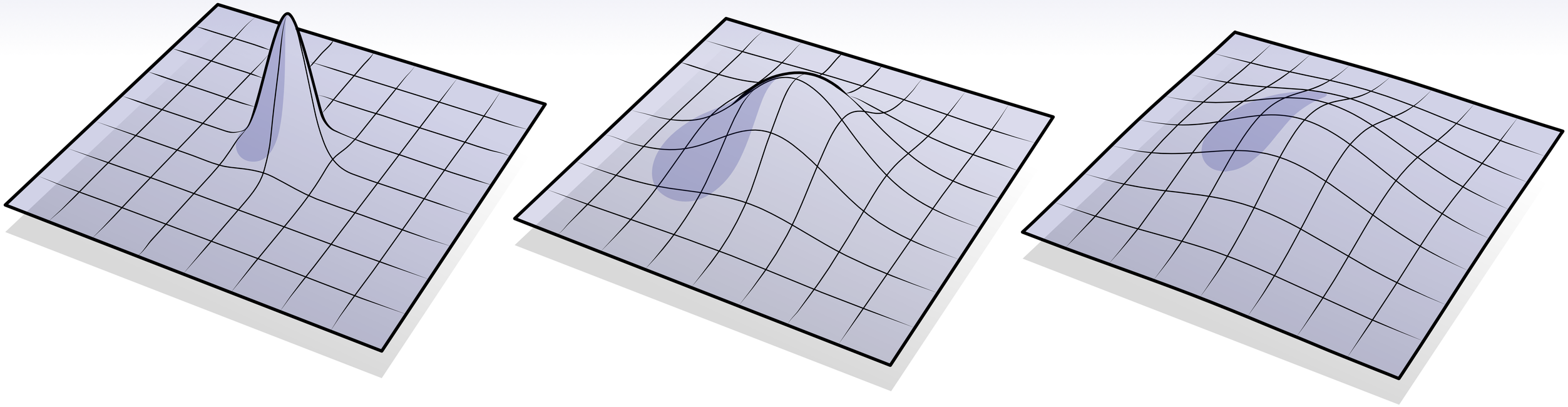
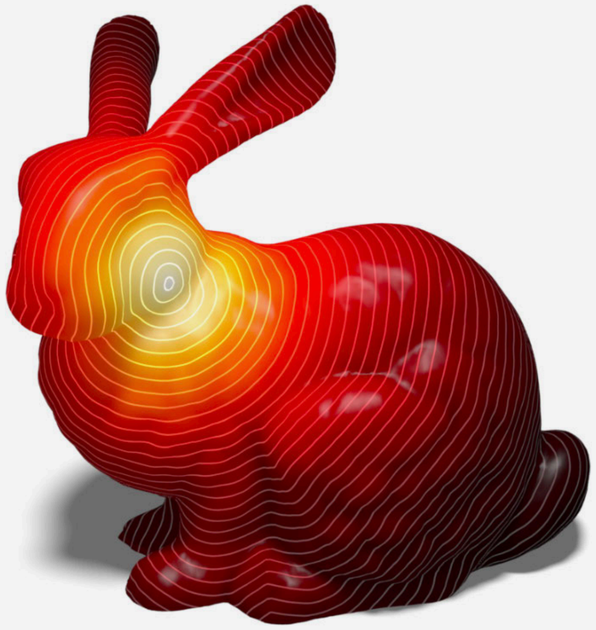


Heat Methods in Geometry Processing

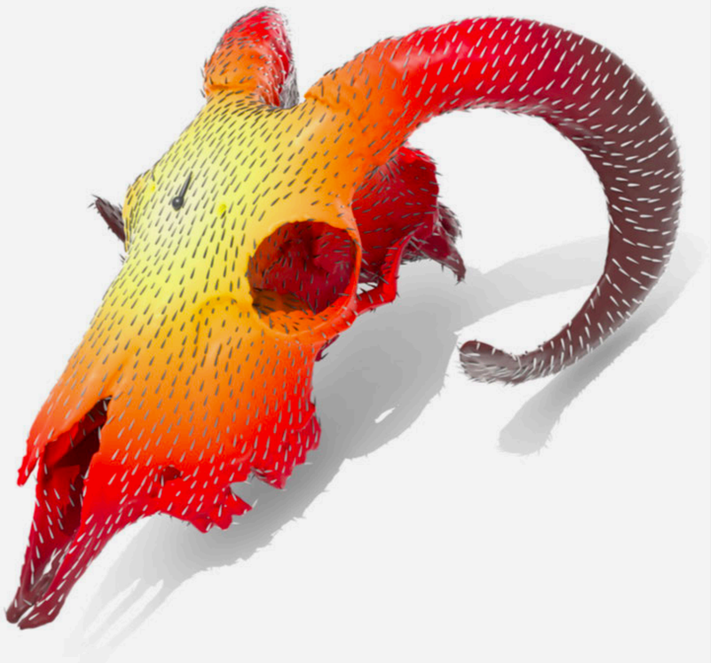
Keenan Crane • IPAM Workshop on Geometric Processing • April 4, 2019

Heat Methods—Overview

Central principle: use *short-time diffusion* as a building block for geometric computation.



The Heat Method for Distance Computation
Crane, Weischedel, Wardetzky
Communications of the ACM (2017)

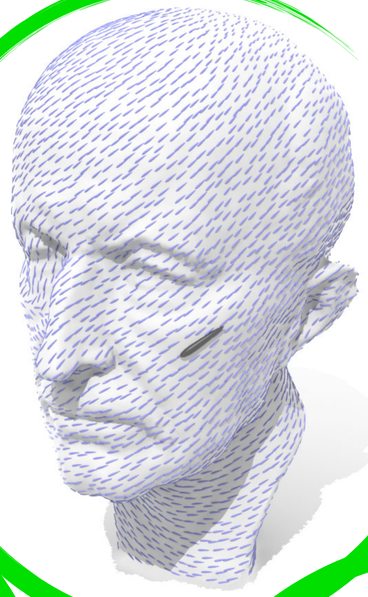


The Vector Heat Method
Sharp, Soliman, Crane
ACM Transactions on Graphics (2019)

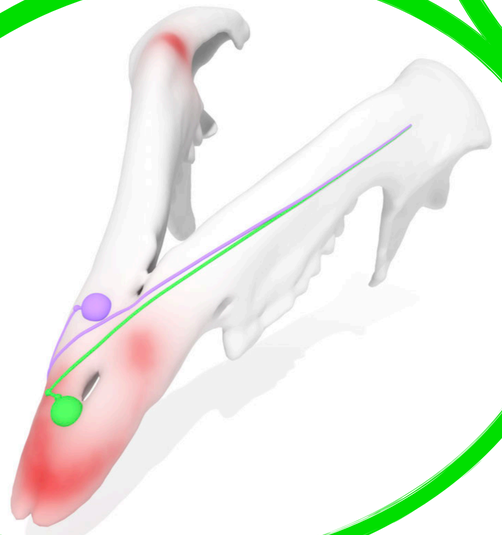
Heat Methods—Atomic Geometric Operations

geodesic distance

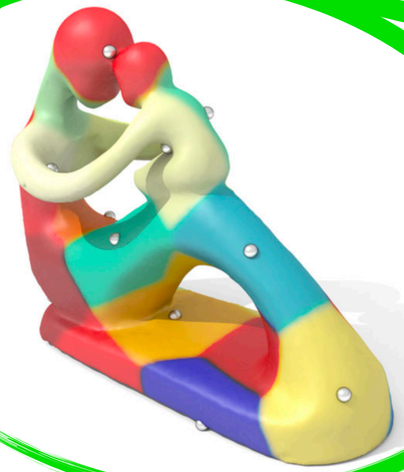
parallel transport



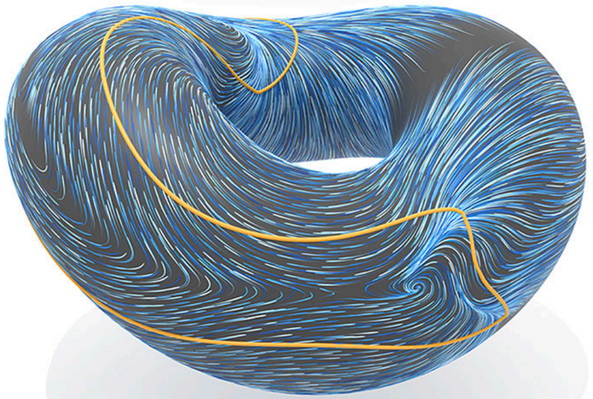
logarithmic map



Karcher means/geometric medians

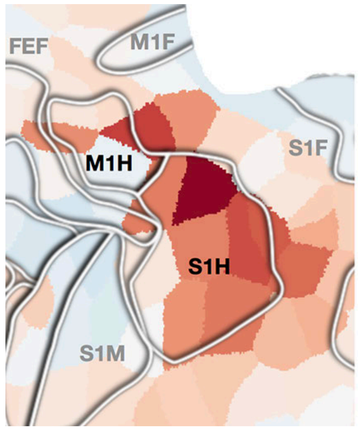


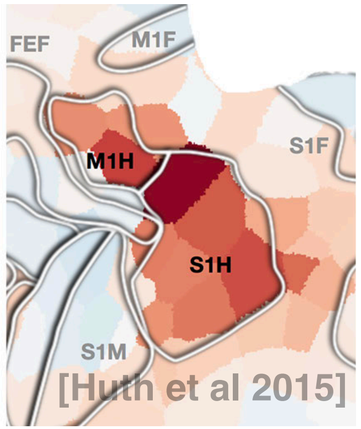
centroidal Voronoi diagrams



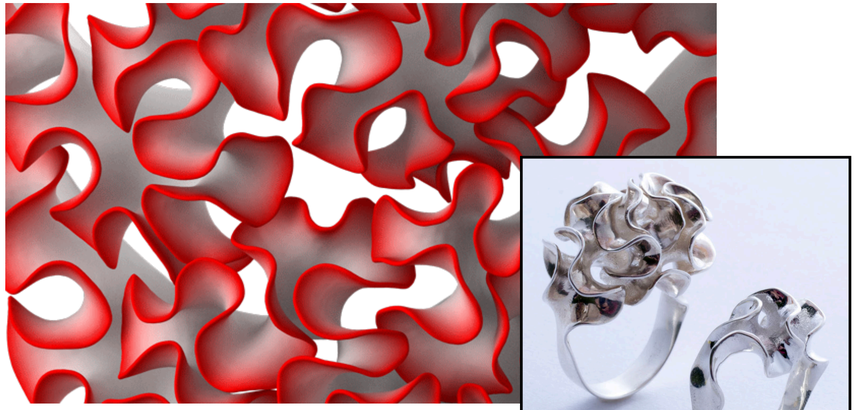
velocity extrapolation

Heat Methods—Applications

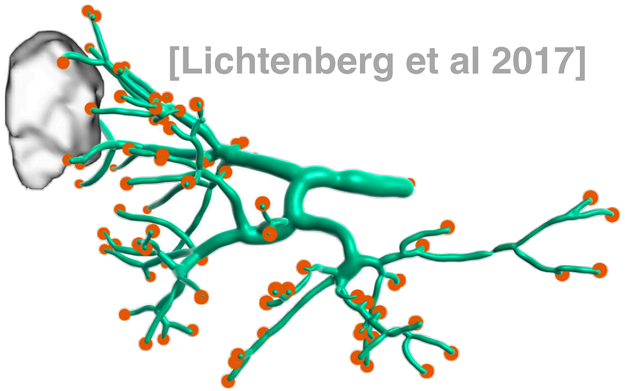




[Huth et al 2015]

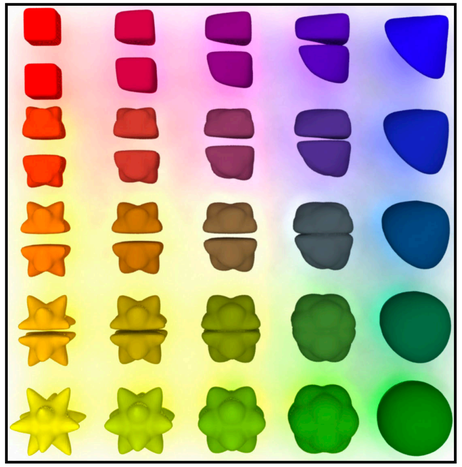


[Louis-Rosenberg 2015]



[Lichtenberg et al 2017]

[Mohamed et al 2016]



[Solomon et al 2015]

[Narayanan et al 2018]

Heat Methods—Overview

- Why is principle of *short-time diffusion* useful for computation?
- [GENERALITY] Well defined on any (discrete) manifold with

  (i) an inner product / mass matrix

  (ii) a gradient operator (or covariant derivative)

  (*E.g.*, polygon meshes, point clouds, voxel grids, …)

$$\Delta = \nabla^* \nabla$$

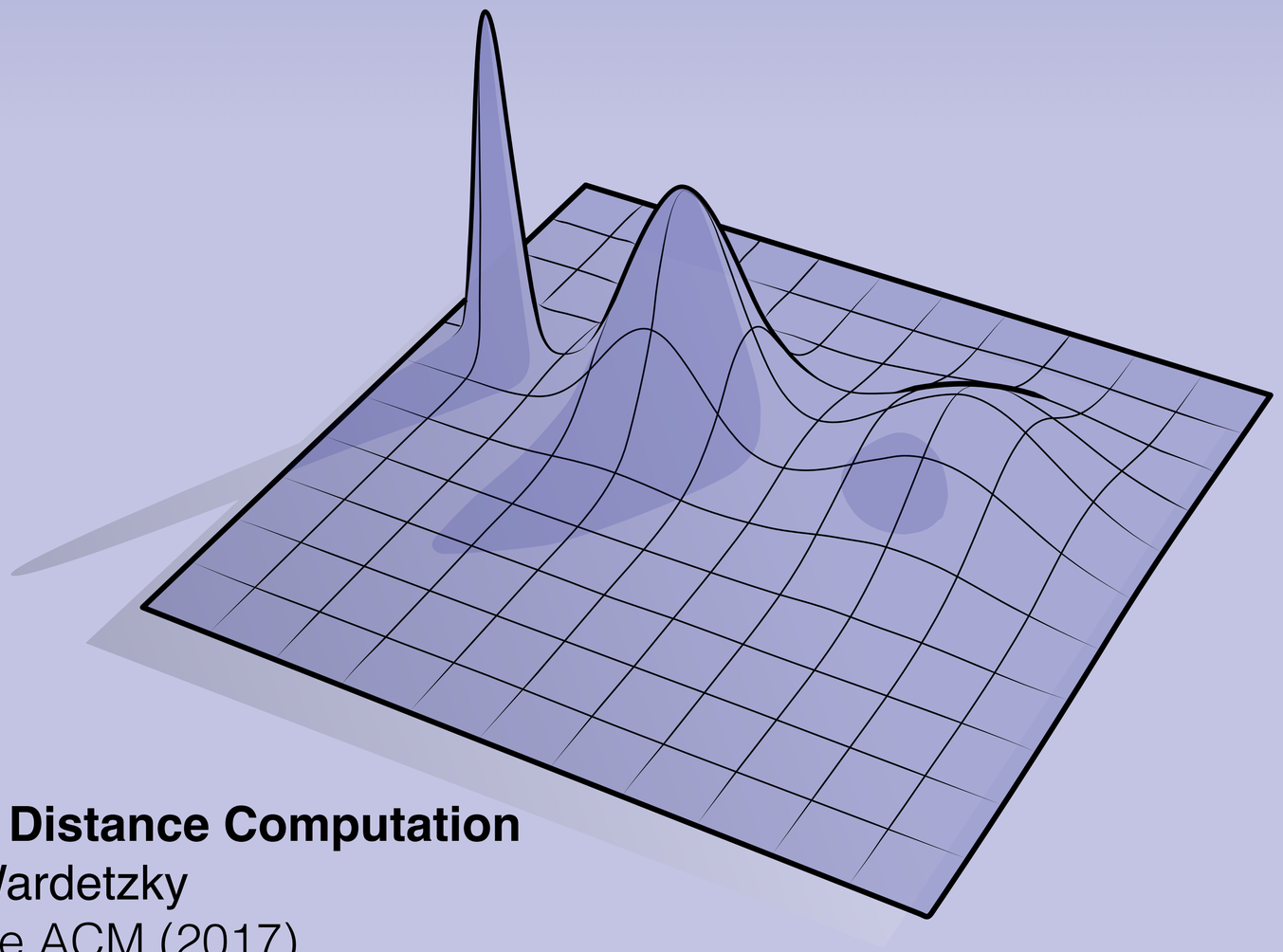
- [EFFICIENCY] Amounts to standard linear PDEs:
  - heat equation
  - Poisson equation

  (Easy to parallelize, prefactor, …)

$$\frac{d}{dt} u = \Delta u$$
$$\Delta u = f$$

- **Easy to implement** ⇒ broad adoption in real applications

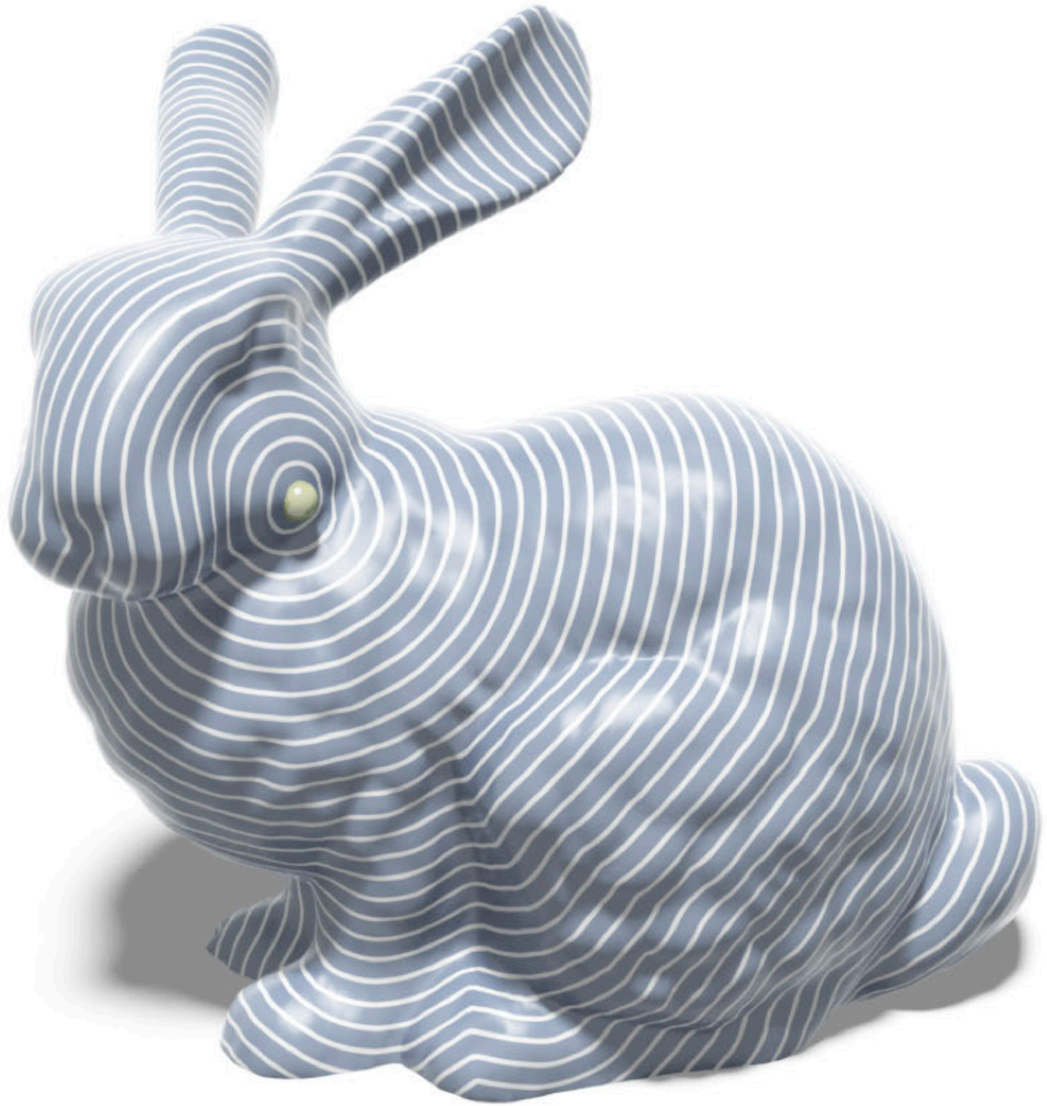
Part I:
The Heat Method



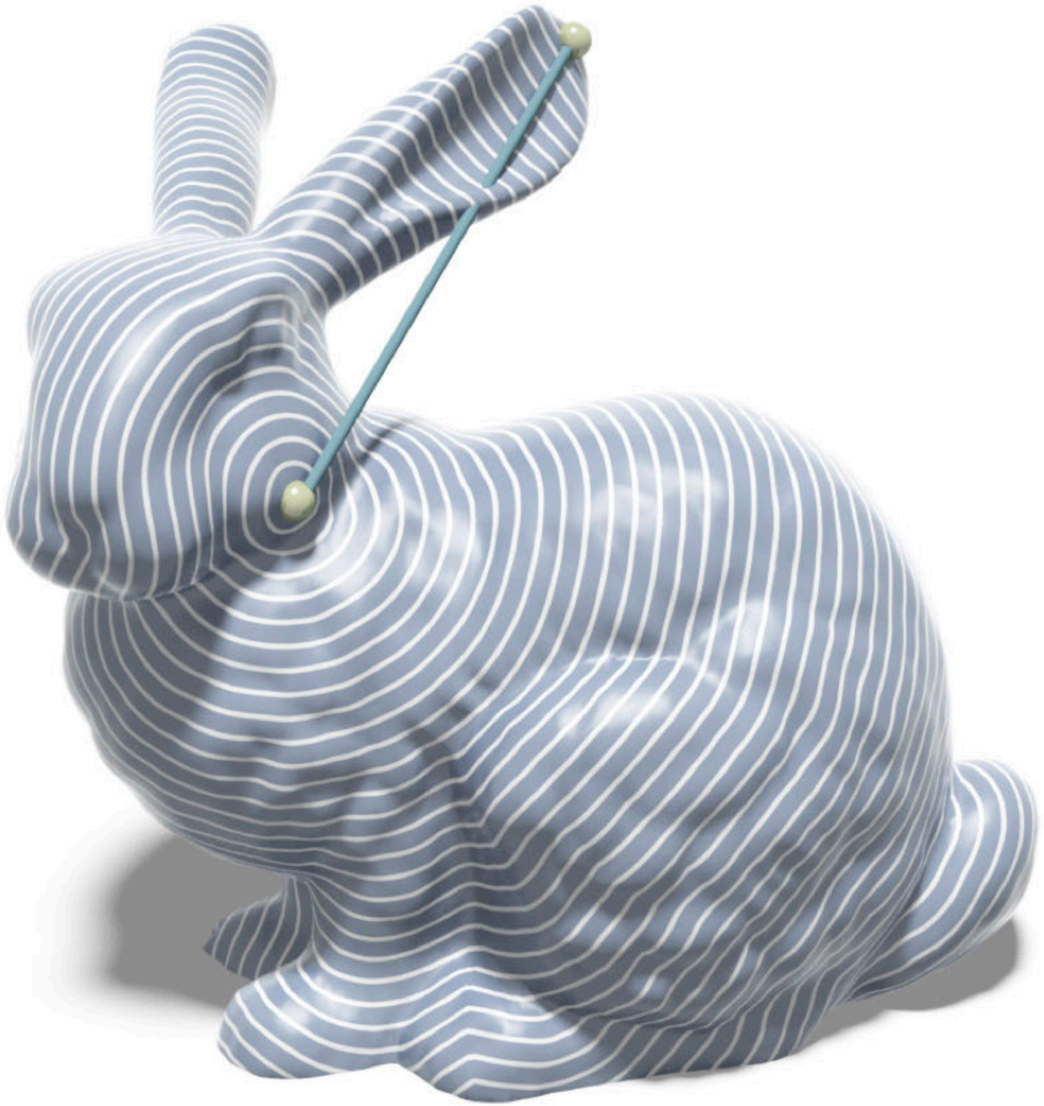
The Heat Method for Distance Computation
Crane, Weischedel, Wardetzky
Communications of the ACM (2017)

Motivating Problem—Geodesic Distance

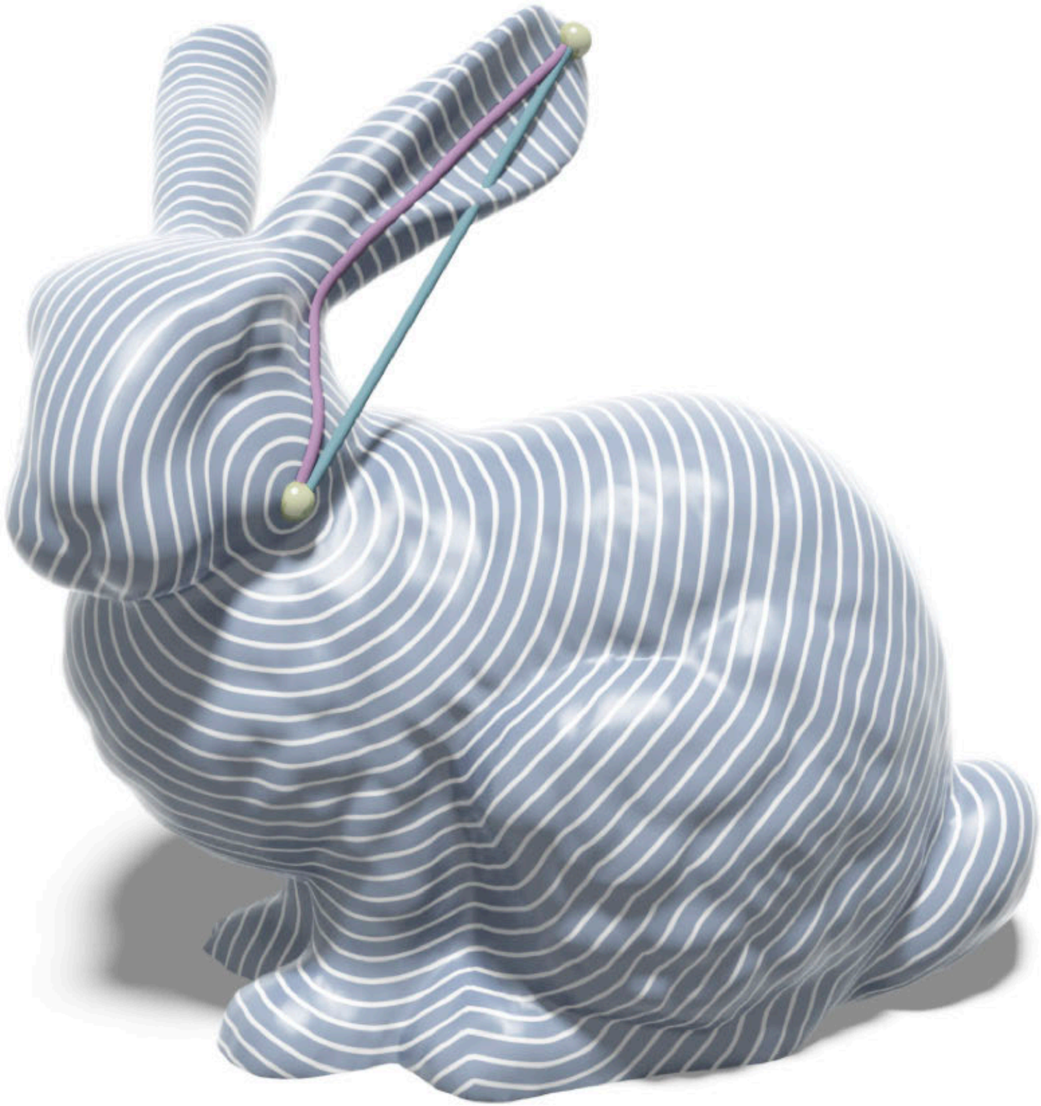
Motivating Problem—Geodesic Distance



Motivating Problem—Geodesic Distance



Motivating Problem—Geodesic Distance

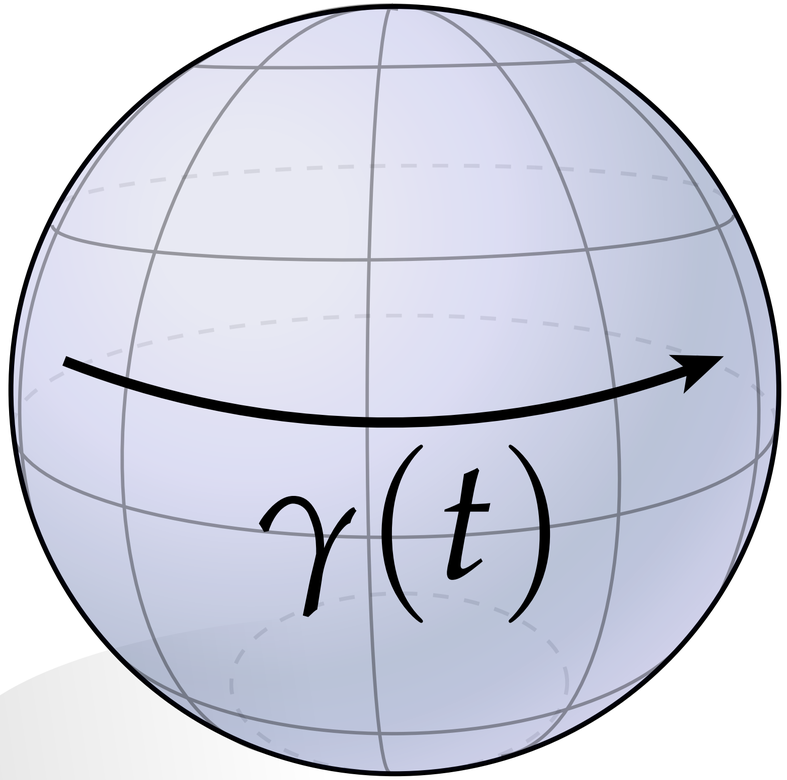


Geodesic Equation

tangent to curve

$$\nabla_{\dot\gamma}\dot\gamma = 0$$

"tangent doesn't turn"



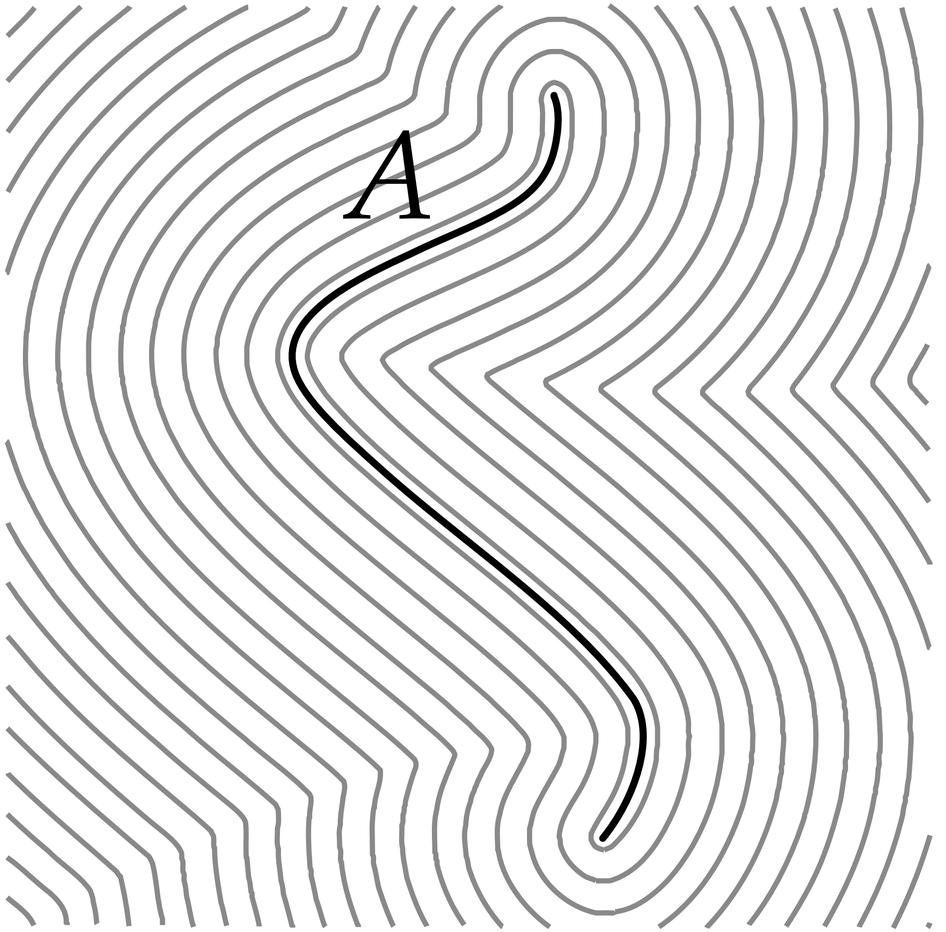
- nonlinear ODE
- track "windows" of geodesic paths
- $O(n^2 \log n)$; $O(n^2)$ [Mitchell et al 1987; Chen-Han 1990]

Eikonal Equation

distance to source

$$|\nabla \phi| = 1$$

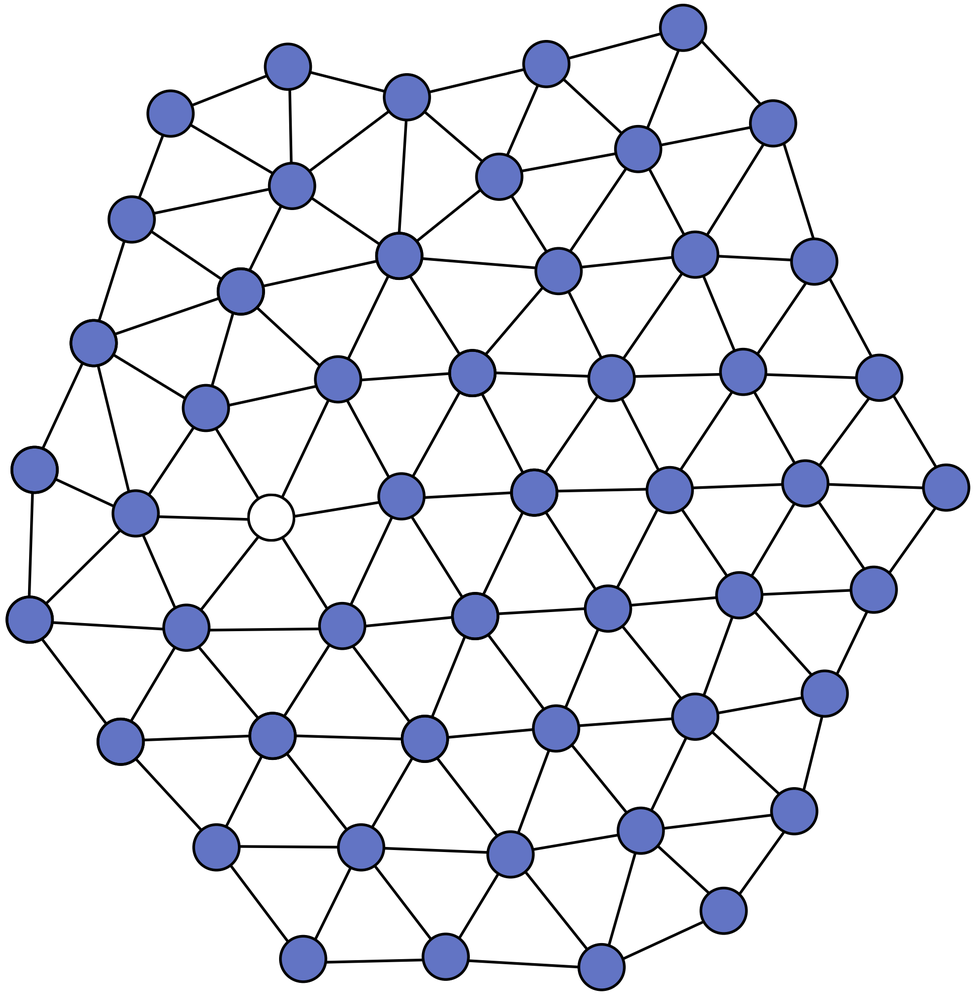
"distance changes at one meter per meter"

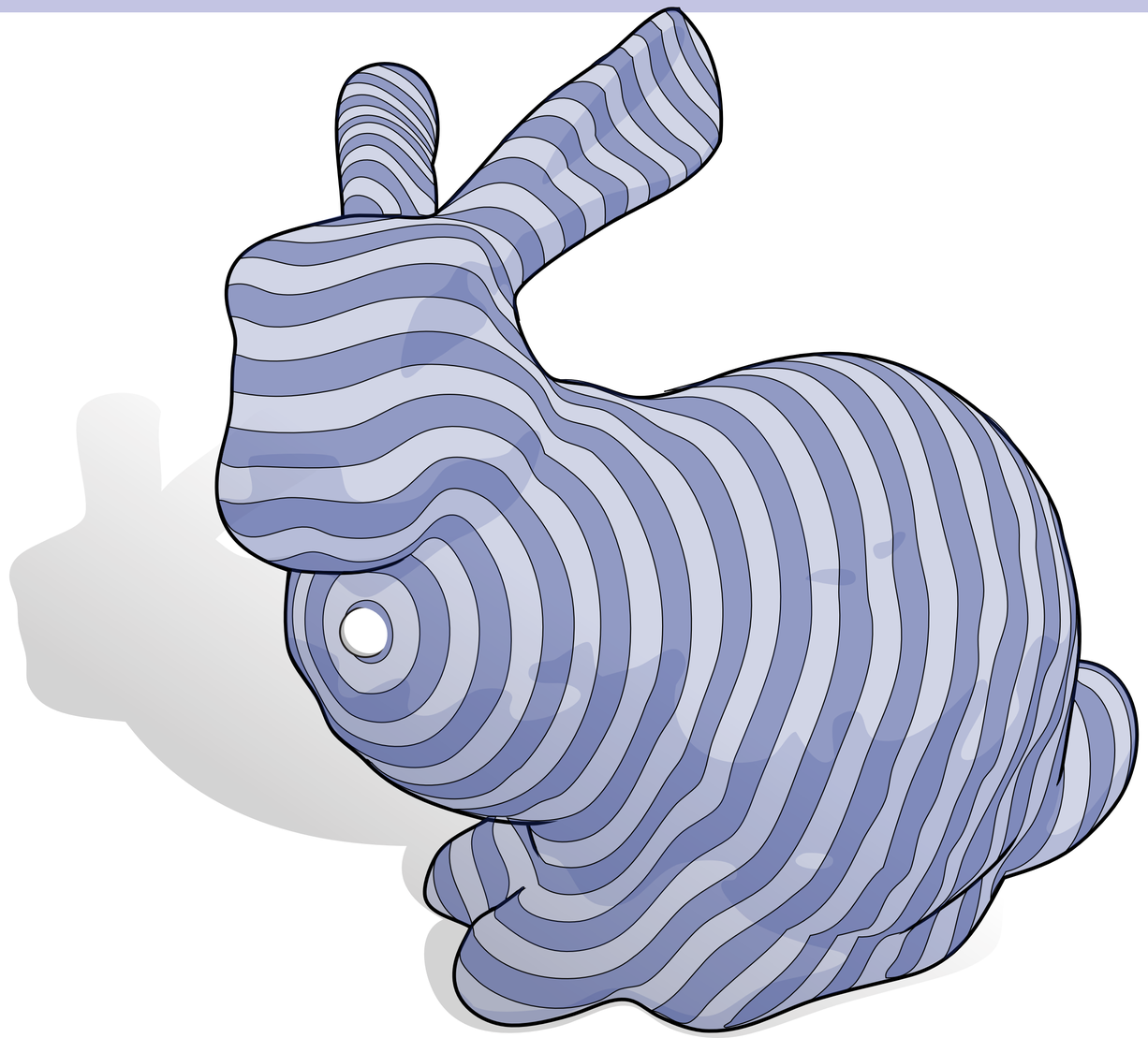


$$\phi|_A = 0$$

- nonlinear, hyperbolic PDE
- solve by propagating outward from source
- fast marching method [Kimmel & Sethian 1998]

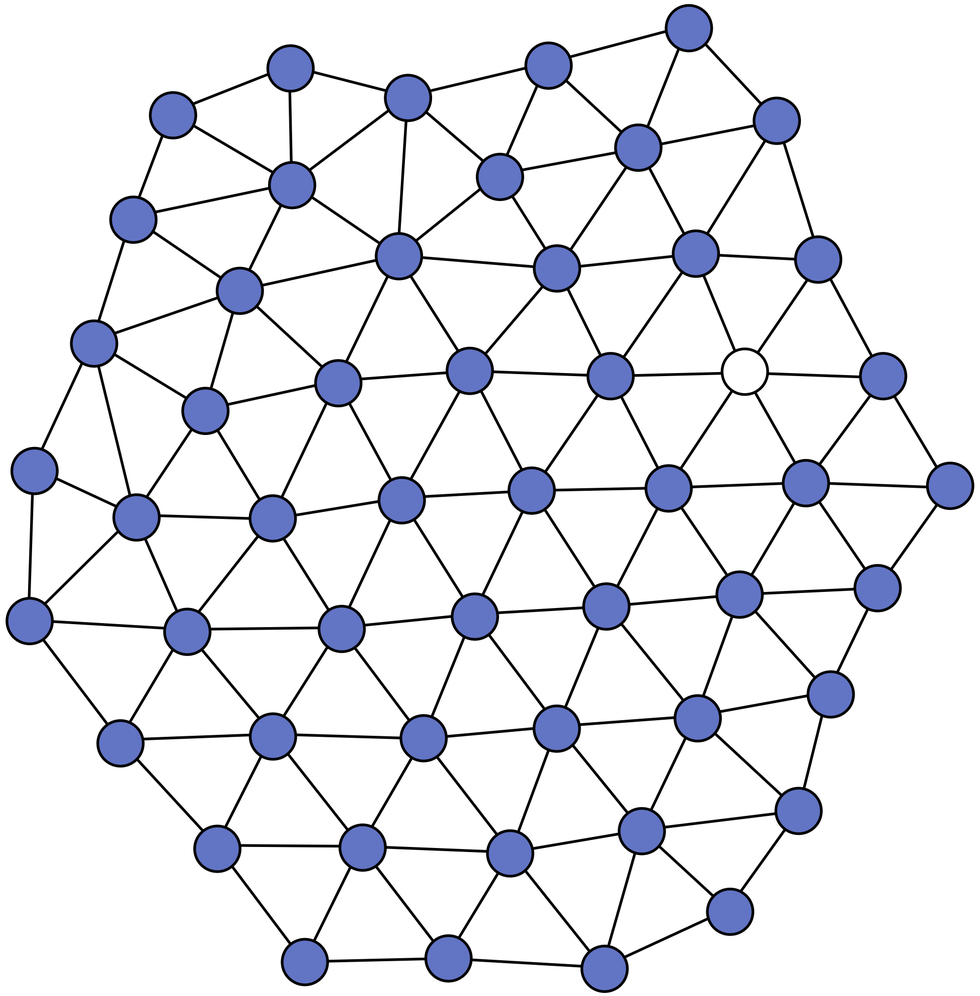
Challenges—Reusing Computation

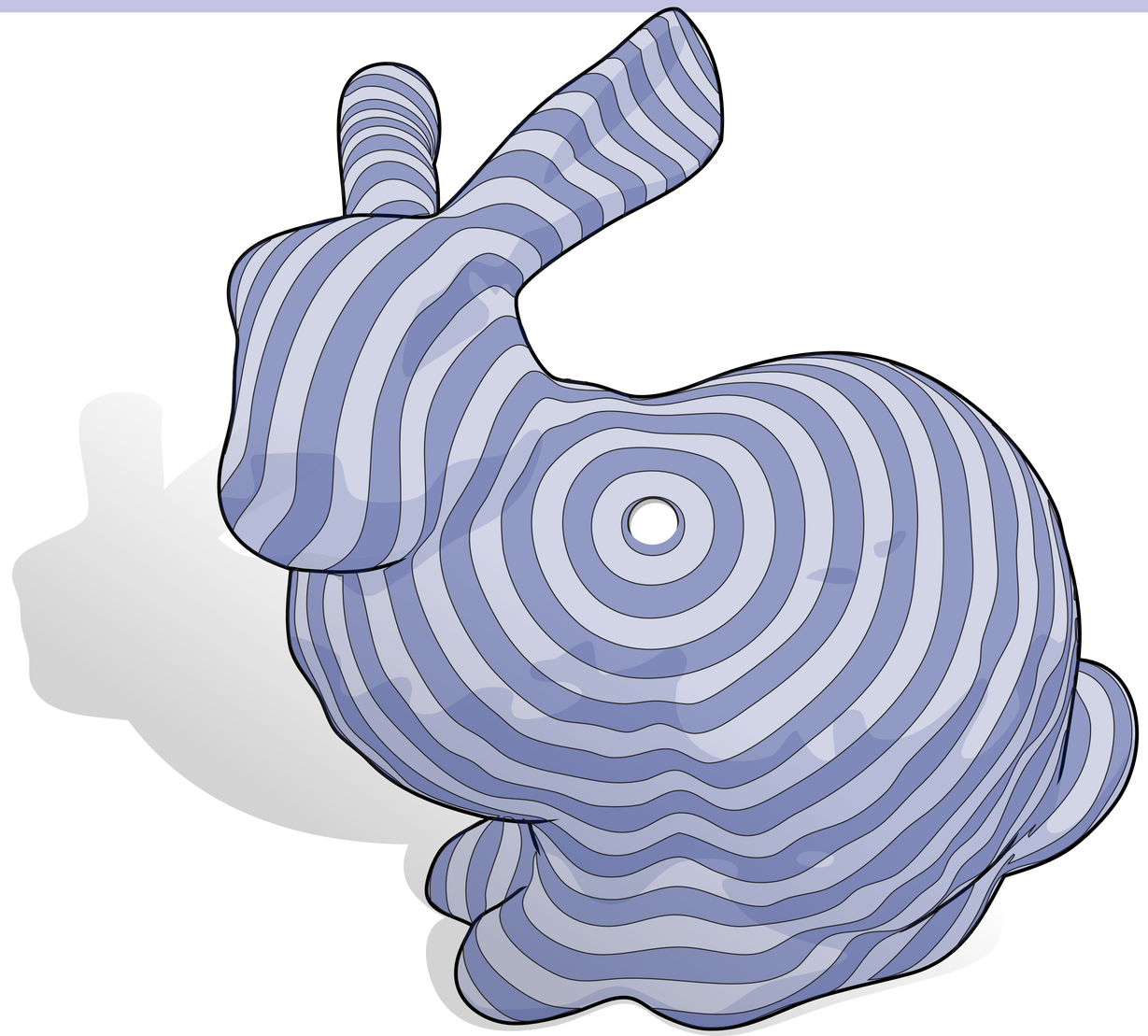




[Dijkstra 1959, Mitchell et al 1987, Chen & Han 1990, Sethian & Kimmel 1998, Surazhsky et al 2005...]

Challenges—Reusing Computation

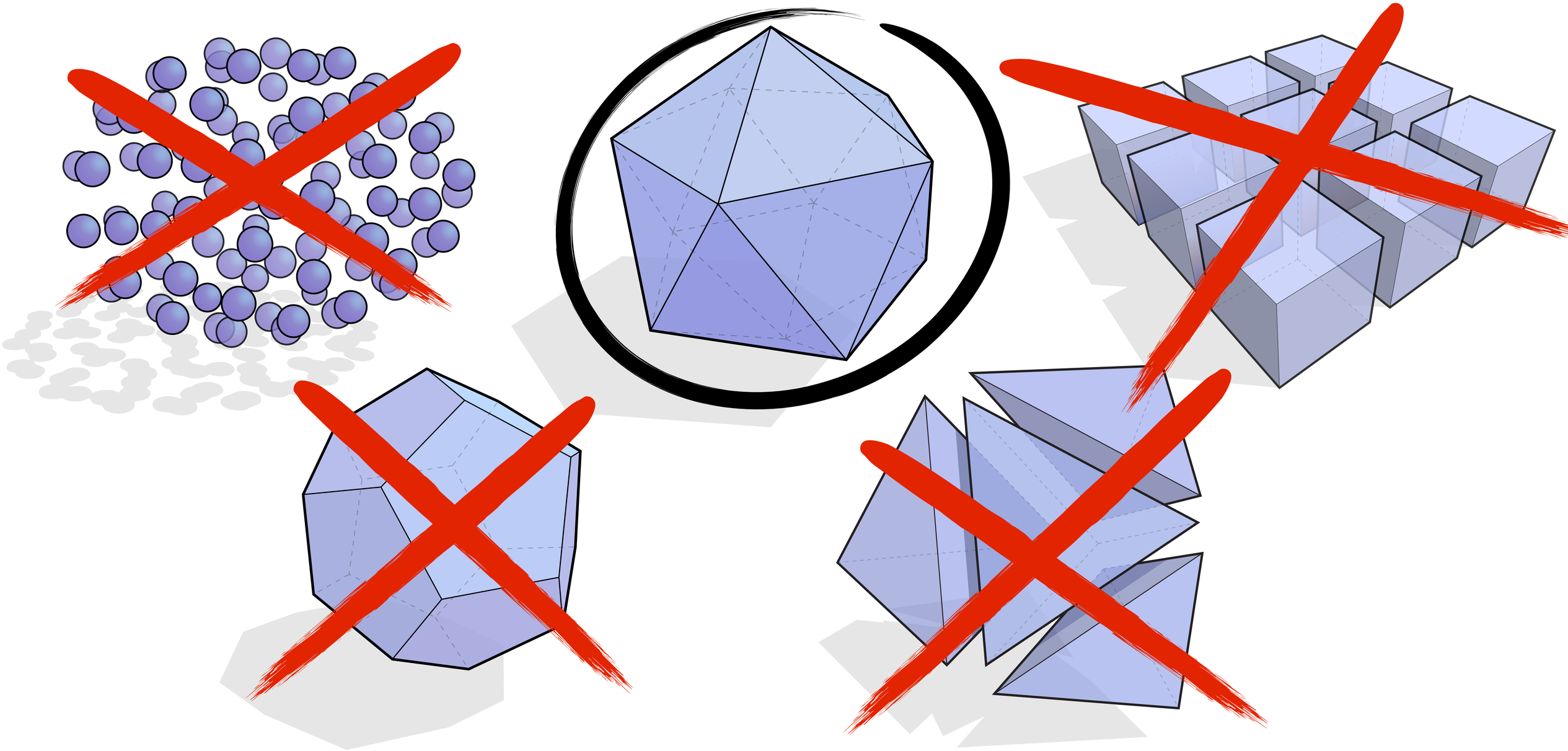




[Dijkstra 1959, Mitchell et al 1987, Chen & Han 1990, Sethian & Kimmel 1998, Surazhsky et al 2005...]

Challenges—Parallelism

Challenges—Discretization



Diffusion Equation

distribution of heat

$$\frac{d}{dt}u = \Delta u$$

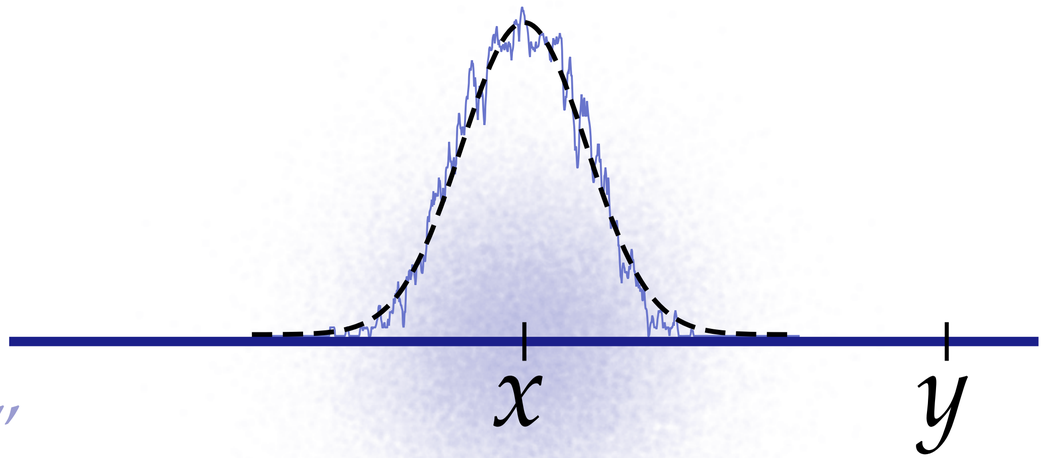
"smooth out bumps"

- linear, parabolic
- solve by repeated local averaging
- numerical strategies studied since *at least* Courant 1928…

Heat Kernel

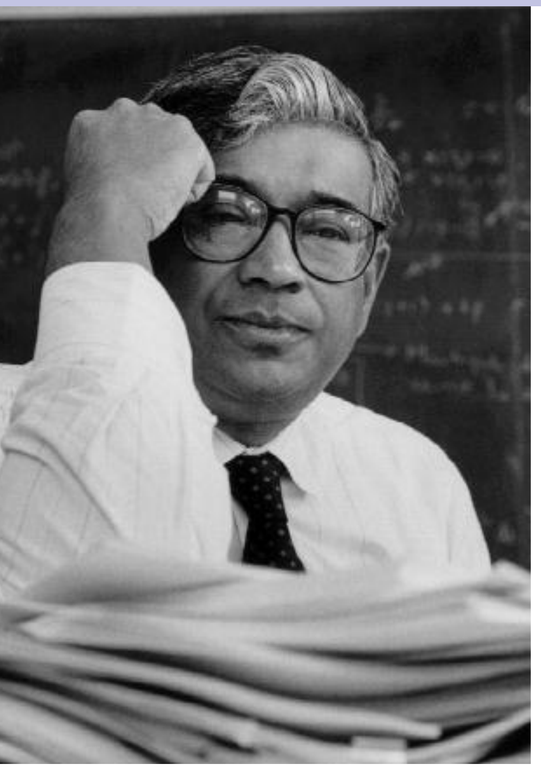
$$k_t(x,y)$$

"heat diffused from x to y after time t"



$$\begin{aligned} \frac{d}{dt}u &= \Delta u \\ u(0) &= \delta_x \end{aligned}$$

Varadhan's Formula

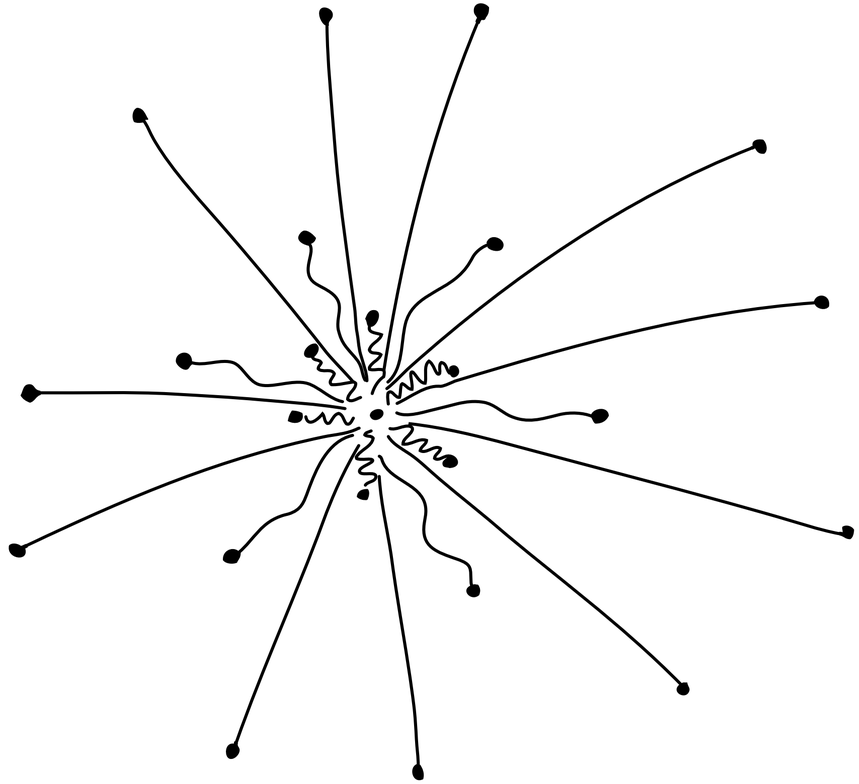


[Varadhan 1967]

$$\phi(x,y) = \lim_{t \to 0} \sqrt{-4t \log k_t(x,y)}$$

distance

heat kernel

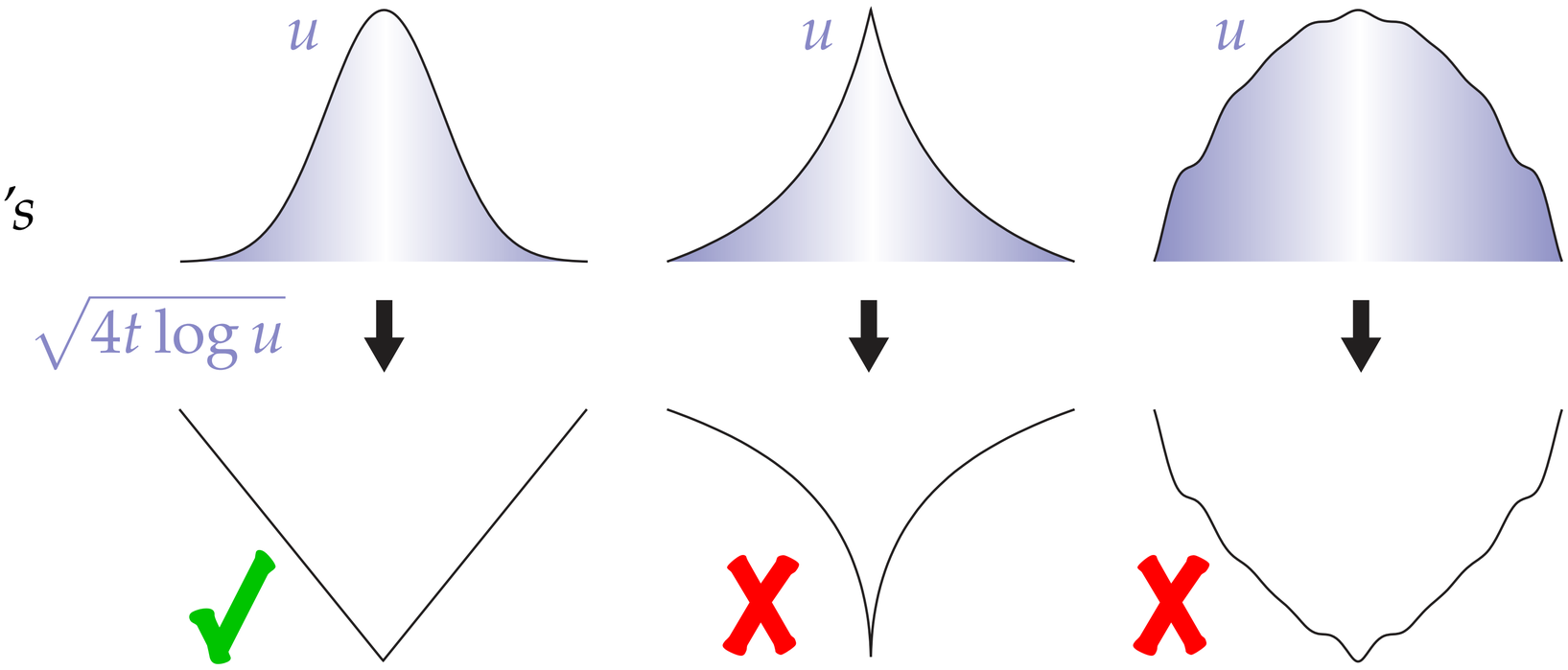


Varadhan's Formula—Approximation Error

Varadhan's formula

geodesic distance

approximate heat kernel

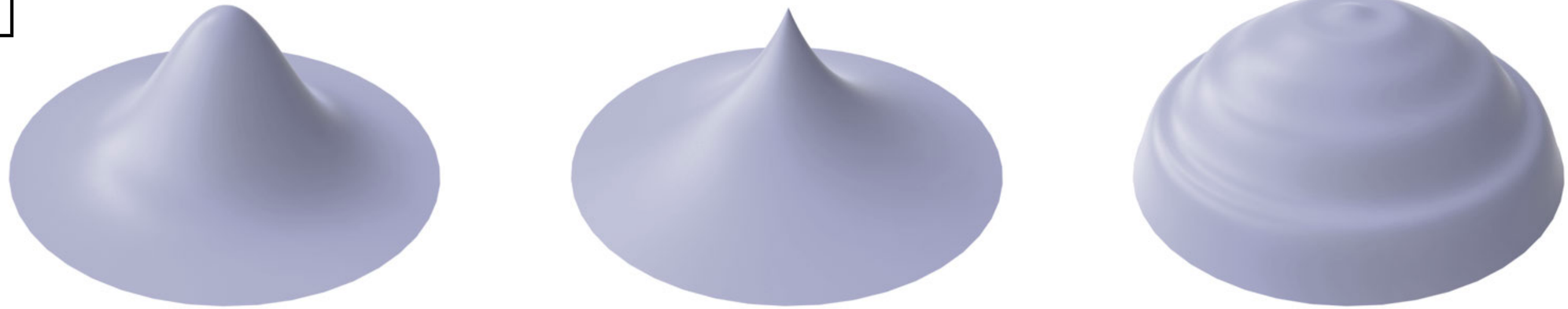


distance approximation

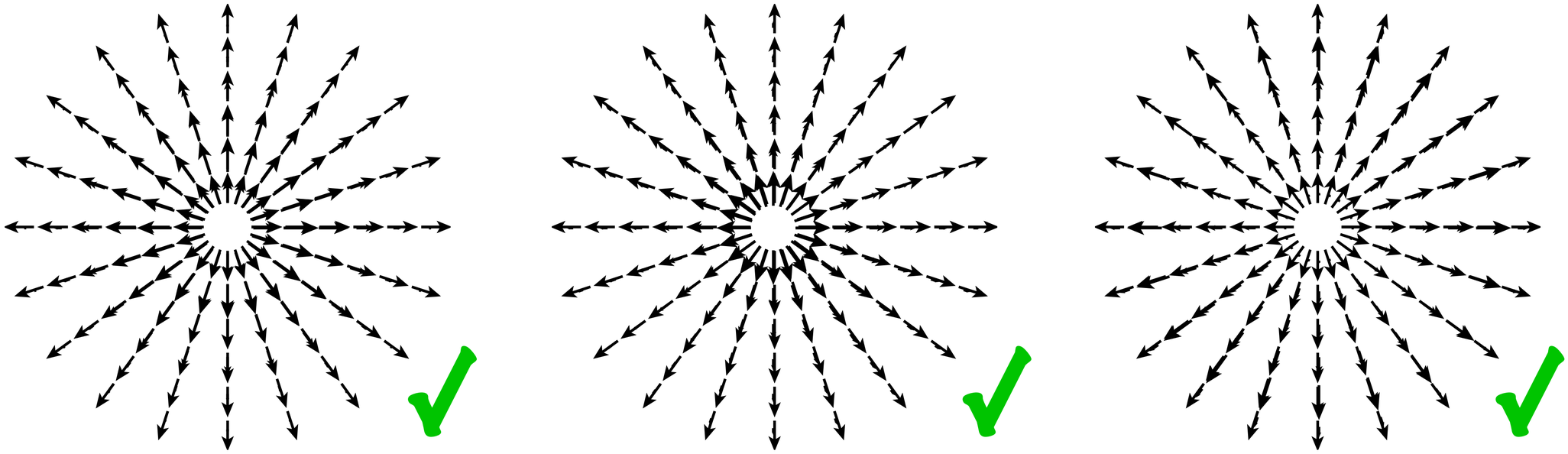
Gradient Normalization

$$|\nabla \phi| = 1$$

u



$$\frac{-\nabla u}{|\nabla u|}$$



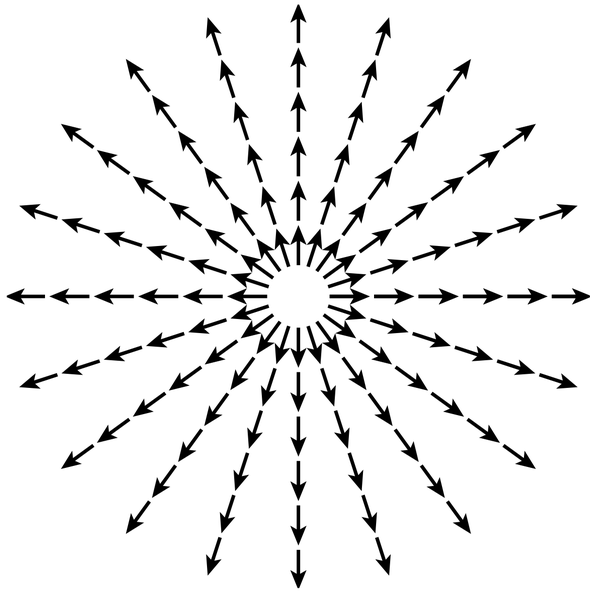
Recovering Distance

$$\min_{\phi} ||\nabla\phi - X||^2$$

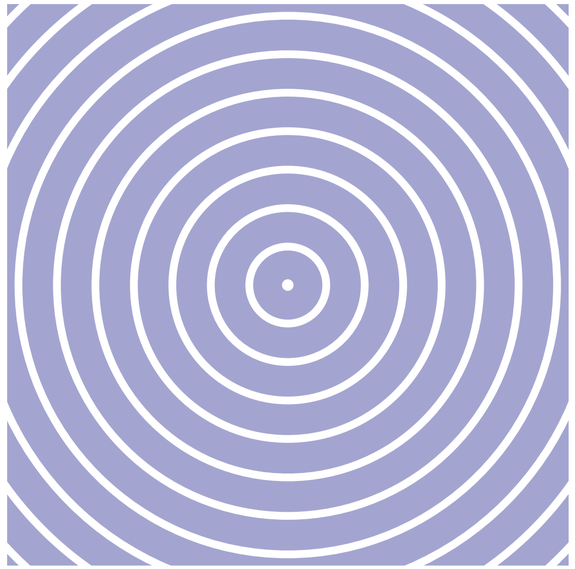
$$\Longleftrightarrow$$

$$\Delta\phi = \nabla \cdot X$$

Poisson equation

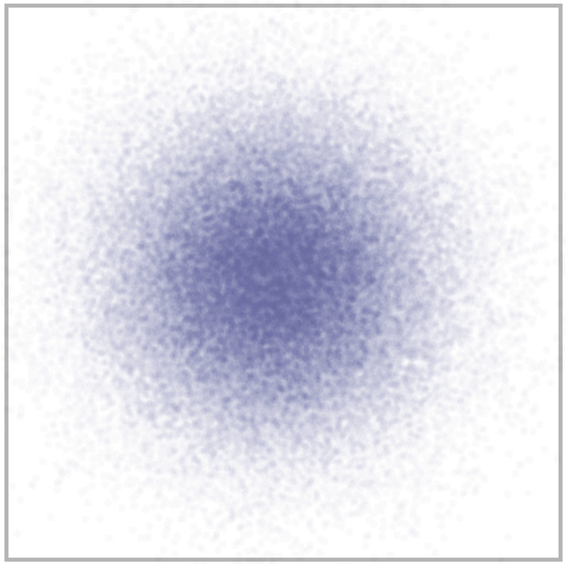


X

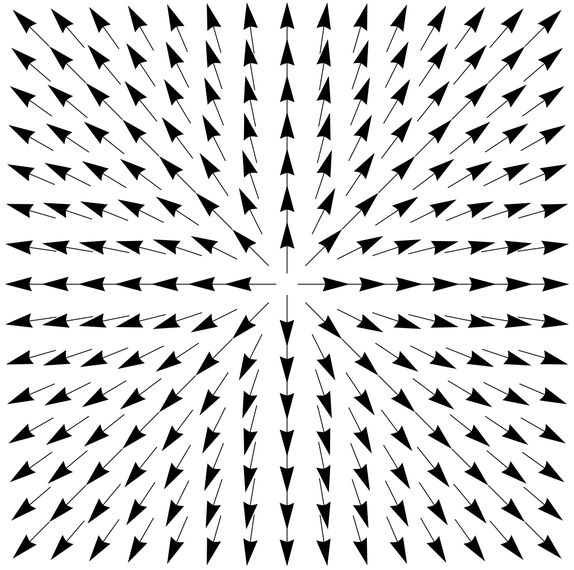


ϕ

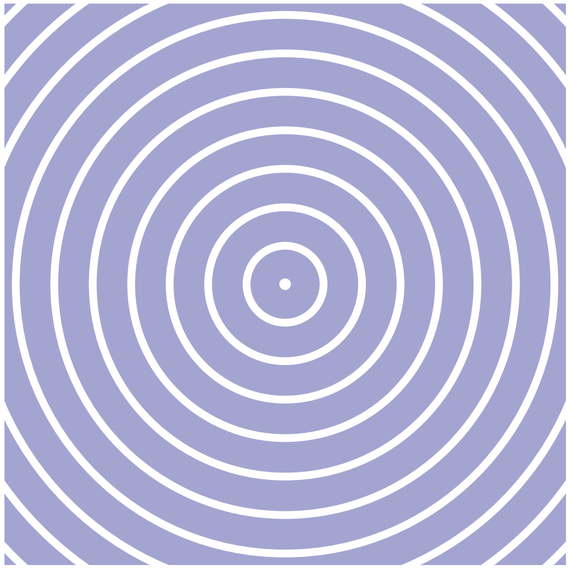
The Heat Method



u



X



ϕ

LINEAR

I. Solve heat equation

EASY

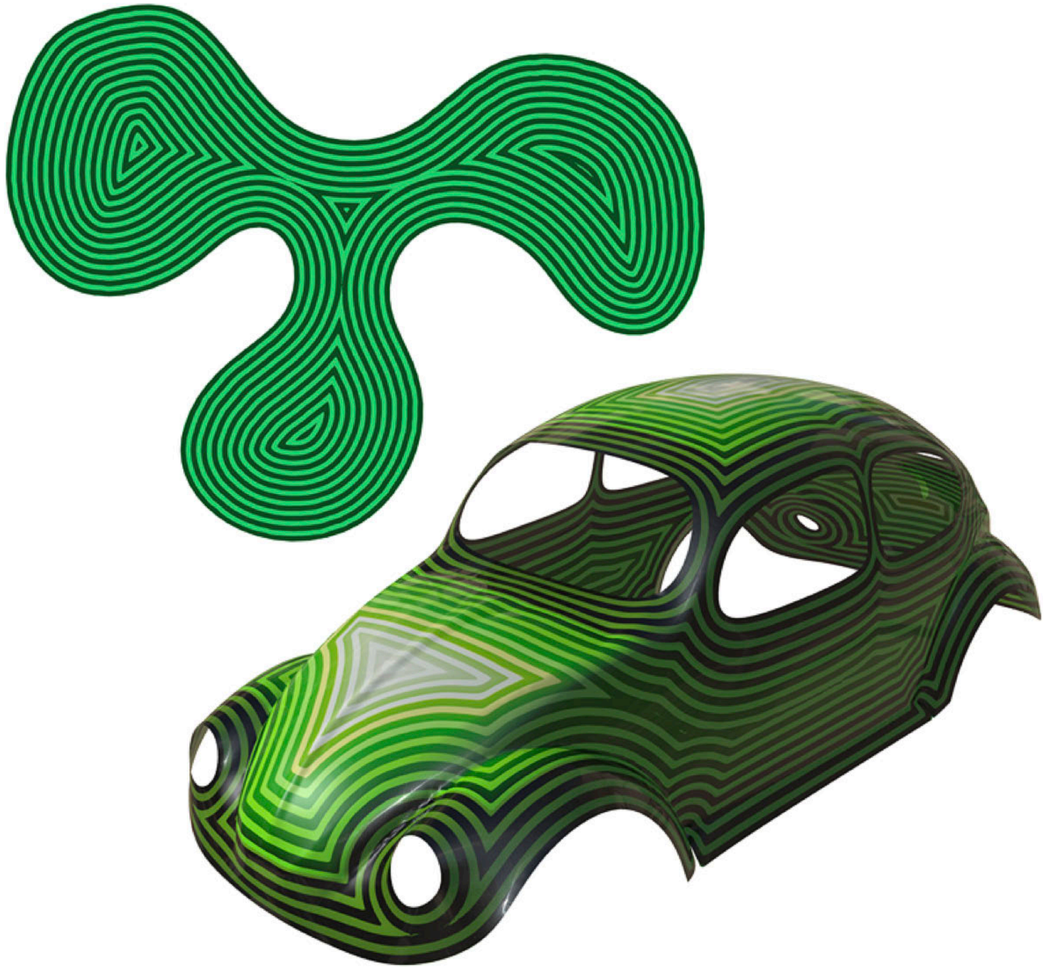
II. Normalize gradient

LINEAR

III. Solve Poisson equation

Heat Method—Examples

distance to point source



distance to boundary curve

Heat Method—Time Discretization

- So far, heat method is an algorithm in the smooth setting. Many ways to discretize…
- Discretize in *time* using backward Euler step
  - Smaller t values tend toward *graph distance* rather than geodesic distance
  - Best time step depends on mesh resolution; optimal strategy is related to scaling behavior of heat equation ($t = h^2$)

$$\frac{d}{dt}u = \Delta u$$

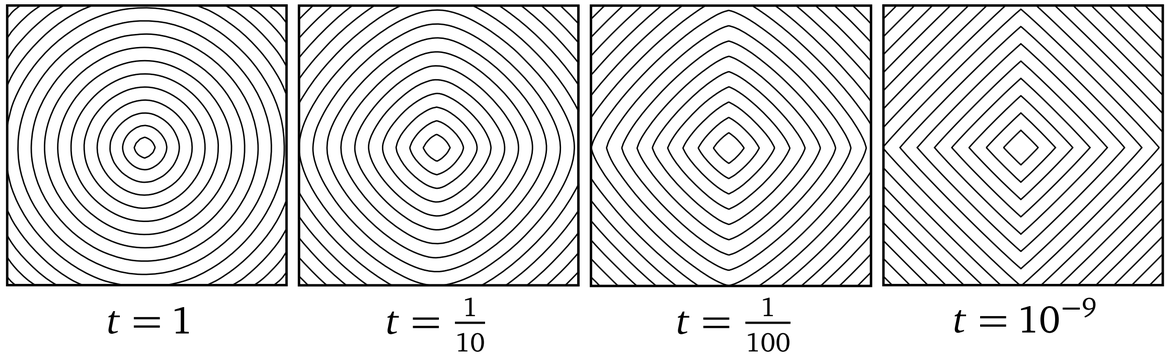
heat equation

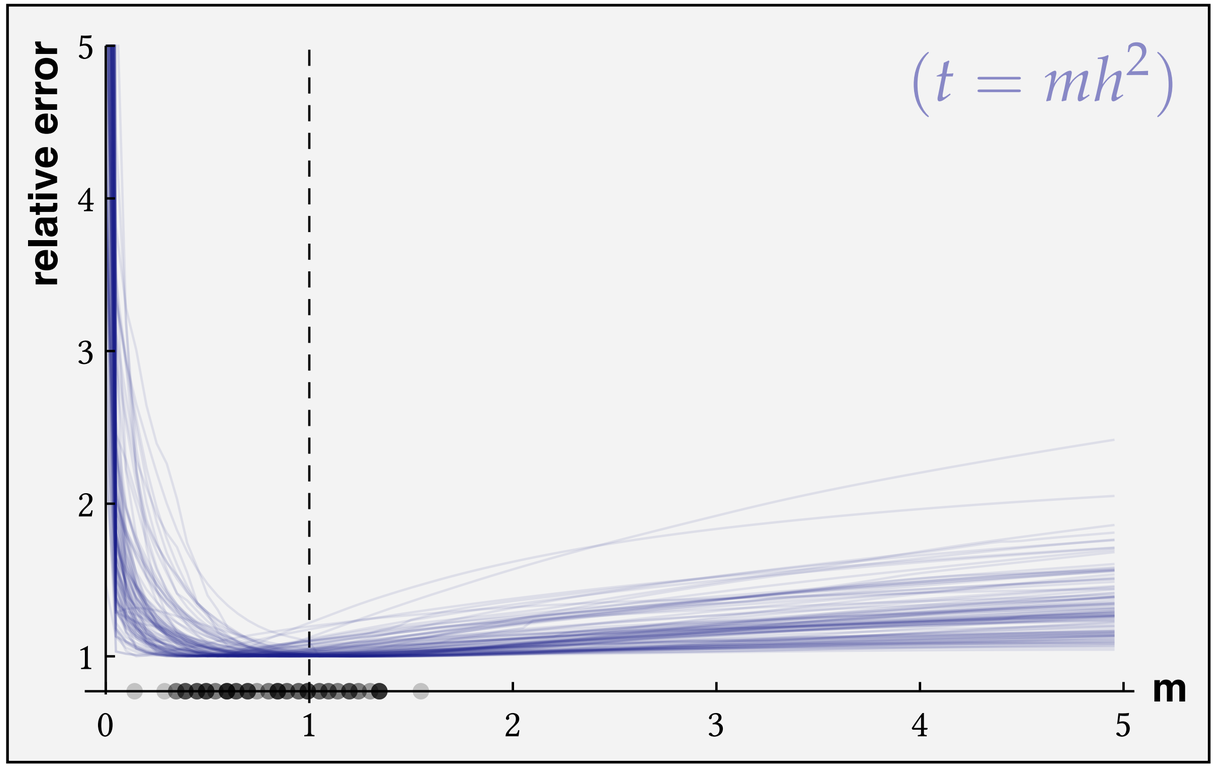
$$\frac{1}{t}(u_t - u_0) \approx \Delta u_t$$

backward Euler

$$(\mathrm{id} - t\Delta)u_t = u_0$$

linear elliptic PDE



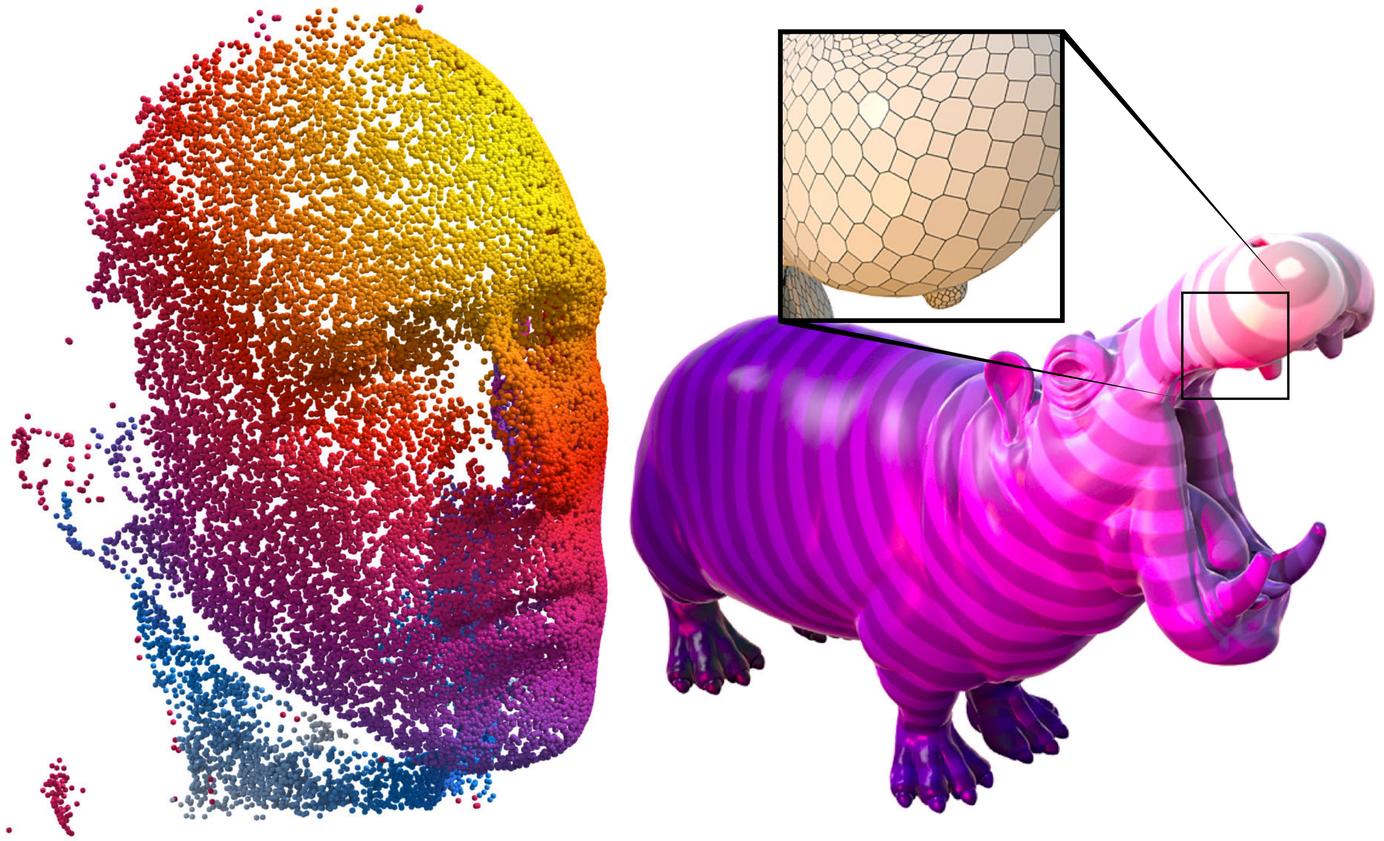


Heat Method—Spatial Discretization

- To discretize in *space*, just need:
  i. discrete gradient operator
  ii. inner product (mass matrix)
- Divergence is then adjoint of gradient; Laplacian is divergence of gradient
- *One possible* discretization: piecewise linear elements on triangle meshes
- But can easily adapt to point clouds, polygon meshes, voxelizations, …
- Often not clear how to adapt traditional algorithms to these settings (use triangles as starting point)

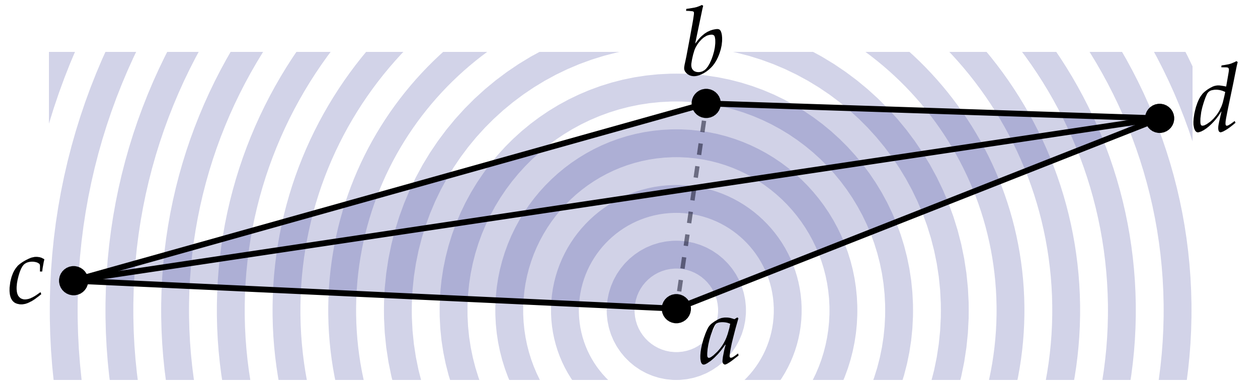
$$\langle \operatorname{grad} f, X \rangle = \langle f, \operatorname{div} X \rangle$$

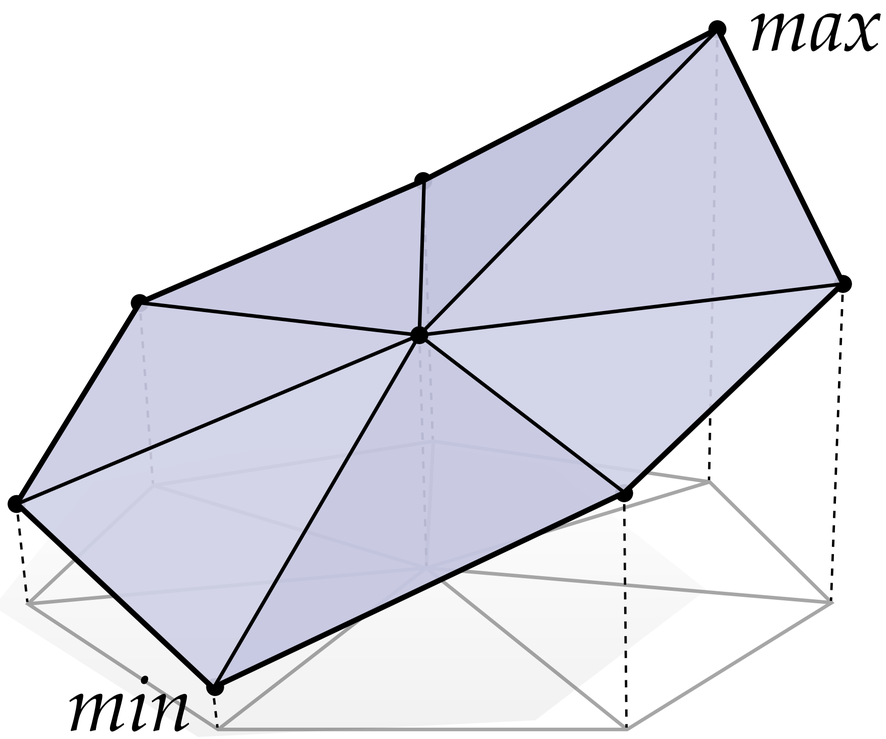
$$\Delta = \operatorname{div} \circ \operatorname{grad}$$



Accuracy and Triangulation Quality

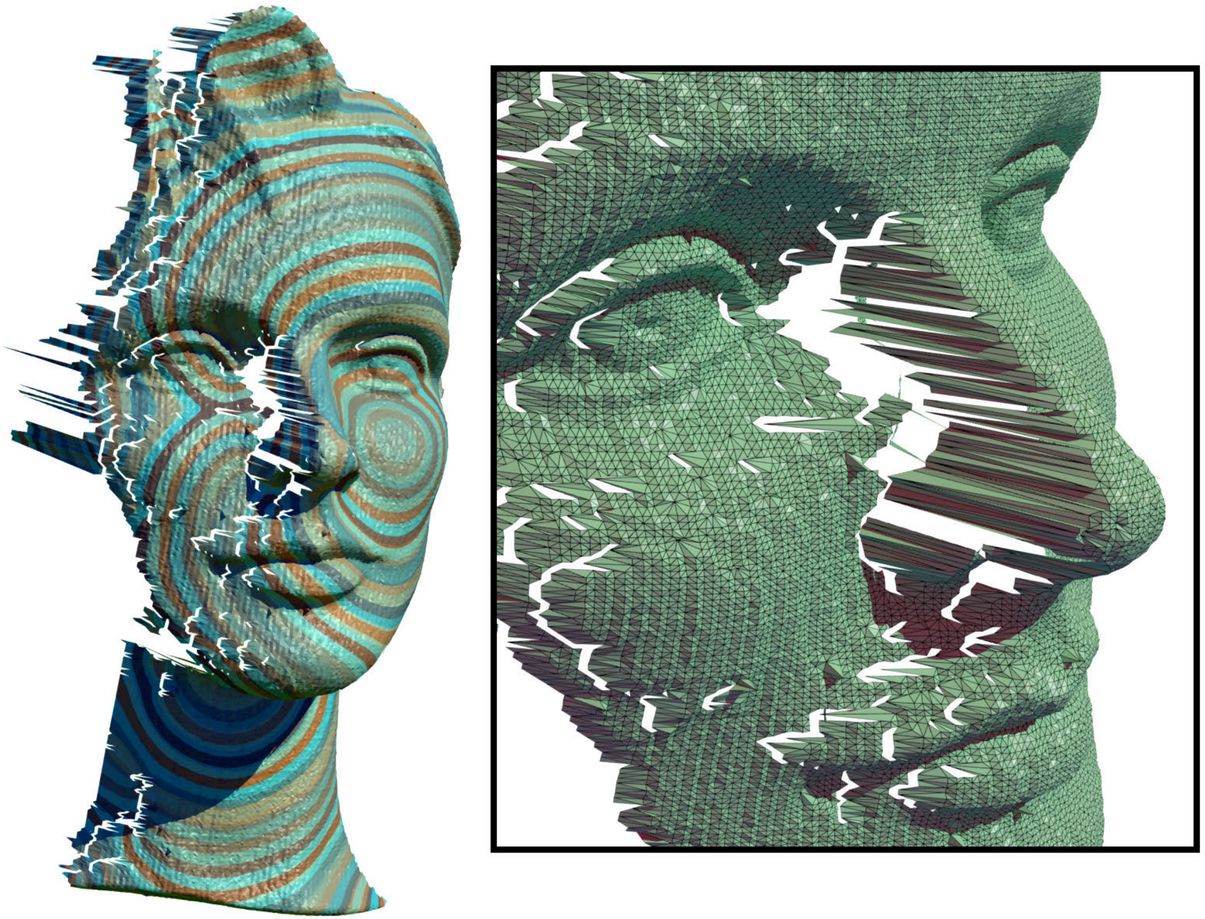
- Most basic approach: PL FEM
- Like <u>any</u> method based on finite elements, accuracy depends on quality of triangulation
- **Hyperbolic** problems like eikonal equation need **nonobtuse** triangulations to avoid violation of *causality* (very hard to obtain—especially in higher dimensions)
- **Elliptic** problems used by heat method need only **Delaunay** triangulation, to avoid violation of *maximum principle* (easy to obtain)





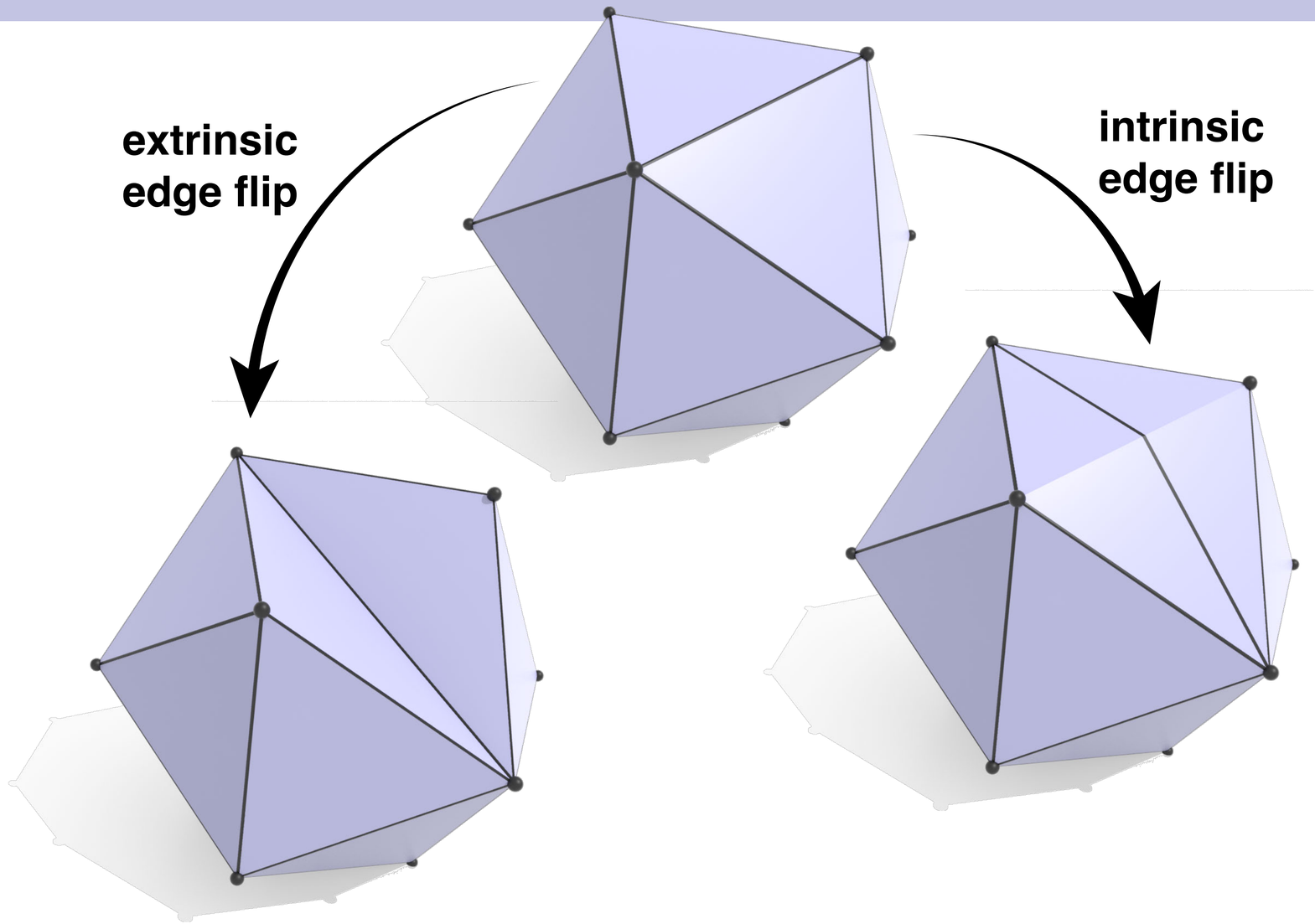
Robustness

Fairly robust even for poor-quality, non-Delaunay triangulation:



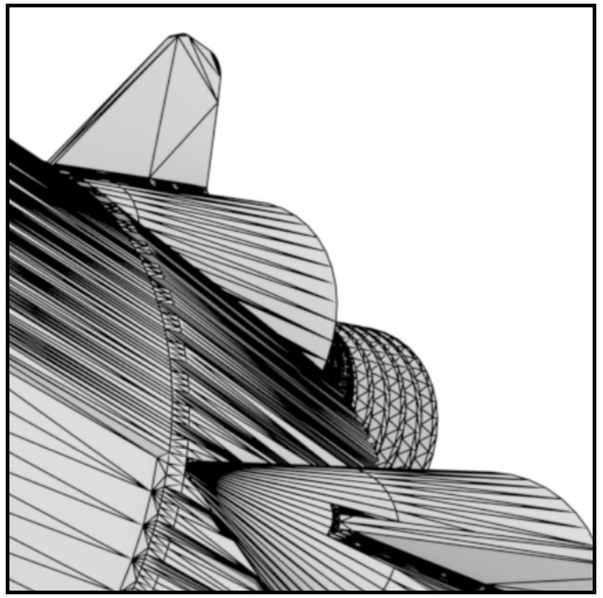
Improving Accuracy—Intrinsic Triangulations

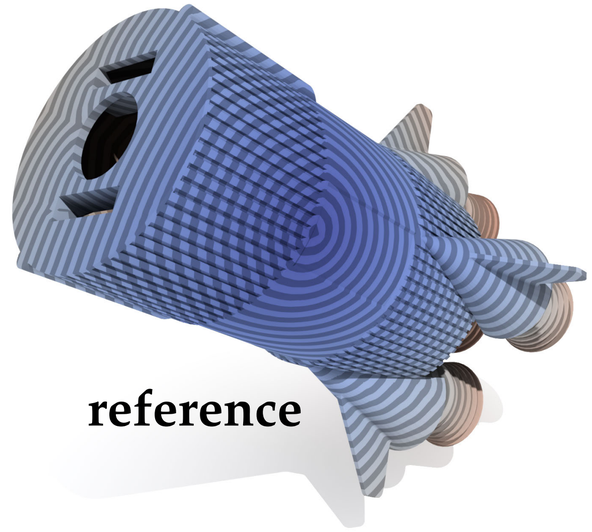
- Can also achieve Delaunay criterion without making **any** change to geometry!
- How? Use *intrinsic triangulation*



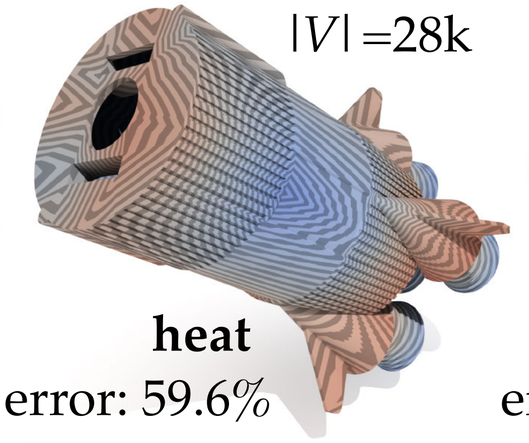
Improving Accuracy—Intrinsic Triangulations

- Can also achieve Delaunay criterion without making **any** change to geometry!
- How? Use *intrinsic triangulation*
- Leads to good results even on "horrifically bad" meshes
- *E.g.*, run simple *edge flip algorithm*. Same # of vertices; similar cost to prefactorization (default in CGAL)
- Can also get accuracy very close to exact polyhedral distance via *intrinsic Delaunay refinement*



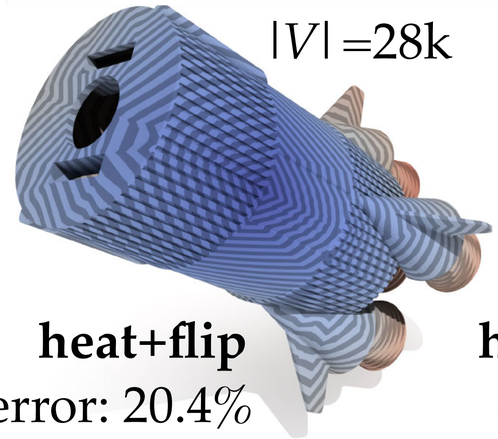


reference



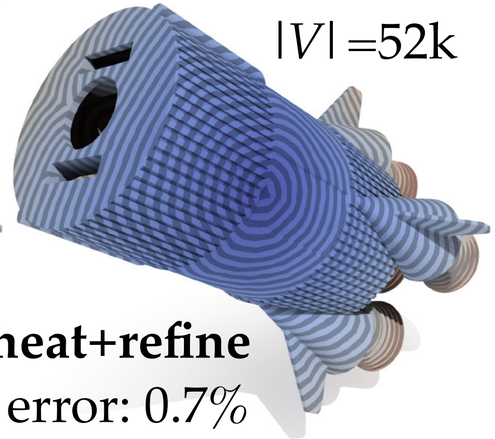
$|V|$ =28k

heat
error: 59.6%



$|V|$ =28k

heat+flip
error: 20.4%

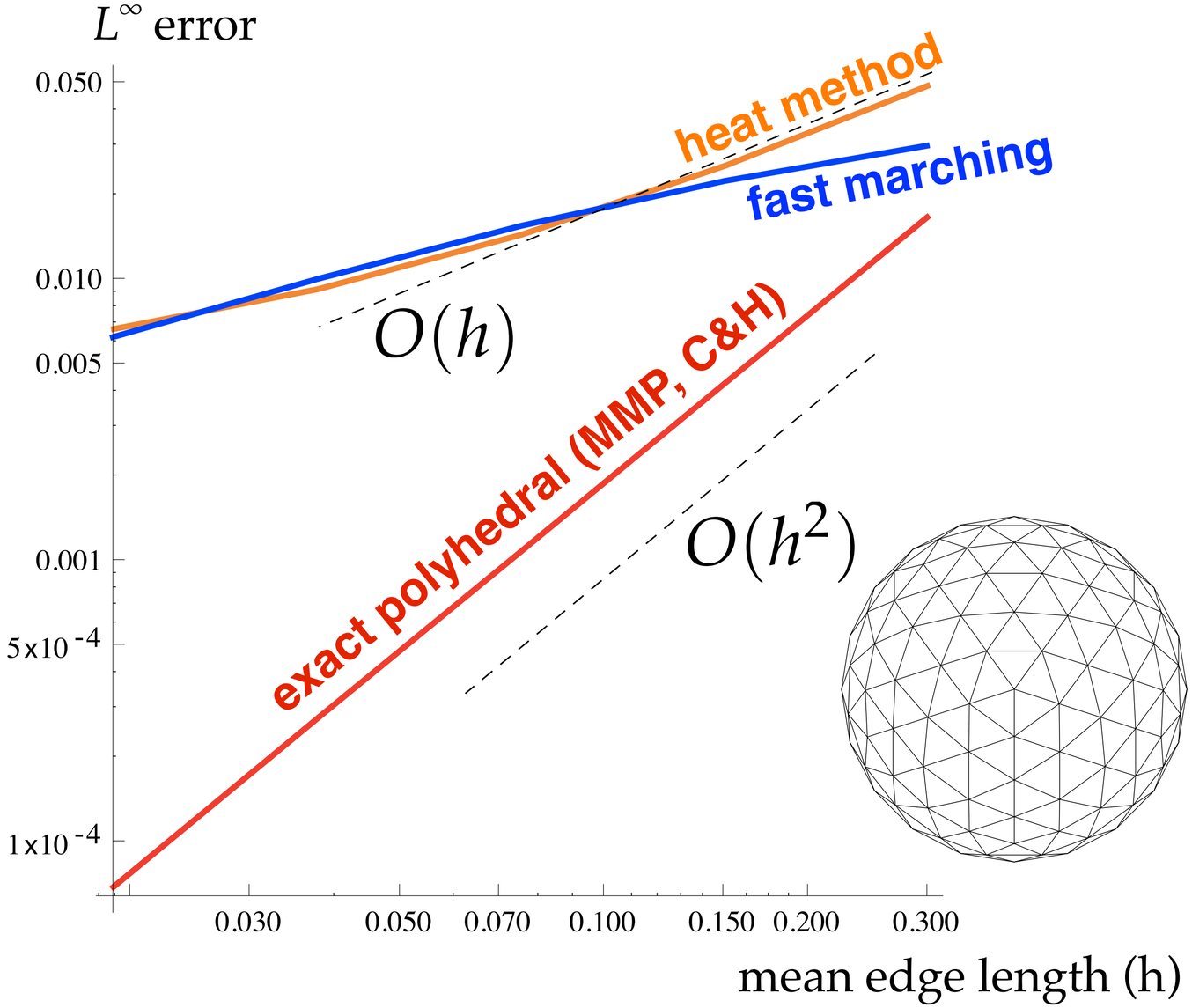


$|V|$ =52k

heat+refine
error: 0.7%

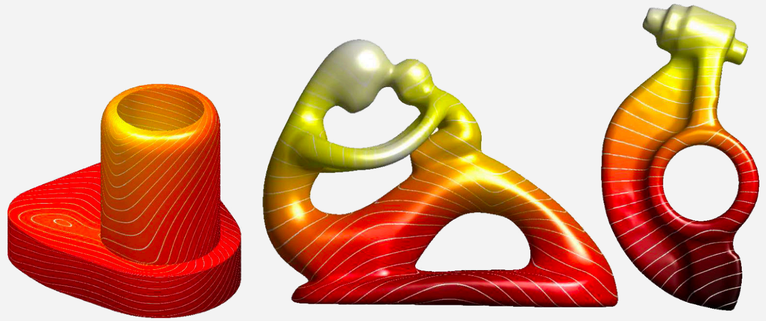
Accuracy—Piecewise Linear

- On Delaunay triangle mesh, heat method exhibits expected (linear) convergence
- Same as fast marching on nonobtuse triangulation
- On triangle meshes, ***<u>can't possibly</u>*** do better than 2nd-order accuracy!
  - triangulation of smooth surface introduces approximation error
  - even "exact" polyhedral schemes are not correct…

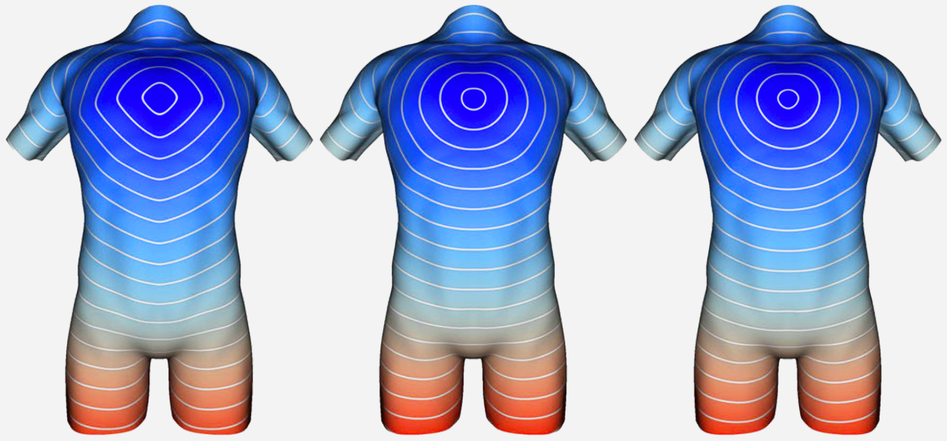


Improving Accuracy—Higher Order Elements

- Rather than improve shape of mesh elements, can also use higher-order basis functions
- Recent work on C^1 finite elements, C^2 subdivision finite elements for unstructured meshes
- Since heat method involves solving standard elliptic PDEs, easy to translate algorithm into this setting
- Error now closer to $O(h^2)$ than $O(h)$; matrices are same size, but denser



C1 finite elements on non-tensor-product manifolds
Nguyen, Karčiauskas, Peters
Applied Mathematics and Computation (2016)



Subdivision Exterior Calculus
de Goes, Desbrun, Meyer, DeRose
ACM Transactions on Graphics (2016)

Improving Accuracy—Fixed Point Iteration

Can nicely connect eikonal and diffusion perspective:

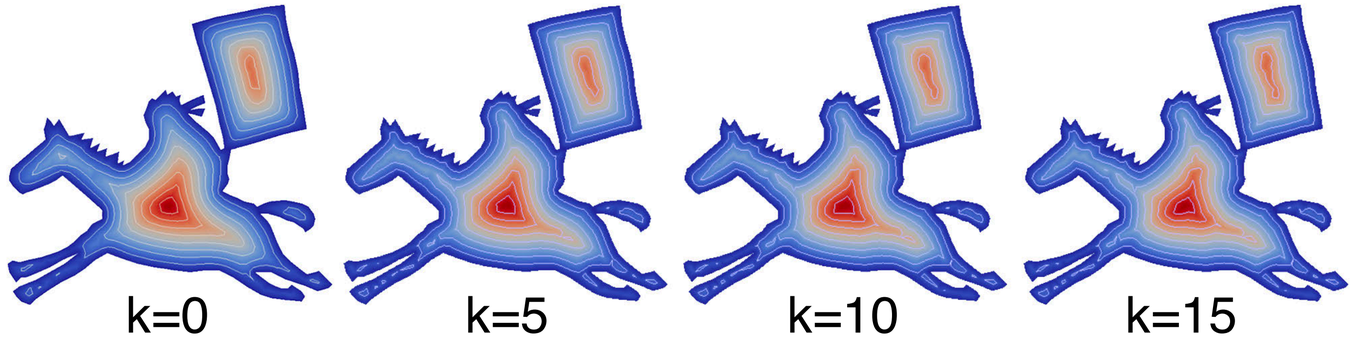
$$\boxed{\nabla\phi = 1} \implies \int_M (|\nabla\phi| - 1)^2 \, dA \implies \boxed{\Delta\phi = \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}}$$

eikonal equation — **variational principle** — **Euler-Lagrange equation**

- Can view heat method as 1st step of fixed point scheme w/ intelligent initial guess
- Further iterations improve accuracy; still just use fixed prefactorization

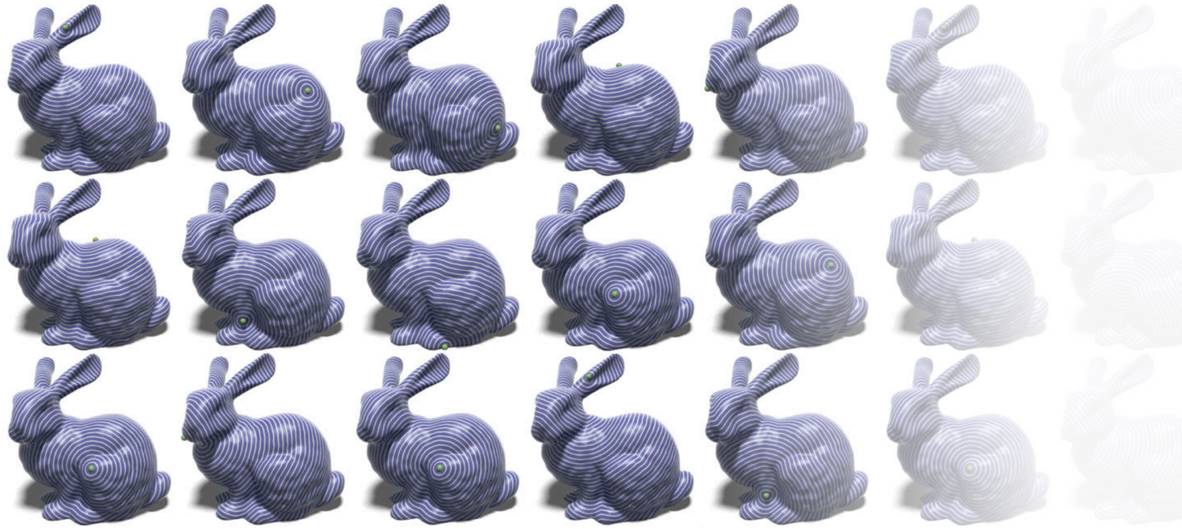
$$\Delta\phi^0 = \nabla \cdot \frac{\nabla u}{|\nabla u|}$$

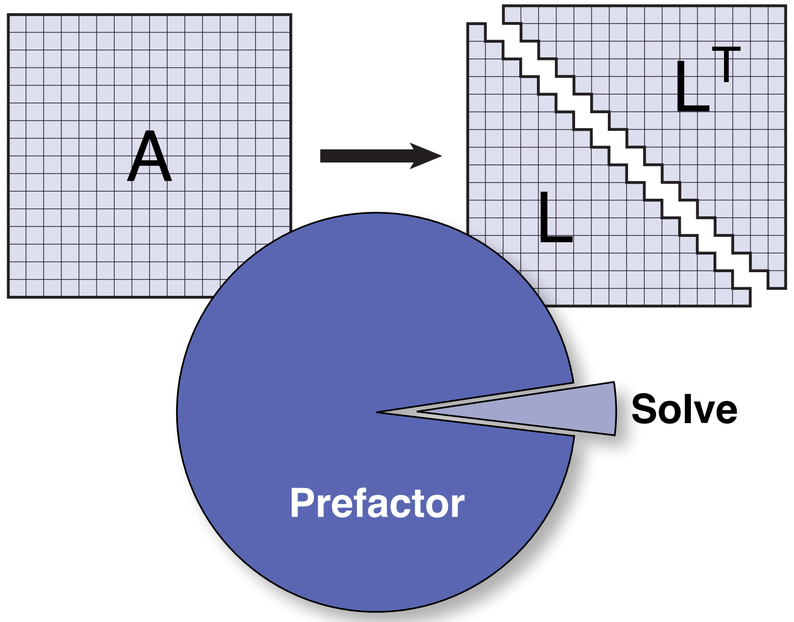
$$\Delta\phi^{k+1} = \nabla \cdot \frac{\nabla\phi^k}{|\nabla\phi^k|}$$



Improving Performance—Prefactorization

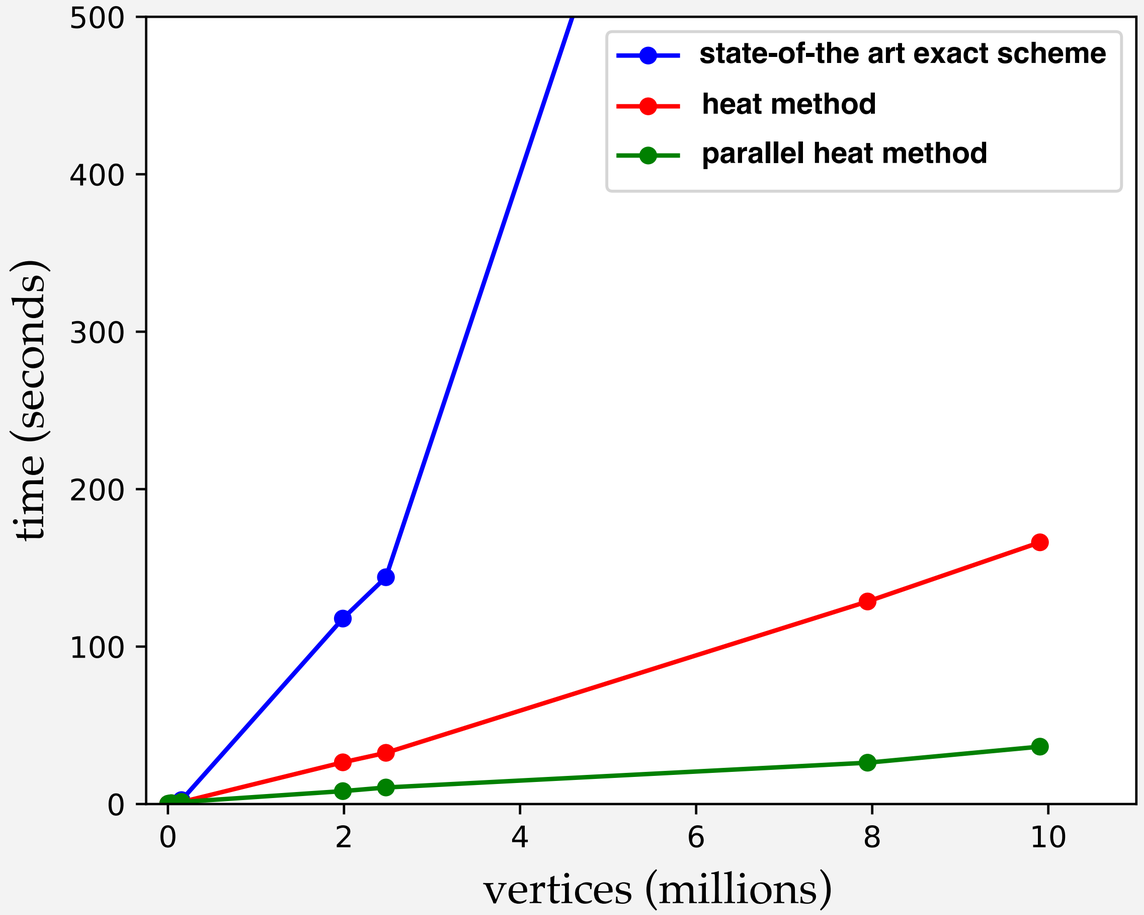
- Suppose we want to compute the distance to many sources or subsets of a given domain
- Computational bottleneck is solving heat equation & Poisson equation
- Can dramatically reduce amortized cost by *prefactoring* matrices
  - Each new source is now just back substitution
  - Around 10x faster in practice
- **Example:** all-pairs distance on 28k triangle mesh
  - *exact polyhedral:* ~7.4 hours
  - *fast marching:* ~55 minutes
  - *heat method:* **~5 minutes**





Improving Performance—Parallelization

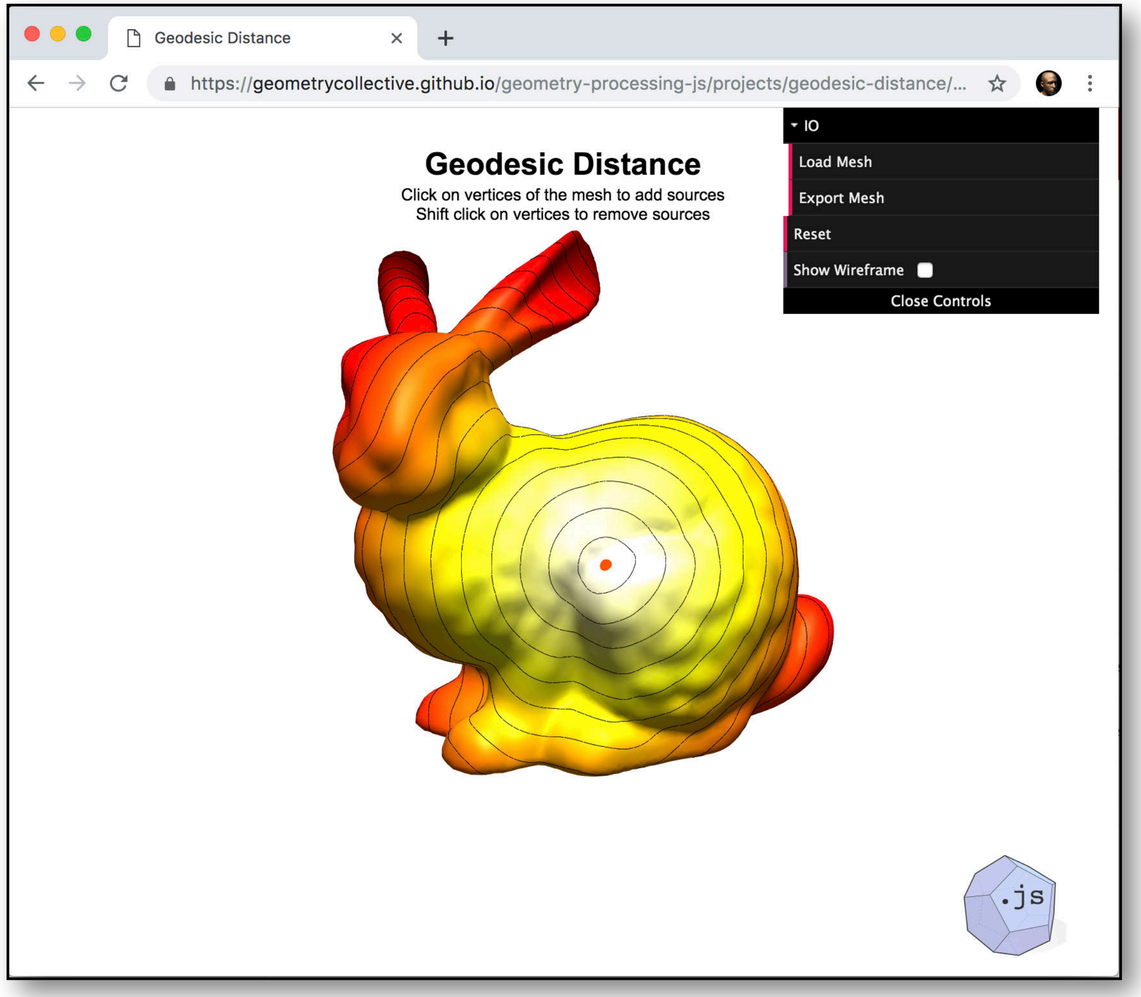
- Factorization uses significant memory for large meshes, volumetric problems
- Not *forced* to use direct solver…
- E.g., replace Cholesky factorization with ADMM scheme
- Exhibits linear scaling in time & memory (empirical)
- Scales up to ~200 million triangles in practice (8 core CPU)
- "GPU friendly" algorithm



Parallel and Scalable Heat Method
Tao, Zhang, Deng, Fang, Peng, He
arXiv:1812.06060 (2018)

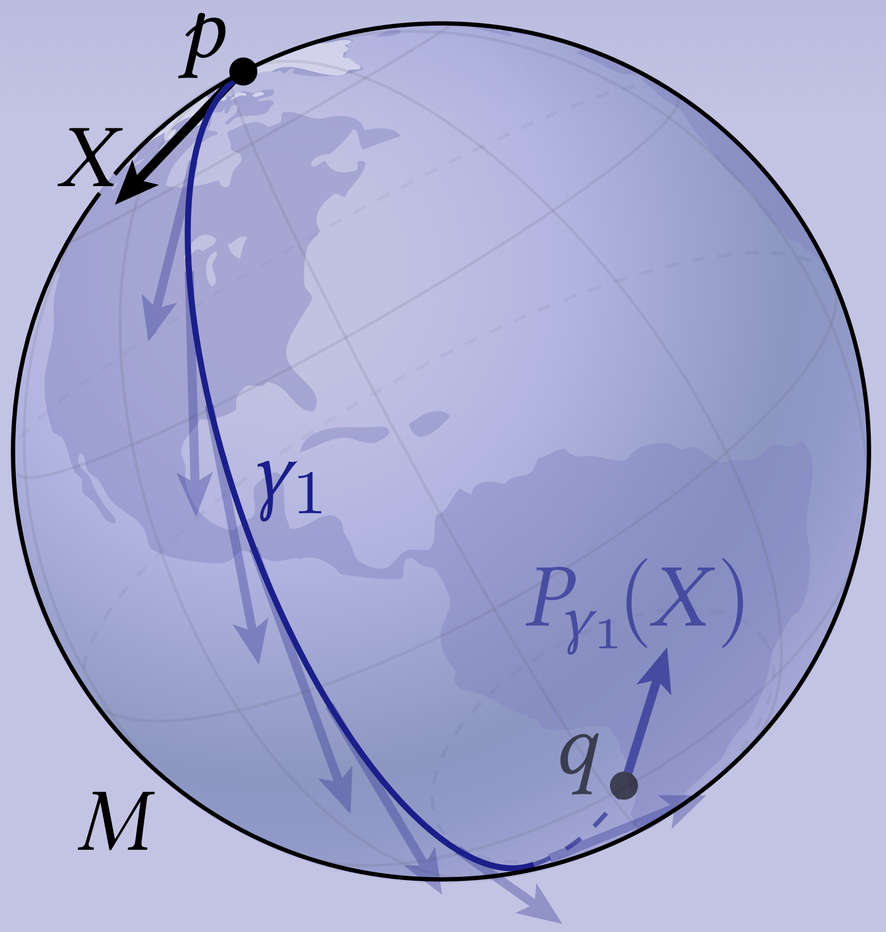
Code

- Many free and open-source implementations available:
  - C/C++
  - Python
  - MATLAB
  - Mathematica
  - Unity
  - CGAL
  - *Javascript*
  - …



http://geometry.cs.cmu.edu/js

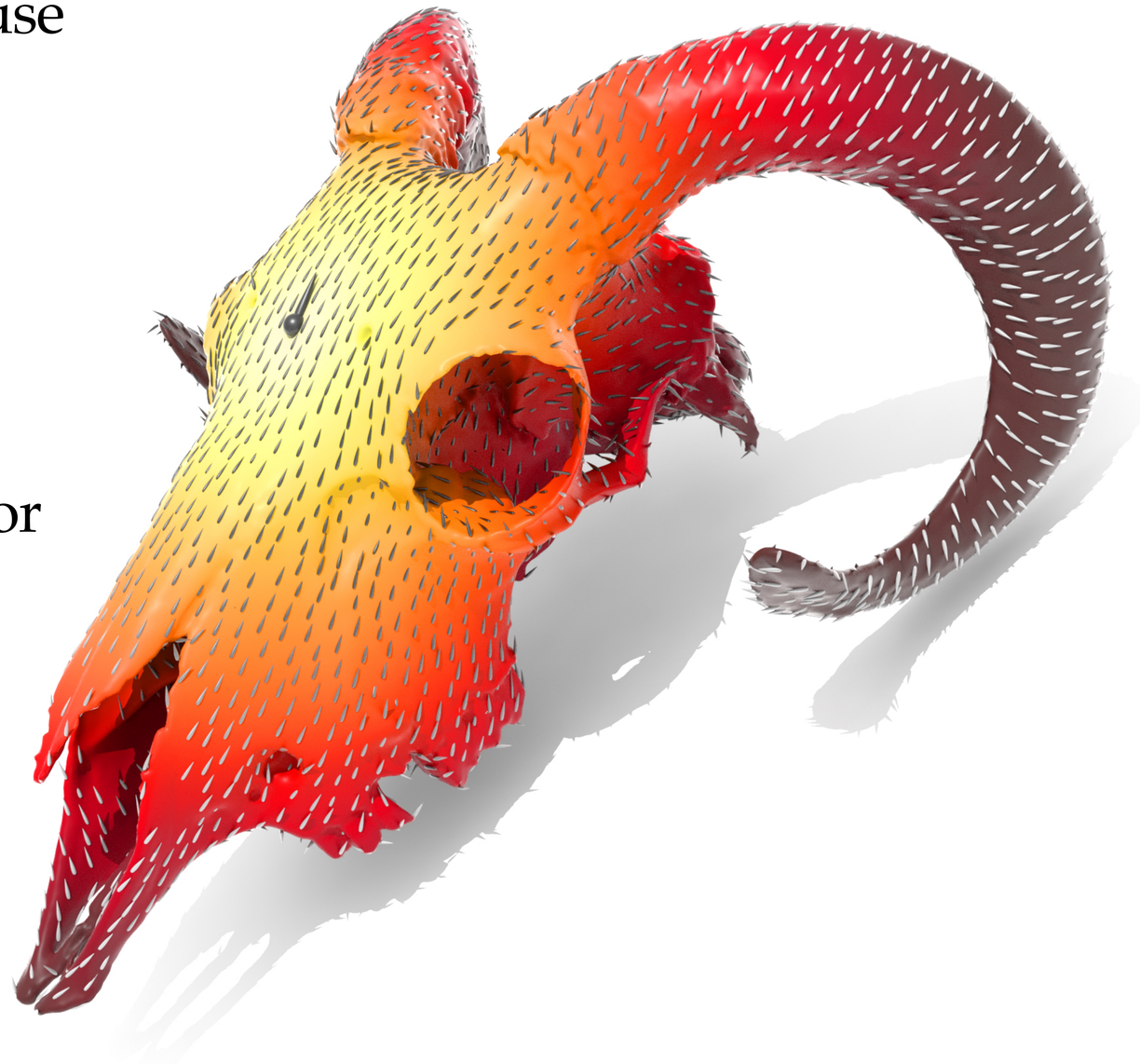
Part II:
The Vector Heat Method



The Vector Heat Method
Sharp, Soliman, Crane
ACM Transactions on Graphics (2019)

The Vector Heat Method—Overview

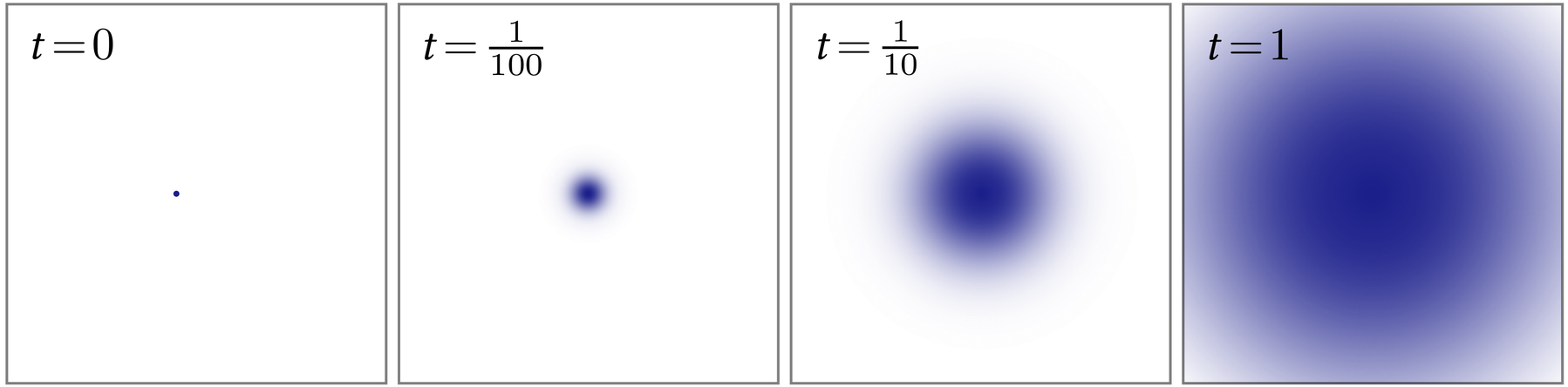
- **Natural question:** what happens if we diffuse vector- rather than scalar-valued data?
- **Short answer:** we obtain *parallel transport along minimal geodesics*
- Initially sounds like a strange quantity… (“why compute this?”)
- Turns out to be *remarkably* useful quantity for geometry processing:
  - velocity extrapolation
  - logarithmic map
  - Karcher means / geometric medians
  - centroidal Voronoi diagrams
  - …



Vector Diffusion Equation

scalar diffusion

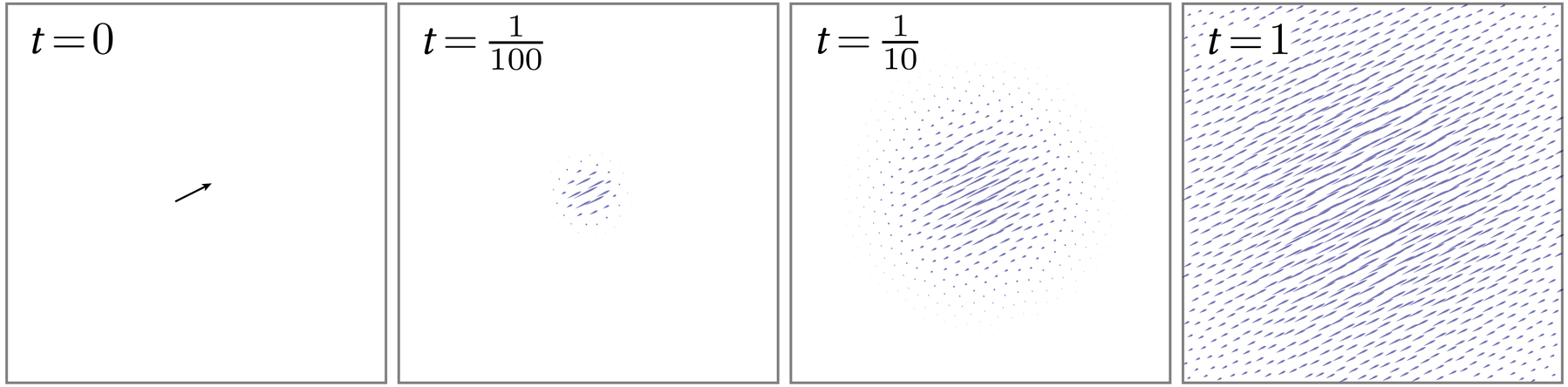
$$\frac{d}{dt}\phi = \Delta\phi$$



vector diffusion

$$\frac{d}{dt}X = \Delta^{\nabla}X$$

connection Laplacian



Connection Laplacian

- 2nd-order elliptic operator on *tangent vector fields*
- Like scalar Laplacian, many ways to interpret connection Laplacian:
  - infinitesimal generator of vector diffusion
  - composition of covariant derivative w/ adjoint
  - trace of second covariant derivative
  - Hessian of vector Dirichlet energy
- Physics: *"magnetic Schrödinger operator"*
- **Note:** <u>not</u> the same as Hodge Laplacian (related by *Weitzenböck identity*)

$$\Delta^{\nabla_1}\alpha = \Delta_1\alpha + \tfrac{1}{2}K\alpha$$

scalar	**vector**
$u(t) = e^{t\Delta}u(0)$	$X(t) = e^{t\Delta^\nabla}X(0)$
$\Delta = \mathrm{div} \circ \mathrm{grad}$	$\Delta^\nabla = \nabla^*\nabla$
$\Delta u = \mathrm{tr}(\mathrm{Hess}(u))$	$\Delta^\nabla X = \mathrm{tr}(\nabla^2 X)$
$\int_M \lvert\mathrm{grad}\ u\rvert^2\, dA$	$\int_M \lvert\nabla X\rvert^2\, dA$

"how parallel is *X*"?

Vector Heat Kernel

$(\dim = 2)$

scalar heat kernel

$$k_t(x,y) = e^{-d(x,y)^2/4t} \left(\sum_{i=0}^{\infty} t^i \Phi_i(x,y) \right)$$

time

Gaussian of distance

"heat kernel coefficients"

$$\Phi_0(x,y) = \mathrm{id}$$

identity map

vector heat kernel

$$k_t^{\nabla}(x,y) = e^{-d(x,y)^2/4t} \left(\sum_{i=0}^{\infty} t^i \Psi_i(x,y) \right)$$

$$\Psi_0(x,y) = P_{\gamma_{x \to y}}$$

parallel transport map

$$\boxed{\lim_{t \to 0} \frac{k_t^{\nabla}(x,y)}{k_t(x,y)} = P_{\gamma_{x \to y}}}$$

Vector Heat Method—First Attempt

$$\lim_{t \to 0} \frac{k_t^{\nabla}(x,y)}{k_t(x,y)} = P_{\gamma_{x \to y}}$$

- Candidate algorithm:
  - Compute a short-time scalar heat kernel
  - Compute a short-time vector heat kernel
  - Take the quotient
- Just as in scalar heat method, get approximation error for $t > 0$
- *E.g.*, in the case of **multiple** source vectors, transported vectors have wrong *magnitude*

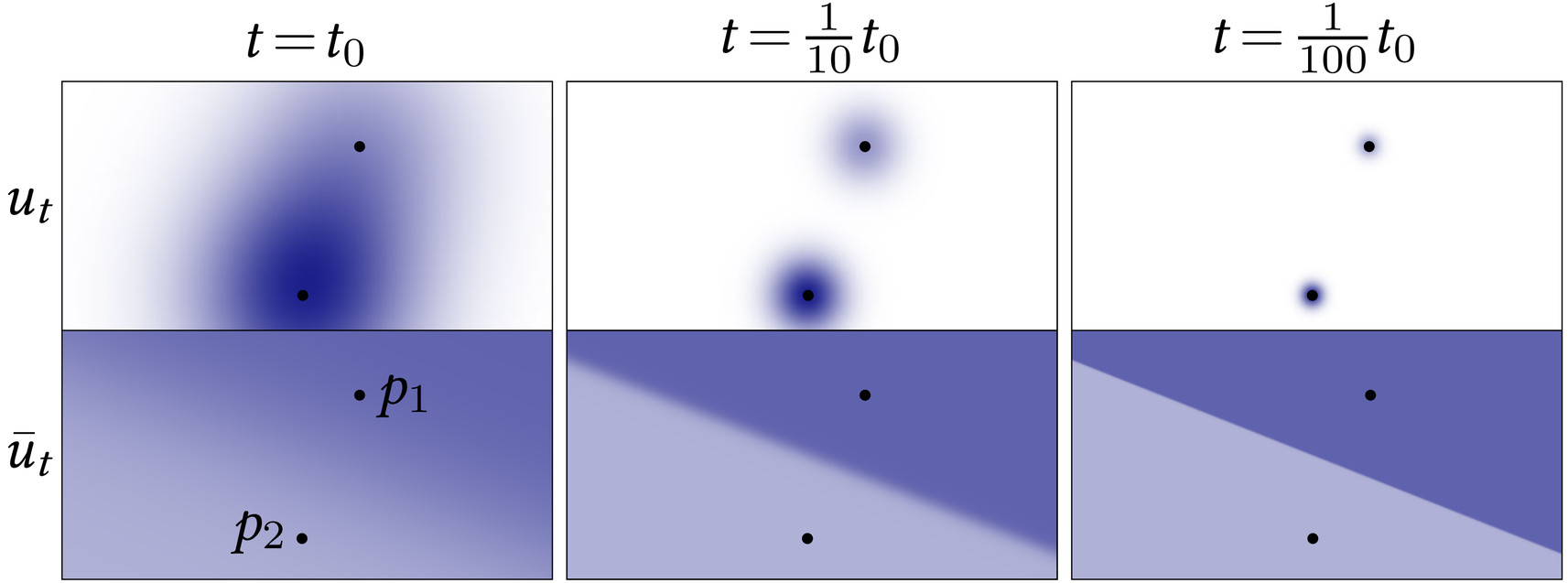
magnitude of closest vector

ratio of heat kernels

Closest-Point Interpolation by Convolution

- Suppose we only wanted the *magnitude* at the closest point. Could:
  - convolve with a Gaussian (u_t)
  - normalize by sum of Gaussians $(\bar{u}_t)$
  - take limit as t $\rightarrow$ 0

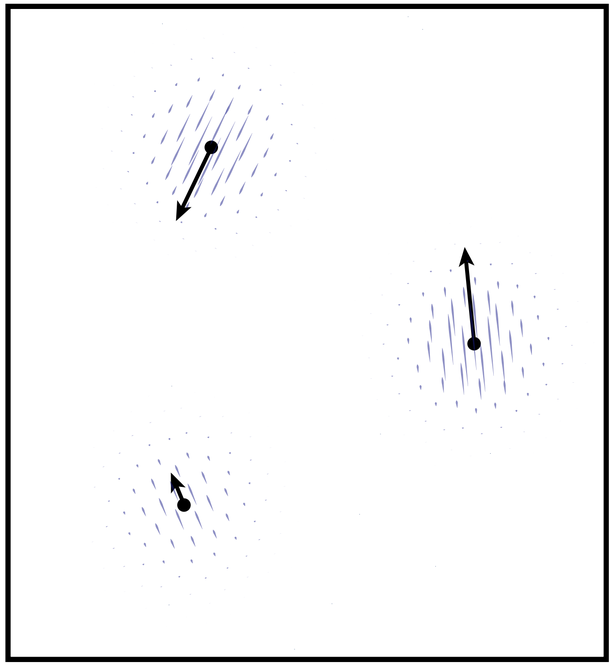
$$\bar{u}_t = \frac{u_1 G_{t,p_1} + u_2 G_{t,p_2}}{G_{t,p_1} + G_{t,p_2}}$$



Closest-Point Interpolation by Diffusion

More generally: replace convolution w/ short-time heat flow:

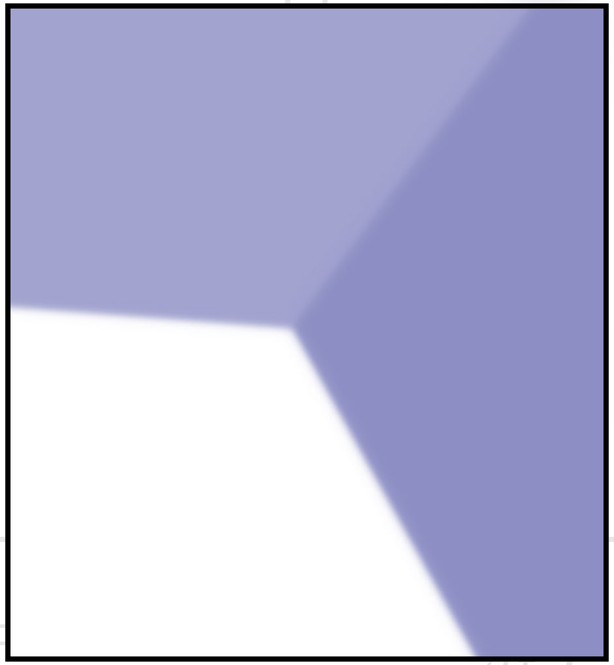
The Vector Heat Method



$$\frac{d}{dt}Y = \Delta^{\nabla} Y$$

$$Y(0) = X$$

I. Diffuse vectors



$$\frac{d}{dt}u = \Delta \frac{u}{|\phi|}$$

$$u(0) = |X|$$

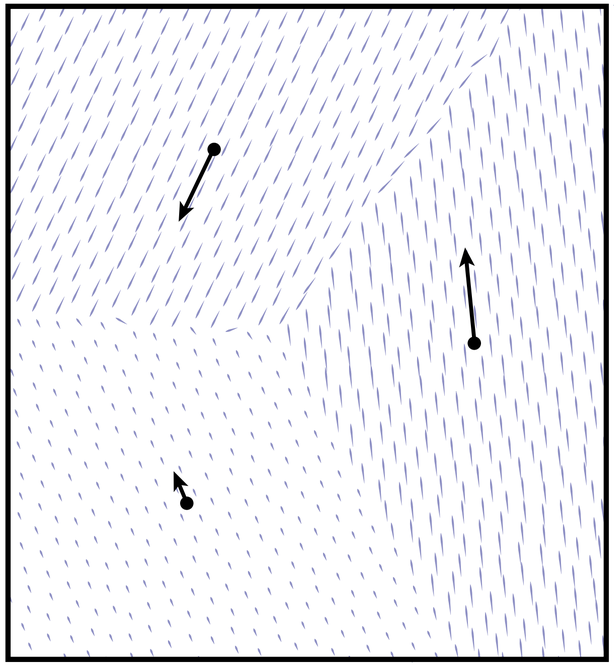
II. Diffuse magnitudes

$$\frac{d}{dt}\phi = \phi$$

$$\phi(0) = \mathbb{1}_X$$

III. Diffuse indicator

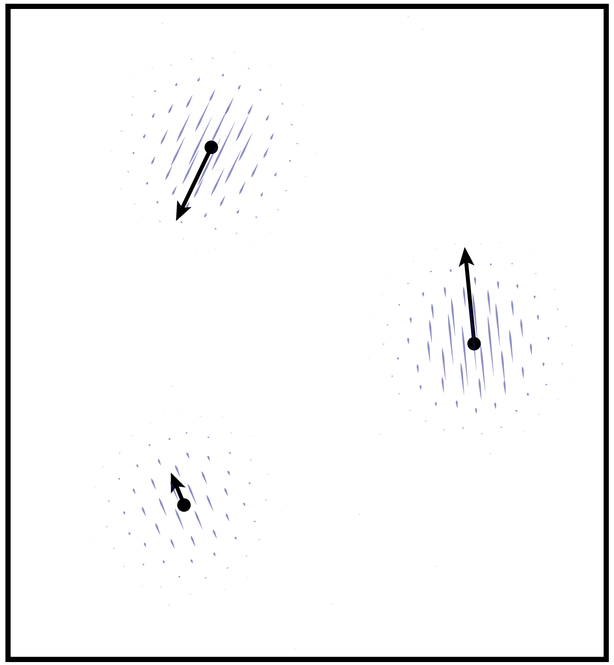
magnitude of closest vector



$$\overline{X} = \frac{u}{\phi}\frac{Y}{|Y|}$$

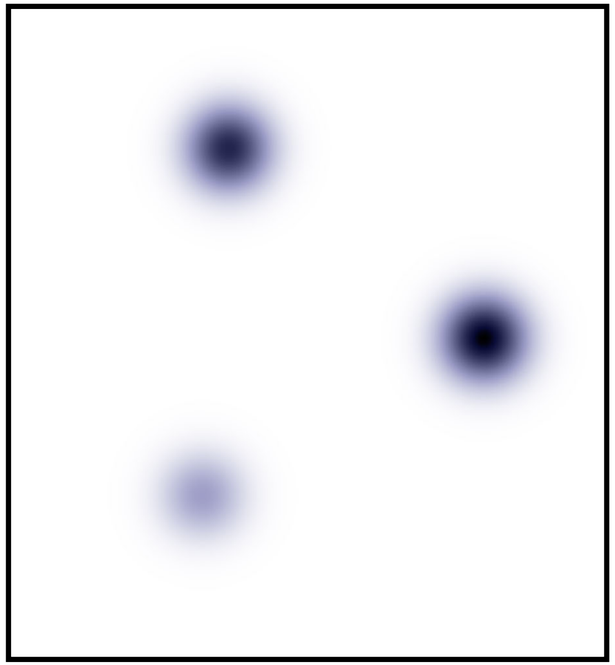
IV. Scale vectors

The Vector Heat Method



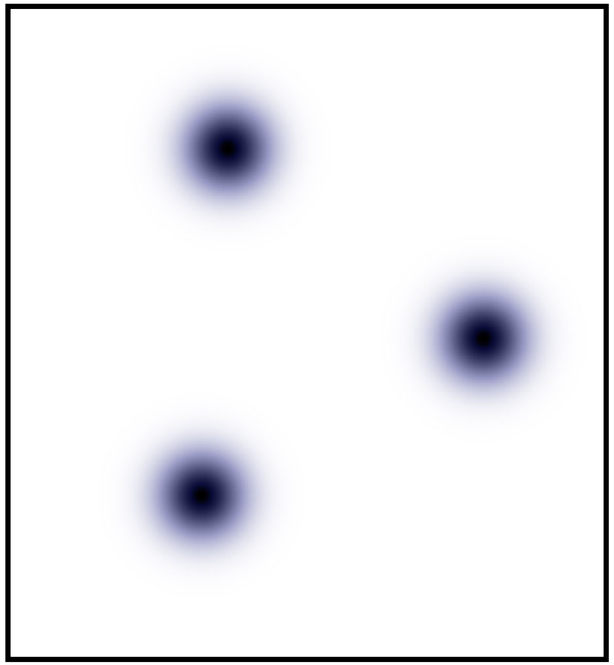
LINEAR

I. Diffuse vectors



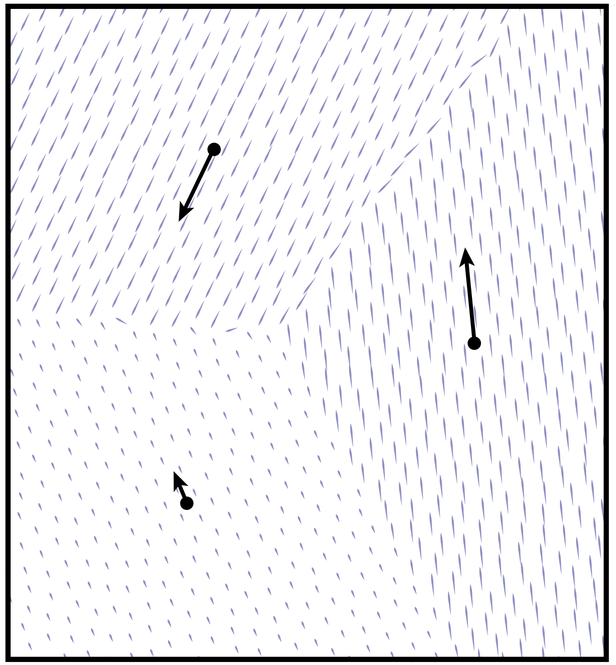
LINEAR

II. Diffuse magnitudes



LINEAR

III. Diffuse indicator

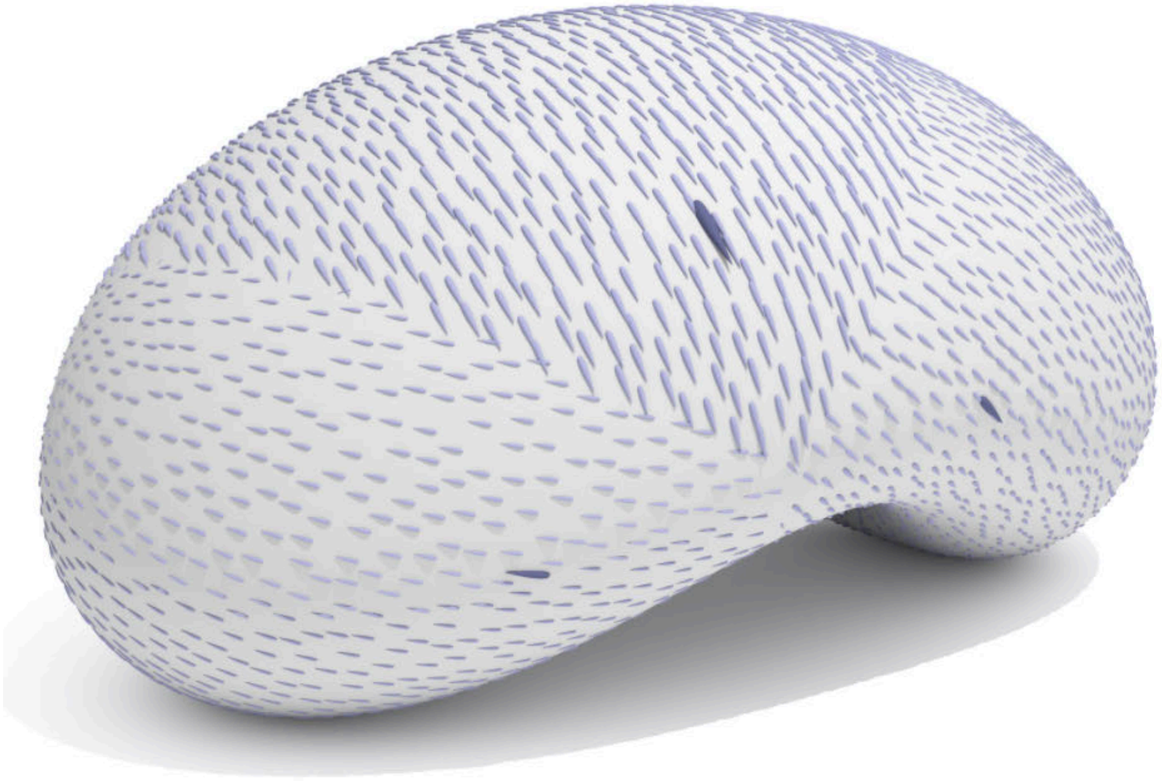


EASY

IV. Scale vectors

Parallel Transport by Diffusion

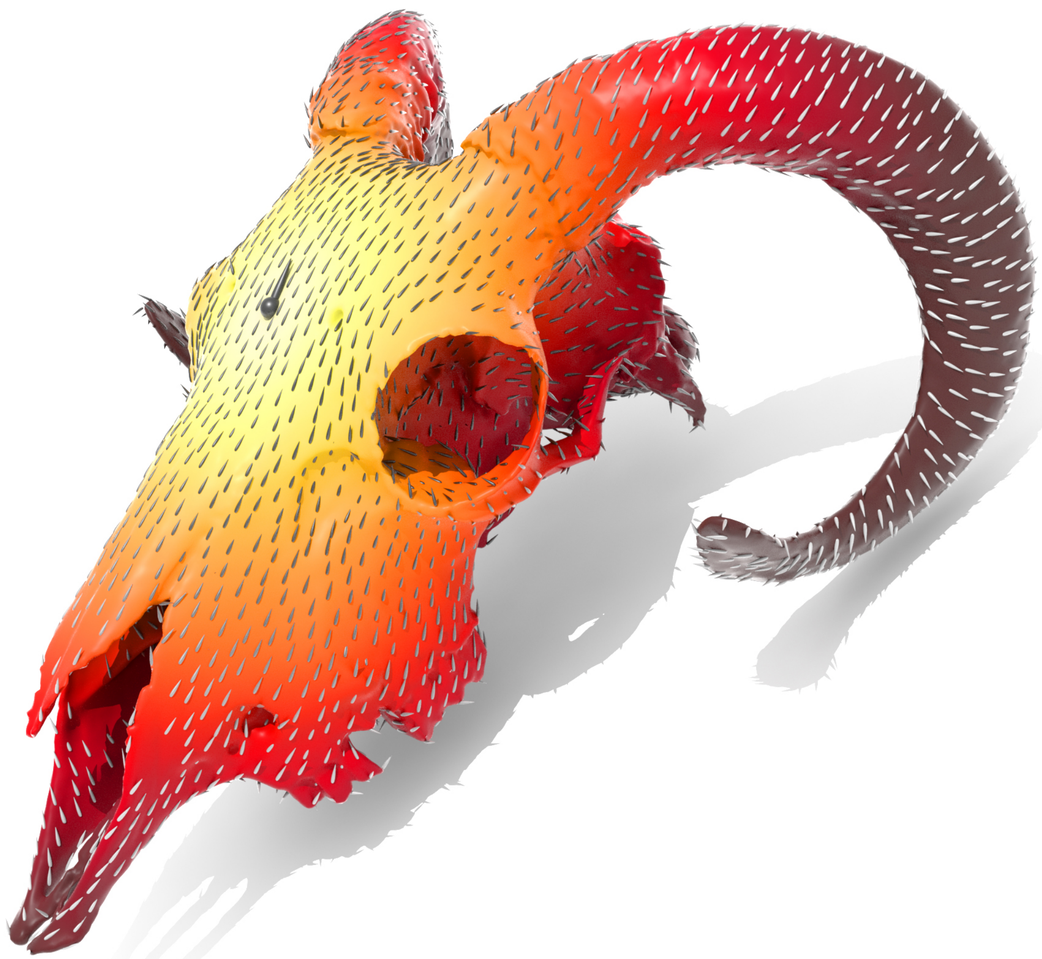
On curved surfaces, yields parallel transport along shortest geodesics:

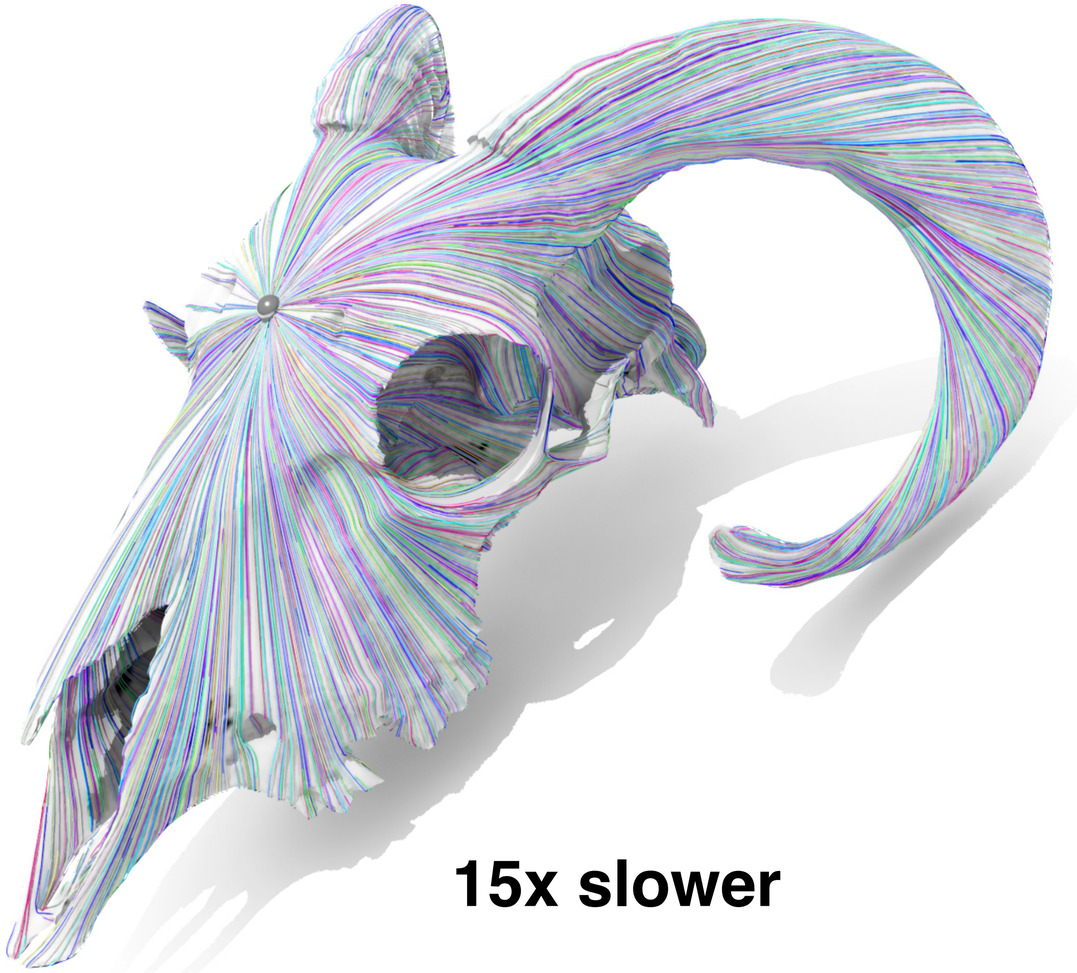


Did not have to explicitly find geodesics… or perform parallel transport!

Why Not Just Transport Vectors Explicitly?

Even just *tracing* geodesics (given distance function) is much slower:





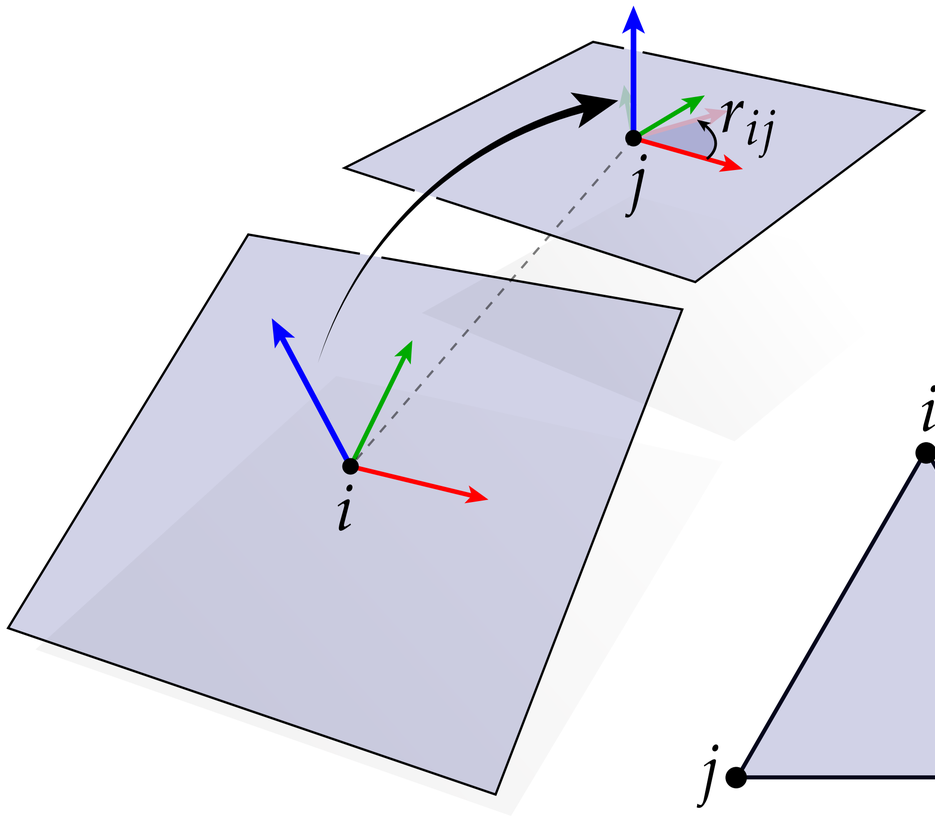
15x slower

…not to mention additional cost of *computing* distance, applying transport.

Connection Laplacian—Discretization Principle

- Discretization same as scalar heat method, but need *discrete connection Laplacian*
- Surprisingly little work on discretization [Zahariev 2008, Singer & Wu 2012, Knöppel et al 2013, Knöppel et al 2015]
- Simple approach on any discrete manifold:
  - start w/ scalar "graph-like" Laplacian
  - measure smallest rotation between neighboring tangent spaces
  - augment off-diagonal entries w/ rotations
- Hessian of discrete Dirichlet energy:

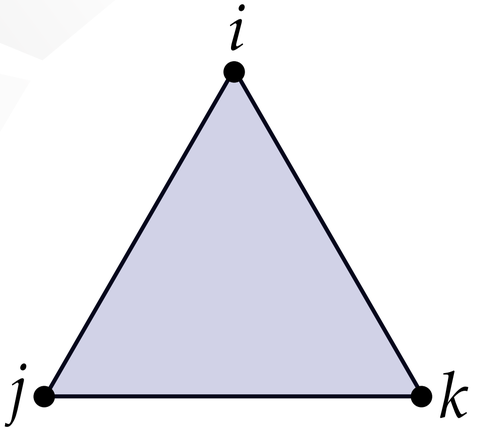
$$E(X) := \sum_{ij} L_{ij} |X_j - r_{ij} X_i|^2$$



rotation

$r_{ij} \in \mathbb{C}$

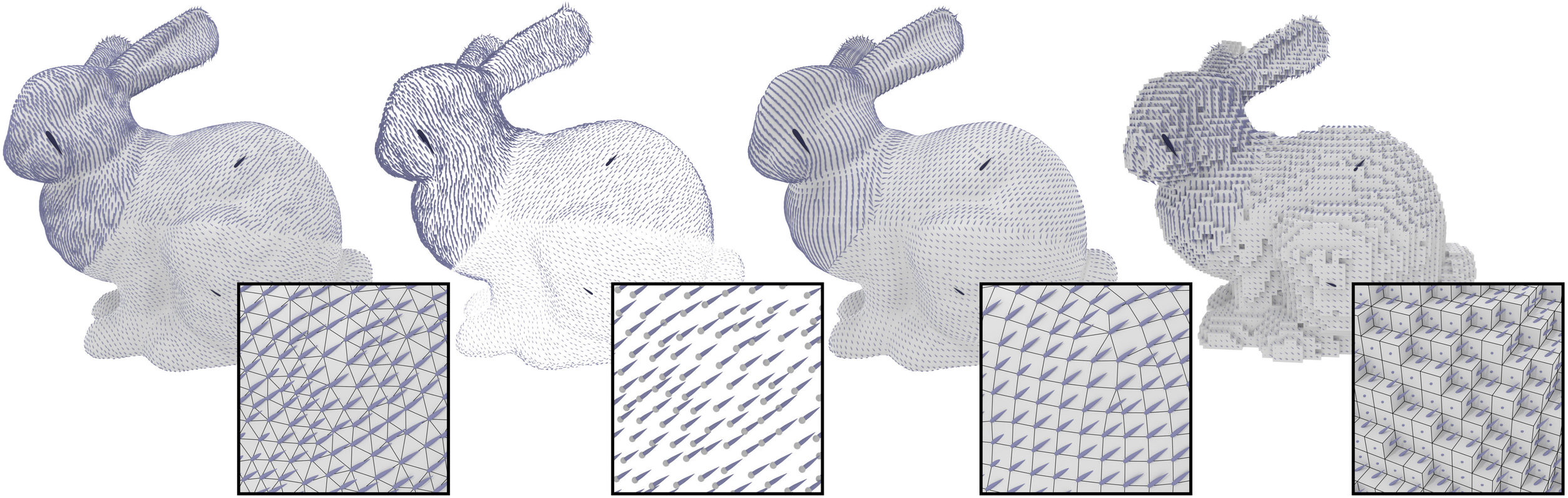
$|r_{ij}| = 1$



$$\begin{bmatrix} L_{ii} & L_{ij} & L_{ik} \\ L_{ji} & L_{jj} & L_{jk} \\ L_{ki} & L_{kj} & L_{kk} \end{bmatrix} \Longrightarrow \begin{bmatrix} L_{ii} & r_{ij} L_{ij} & r_{ik} L_{ik} \\ \bar{r}_{ij} L_{ji} & L_{jj} & r_{jk} L_{jk} \\ \bar{r}_{ik} L_{ki} & \bar{r}_{jk} L_{kj} & L_{kk} \end{bmatrix}$$

Connection Laplacian—Other Surface Representations

Can build on top of existing discrete Laplace operators:



simplicial surfaces
[MacNeal 1949]

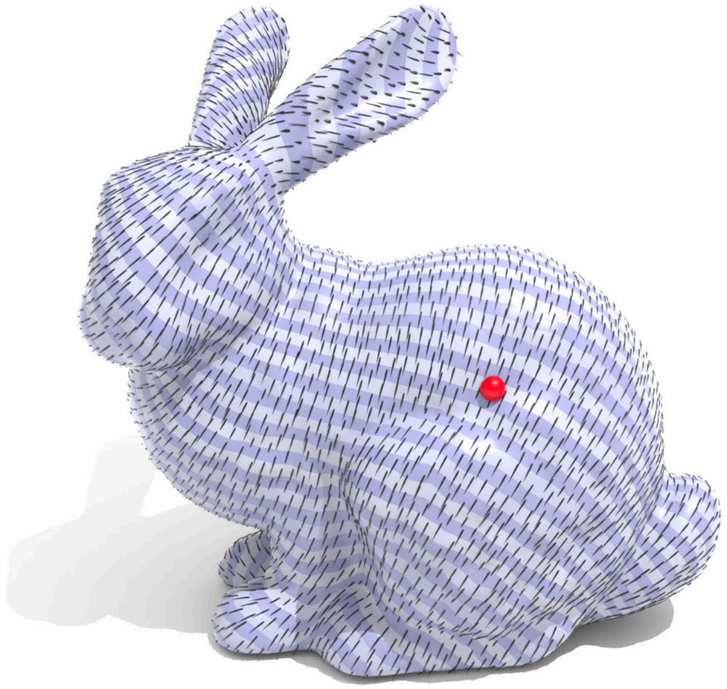
point clouds
[Liu et al 2012]

polygonal meshes
[Alexa & Wardetzky 2011]

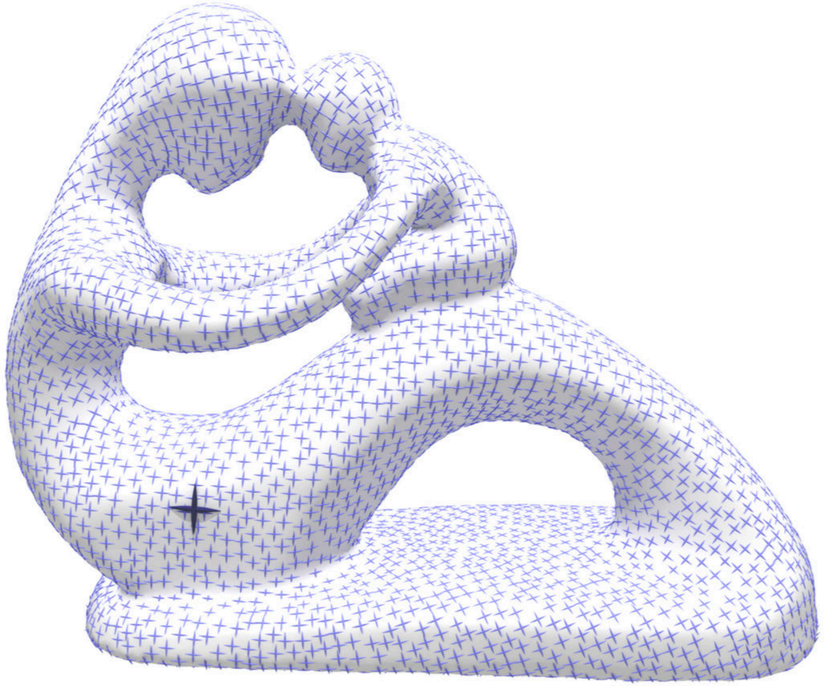
voxelizations
[Caissard et al 2017]

Connection Laplacian—Other Vector Bundles

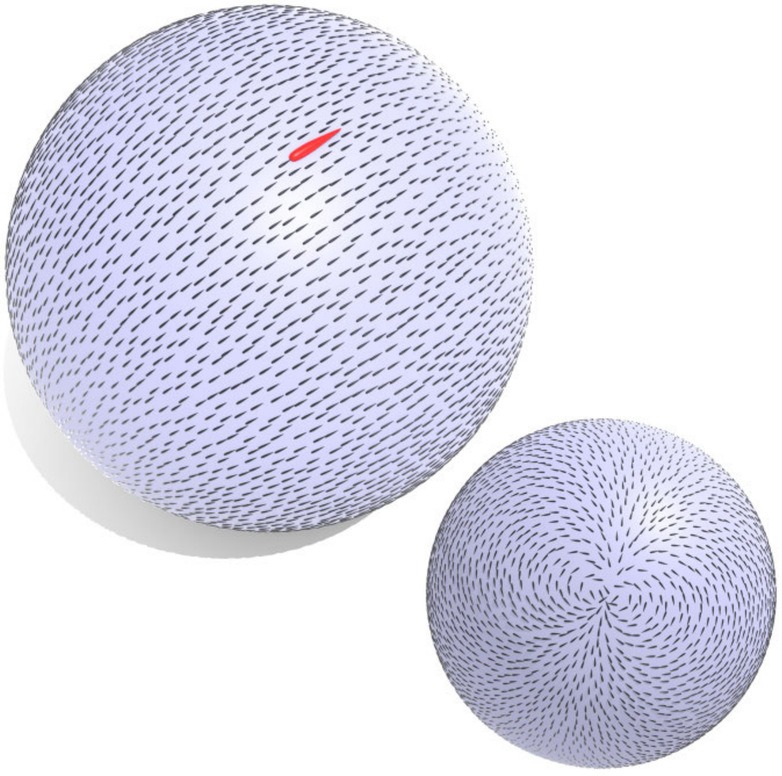
Can work with other kinds of vector-valued data:



non-metric connections



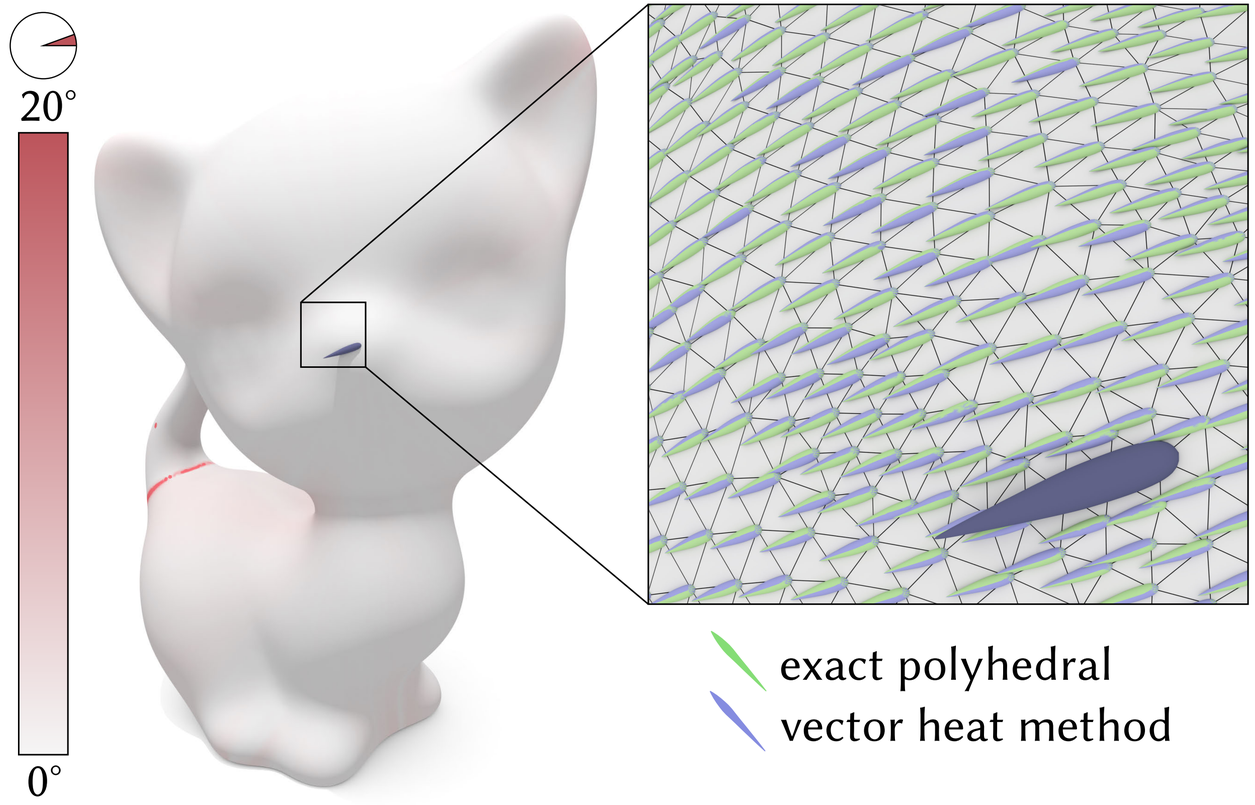
symmetric direction fields

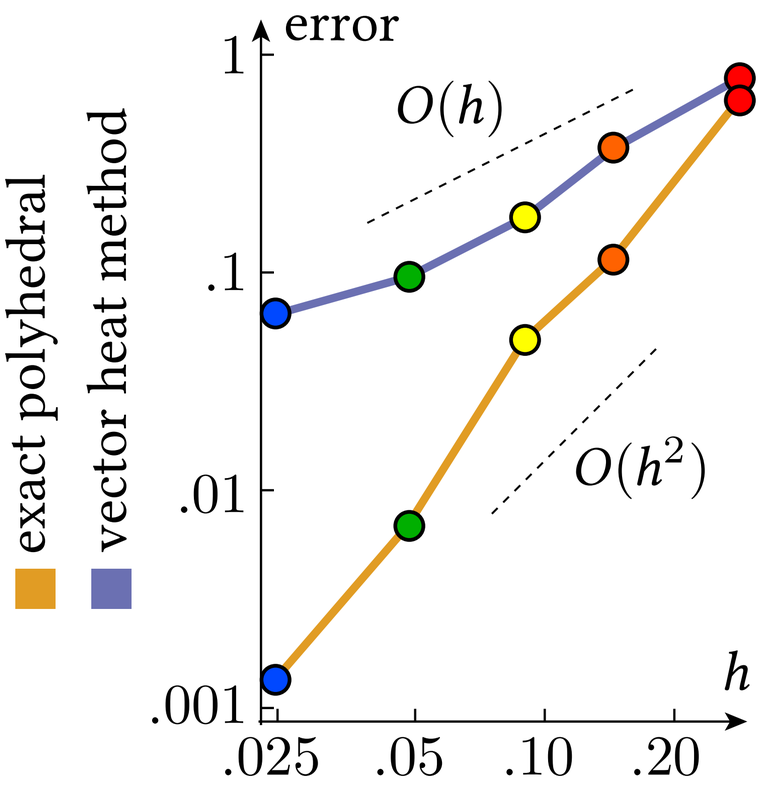


differential *k*-forms

Convergence & Accuracy

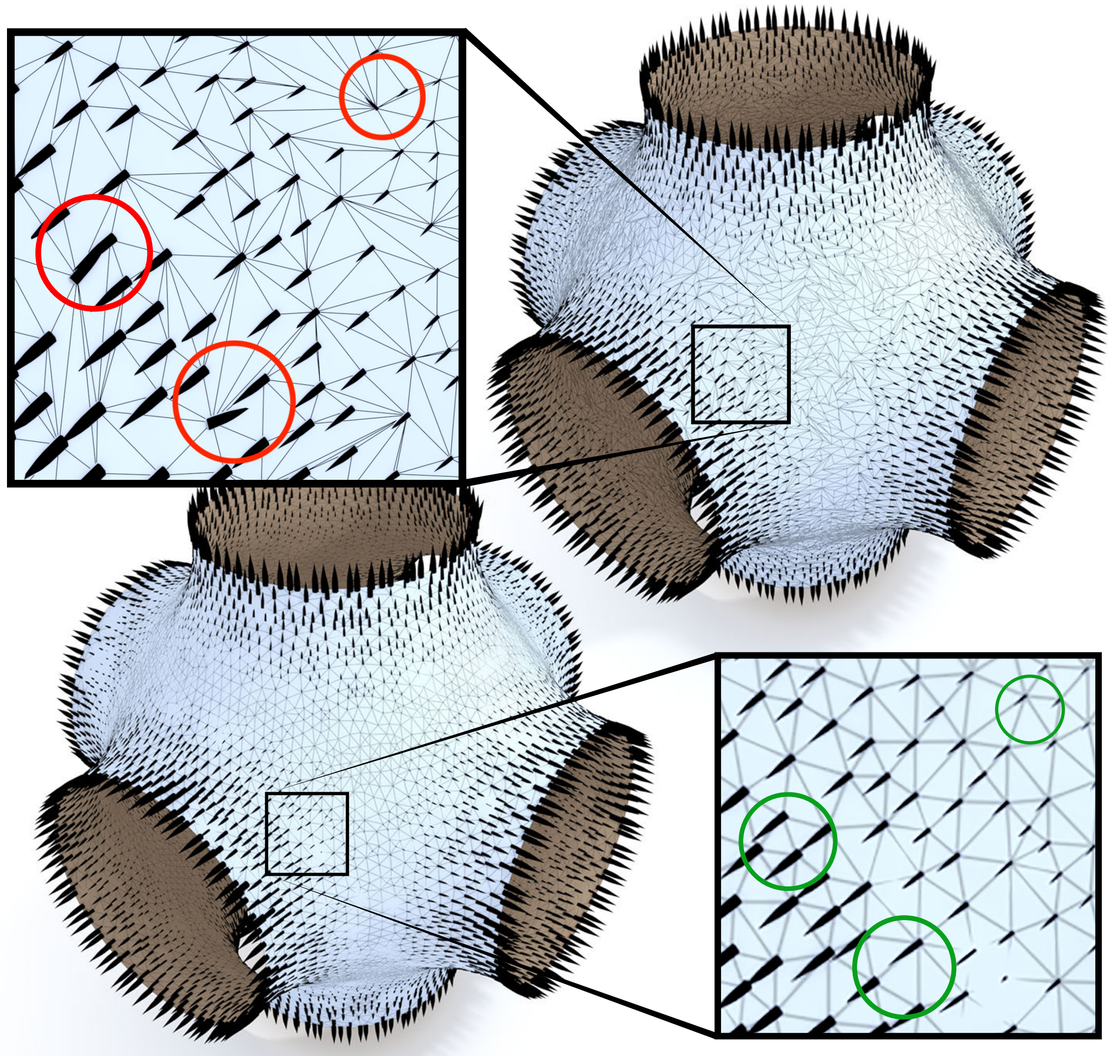
- Even though computing parallel transport "by diffusion," results are very close to exact (***slow***) transport along polyhedral surface
- Like scalar heat method, convergence is (*empirically*) first-order on triangle meshes





Vector Heat Method—Triangulation Quality

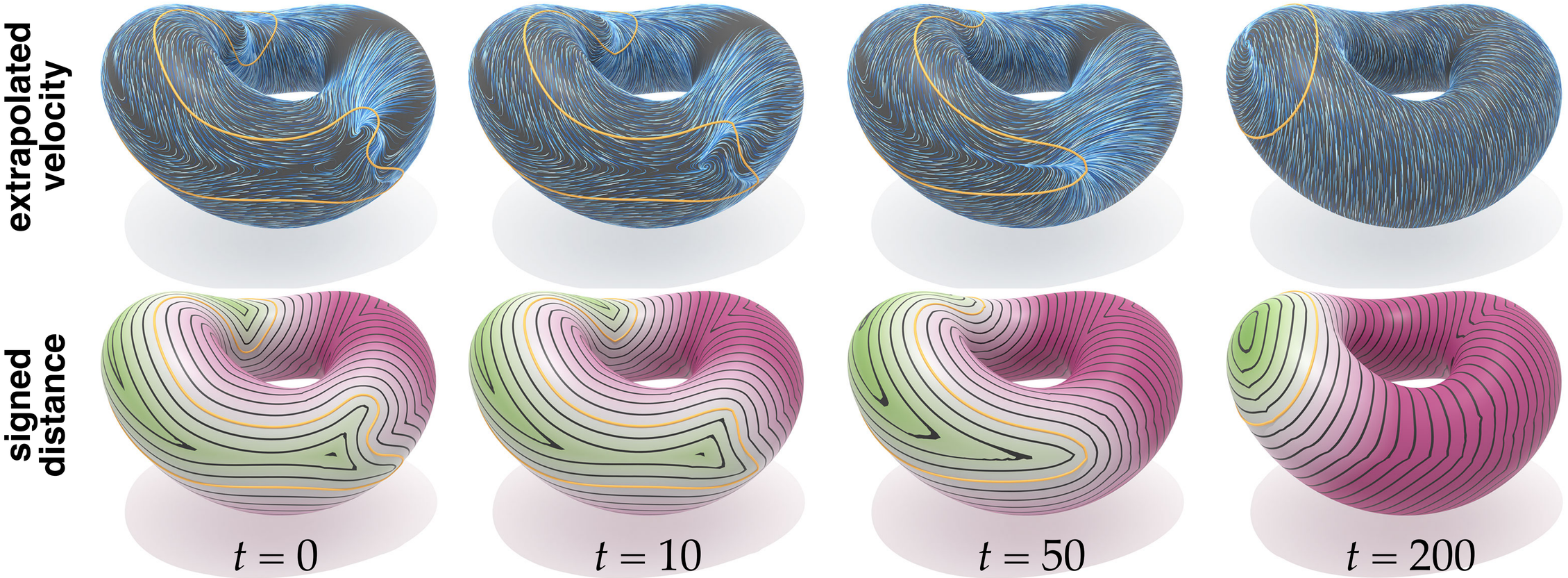
- In simplicial case, accuracy influenced by triangle quality
- In scalar case, intrinsic Delaunay condition ensures maximum principle
- We can **extend the classic Delaunay flip algorithm** to also track tangent coordinate systems
- Ensures that connection Laplacian is positive semidefinite, maximum-like principle (each vector is contained in convex cone of transported neighbors)
- A lot more to say here…



Application—Level Set Velocity Extrapolation

Benefits of vector heat method for level set simulation:

- fast **global** extension of level set velocities (just backsubstitution!)
- minimal geodesic property closely preserves level set *without* redistancing
- works directly on curved domains, general polygonal meshes, point clouds…



Application—Logarithmic Map

- Exponential map traces out a geodesic along a given tangent vector
- Logarithmic map asks opposite question: *which tangent vector reaches a given point?*
- Fast queries of the log map are quite valuable for geometric computation…
- Easily computed via vector heat method:

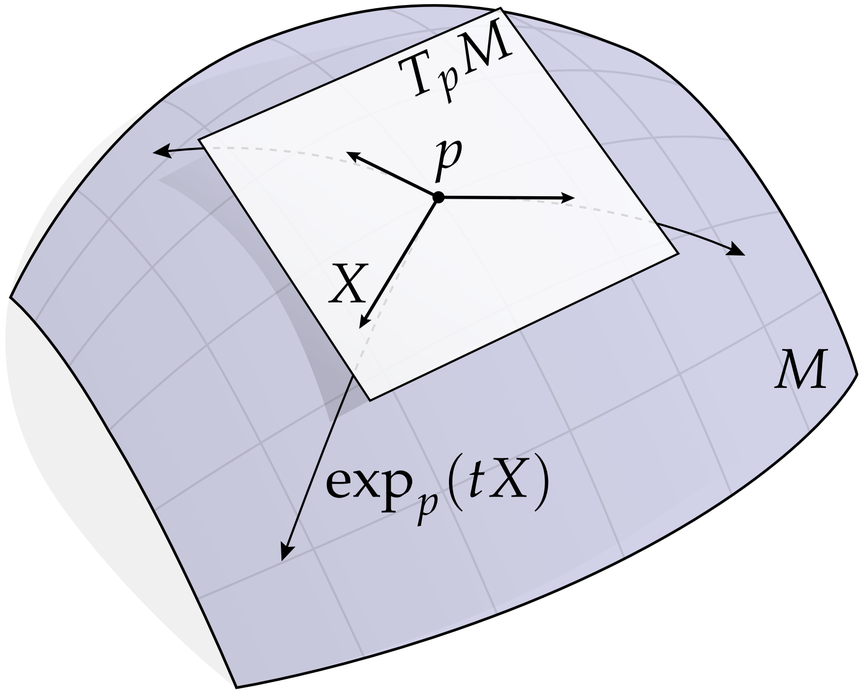
I. extend "horizontal" vector to field H

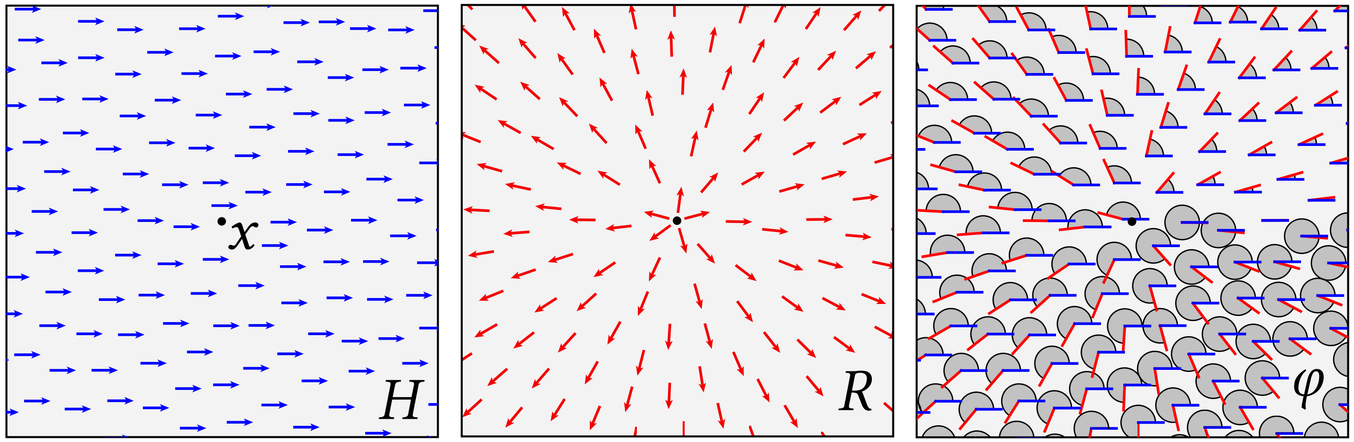
II. extend radial vector field to field R

III. compute angle φ between H and R

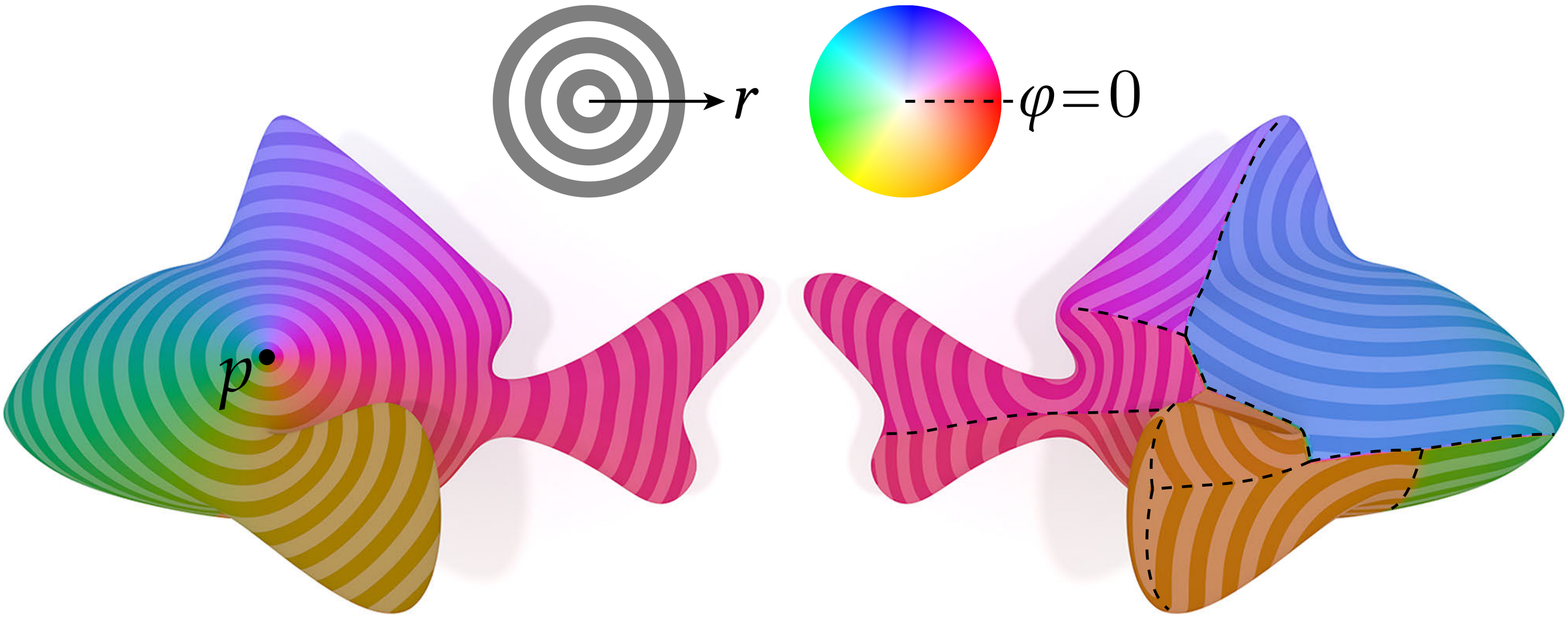
IV. solve Poisson equation for radius r

- Log map is then $(r \cos \varphi, r \sin \varphi)$



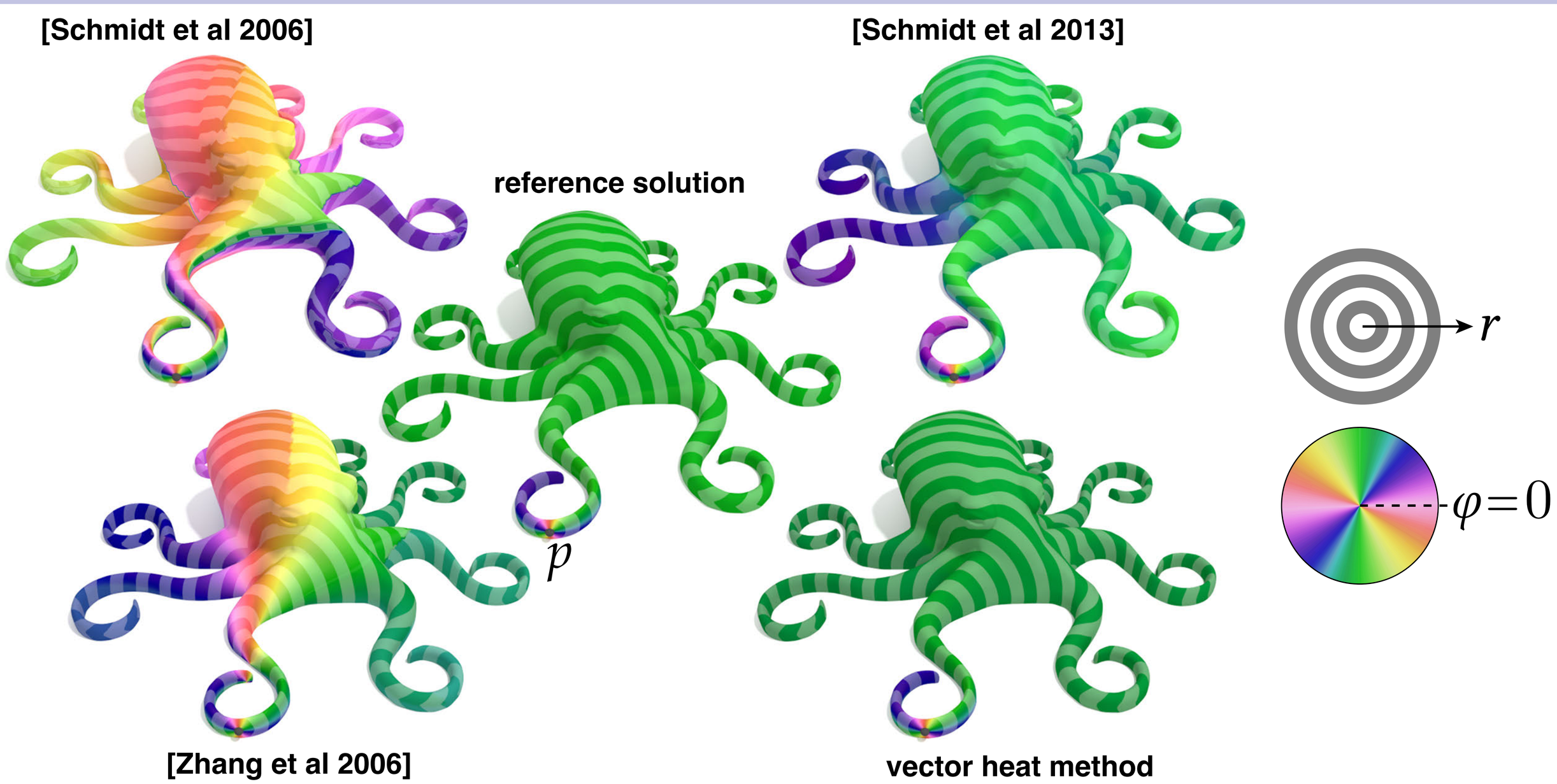


Log Map via Vector Heat Method



Importantly: get accurate log map over *entire* surface

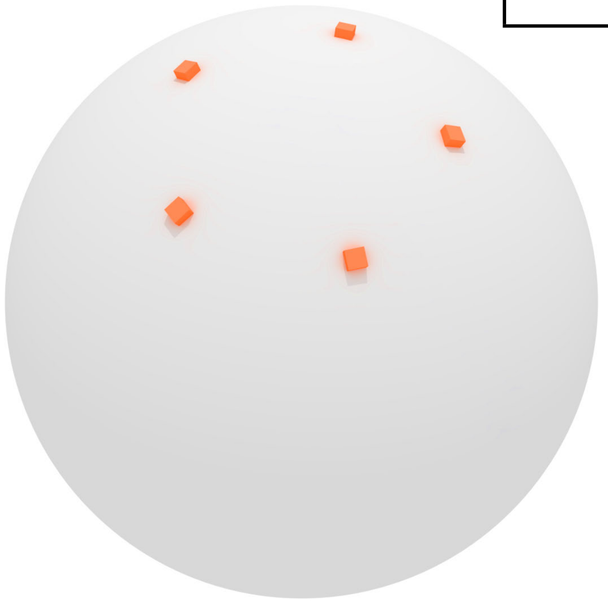
Comparison to Local Approximations

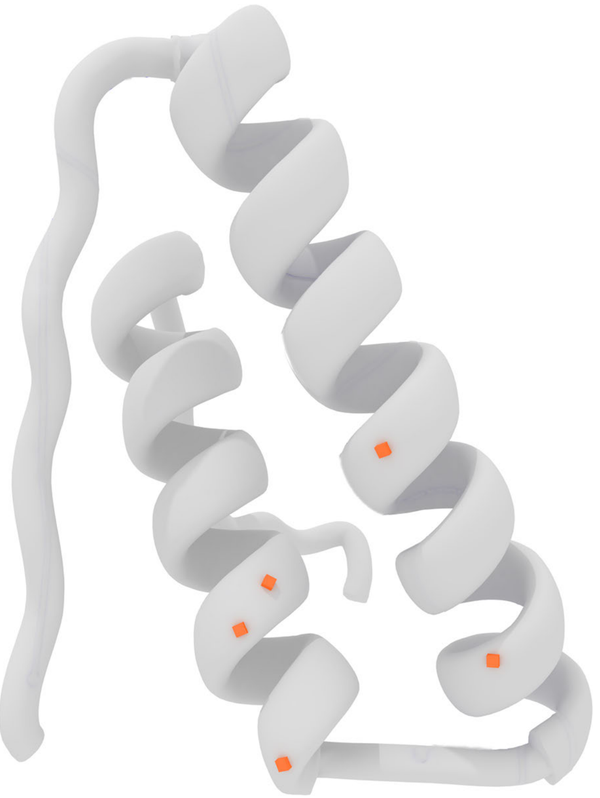


Application—Geometric Centers

- **Motivating question:** what is the "center" of points on a curved manifold?
- Can't just project Euclidean center…
- Instead, define center as a point that minimizes sum of (squared) geodesic distances to point set
  - $p = 2$: *Karcher/Fréchet mean*
  - $p = 1$: *geometric median*
- Naive algorithm: evaluate this sum for every point on the manifold (*expensive!*)
- Can do better using the log map…

$$E(x) = \frac{1}{n} \sum_i d(x, y_i)^p$$



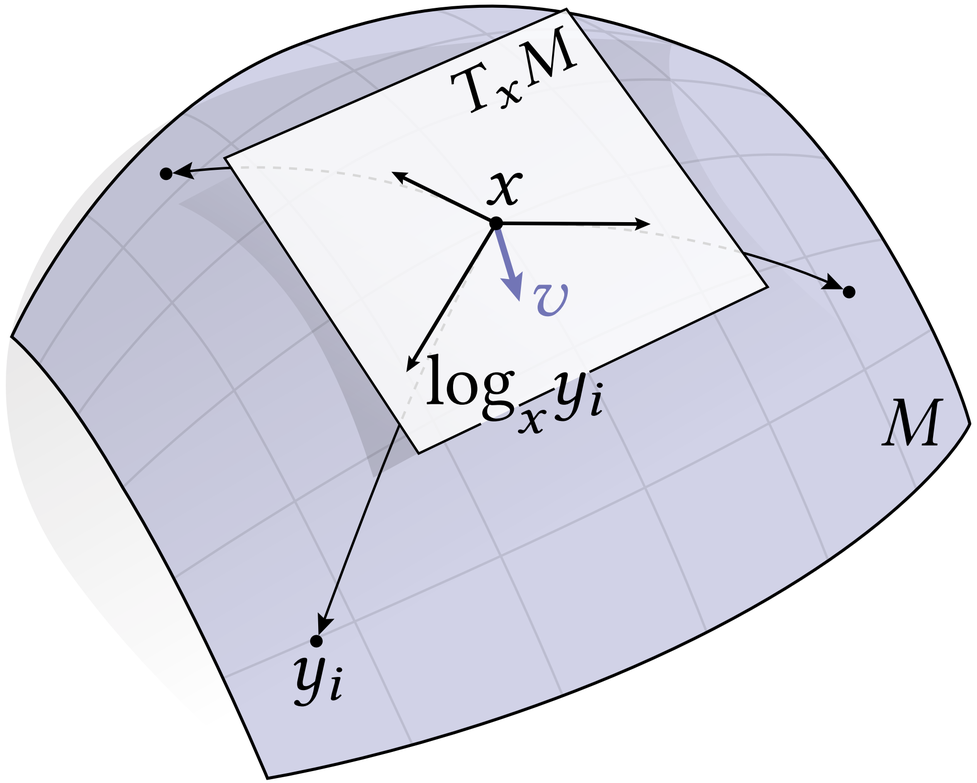


Karcher Mean / Geometric Median via Log Map

- Iterative algorithm:
  - pick a random initial starting point x
  - compute the log v_i of all points y_i
  - compute the mean v of all the v_i
  - if v is nonzero, move x to $\exp_x(v)$ and repeat
- Can trivially modify for $p = 1$ (Weiszfeld 1937)
- Cost? Compute log map once per iteration—*independent of the number of points y_i*
- No more expensive to compute mean of a (probability) distribution on manifold

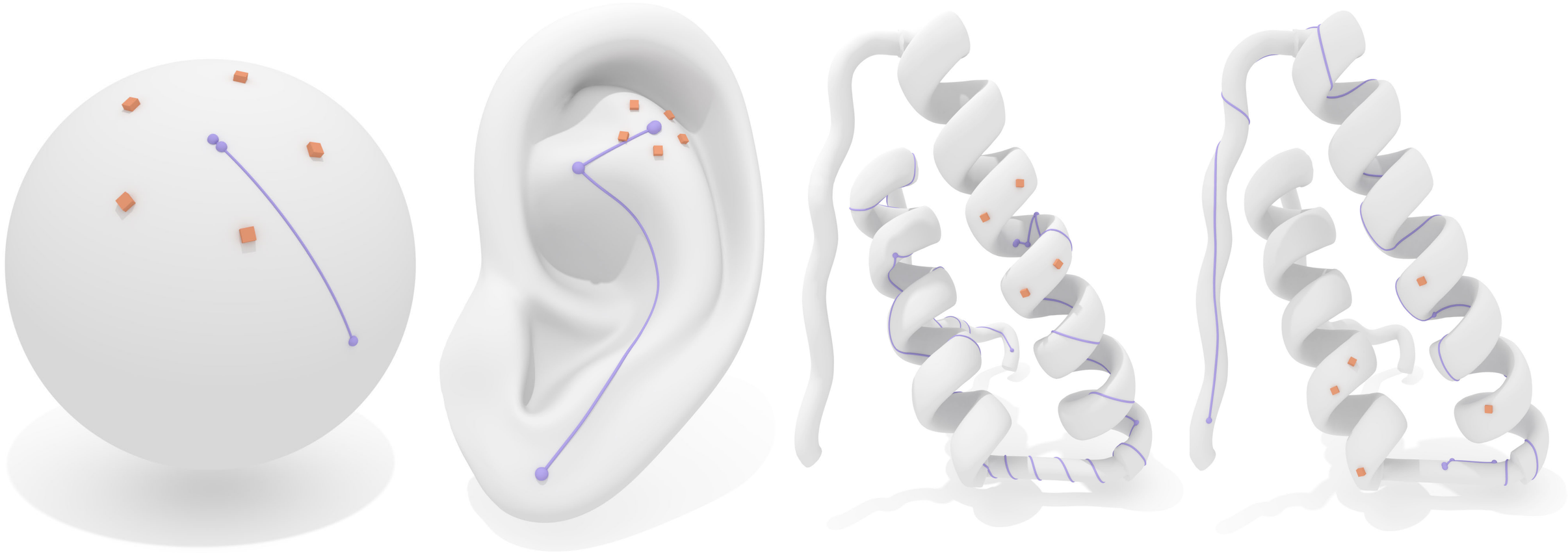
$$v \leftarrow \frac{1}{n}\sum_i \log_x(y_i) \qquad x \leftarrow \exp_x(v)$$

$$E(x) = \frac{1}{n}\sum_i d(x, y_i)^p$$



Karcher Mean via Vector Heat Method

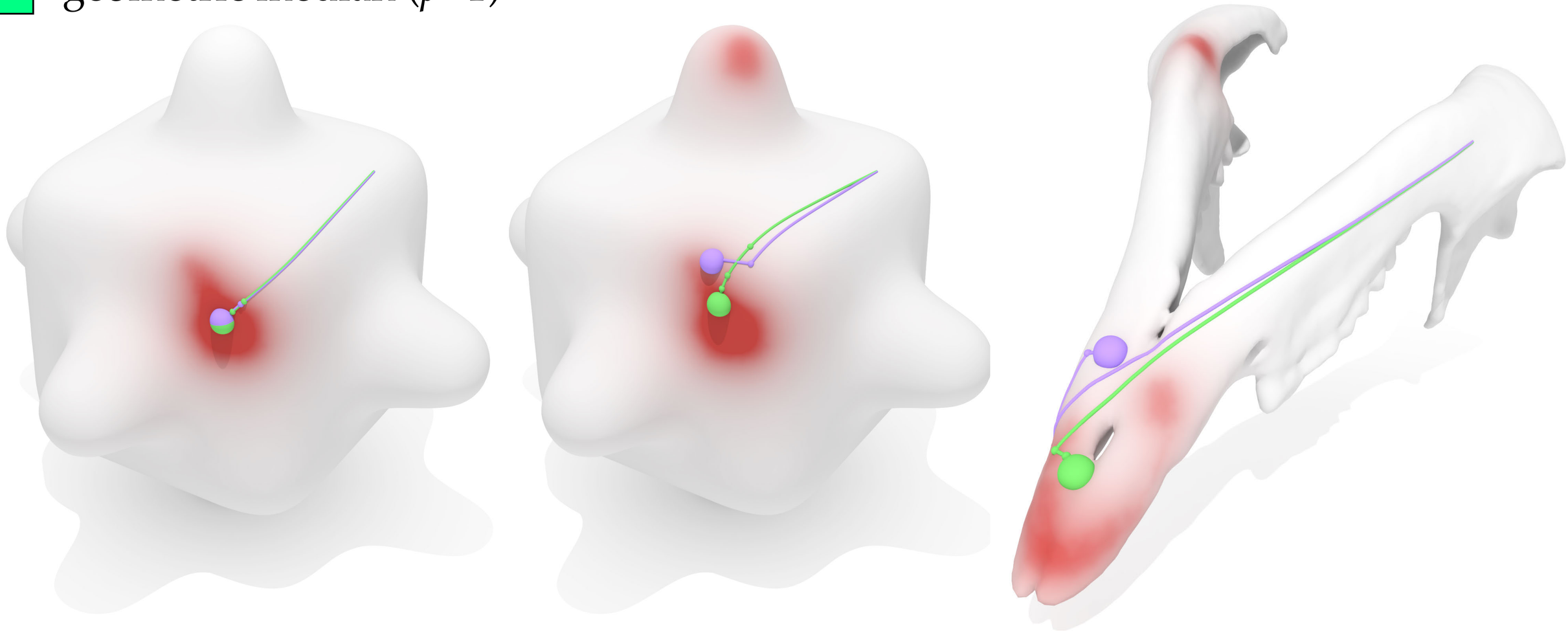
Converges in ~10 iterations, even on complex shapes:



Geometric Centers of Distributions

Karcher mean (p=2)

geometric median (p=1)



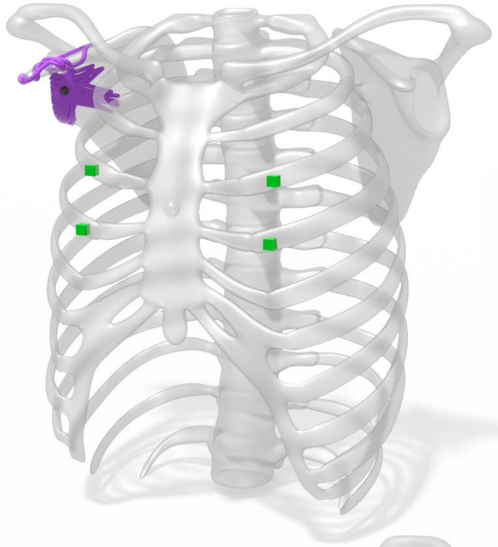
Karcher Mean—Heat Method vs. Local Methods

■ **site**

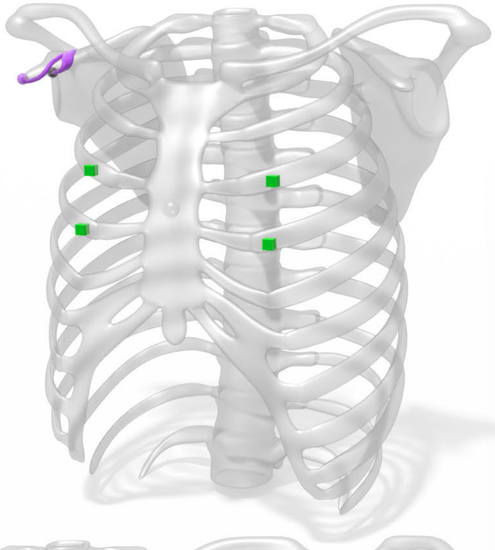
● **start**

● **end**

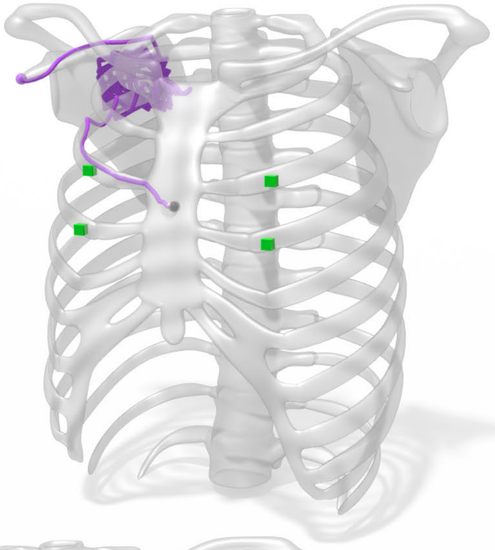
[Zhang et al 2006]
(timeout after 5 minutes)



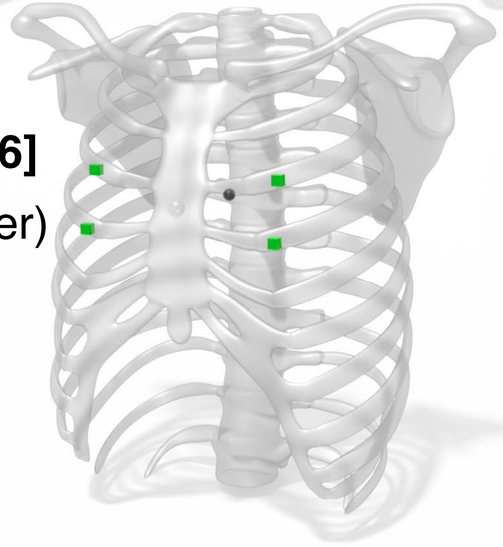
[Schmidt et al 2006]
(timeout after 5 minutes)

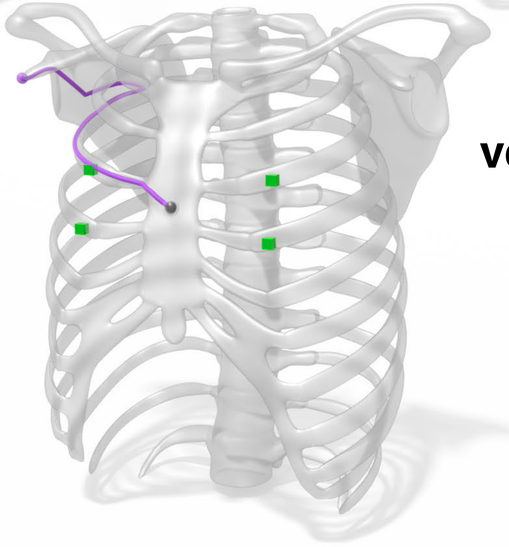


[Schmidt et al 2013]
(57.2s)



[Panozzo et al 2006]
(217.8s, wrong center)



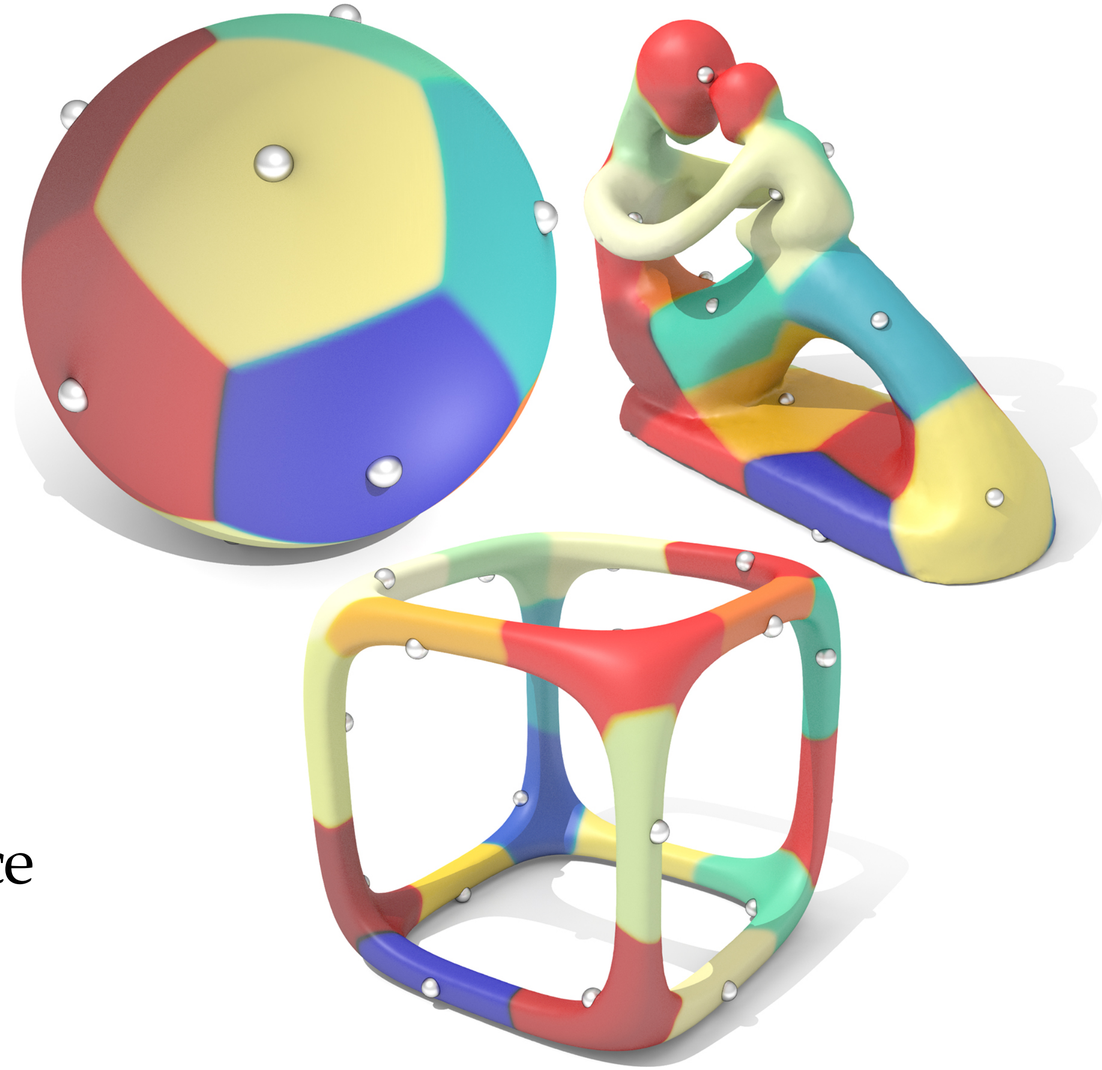


vector heat method
(7.8s)

Centering demands
globally accurate log map!

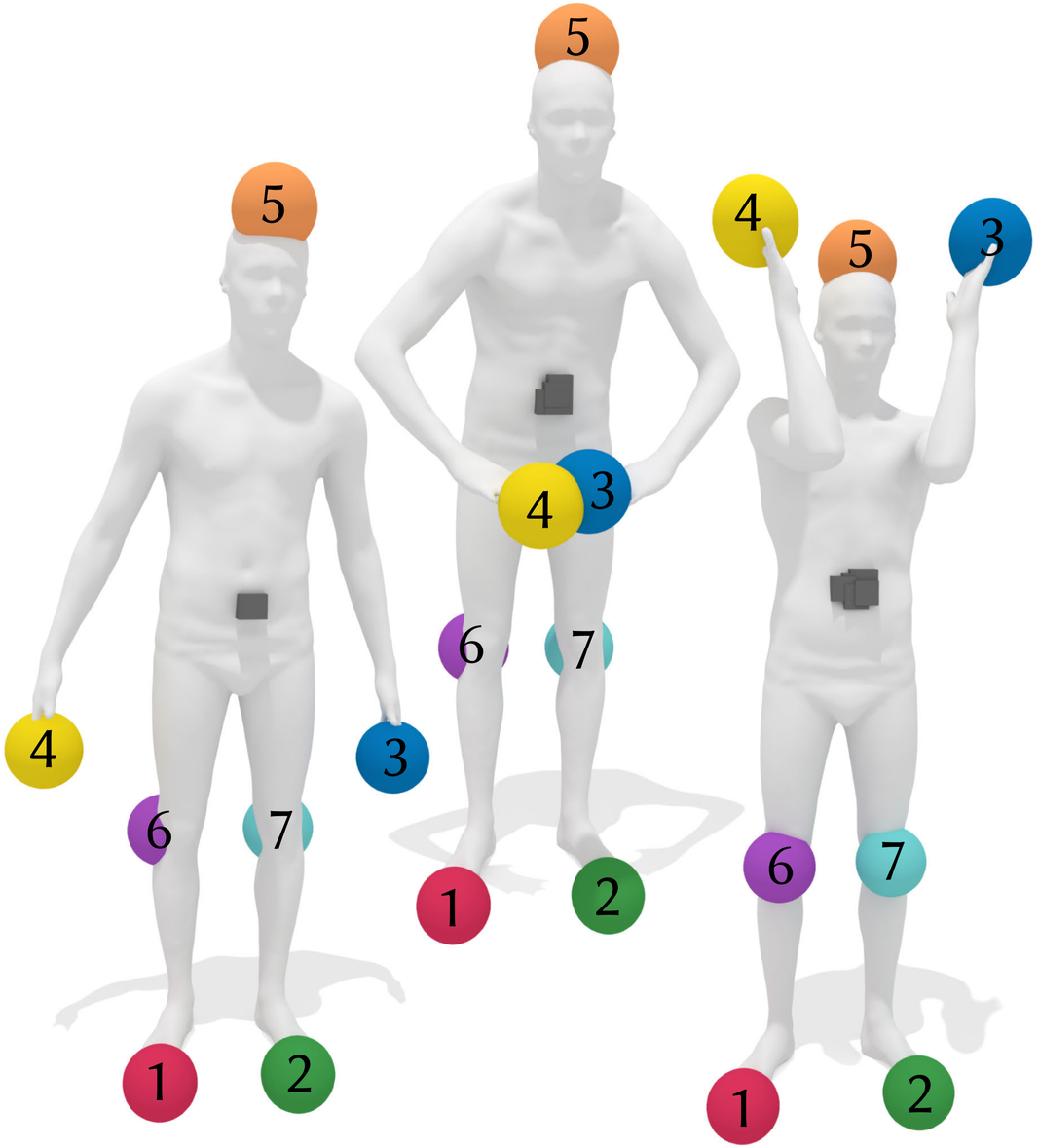
Application—Centroidal Voronoi Diagrams

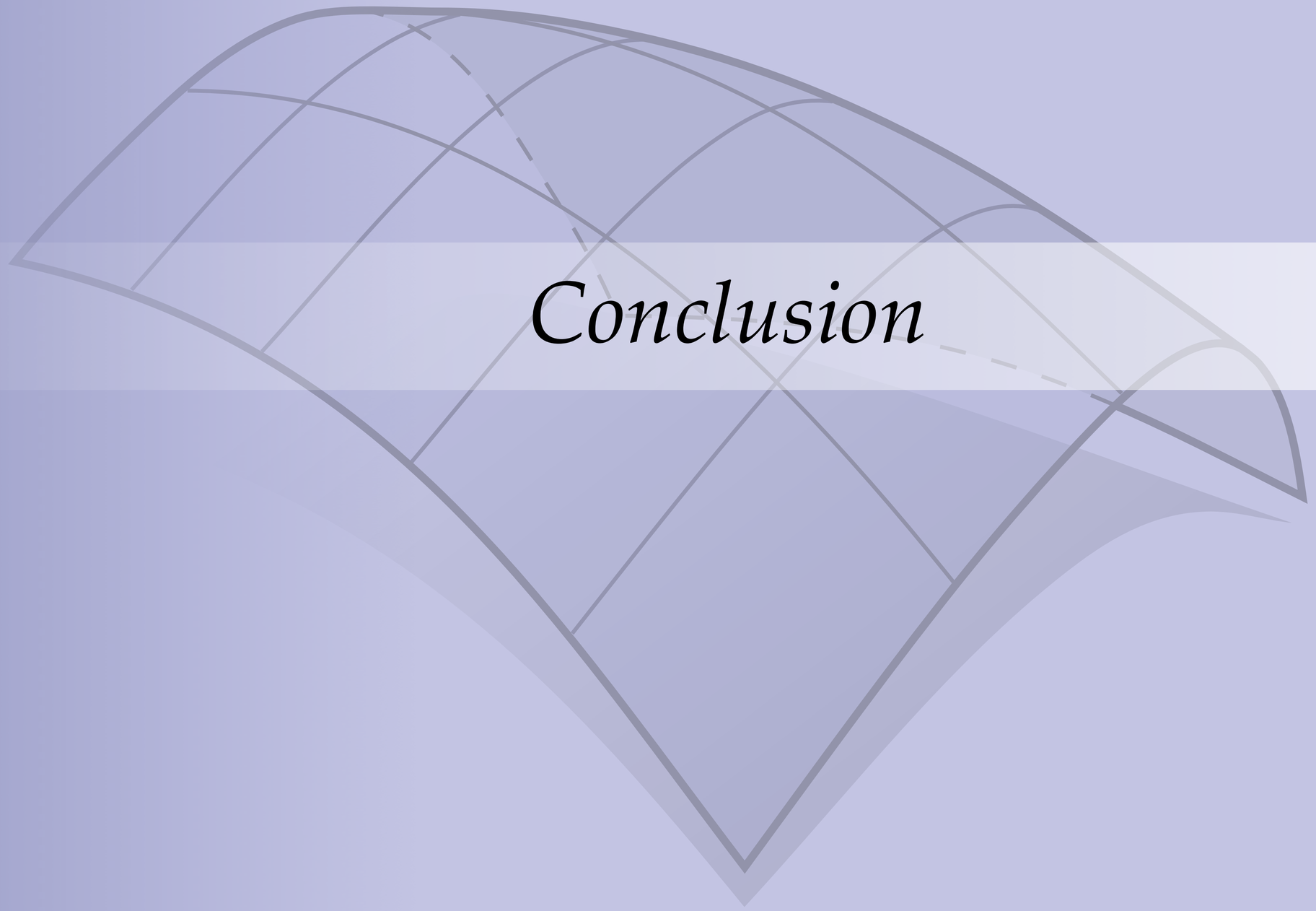
- Voronoi diagram partitions surface into regions closest to a set of sites; useful for clustering, approximation, geometric queries, …
- Centroidal Voronoi diagram moves cites to cell centroids ⇒ better shaped cells
- Much more complicated on surfaces than in R^n!
  - cells may not be disk-like
  - cells may not contain their centroid
  - points that are close in R^n may be far on surface
- Can use Karcher mean to efficiently compute correct GCVT (even for *large* cells)



Application—Ordered Intrinsic Landmarks

- Shape matching algorithms require not only landmarks, but *ordering* of landmarks
- What are canonical points that depend only on the geometry (no auxiliary data, priors, etc.)?
- One idea:
  - compute geometric median of the *entire surface* (constant distribution)
  - iteratively compute furthest point
- No need to solve hard matching problems…

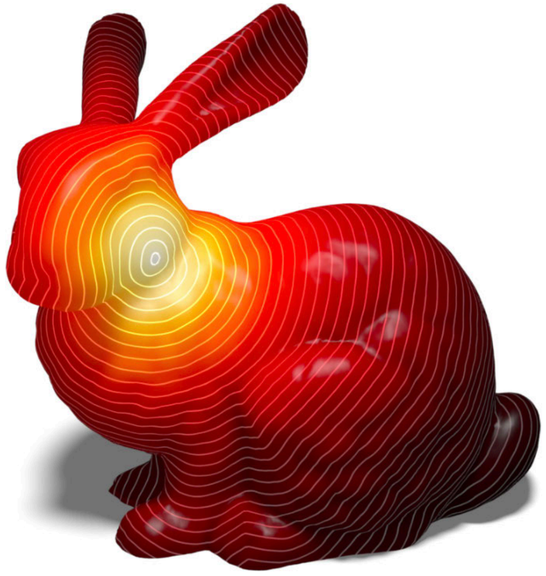


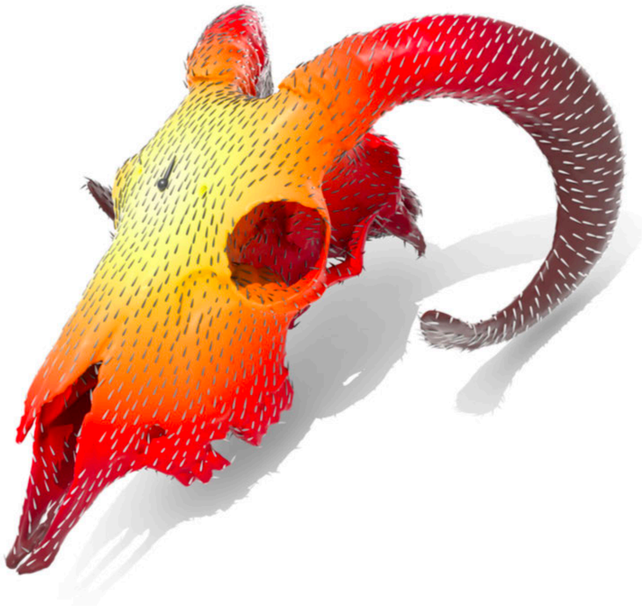


Conclusion

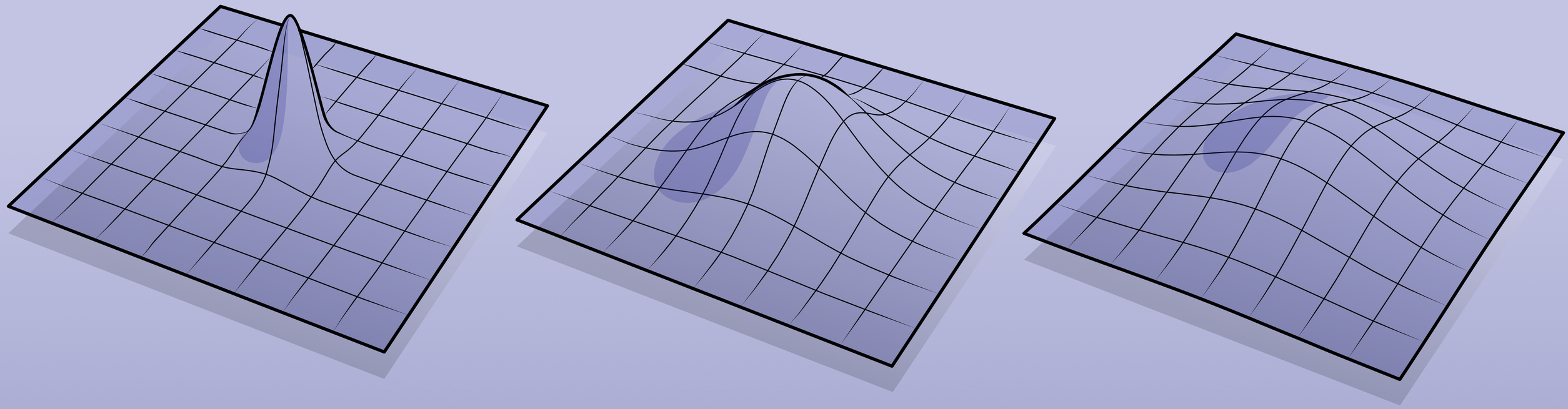
Heat Methods—Summary

- **Key idea:** Reducing computation of geodesic distance + parallel transport to standard *linear* PDEs opens doors to new approaches to geometric algorithms.
- Immediately benefit from any progress on numerical linear algebra/fast linear solvers, discretization of operators…
- Heat method is not just one particular discrete algorithm (*e.g.*, for triangle meshes), but rather a general **principle** that can be applied, discretized & solved in *many* different ways.
- Already fairly mature:
  - many discretizations, improved accuracy, …
  - fast/scalable solvers, widely available code, …
- **Future questions:** hybrid w/ fast marching? anisotropy? other elliptic operators? convergence? high dimensions? higher-order statistics? machine learning? … ?





Thanks!



Heat Methods in Geometry Processing

Keenan Crane • IPAM Workshop on Geometric Processing • April 4, 2019