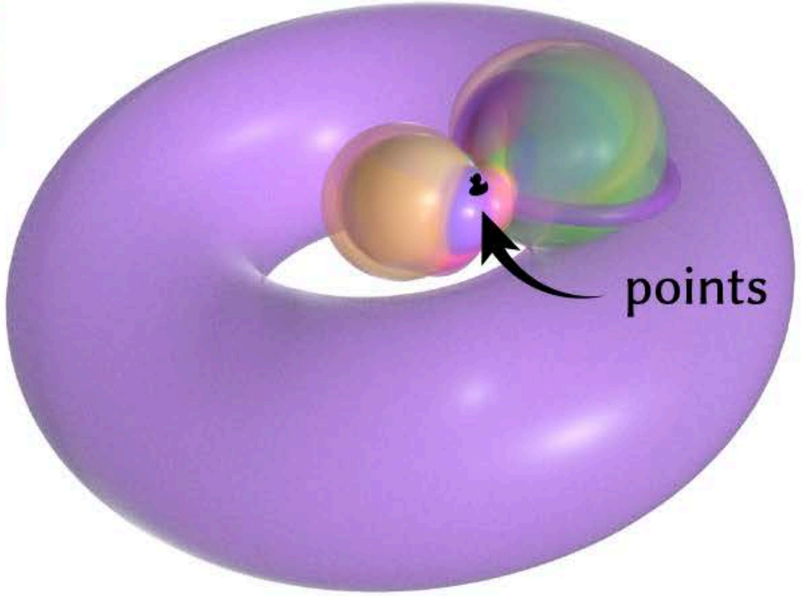
The tori look crazy... but they work together

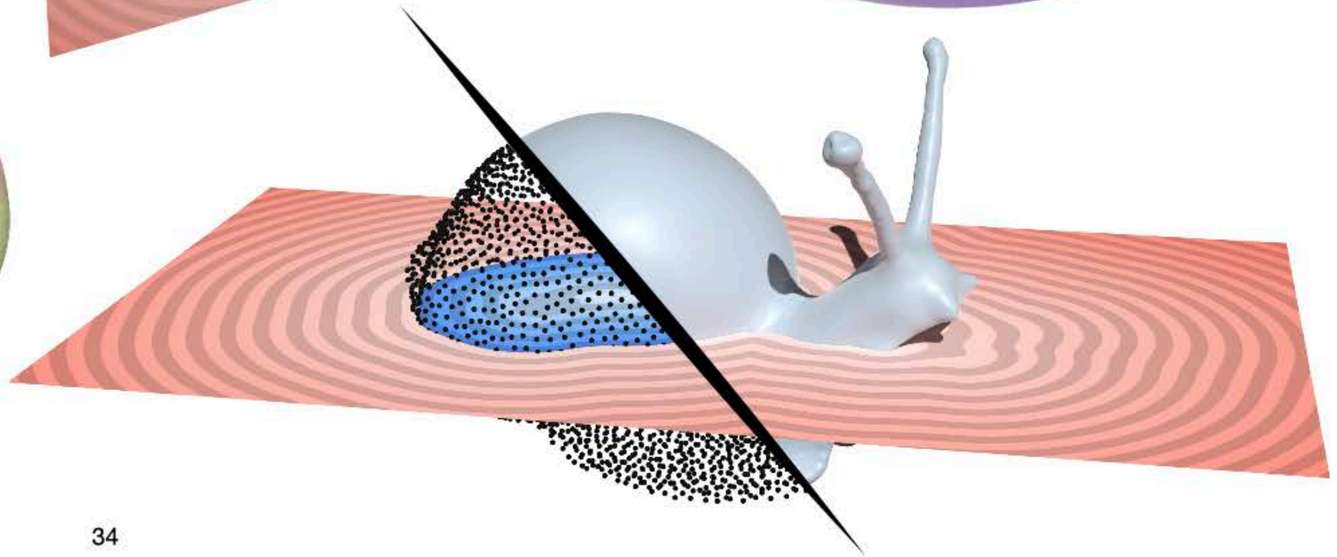
i-th torus SDF

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







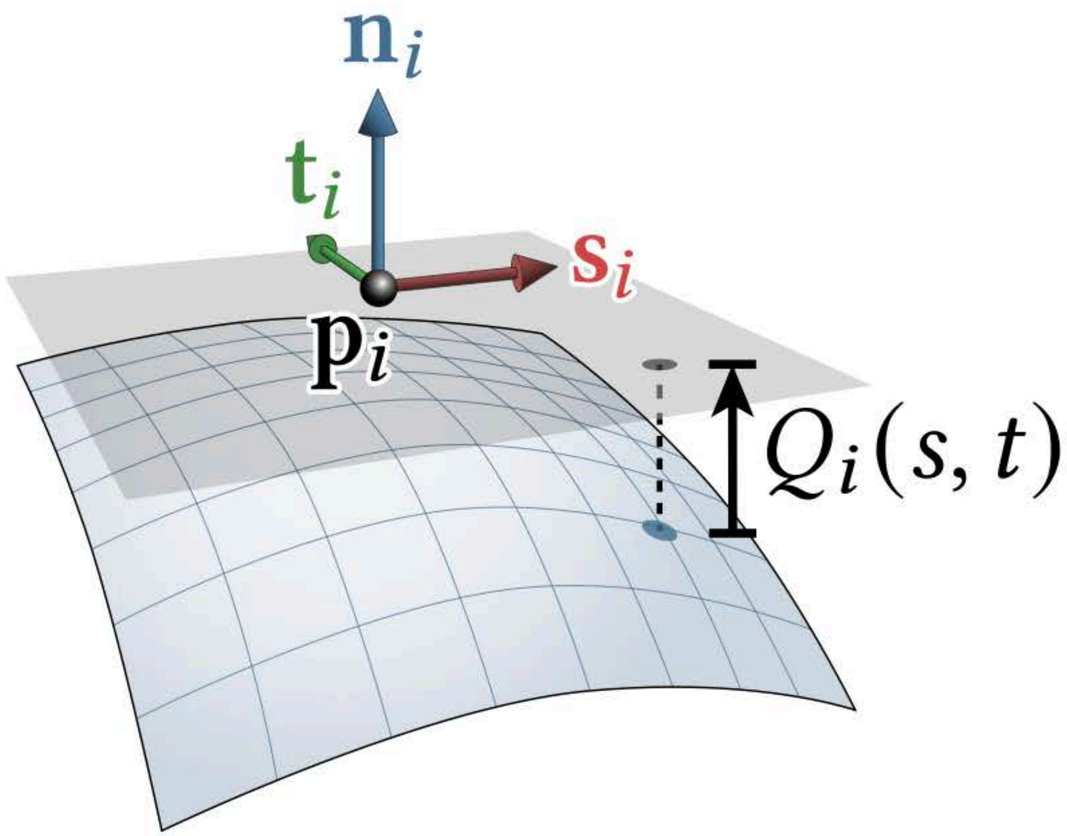


The surface, one polynomial patch at a time

The surface, one polynomial patch at a time

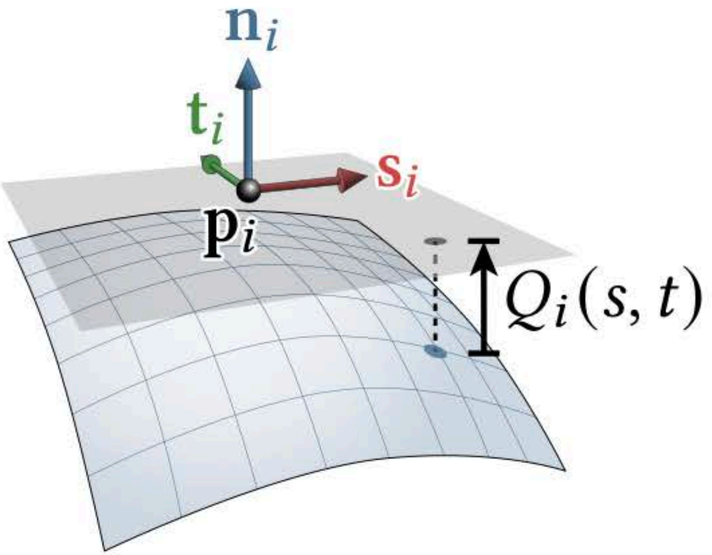
Quadratic polynomial around each point:

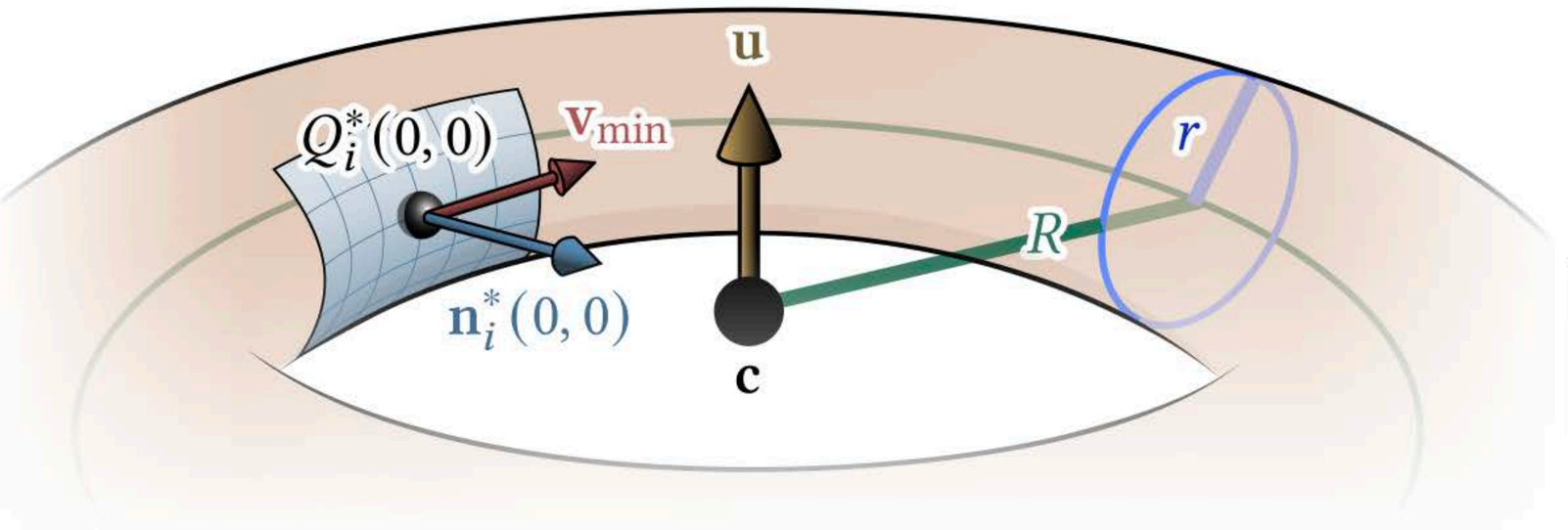
$$Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m$$



The surface, one polynomial patch at a time

$$Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m$$

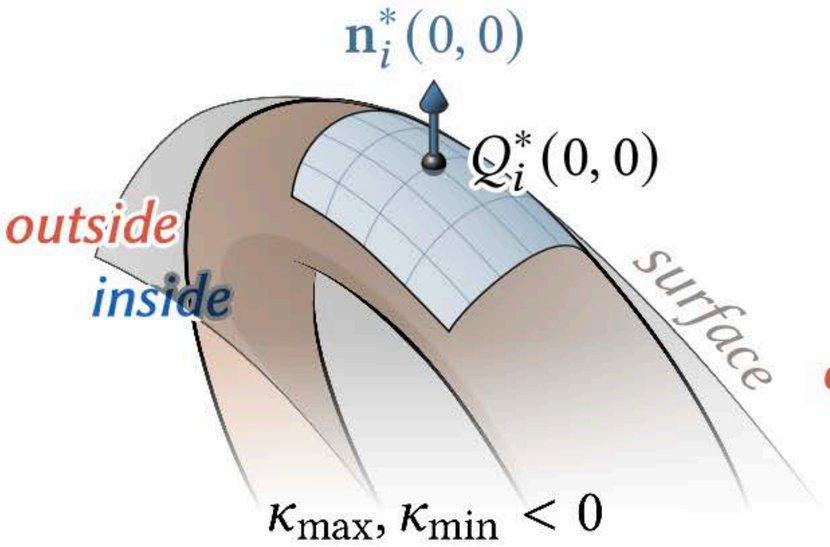


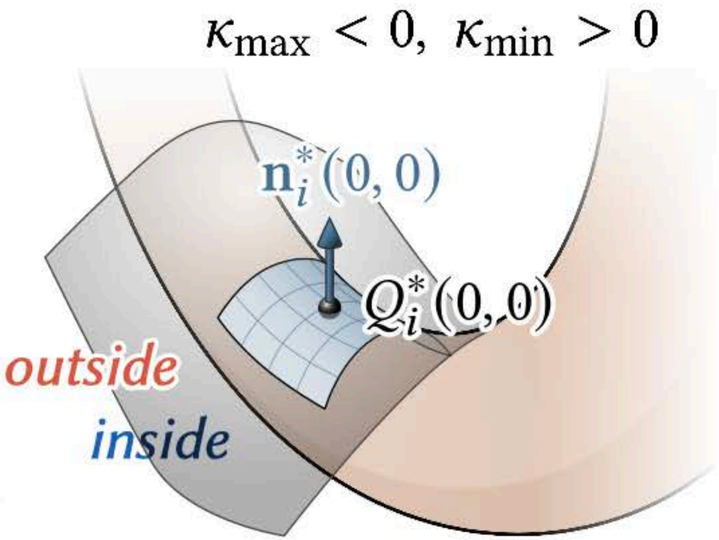


A torus is determined by the six coefficients

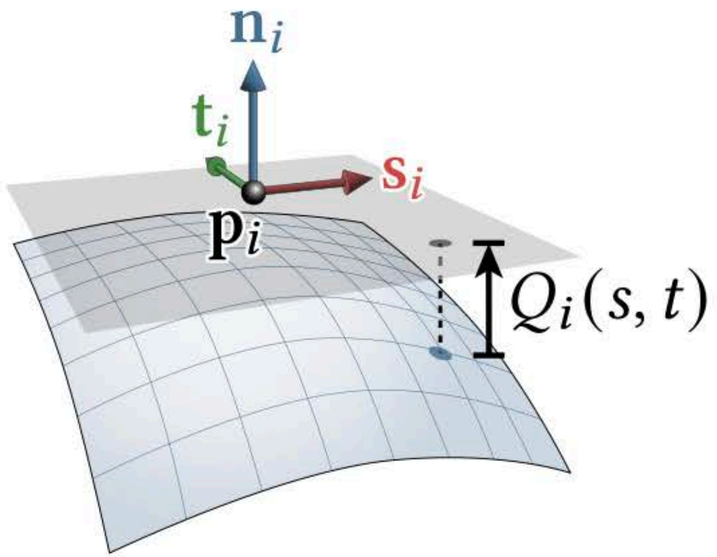
$a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$

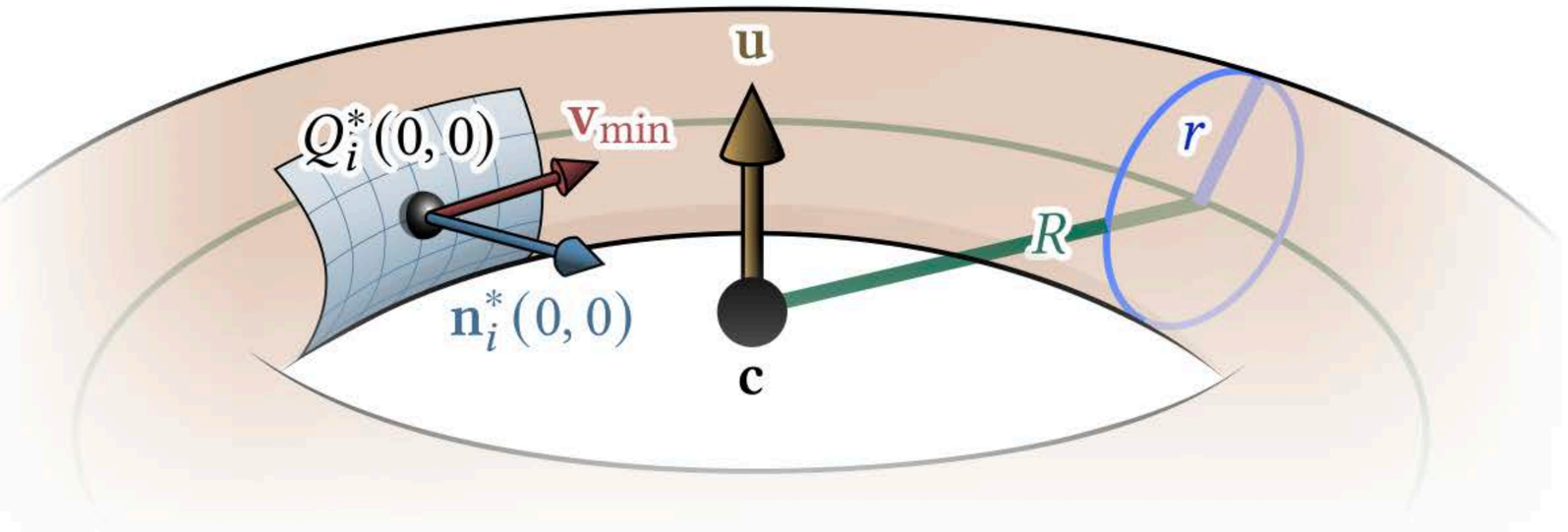
The surface, one polynomial patch at a time





$$Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m$$

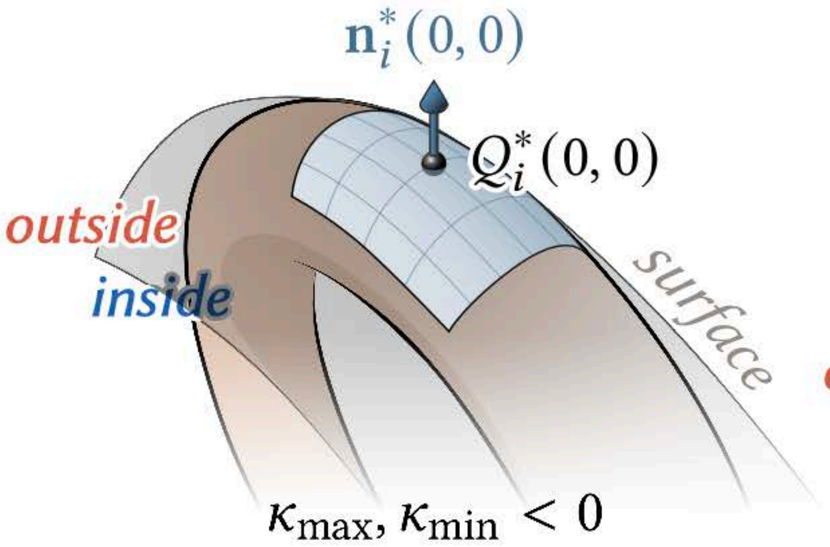


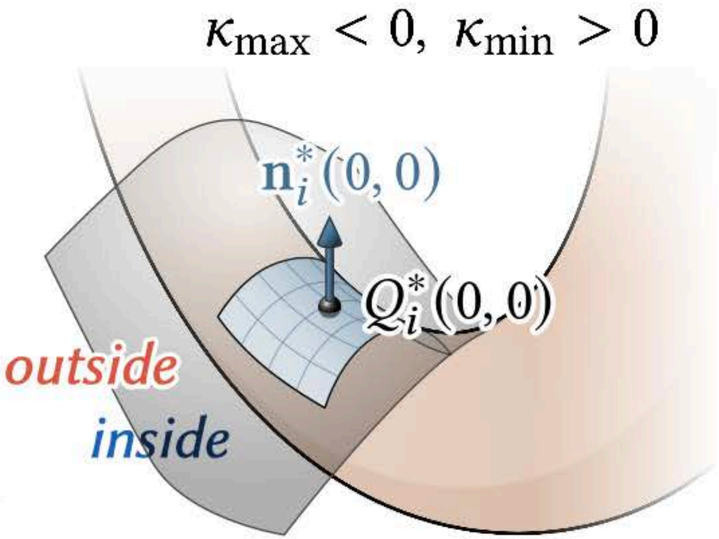


A torus is determined by the six coefficients

$a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$

The surface, one polynomial patch at a time



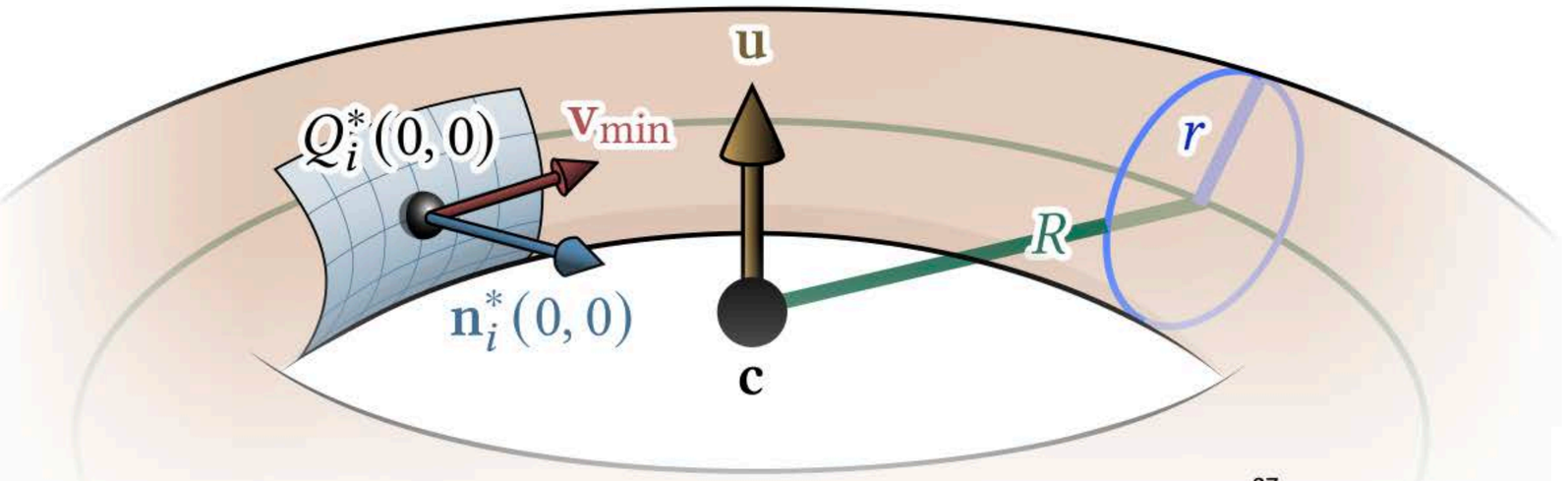


$$Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m$$

SDF of a torus:

$$\phi_{\mathbb{T}} = \|\mathbf{d}\| - r,$$

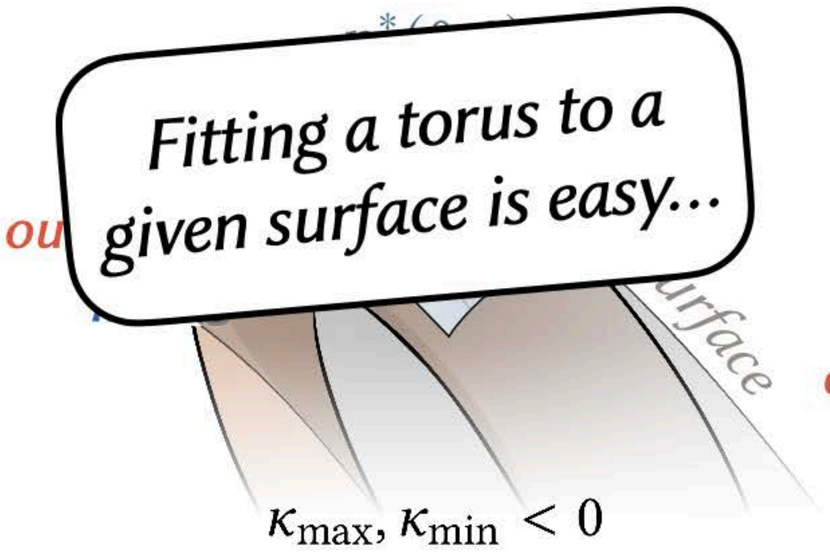
$$\mathbf{d} := (\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| - R, \langle \mathbf{x}-\mathbf{c}, \mathbf{u} \rangle)$$

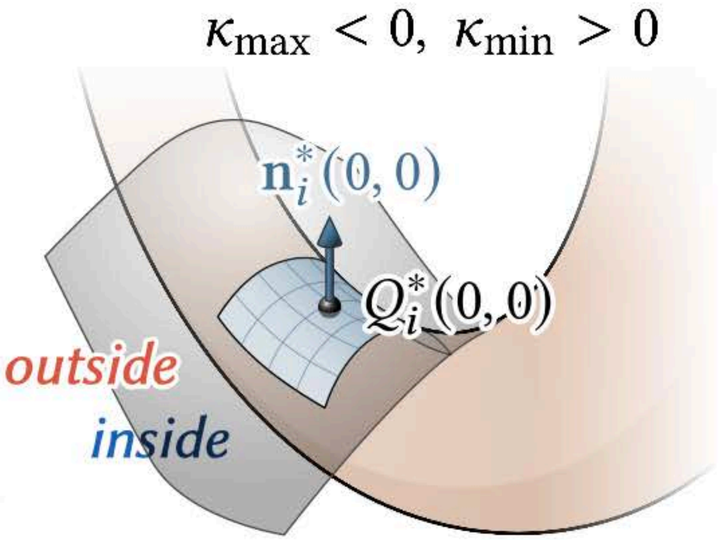


A torus is determined by the six coefficients

$a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$

The surface, one polynomial patch at a time



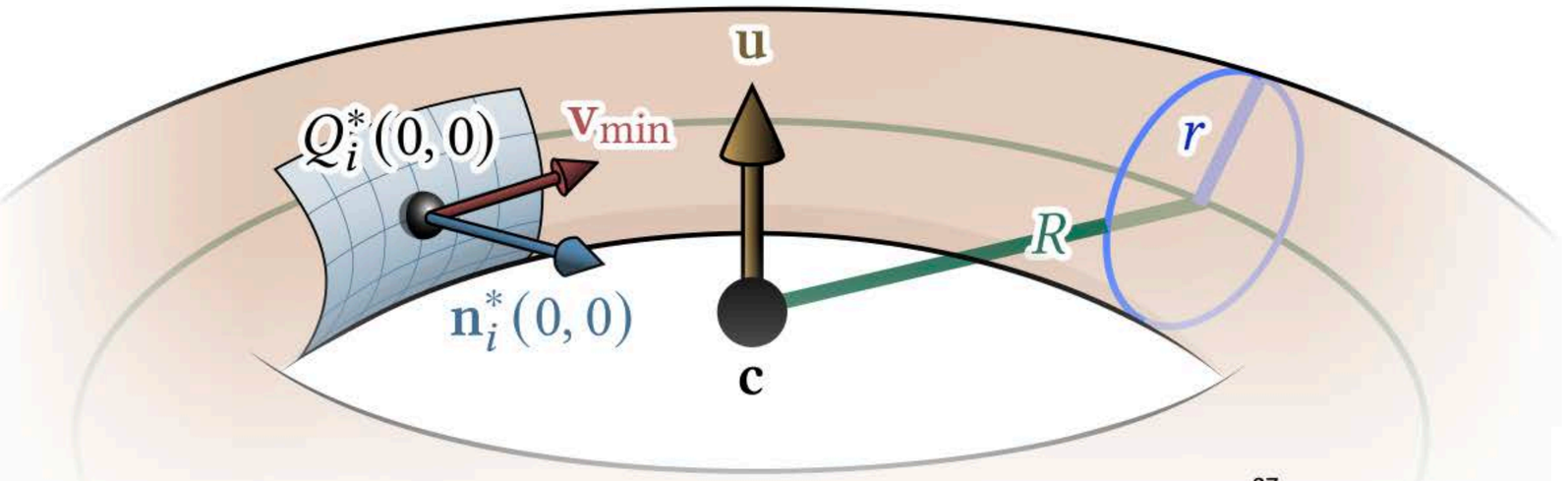


$$Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m$$

SDF of a torus:

$$\phi_{\mathbb{T}} = \|\mathbf{d}\| - r,$$

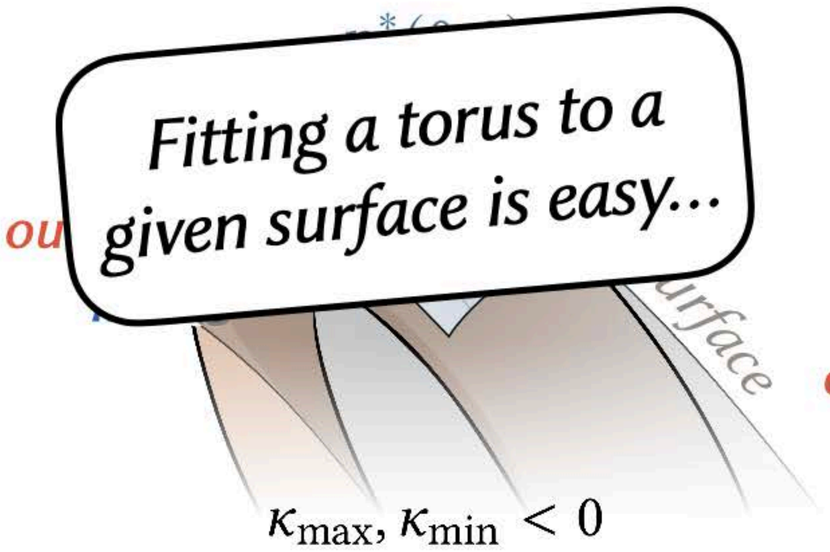
$$\mathbf{d} := (\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| - R, \langle \mathbf{x}-\mathbf{c}, \mathbf{u} \rangle)$$

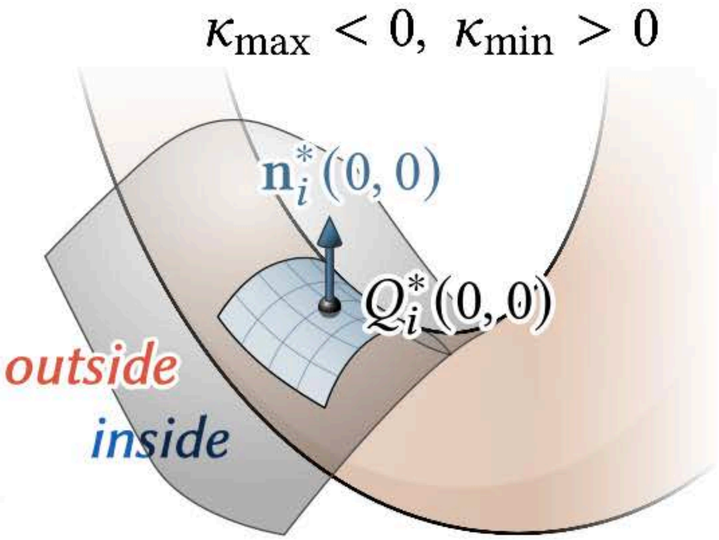


A torus is determined by the six coefficients

$a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$

The surface, one polynomial patch at a time



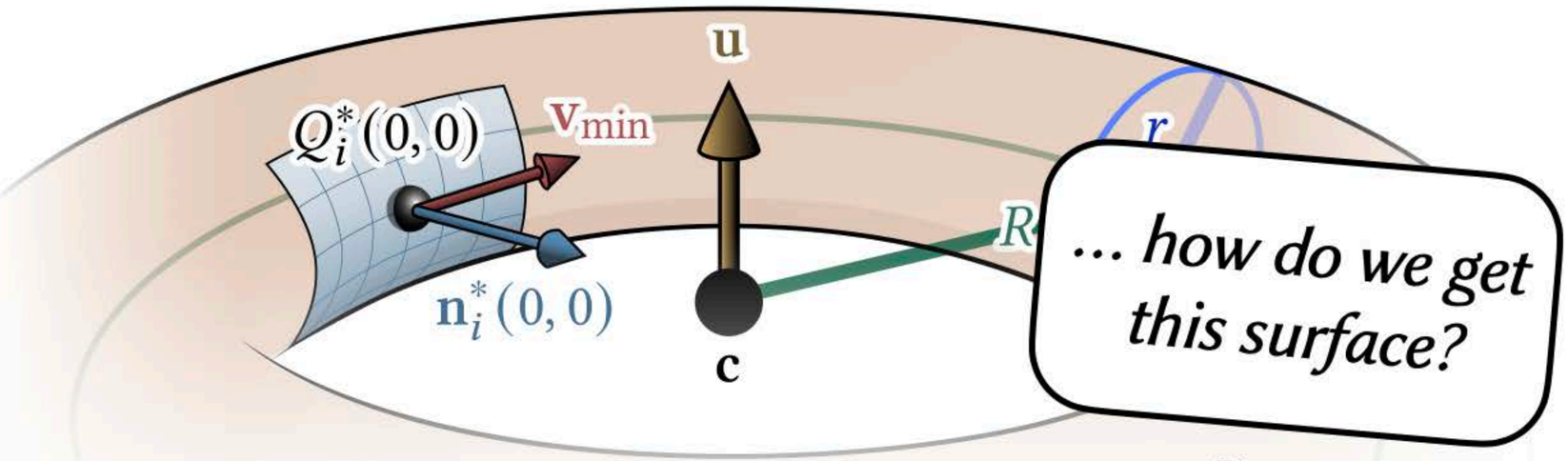


$$Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m$$

SDF of a torus:

$$\phi_{\mathbb{T}} = \|\mathbf{d}\| - r,$$

$$\mathbf{d} := (\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| - R, \langle \mathbf{x}-\mathbf{c}, \mathbf{u} \rangle)$$



A torus is determined by the six coefficients

$a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$

Inferring the surface, even locally, is hard

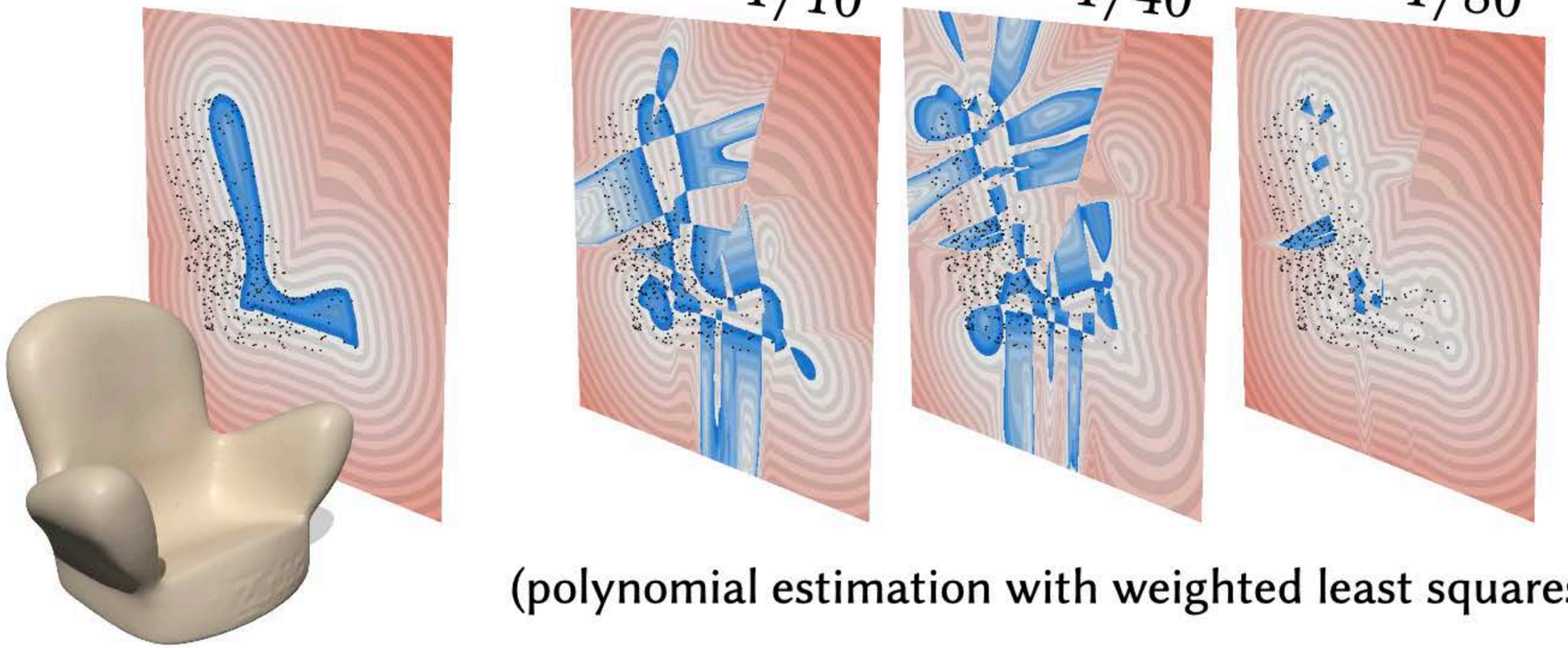
Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are dependent on brittle, domain-specific heuristics for neighborhood size and shape.

ground truth

neighborhood estimation with Gaussians

$\sigma = 1/10$ $\sigma = 1/40$ $\sigma = 1/80$



(polynomial estimation with weighted least squares)

Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are dependent on brittle, domain-specific heuristics for neighborhood size and shape.

A **huge** bag of possible approaches:

Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are dependent on brittle, domain-specific heuristics for neighborhood size and shape.

A **huge** bag of possible approaches:

- Endless kernel-based reconstruction methods (*MLS, partition-of-unity, meshless interpolation…*)

[Lancaster & Salkauskas 1981; Levin 1998; Turk 7 O'Brien 1999; Carr et al. 2001; Boissonnat & Cazals 2002; Amenta & Kil 2004; Shen et al. 2004; Dey & Sun 2005; Ohtake et al. 2005; Fleishman et al. 2005; Adamson & Alexa 2006; Kolluri 2008; Wang et al. 2008; Öztireli et al. 2009; Zagorchev & Goshtasby 2012; Fuhrmann & Goesele 2014; Wei et al. 2023…]

Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are dependent on brittle, domain-specific heuristics for neighborhood size and shape.

A **huge** bag of possible approaches:

- Endless kernel-based reconstruction methods (*MLS, partition-of-unity, meshless interpolation…*)
- Anisotropic kernels *(PCA, quadric error metric, etc.)* ⟶ *brittle*
- Hierarchy ⟶ *brittle*

[Lancaster & Salkauskas 1981; Levin 1998; Turk 7 O'Brien 1999; Carr et al. 2001; Boissonnat & Cazals 2002; Amenta & Kil 2004; Shen et al. 2004; Dey & Sun 2005; Ohtake et al. 2005; Fleishman et al. 2005; Adamson & Alexa 2006; Kolluri 2008; Wang et al. 2008; Öztireli et al. 2009; Zagorchev & Goshtasby 2012; Fuhrmann & Goesele 2014; Wei et al. 2023…]

Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are dependent on brittle, domain-specific heuristics for neighborhood size and shape.

A **huge** bag of possible approaches:

- Endless kernel-based reconstruction methods (*MLS, partition-of-unity, meshless interpolation…*)
- Anisotropic kernels *(PCA, quadric error metric, etc.)* ⟶ *brittle*
- Hierarchy ⟶ *brittle*
- Iterative/nonlinear solvers ⟶ *slow*
- Robust statistics ⟶ *slow*

[Lancaster & Salkauskas 1981; Levin 1998; Turk 7 O'Brien 1999; Carr et al. 2001; Boissonnat & Cazals 2002; Amenta & Kil 2004; Shen et al. 2004; Dey & Sun 2005; Ohtake et al. 2005; Fleishman et al. 2005; Adamson & Alexa 2006; Kolluri 2008; Wang et al. 2008; Öztireli et al. 2009; Zagorchev & Goshtasby 2012; Fuhrmann & Goesele 2014; Wei et al. 2023…]

Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are dependent on brittle, domain-specific heuristics for neighborhood size and shape.

A **huge** bag of possible approaches:

- Endless kernel-based reconstruction methods (*MLS, partition-of-unity, meshless interpolation…*)
- Anisotropic kernels *(PCA, quadric error metric, etc.)* ⟶ *brittle*
- Hierarchy ⟶ *brittle*
- Iterative/nonlinear solvers ⟶ *slow*
- Robust statistics ⟶ *slow*
- *… any one method is unlikely to generalize!*

[Lancaster & Salkauskas 1981; Levin 1998; Turk 7 O'Brien 1999; Carr et al. 2001; Boissonnat & Cazals 2002; Amenta & Kil 2004; Shen et al. 2004; Dey & Sun 2005; Ohtake et al. 2005; Fleishman et al. 2005; Adamson & Alexa 2006; Kolluri 2008; Wang et al. 2008; Öztireli et al. 2009; Zagorchev & Goshtasby 2012; Fuhrmann & Goesele 2014; Wei et al. 2023…]

Inferring the surface, even locally, is hard

Classic methods like *jet-fitting, moving least squares* are extremely dependent on brittle, domain-specific heuristics for neighborhood size and shape.

A **huge** bag of possible approaches:

- Endless kernel-based reconstruction methods *(MLS, partition of unity, meshless methods, etc.)*
- Anisotropic kernels *(PCA, quadric error metric, etc.)* ⟶ *brittle*
- Hierarchy ⟶ *brittle*
- Iterative/nonlinear solvers ⟶ *slow*
- Robust statistics ⟶ *slow*
- ...

Just use a *fixed-size* neighborhood with a *data-driven* approach.

Inferring the surface, even locally, is hard

Just use a *fixed-size* neighborhood with a *data-driven* approach.

local function

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

- Dominated by local surface behavior

Inferring the surface, even locally, is hard

Just use a *fixed-size* neighborhood with a *data-driven* approach.

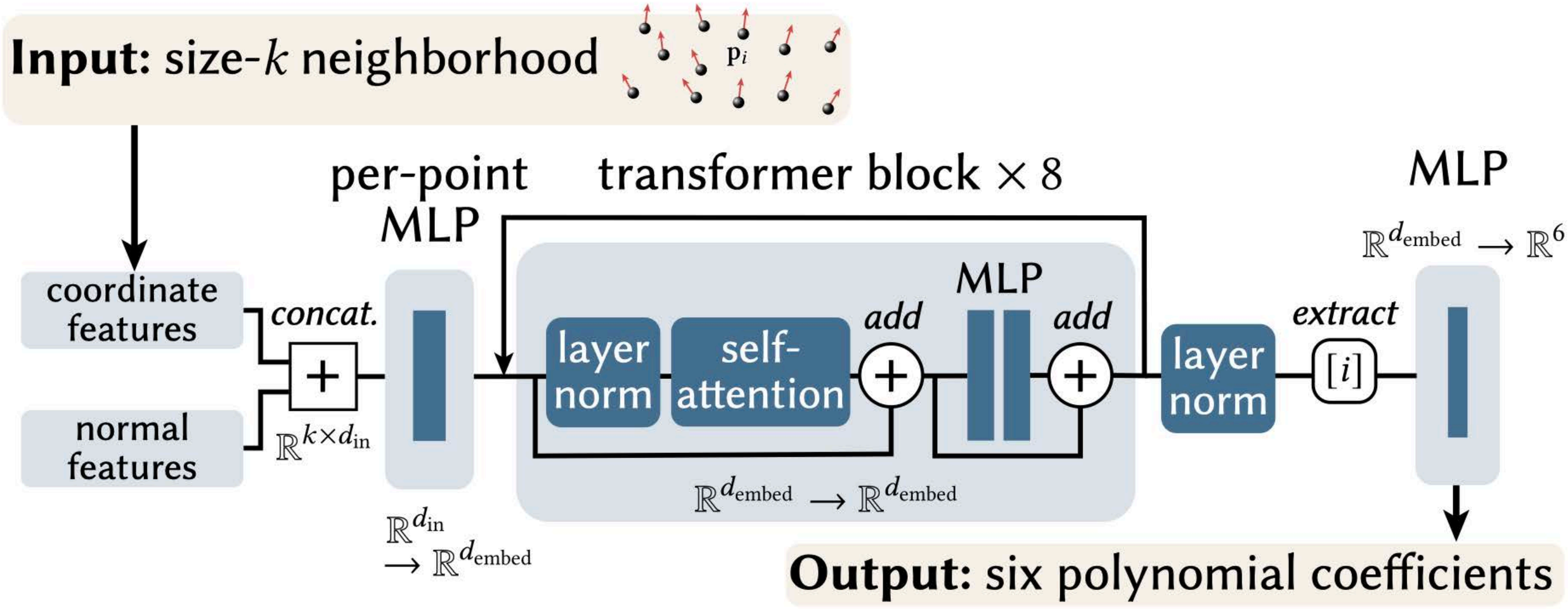
local function

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

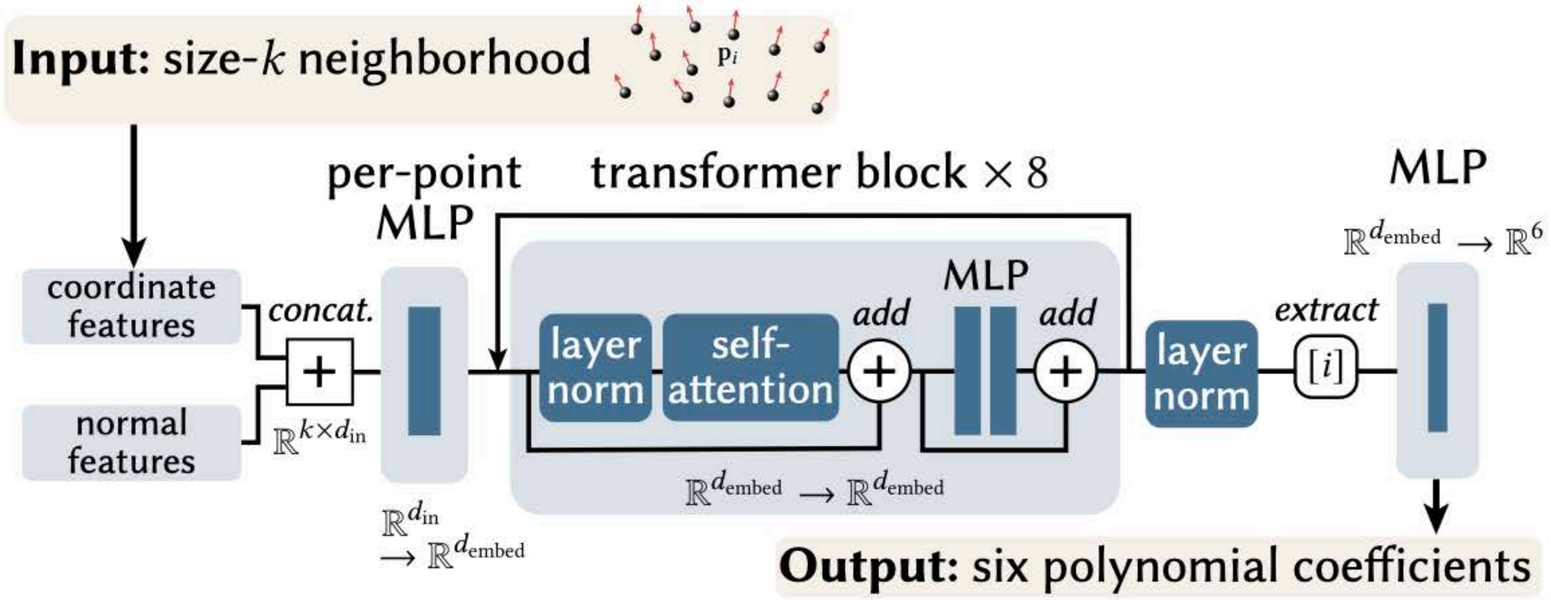
- Dominated by local surface behavior
- Easy to generate data for small surface patches

Network architecture

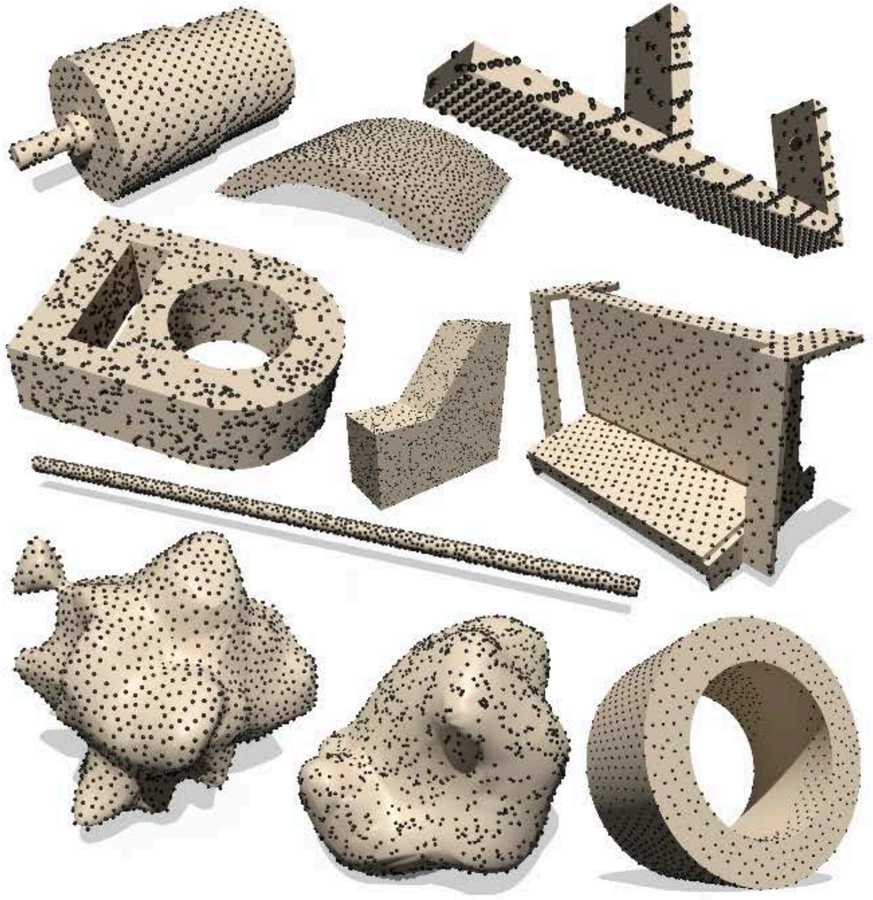
Network architecture



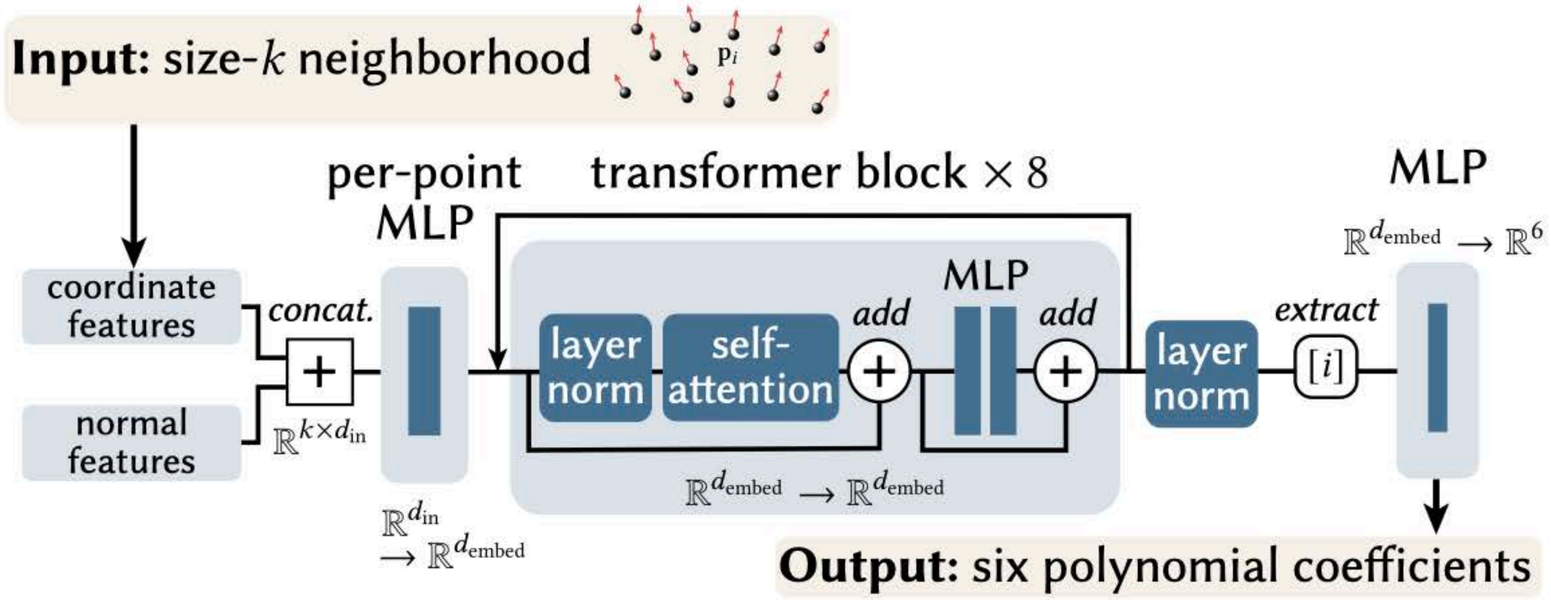
Network architecture



Training data

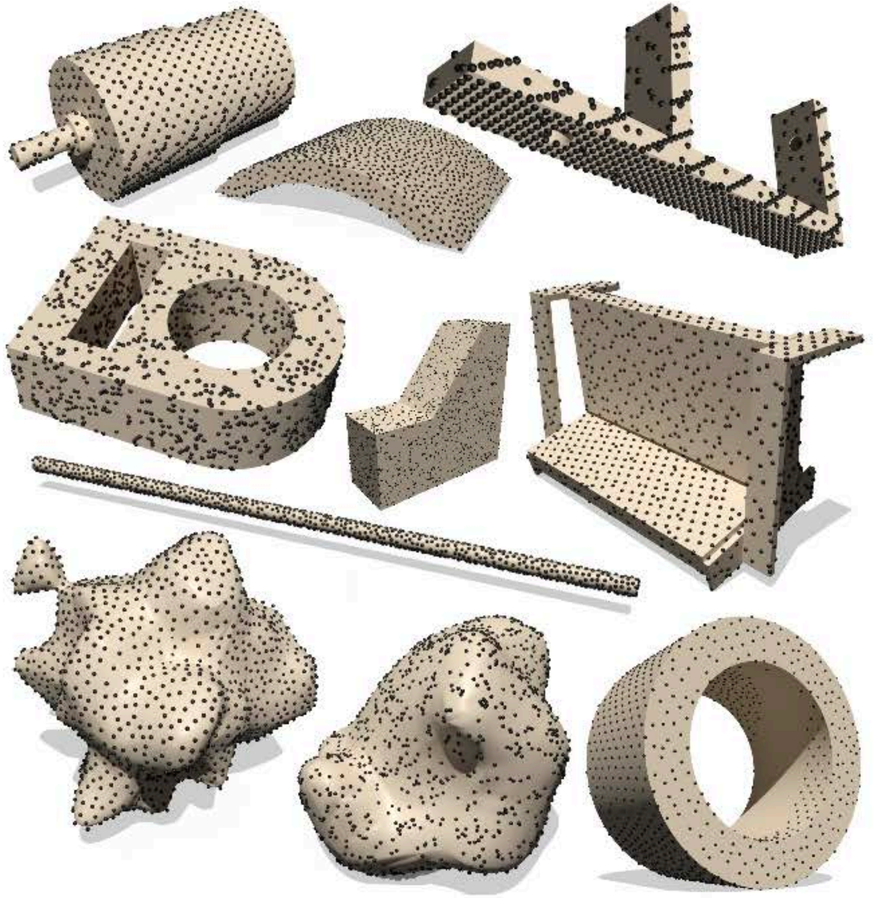


Network architecture

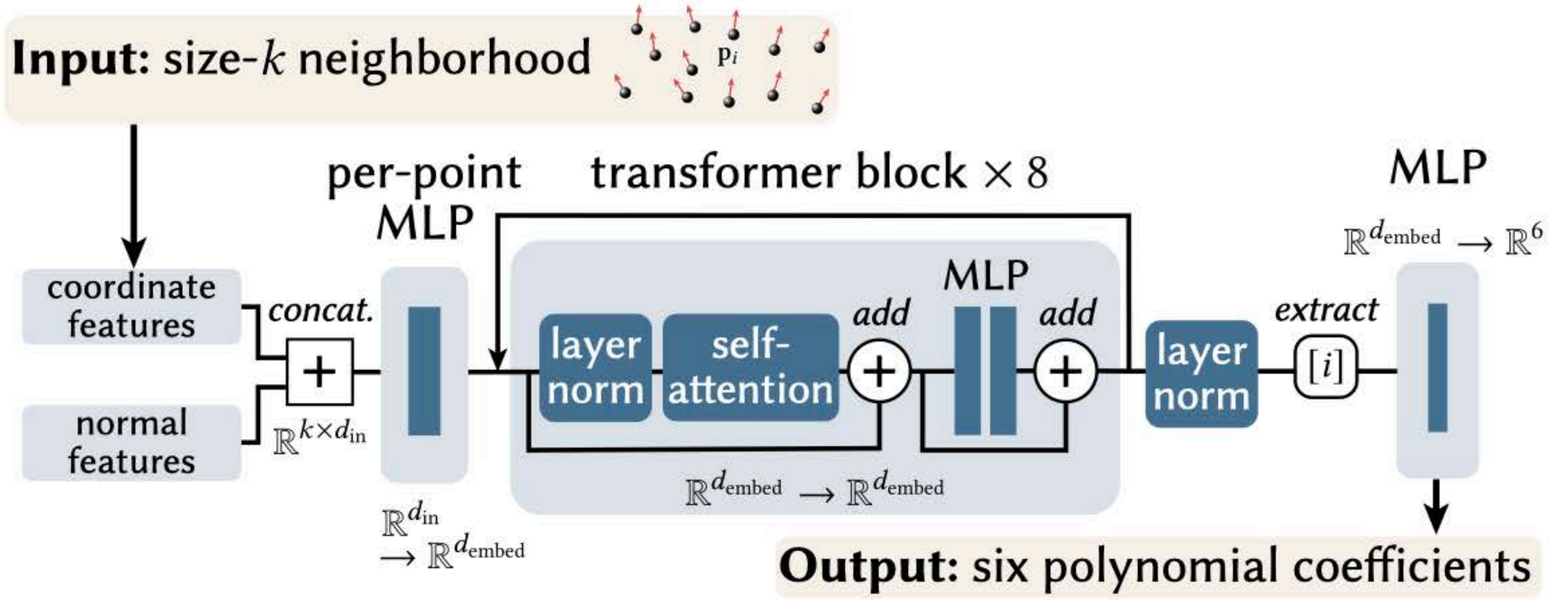


- 10,000 models from ABC dataset [Koch et al. 2019], 2000 procedural

Training data

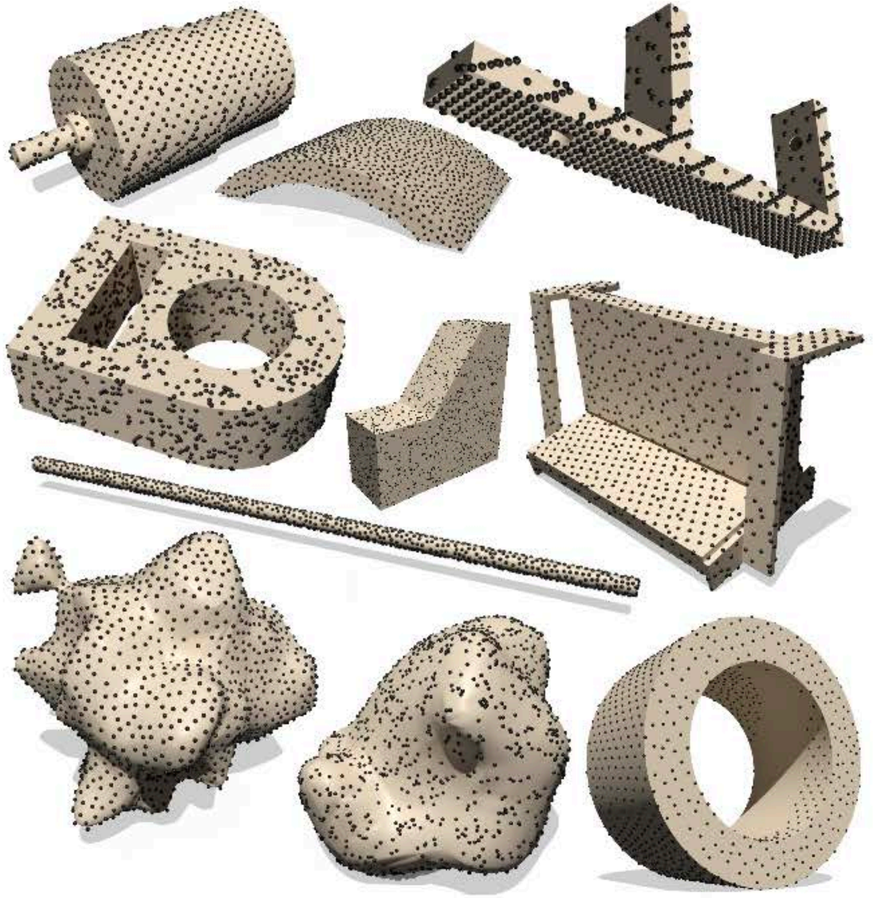


Network architecture

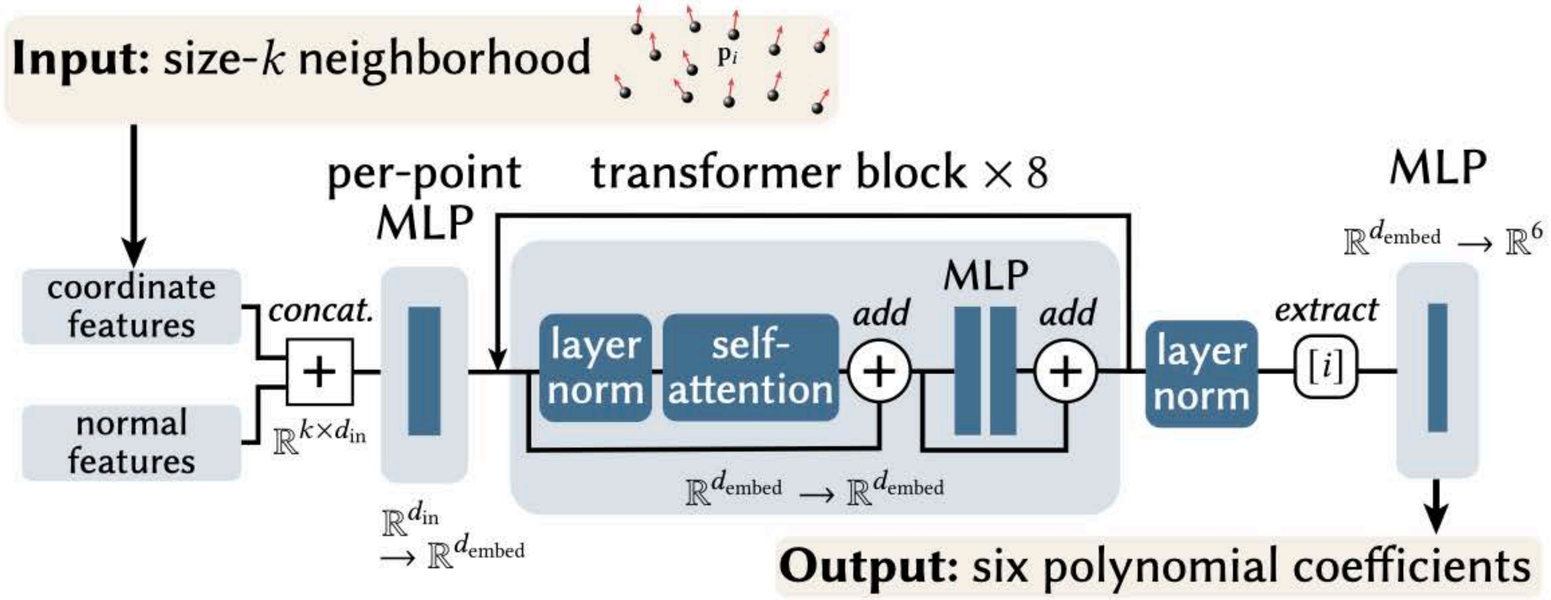


- 10,000 models from ABC dataset [Koch et al. 2019], 2000 procedural
- Normalize and sample into size-2048 point clouds:
  - Uniform sampling
  - Farthest-point sampling
  - Simulated scanning

Training data

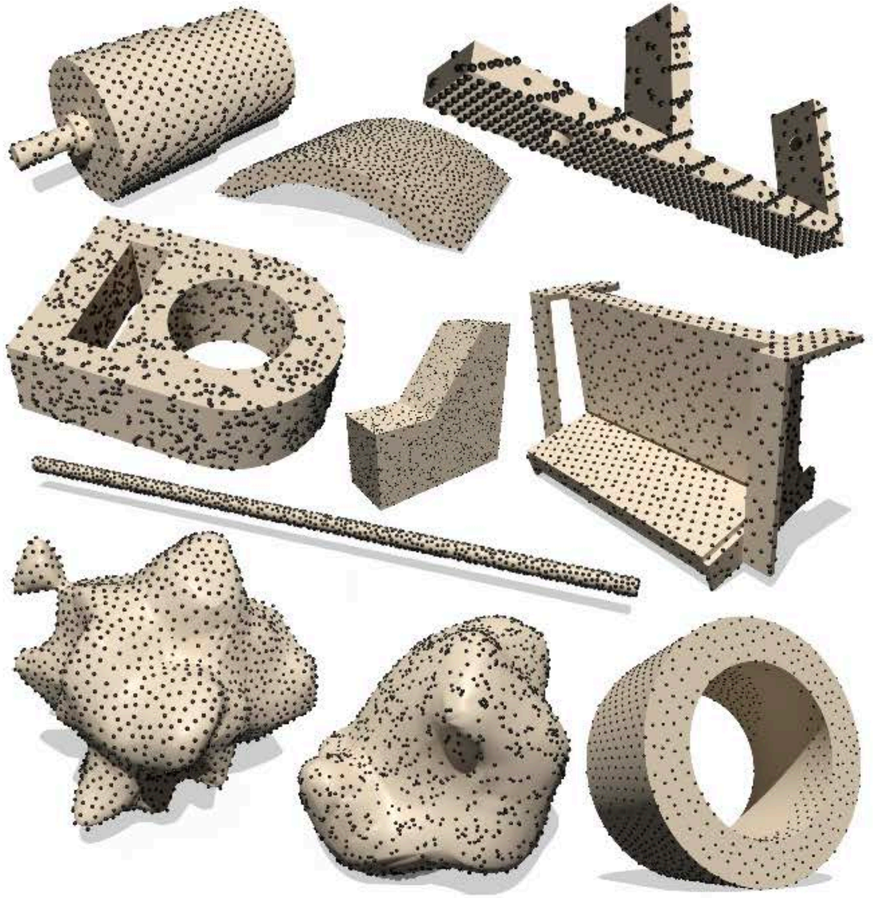


Network architecture

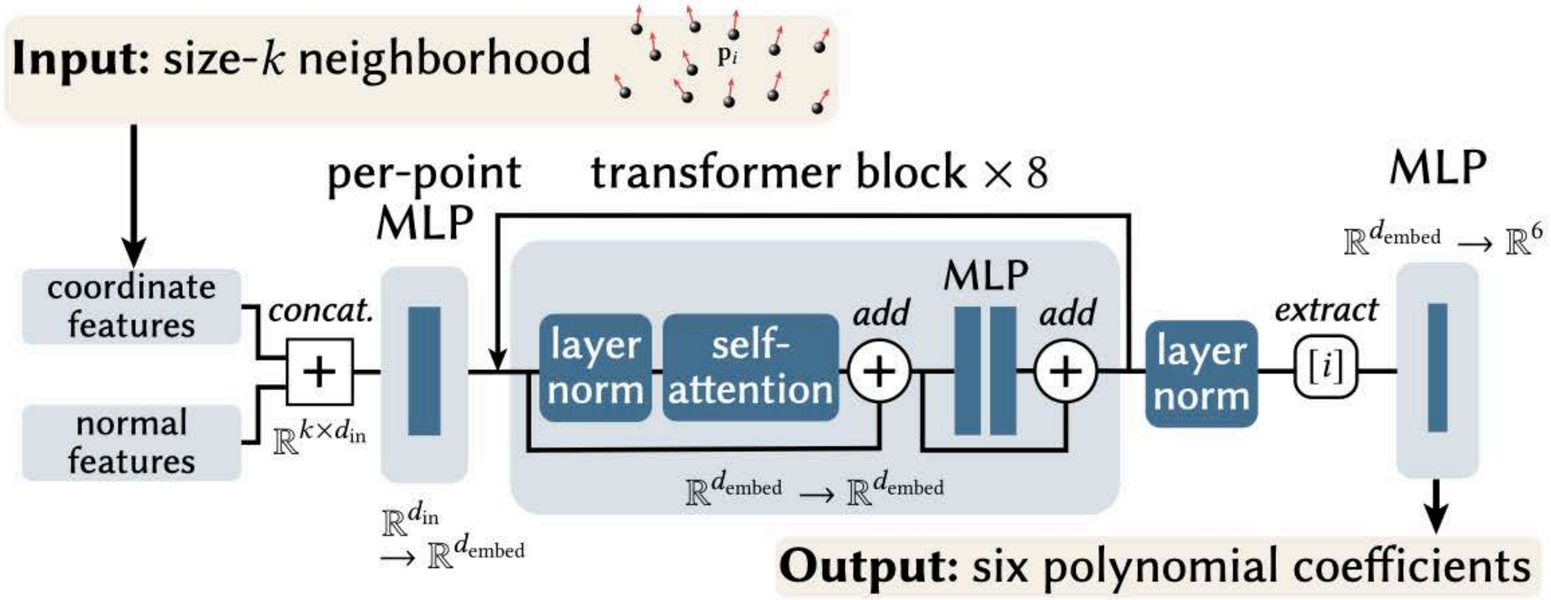


- 10,000 models from ABC dataset [Koch et al. 2019], 2000 procedural
- Normalize and sample into size-2048 point clouds:
  - Uniform sampling
  - Farthest-point sampling
  - Simulated scanning
- Input neighborhood of size $k = 64$
- >40 million training examples

Training data

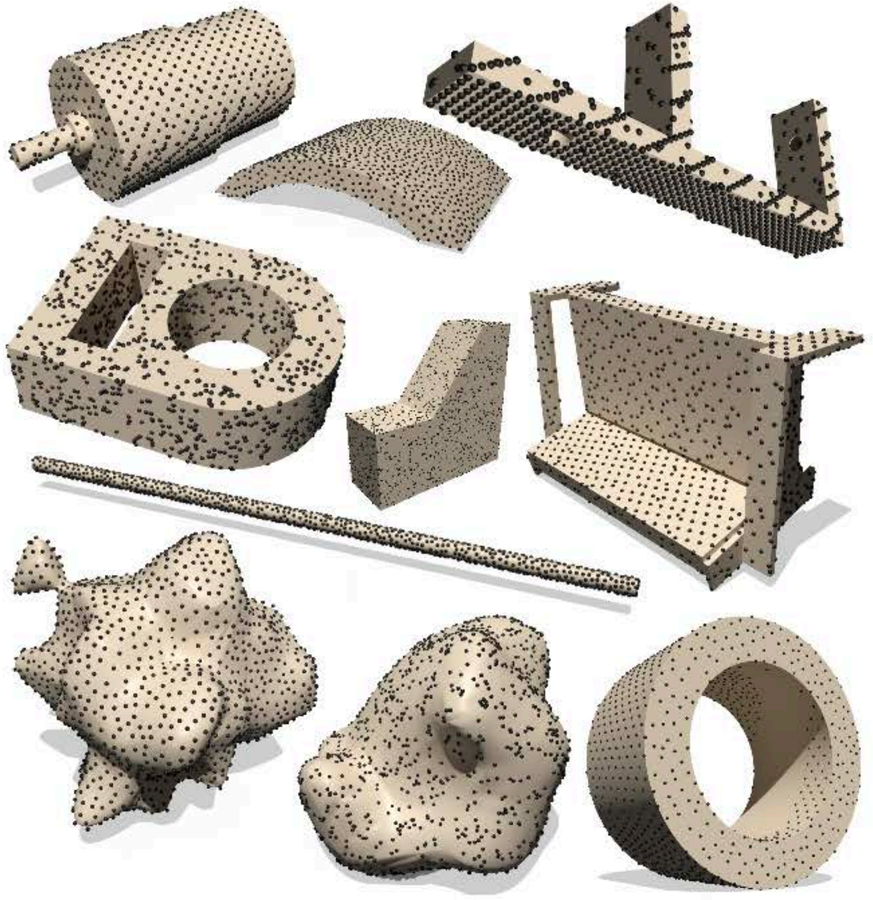


Network architecture

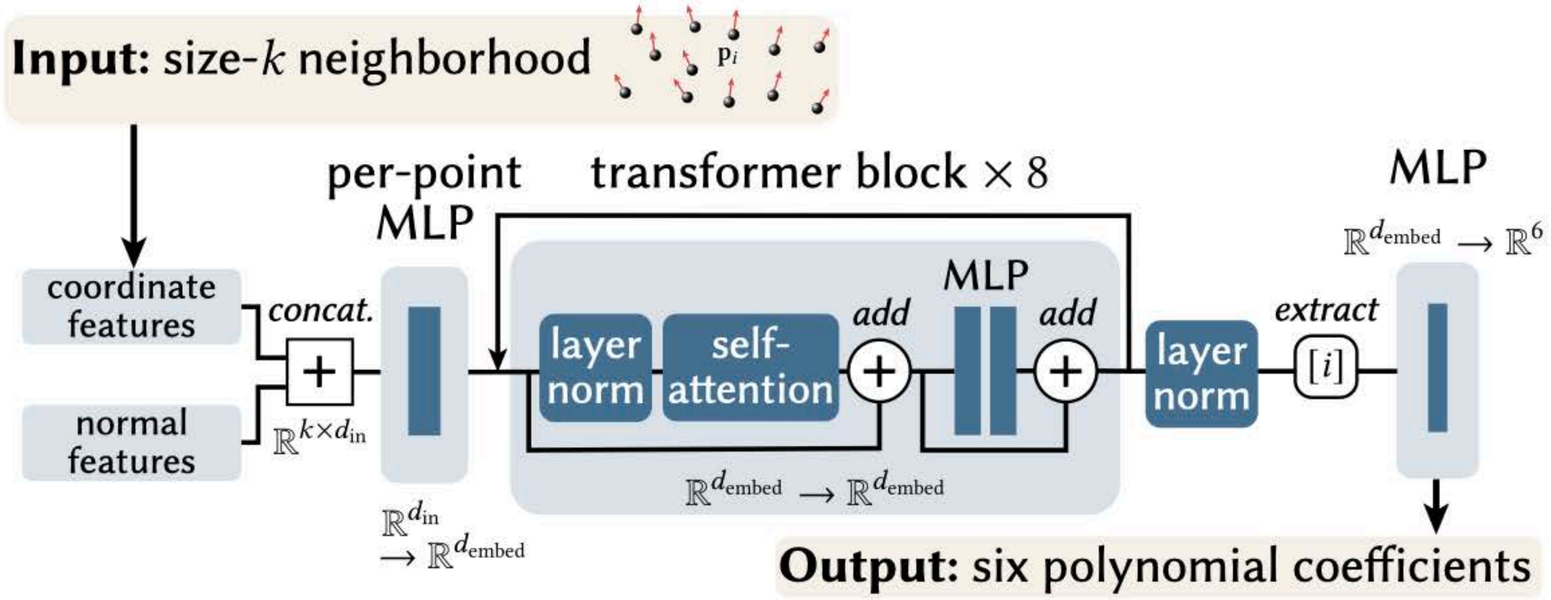


Loss function

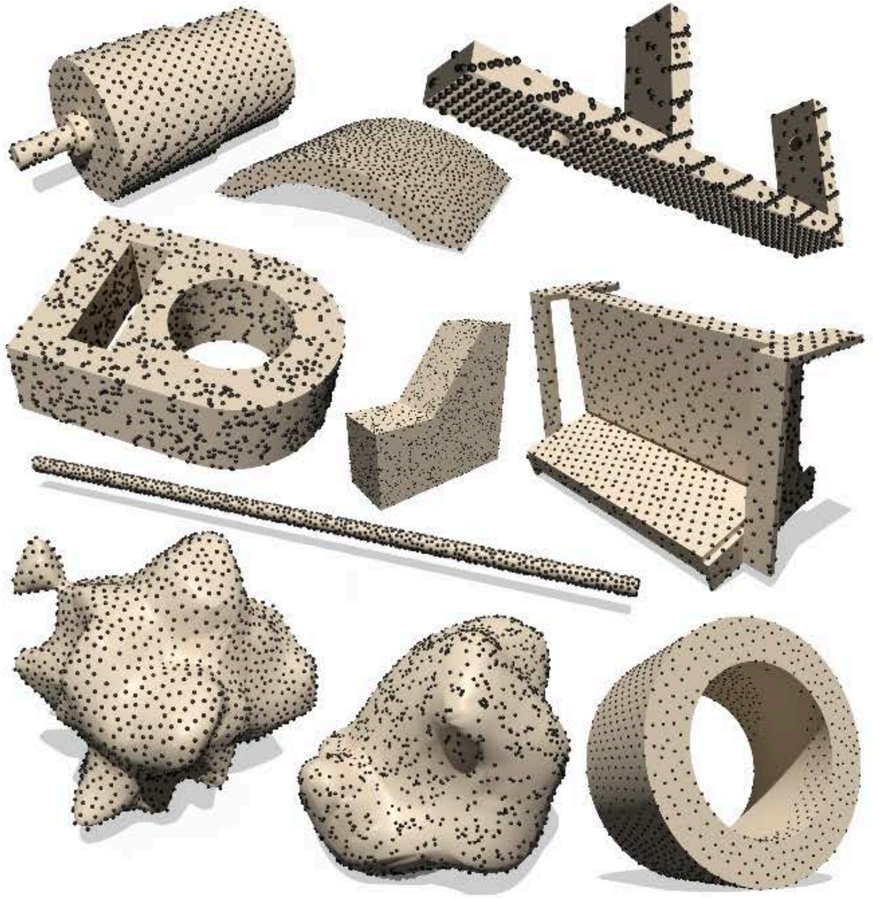
Training data



Network architecture



Training data

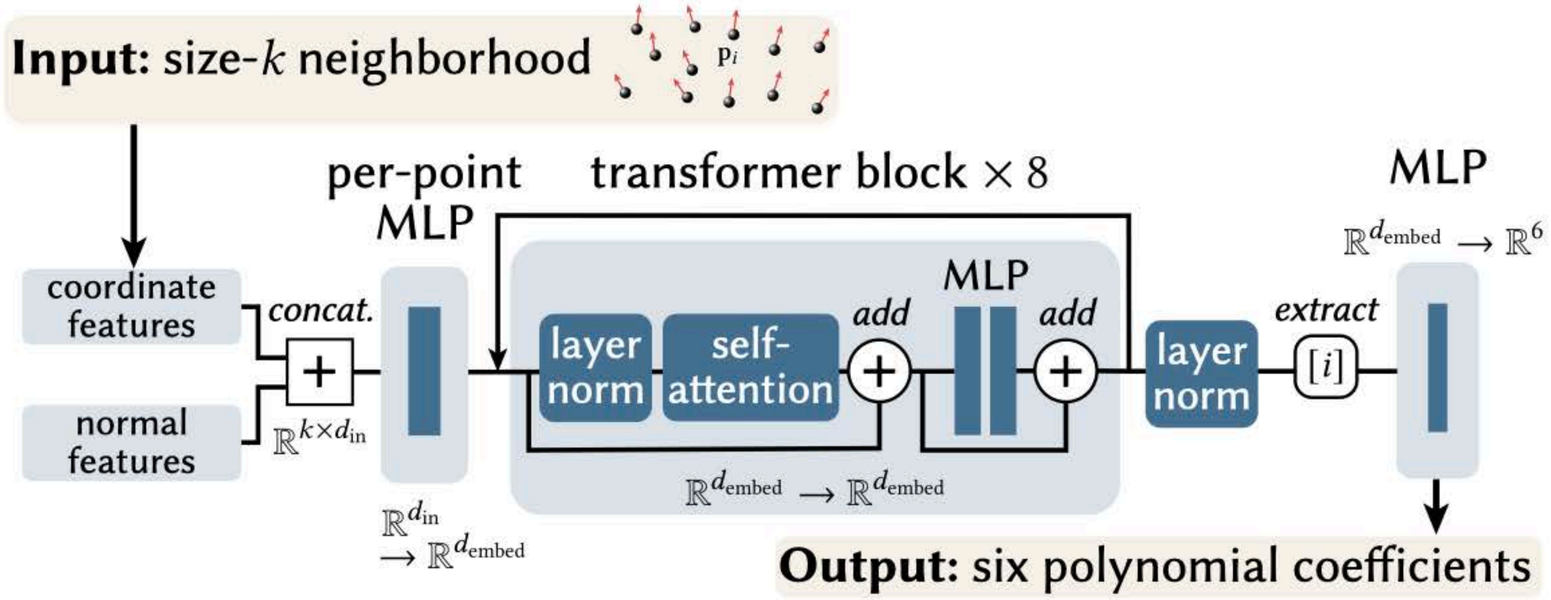


Loss function

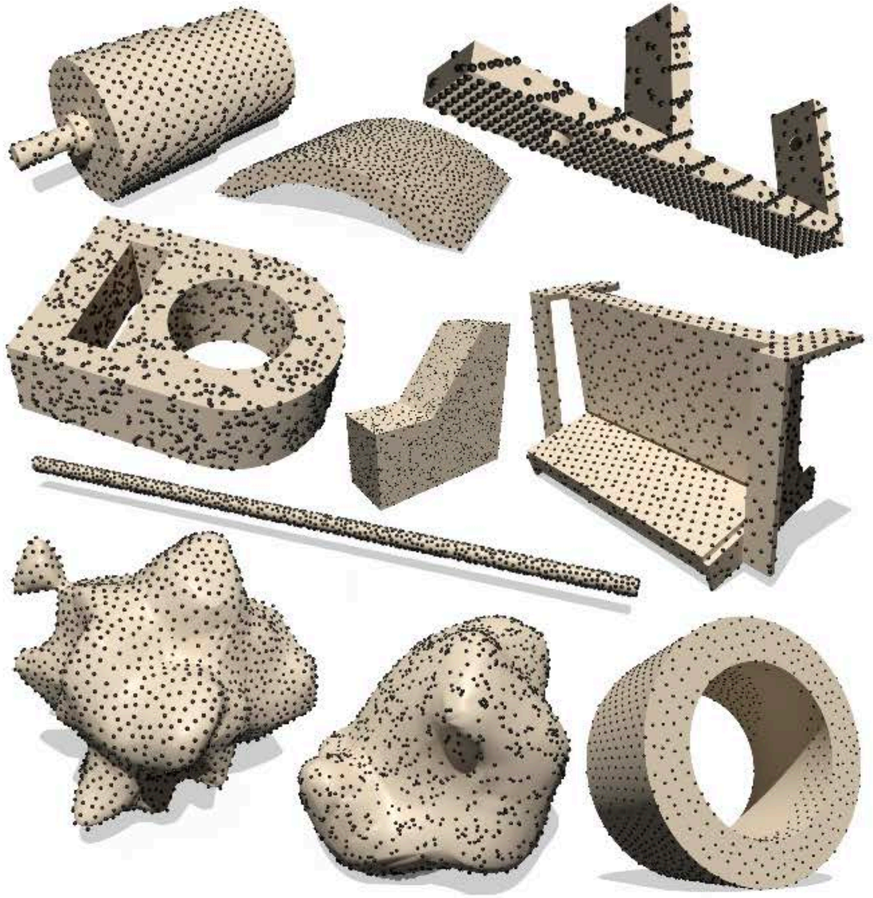
$$\mathcal{L} := \mathcal{L}_{\text{distance}}$$

$$\text{where } \mathcal{L}_{\text{distance}} := \frac{1}{Q} \sum_{j=1}^{Q} \left| \phi(\mathbf{q}_j) - \phi_{\text{true}}(\mathbf{q}_i) \right|$$

Network architecture



Training data



Loss function

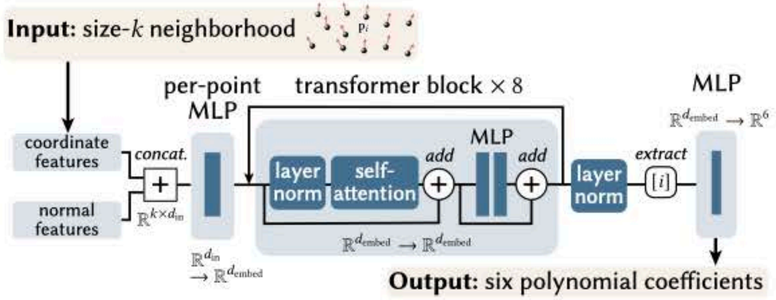
$$\mathcal{L} \coloneqq \mathcal{L}_{\text{distance}} + \mathcal{L}_{\text{blend}}$$

$$\text{where } \mathcal{L}_{\text{distance}} \coloneqq \frac{1}{Q} \sum_{j=1}^{Q} \left| \phi(\mathbf{q}_j) - \phi_{\text{true}}(\mathbf{q}_i) \right|$$

$$\mathcal{L}_{\text{blend}} \coloneqq \frac{1}{Q} \sum_{j=1}^{Q} \left| 1 - \|\nabla \phi(\mathbf{q}_j)\| \right|$$

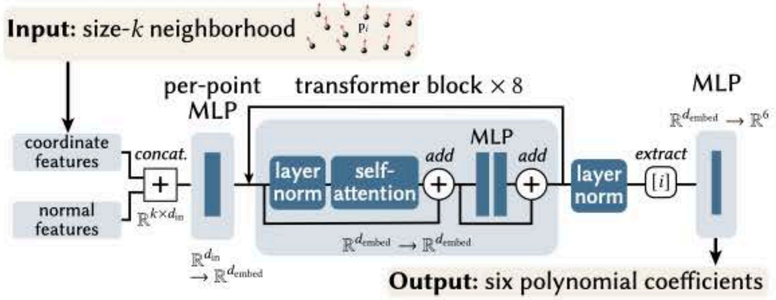
The *points as tori (PAT)* algorithm

Pre-training



The *points as tori (PAT)* algorithm

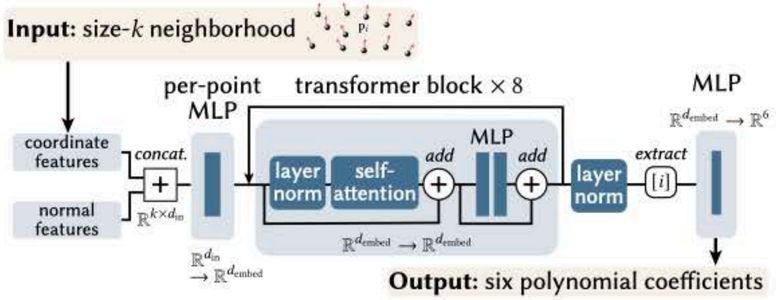
Pre-training



~45 hours on a single NVIDIA RTX 3090

The *points as tori (PAT)* algorithm

Pre-training

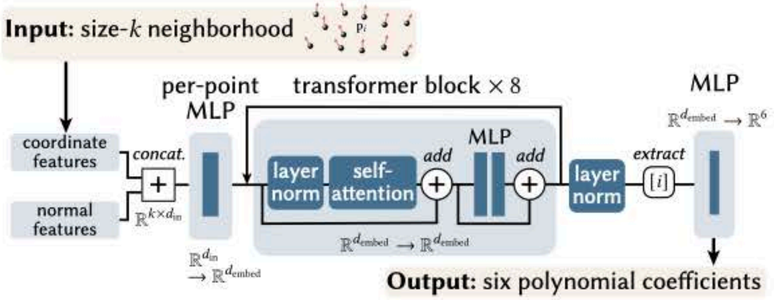


~45 hours on a single NVIDIA RTX 3090

Given an input point cloud $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$ ...

The *points as tori (PAT)* algorithm

Pre-training



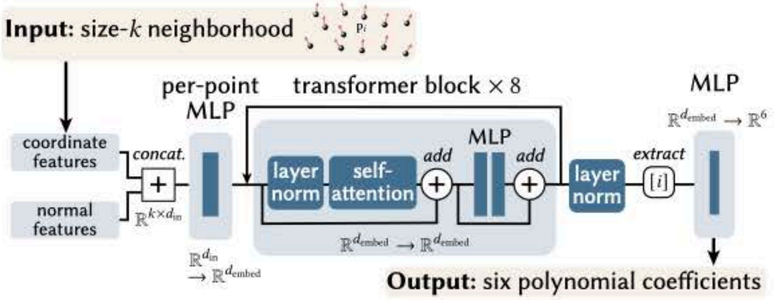
~45 hours on a single NVIDIA RTX 3090

Given an input point cloud $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$ ...

One-time precompute:
Apply pre-trained network to each point neighborhood (parallelizable)

The *points as tori (PAT)* algorithm

Pre-training



~45 hours on a single NVIDIA RTX 3090

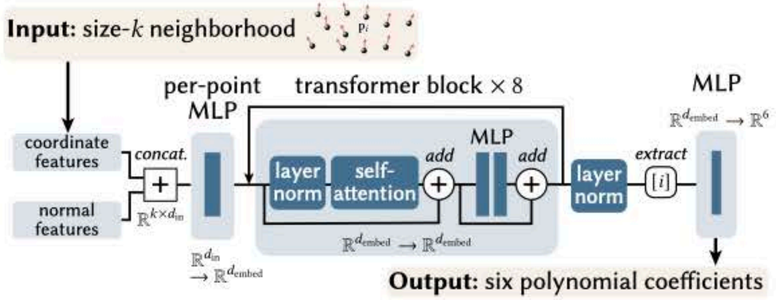
Given an input point cloud $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$ ...

One-time precompute:
Apply pre-trained network to each point neighborhood (parallelizable)

Inference time:
Evaluate the SDF at $\mathbf{x} \in \mathbb{R}^3$ as $\phi(\mathbf{x}) = \dfrac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}$

The *points as tori (PAT)* algorithm

Pre-training



~45 hours on a single NVIDIA RTX 3090

Given an input point cloud $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$ *...*

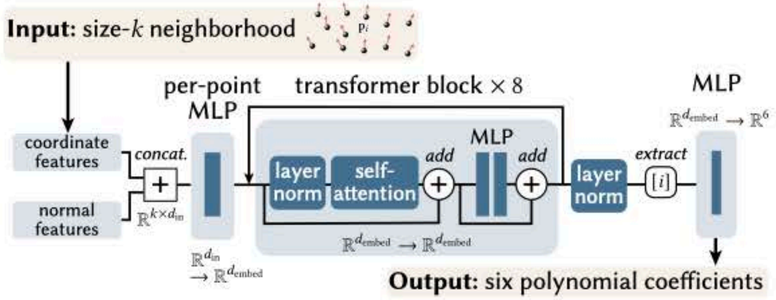
One-time precompute:
Apply pre-trained network to each point neighborhood (parallelizable)

Inference time:
Evaluate the SDF at $\mathbf{x} \in \mathbb{R}^3$ as $\phi(\mathbf{x}) = \dfrac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}$

10^{-4} —10^{-3} seconds for a single query to 1M-10M+ points

The *points as tori (PAT)* algorithm

Pre-training



~45 hours on a single NVIDIA RTX 3090

Given an input point cloud $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$ ...

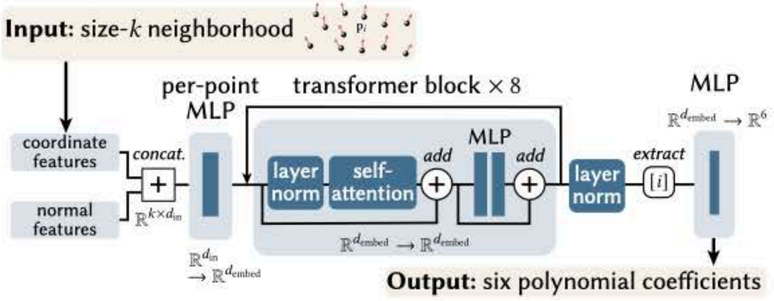
One-time precompute:
Apply pre-trained network to each point neighborhood (parallelizable)

Inference time:
Evaluate the SDF at $\mathbf{x} \in \mathbb{R}^3$ as $\phi(\mathbf{x}) = \dfrac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}$

10^{-4} —10^{-3} seconds for a single query to 1M-10M+ points
multiple queries trivially parallelizable

The *points as tori (PAT)* algorithm

Pre-training



~45 hours on a single NVIDIA RTX 3090

Given an input point cloud $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$ ...

One-time precompute:
Apply pre-trained network to each point neighborhood (parallelizable)

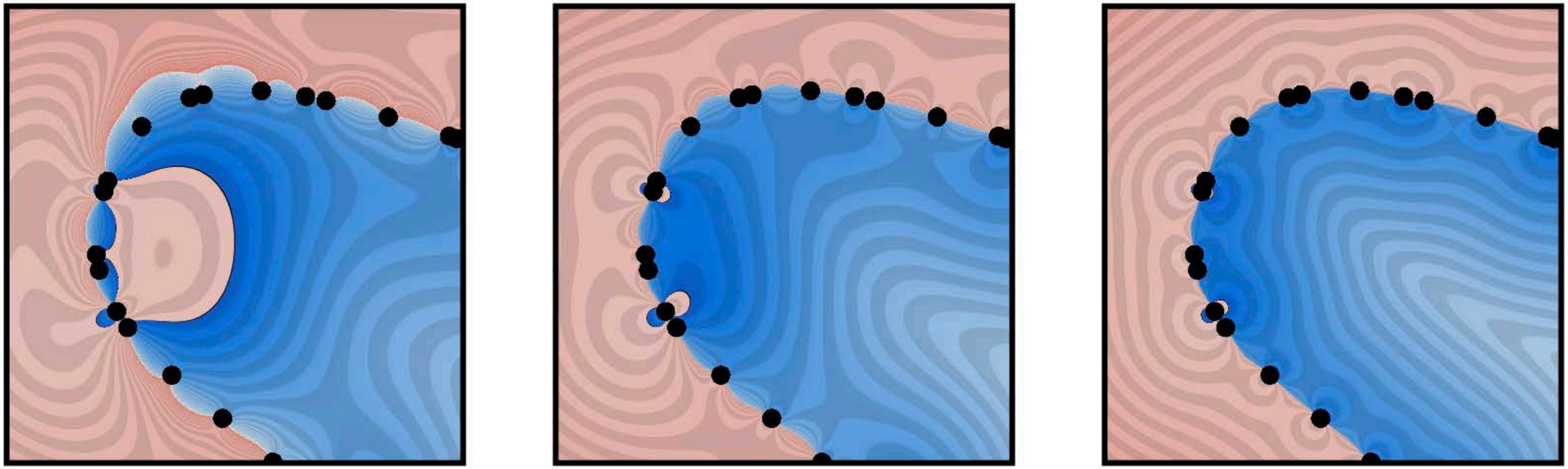
Inference time:
Evaluate the SDF at $\mathbf{x} \in \mathbb{R}^3$ as $\phi(\mathbf{x}) = \dfrac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)}$

simple formula for λ

10^{-4} —10^{-3} seconds for a single query to 1M-10M+ points
multiple queries trivially parallelizable

Better than naive convolutional distance

smaller λ ⟵ ⟶ *larger* λ



worse eikonality
underfit level sets

better eikonality
overfit level sets

Better than naive convolutional distance

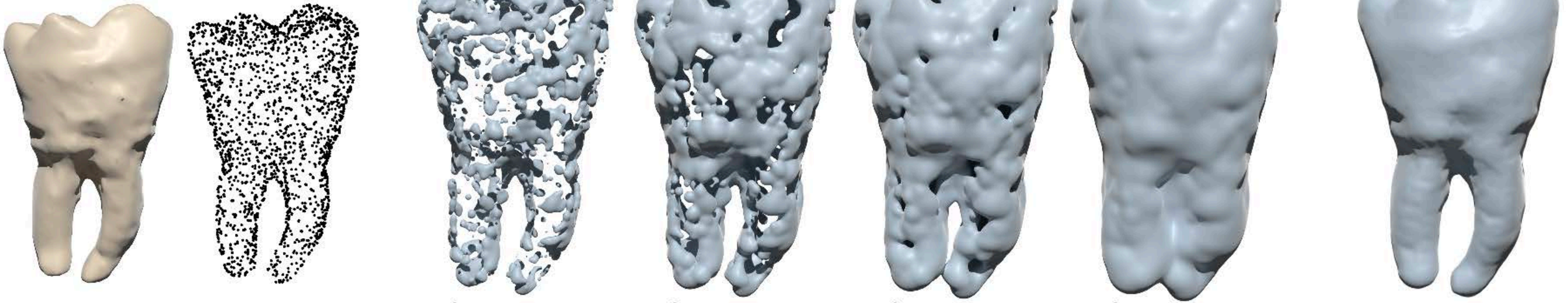
naive "LogSumExp" distances

[Madan & Levin 2022]

ground truth

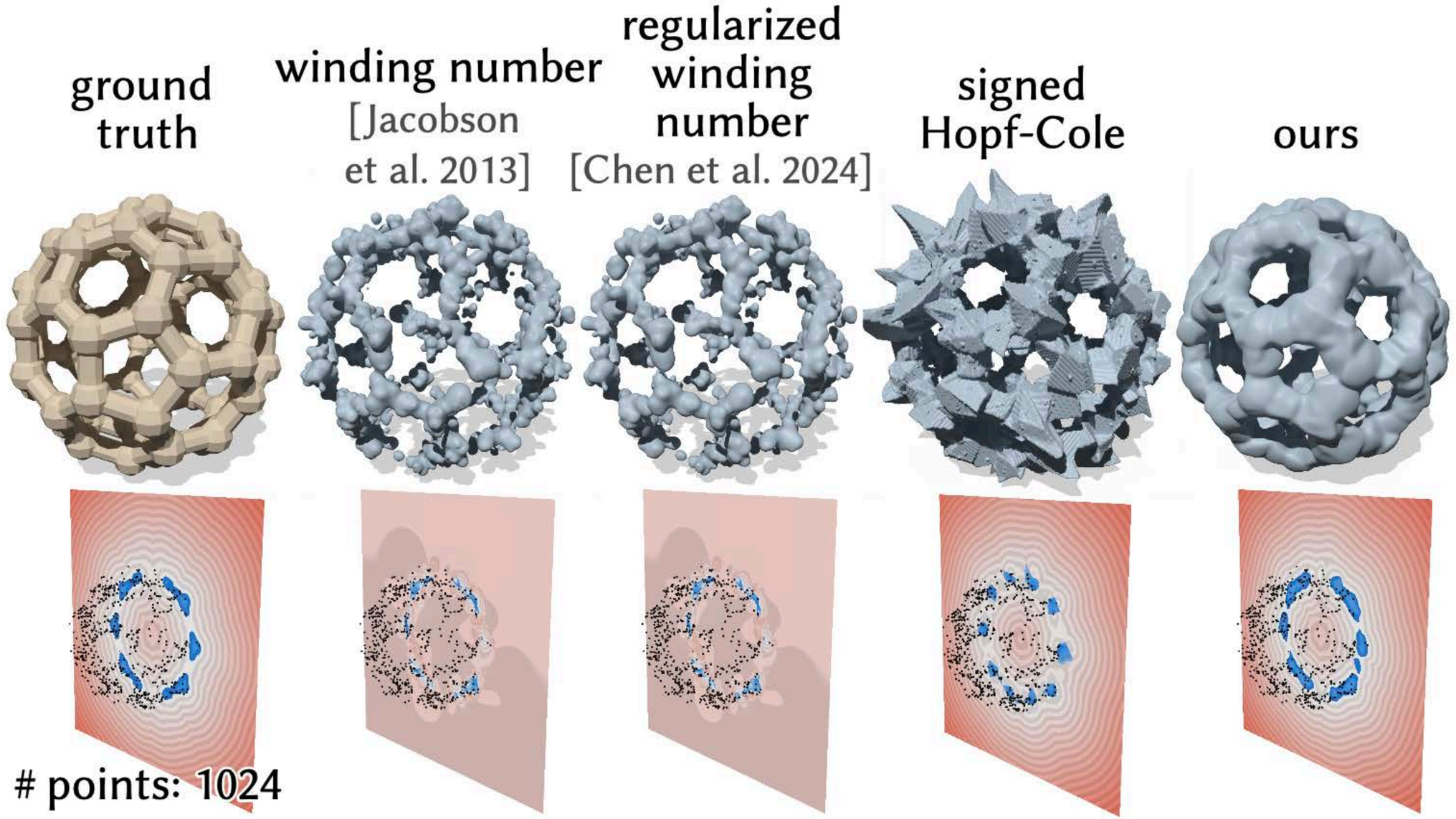
input

ours, no tuning

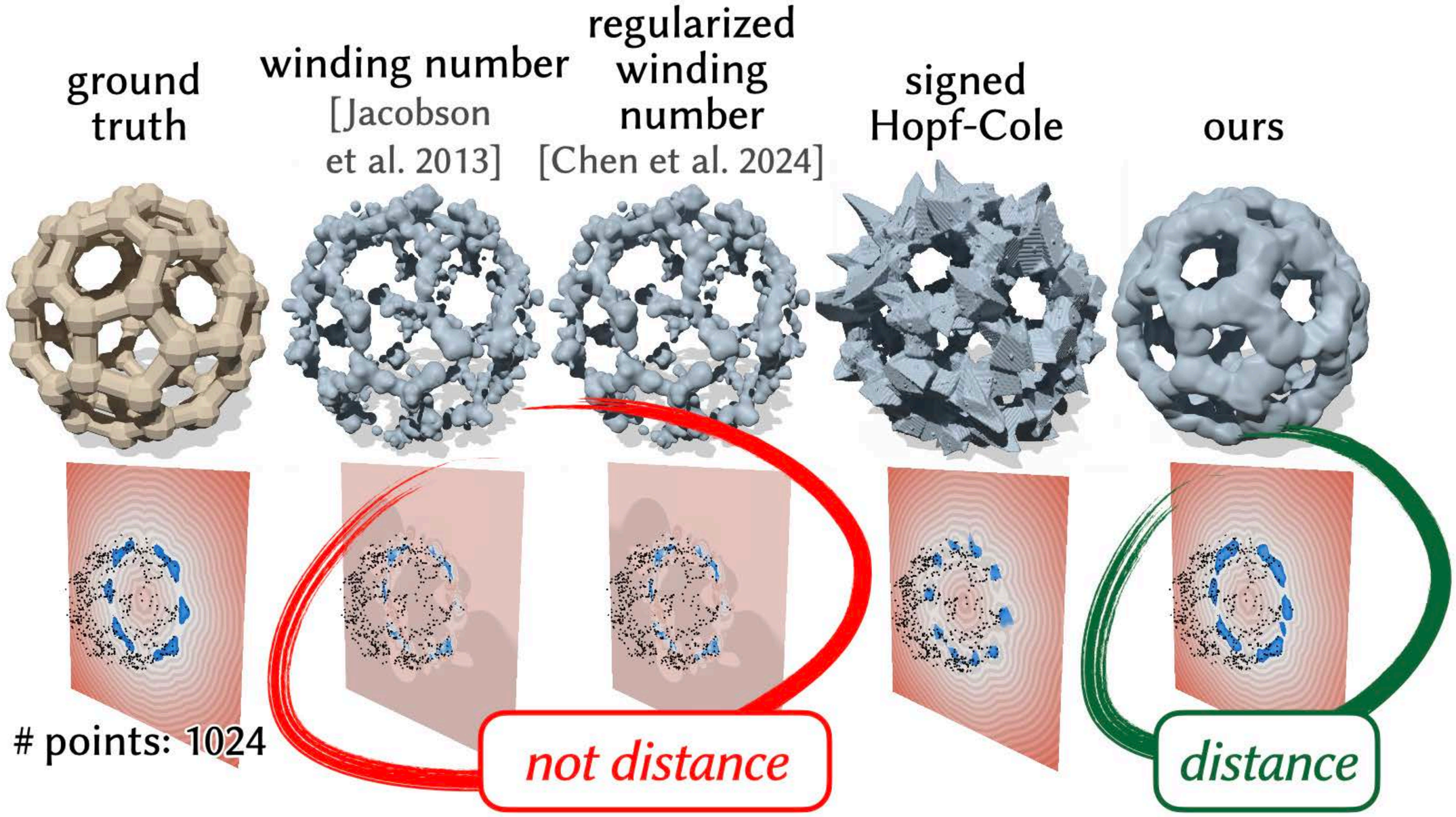


$\lambda = 50$ $\lambda = 40$ $\lambda = 35$ $\lambda = 30$

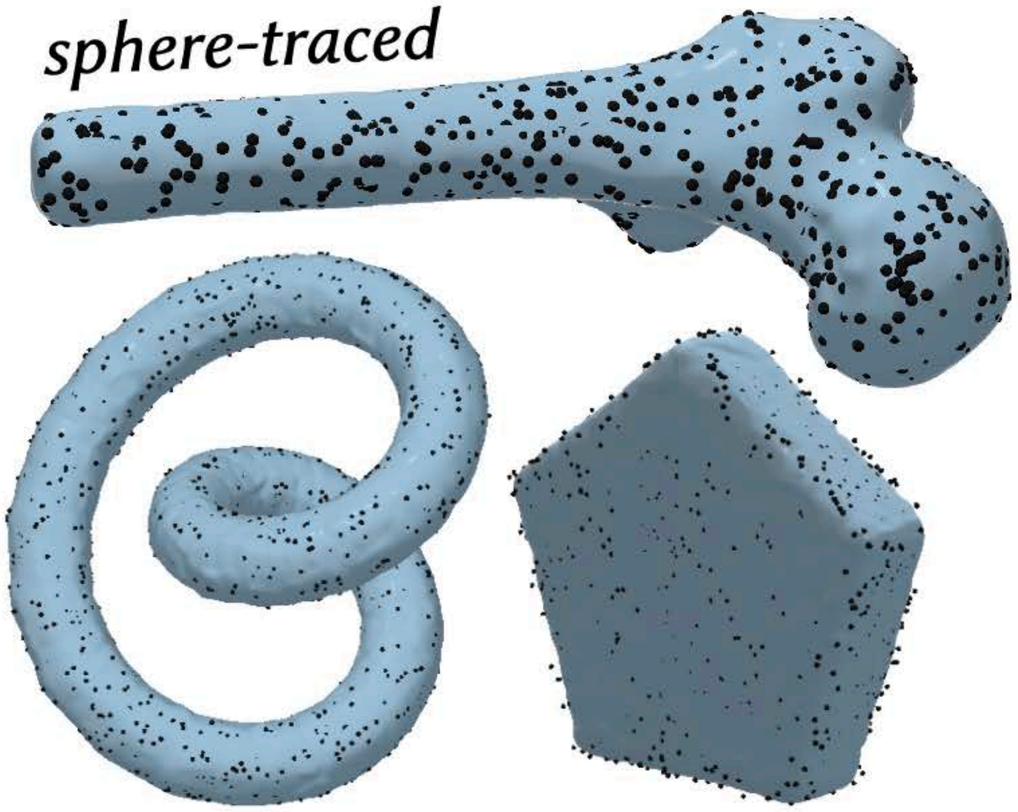
Better than winding number, and almost as fast



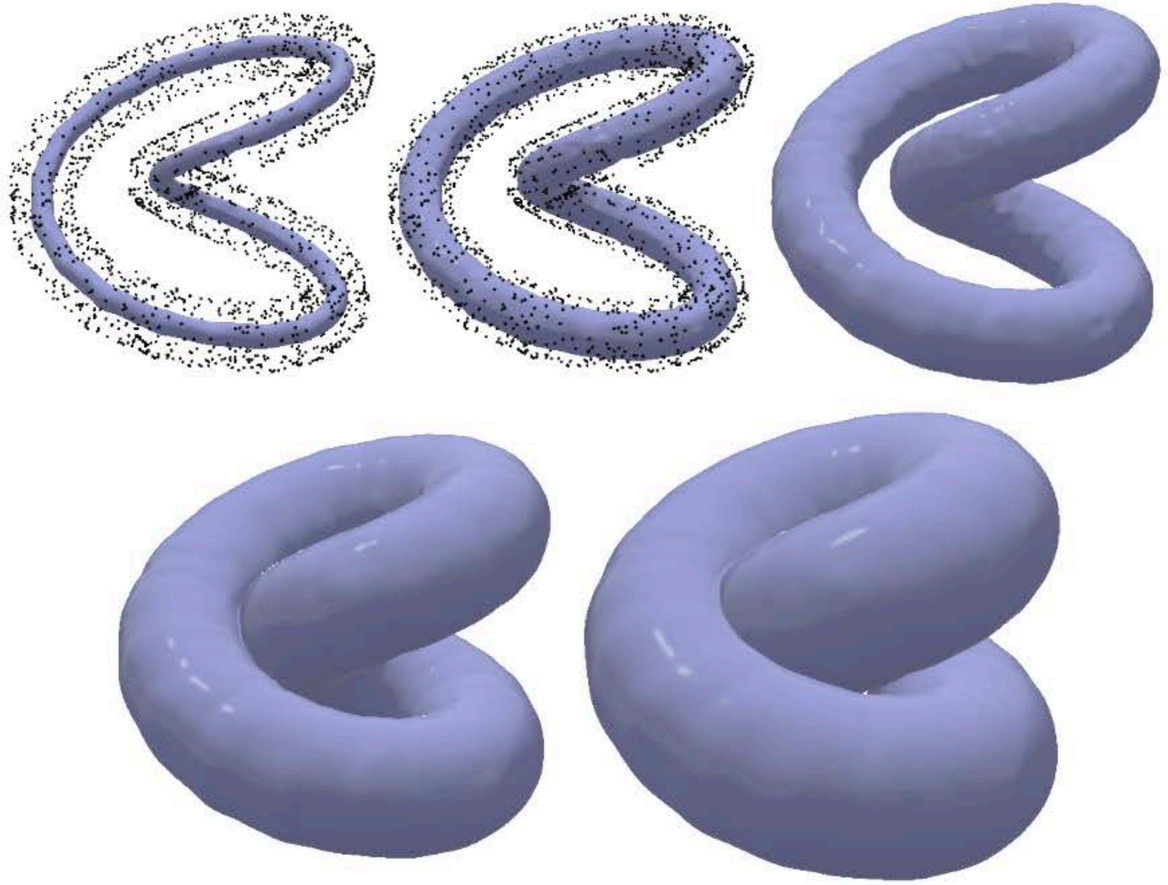
Better than winding number, and almost as fast



Pointwise evaluation

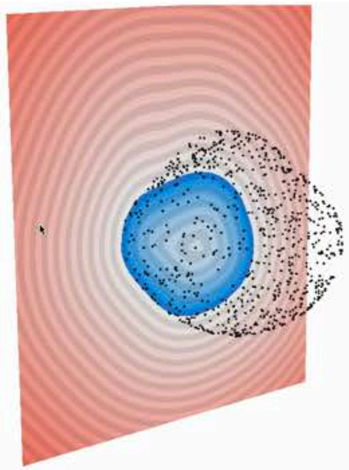


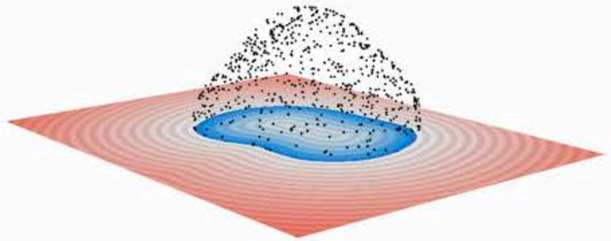
zero level set

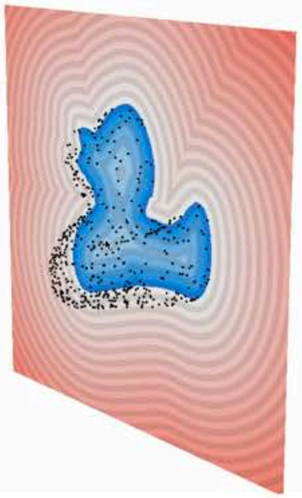


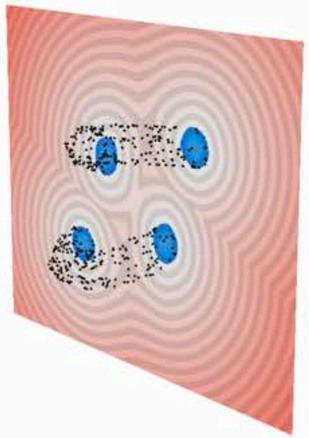
sphere-traced offset surfaces

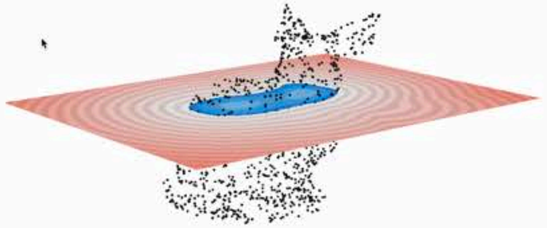
Pointwise evaluation



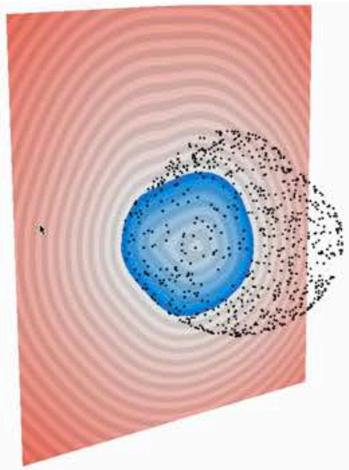


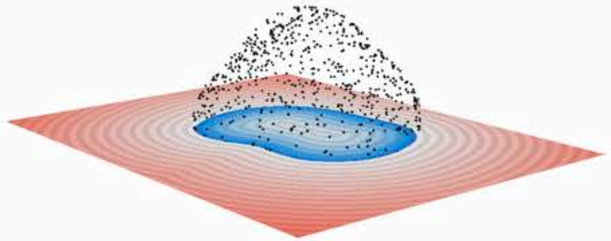


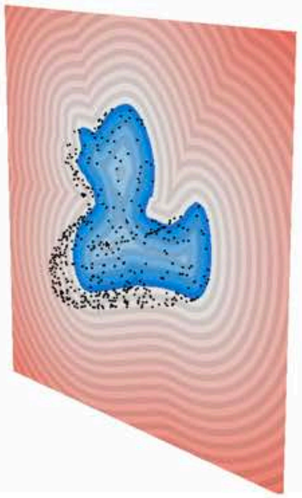


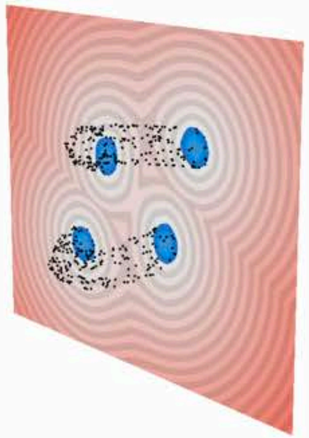


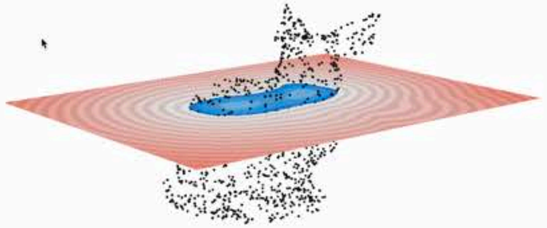
Pointwise evaluation









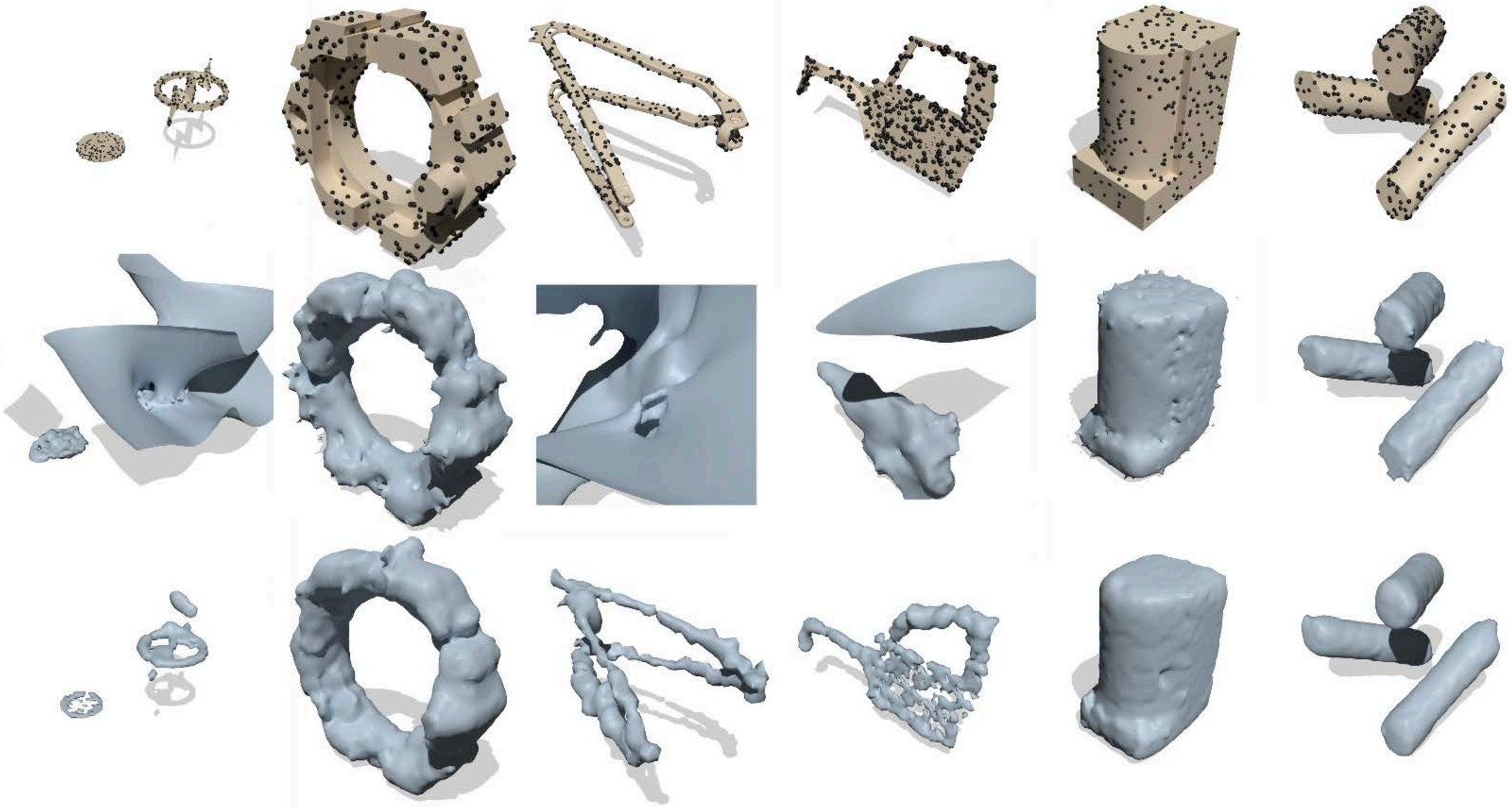


Reconstruction quality on sparse point clouds

true
(512 points)

signed heat method
Feng & Crane
2024

ours



Reconstruction quality on sparse point clouds

true
(512 points)

signed heat method
Feng & Crane 2024

ours

Reconstruction quality

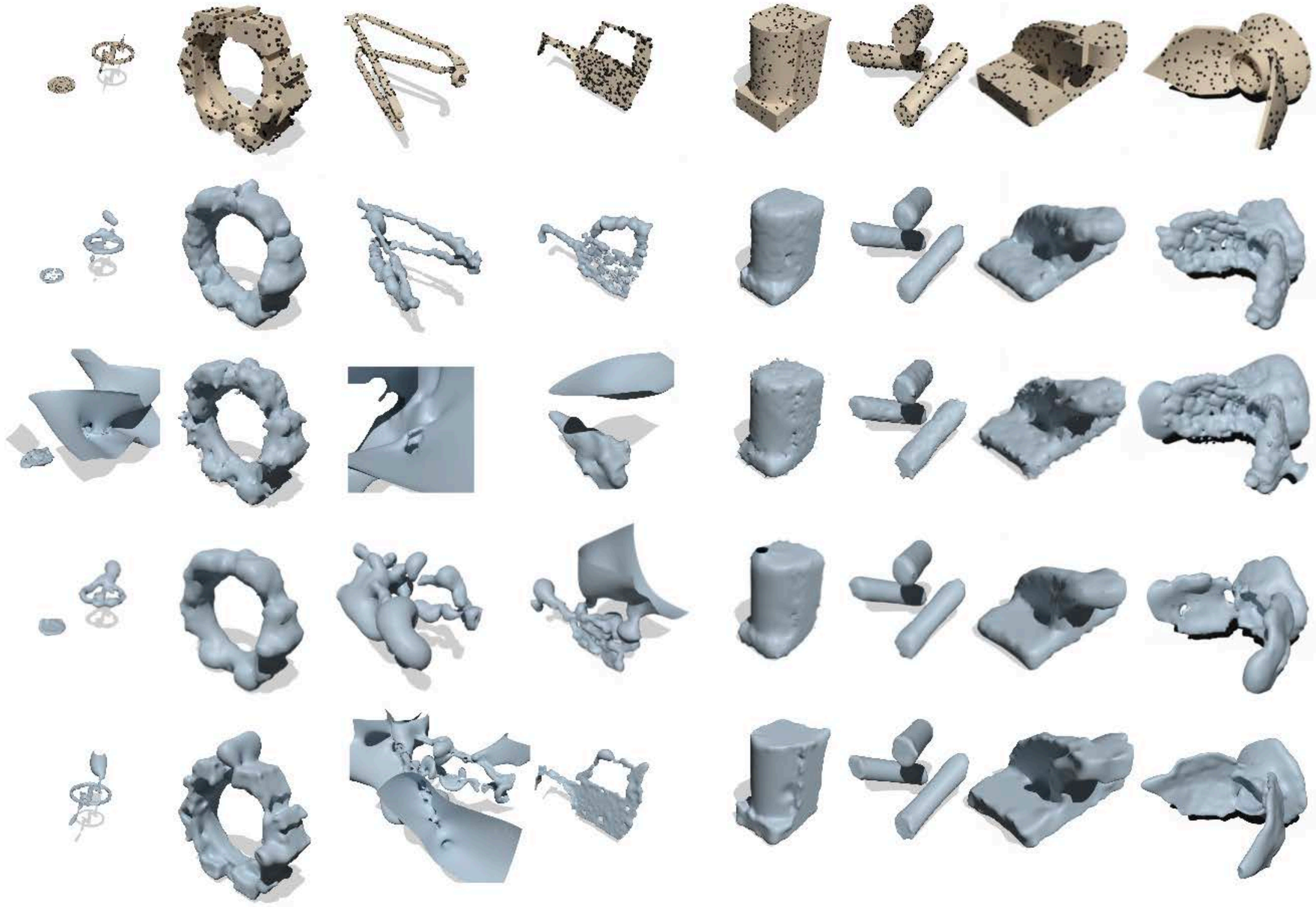
true
(512 points)

ours

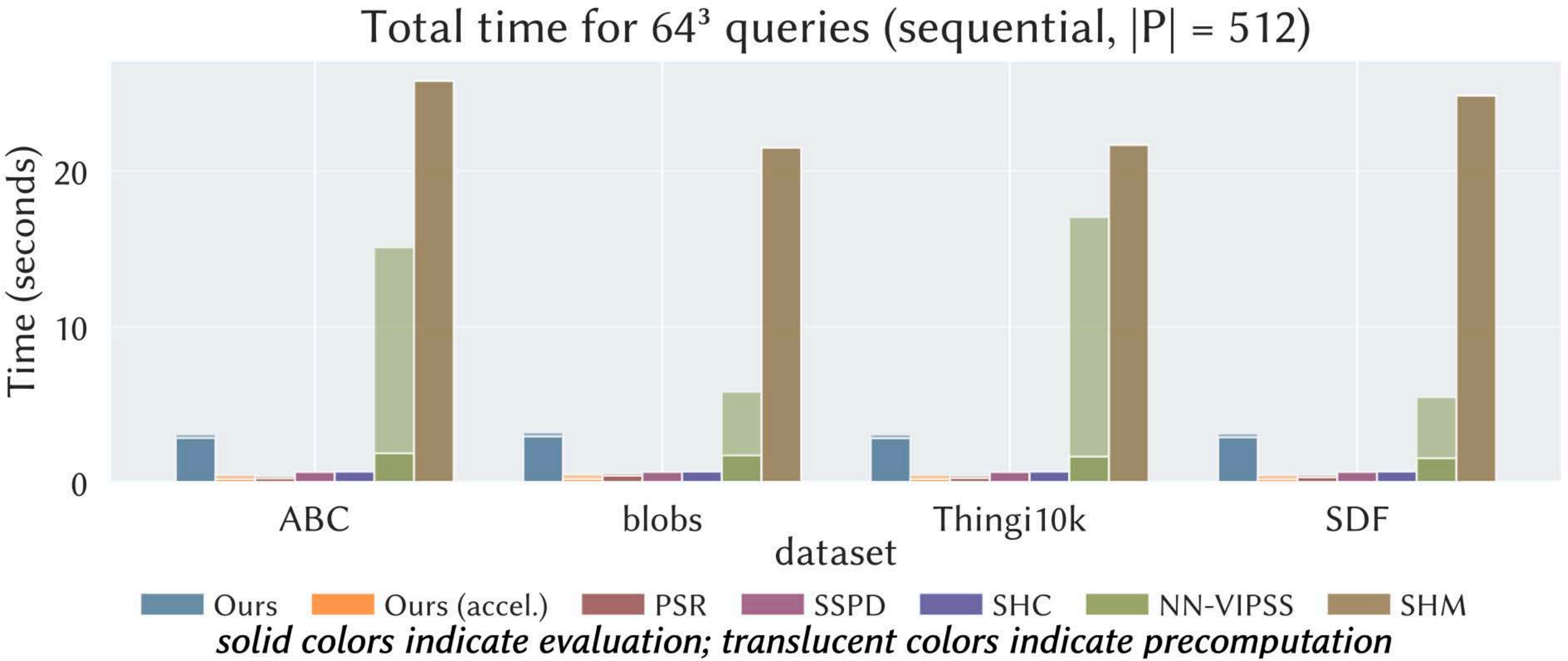
signed heat method
Feng & Crane 2024

SPSR
Kazhdan & Hoppe 2013

NN-VIPSS
w/ normals
Xia & Ju 2025

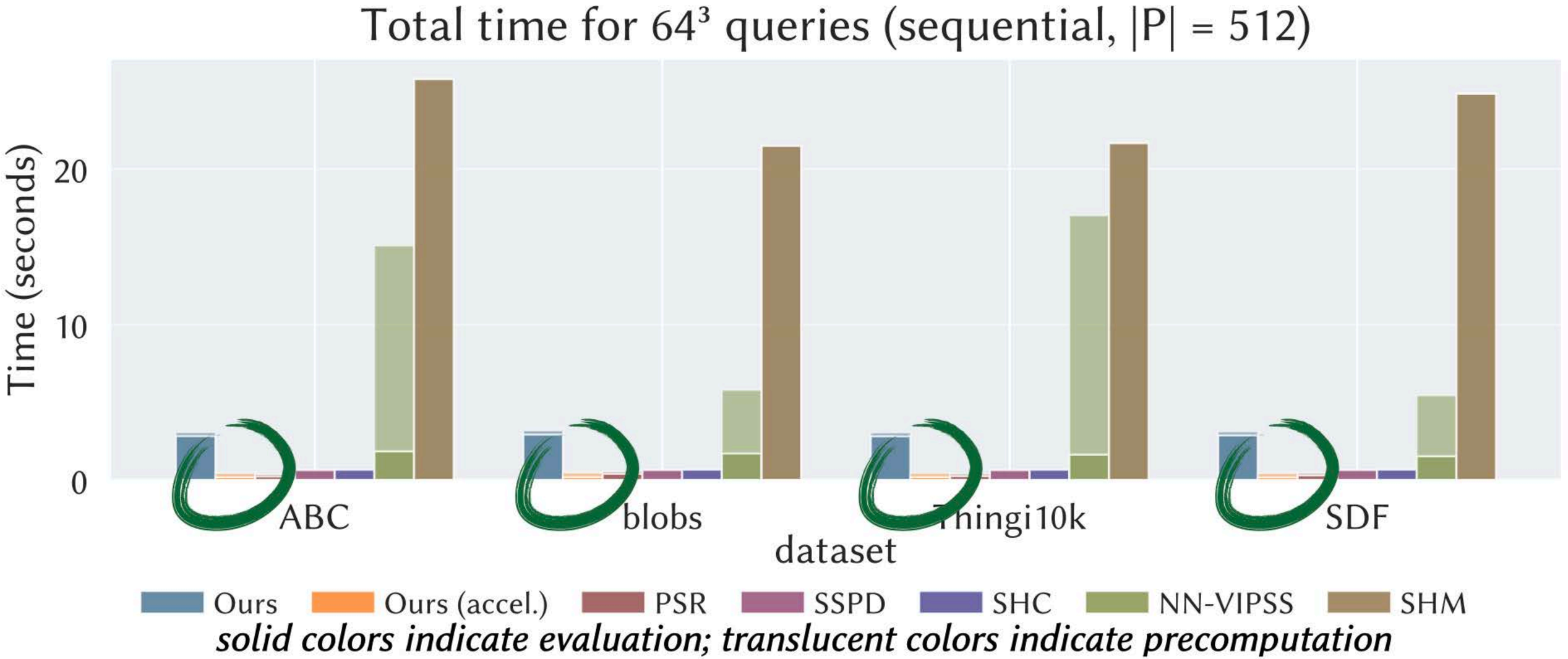


Performance

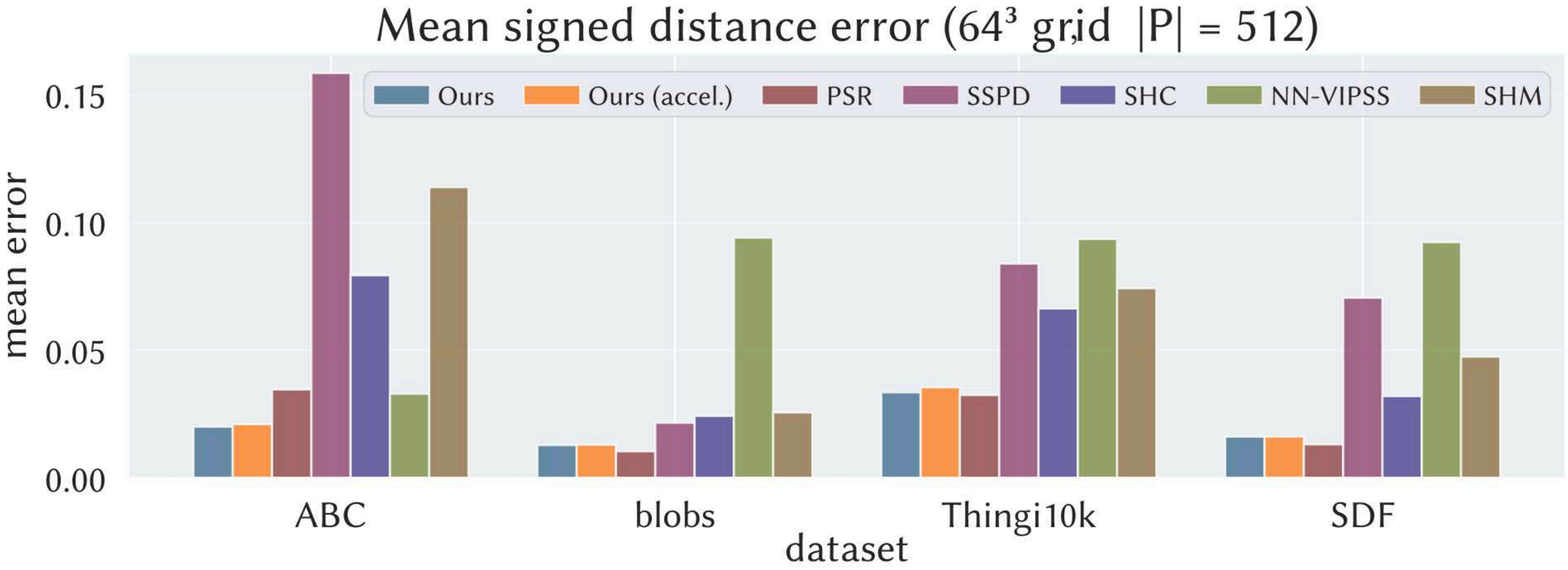


solid colors indicate evaluation; translucent colors indicate precomputation

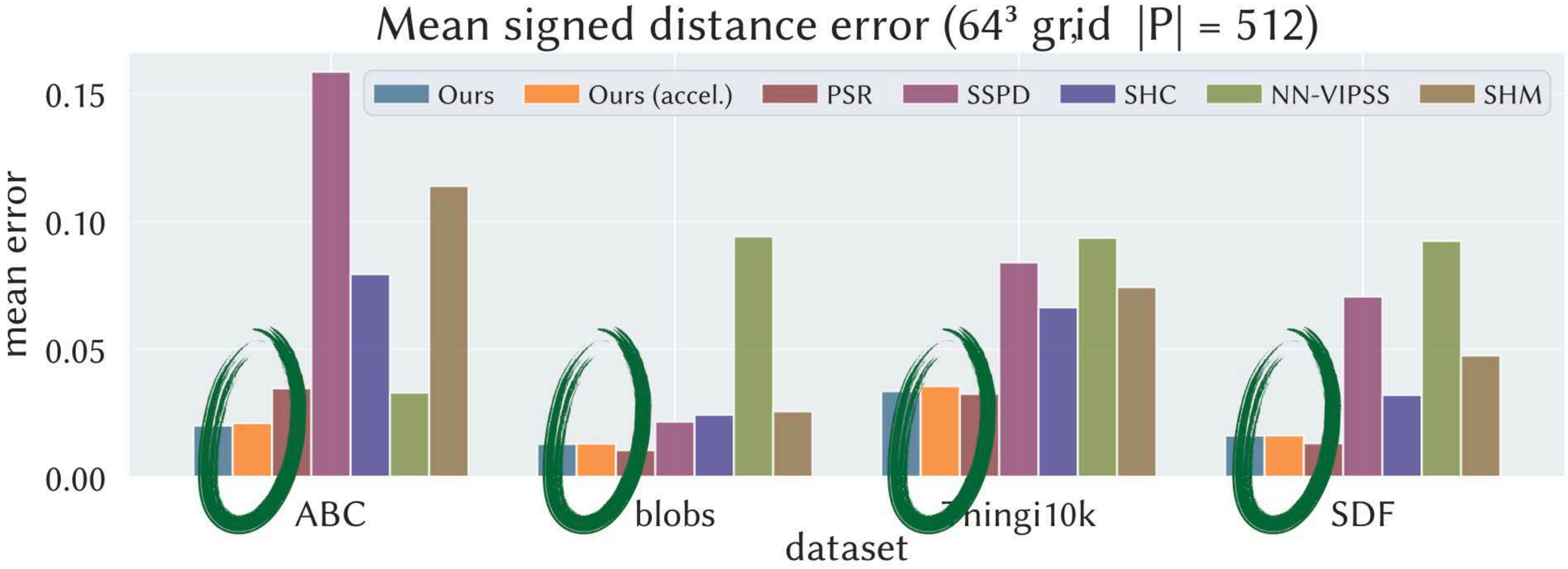
Performance



Accuracy


Mean signed distance error (64³ grid |P| = 512)
Ours
Ours (accel.)
PSR
SSPD
SHC
NN-VIPSS
SHM
mean error
0.15
0.10
0.05
0.00
ABC
blobs
Thingi10k
SDF
dataset

Accuracy


Mean signed distance error (64³ grid |P| = 512)
Ours
Ours (accel.)
PSR
SSPD
SHC
NN-VIPSS
SHM
mean error
0.15
0.10
0.05
0.00
ABC
blobs
Thingi10k
SDF
dataset

We intentionally do *not* learn end-to-end

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

re-train for *every* shape

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

re-train for *every* shape

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

simple, analytical formula

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

re-train for *every* shape

local surface

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

simple, analytical formula

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

re-train for *every* shape

re-usable, pre-trained network

local surface

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

simple, analytical formula

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian
[Wang et al. 2023]

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

re-train for *every* shape

re-usable, pre-trained network

local surface

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

simple, analytical formula

less training

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

N Neural Representation of U rcing Singular Hessian
[

1-Lipschitz Neural Distance Fields
[Coiffier & Béthune 2024]

HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition
[Wang et al. 2025]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

We intentionally do *not* learn end-to-end

eikonal equation

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

nonlinear

many local minima

re-train for *every* shape

re-usable, pre-trained network

local surface

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

simple, analytical formula

Implicit Geometric Regularization for Learning Shapes
[Gropp et al. 2020]

Neural Unsigned Distance Fields for Implicit Function Learning
[Chibane et al. 2020]

Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces
[Ma et al. 2021]

Phase Transitions, Distance Functions, and Implicit Neural Representations
[Lipman 2019]

Critical Regularizations for Neural Surface Reconstruction in the Wild
[Zhang et al. 2022]

p-Poisson Surface Reconstruction in Curl-Free Flow From Point Clouds
[Park et al. 2023]

StEik: Stabilizing the Optimization of Neural Signed Distance Functions and Finer Shape Representation
[Yang et al. 2023]

N… Neural Representation of U… rcing Singular Hessian
[…]

1-Lipschitz Neural Distance Fields

less training

faster inference

…ptimization with an
[…]

efunc: An Efficient Function Representation without Neural Networks
[Zhang & Wonka 2025]

etc. …

Higher-dimensional manifold learning

Higher-dimensional manifold learning

Goal: Sample from a target distribution.

One approach: Sample from an easy-to-sample *source distribution*, then apply a transformation.

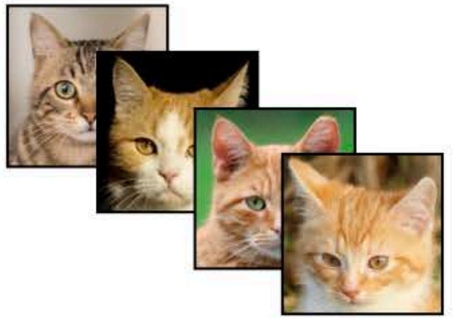
ρ_0 — *transform samples* → ρ_1

source distribution → target distribution

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

Song et al. 2021, Liu et al. 2022, etc.

Ex: (“images of cats”)



Connection to generative models

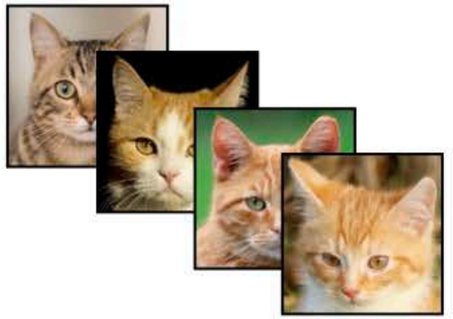
$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

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

source distribution

target distribution

Ex: (“images of cats”)



training data $\{x_i\}_{i=1}^N$

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

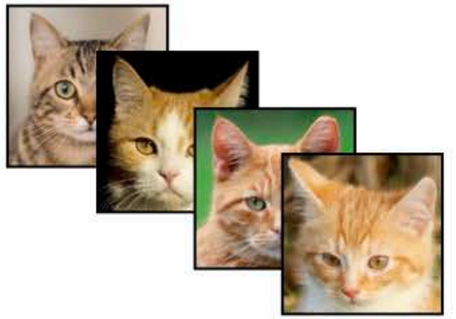
$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{1}{t} z + \left(\frac{1}{t} - 1\right) \nabla \log \rho(t, z) \qquad t \in (0, 1]$$

source distribution

target distribution

Ex: (“images of cats”)

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



training data $\{x_i\}_{i=1}^{N}$

Flow Straight and Fast

[Liu et al. 2021]

Closed-form Diffusion Models

[Scarvelis et al. 2023]

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

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

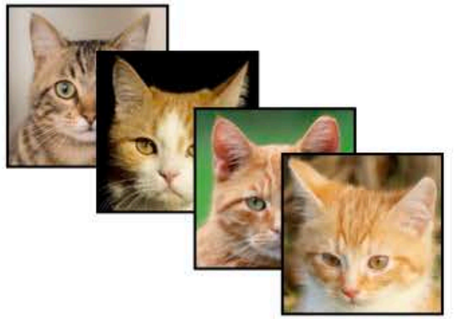
source distribution

target distribution

Ex: (“images of cats”)

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

current sample



training data $\{x_i\}_{i=1}^{N}$

Flow Straight and Fast

[Liu et al. 2021]

Closed-form Diffusion Models

[Scarvelis et al. 2023]

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

source distribution

target distribution

Ex: (“images of cats”)

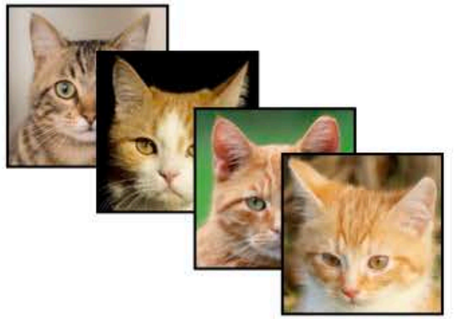
$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{1}{t} z + \left(\frac{1}{t} - 1\right) \nabla \log \rho(t, z) \quad t \in (0, 1]$$

pushes sample z towards closest training point

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

current sample

approximation of $\operatorname{argmin}_{t x_i} \|z - t x_i\|^2$



training data $\{x_i\}_{i=1}^{N}$

Flow Straight and Fast

[Liu et al. 2021]

Closed-form Diffusion Models

[Scarvelis et al. 2023]

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

source distribution

target distribution

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

pushes sample z towards closest training point

Flow-based models can only "memorize" their training data!

[Liu et al. 2021, Scarvelis et al. 2023, Gu et al. 2023, Gao & Li 2024, Carlini et al. 2023, Jain et al. 2024, Biroli et al. 2024, Gu et al. 2025, Somepalli et al. 2022; Pidstrigach 2022, Yoon et al. 2023, Kamb & Ganguli 2025]

Connection to generative models

$\rho_0 = \mathcal{N}(0, I)$ — *transform samples* → ρ_1

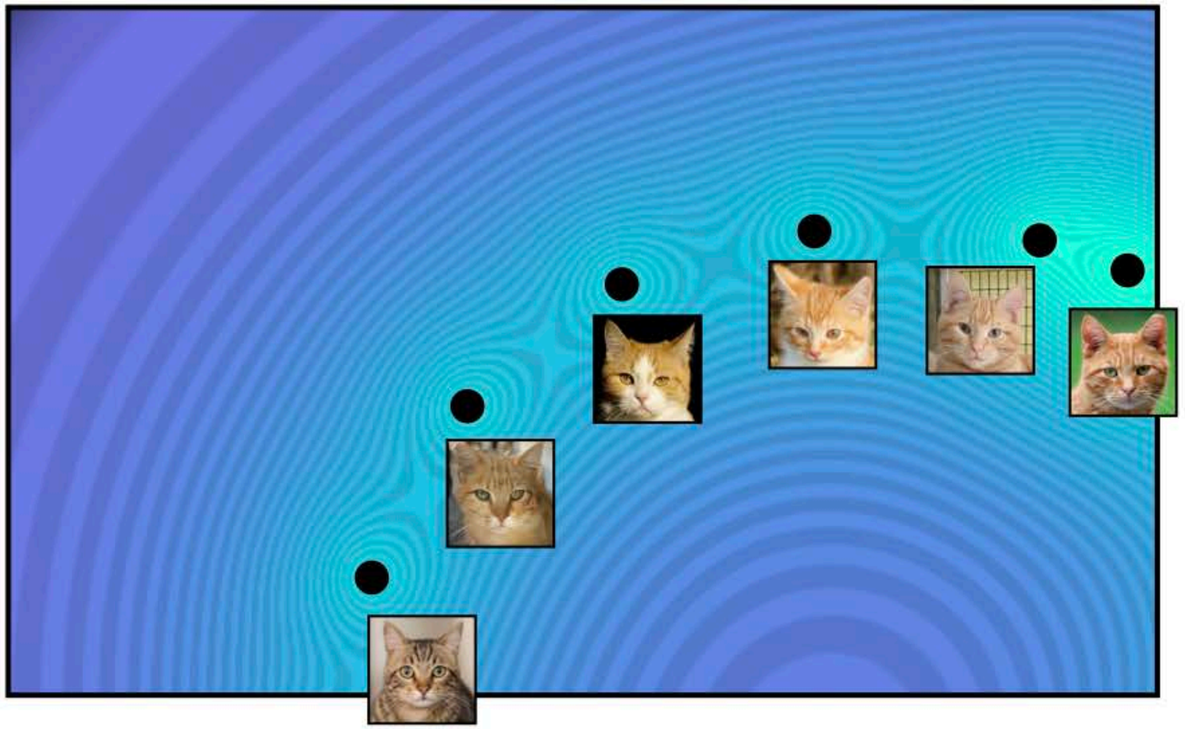
source distribution

target distribution

$$\frac{dz}{dt} = \frac{1}{t} z + \left(\frac{1}{t} - 1\right) \nabla \log \rho(t, z) \qquad t \in (0, 1]$$

pushes sample z towards closest training point

Flow-based models can only "memorize" their training data!



Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

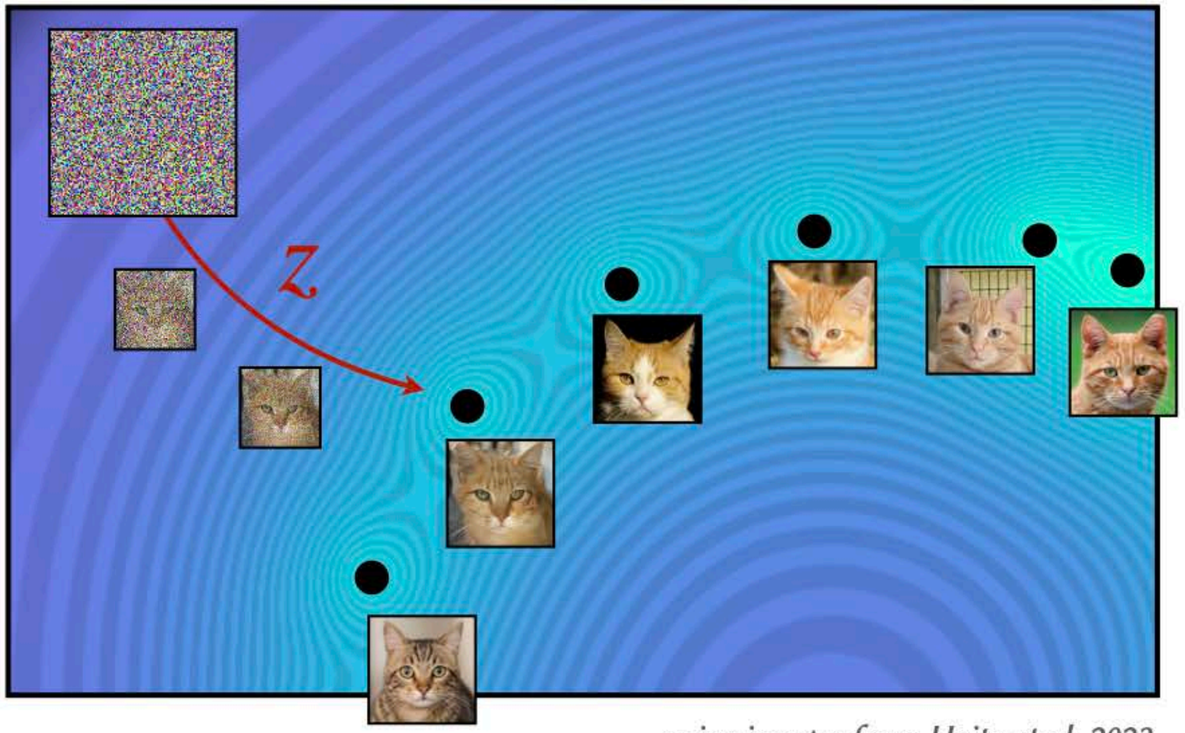
source distribution

target distribution

$$\frac{dz}{dt} = \frac{1}{t} z + \left(\frac{1}{t} - 1\right) \nabla \log \rho(t, z) \qquad t \in (0, 1]$$

pushes sample z towards closest training point

Flow-based models can only "memorize" their training data!



noisy images from Heitz et al. 2023

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

source distribution

$$\frac{dz}{dt} = \frac{1}{t} z + \left(\frac{1}{t} - 1\right) \nabla \log \rho(t, z) \qquad t \in (0, 1]$$

target distribution

pushes sample z towards closest training point

Flow-based models can only "memorize" their training data!

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

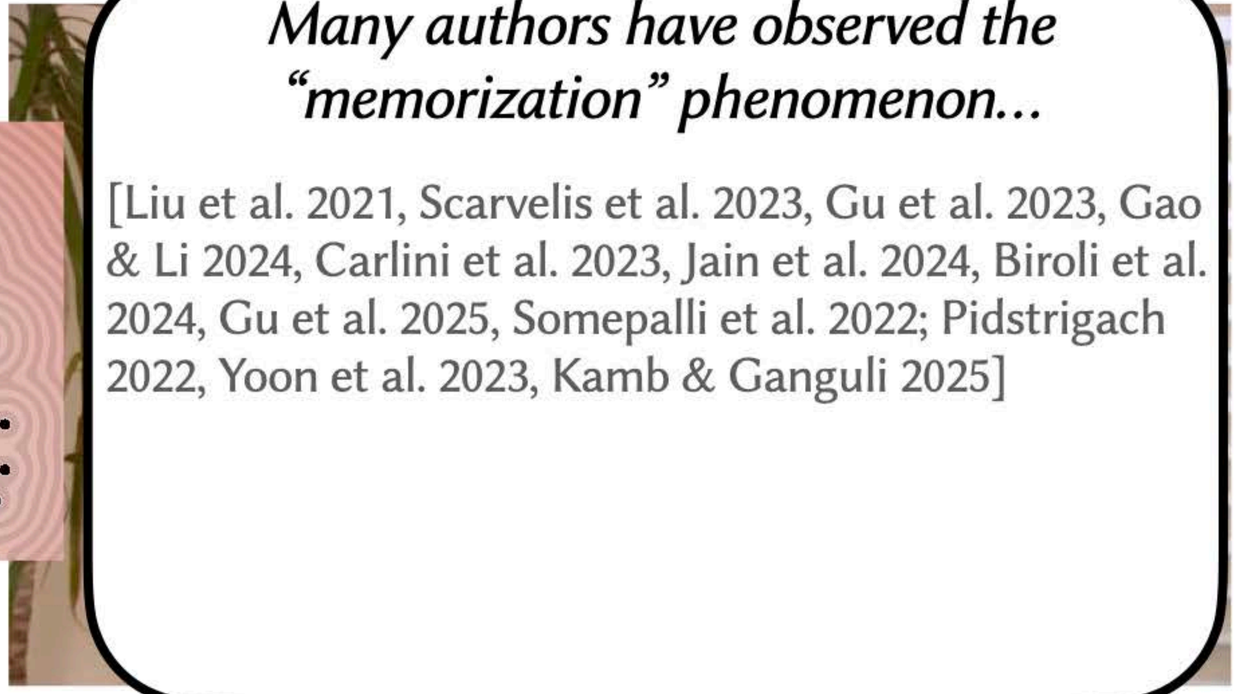
source distribution

target distribution

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

pushes sample z towards closest training point

Flow-based models can only "memorize" their training data!



Many authors have observed the "memorization" phenomenon…

[Liu et al. 2021, Scarvelis et al. 2023, Gu et al. 2023, Gao & Li 2024, Carlini et al. 2023, Jain et al. 2024, Biroli et al. 2024, Gu et al. 2025, Somepalli et al. 2022; Pidstrigach 2022, Yoon et al. 2023, Kamb & Ganguli 2025]

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

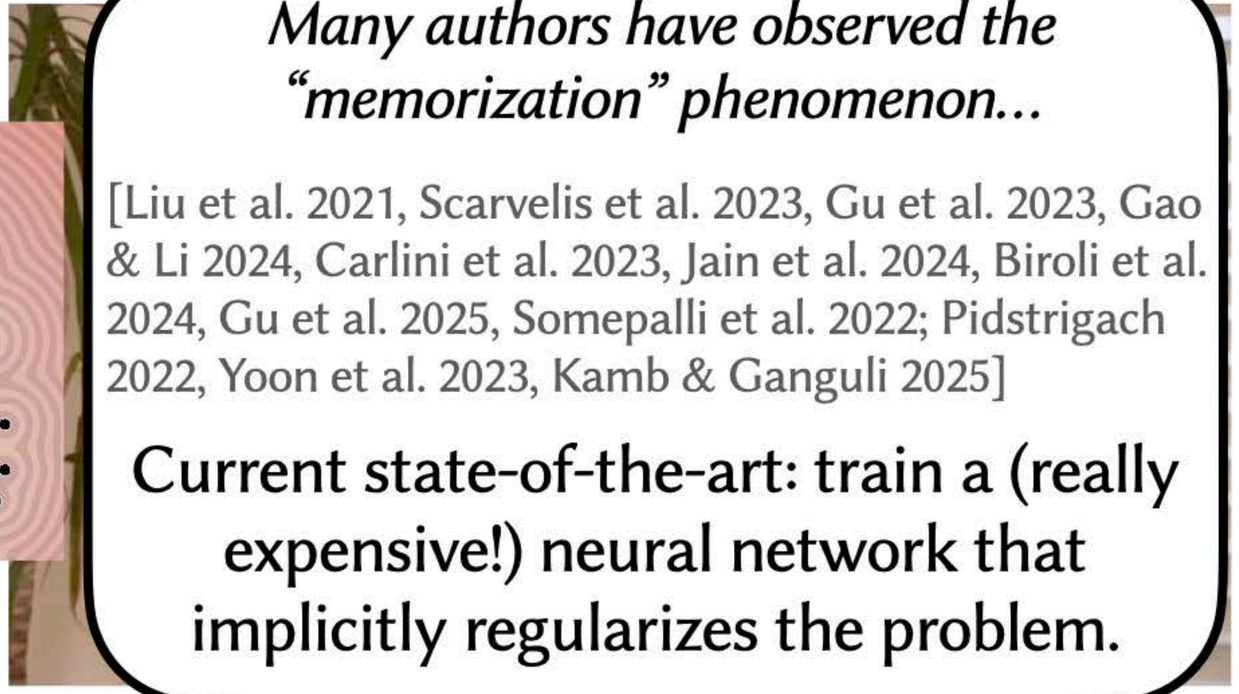
source distribution

target distribution

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

pushes sample z towards closest training point

Flow-based models can only "memorize" their training data!



Many authors have observed the "memorization" phenomenon…

[Liu et al. 2021, Scarvelis et al. 2023, Gu et al. 2023, Gao & Li 2024, Carlini et al. 2023, Jain et al. 2024, Biroli et al. 2024, Gu et al. 2025, Somepalli et al. 2022; Pidstrigach 2022, Yoon et al. 2023, Kamb & Ganguli 2025]

Current state-of-the-art: train a (really expensive!) neural network that implicitly regularizes the problem.

Connection to generative models

$$\rho_0 = \mathcal{N}(0, I) \xrightarrow{\text{transform samples}} \rho_1$$

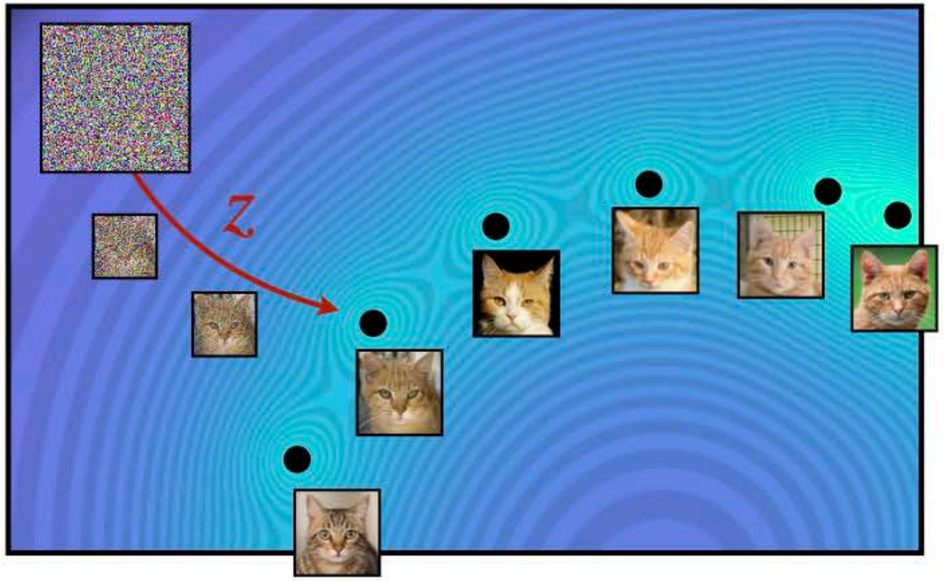
source distribution

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

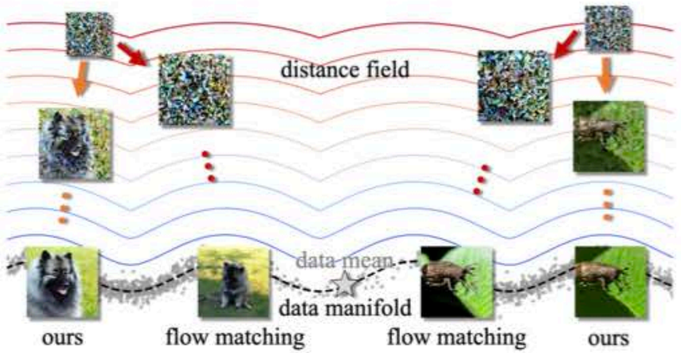
target distribution

pushes sample z towards closest training point

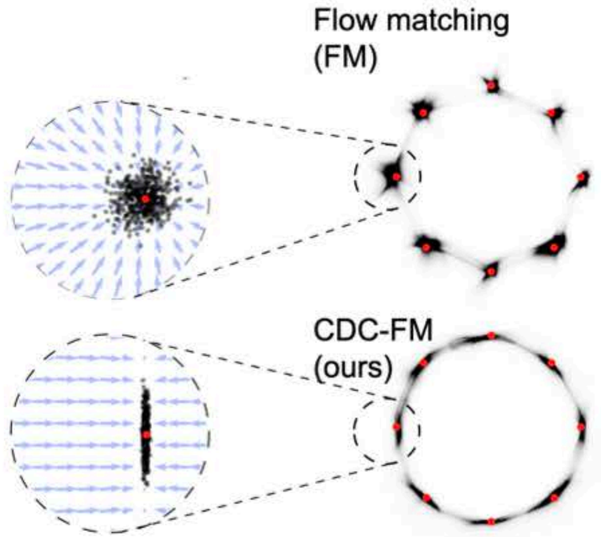
Flow-based models can only "memorize" their training data!



Distance Marching for Generative Modeling
[Wang et al. 2026]



Carré du champ Flow Matching: Better Quality-Generalisation Tradeoff in Generative Models
[Bamberger et al. 2026]

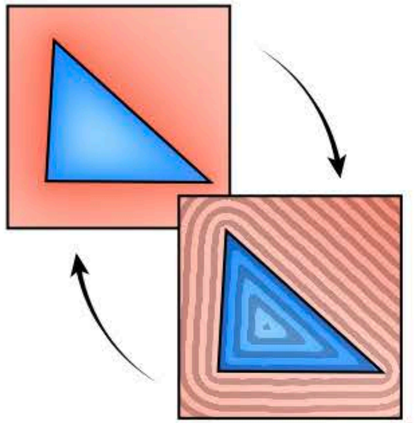


Conclusion

Takeaways

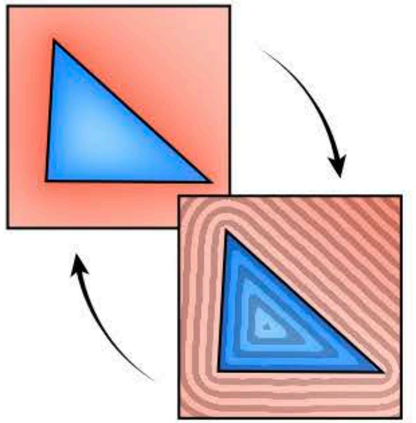
Takeaways

- Signed distance, winding numbers, and Poisson Surface Reconstruction are, in some sense, ***equivalent***



Takeaways

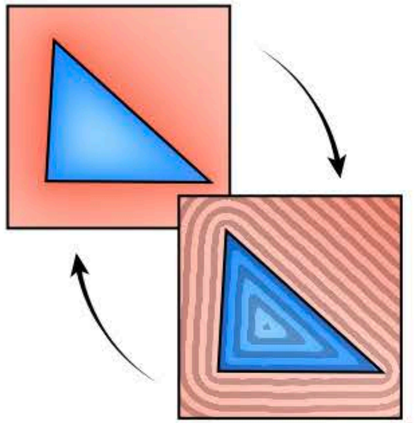
- Signed distance, winding numbers, and Poisson Surface Reconstruction are, in some sense, ***equivalent***
- Decades of distance approximations can be distilled into two formulas

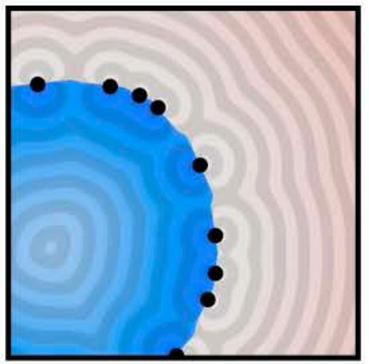


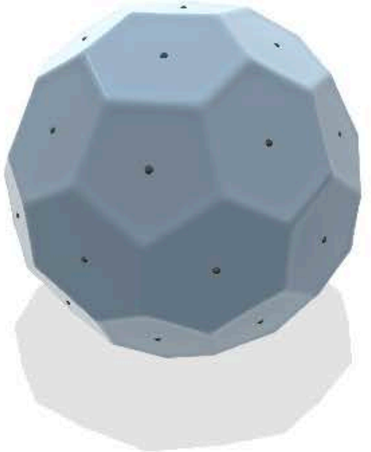
Takeaways

- Signed distance, winding numbers, and Poisson Surface Reconstruction are, in some sense, ***equivalent***

- Decades of distance approximations can be distilled into two formulas — but **none of them work for point clouds**



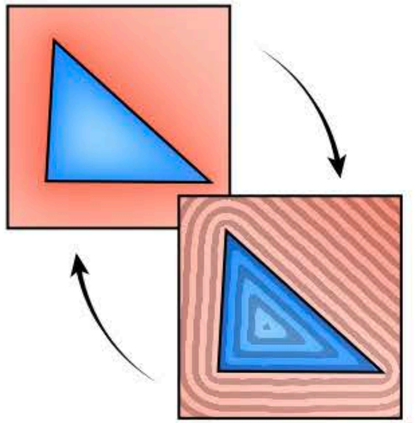


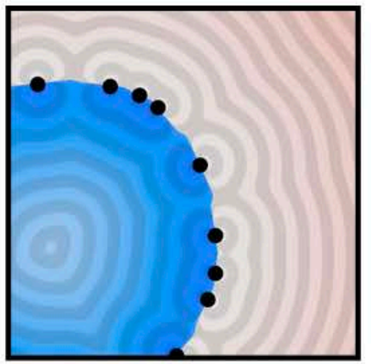


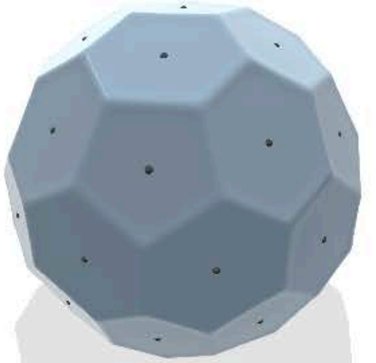
Takeaways

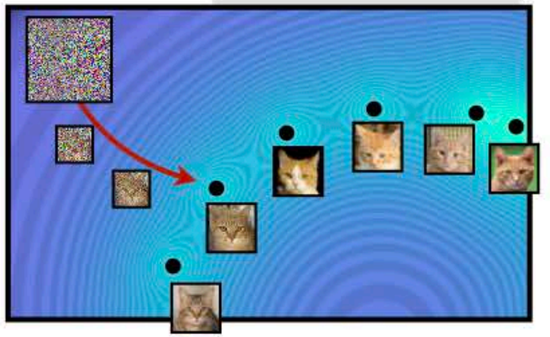
- Signed distance, winding numbers, and Poisson Surface Reconstruction are, in some sense, ***equivalent***

- Decades of distance approximations can be distilled into two formulas — but **none of them work for point clouds**



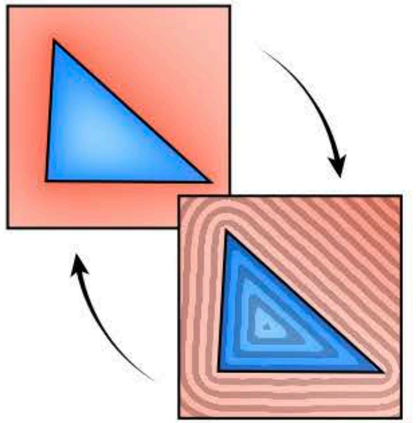




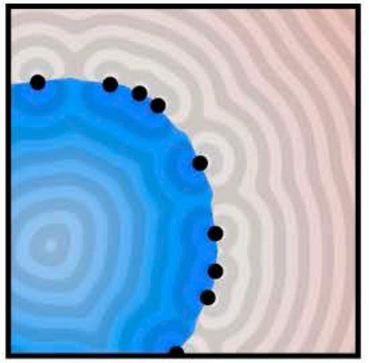


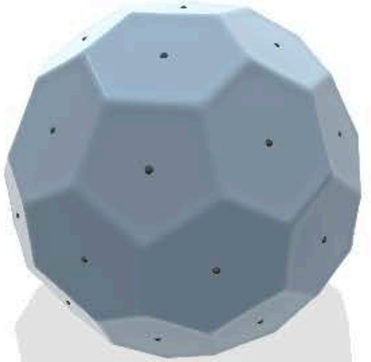
Takeaways

- Signed distance, winding numbers, and Poisson Surface Reconstruction are, in some sense, ***equivalent***

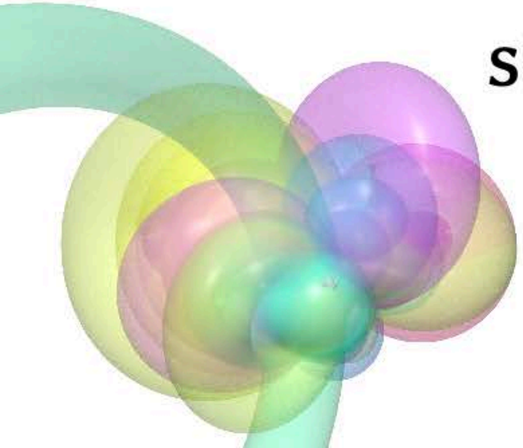


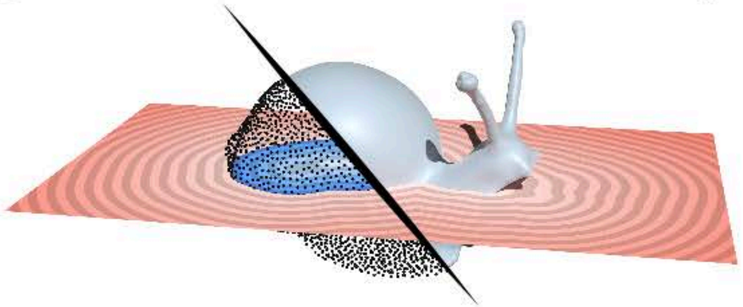
- Decades of distance approximations can be distilled into two formulas — but **none of them work for point clouds**

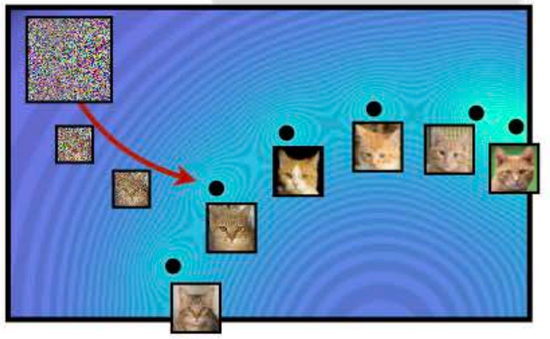




- We treat ***points as tori***, and approximate true signed distance directly from point clouds







Future directions

- Accelerating pre-computation (torus fitting)

Future directions

- Accelerating pre-computation (torus fitting)

Precompute times using one NVIDIA RTX 3090:

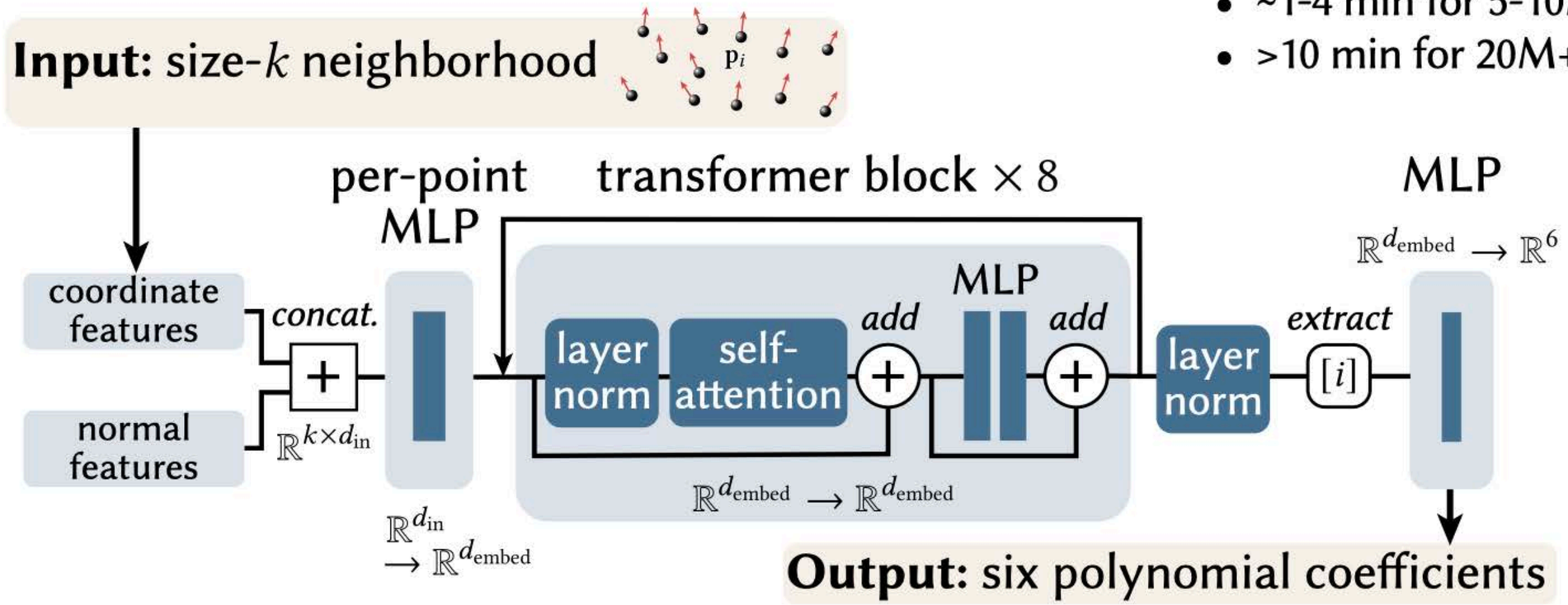
- <10s for 100k points
- <60s for 1M points
- ~1-4 min for 5-10M points
- >10 min for 20M+ points

Future directions

- Accelerating pre-computation (torus fitting)

Precompute times using one NVIDIA RTX 3090:

- <10s for 100k points
- <60s for 1M points
- ~1-4 min for 5-10M points
- >10 min for 20M+ points

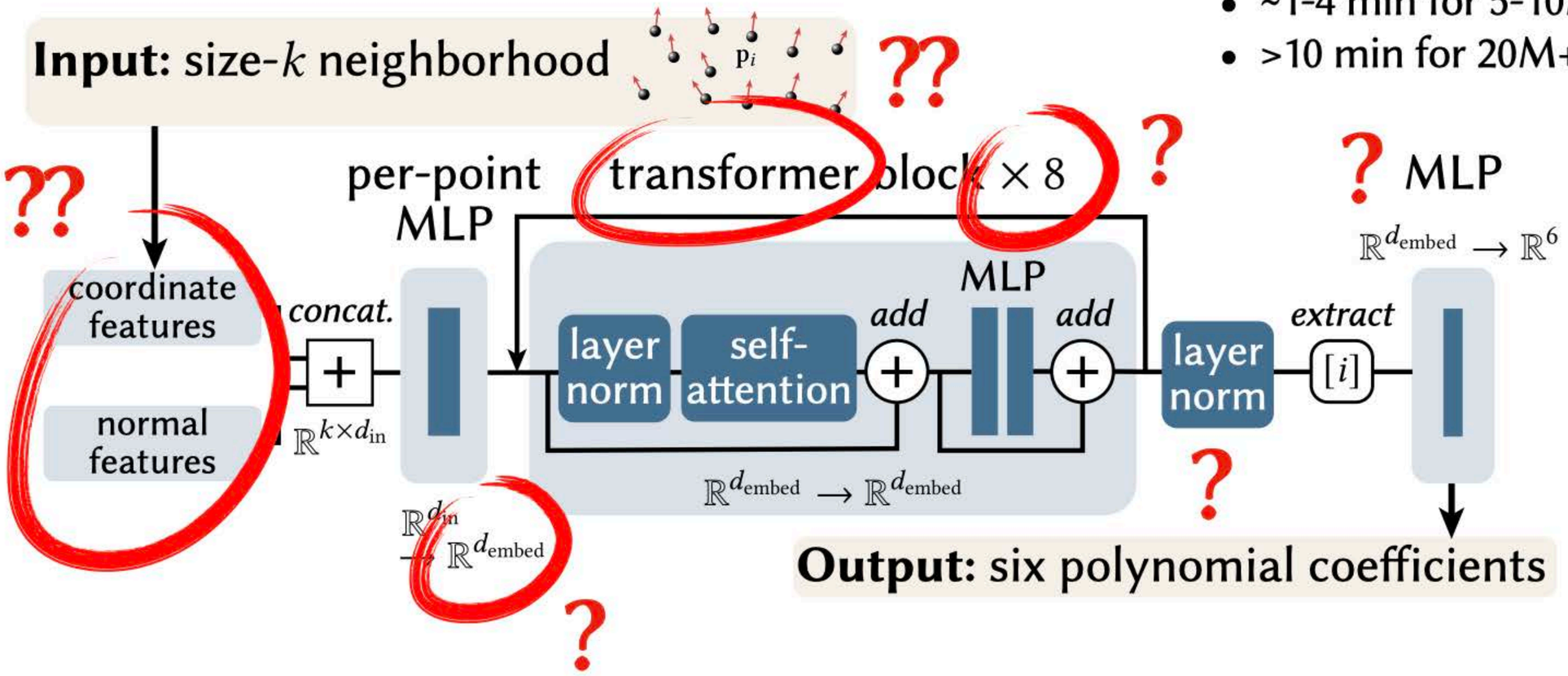


Future directions

- Accelerating pre-computation (torus fitting)

Precompute times using one NVIDIA RTX 3090:

- <10s for 100k points
- <60s for 1M points
- ~1-4 min for 5-10M points
- >10 min for 20M+ points

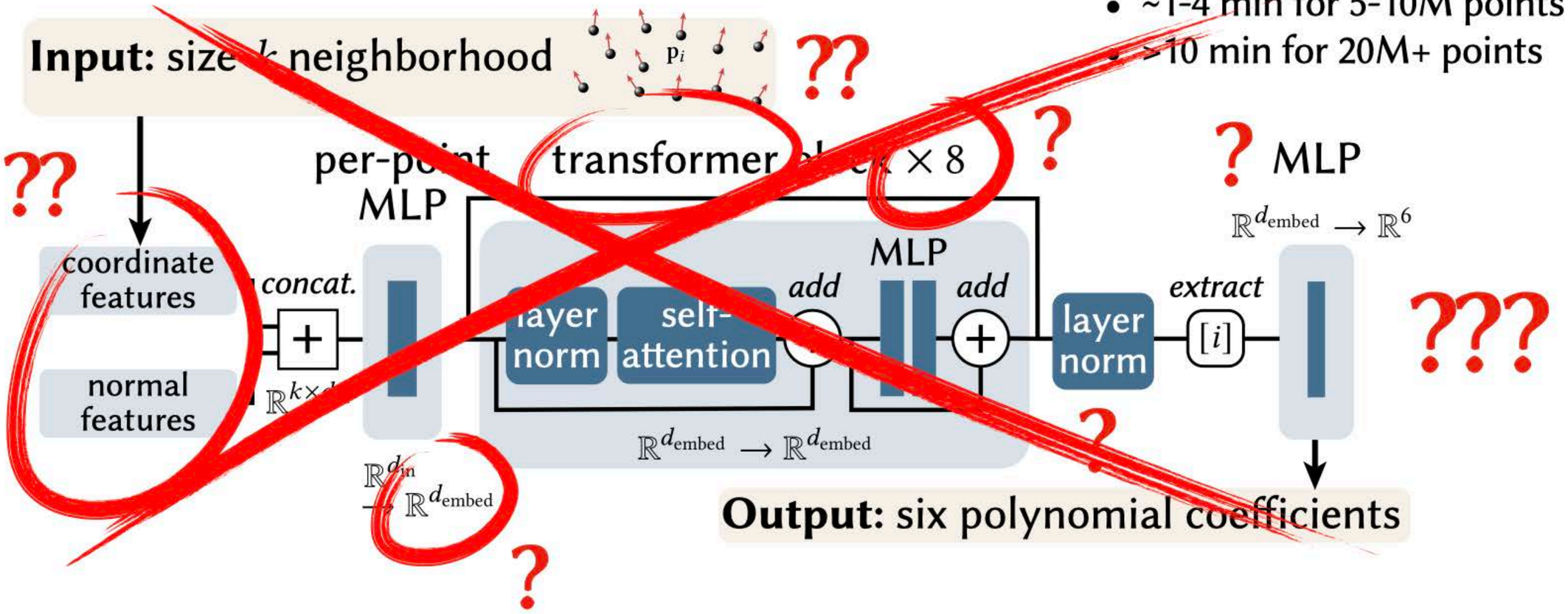


Future directions

- Accelerating pre-computation (torus fitting)

Precompute times using one NVIDIA RTX 3090:

- <10s for 100k points
- <60s for 1M points
- ~1-4 min for 5-10M points
- >10 min for 20M+ points

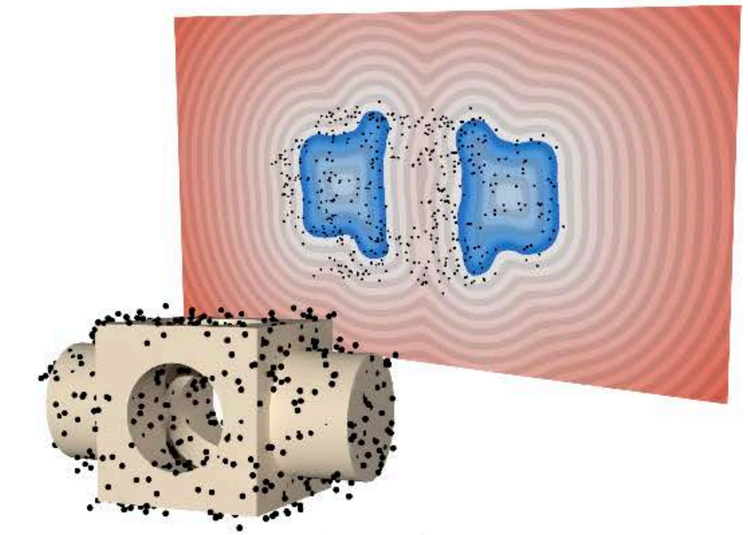


Future directions

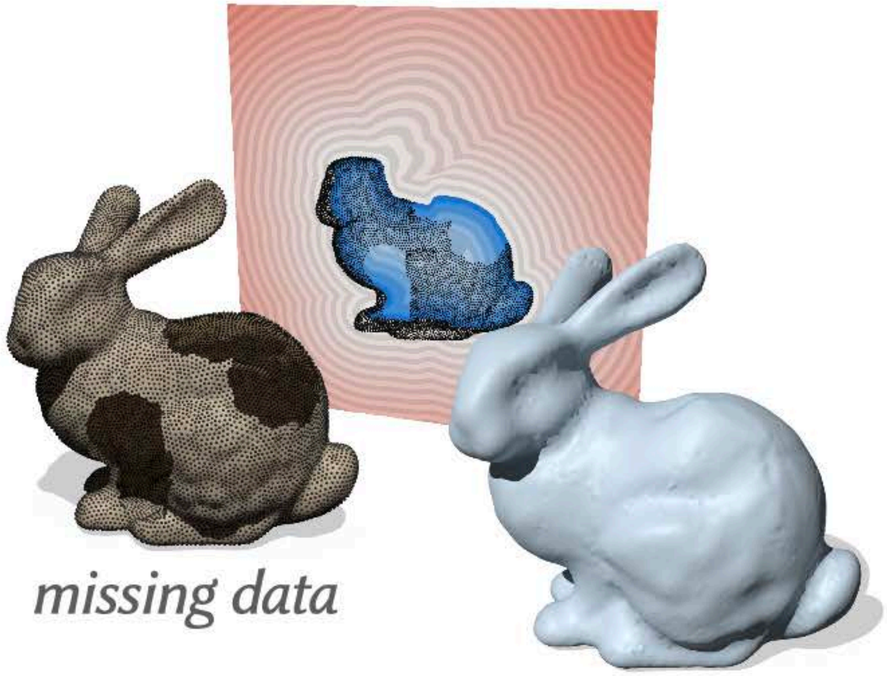
- Accelerating pre-computation (torus fitting)
- Robustness to corruption (noise, missing data, outliers, etc.)

Future directions

- Accelerating pre-computation (torus fitting)
- Robustness to corruption (noise, missing data, outliers, etc.)



noisy point cloud



missing data

unseen sampling density

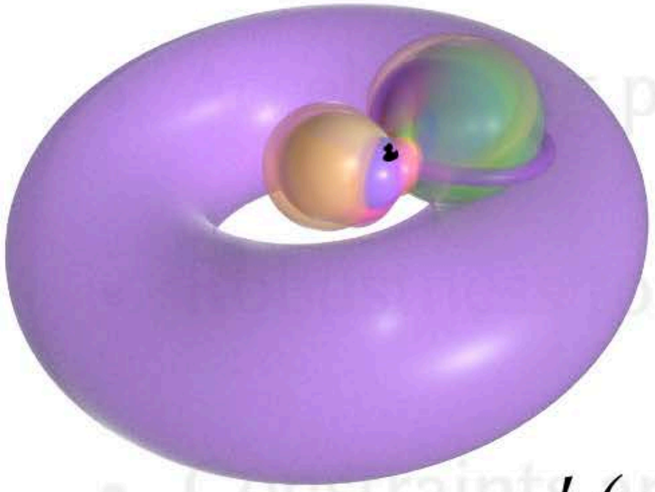
Future directions

- Accelerating pre-computation (torus fitting)
- Robustness to corruption (noise, missing data, outliers, etc.)

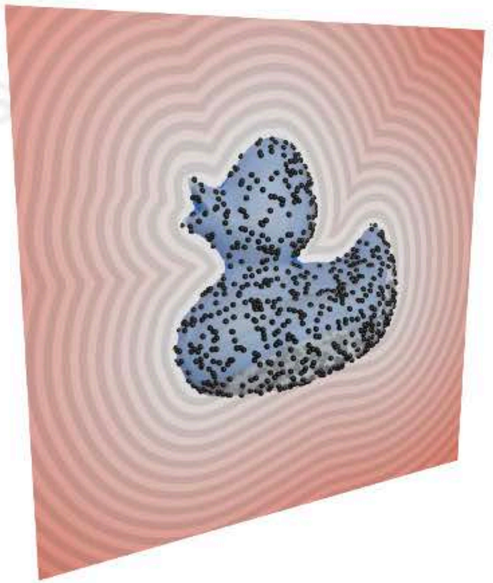
Future directions

- Accelerating pre-computation (torus fitting)
- Robustness to corruption (noise, missing data, outliers, etc.)
- Constraints and priors
- Compression/simplification
- …

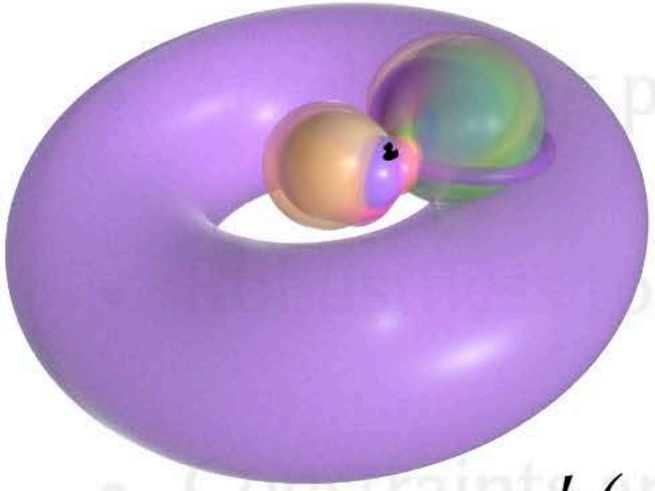
Future directions



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

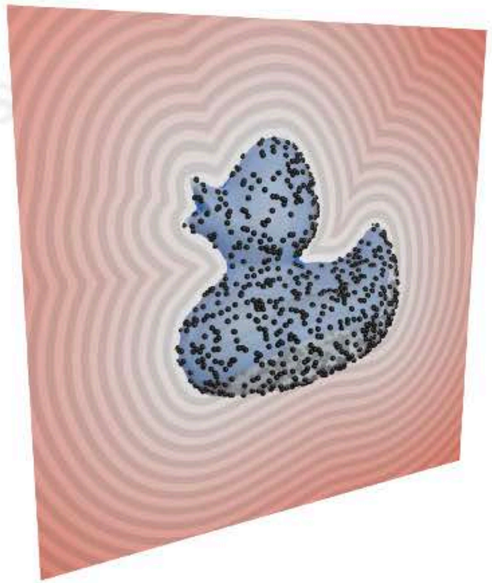


Future directions

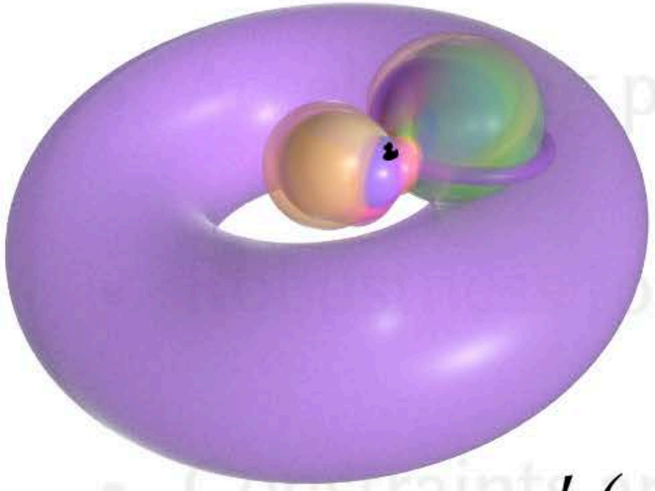


differentiable

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



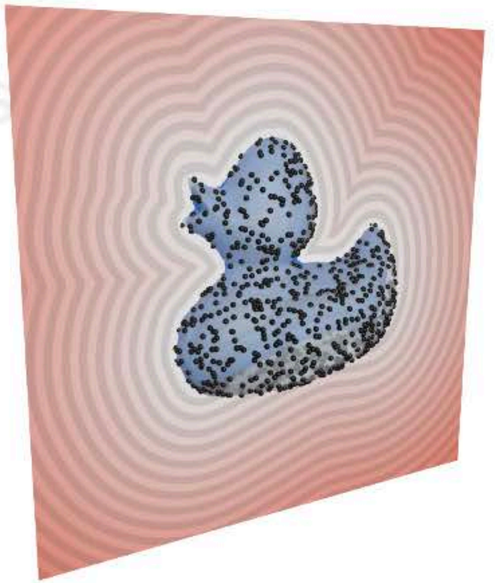
Future directions



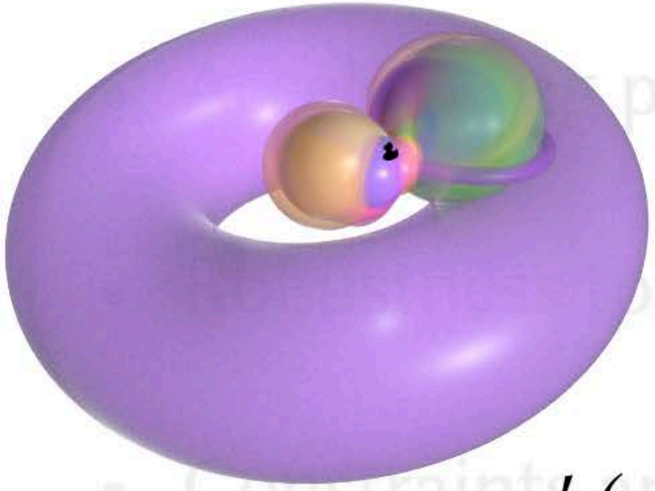
point-based

differentiable

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



Future directions

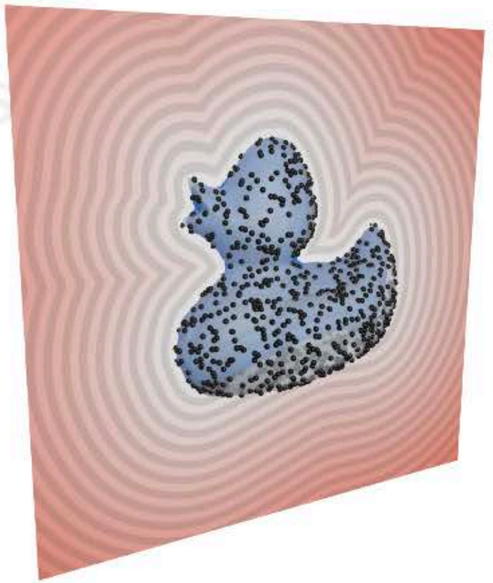


point-based

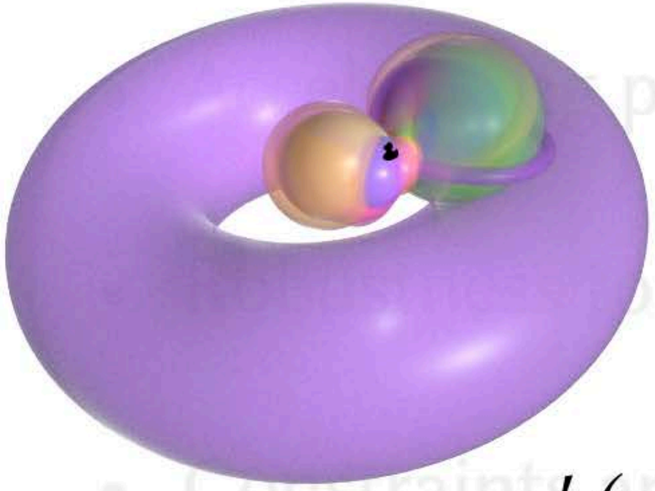
differentiable

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

output-sensitive



Future directions

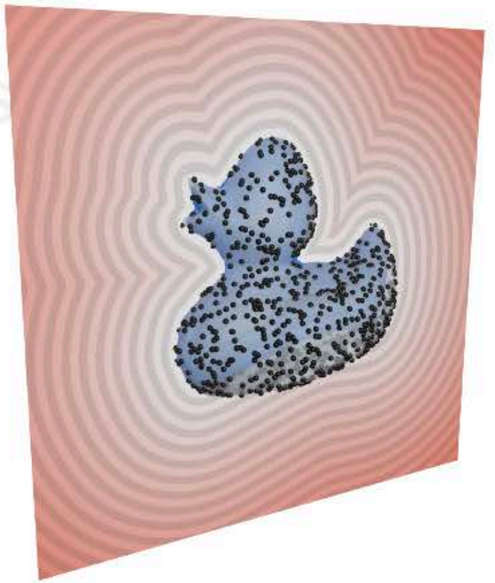


point-based

differentiable

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

output-sensitive



- Torus signed distance as a differentiable shape representation

Thanks!