A Heat Method for Generalized Signed Distance

Nicole Feng, Keenan Crane

Carnegie Mellon University

SIGGRAPH 2024

Signed distance functions (SDFs) are essential

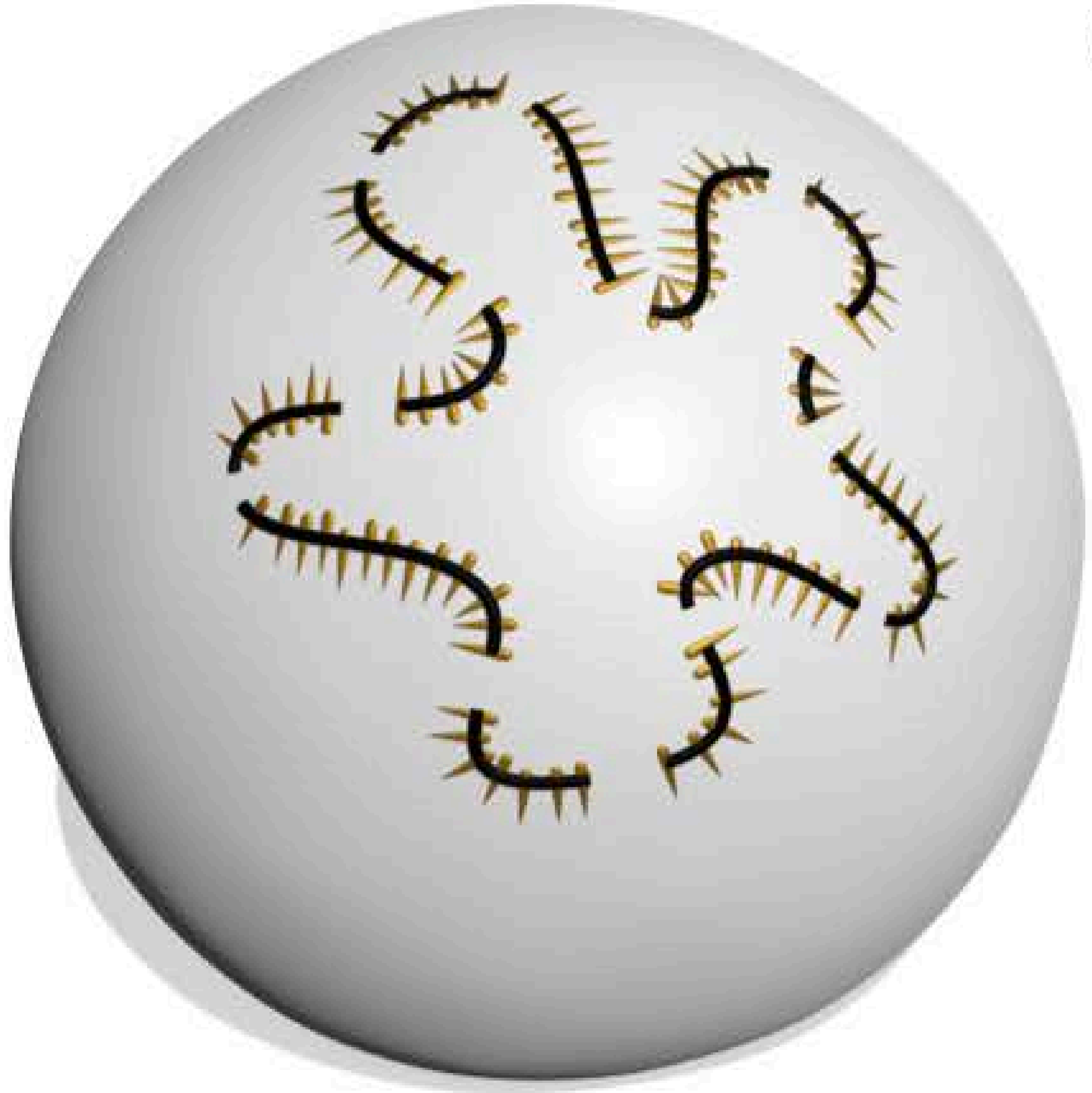
Challenge: SDFs from messy, real-world input

"broken" geometry

Signed Heat Method (SHM)

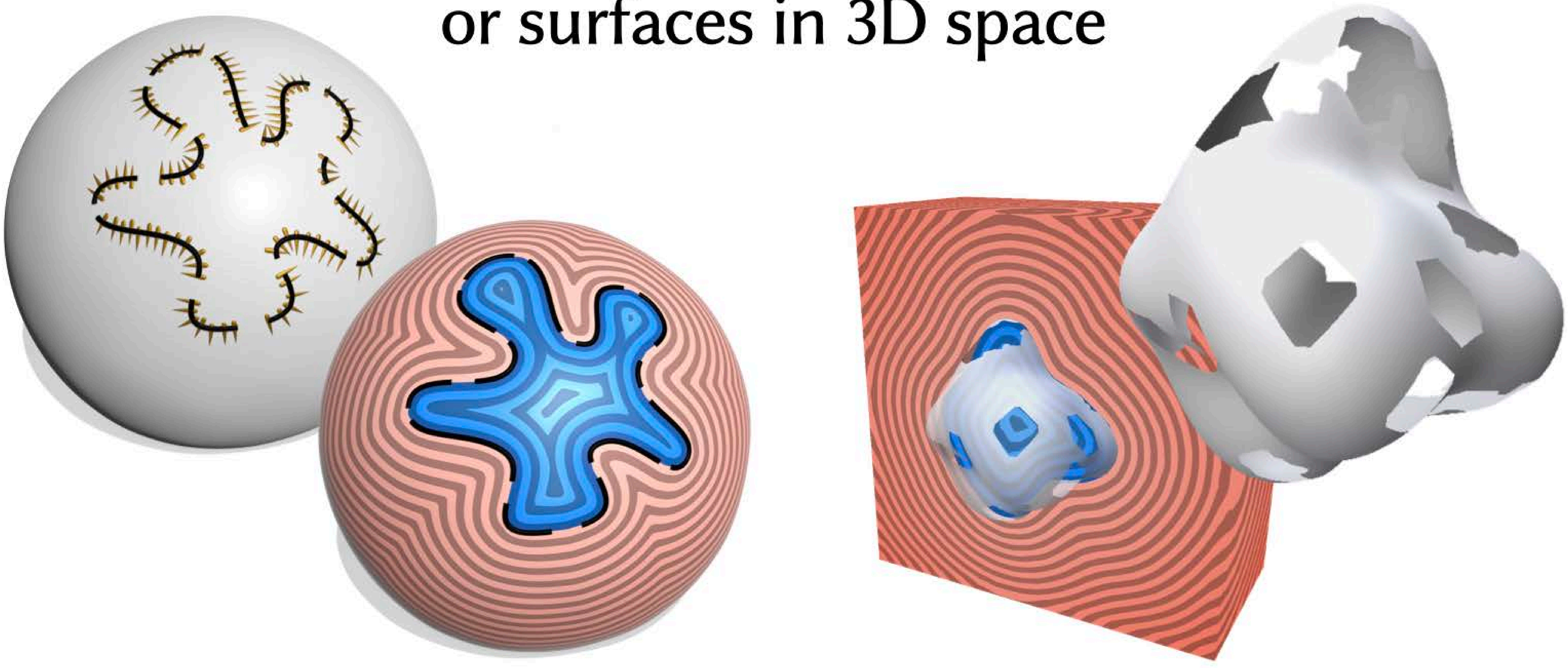
Signed Heat Method (SHM)

Input: (possibly broken) oriented curves on a surface,
or surfaces in 3D space



Signed Heat Method (SHM)

Input: (possibly broken) oriented curves on a surface, or surfaces in 3D space

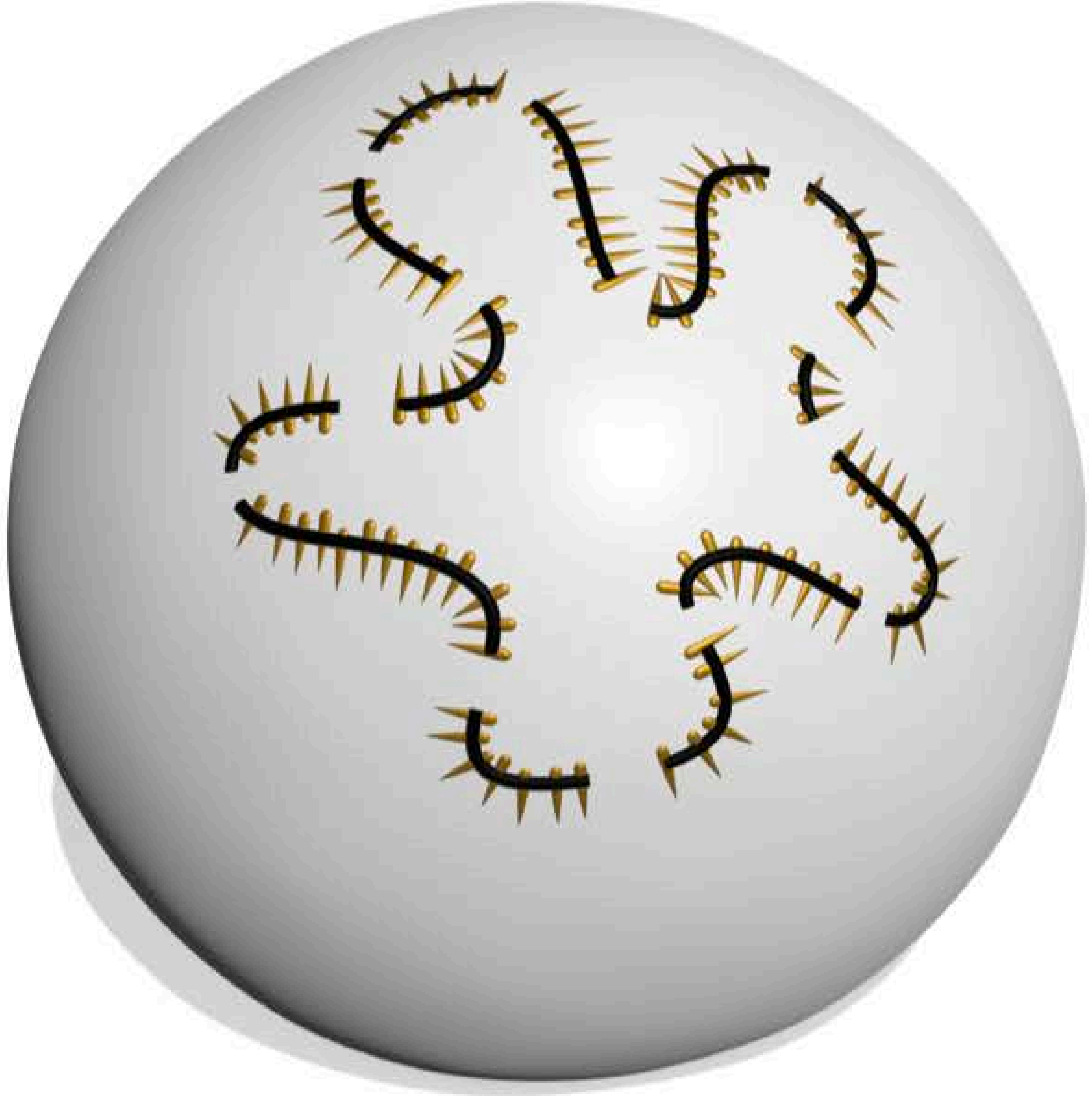


Output: signed distance approximation

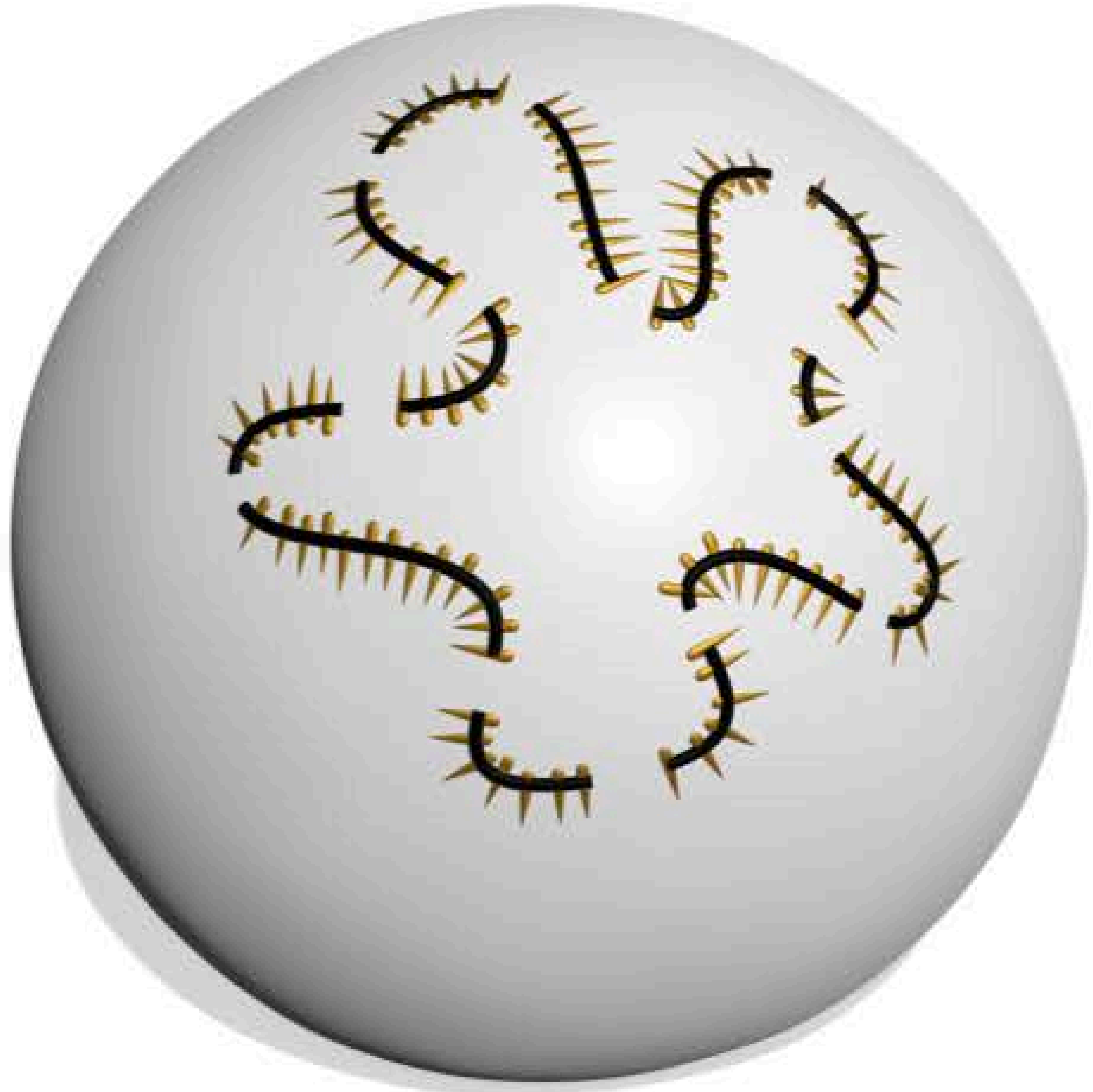
Applies to all discretizations & dimensions

Basic idea

input

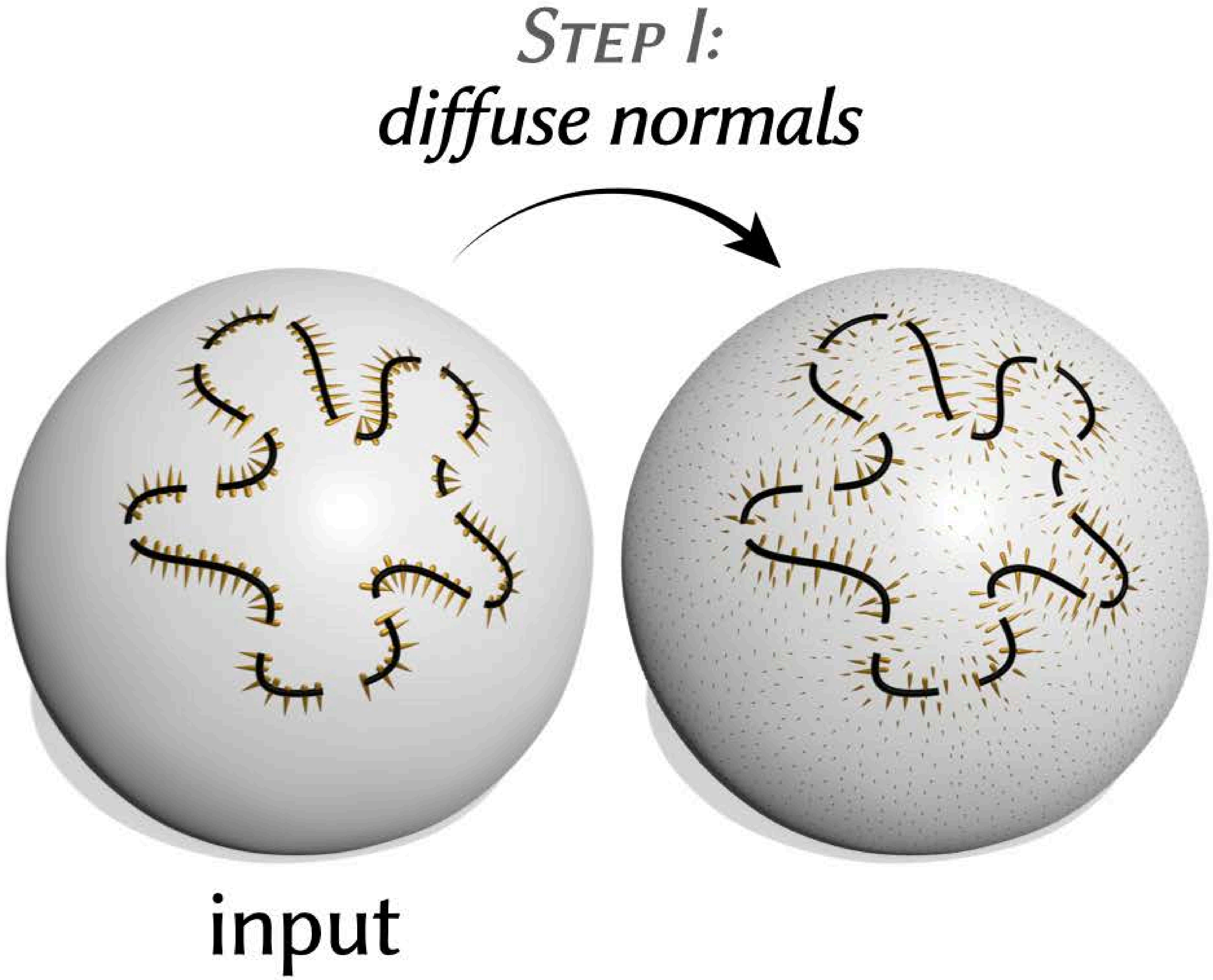


Basic idea

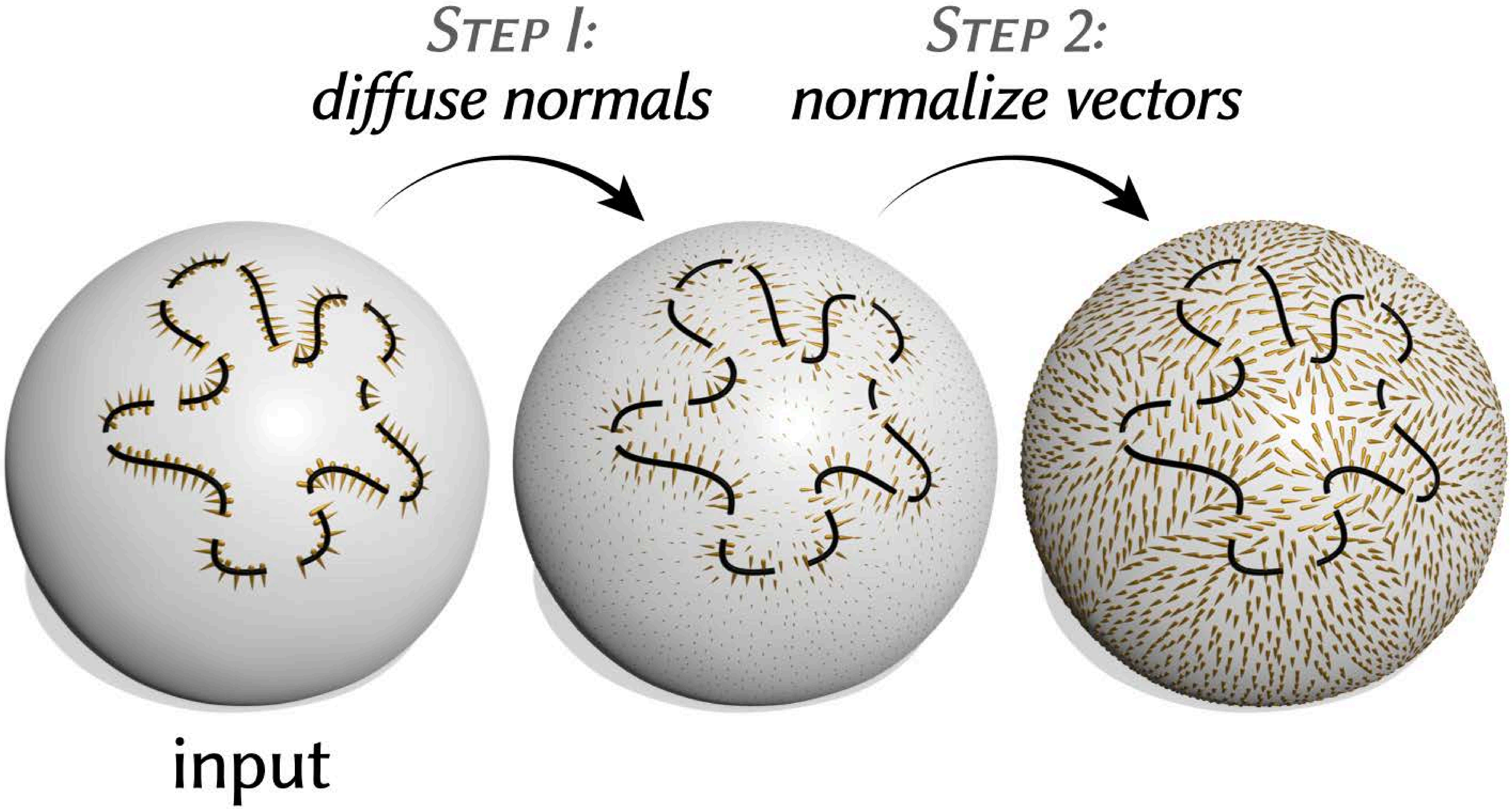


input

Basic idea



Basic idea



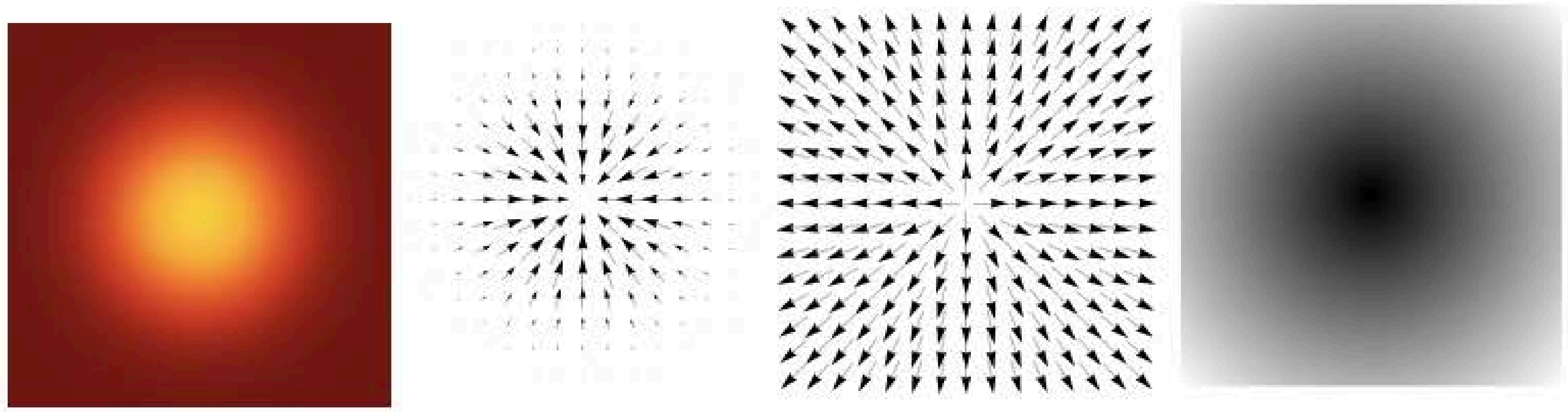
Basic idea


STEP 1:
diffuse normals
STEP 2:
normalize vectors
STEP 3:
integrate
input
generalized
signed distance

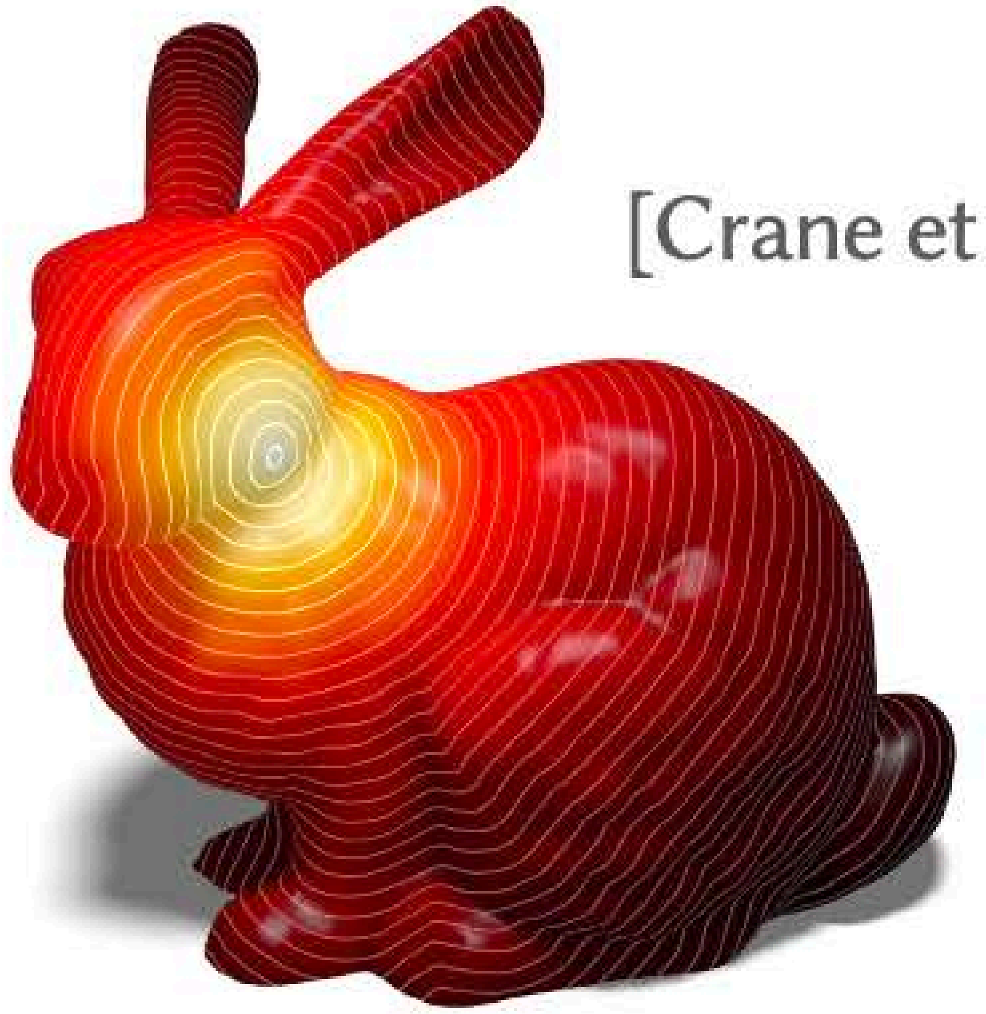
Past work: heat methods in geometry processing

Past work: heat methods in geometry processing

Unsigned Heat Method (UHM) for geodesic distance
[Crane, Weischedel, Wardetzky 2013]



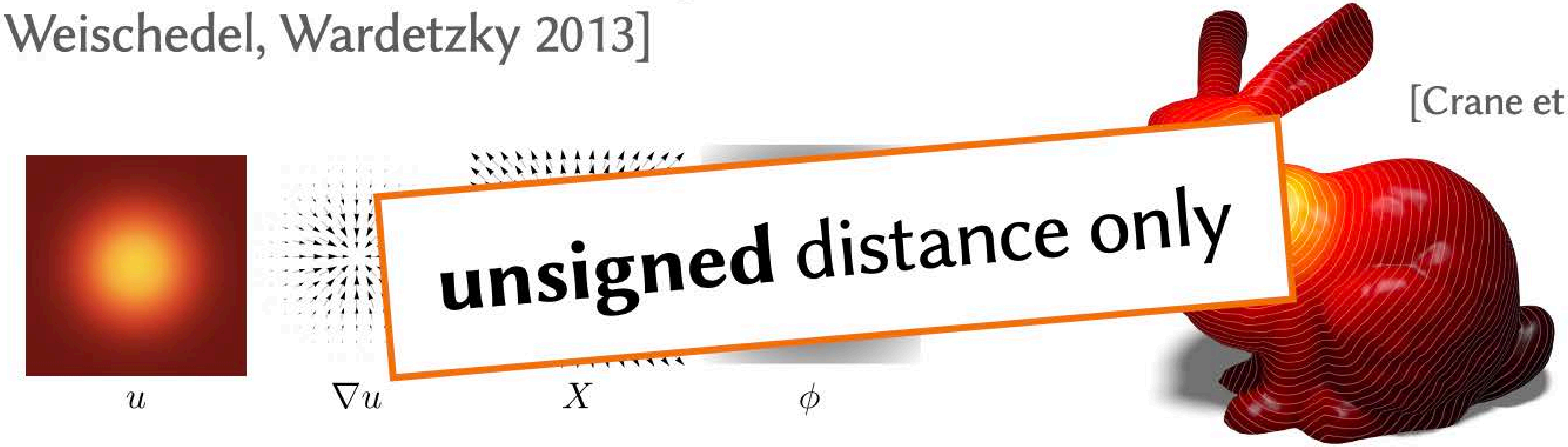
u ∇u X ϕ



[Crane et al. 2013]

Past work: heat methods in geometry processing

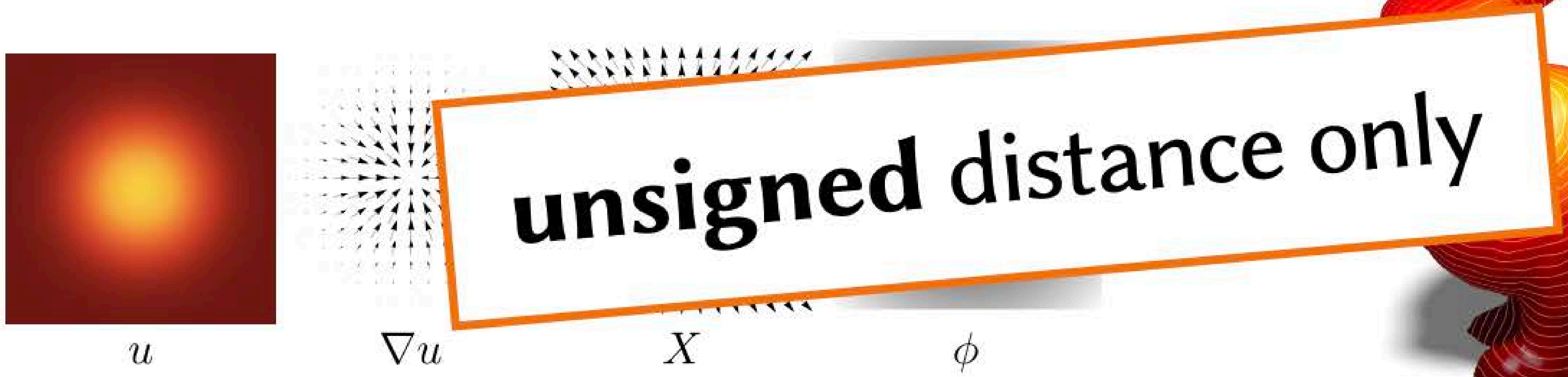
Unsigned Heat Method (UHM) for geodesic distance
[Crane, Weischedel, Wardetzky 2013]



[Crane et al. 2013]

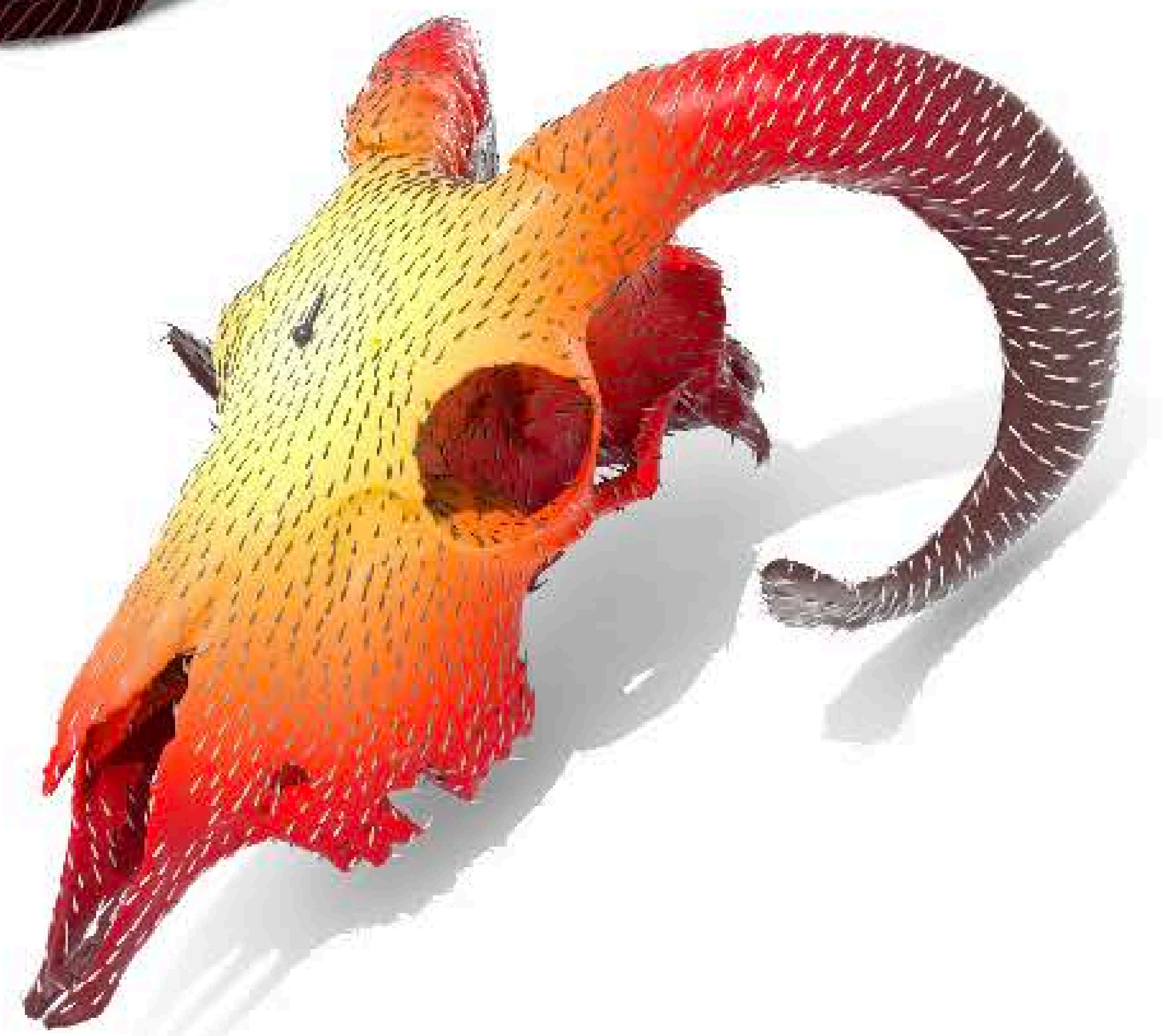
Past work: heat methods in geometry processing

Unsigned Heat Method (UHM) for geodesic distance
[Crane, Weischedel, Wardetzky 2013]



[Crane et al. 2013]

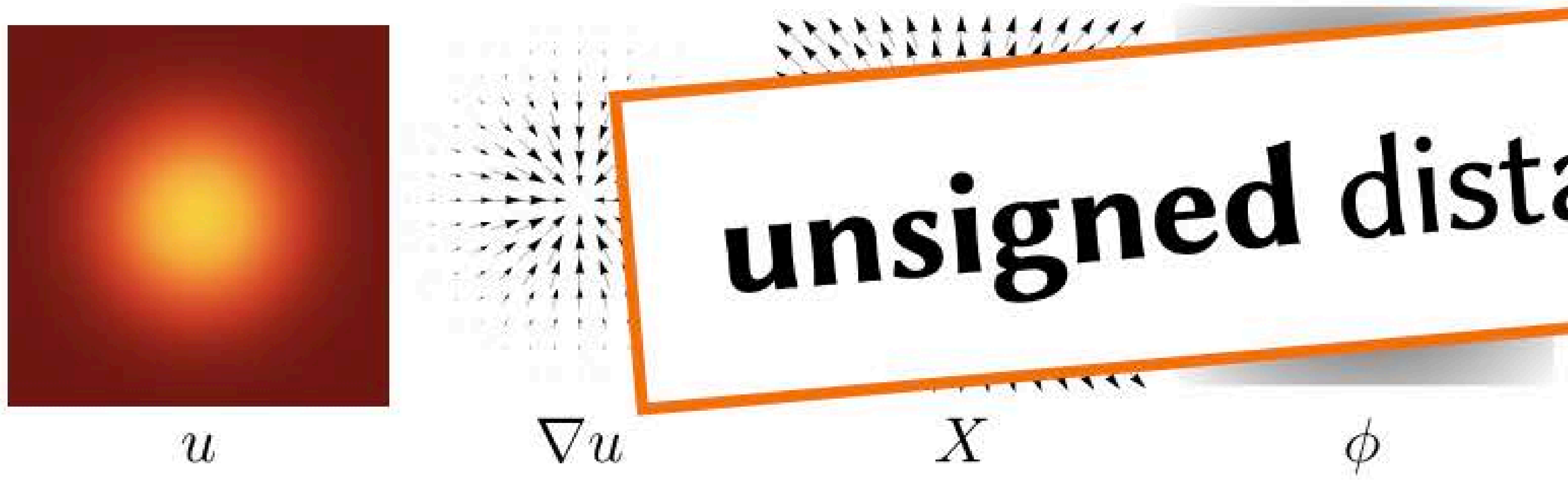
[Sharp et al. 2019]



Vector Heat Method (VHM) for parallel transport
[Sharp, Soliman, Crane 2019]

Past work: heat methods in geometry processing

Unsigned Heat Method (UHM) for geodesic distance
[Crane, Weischedel, Wardetzky 2013]

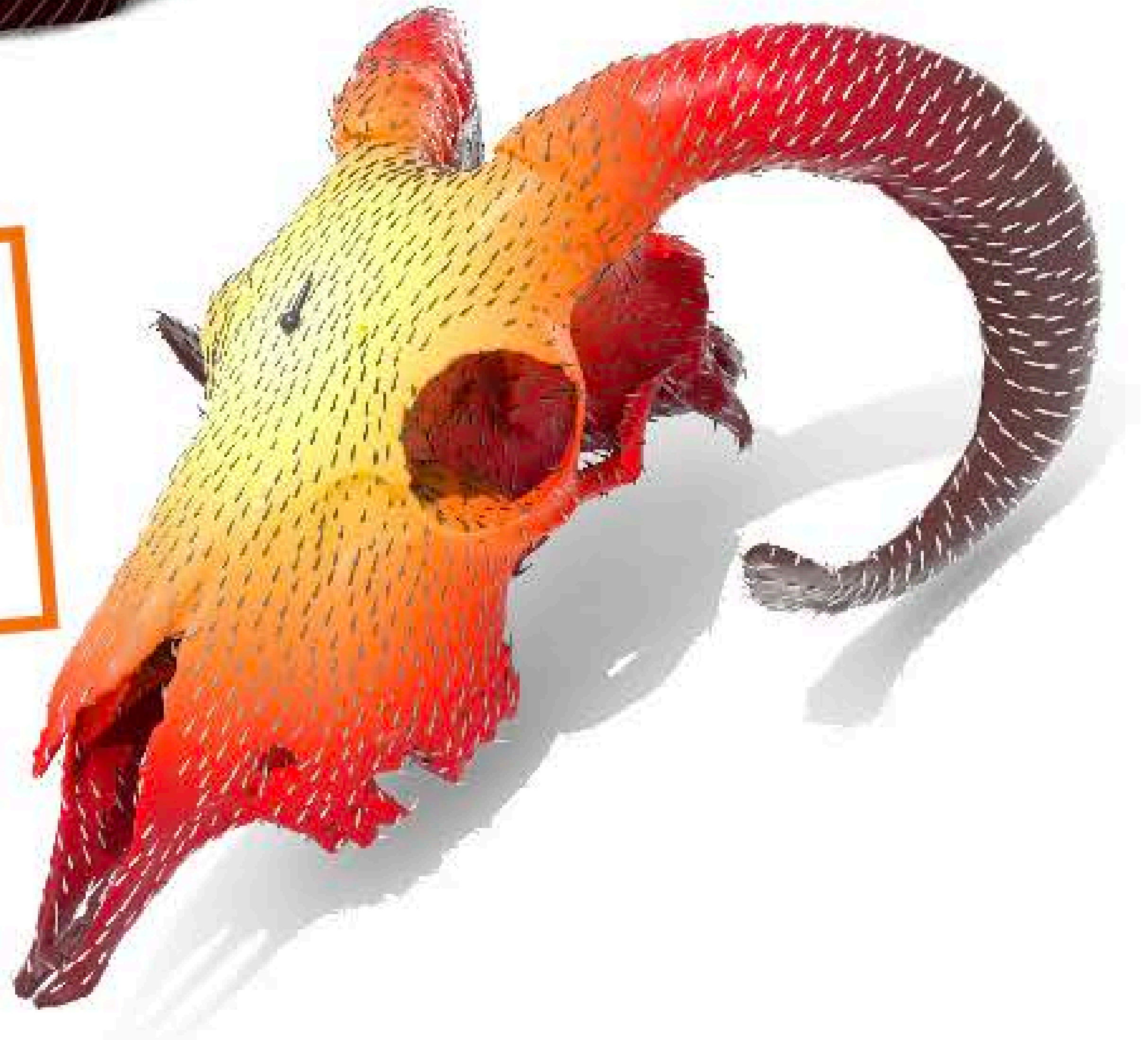


unsigned distance only

[Crane et al. 2013]

Vector Heat Method (VHM) for parallel transport
[Sharp, Soliman, Crane 2019]

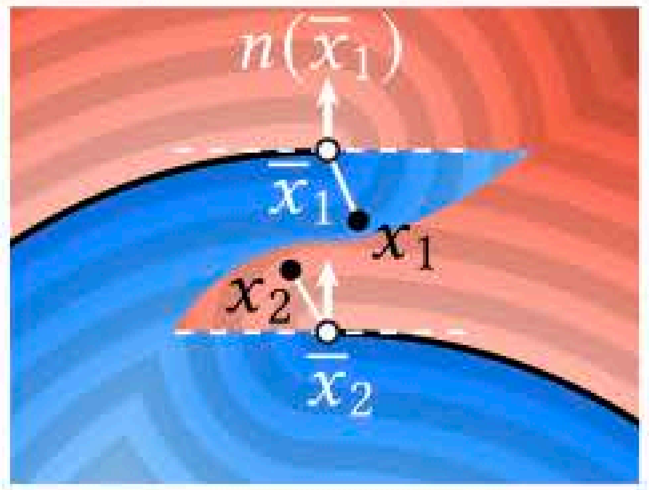
doesn’t compute signed distance



[Sharp et al. 2019]

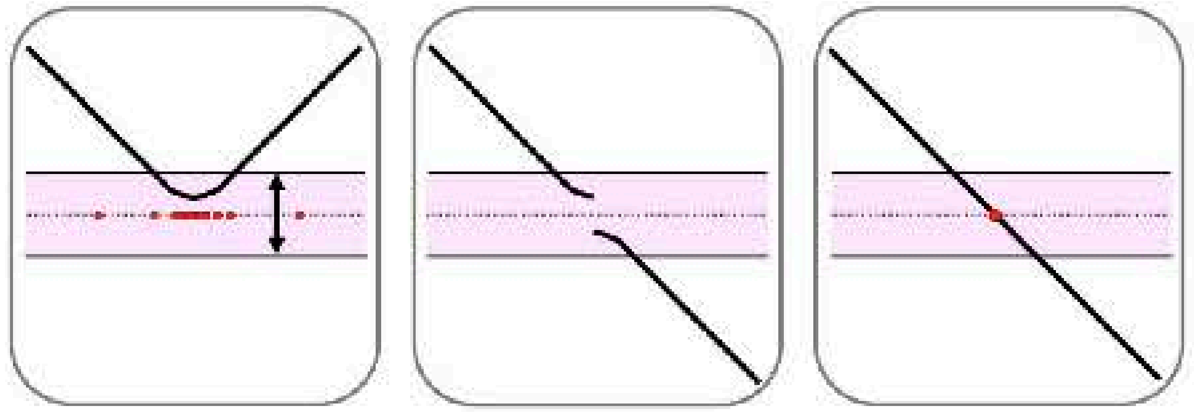
Past work in robust distance

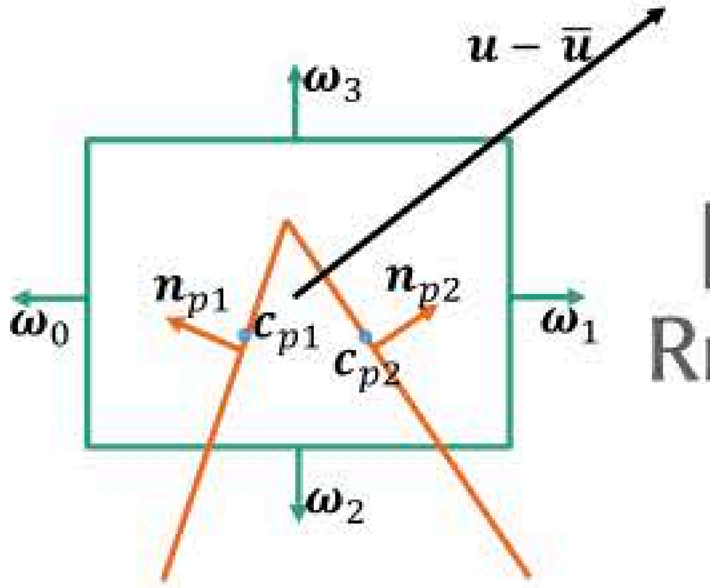
Pseudonormal distance → *not robust*
[Bærentzen 2005]



Displaced Signed Distance → *pseudonormal-like*
[Brunton & Rmaileh 2021]

[Mullen et al. 2010]

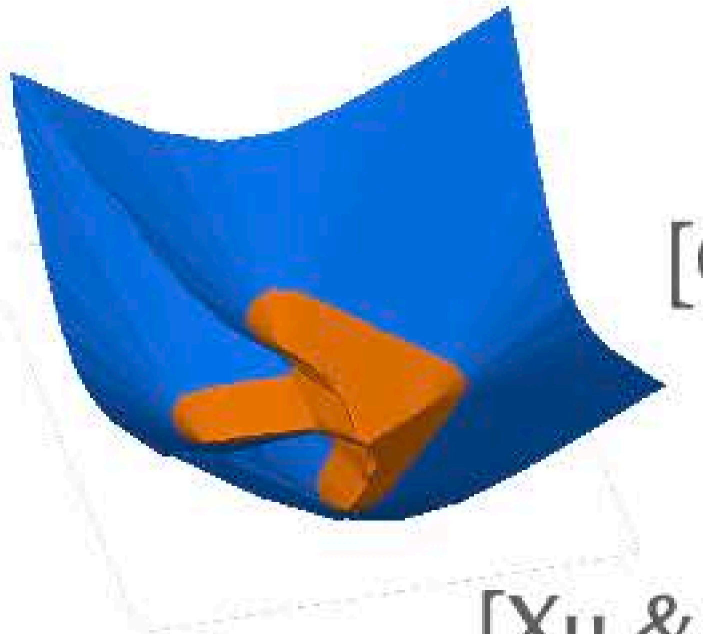




[Brunton & Rmaileh 2021]

Signing unsigned distance → *Euclidean only*
[Mullen et al. 2010]

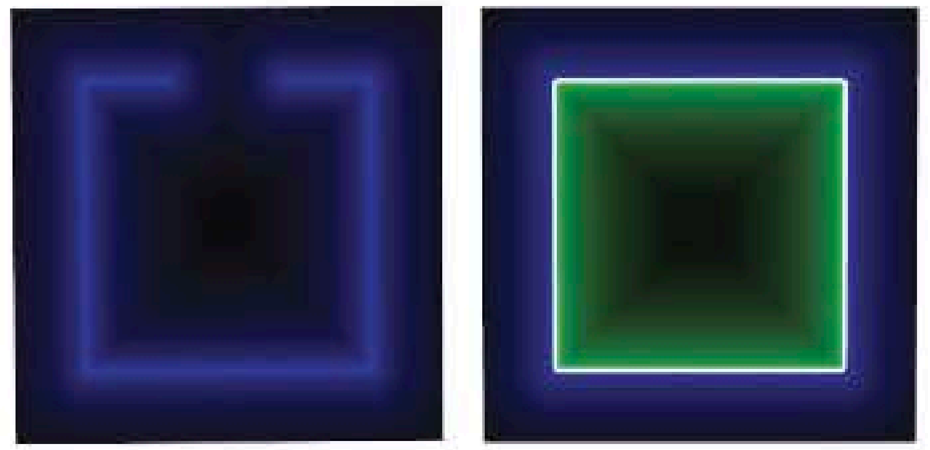
Smooth Signed Distance → *not true distance*
[Calakli & Taubin 2011]



[Calakli & Taubin 2011]

“Heal” gaps with morphological fusing → *over-regularized*
[Xu & Barbič 2014]

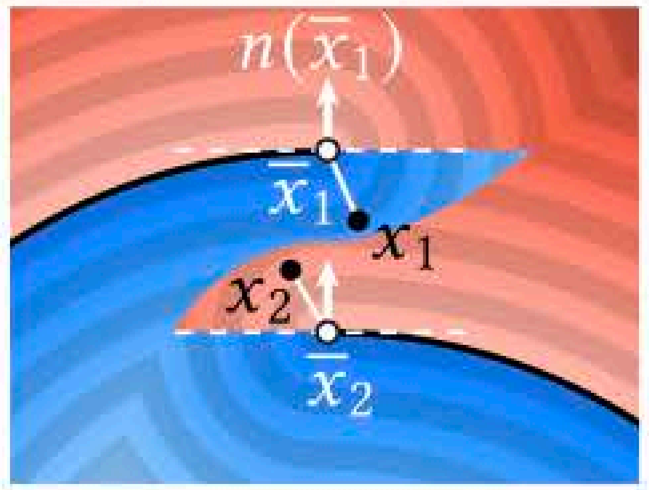
[Xu & Barbič 2014]



Neural “distance” functions → *not true distance*
[Park et al. 2019; Atzmon and Lipman 2019; Gropp et al. 2020]

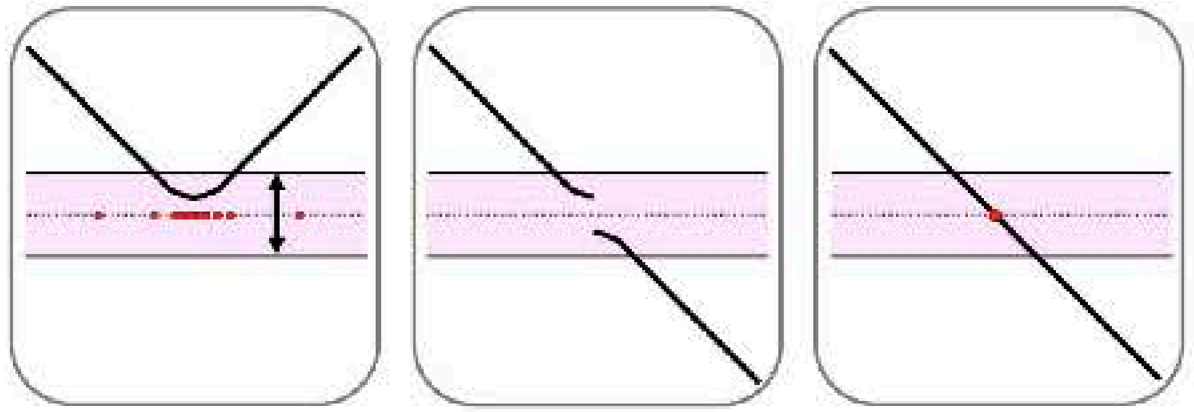
Past work in robust distance

Pseudonormal distance $\longrightarrow$ *not robust*
[Bærentzen 2005]



Displaced Signed Distance $\longrightarrow$ *pseudonormal-like*
[Brunton & Rmaileh 2021]

[Mullen et al. 2010]

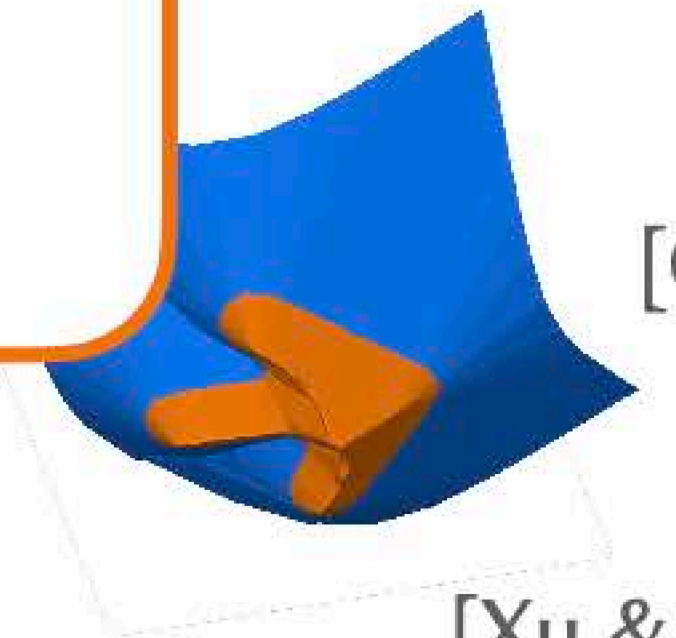


Signing unsigned dis
[Mullen et al. 2010]

runton &
aileh 2021]

Still no go-to method for robust signed distance…

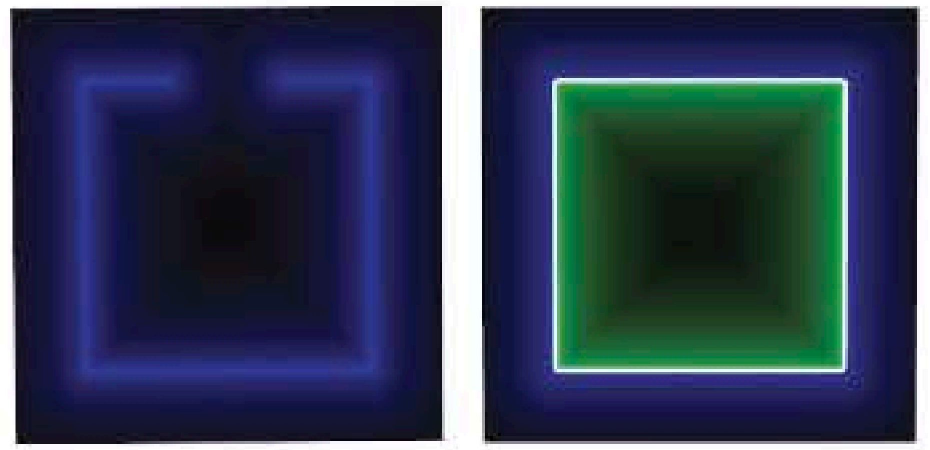
Smooth Signed Dista
[Calakli & Taubin 2011



[Calakli & Taubin 2011]

"Heal" gaps with morphological fusing $\longrightarrow$ *over-regularized*
[Xu & Barbič 2014]

[Xu & Barbič 2014]

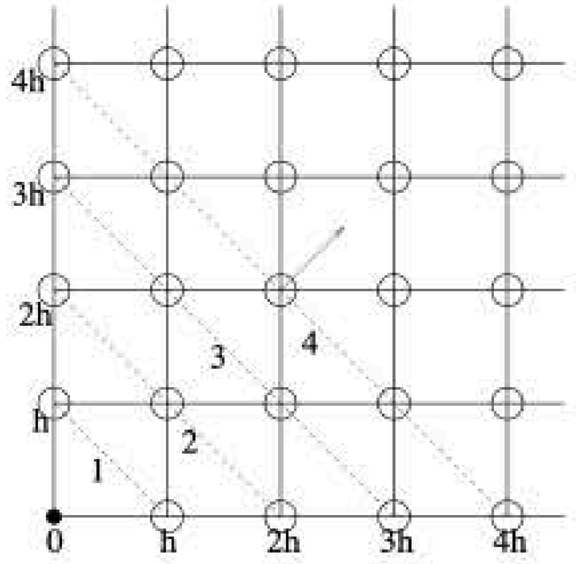


Neural "distance" functions $\longrightarrow$ *not true distance*
[Park et al. 2019; Atzmon and Lipman 2019; Gropp et al. 2020]

Many <u>unsigned</u> geodesic distance algorithms...

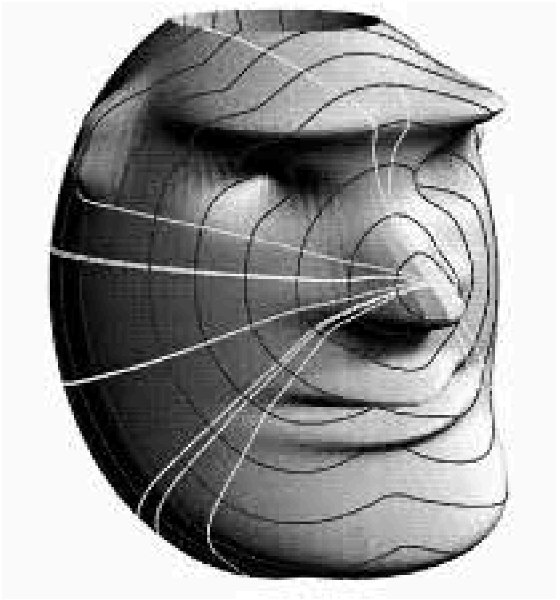
fast sweeping

[Zhao 2005]

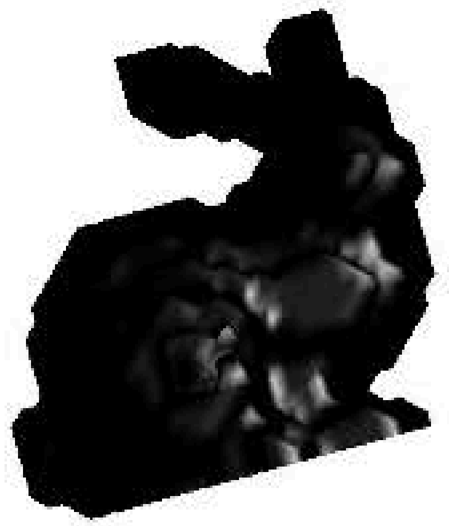


fast marching & wave-based

[Kimmel & Sethian 1998]

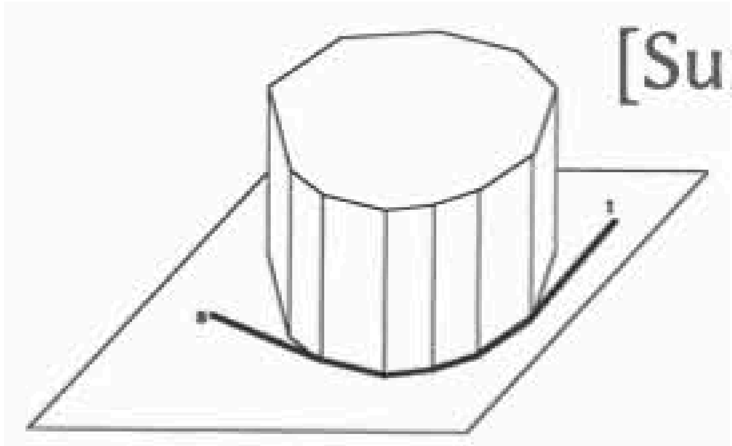


[Gurumoorthy & Rangarajan 2009]

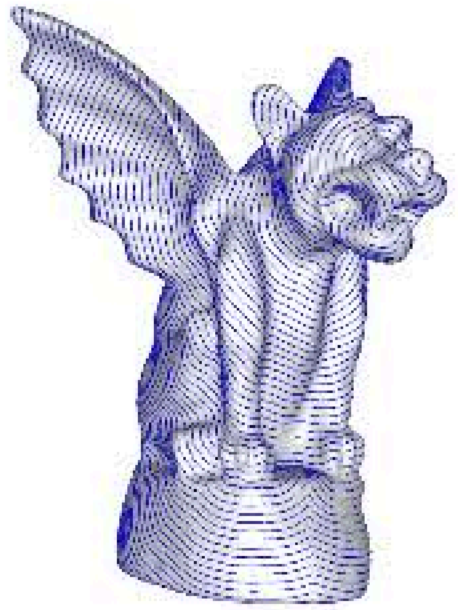


window-based methods

[Mitchell et al. 1987]



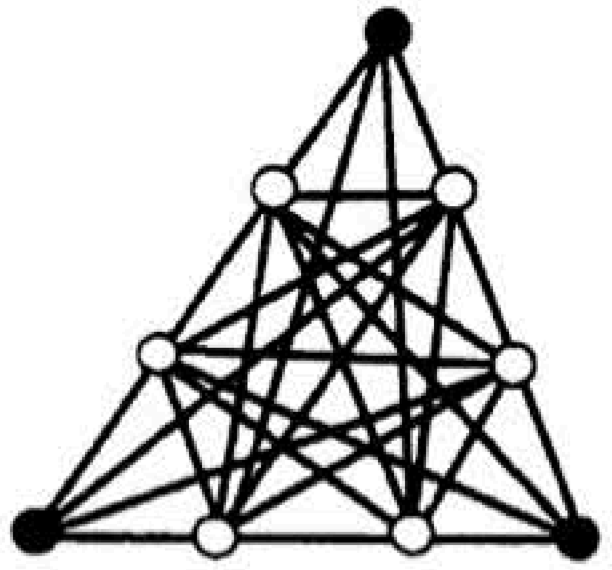
[Surazhsky et al. 2005]



... and many more...

graph-based

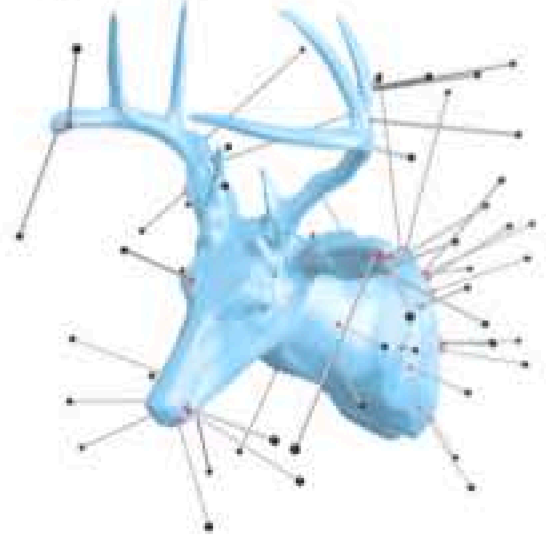
[Lanthier 1999]



closest-point queries

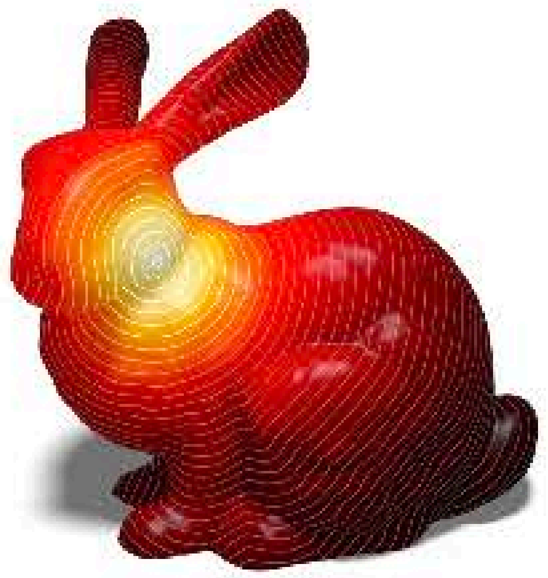
[Sawhney 2021]

[Sharp & Jacobson 2022]



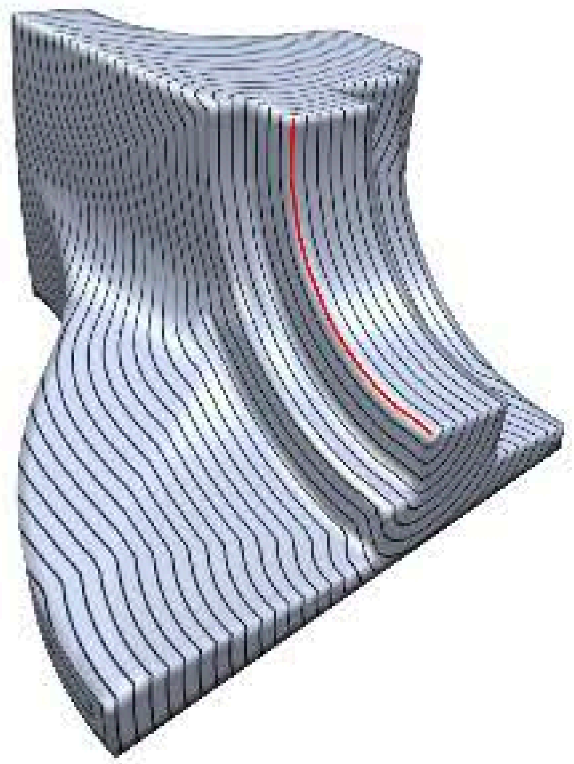
diffusion-based

[Crane et al. 2013]

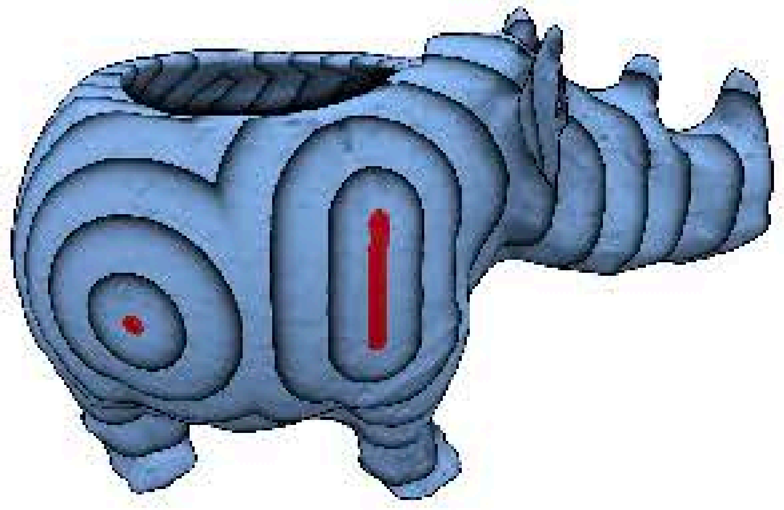


virtual source propagation

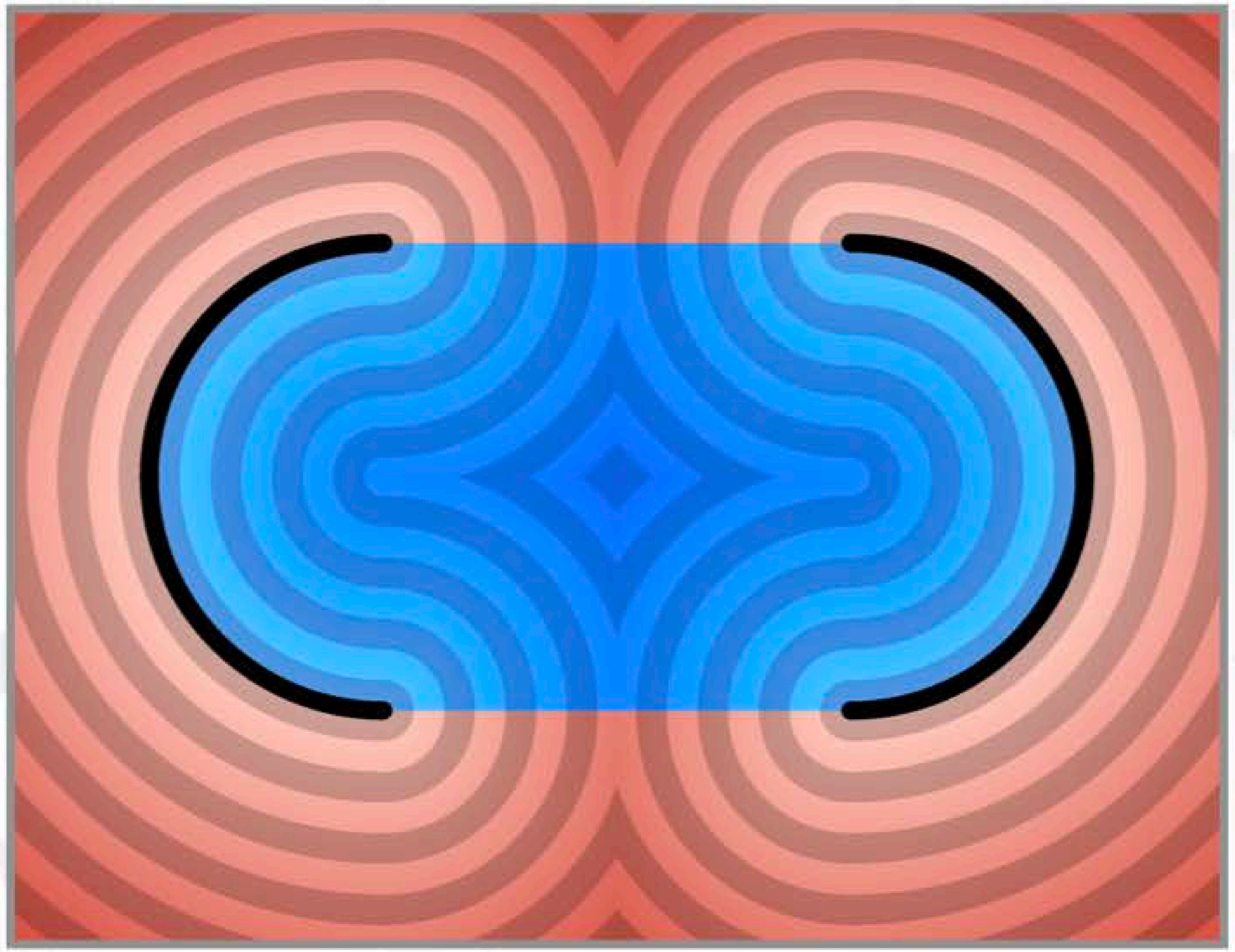
[Bommes & Kobbelt 2007]



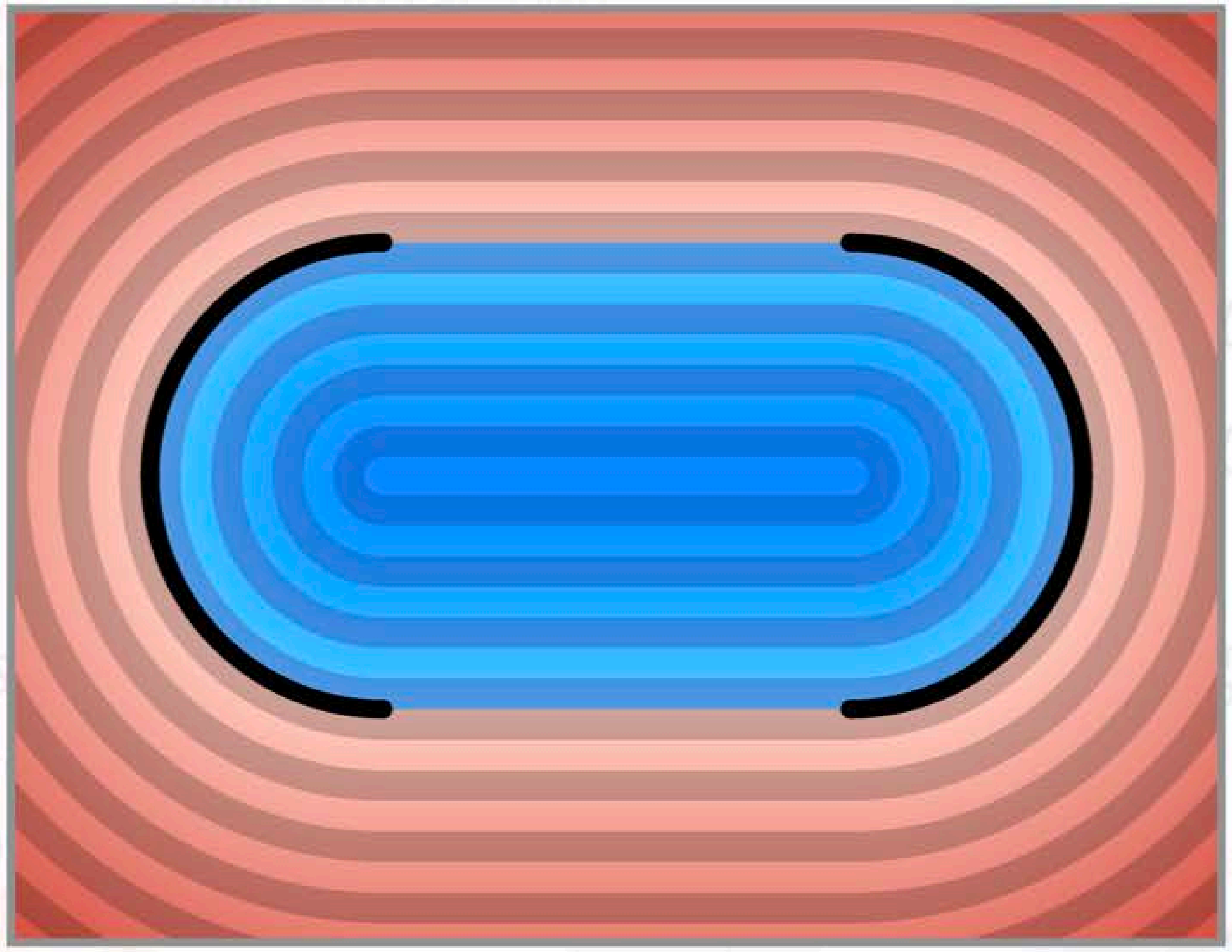
[Trettner et al. 2021]



Signing unsigned distance doesn't work



signed unsigned distance

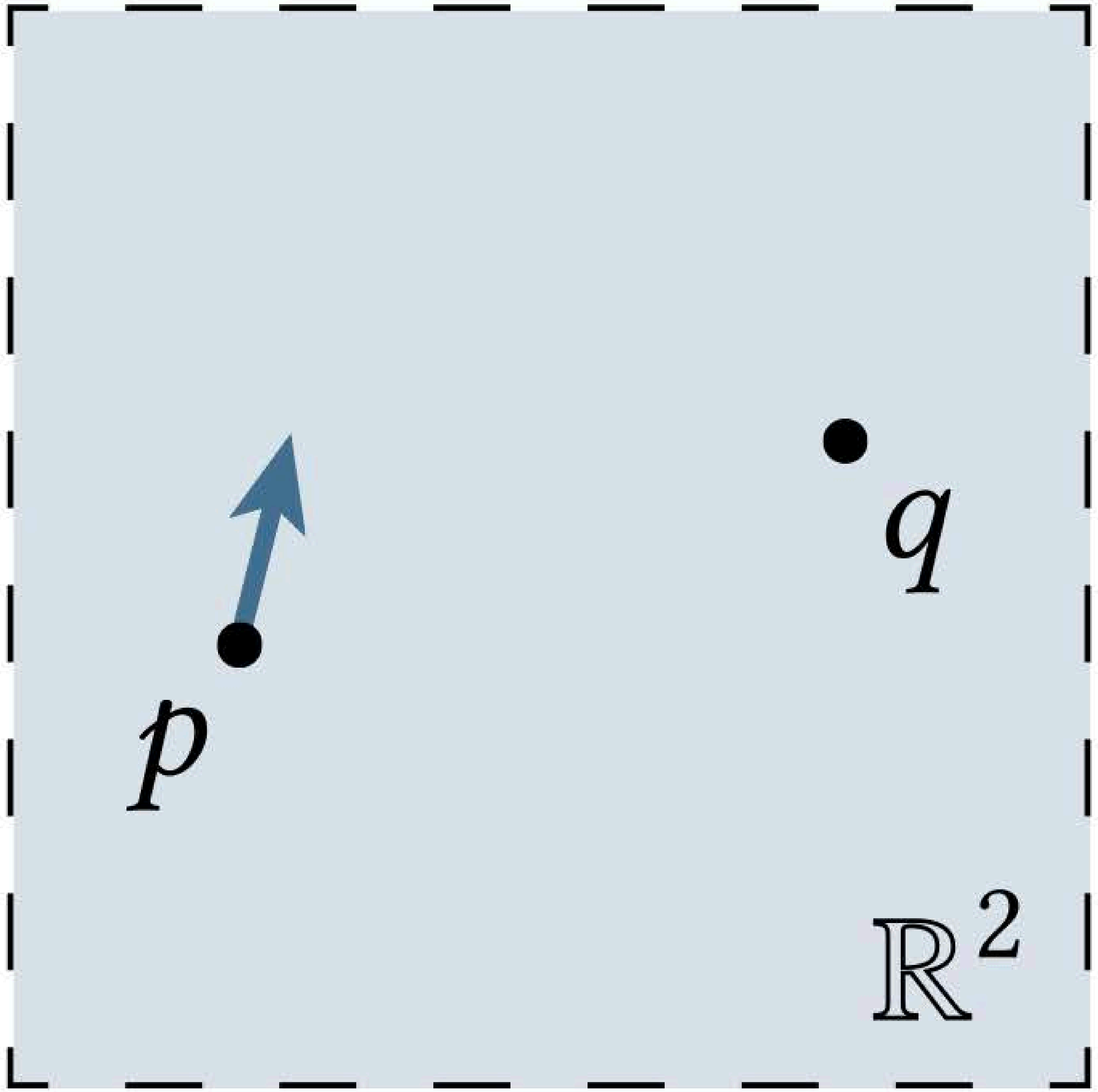


ground truth

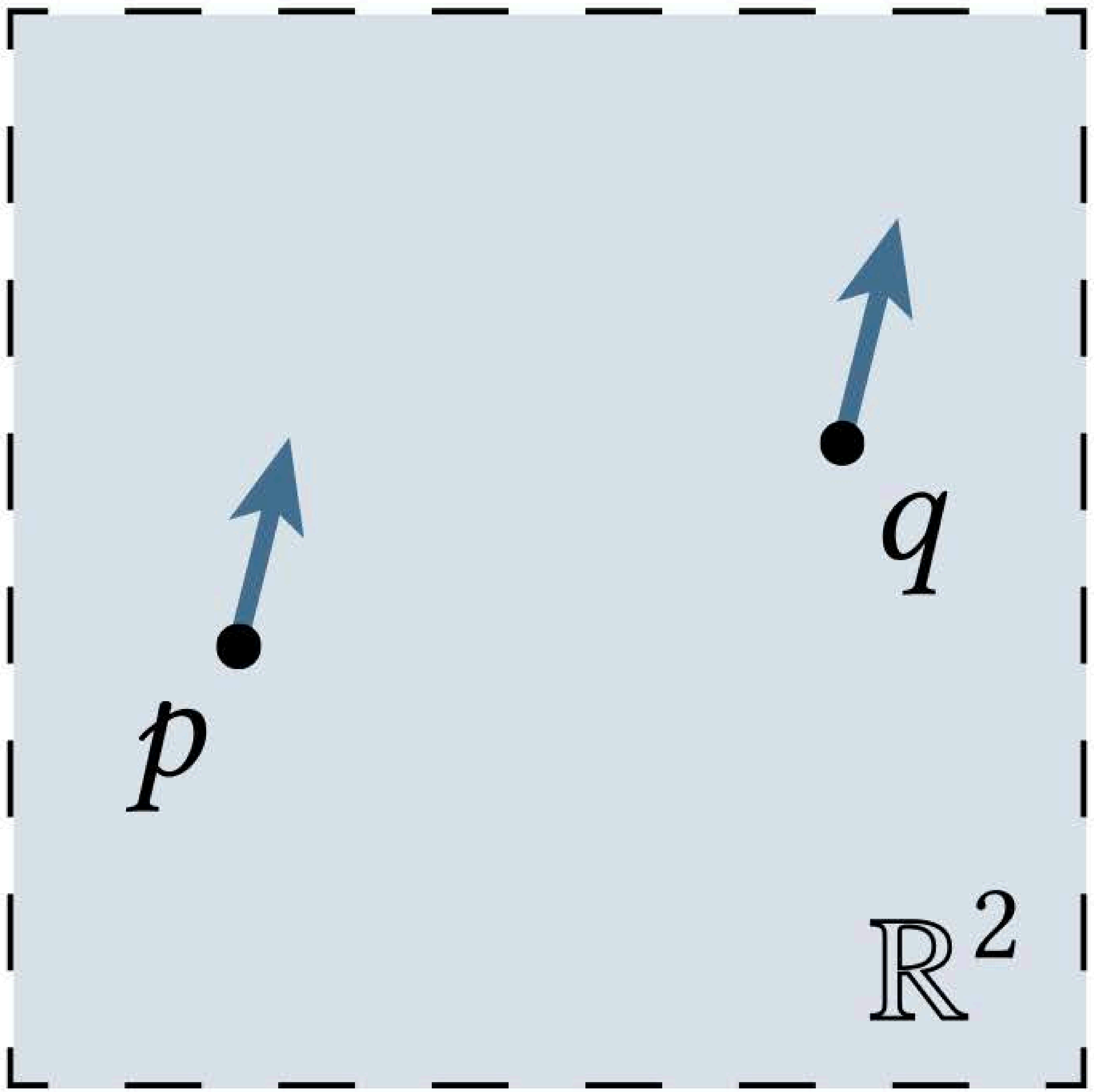
ALGORITHM

Parallel transport along shortest geodesics

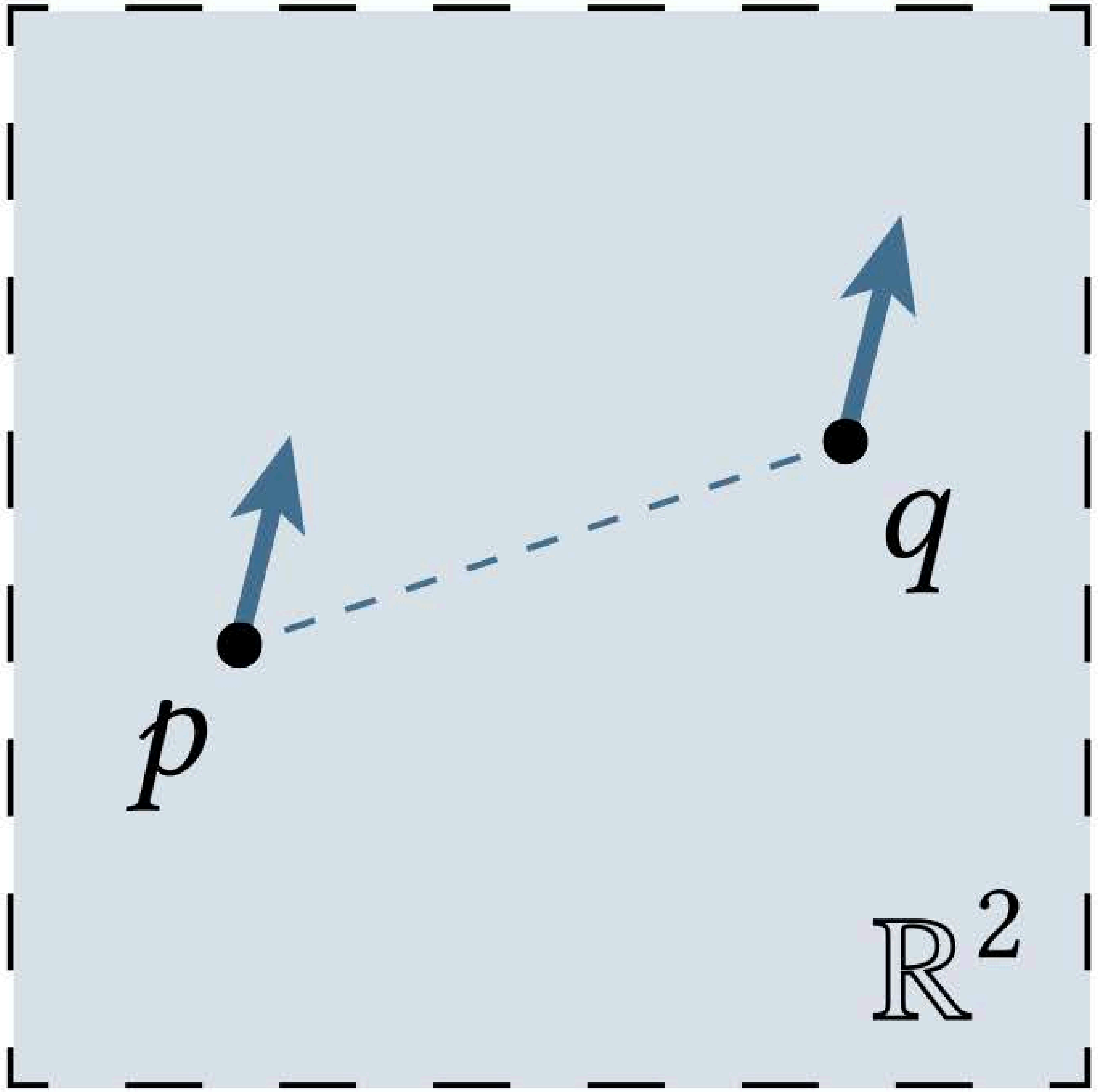
Parallel transport along shortest geodesics



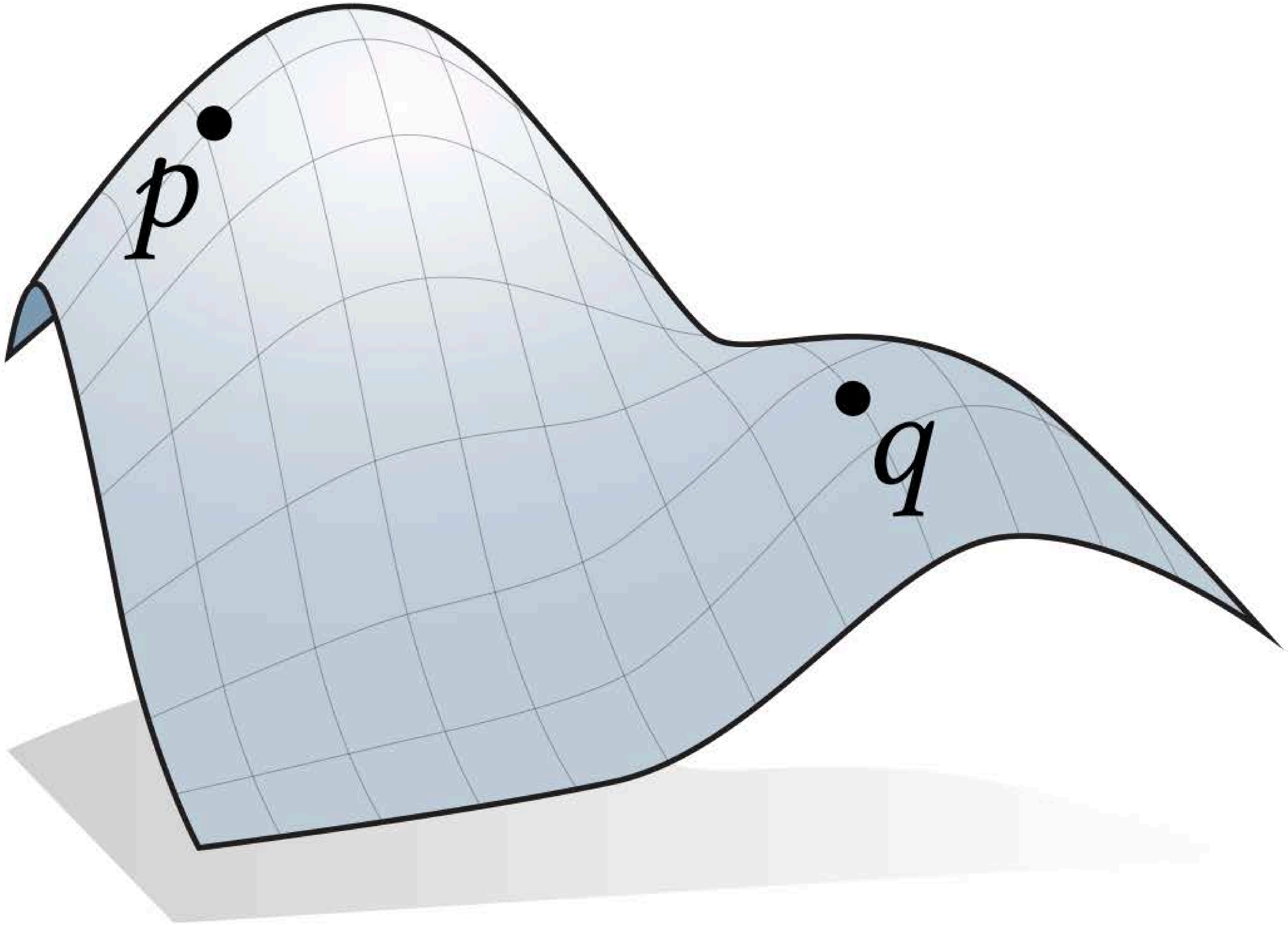
Parallel transport along shortest geodesics



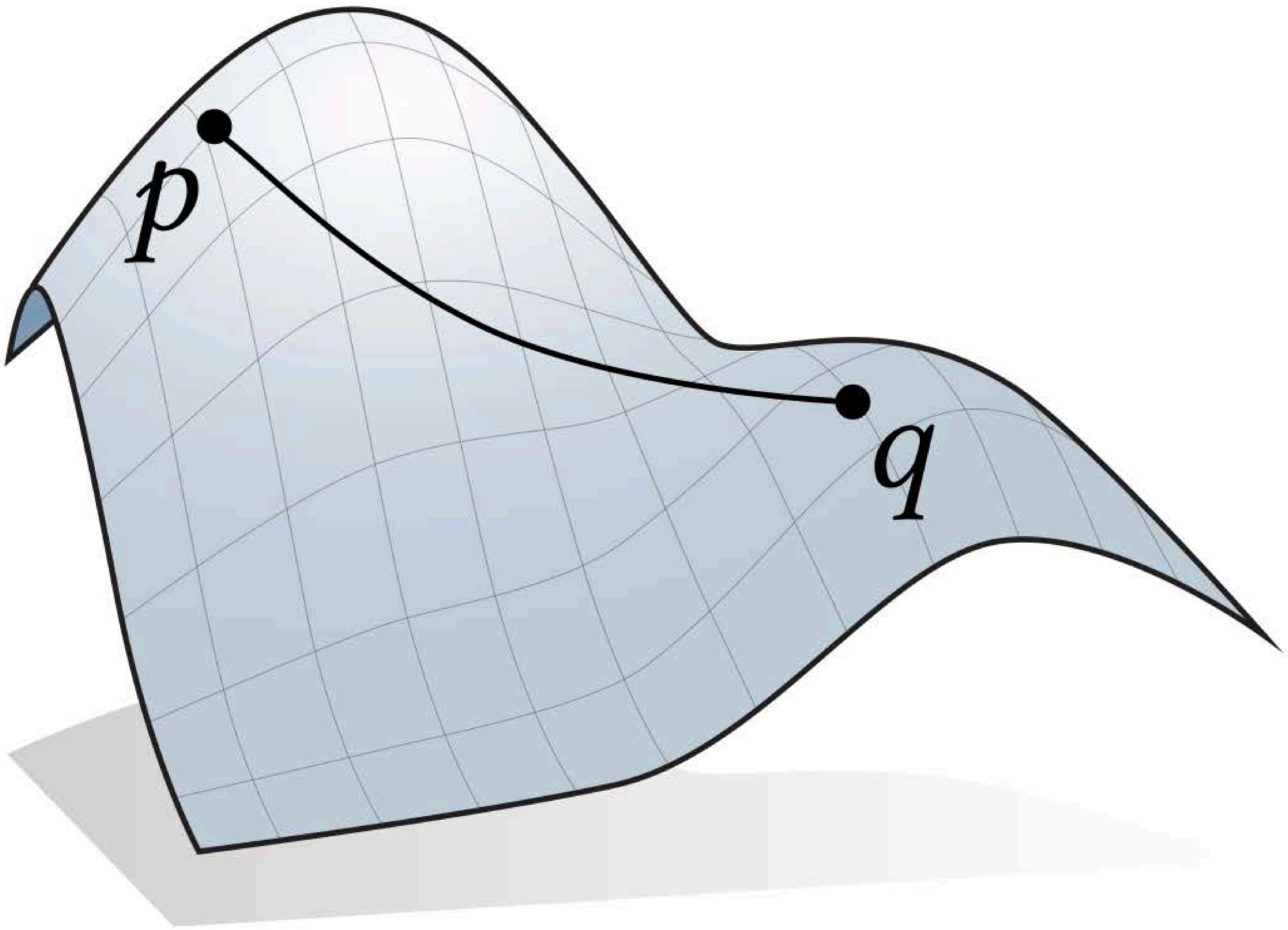
Parallel transport along shortest geodesics



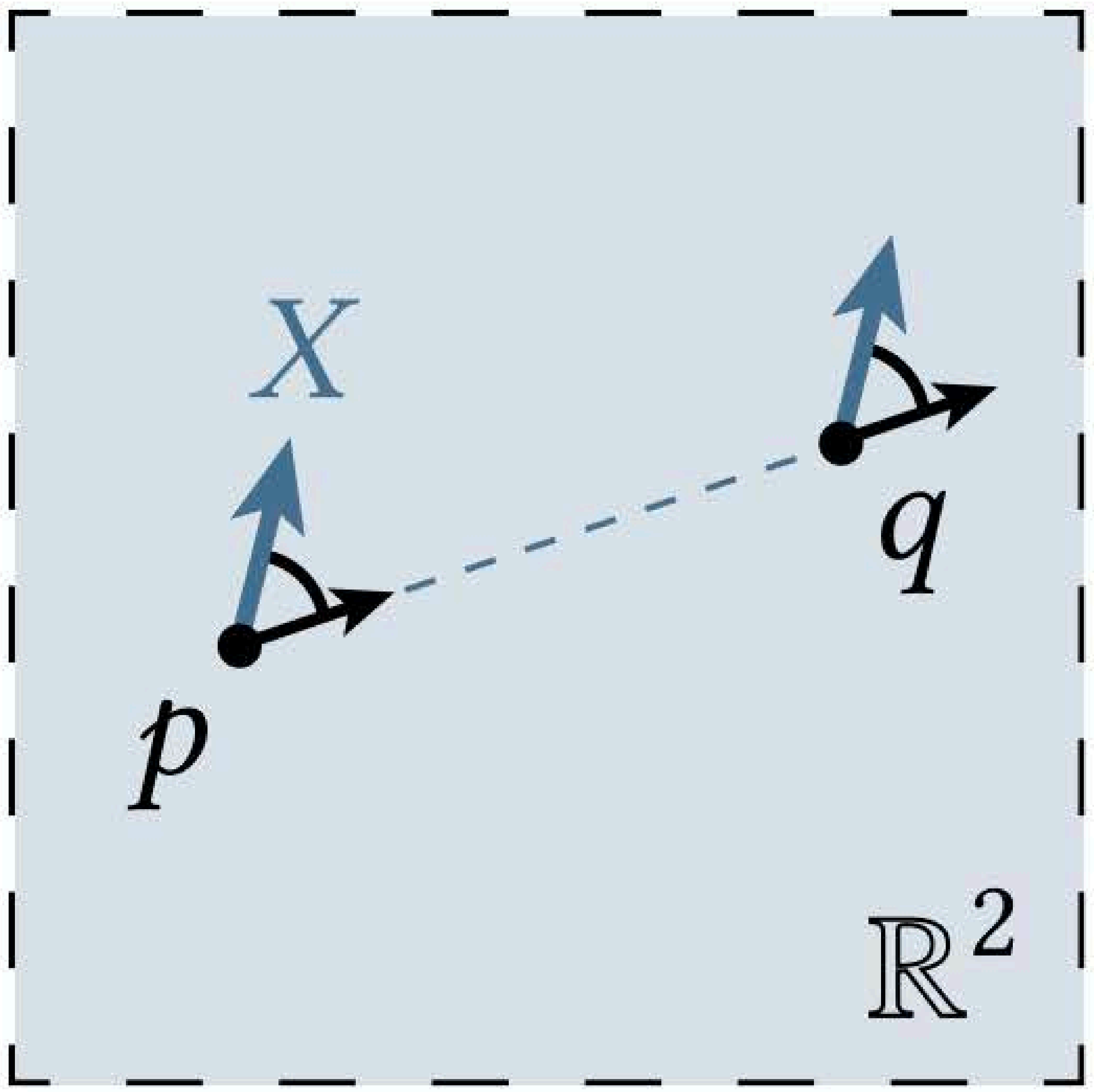
Parallel transport on surfaces

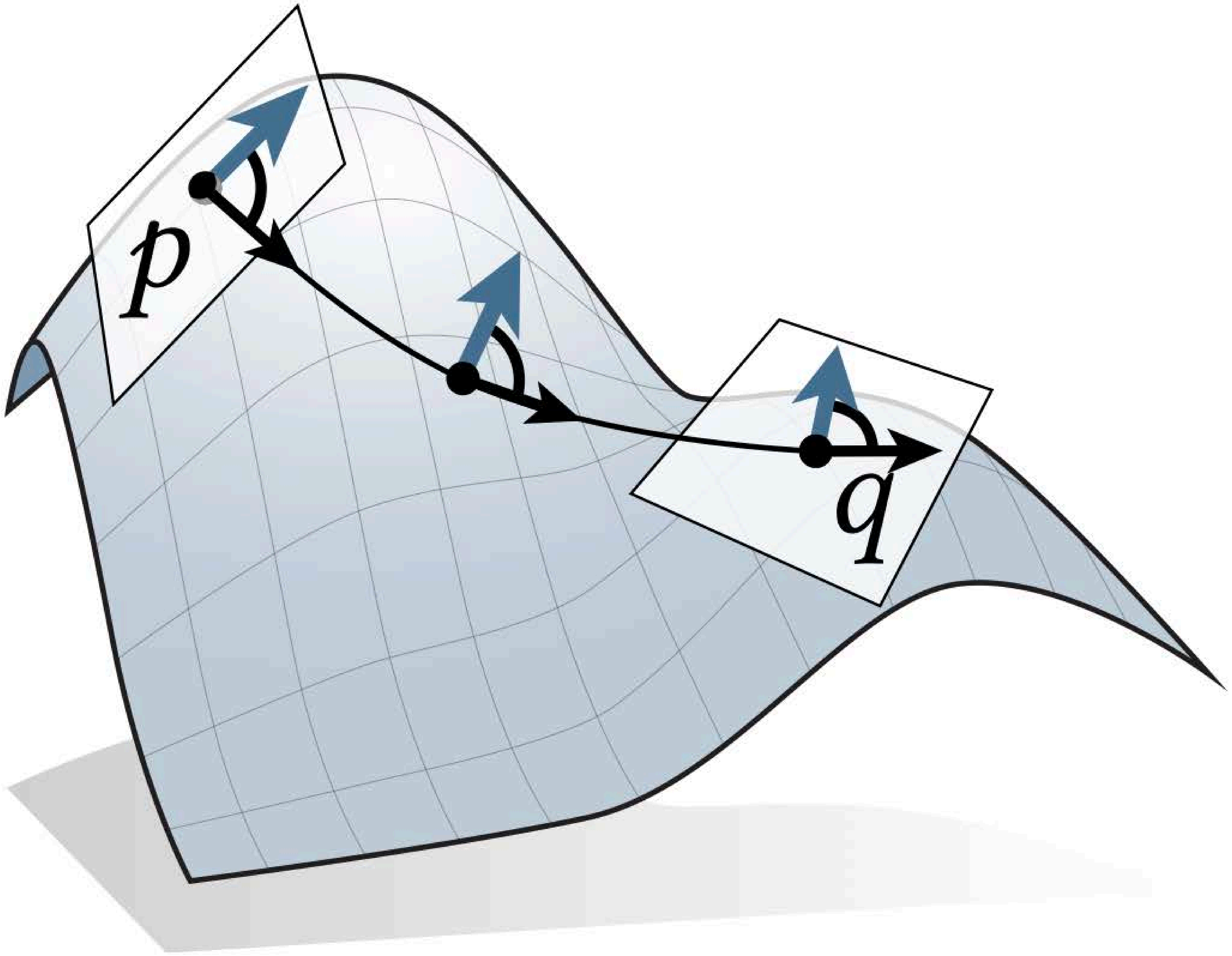


Parallel transport on surfaces



Parallel transport on surfaces



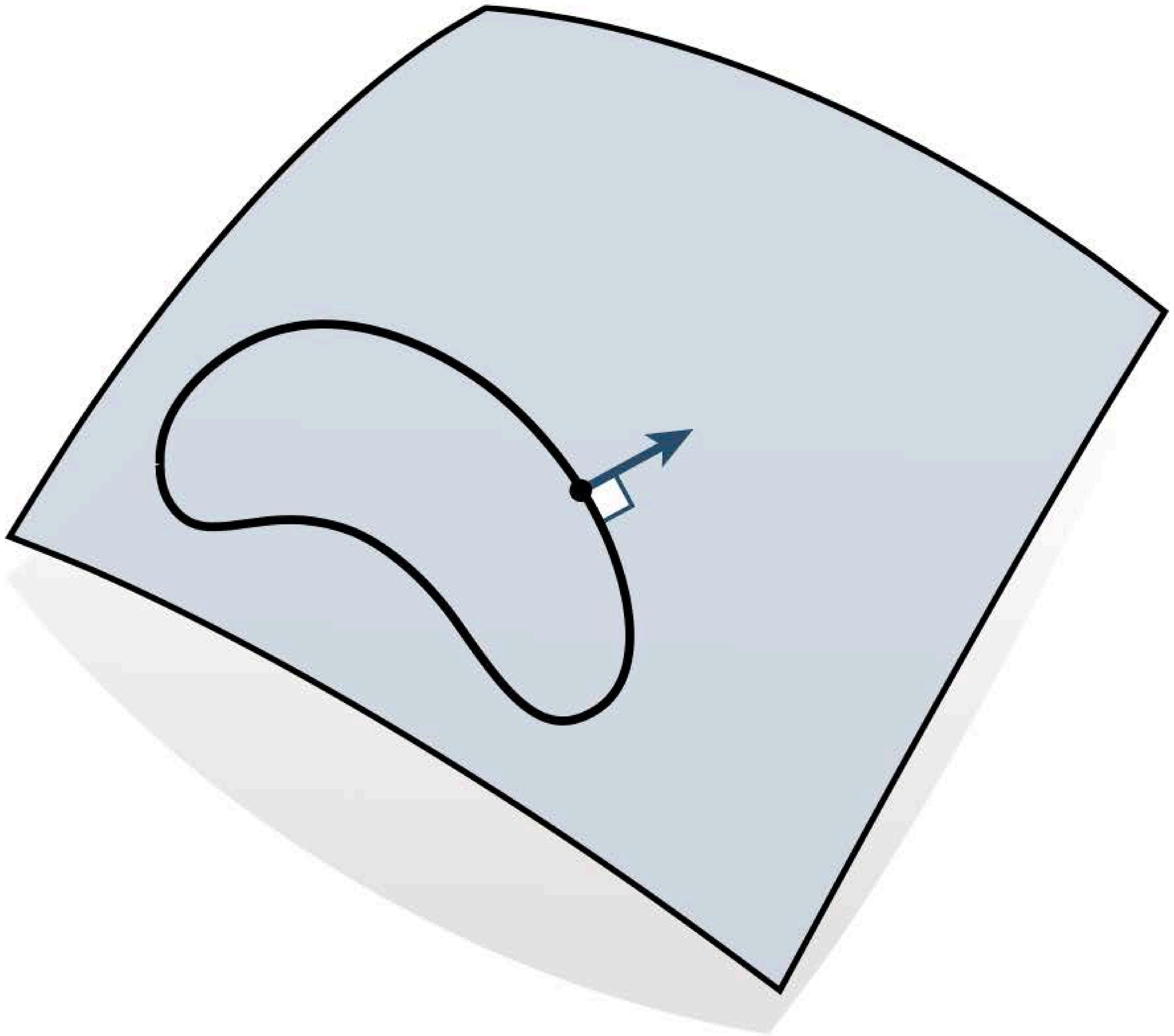


Parallel transport can be computed using diffusion

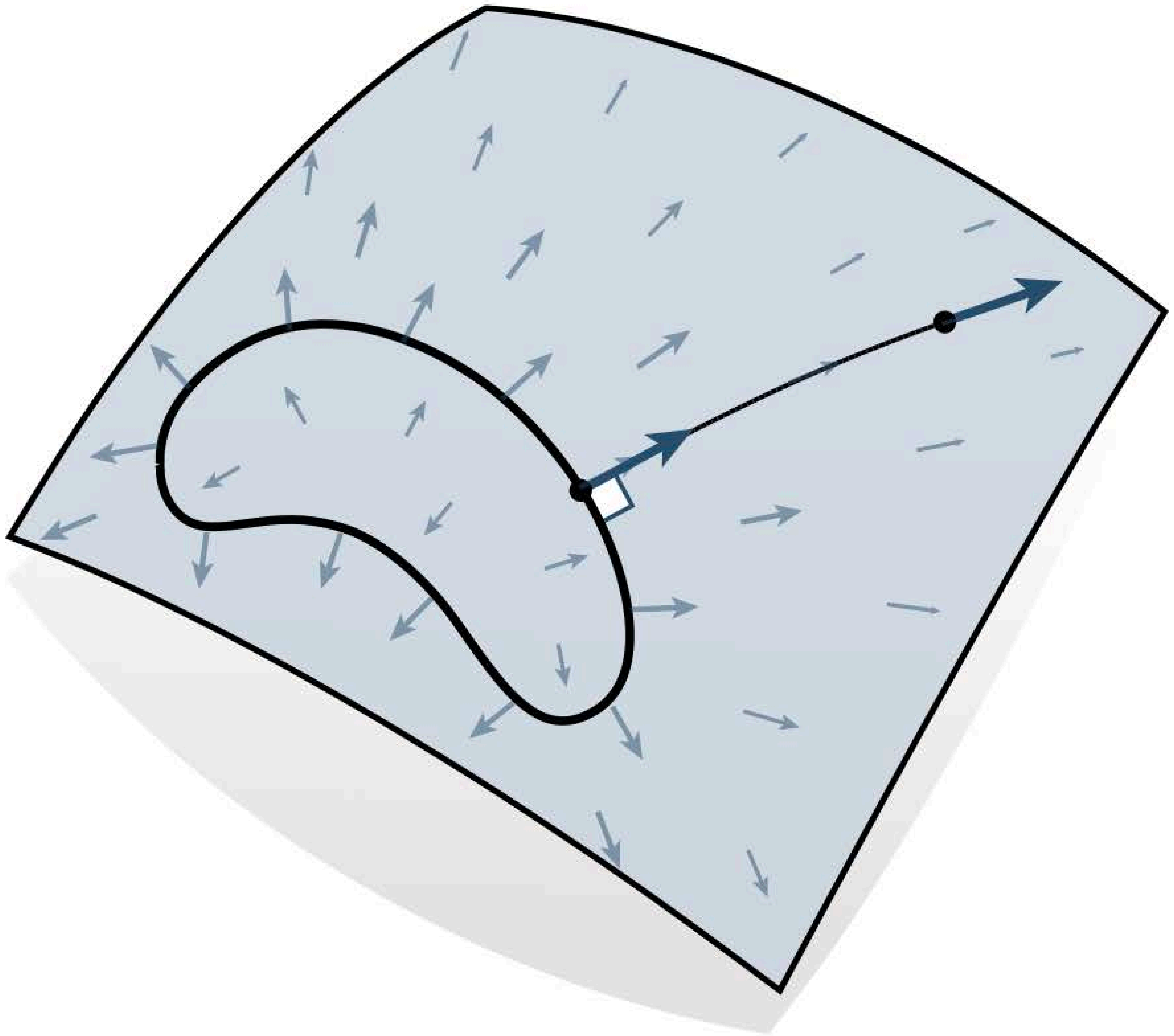
Theorem. Vector heat diffusion yields parallel transport along shortest geodesics, as diffusion time $\to 0$.

N. Berline, E. Getzler, M. Vergne, *Heat Kernels and Dirac Operators* (1992)

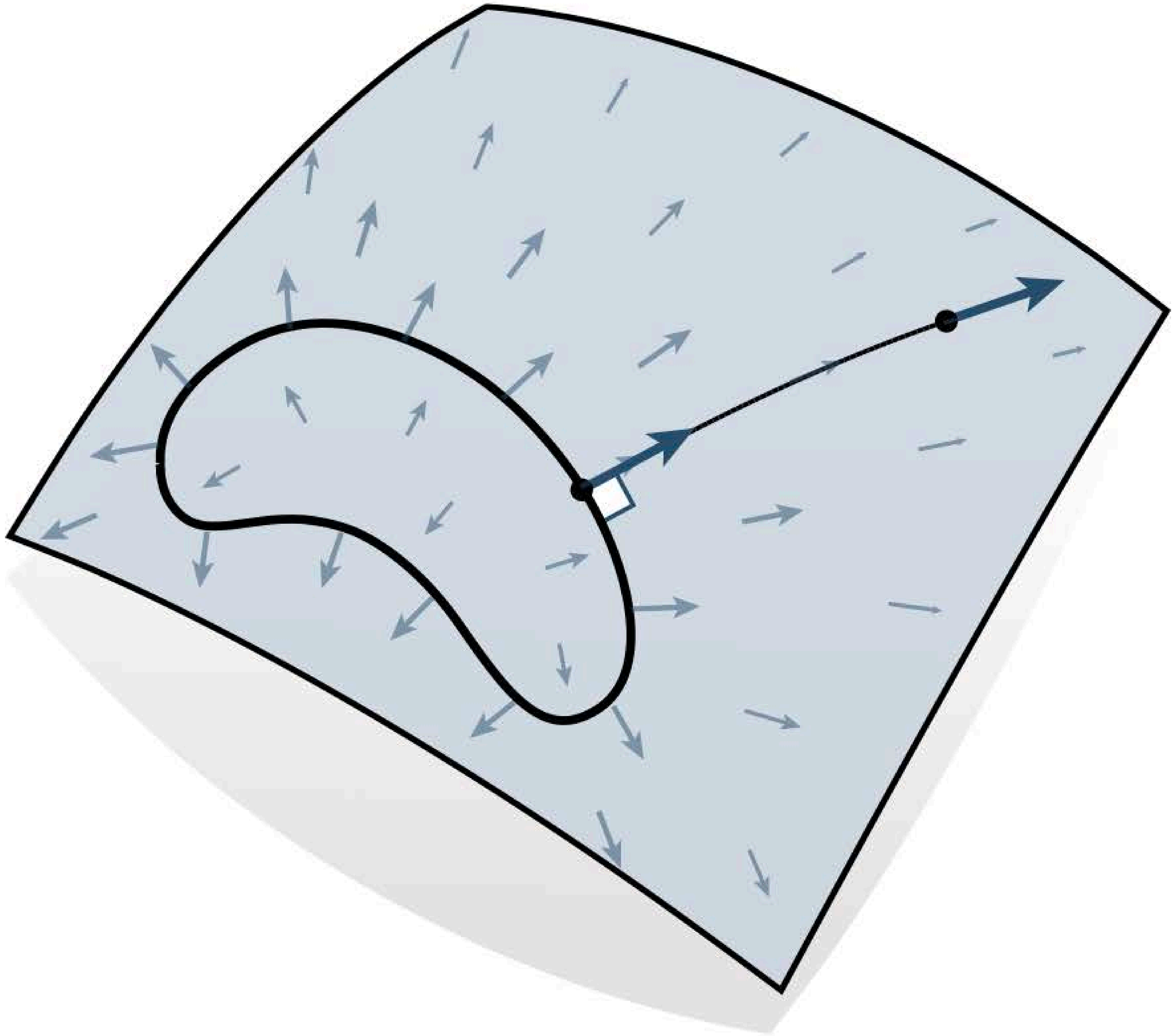
Key insight #1



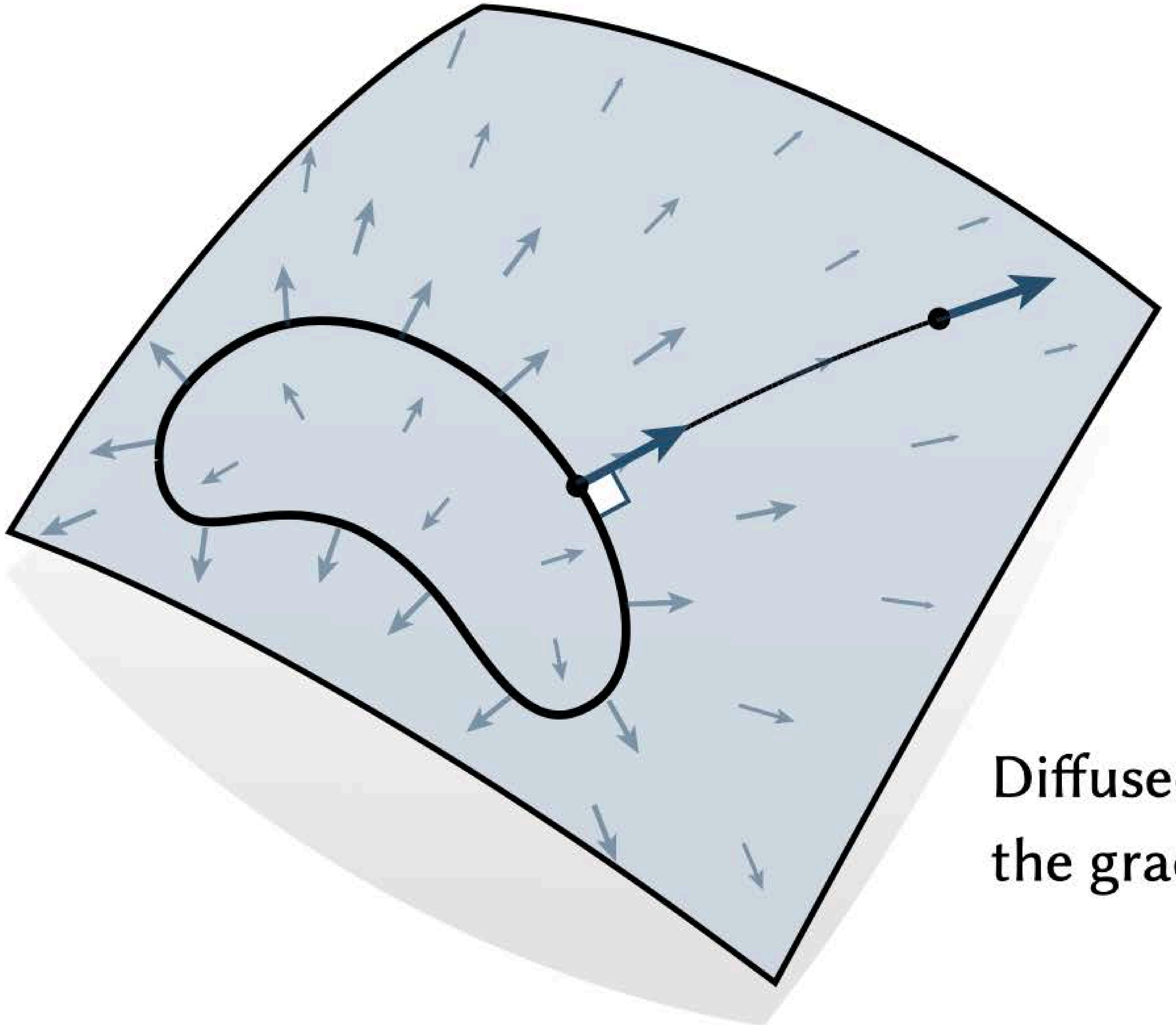
Key insight #1



Key insight #1

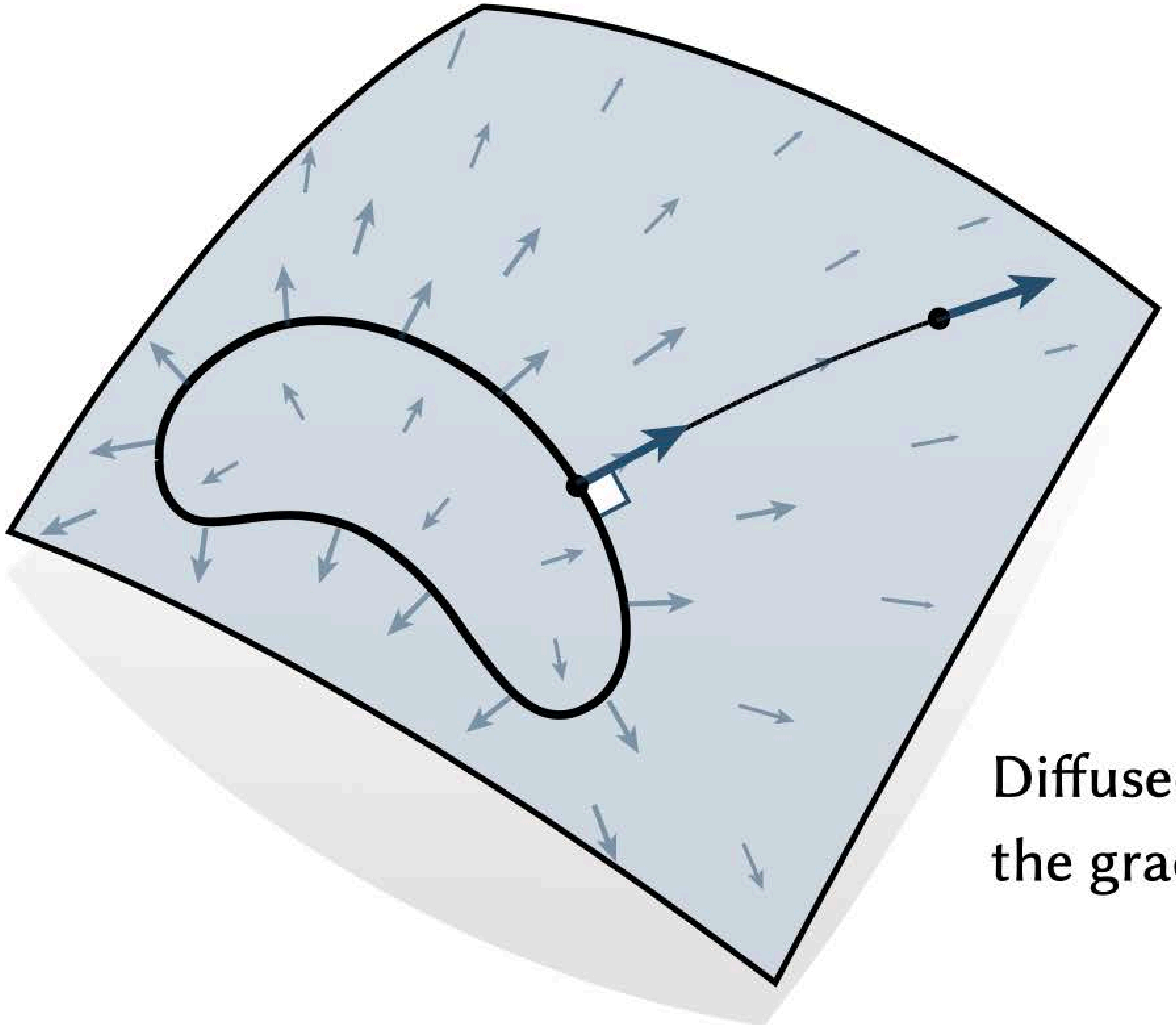


Key insight #1



Diffused vectors are parallel to the gradient of distance.

Key insight #1

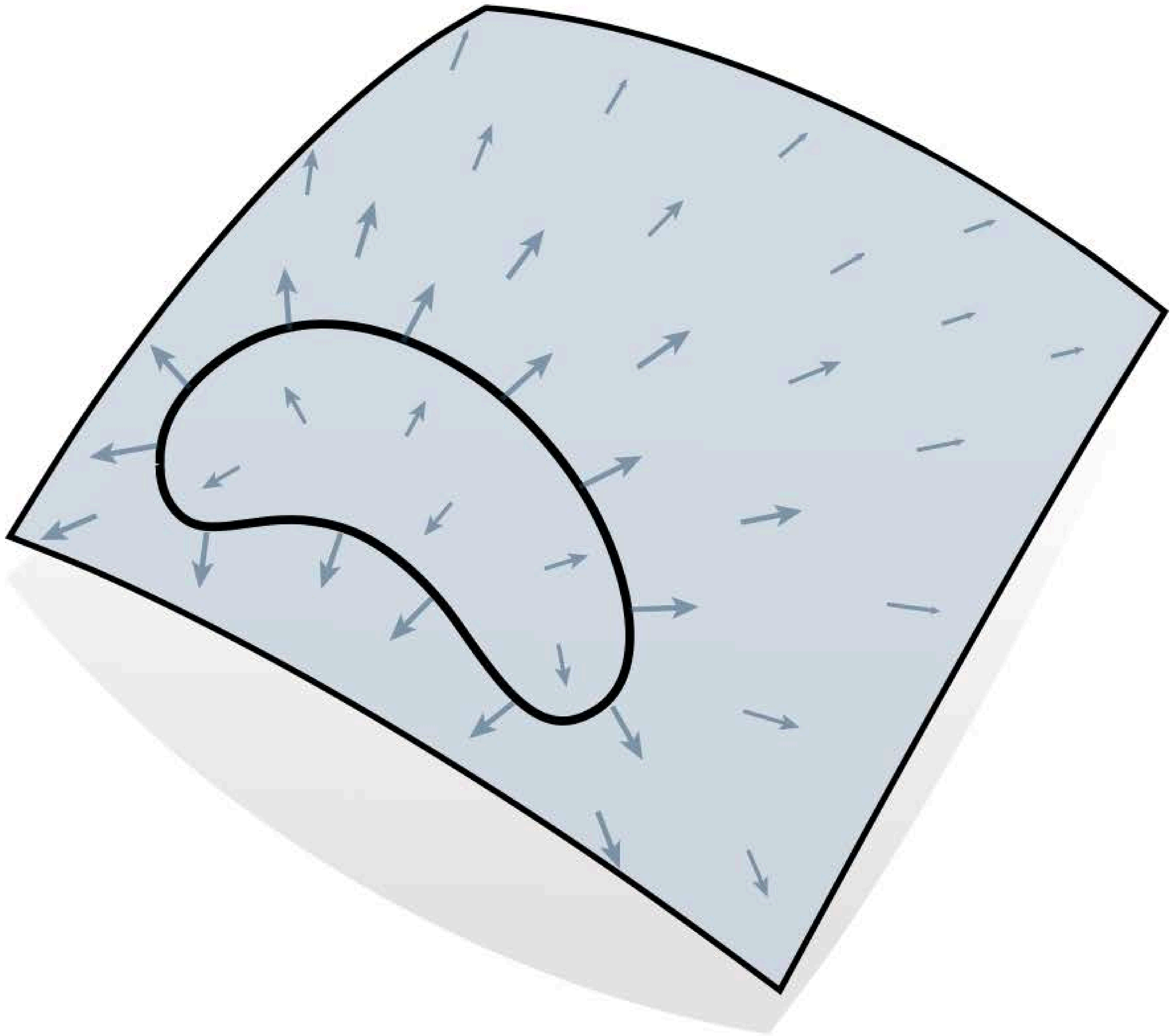


Diffused vectors are parallel to the gradient of <u>signed</u> distance.

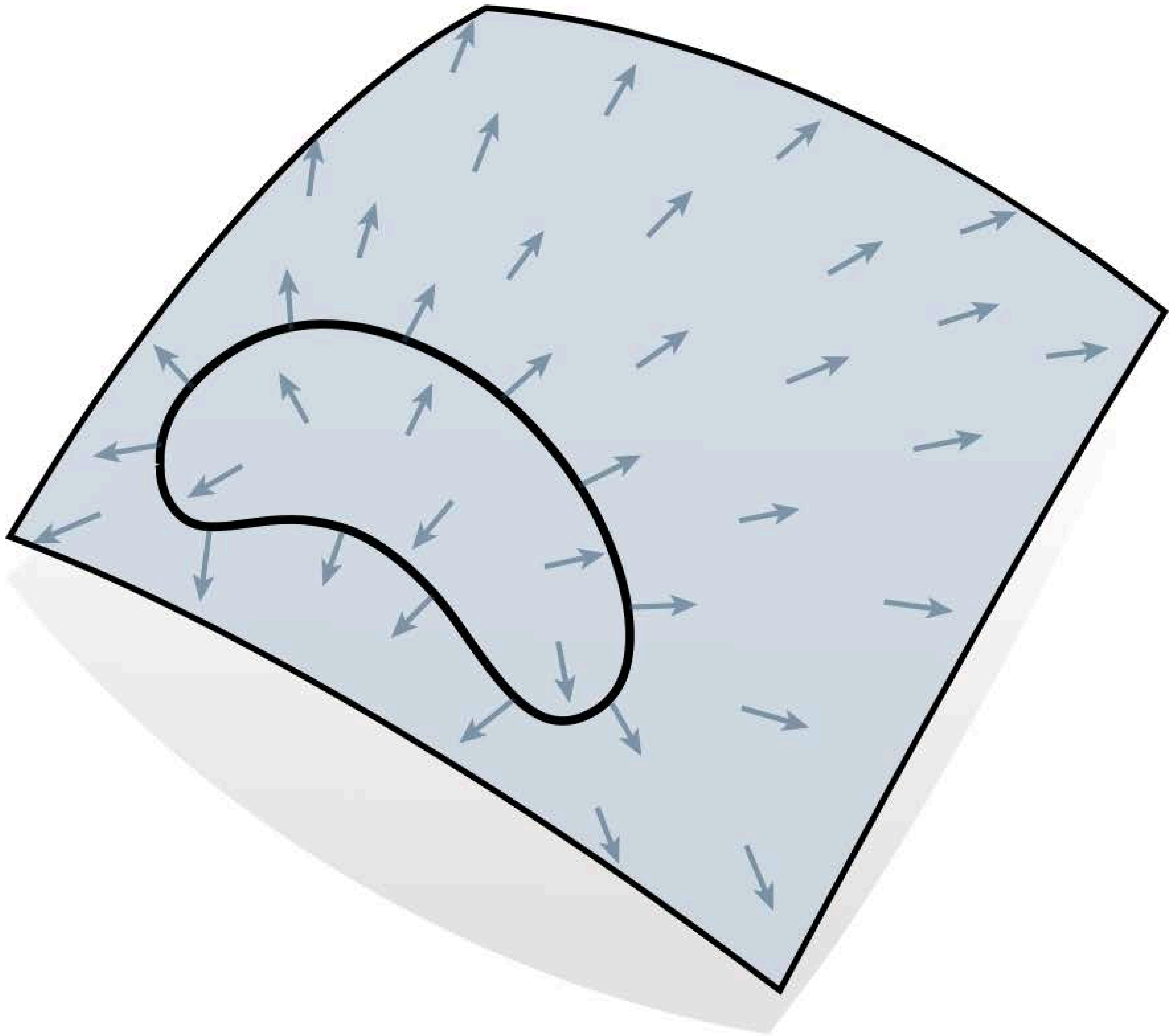
Key insight #2

We can normalize and integrate the diffused vectors to obtain signed distance.

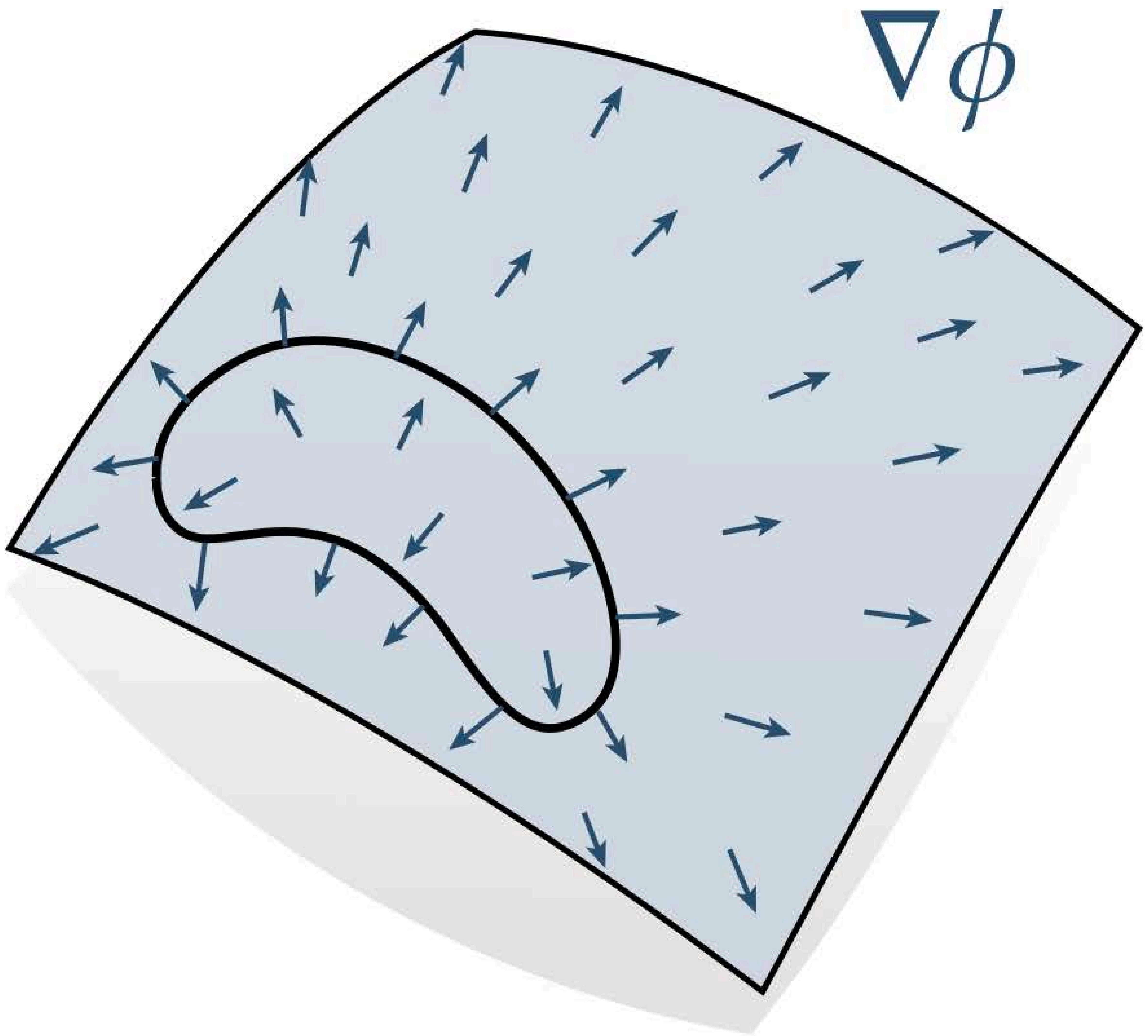
Key insight #2



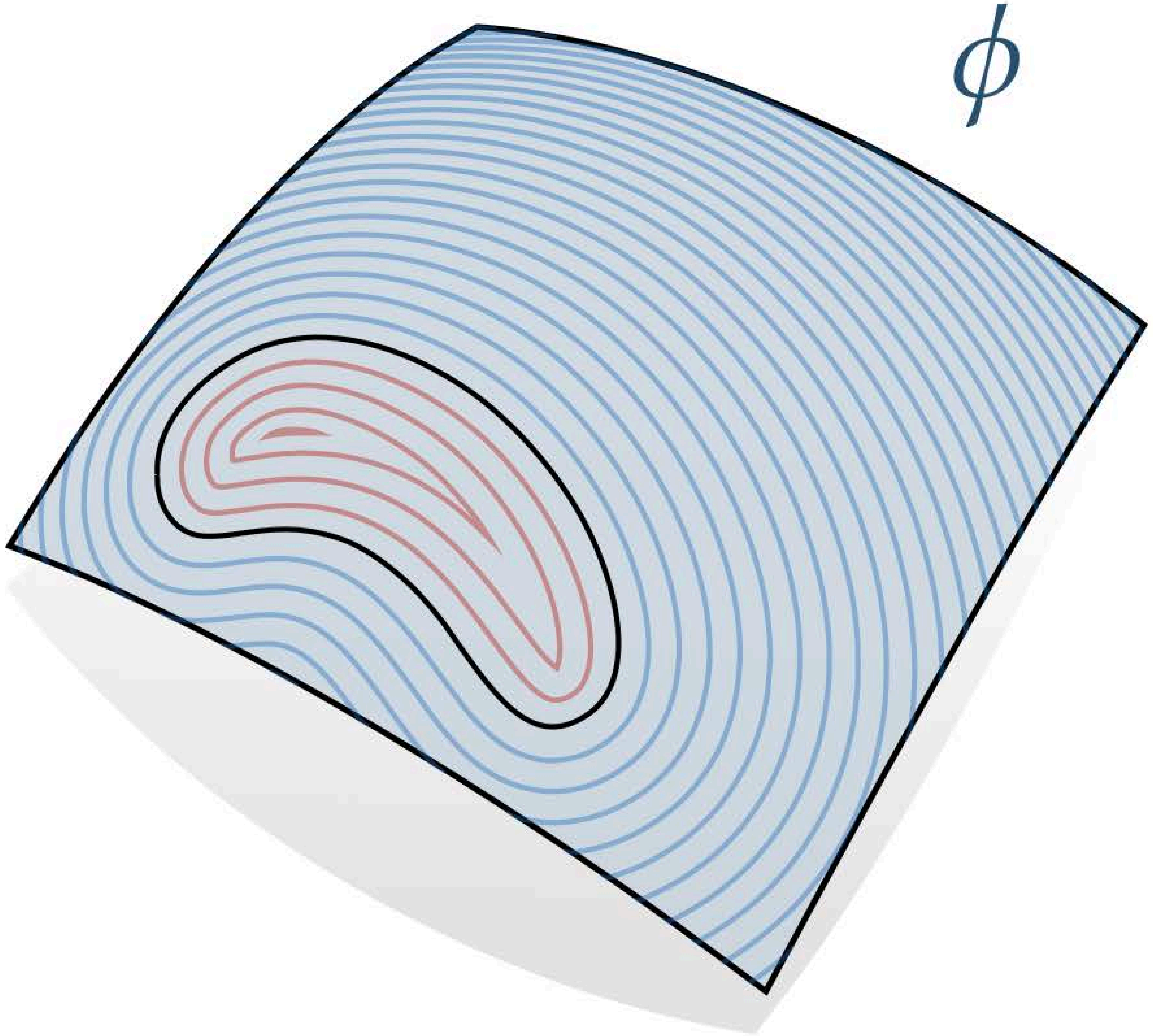
Key insight #2



Key insight #2



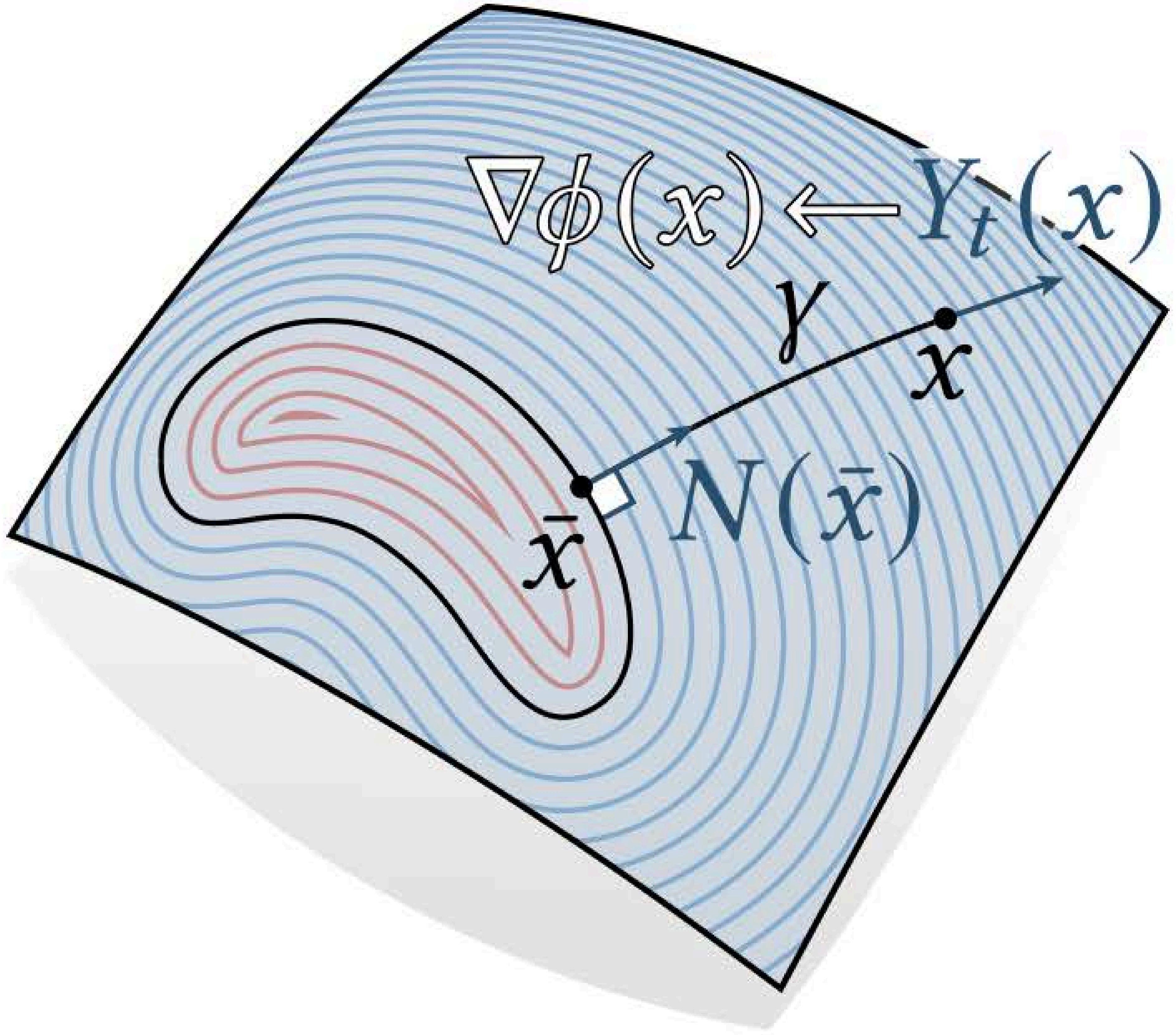
Key insight #2



Key insights

(1) Diffused normals will be parallel to the gradient of signed distance;

(2) Normalizing and integrating these vectors recovers an accurate SDF.



Step 1: Vector diffusion

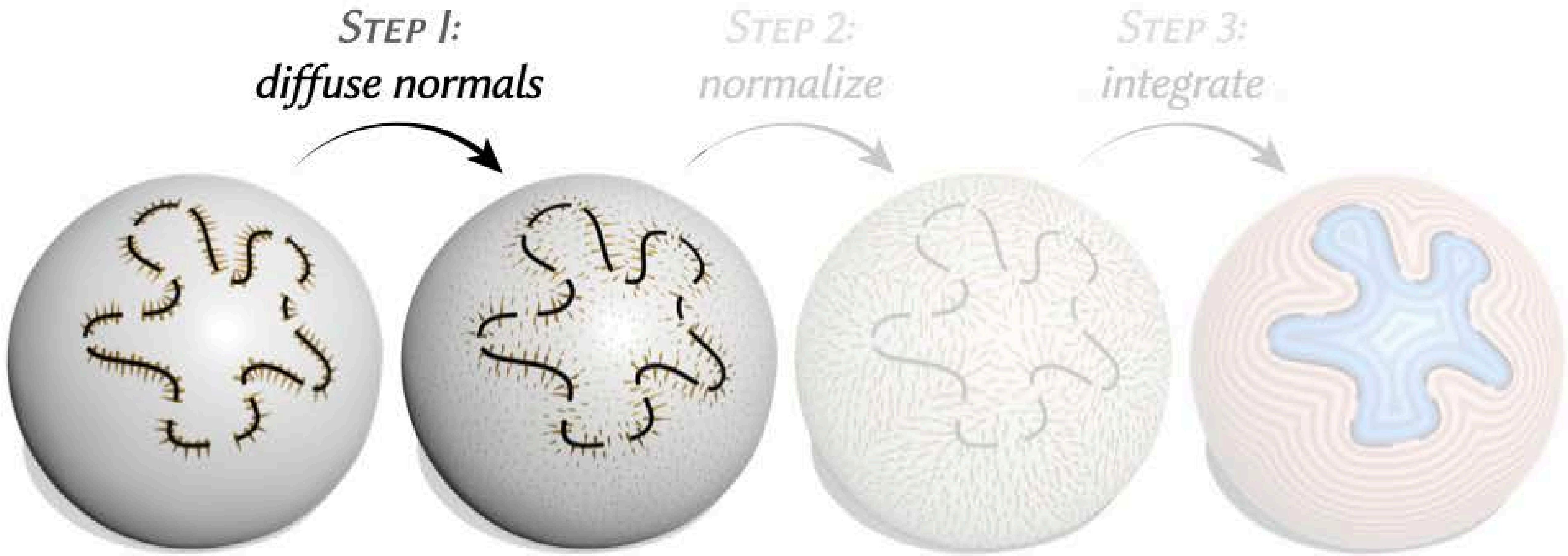
Step 1: Vector diffusion

(1) Integrate the vector heat equation for a short time t.

vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = \Delta^{\nabla} X_t, \quad t > 0,$$

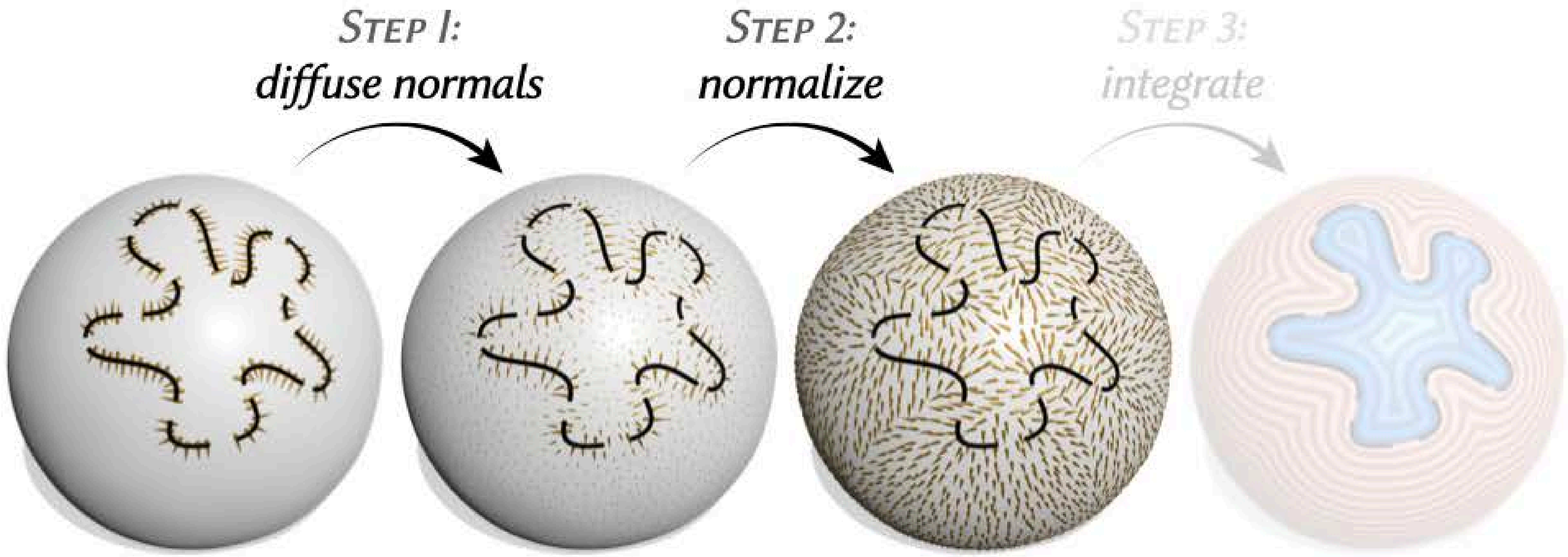
$$X_0 = N\mu_{\Omega}$$



Step 2: Normalize vectors

(2) Normalizing the resulting vector field X_t :

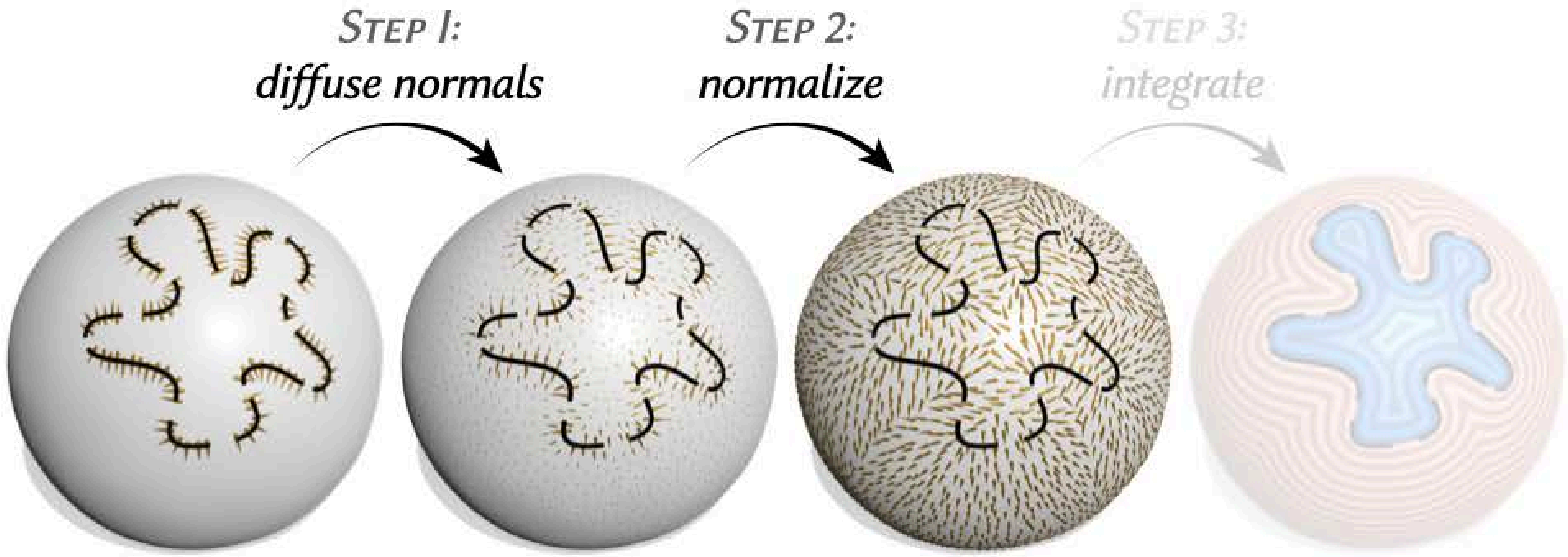
$$Y_t \leftarrow X_t / \|X_t\|$$



Step 2: Normalize vectors

(3) Look for the function ϕ whose gradient is as close as possible to Y_t :

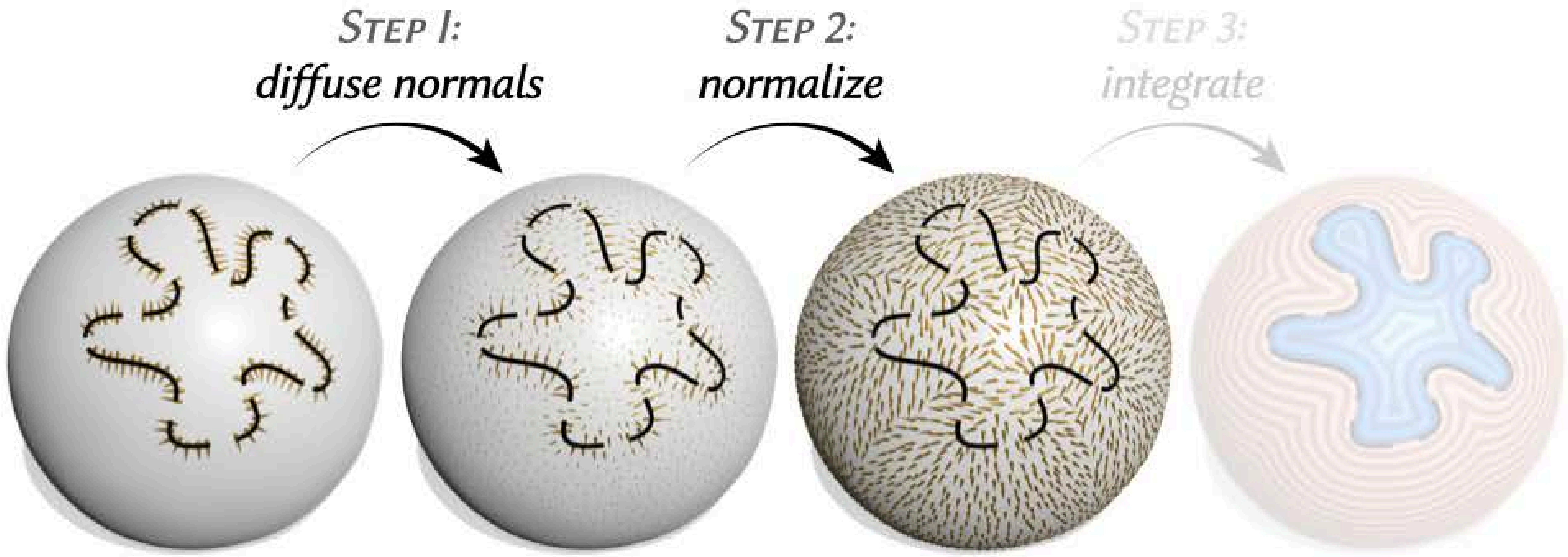
$$\min_{\phi:\, M\to\mathbb{R}} \int_M \|\nabla\phi - Y_t\|_2^2$$



Step 2: Normalize vectors

(3) Look for the function ϕ whose gradient is as close as possible to Y_t :

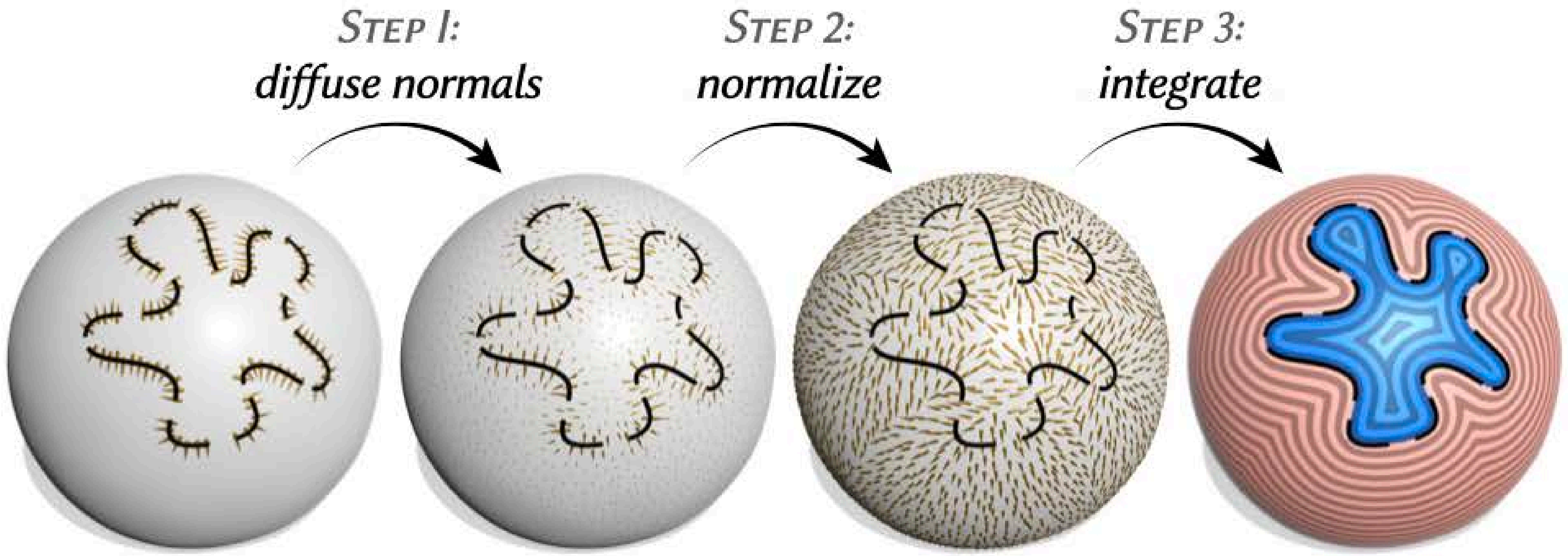
$$\min_{\phi:\, M \to \mathbb{R}} \int_M \|\nabla \phi - Y_t\|_2^2 \quad \longrightarrow \quad \begin{aligned} \Delta\phi &= \nabla \cdot Y_t && \text{on } M \\ \frac{\partial \phi}{\partial n} &= n \cdot Y_t && \text{on } \partial M \end{aligned}$$



Step 2: Normalize vectors

(3) Look for the function ϕ whose gradient is as close as possible to Y_t :

$$\min_{\phi:\, M\to\mathbb{R}} \int_M \|\nabla\phi - Y_t\|_2^2 \longrightarrow \begin{aligned} \Delta\phi &= \nabla\cdot Y_t \quad \text{on } M \\ \frac{\partial\phi}{\partial n} &= n\cdot Y_t \quad \text{on } \partial M \end{aligned}$$



DISCRETIZATION

Time discretization

Step 1:
vector diffusion

vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = \Delta^{\nabla} X_t$$

$$X_0 = N\mu_{\Omega}$$

Time discretization

STEP 1:
vector diffusion

vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = \Delta^\nabla X_t$$
$$X_0 = N\mu_\Omega$$

flow for $t > 0$ ⟶

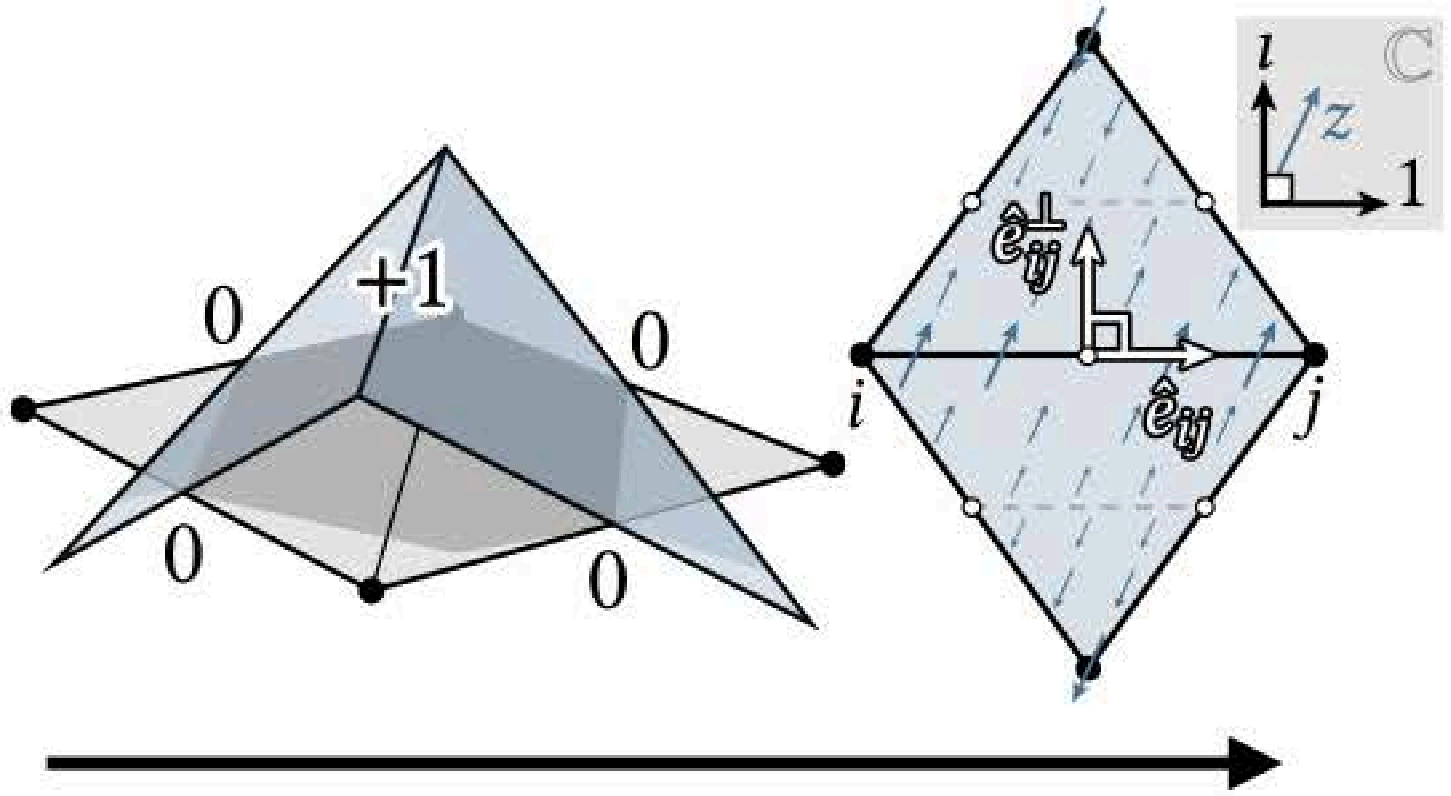
$$(\mathrm{id} - t\Delta^\nabla) X_t = X_0$$

The Vector Heat Method
N. Sharp, Y. Soliman, K. Crane (2020)

Spatial discretization on triangle meshes

STEP 1:
vector diffusion

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



Crouziex-Raviart basis functions

mass matrix · *connection Laplacian* · *initial normal vectors*

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

sparse linear system

Tangent Vector Fields on Triangulated Surfaces-An Edge-Based Approach
A. Djerbetian & M. Ben-Chen (2016)

A Simple Discretization of the Vector Dirichlet Energy
O. Stein, M. Wardetzky, A. Jacobson, E. Grinspun (2020)

Normalize vectors

Step 1:

vector diffusion

vector heat equation

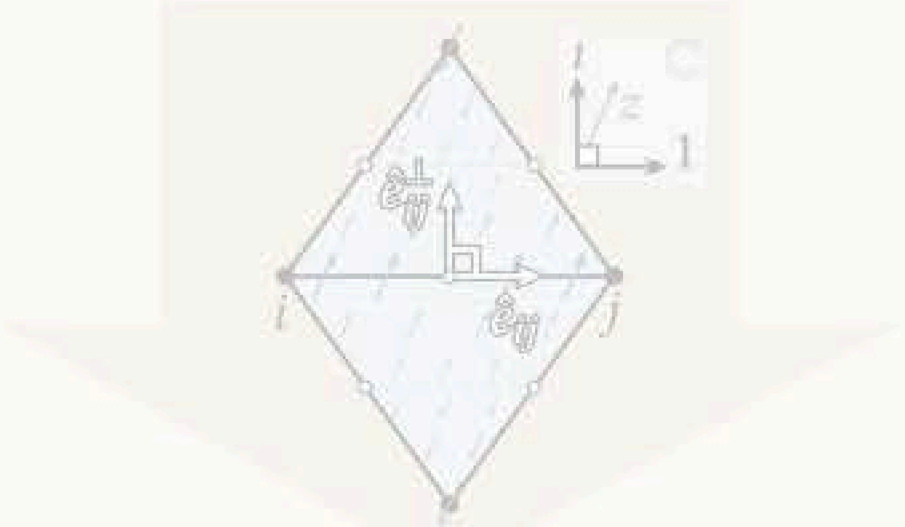
$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = \Delta^\nabla X_t$$

$$X_0 = N\mu_\Omega$$

↓

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

flow for $t > 0$



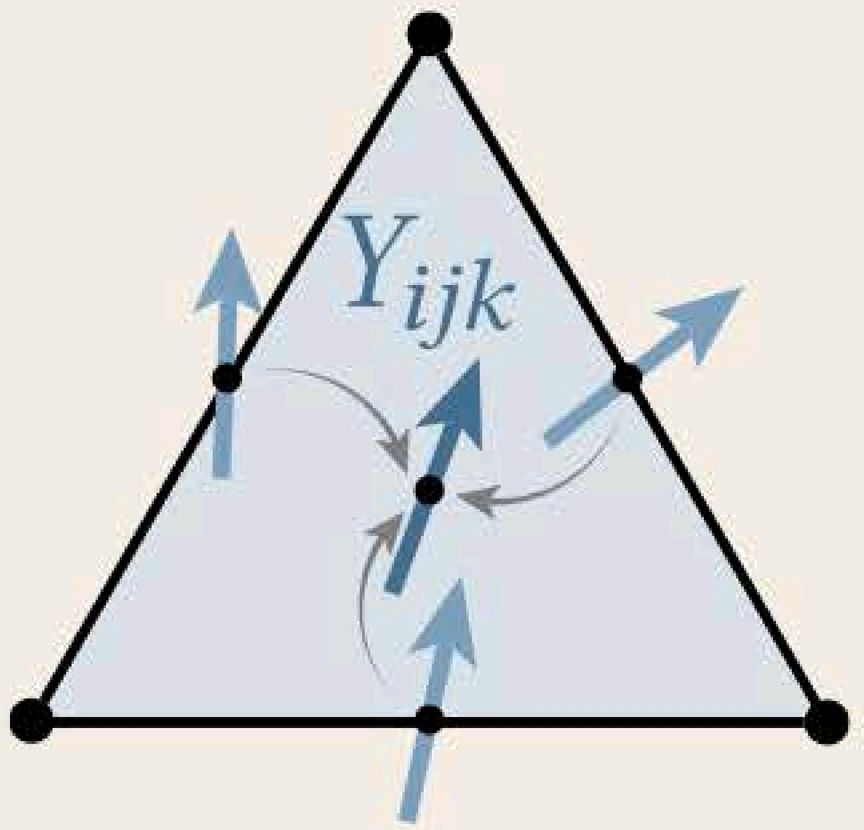
$$(\mathsf{M} + t\mathsf{L}^\nabla)\mathsf{X} = \mathsf{X}_0$$

sparse linear system

Step 2:

normalization

Average edge-based vectors onto faces, and normalize.



Integrate vector field

Step 1: vector diffusion

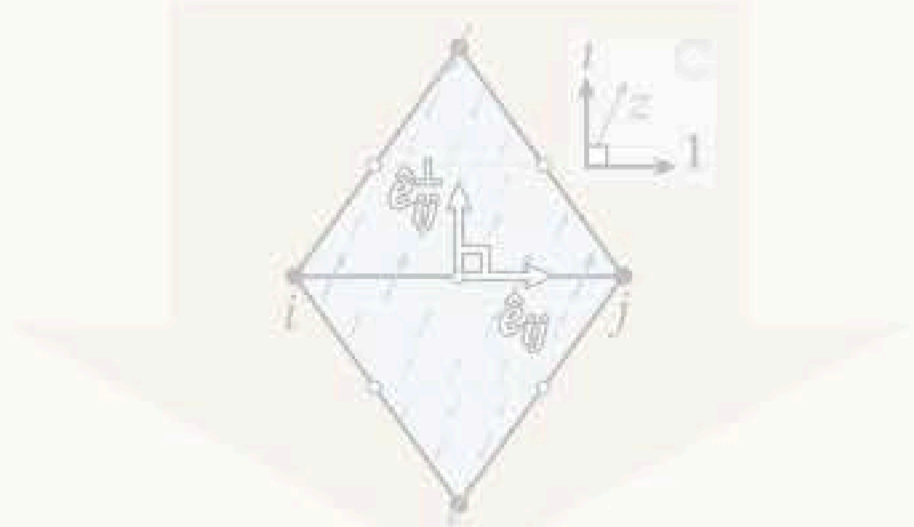
vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = \Delta^\nabla X_t$$

$$X_0 = N\mu_\Omega$$

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

flow for $t > 0$

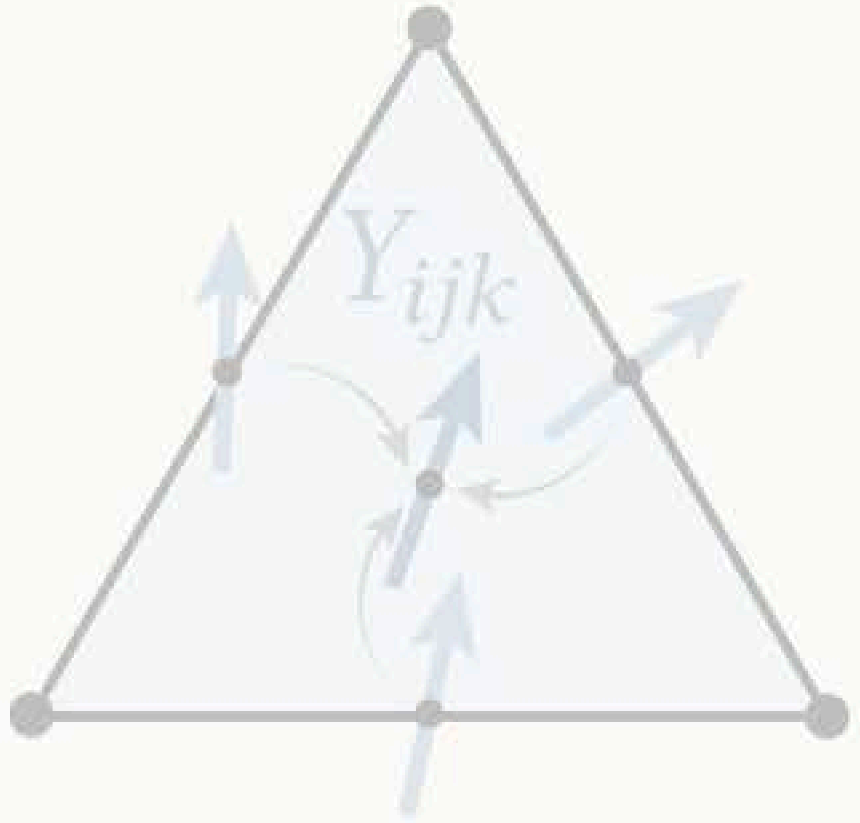


$$(\mathsf{M} + t\mathsf{L}^\nabla)\mathsf{X} = \mathsf{X}_0$$

sparse linear system

Step 2: normalization

Average edge-based vectors onto faces, and normalize.



Step 3: integration

Poisson equation

$$\Delta\phi = \nabla \cdot Y_t \quad \text{on } M$$

$$\frac{\partial\phi}{\partial n} = n \cdot Y_t \quad \text{on } \partial M$$

Integrate vector field

STEP 1: vector diffusion

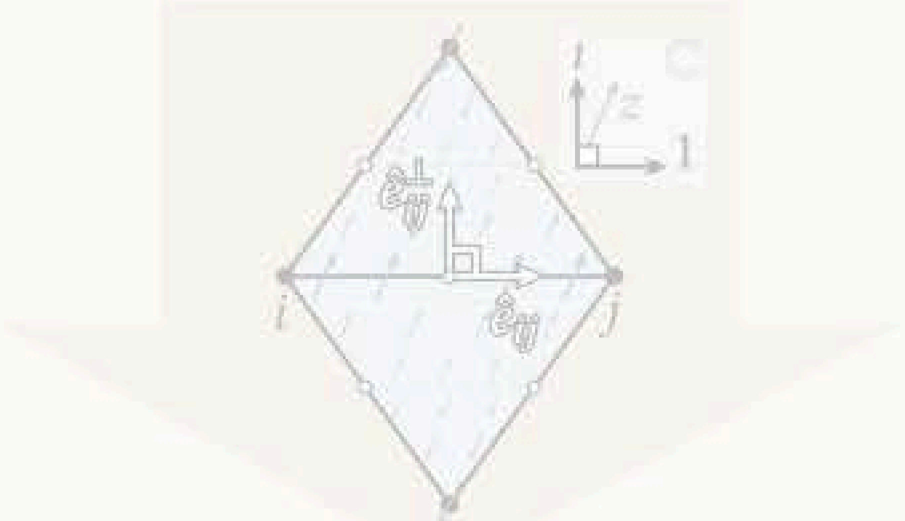
vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = \Delta^\nabla X_t$$
$$X_0 = N\mu_\Omega$$

↓

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

flow for $t > 0$

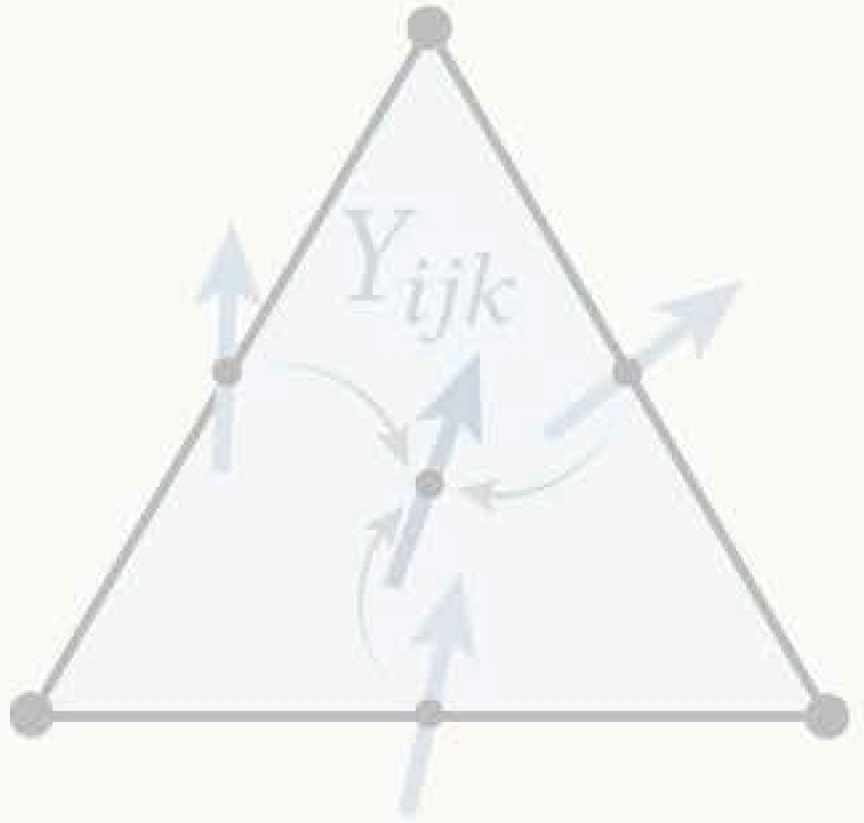


$$(\mathsf{M} + t\mathsf{L}^\nabla)\mathsf{X} = \mathsf{X}_0$$

sparse linear system

STEP 2: normalization

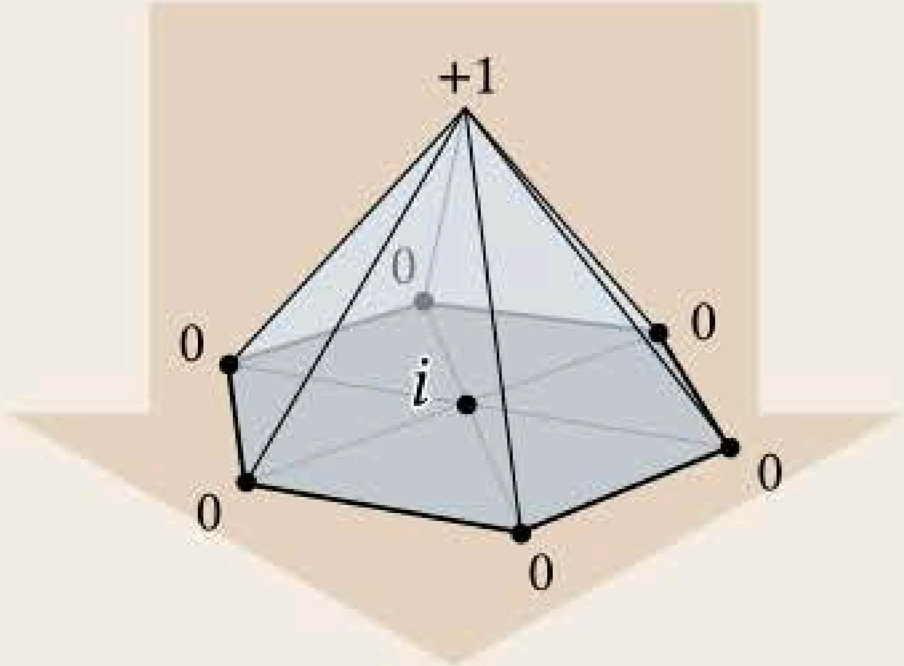
Average edge-based vectors onto faces, and normalize.



STEP 3: integration

Poisson equation

$$\Delta\phi = \nabla\cdot Y_t \quad \text{on } M$$
$$\frac{\partial\phi}{\partial n} = n\cdot Y_t \quad \text{on } \partial M$$



cotan Laplacian $\mathsf{C}\phi = \mathsf{b}$ discrete divergence

sparse linear system

Algorithm summary

STEP 1: vector diffusion

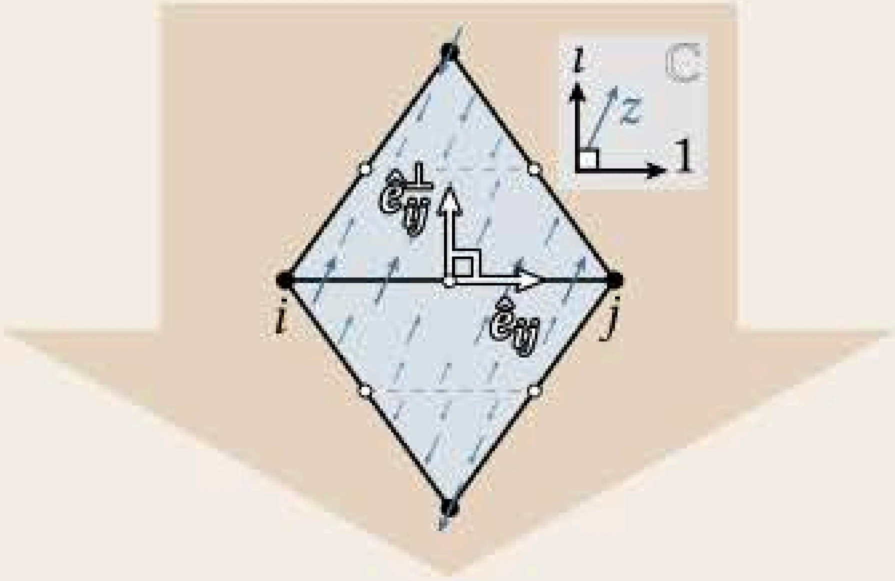
vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = \Delta^{\nabla} X_t$$
$$X_0 = N\mu_\Omega$$

↓

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

flow for $t > 0$

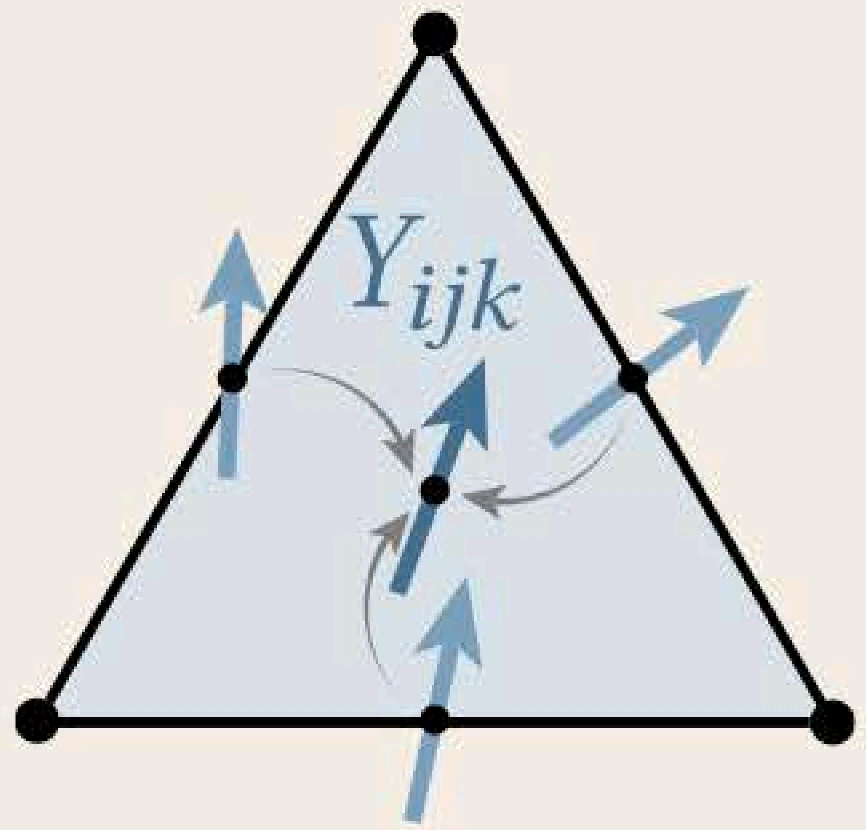


$$(\mathsf{M} + t\mathsf{L}^{\nabla})\mathsf{X} = \mathsf{X}_0$$

sparse linear system

STEP 2: normalization

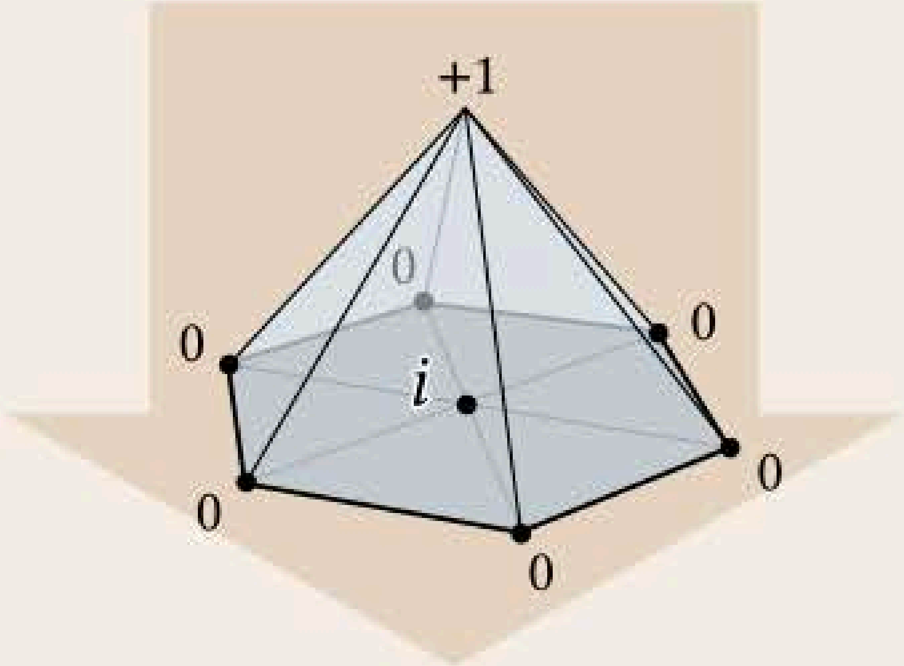
Average edge-based vectors onto faces, and normalize.



STEP 3: integration

Poisson equation

$$\Delta\phi = \nabla \cdot Y_t \quad \text{on } M$$
$$\frac{\partial \phi}{\partial n} = n \cdot Y_t \quad \text{on } \partial M$$



cotan Laplacian $\quad \mathsf{C}\phi = \mathsf{b} \quad$ discrete divergence

sparse linear system

Algorithm summary

Step 1: vector diffusion

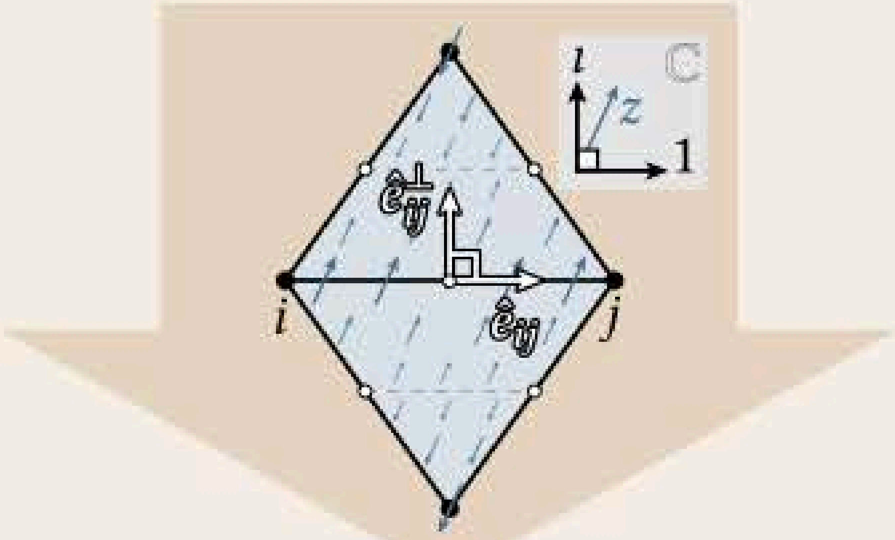
vector heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = \Delta^\nabla X_t$$
$$X_0 = N\mu_\Omega$$

↓

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

flow for $t > 0$

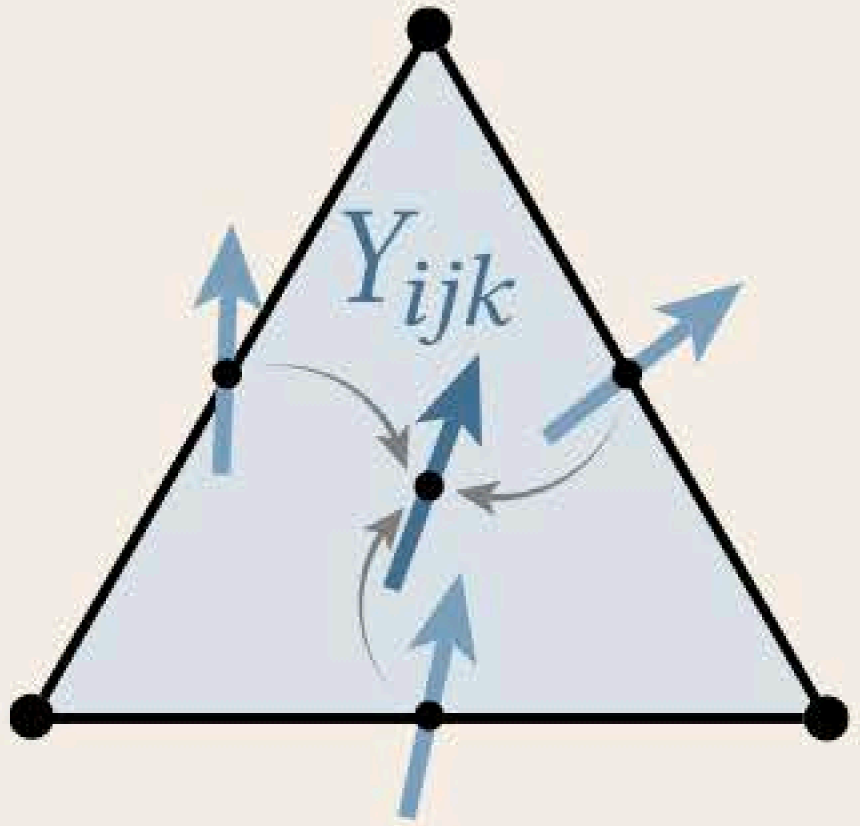


$$(\mathsf{M} + t\mathsf{L}^\nabla)\mathsf{X} = \mathsf{X}_0$$

sparse linear system

Step 2: normalization

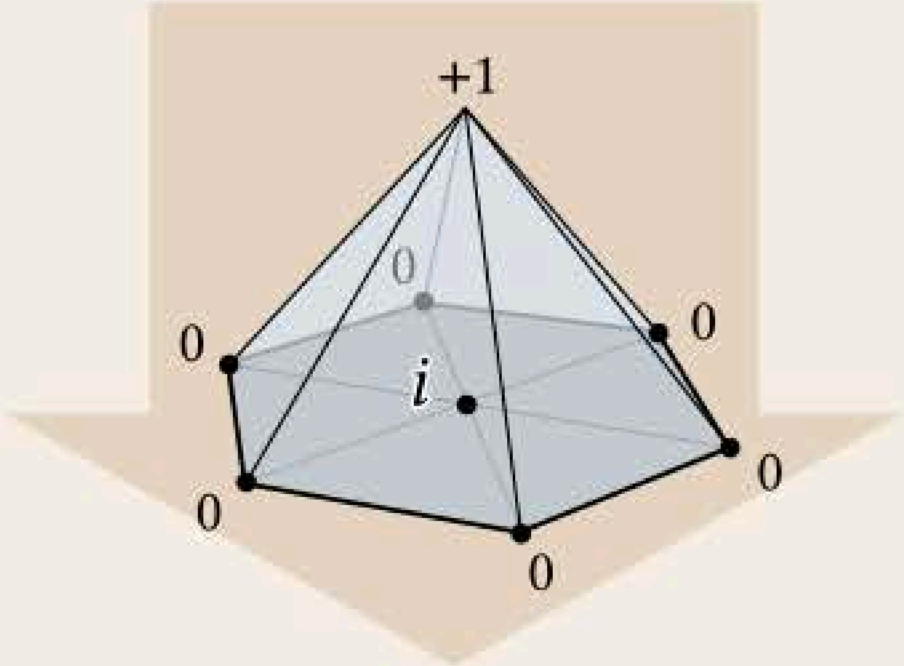
Average edge-based vectors onto faces, and normalize.



Step 3: integration

Poisson equation

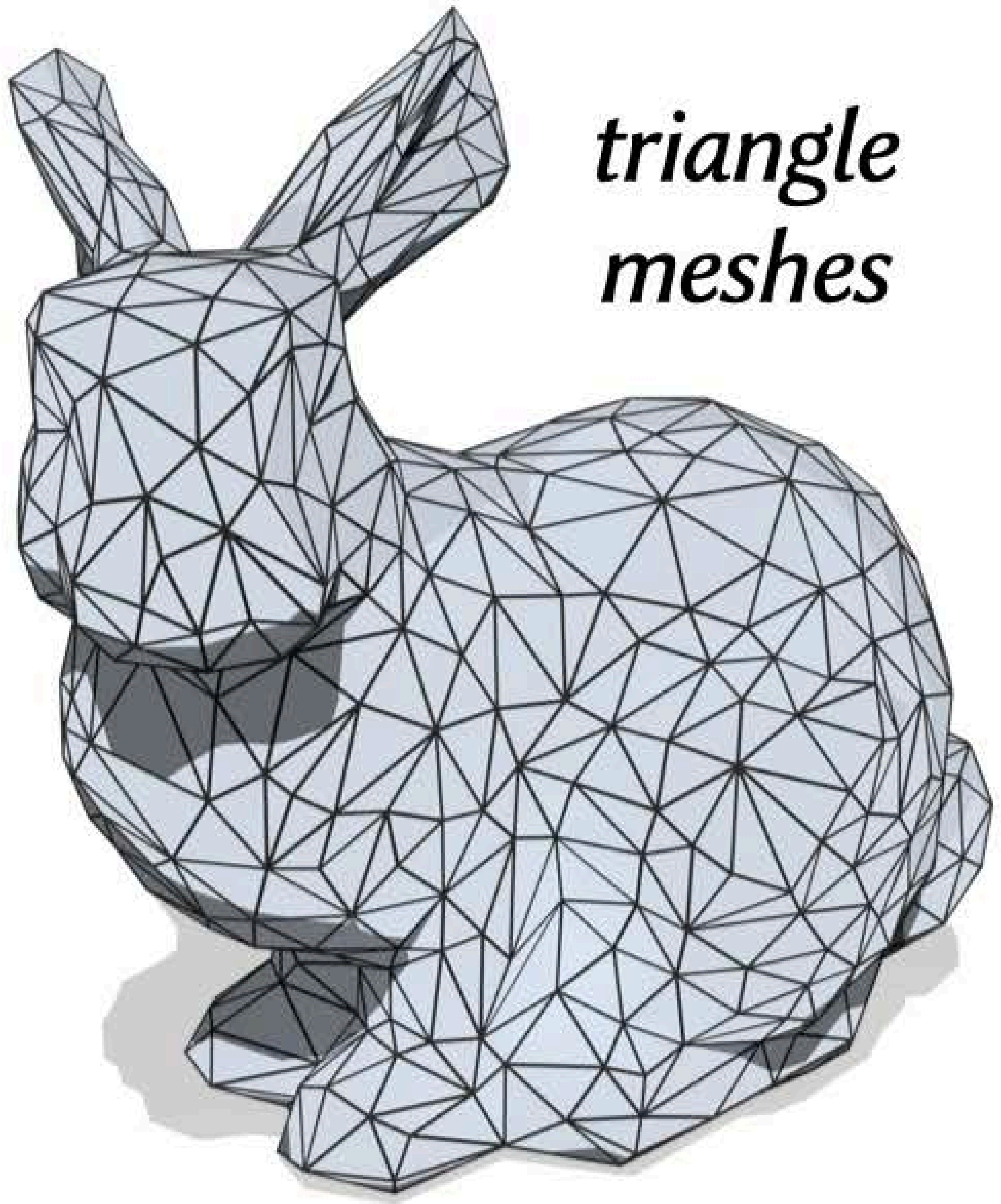
$$\Delta\phi = \nabla \cdot Y_t \quad \text{on } M$$
$$\frac{\partial\phi}{\partial n} = n \cdot Y_t \quad \text{on } \partial M$$



cotan Laplacian $\mathsf{C}\phi = \mathsf{b}$ discrete divergence

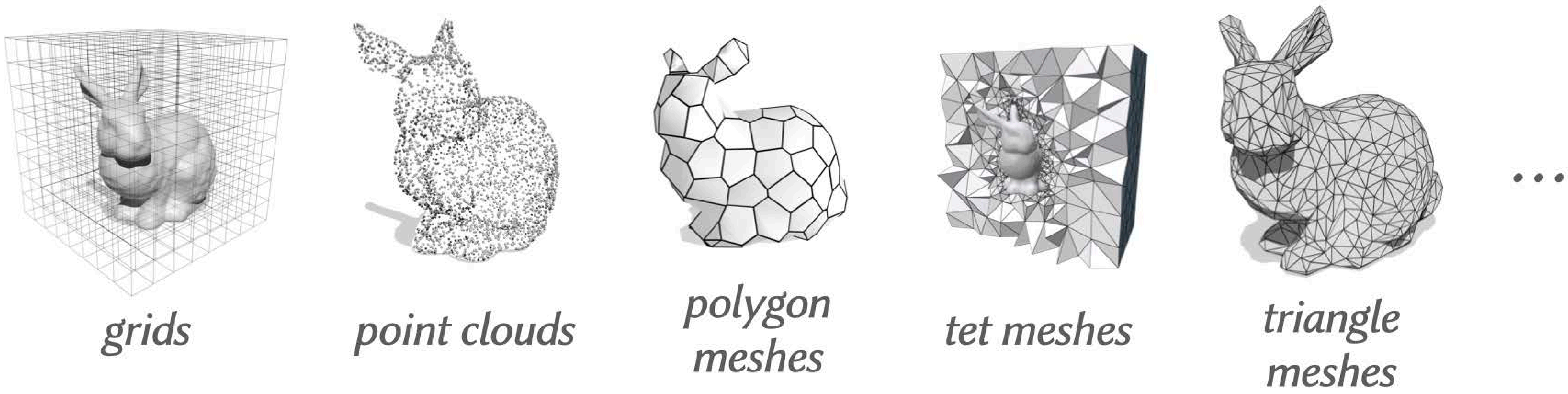
sparse linear system

Beyond triangle meshes...

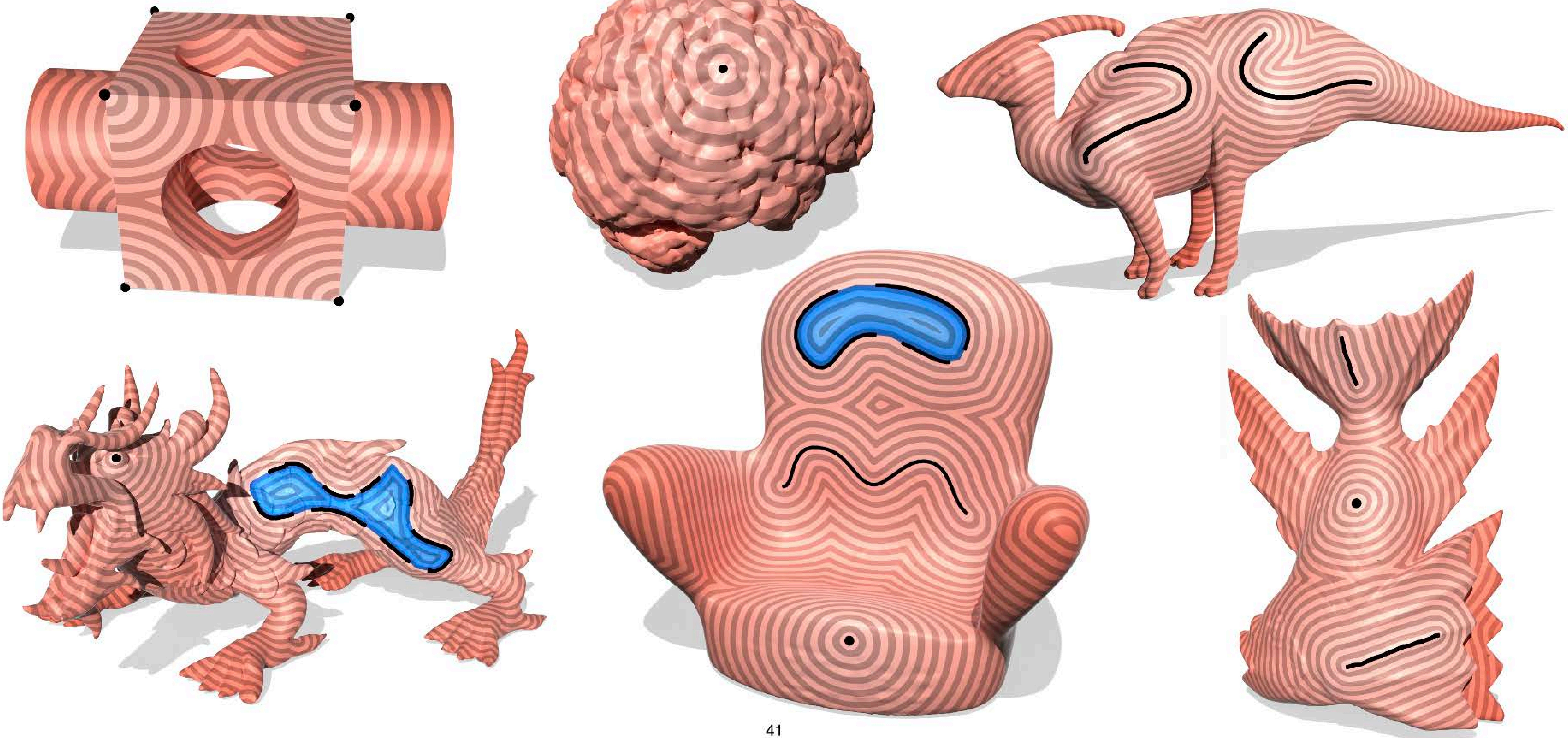


triangle meshes

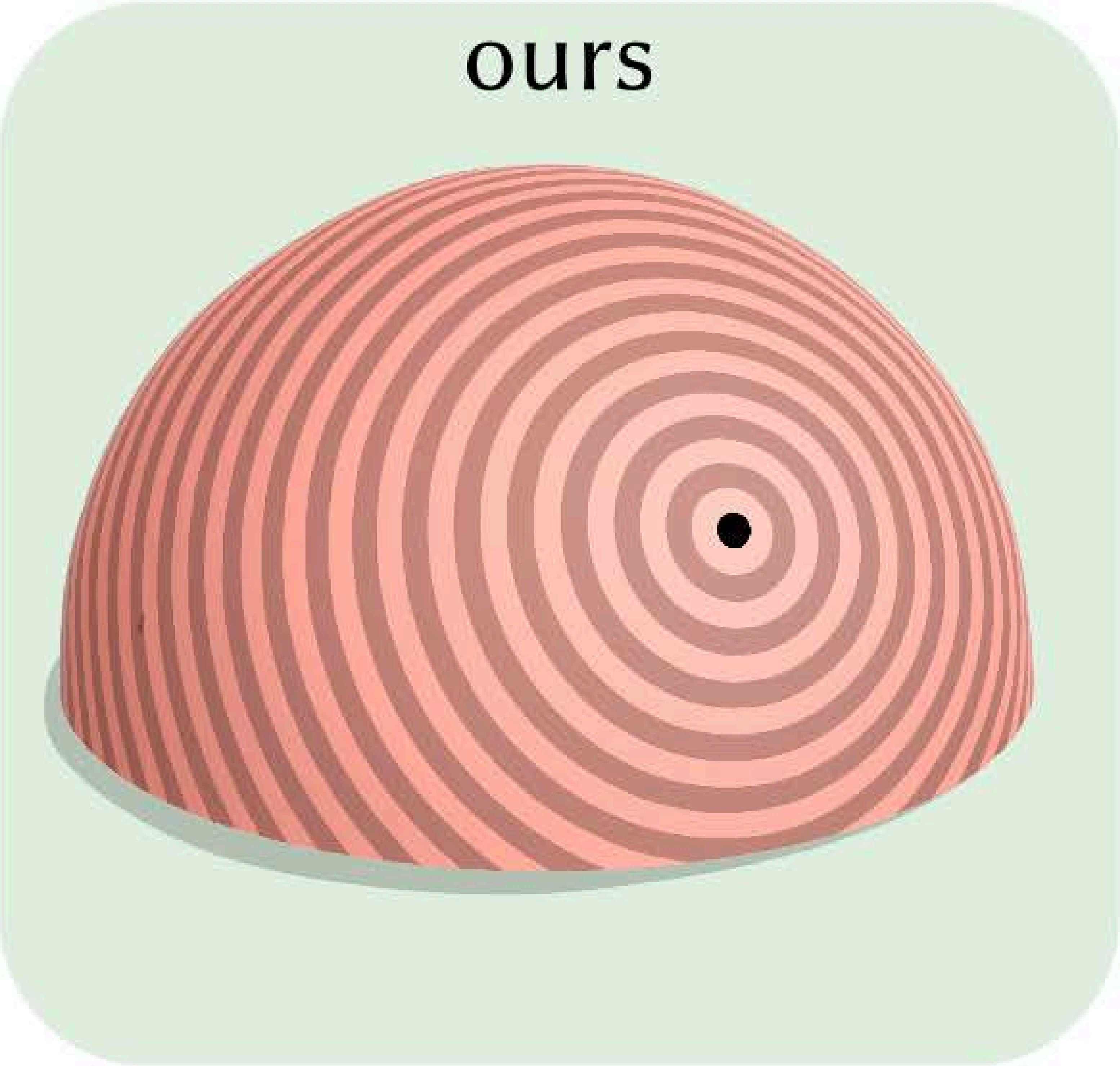
Our method applies to any data structure



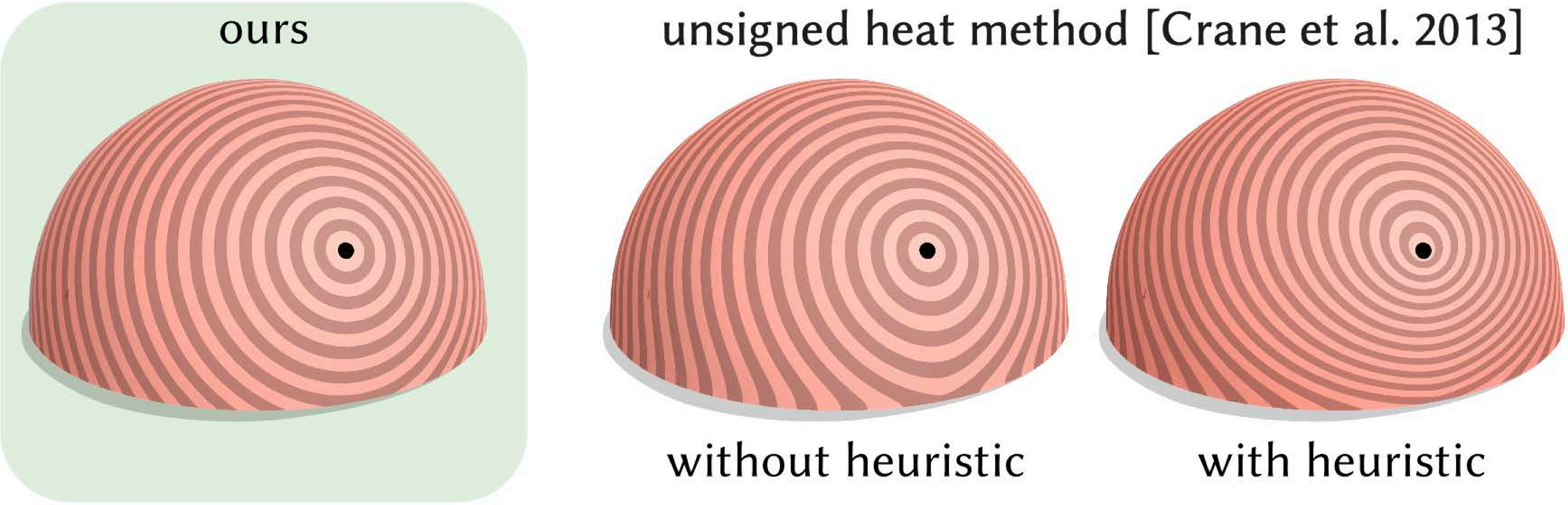
Can mix signed & unsigned distance



Domains with boundary



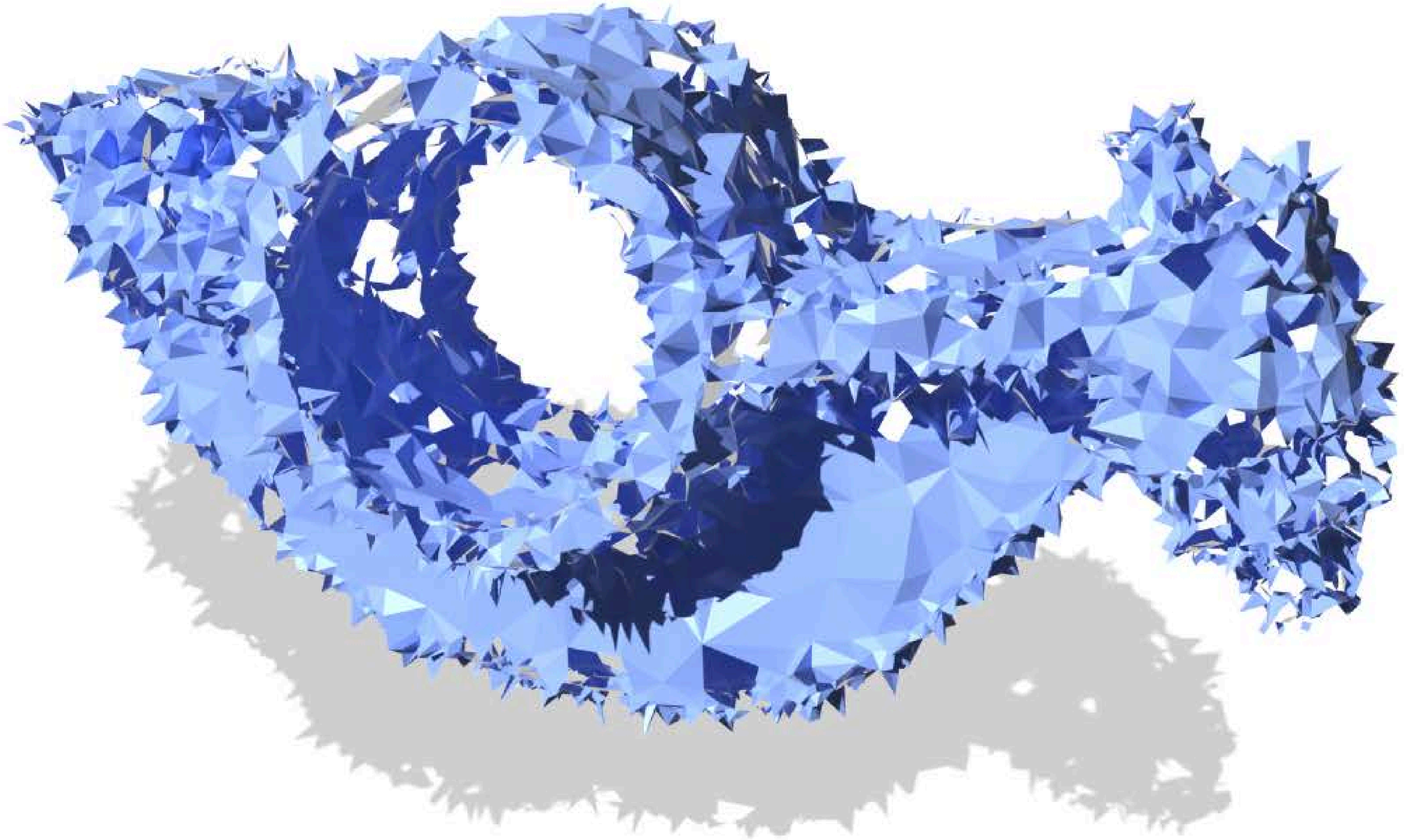
Domains with boundary



RESULTS

Offset surfaces in 3D

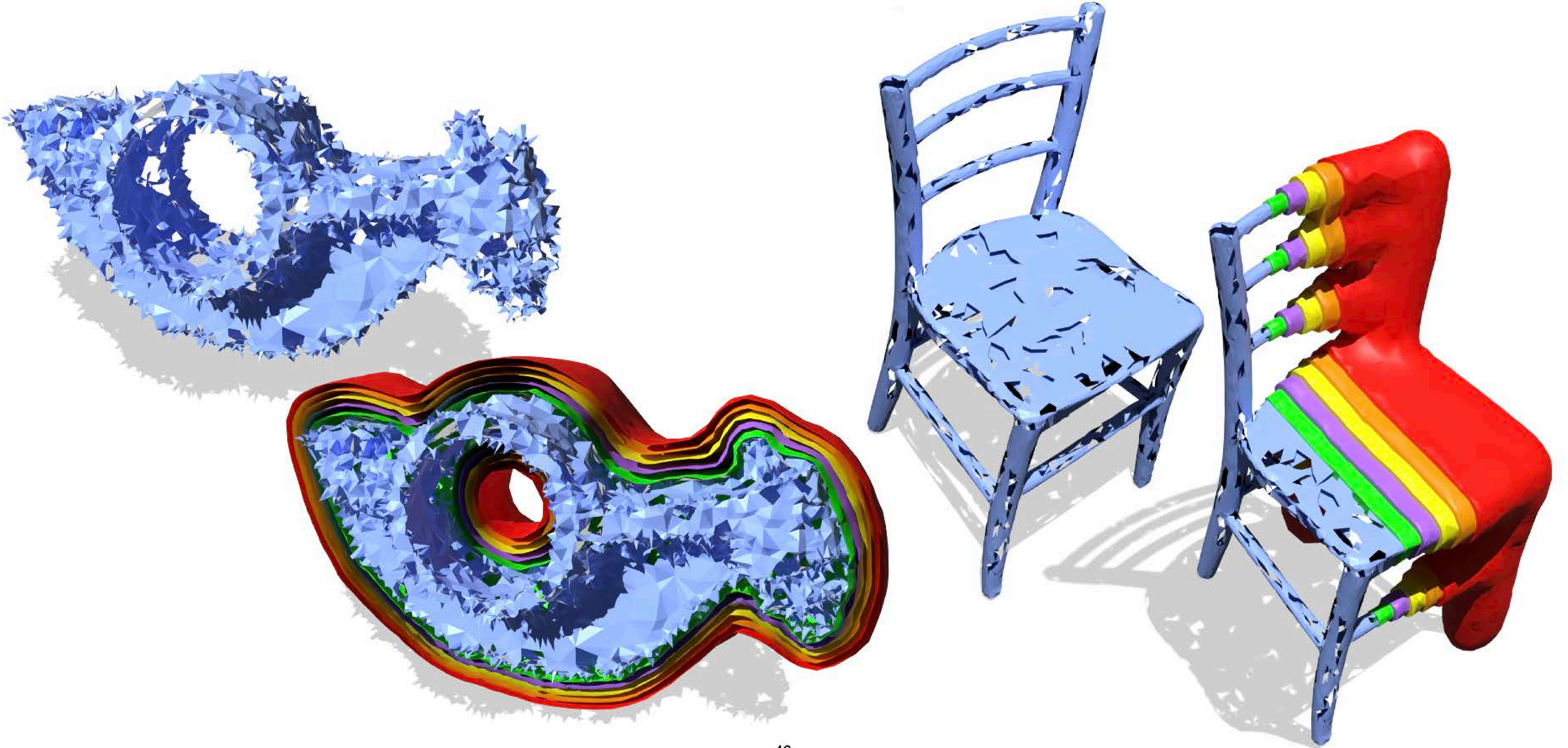
input



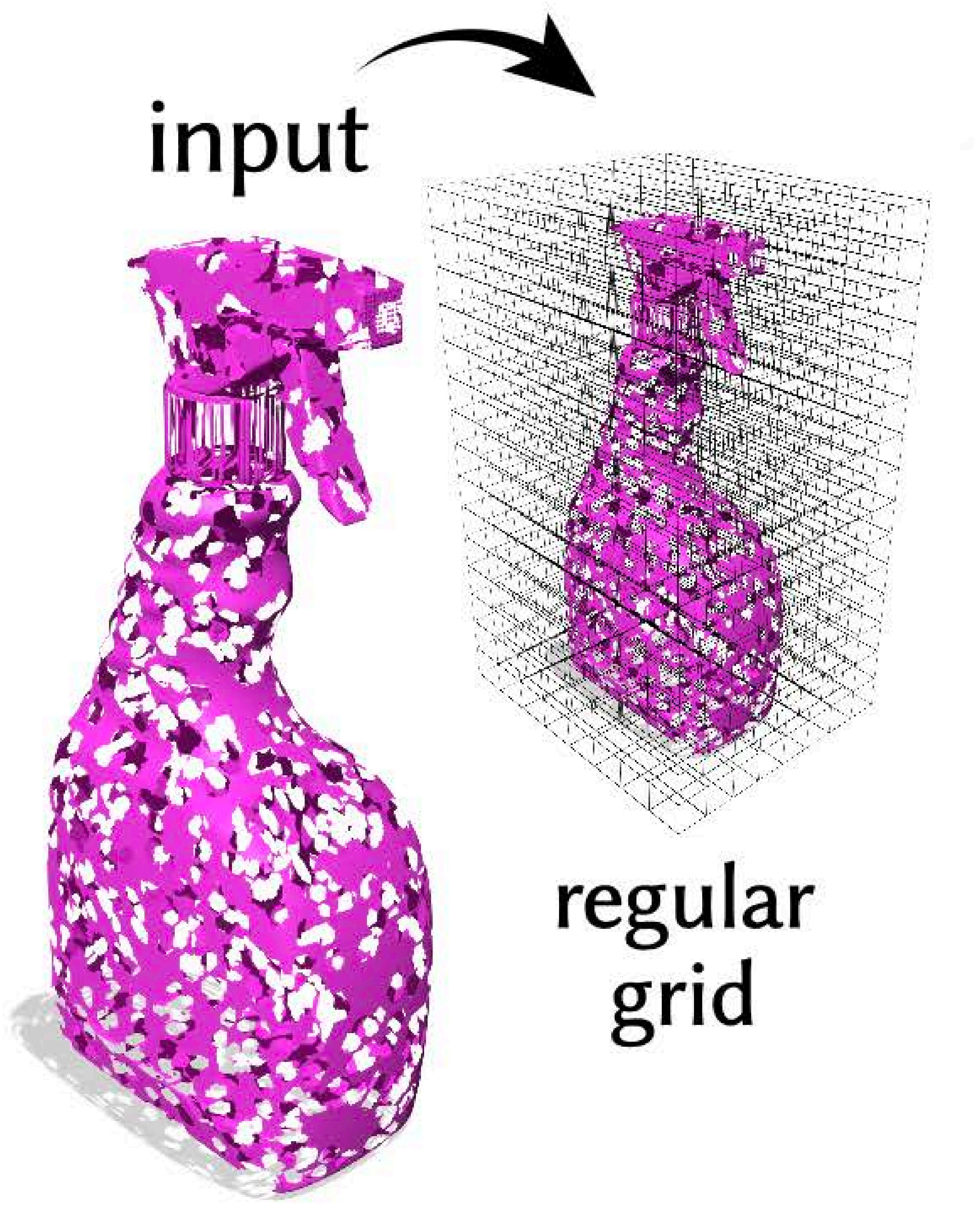
Offset surfaces in 3D

Offset surfaces in 3D

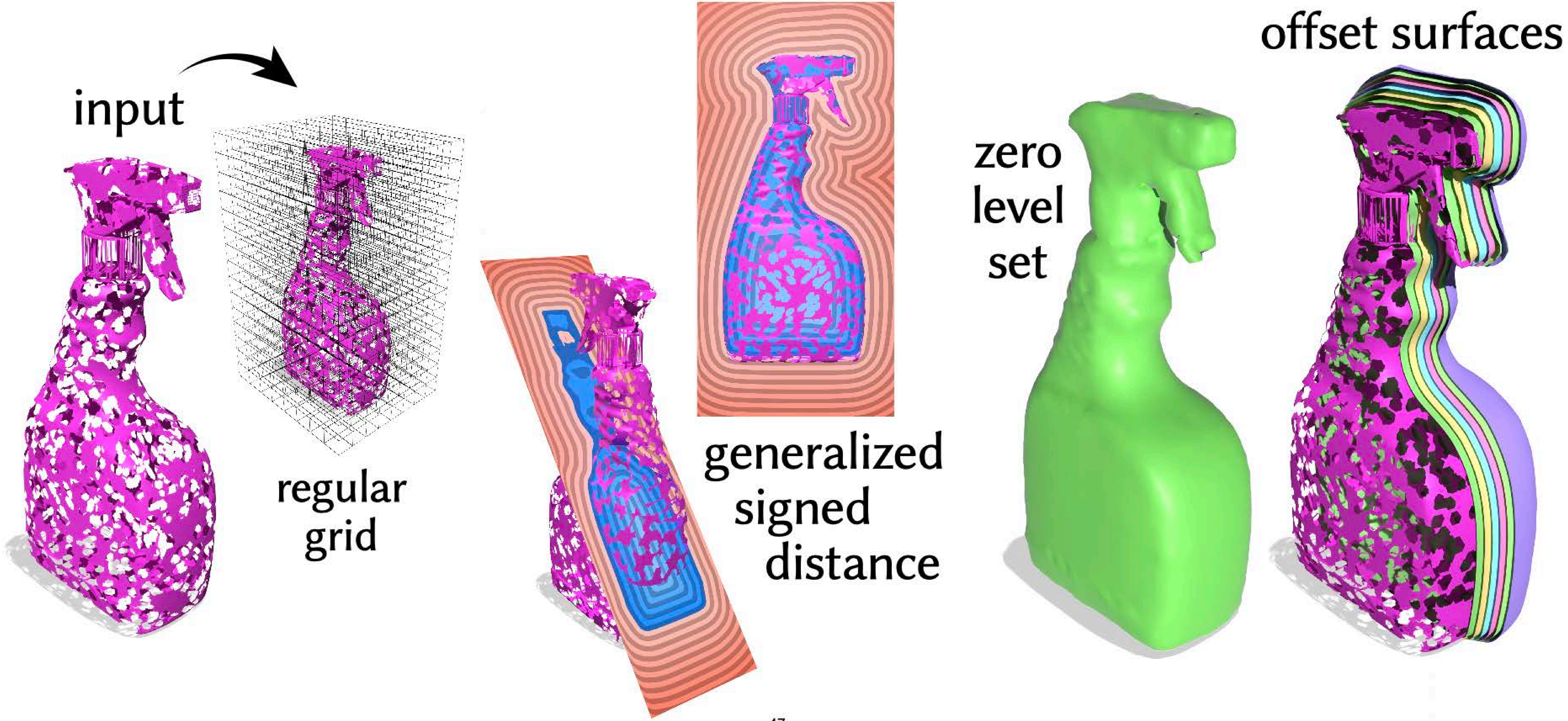
Offset surfaces in 3D



Volumetric grid domains

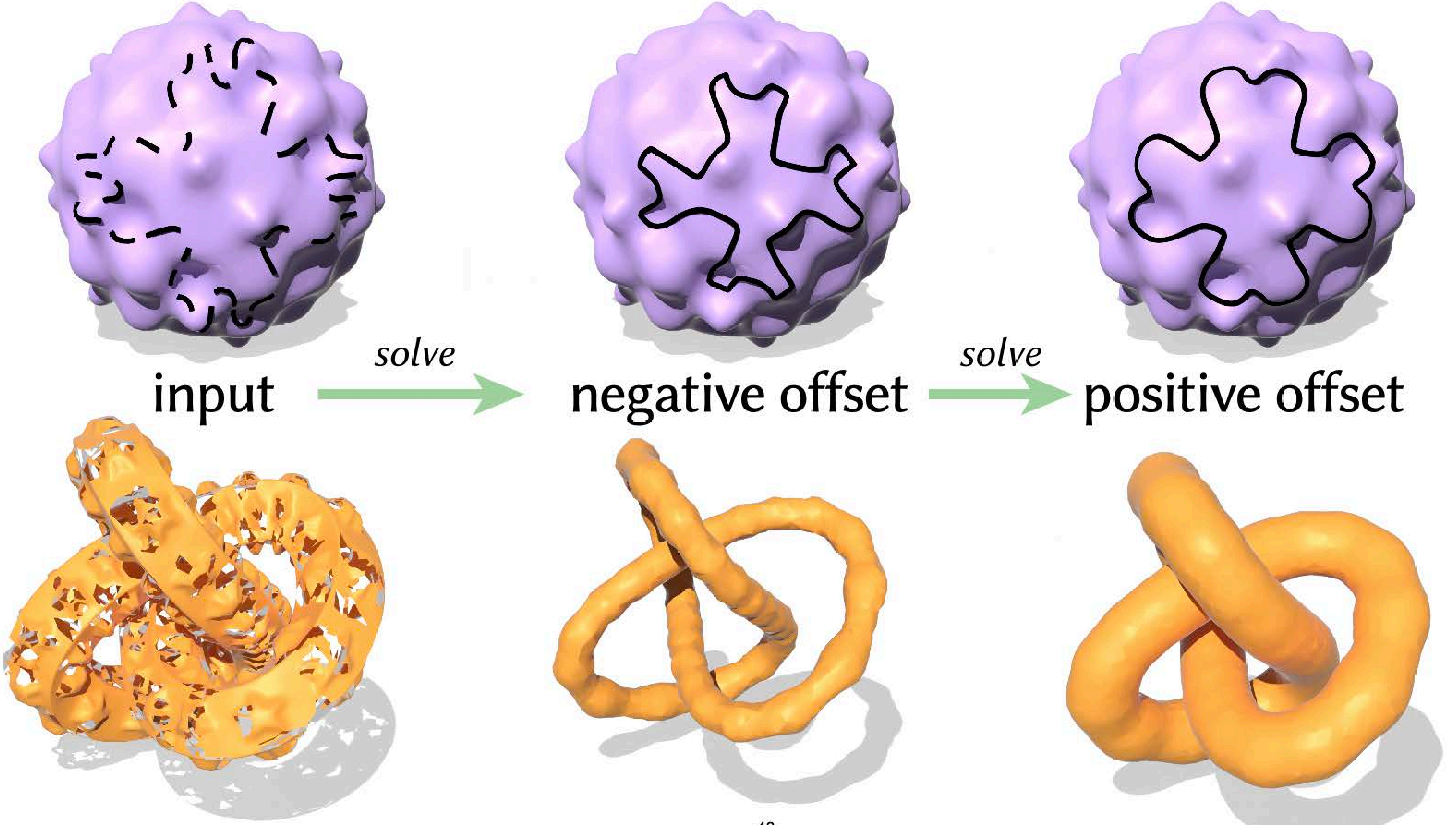


Volumetric grid domains



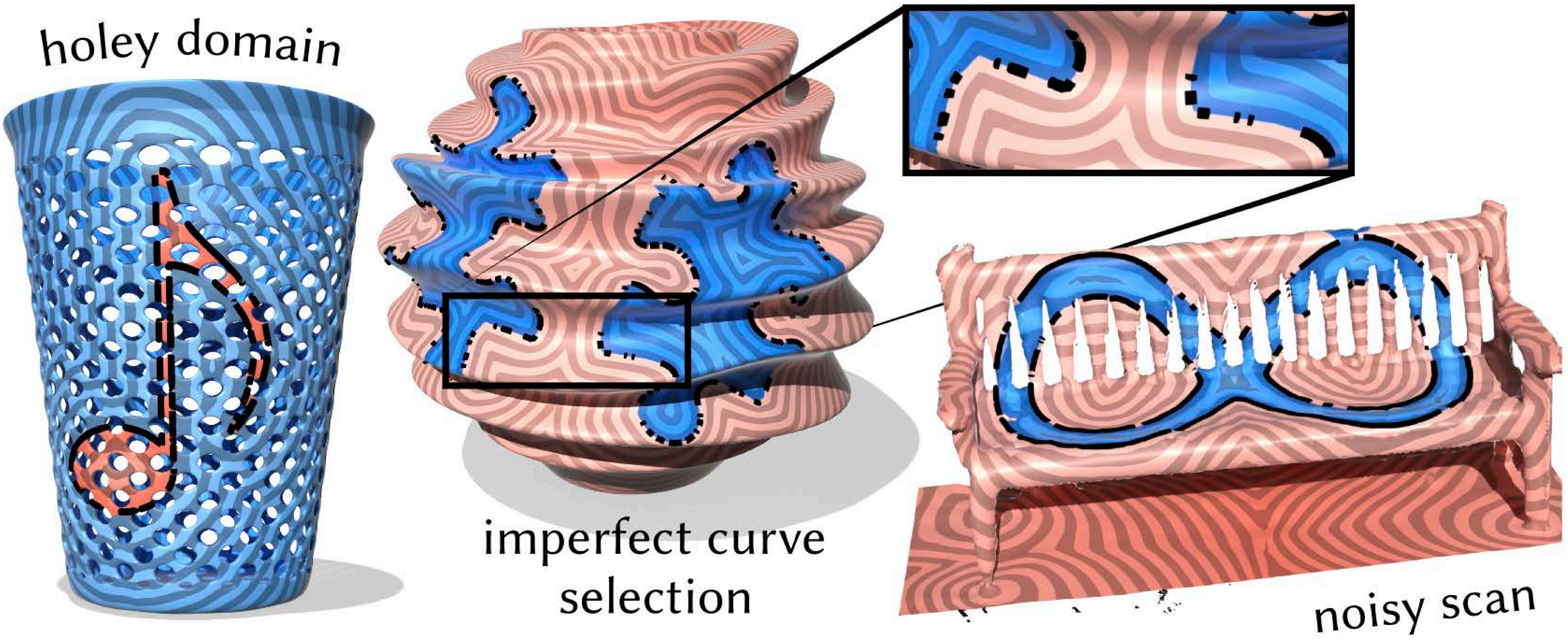
Generalized morphological operations

Generalized morphological operations

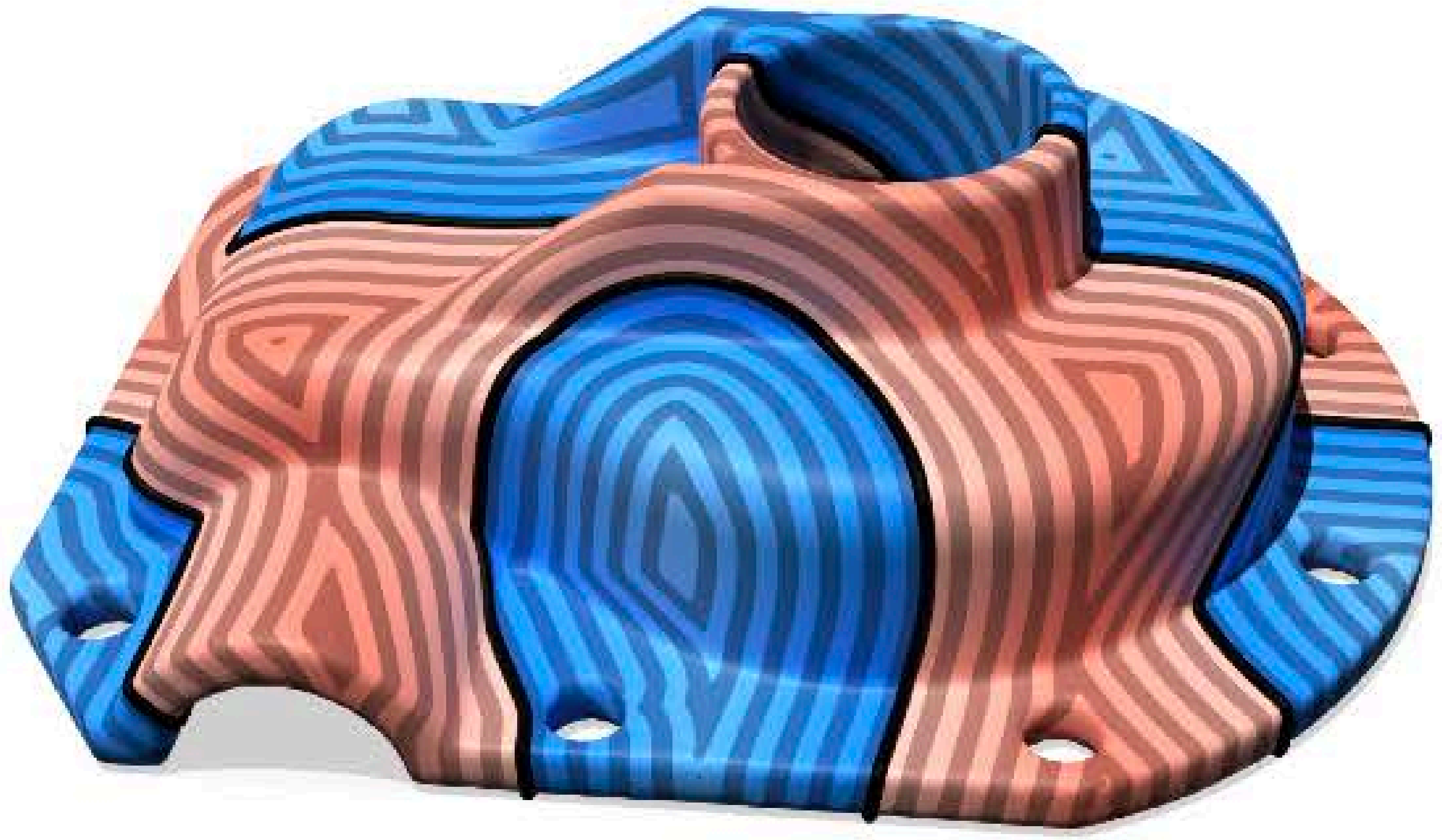


Robustness to defects in the source geometry

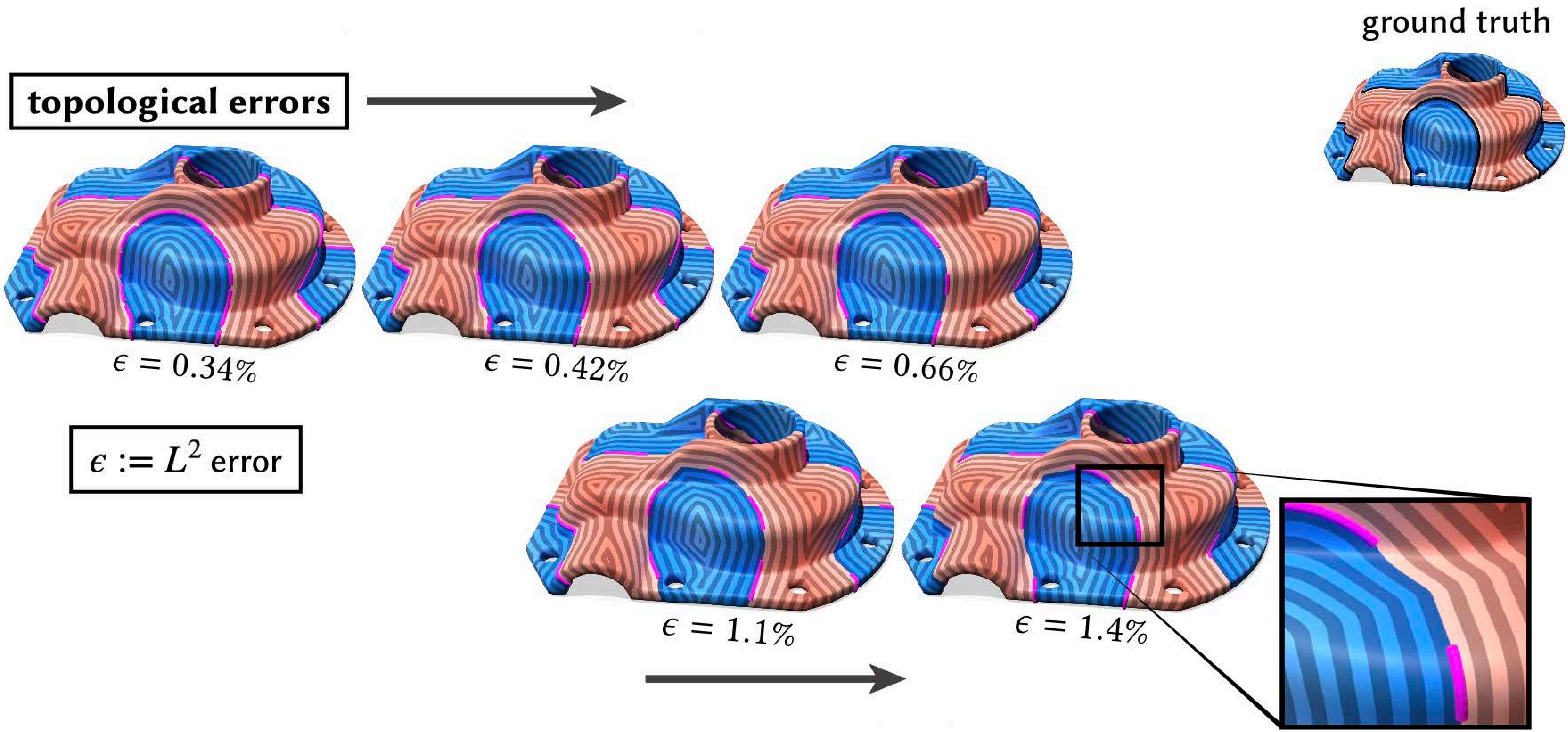
Robustness to defects in the source geometry



Robustness to errors in the source geometry



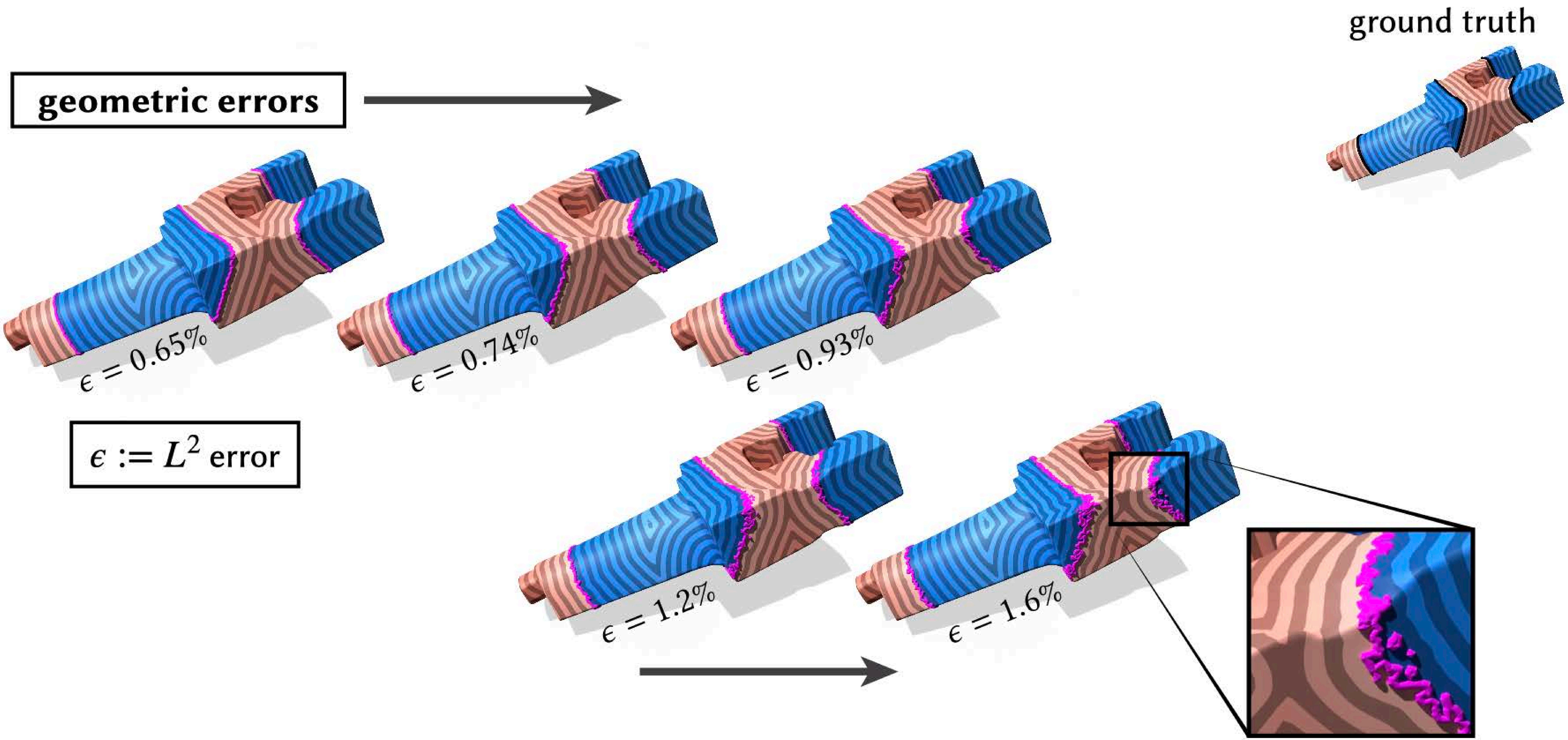
Robustness to errors in the source geometry



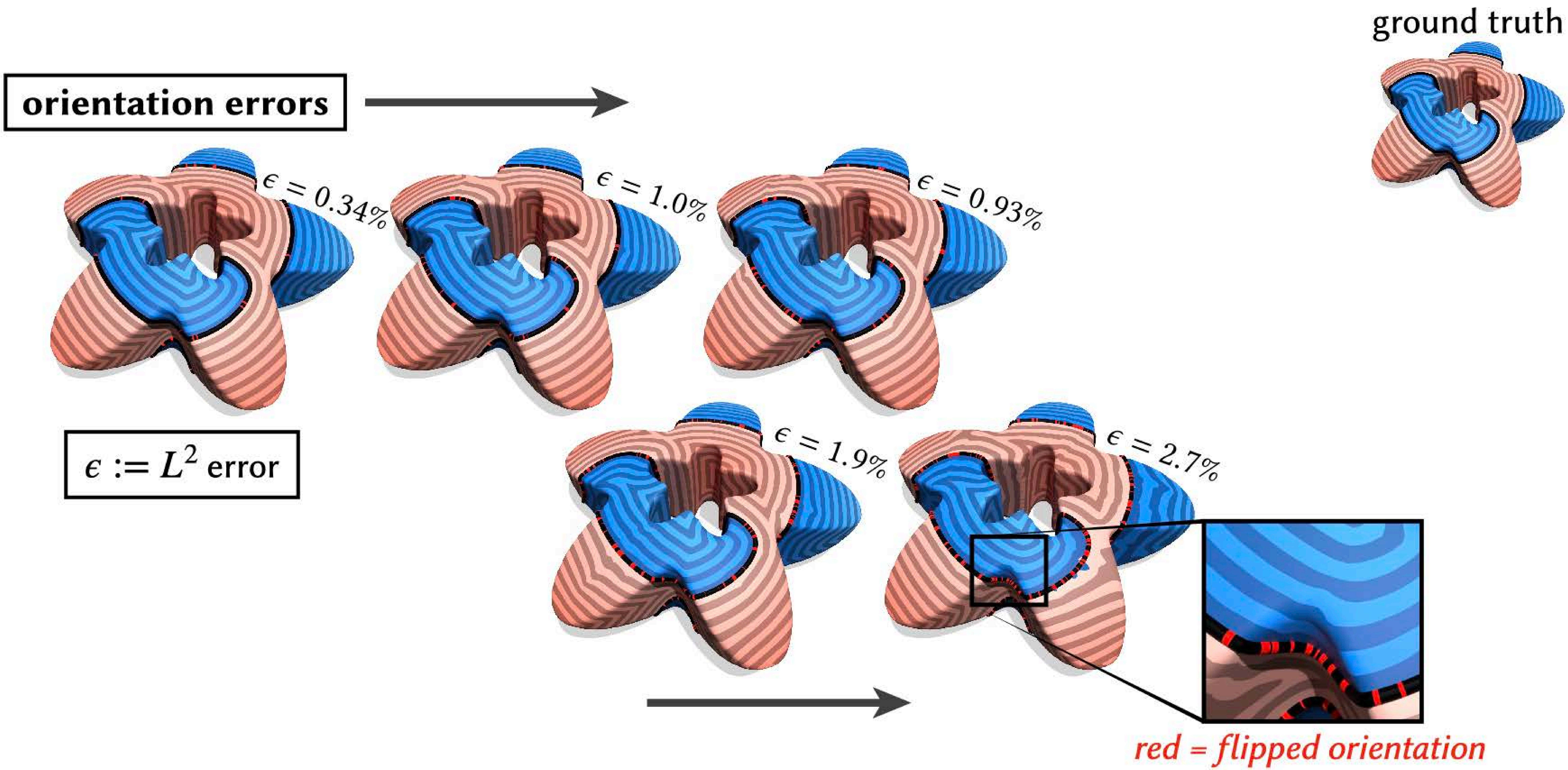
Robustness to errors in the source geometry

geometric errors

$\epsilon := L^2$ error



Robustness to errors in the source geometry

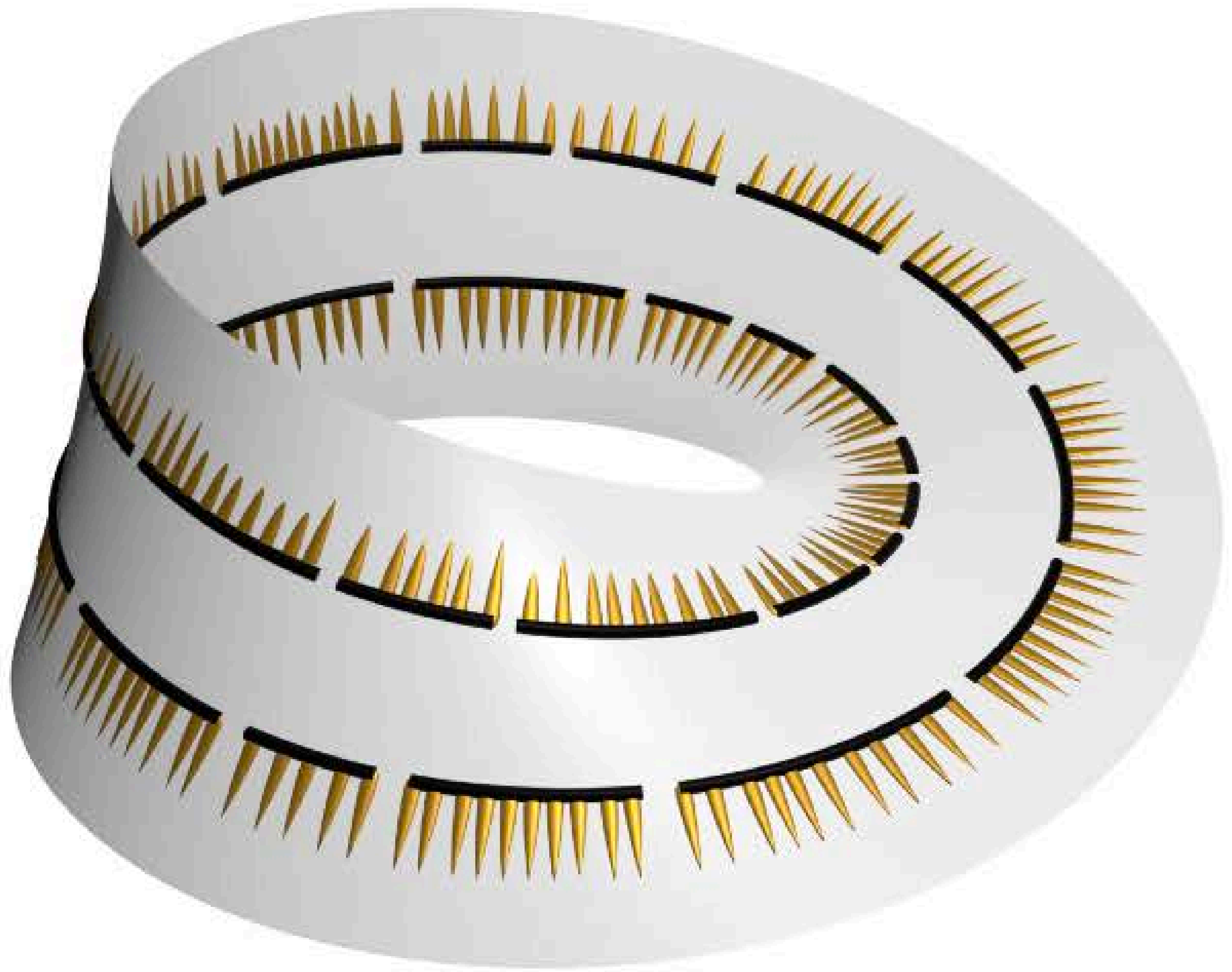


Robustness to errors in the source geometry

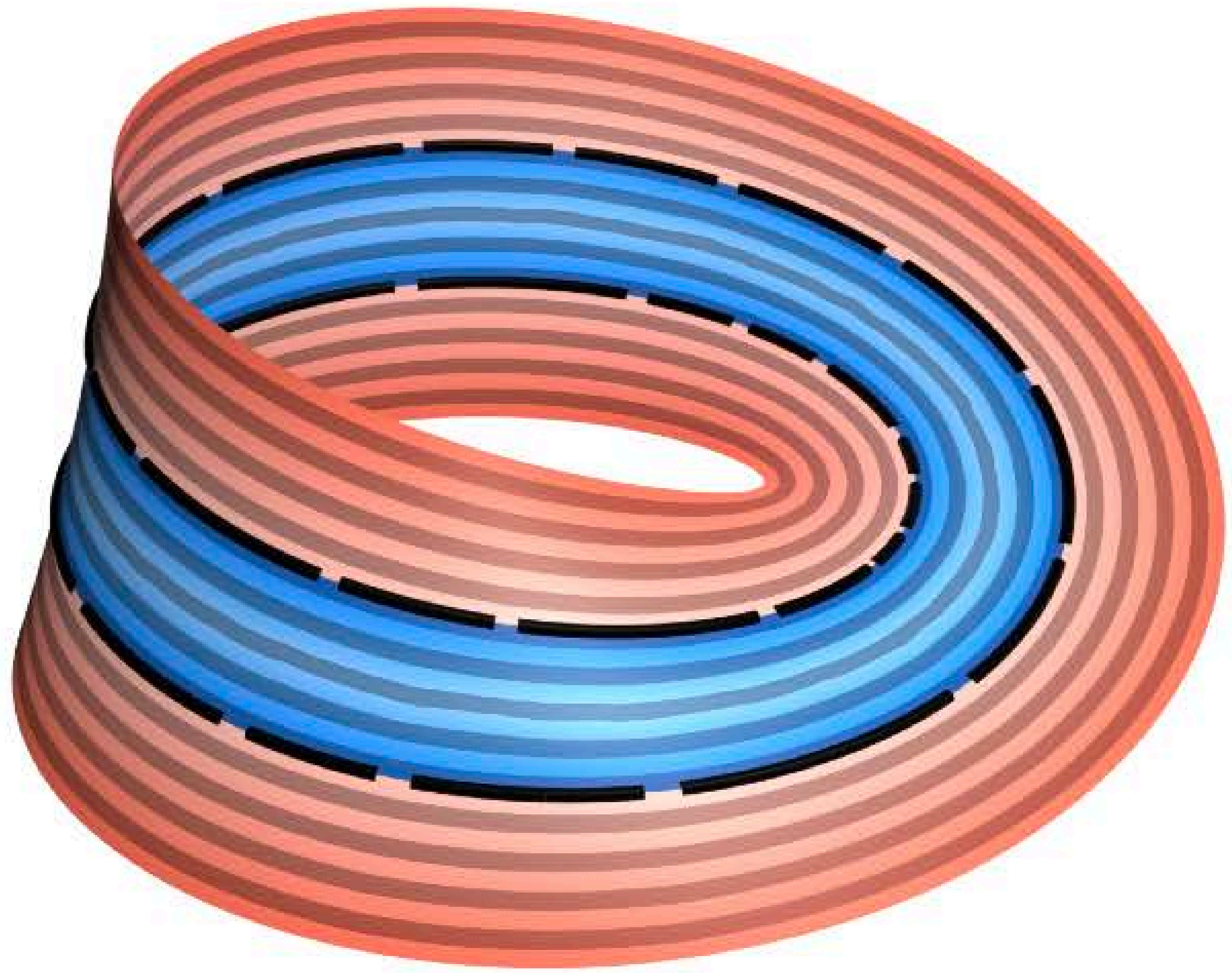
Robustness to errors in the domain geometry

Non-orientable surfaces

input



generalized signed distance



Optional Extensions

Preserving level sets

$$\textit{Step 3:}\ \min_{\phi} \|\nabla\phi - Y_t\|_2^2 \quad \longrightarrow \quad \mathrm{C}\phi = \mathrm{b}$$

Preserving level sets

Step 3: $\min_{\phi} \|\nabla\phi - Y_t\|_2^2 \quad \rightarrow \quad \mathrm{C}\phi = \mathrm{b}$

possible slight deviation

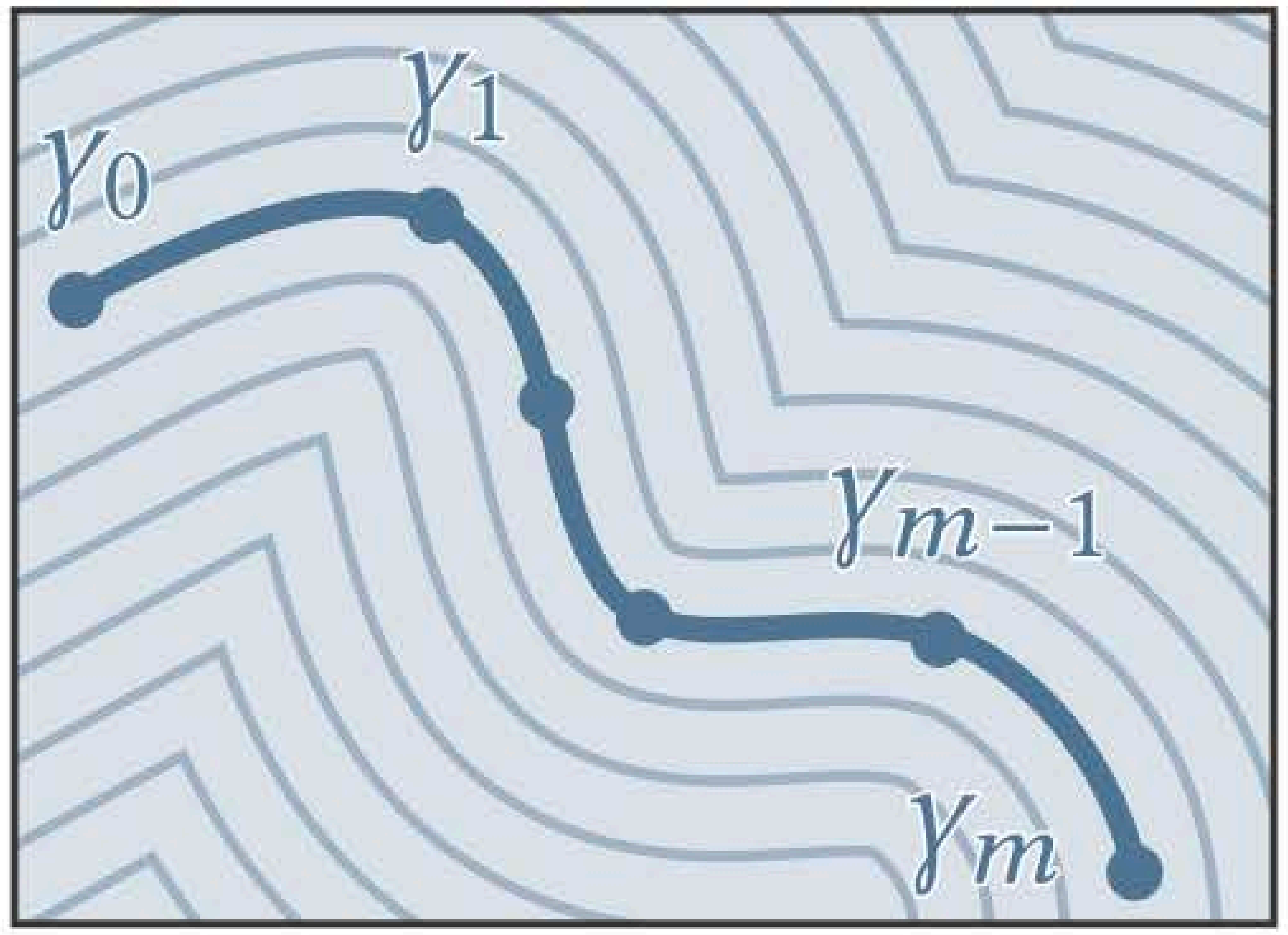
Preserving level sets

Simply constrain ϕ to be constant along curve!

Step 3: $\min_{\phi} \|\nabla\phi - Y_t\|_2^2$ s.t. ϕ constant along (each) curve $\rightarrow \begin{bmatrix} C & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \mu \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$

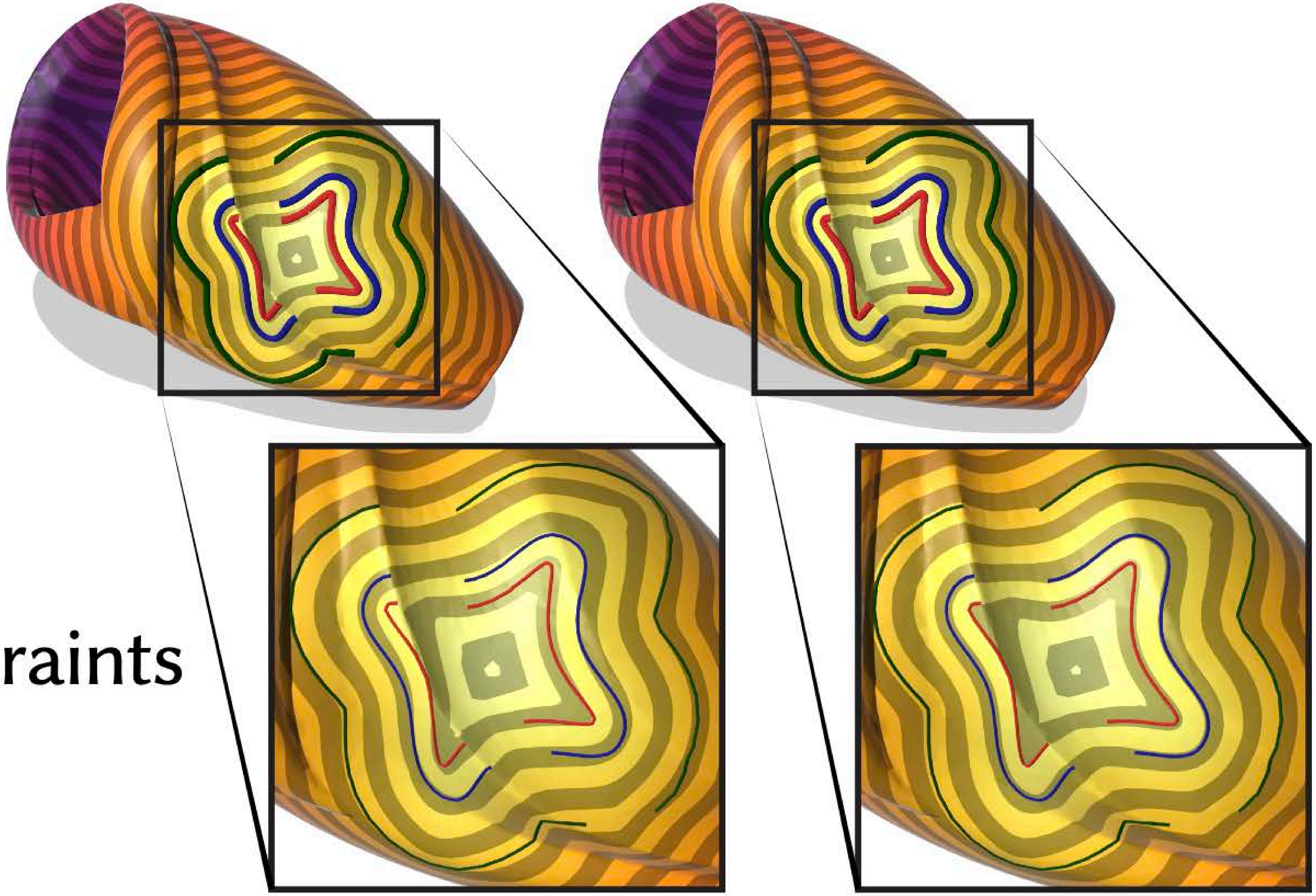
without constraints

with constraints



$$\phi(\gamma_0) = \phi(\gamma_1) = \cdots = \phi(\gamma_m)$$

Match multiple level sets

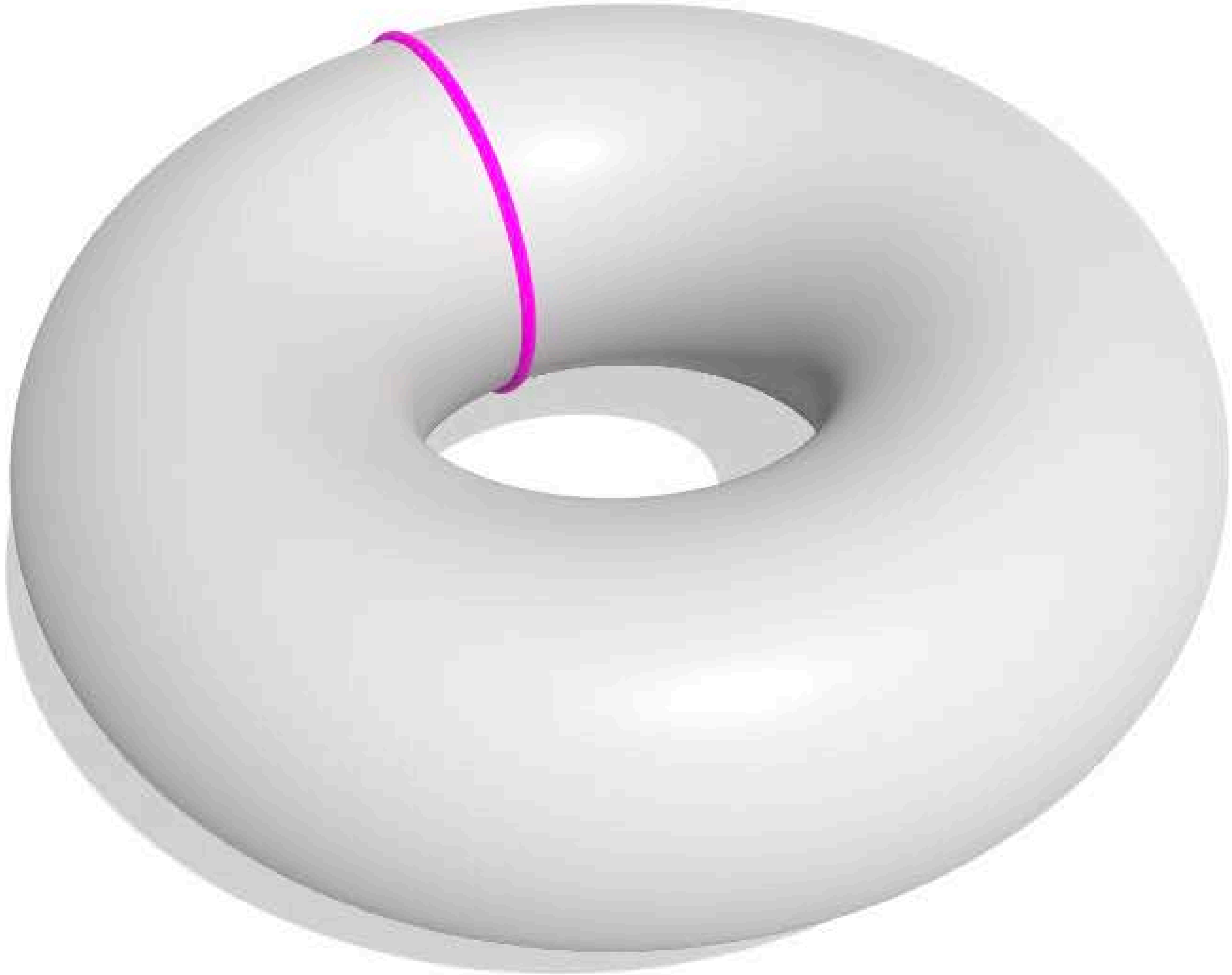


no constraints

constant per component

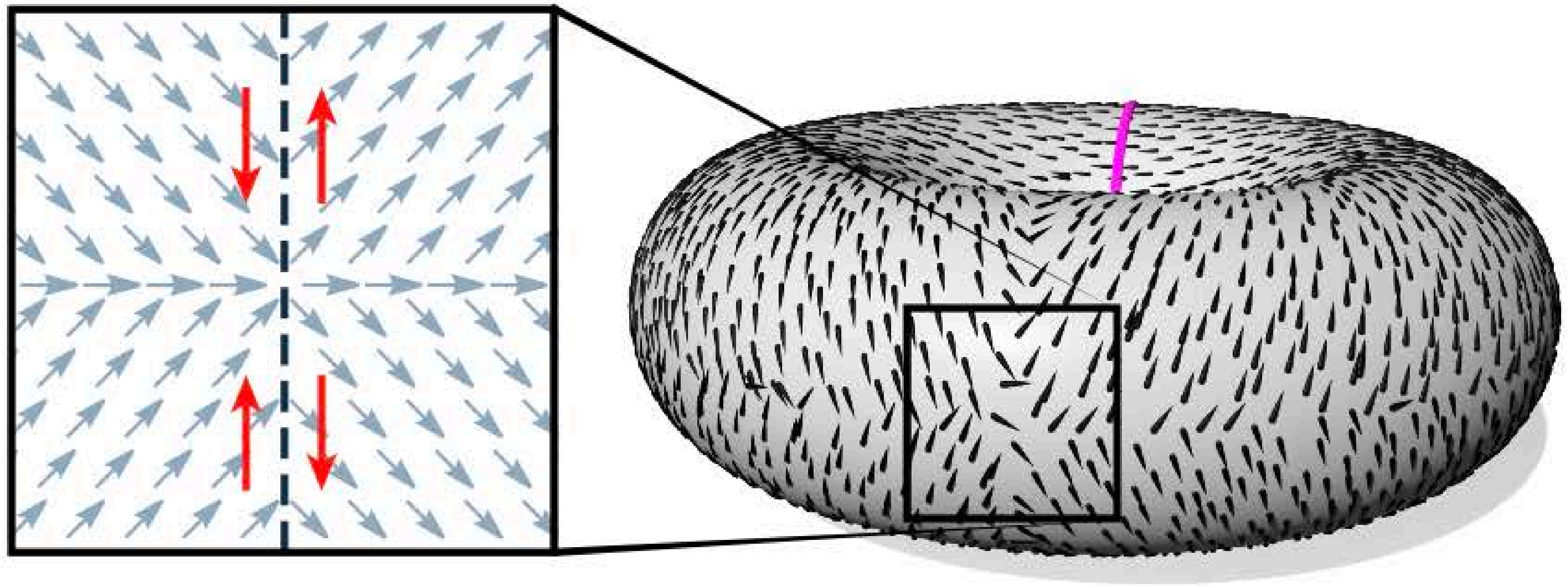
Generalizing signed distance to nonbounding curves

input



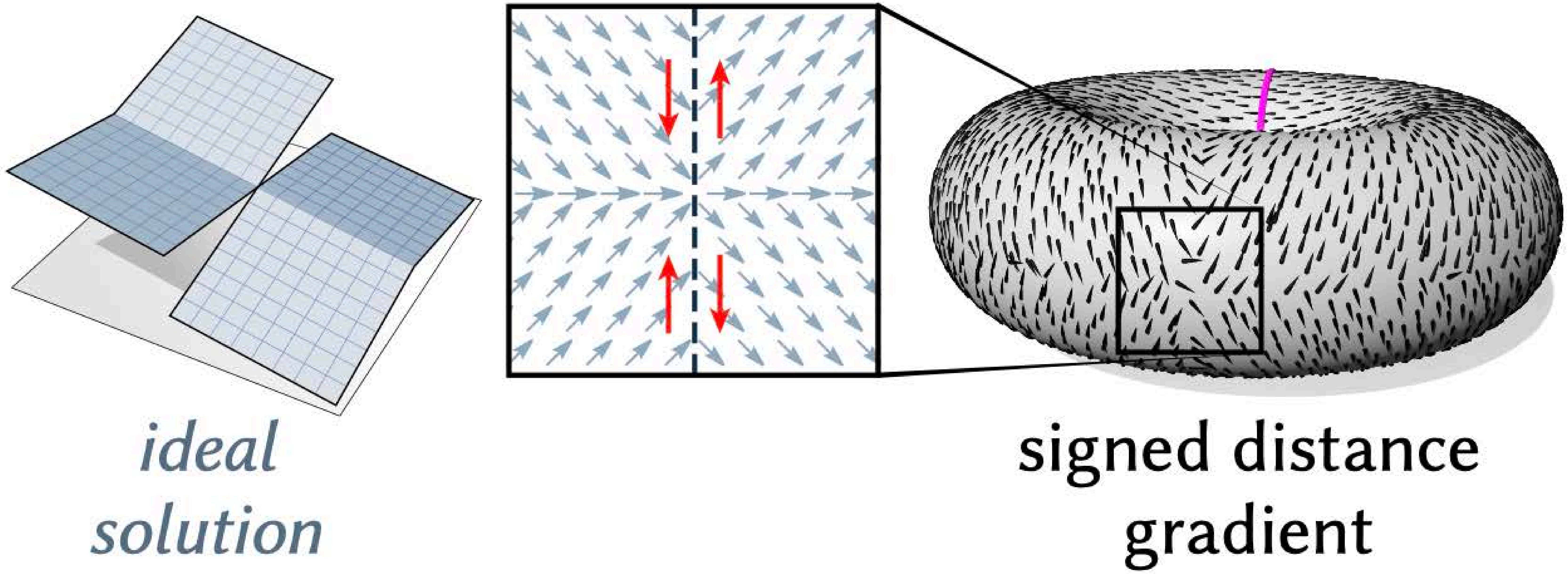
Gradient of signed distance may not be easily integrable

Gradient of signed distance may not be easily integrable

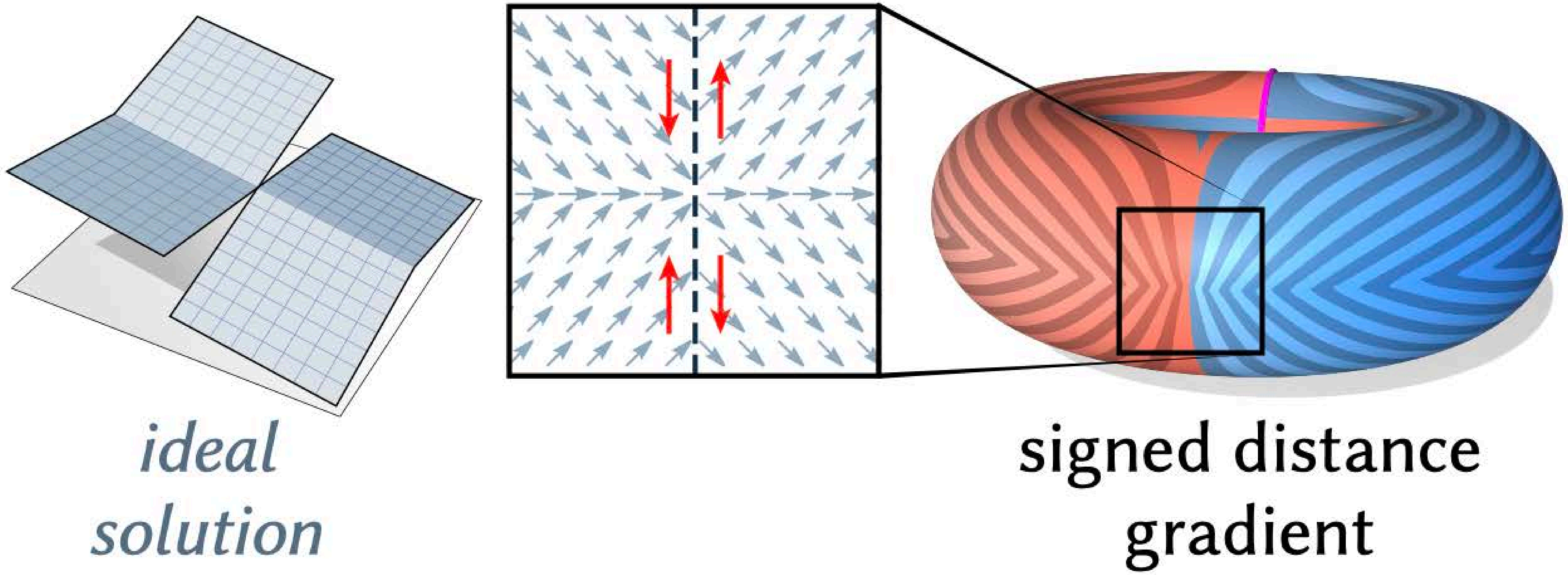


signed distance
gradient

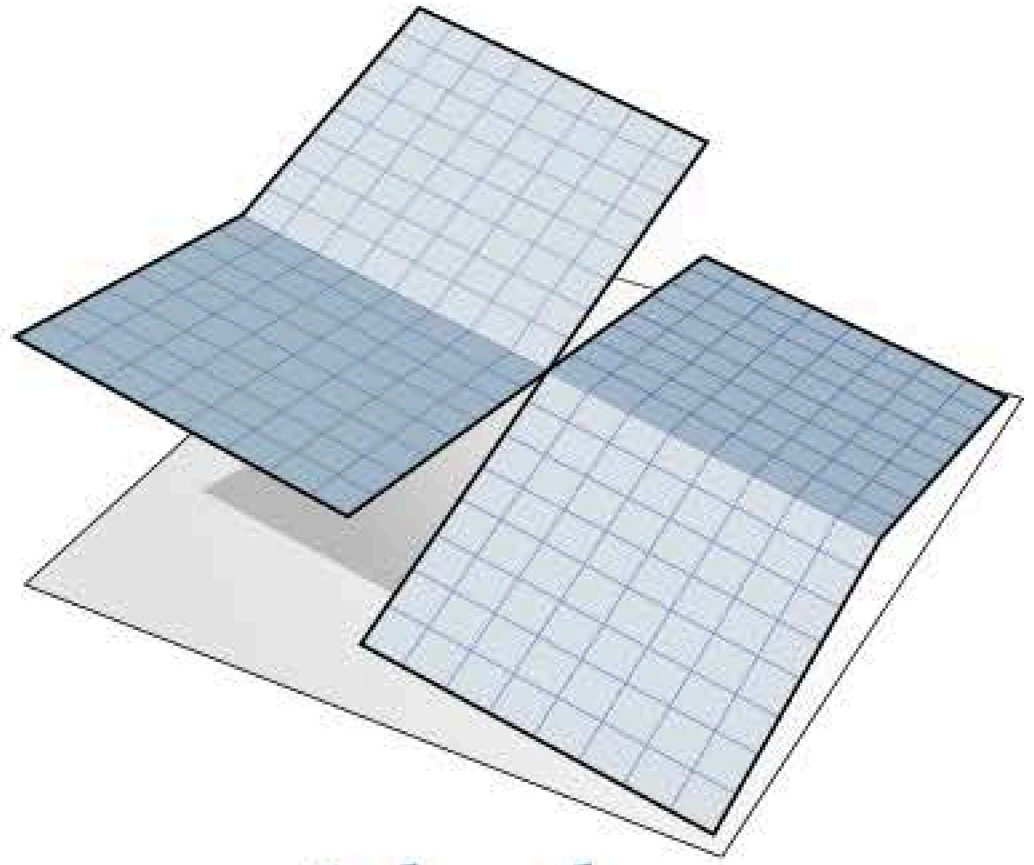
Gradient of signed distance may not be easily integrable



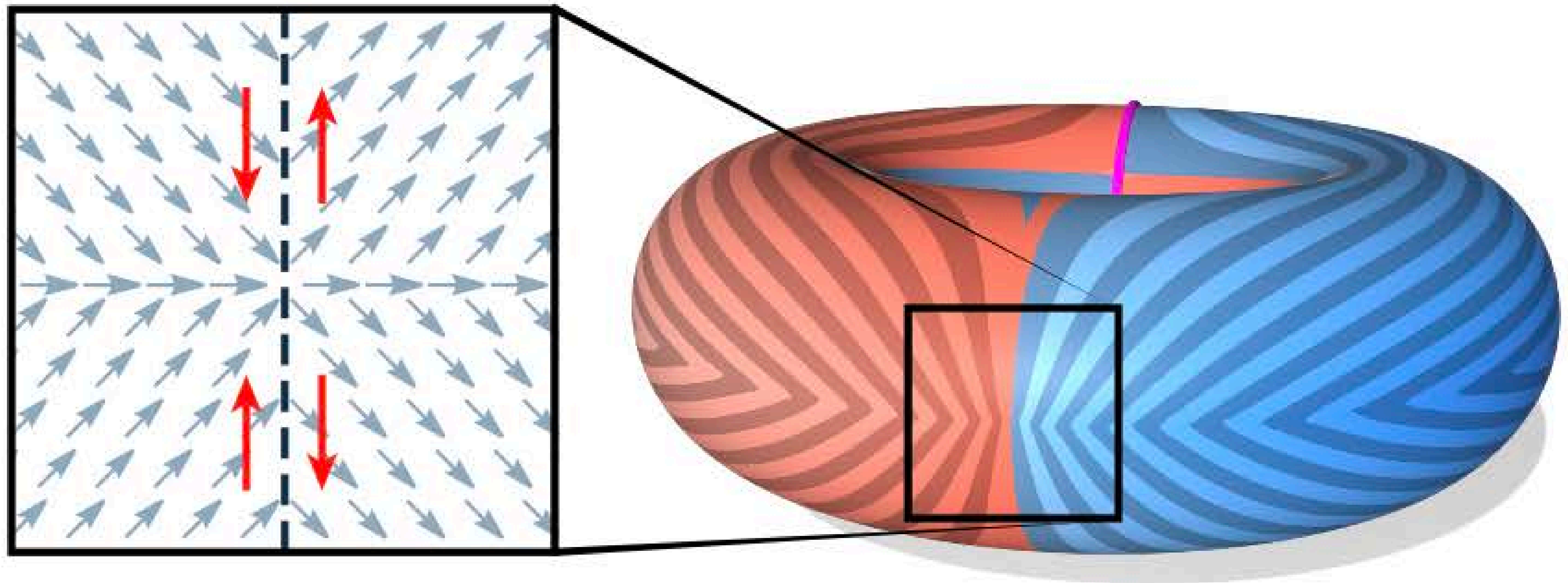
Gradient of signed distance may not be easily integrable



Gradient of signed distance may not be easily integrable



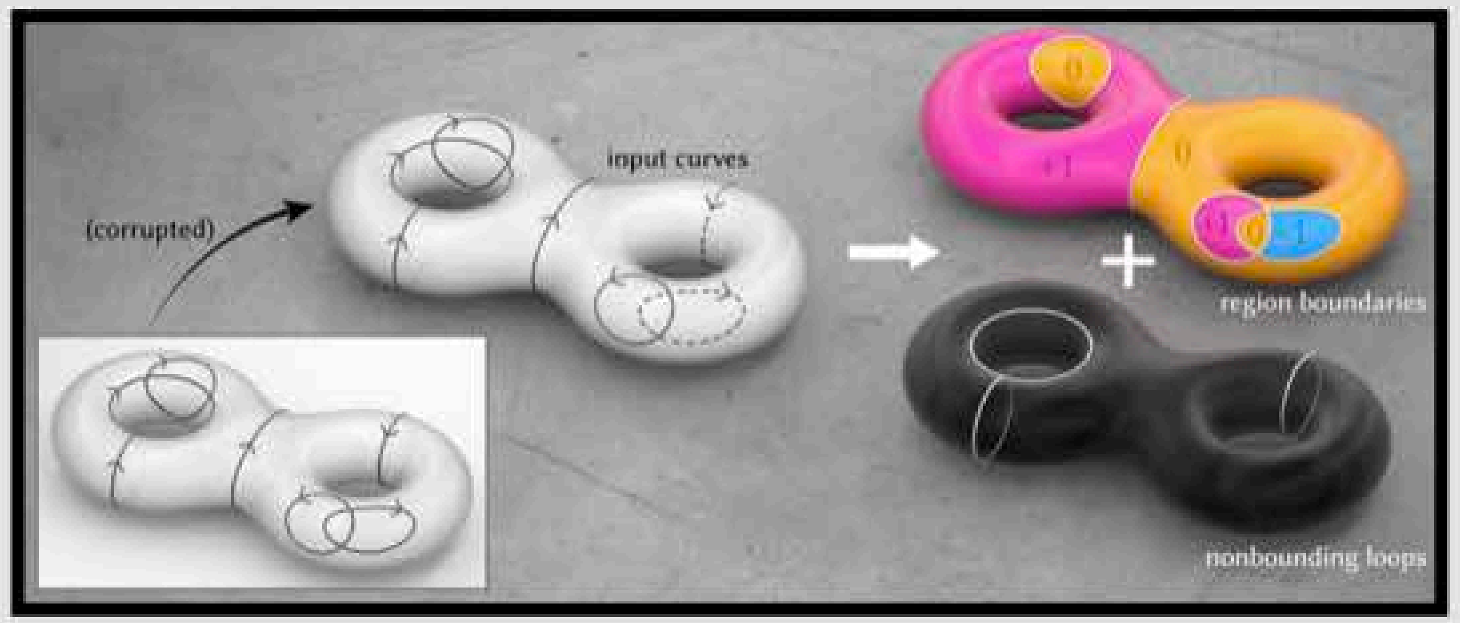
ideal solution



signed distance gradient

Could filter out non-bounding curves:

Feng et al. 2023, "Winding Numbers on Discrete Surfaces"



Instead...

Piecewise continuous distance

Edit Step 3:

$$\min_{\phi\colon M \to \mathbb{R}} \int_M \|\nabla\phi - Y_t\|_2^2$$

Piecewise continuous distance

Edit Step 3:

$$\min_{\phi:\, M\to\mathbb{R}} \int_M \|\nabla\phi - Y_t\|_2^2$$

Allow ϕ to jump where Y is non-integrable,
but otherwise minimize discontinuity

$$\min_{\phi} \sum_{ij\in\text{edges}} [\text{weight}_{\text{integrability}}][\text{jump in } \phi \text{ across } ij]$$

s.t. Y is integrated within each face

integrate Y, allowing for discontinuity

Piecewise continuous distance

Edit Step 3:

$$\min_{\phi:\, M\to\mathbb{R}} \int_M \|\nabla\phi - Y_t\|_2^2$$

Allow ϕ to jump where Y is non-integrable,
but otherwise minimize discontinuity

$$\min_{\phi} \sum_{ij\in \text{edges}} [\text{weight}_{\text{integrability}}][\text{jump in } \phi \text{ across } ij]$$

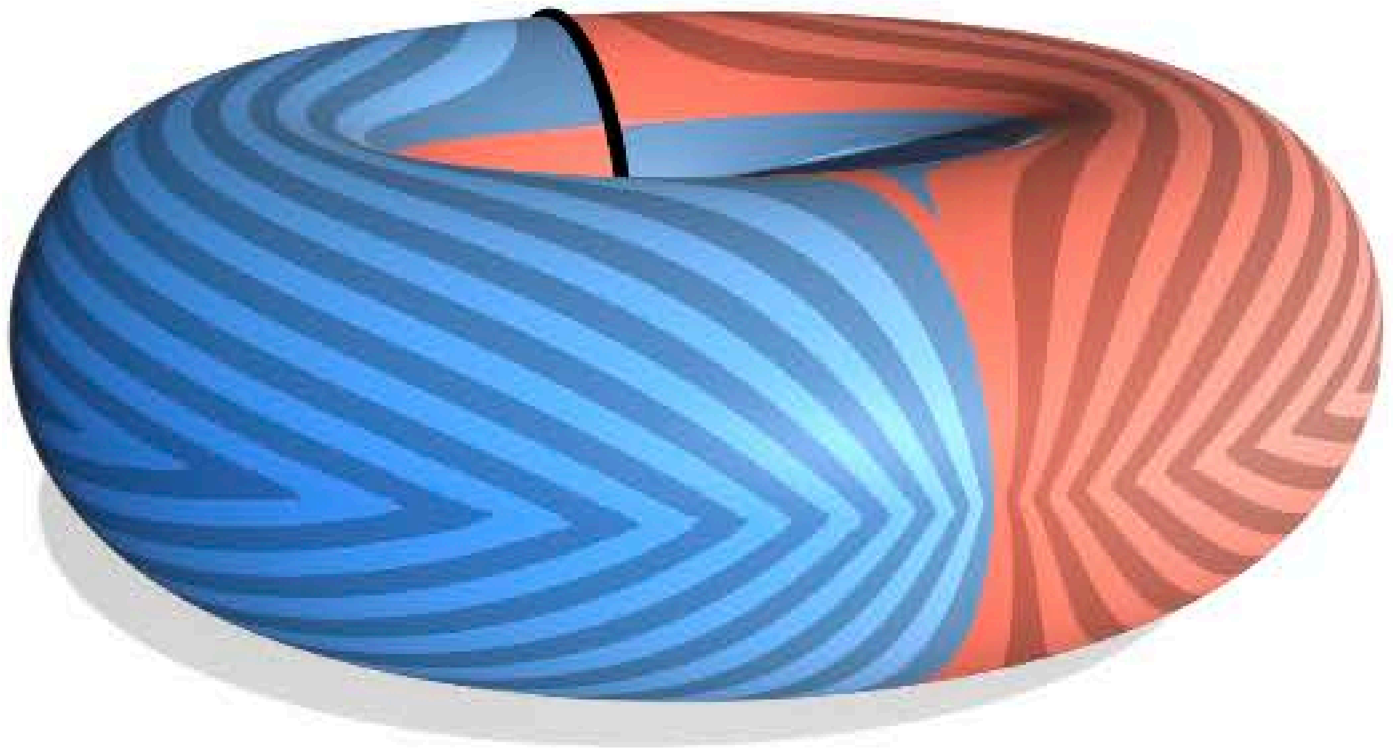
s.t. Y is integrated within each face

integrate Y, allowing for discontinuity

sparse linear program ($|F|$ DOFs)

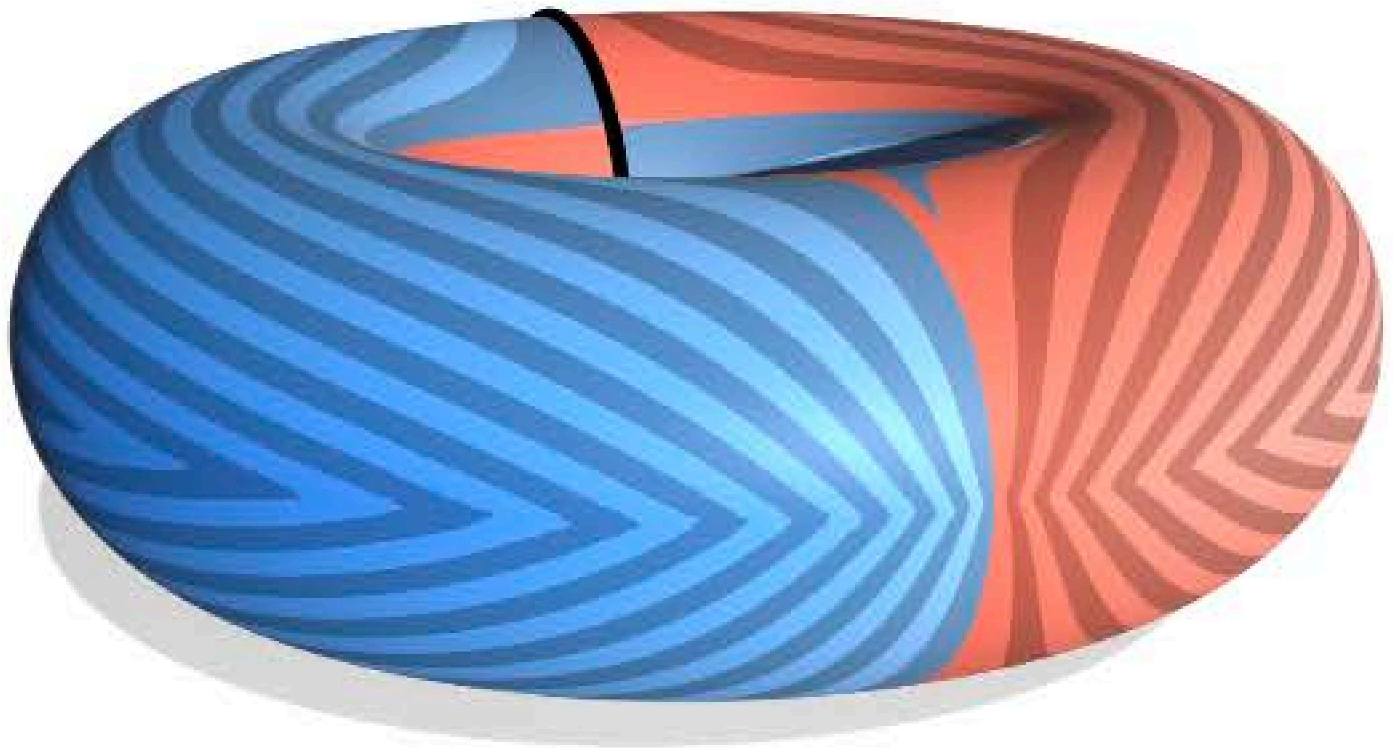
Piecewise continuous distance for nonbounding curves

standard integration (L^2)

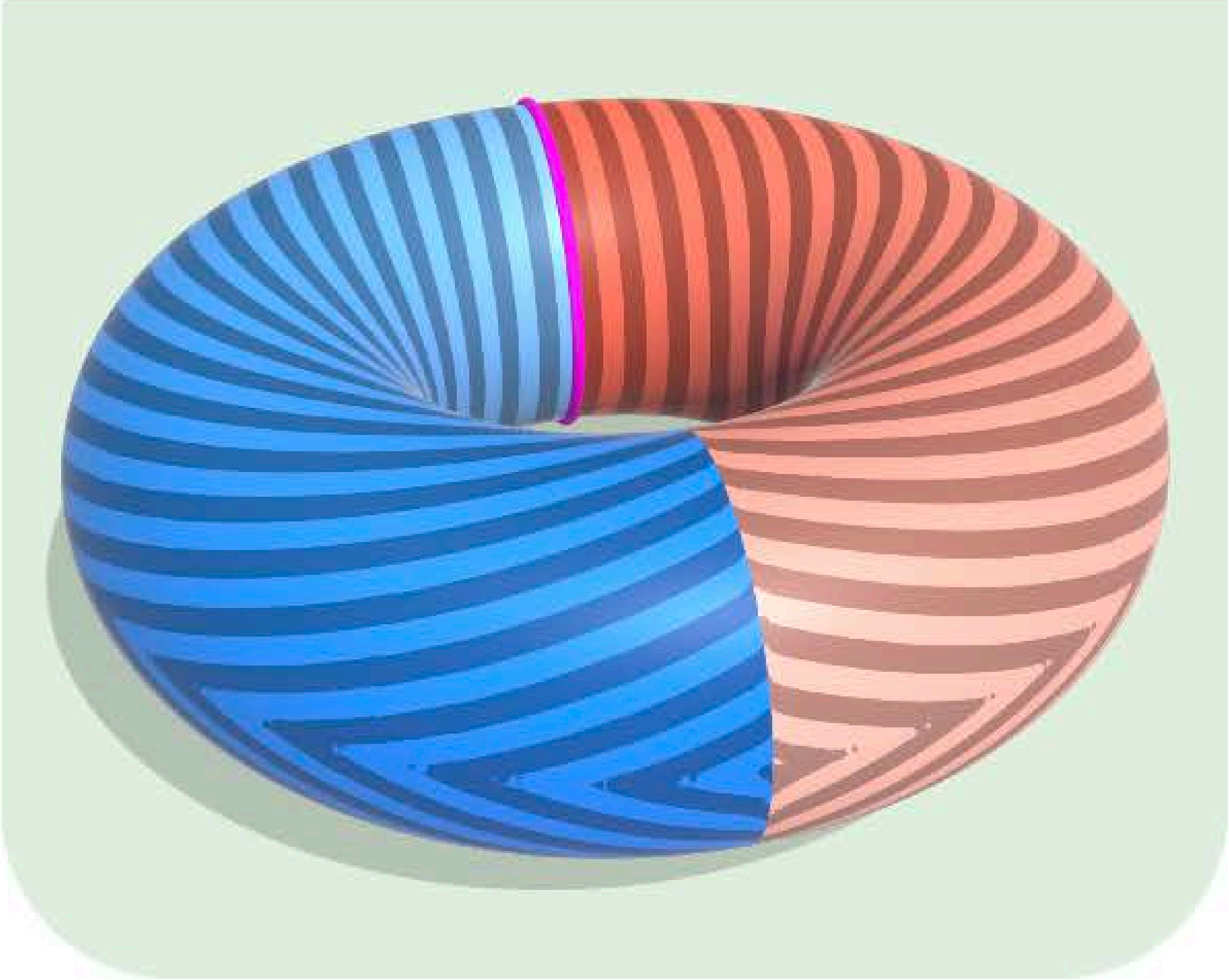


Piecewise continuous distance for nonbounding curves

standard integration (L^2)



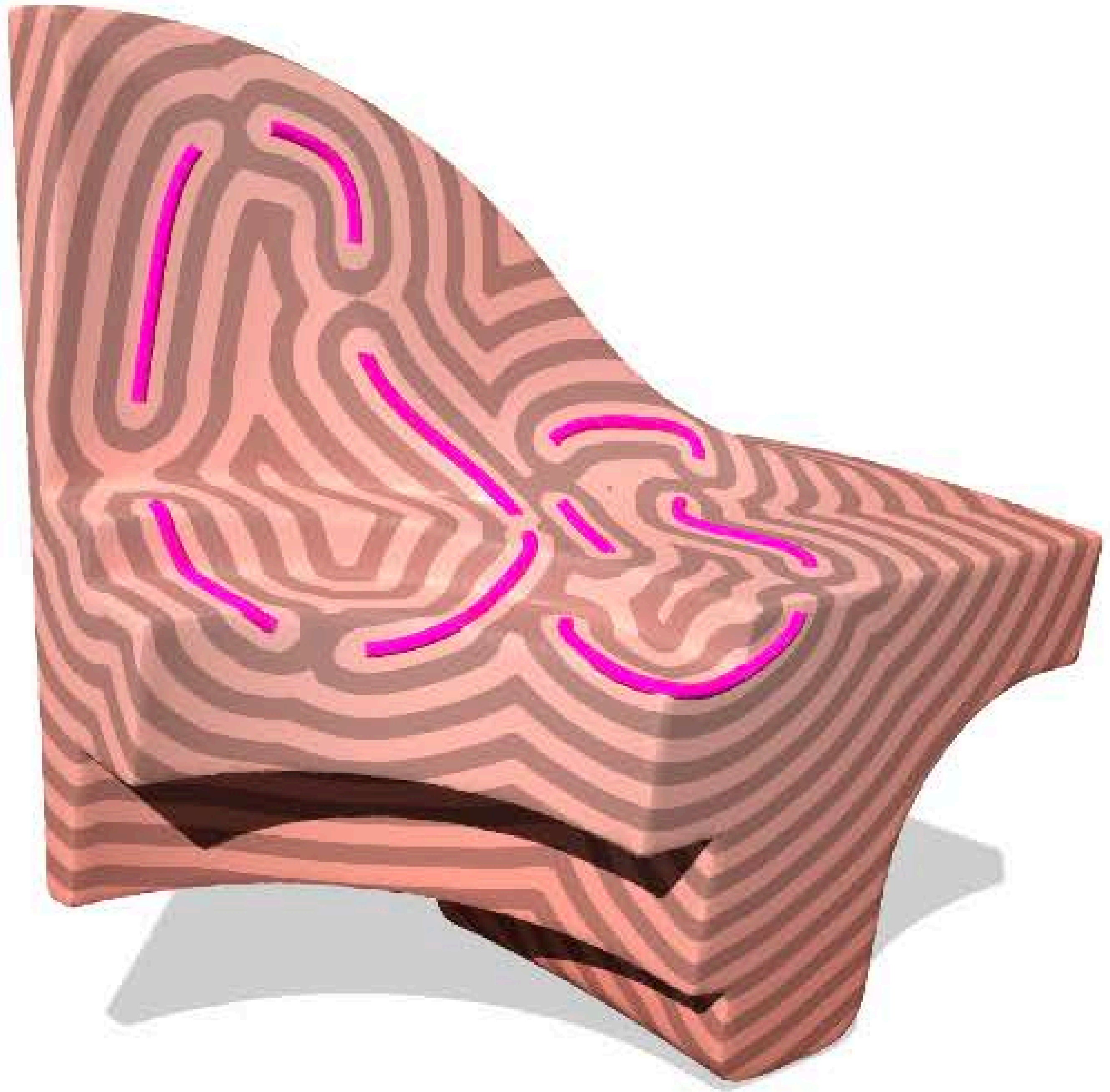
piecewise continuous integration (L^1)



“Sharpening” distance

Unsigned geodesic distance as convex optimization

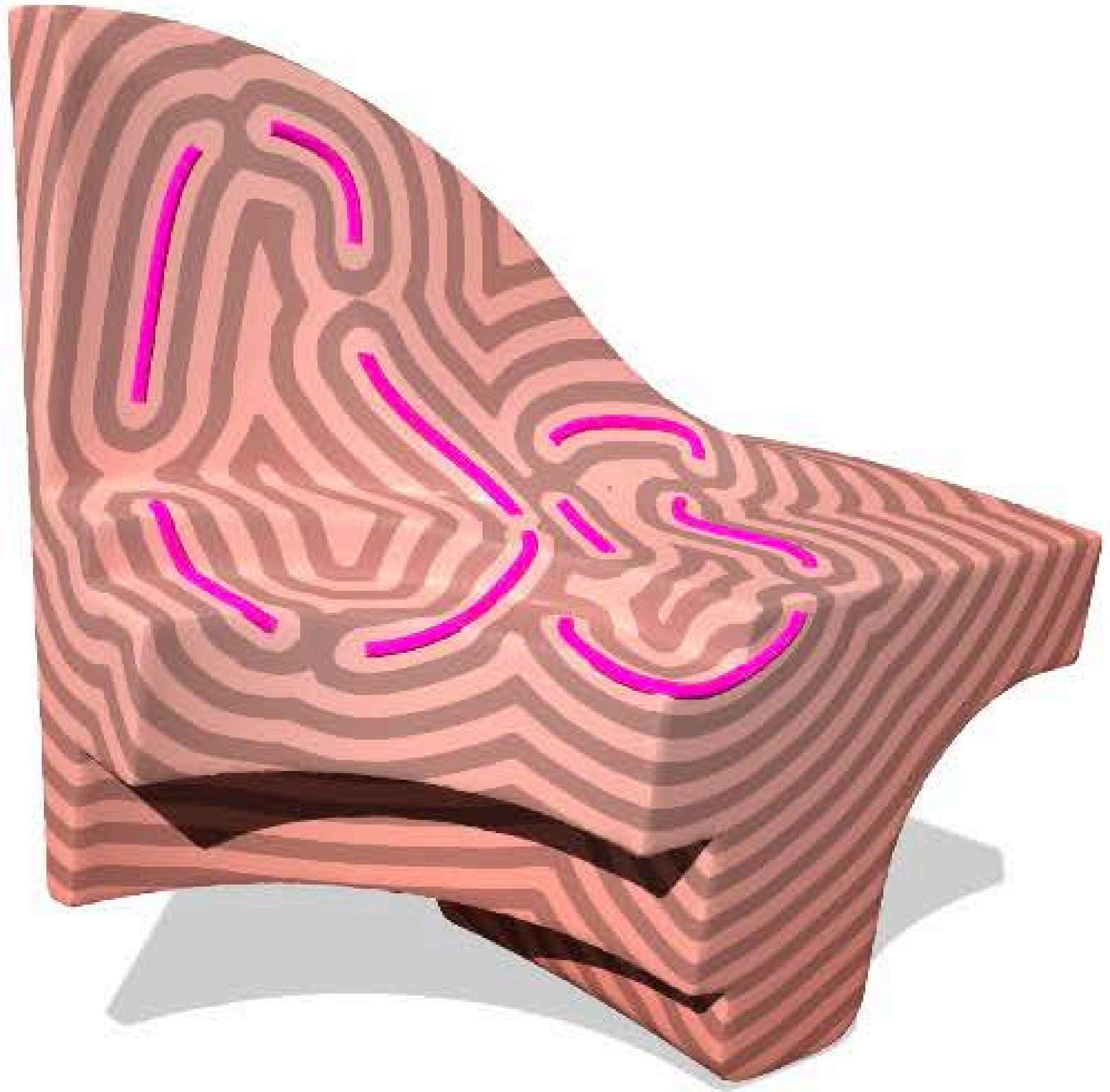
[Dantzig 1963, Belyaev & Fayolle 2020]



“Sharpening” distance

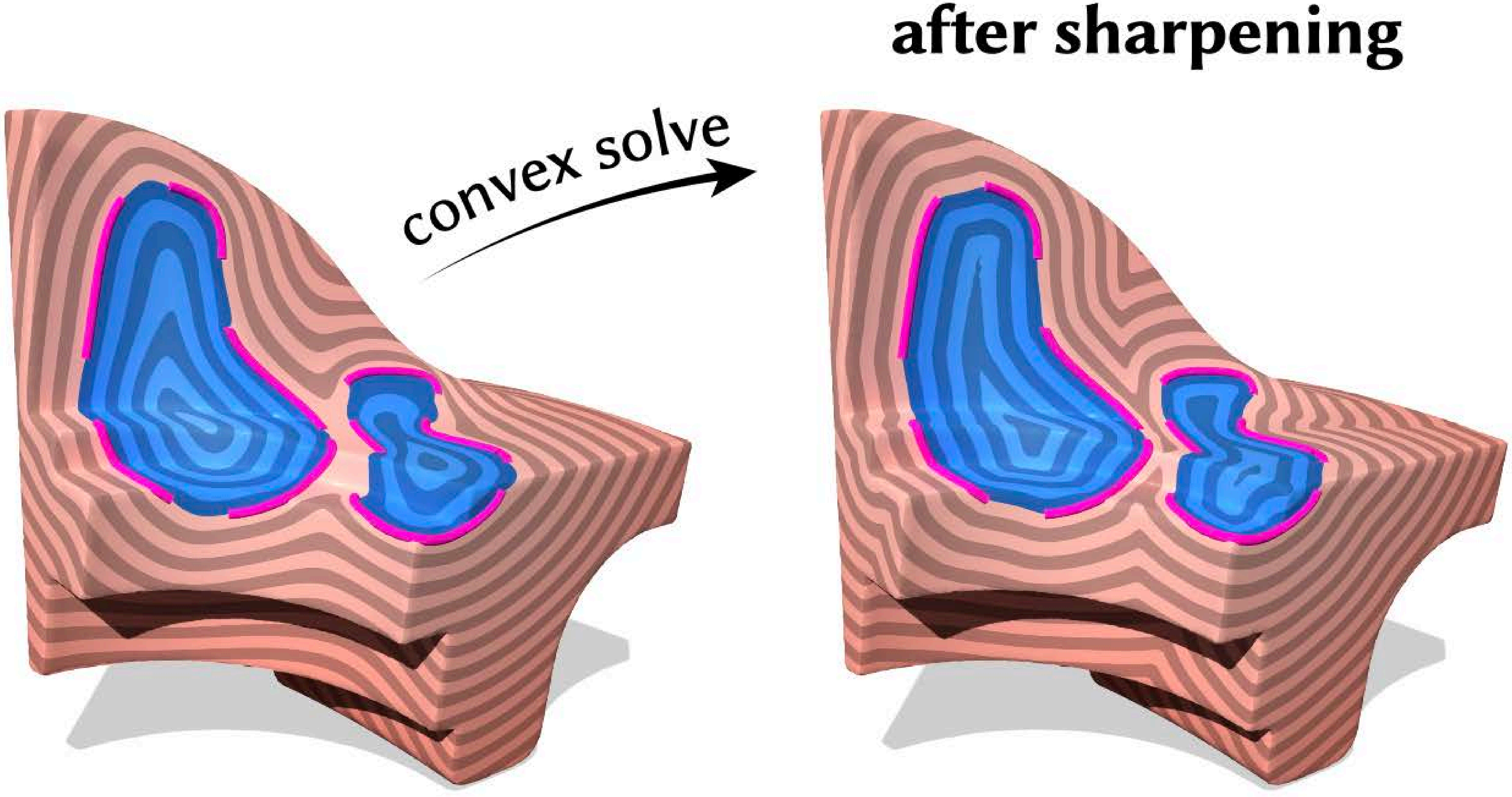
Unsigned geodesic distance as convex optimization

[Dantzig 1963, Belyaev & Fayolle 2020]



unsigned distance only

“Sharpening” distance



solve time: 0.51s

(using $t = 100h^2$ for illustration)

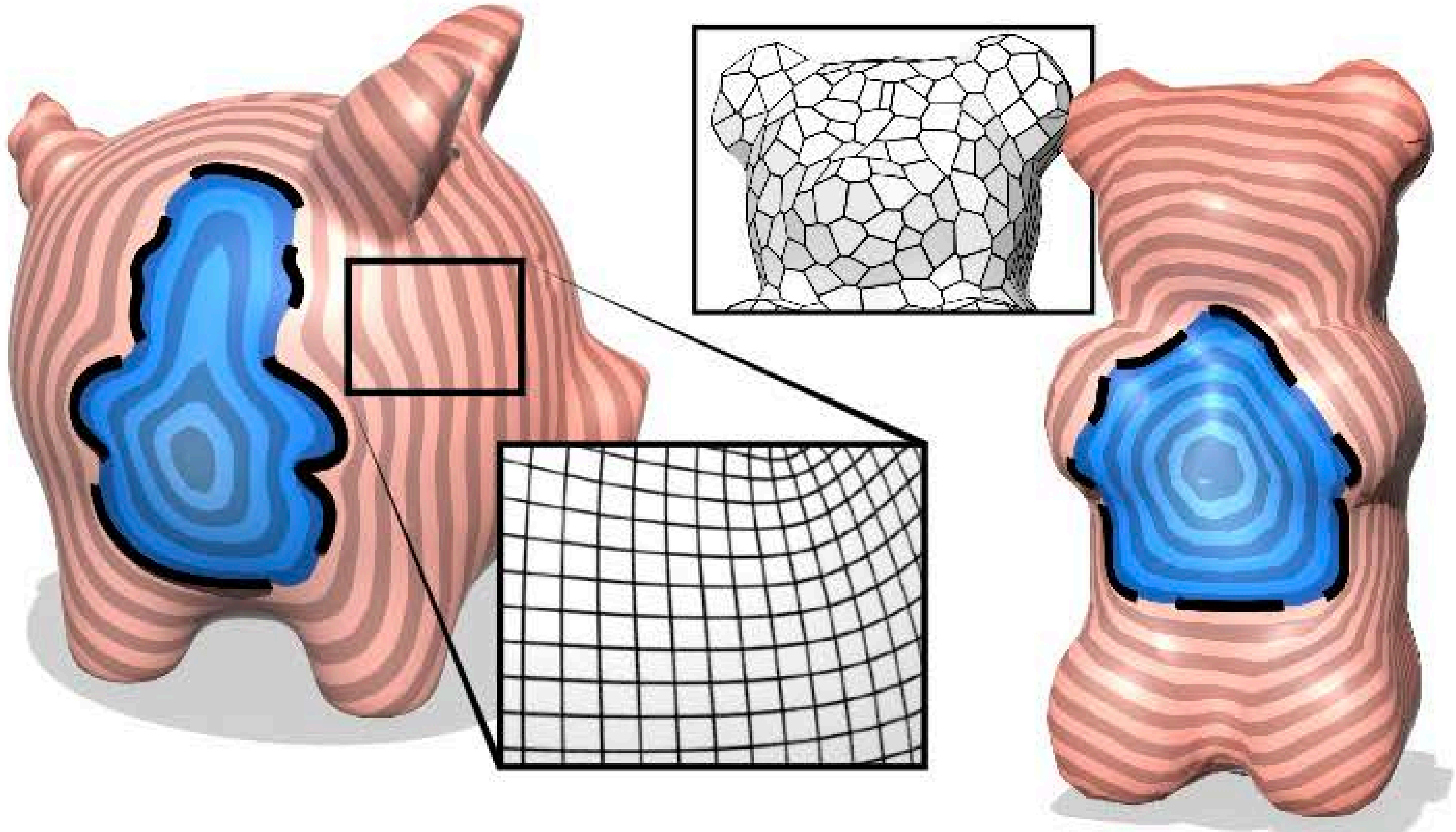
additional time: 0.66s

Other spatial discretizations

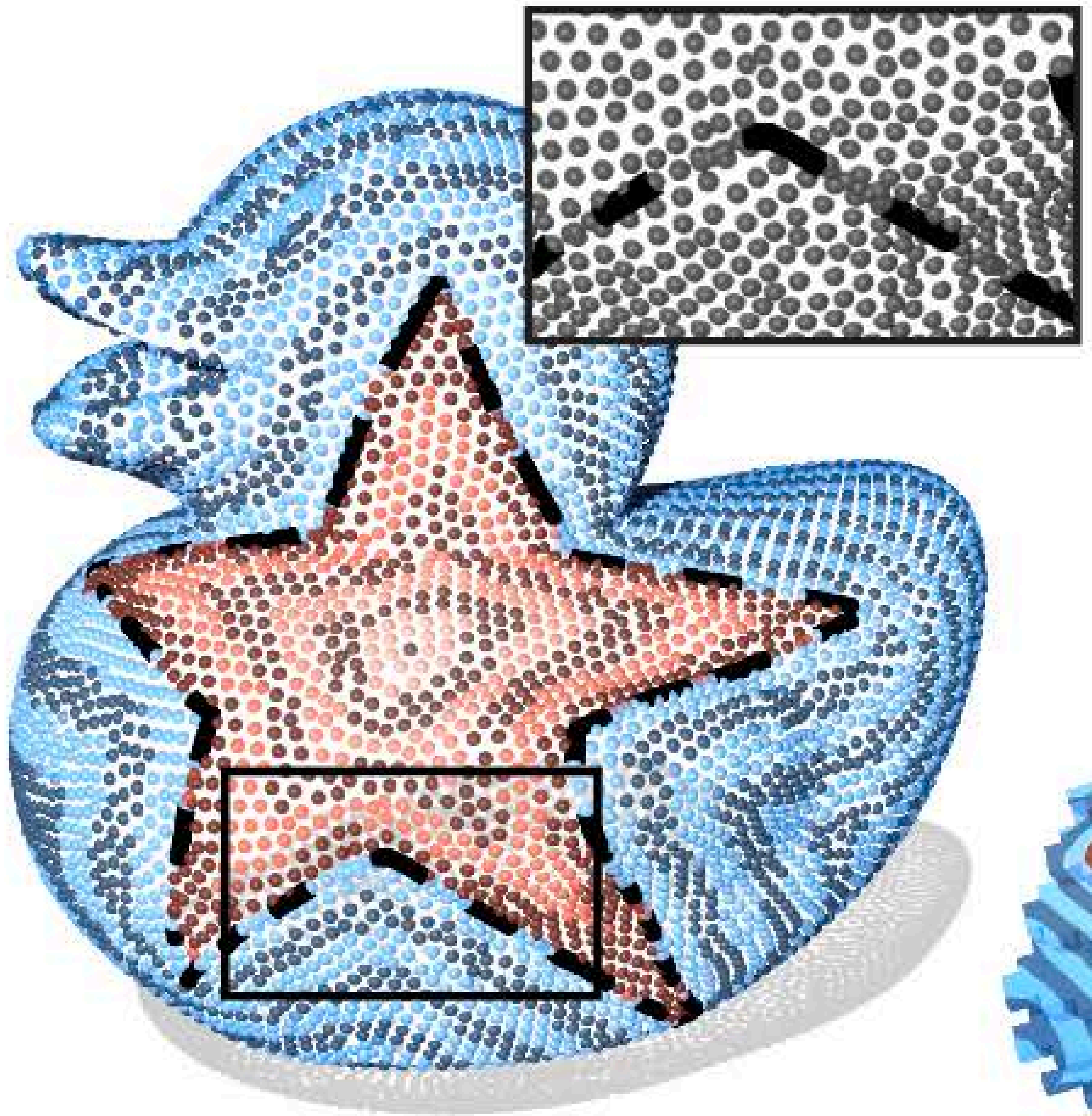
Other spatial discretizations

Other spatial discretizations

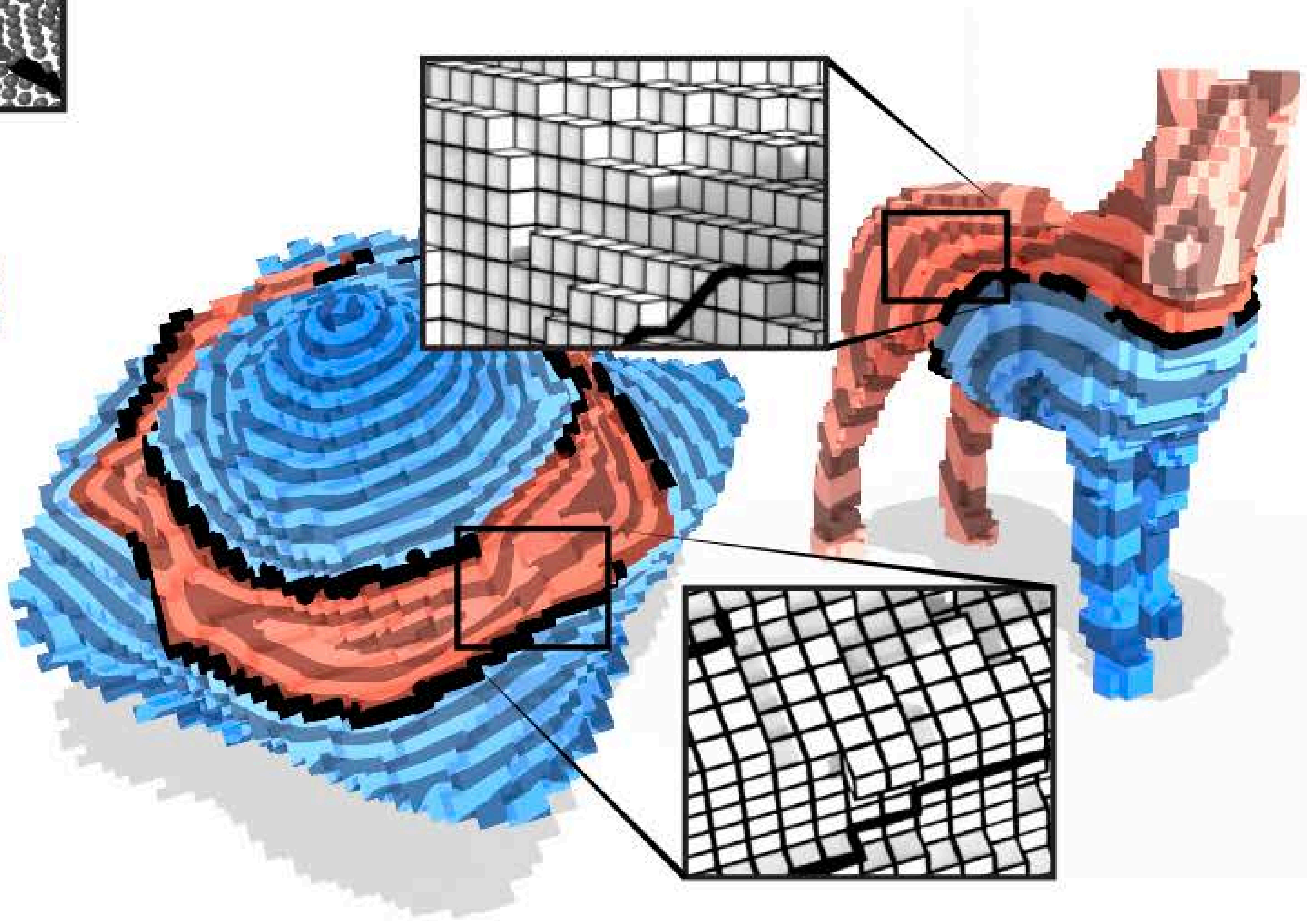
polygon meshes



point clouds



voxelized sufaces



All you need is a Laplacian!

A. Bunge, P. Herholz, M. Kazhdan, M. Botsch. 2020.
Polygon Laplacian Made Simple.

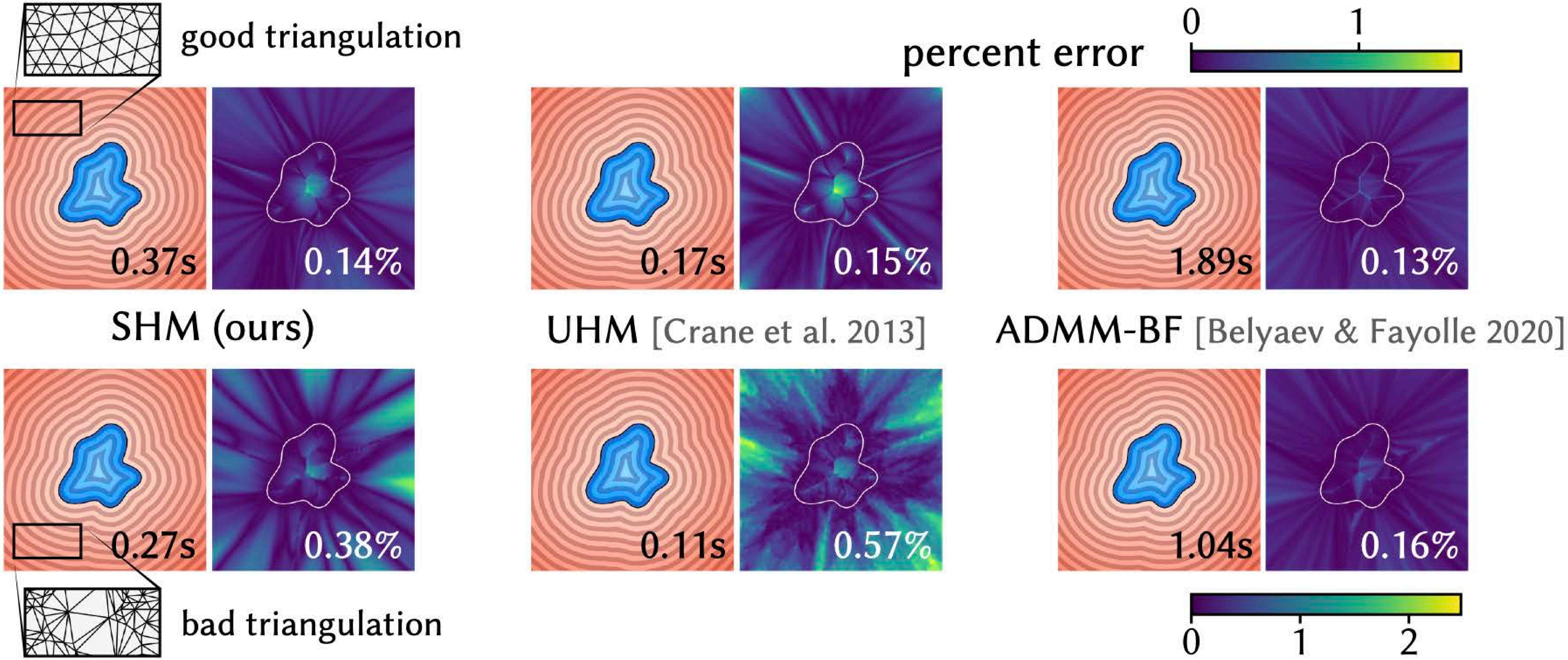
N. Sharp, K. Crane. 2020.
A Laplacian for Nonmanifold Triangle Meshes.

D. Coeurjolly, J. Lachaud. 2022.
A Simple Discrete Calculus for Digital Surfaces.

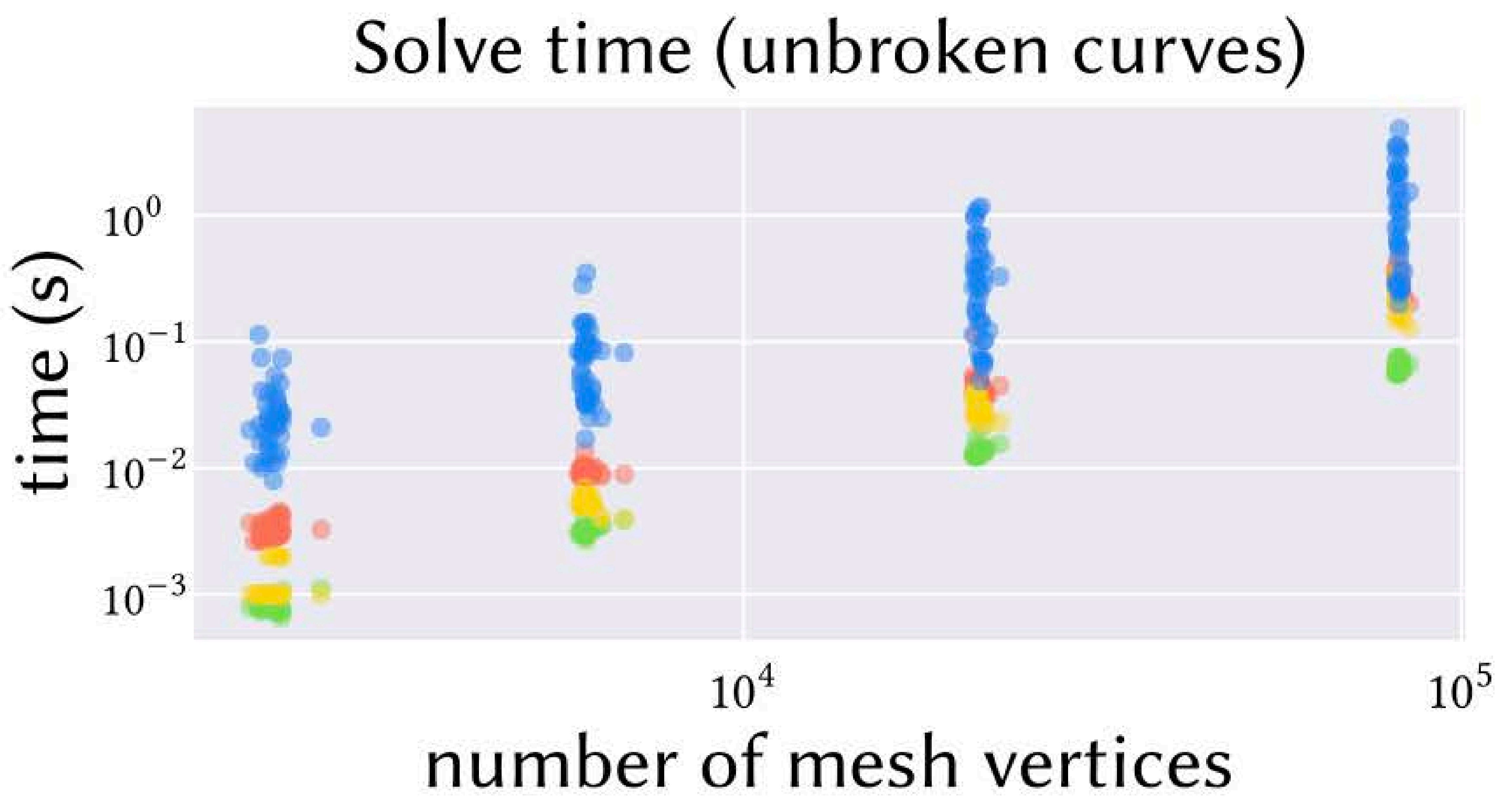
EVALUATION

Evaluation: closed curves on flat domains

Evaluation: closed curves on flat domains



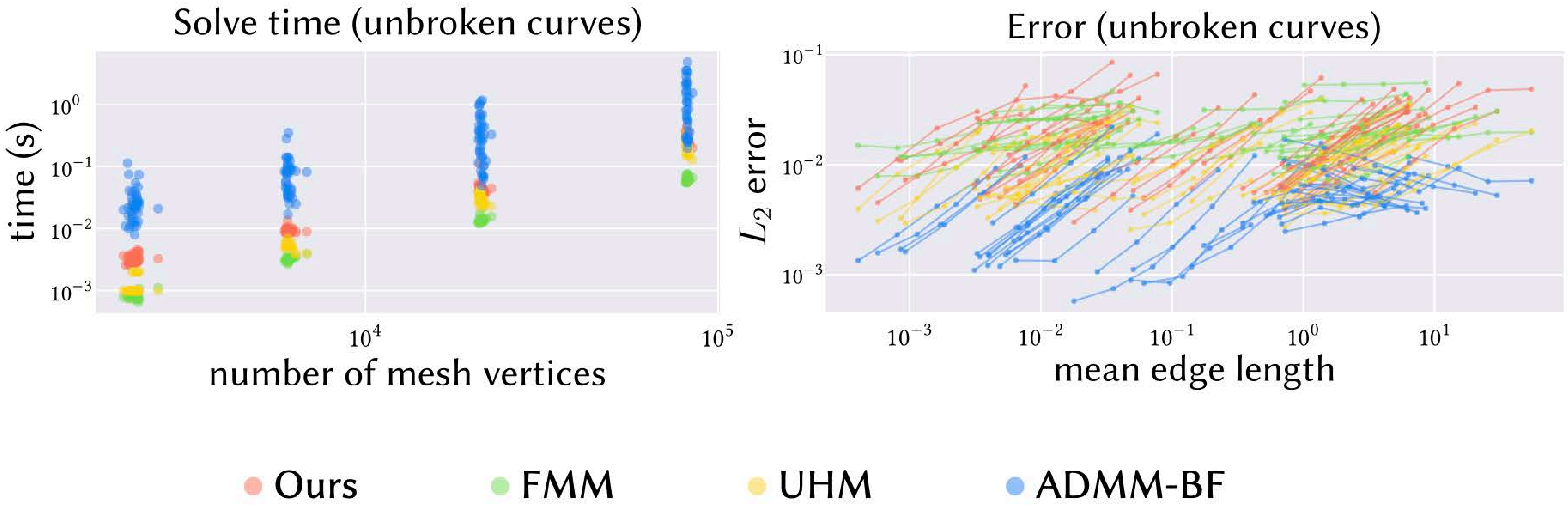
Evaluation: closed curves on curved domains



Ours FMM UHM ADMM-BF

Evaluation: closed curves on curved domains

linear convergence



Better SDFs than repairing then computing distance

Better SDFs than repairing then computing distance

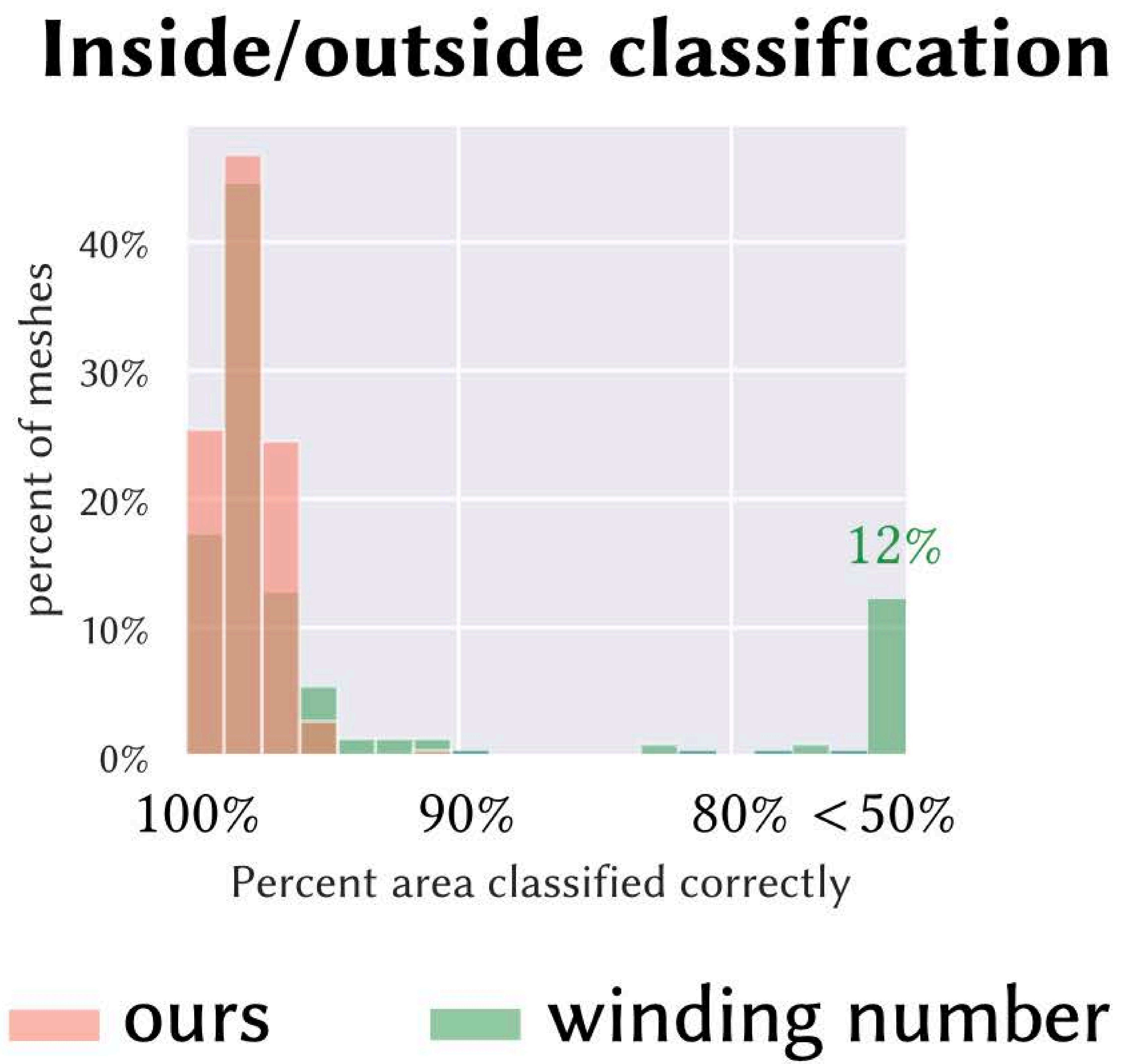
Feng et al. 2023,
“Winding Numbers on
Discrete Surfaces”

Better SDFs than repairing then computing distance

Feng et al. 2023, "Winding Numbers on Discrete Surfaces"

Better inside/outside than winding numbers

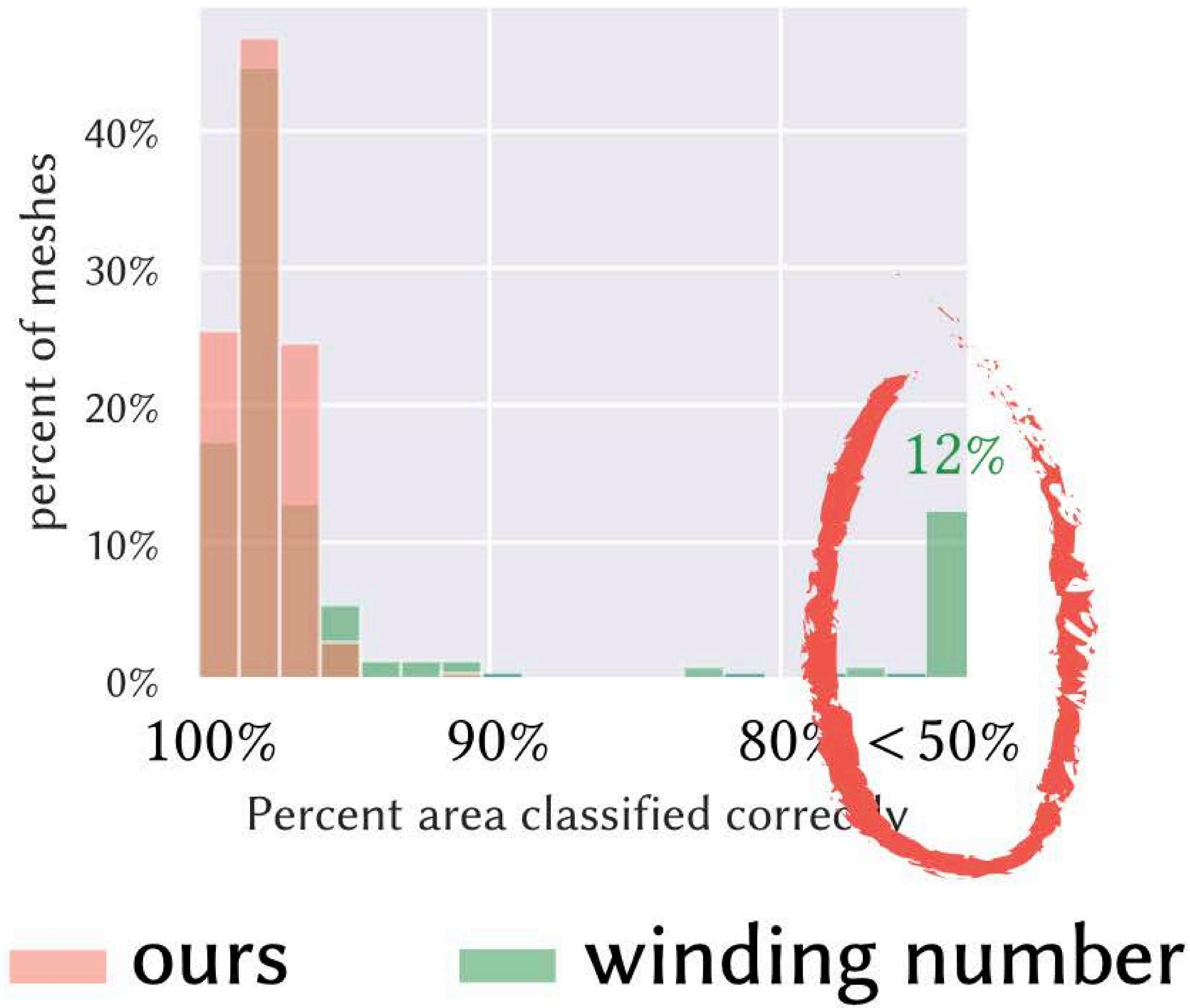
220
examples



Better inside/outside than winding numbers

220
examples

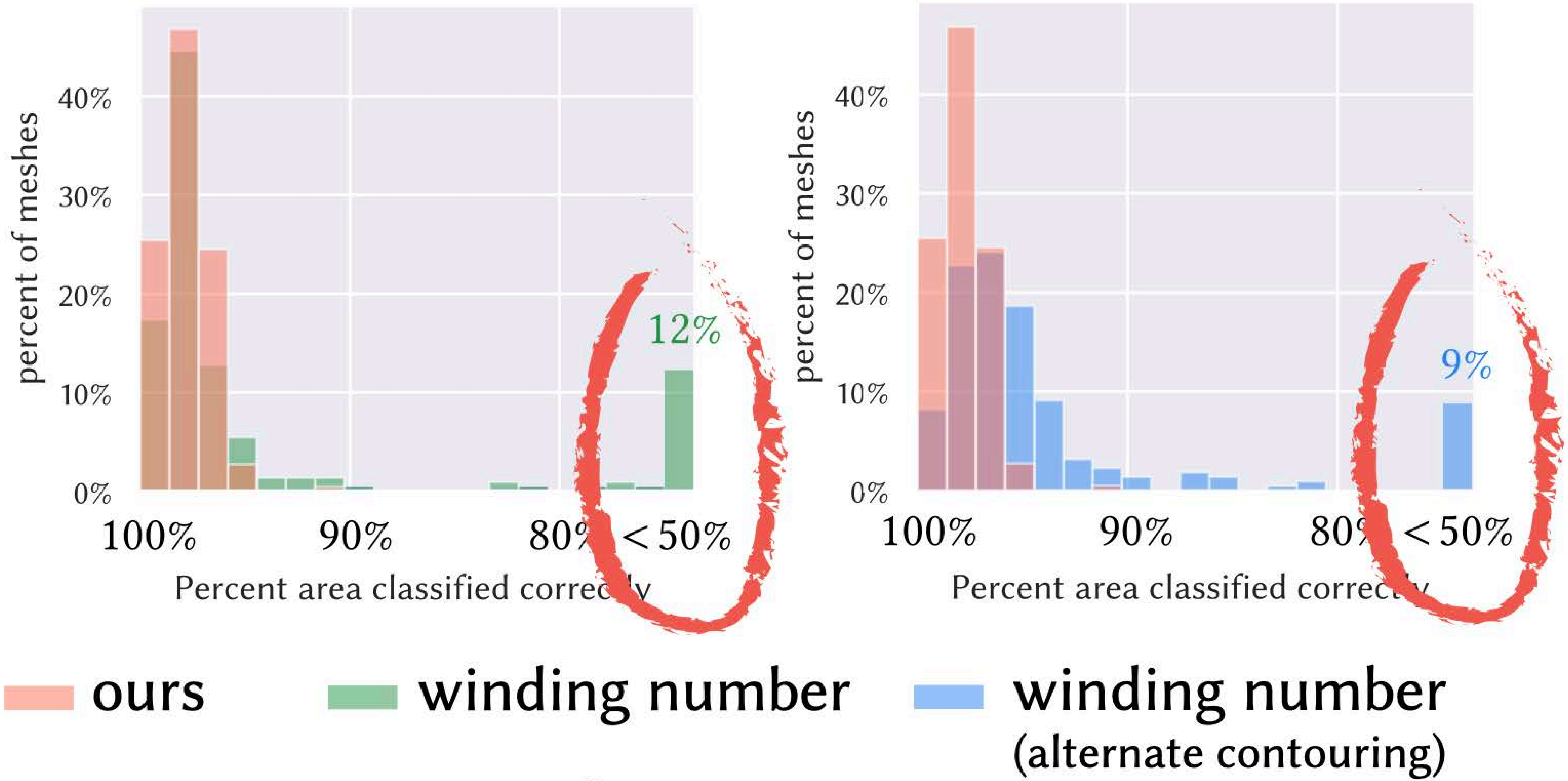
Inside/outside classification



Better inside/outside than winding numbers

220 examples

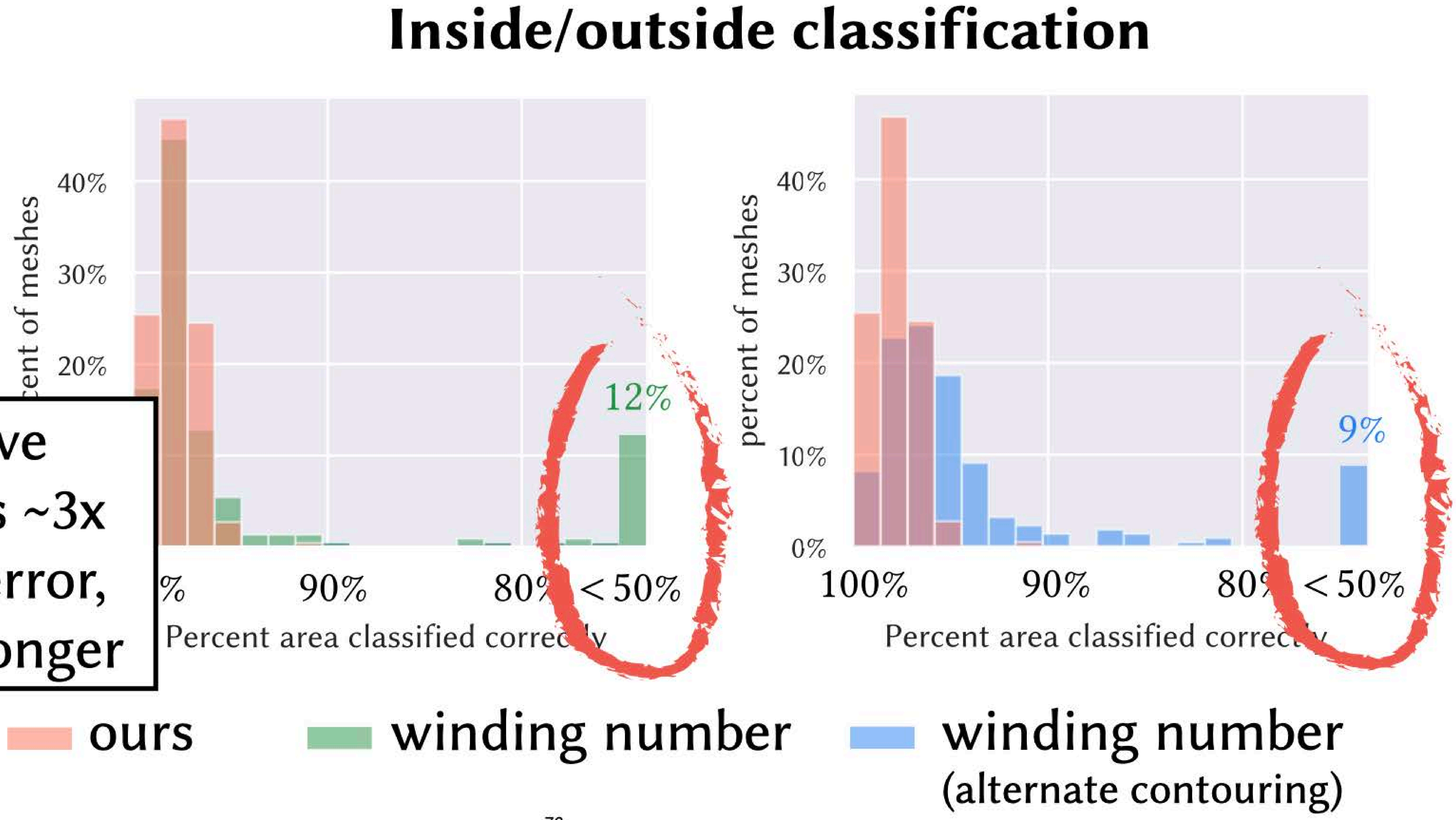
Inside/outside classification



Better inside/outside than winding numbers

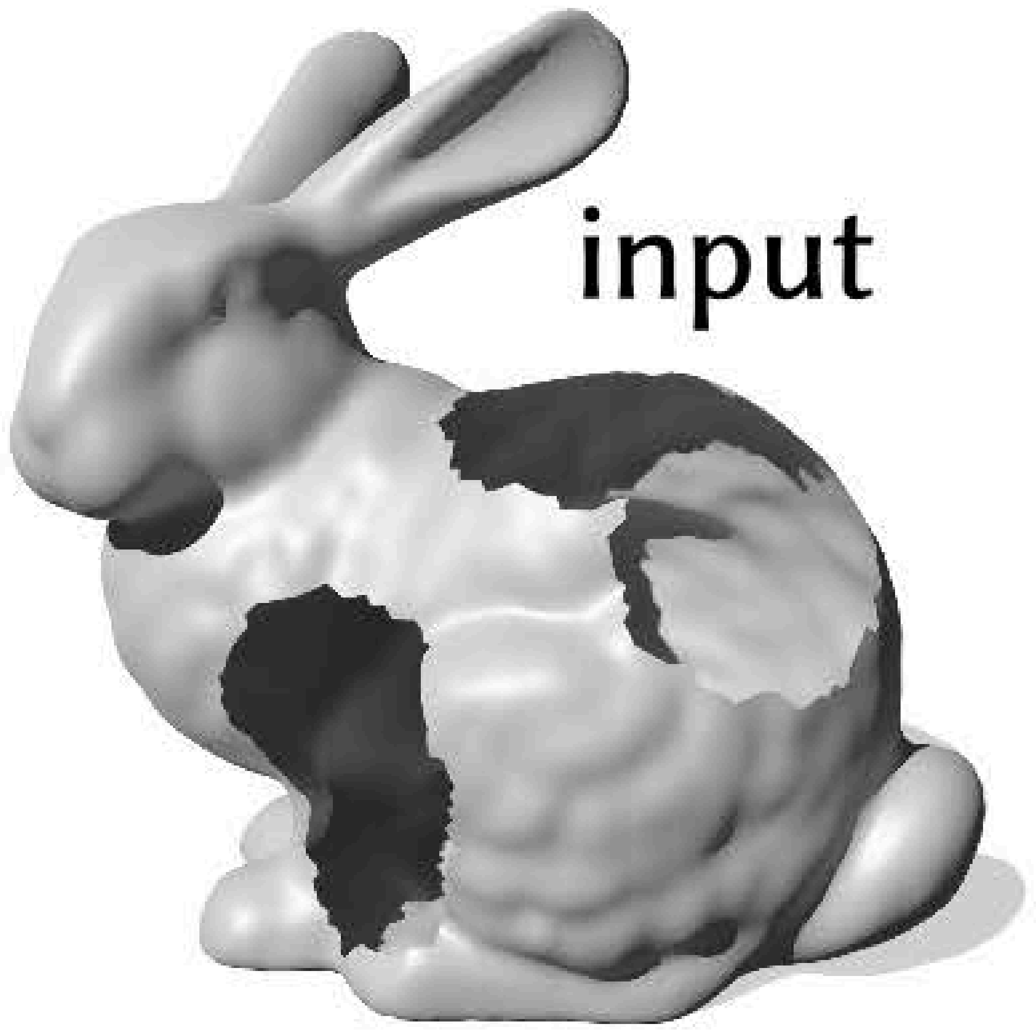
220 examples

Alternative method has ~3x more SDF error, takes ~10x longer

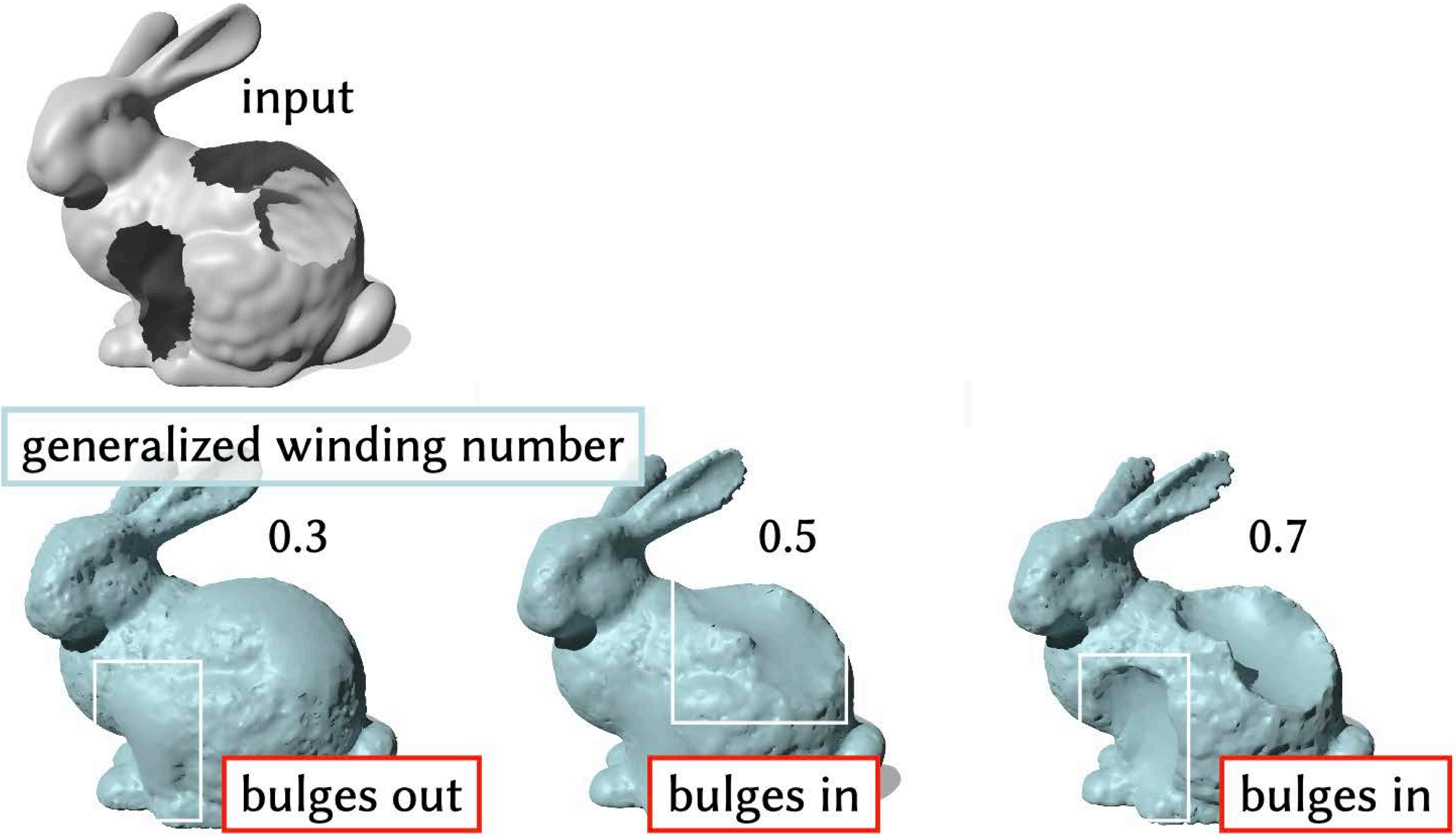


Better reconstruction than winding numbers

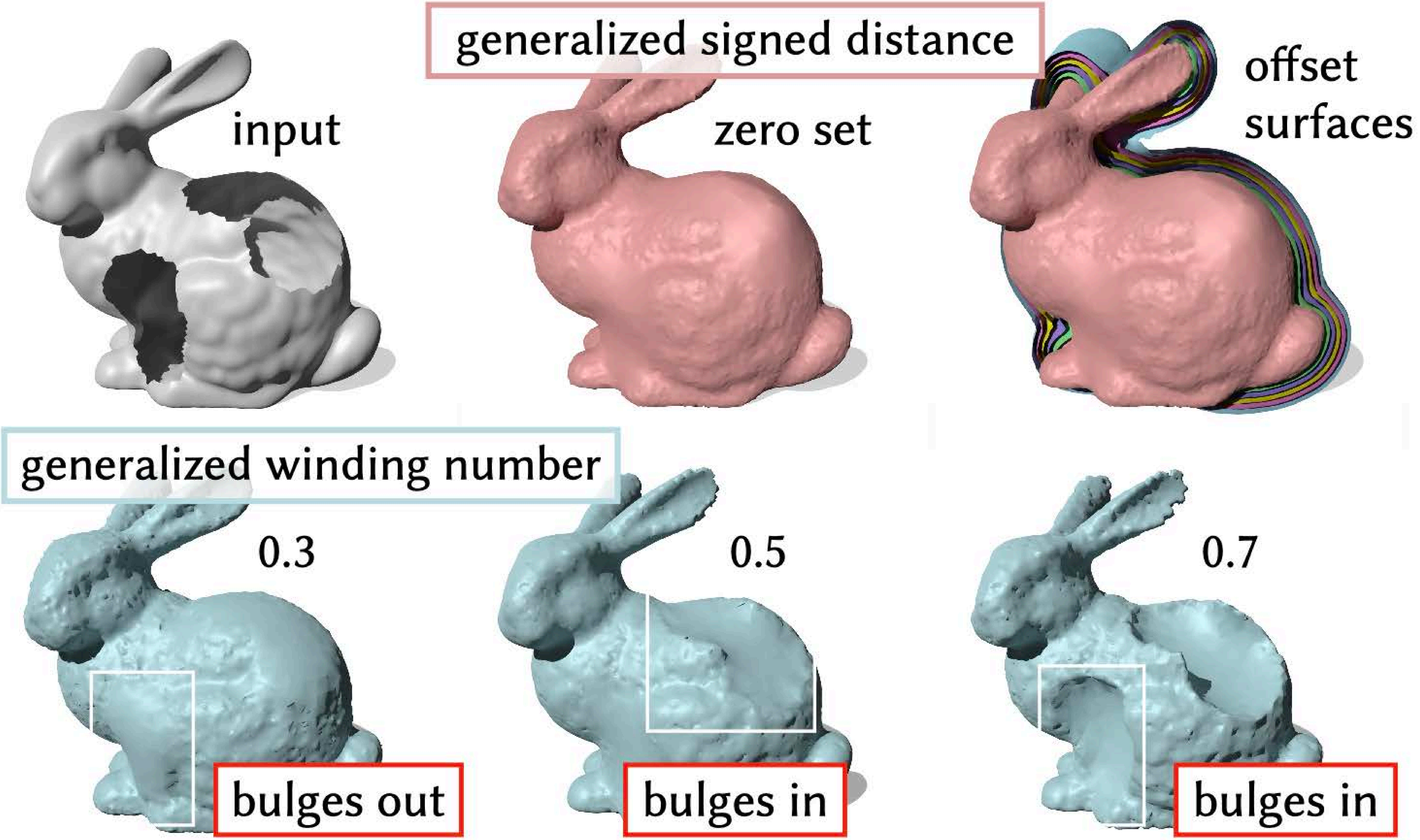
Better reconstruction than winding numbers



Better reconstruction than winding numbers



Better reconstruction than winding numbers



CONCLUSION

Takeaways

Takeaways

- Signed heat method generalizes distance in several ways
  ("broken" geometry, non-orientable source and domain geometry)

Takeaways

- Signed heat method generalizes distance in several ways
  (“broken” geometry, non-orientable source and domain geometry)
- Algorithm is incredibly simple
  (just two sparse linear systems)

Takeaways

- Signed heat method generalizes distance in several ways
  (“broken” geometry, non-orientable source and domain geometry)
- Algorithm is incredibly simple
  (just two sparse linear systems)
- Robust to errors in both the source and domain
  (holes, noise, intersections, non-manifold, inconsistent orientations)

Takeaways

- Signed heat method generalizes distance in several ways
  (“broken” geometry, non-orientable source and domain geometry)
- Algorithm is incredibly simple
  (just two sparse linear systems)
- Robust to errors in both the source and domain
  (holes, noise, intersections, non-manifold, inconsistent orientations)
- Applies to many spatial discretizations in 2D and 3D
  (triangle meshes, polygon meshes, point clouds, digital surfaces, tet meshes, regular grids, …)

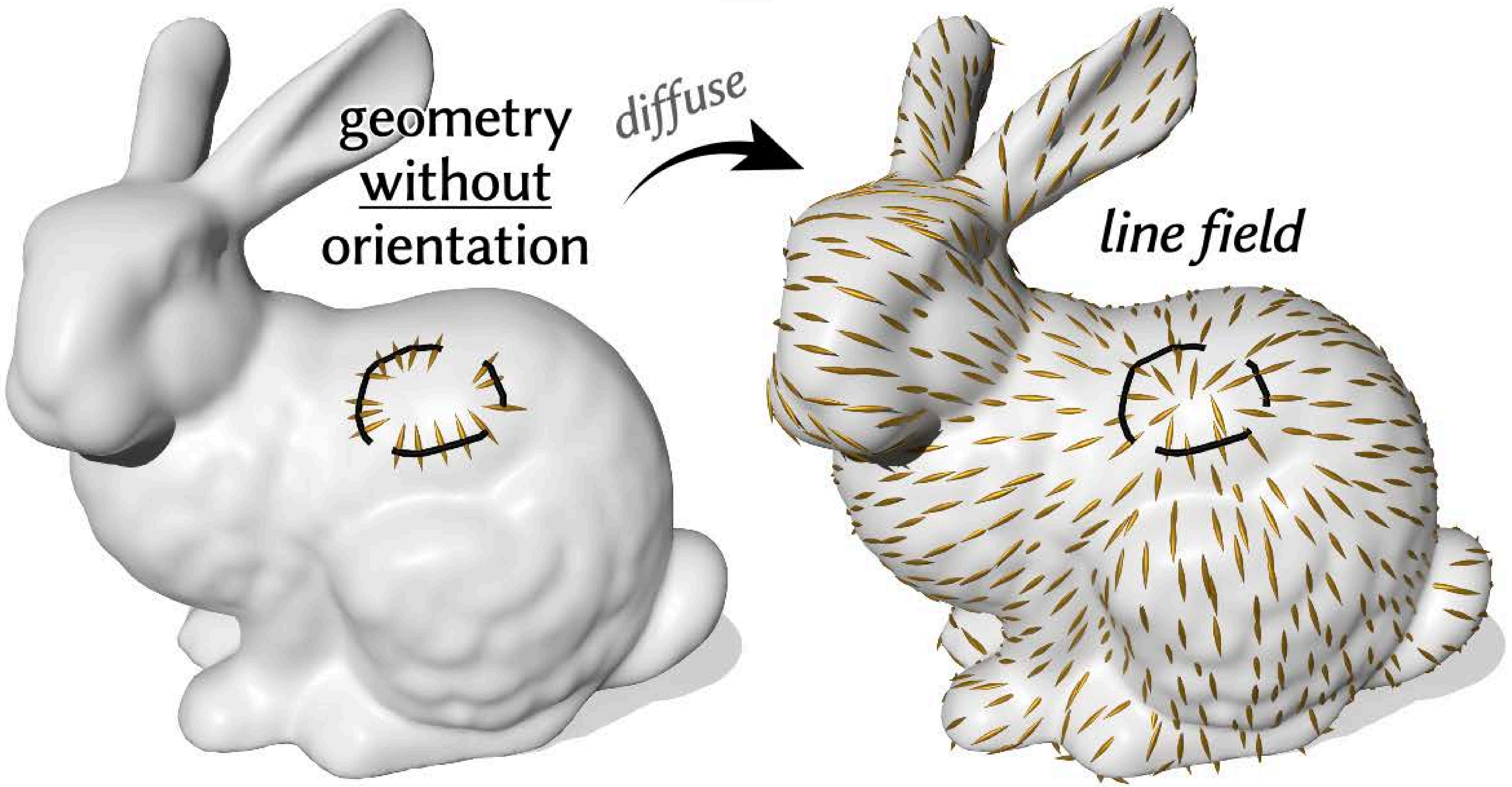
Takeaways

- Signed heat method generalizes distance in several ways
  (“broken” geometry, non-orientable source and domain geometry)
- Algorithm is incredibly simple
  (just two sparse linear systems)
- Robust to errors in both the source and domain
  (holes, noise, intersections, non-manifold, inconsistent orientations)
- Applies to many spatial discretizations in 2D and 3D
  (triangle meshes, polygon meshes, point clouds, digital surfaces, tet meshes, regular grids, ...)
- Good at surface reconstruction

Fun future direction: line field diffusion

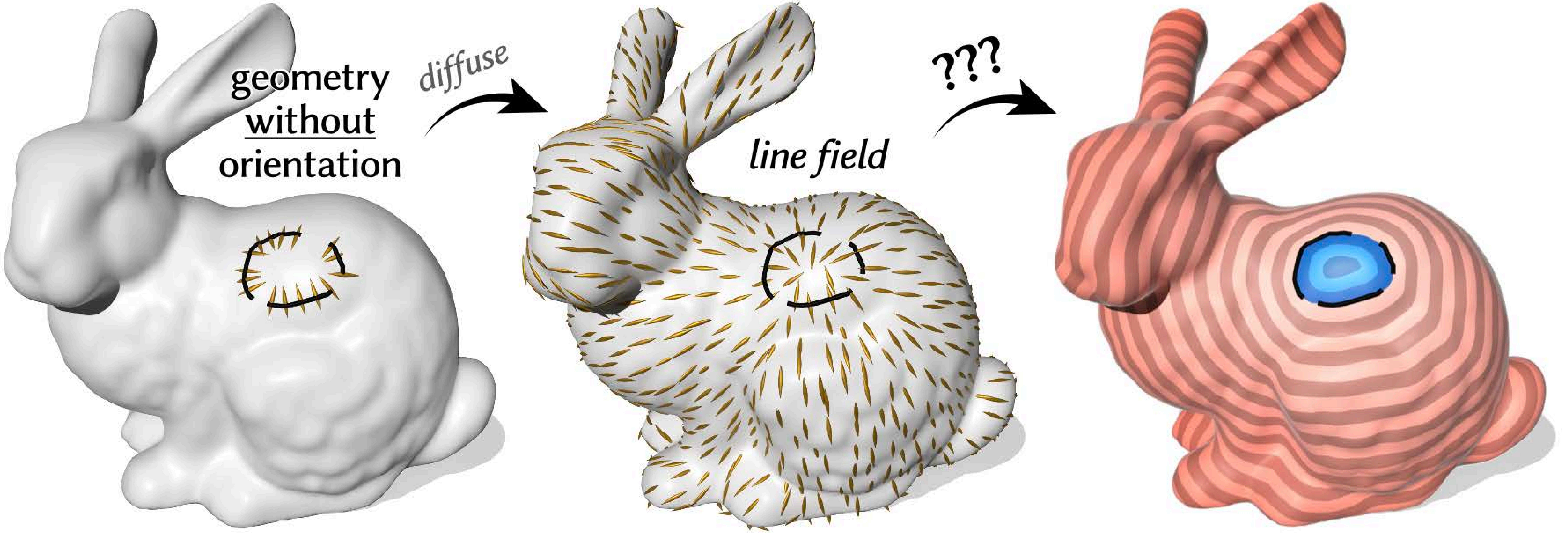
Fun future direction: line field diffusion

Can diffuse line fields, cross fields, …



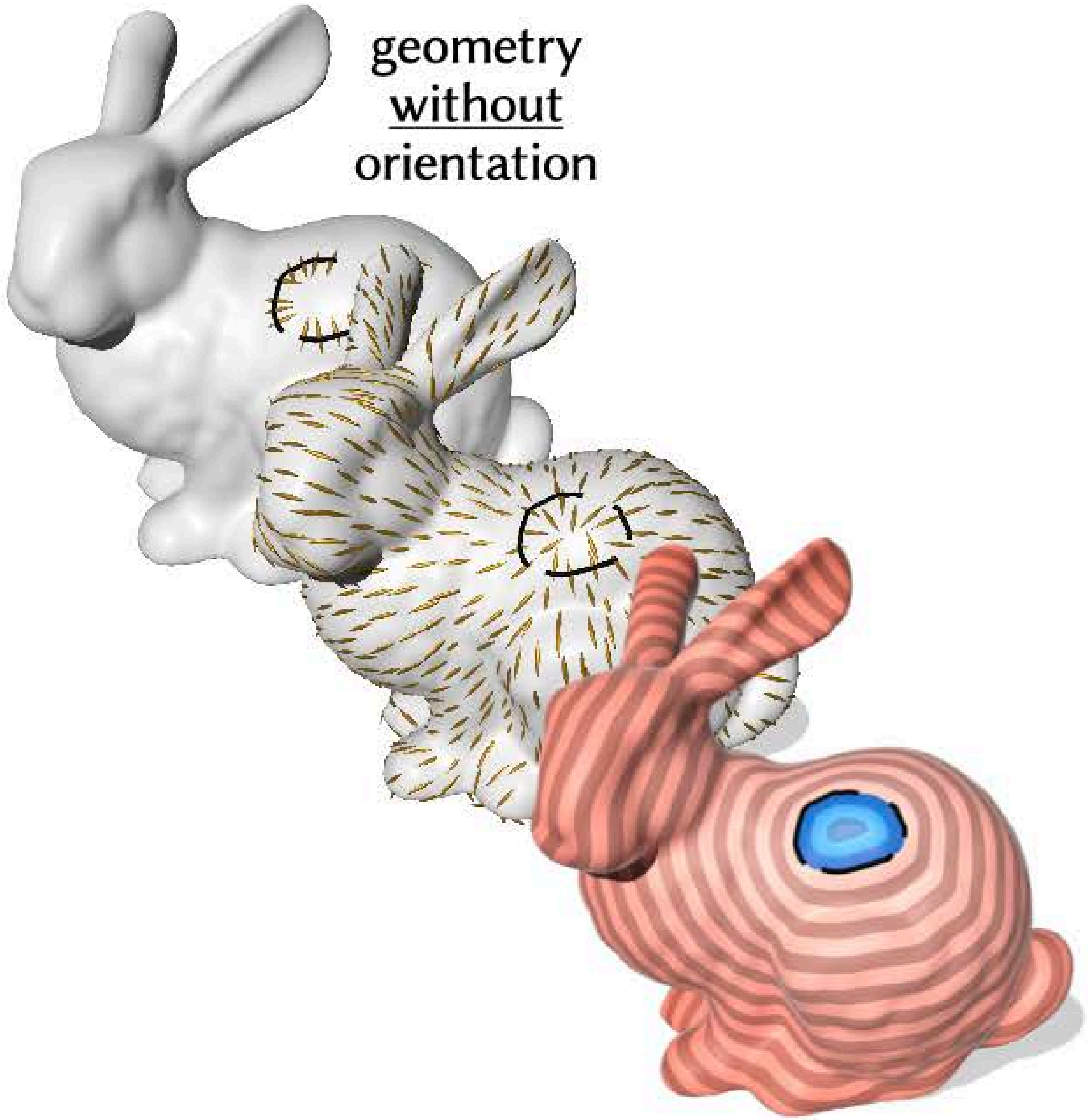
Fun future direction: line field diffusion

Can diffuse line fields, cross fields, … … recover distance *and* orientation?



Fun future direction: line field diffusion

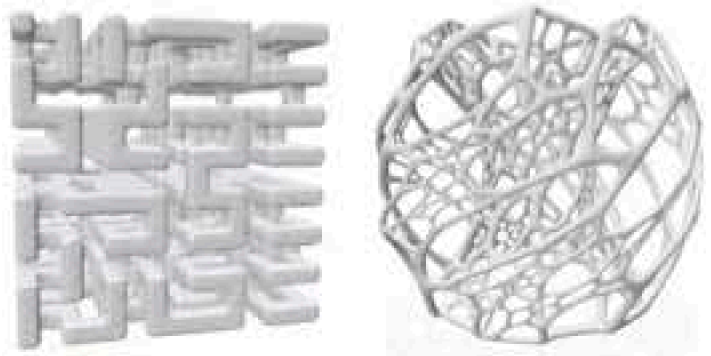
Can diffuse line fields, cross fields, …



… recover distance *and* orientation?

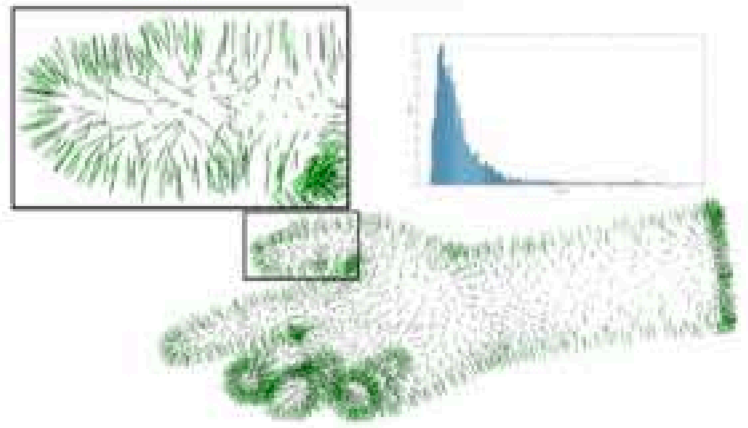
Recovers orientation:

[Liu et al. 2024]



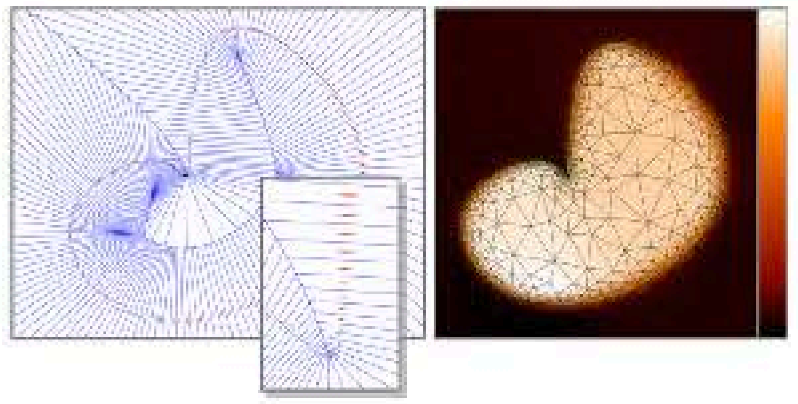
“Consistent Point Orientation for Manifold Surfaces via Boundary Integration”

[Gotsman & Hormann 2024]



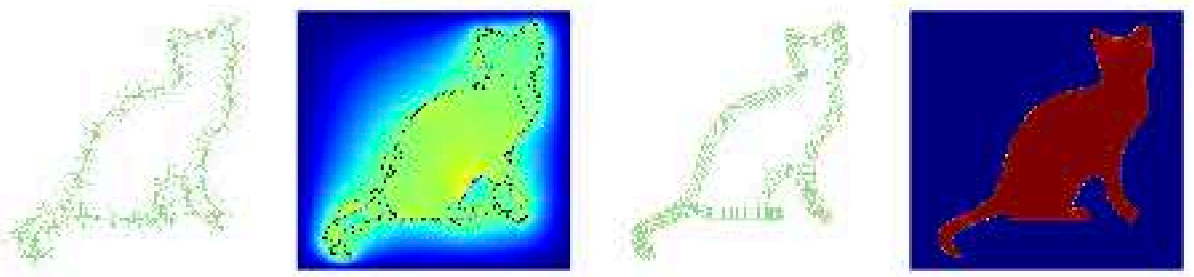
“A Linear Method to Consistently Orient Normals of a 3D Point Cloud”

[Alliez et al. 2007]



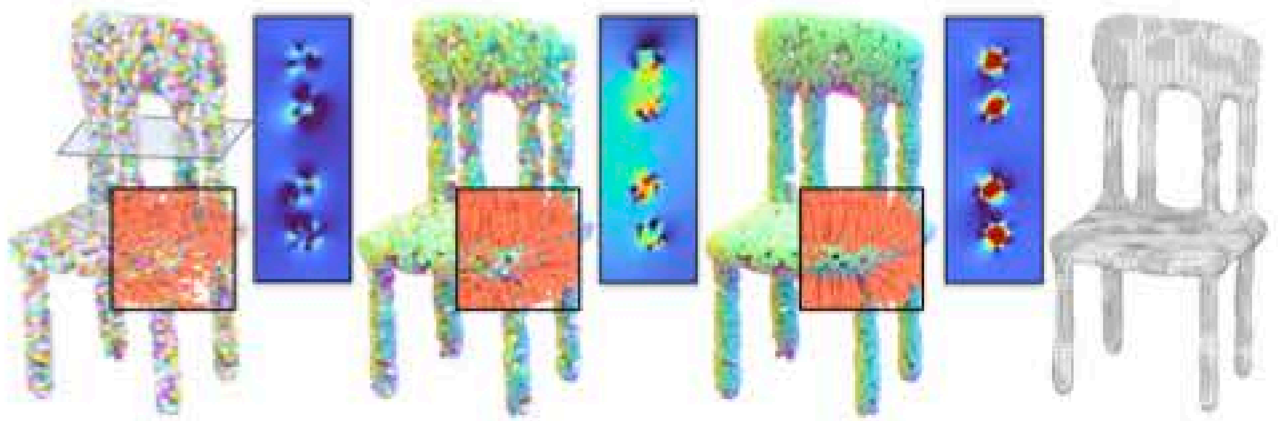
“Voronoi-based Variational Reconstruction of Unoriented Point Sets”

[Hou et al. 2022]



“Iterative poisson surface reconstruction (iPSR) for unoriented points”

[Xu et al. 2023]



“Globally Consistent Normal Orientation for Point Clouds by Regularizing the Winding-Number Field”

Thanks!