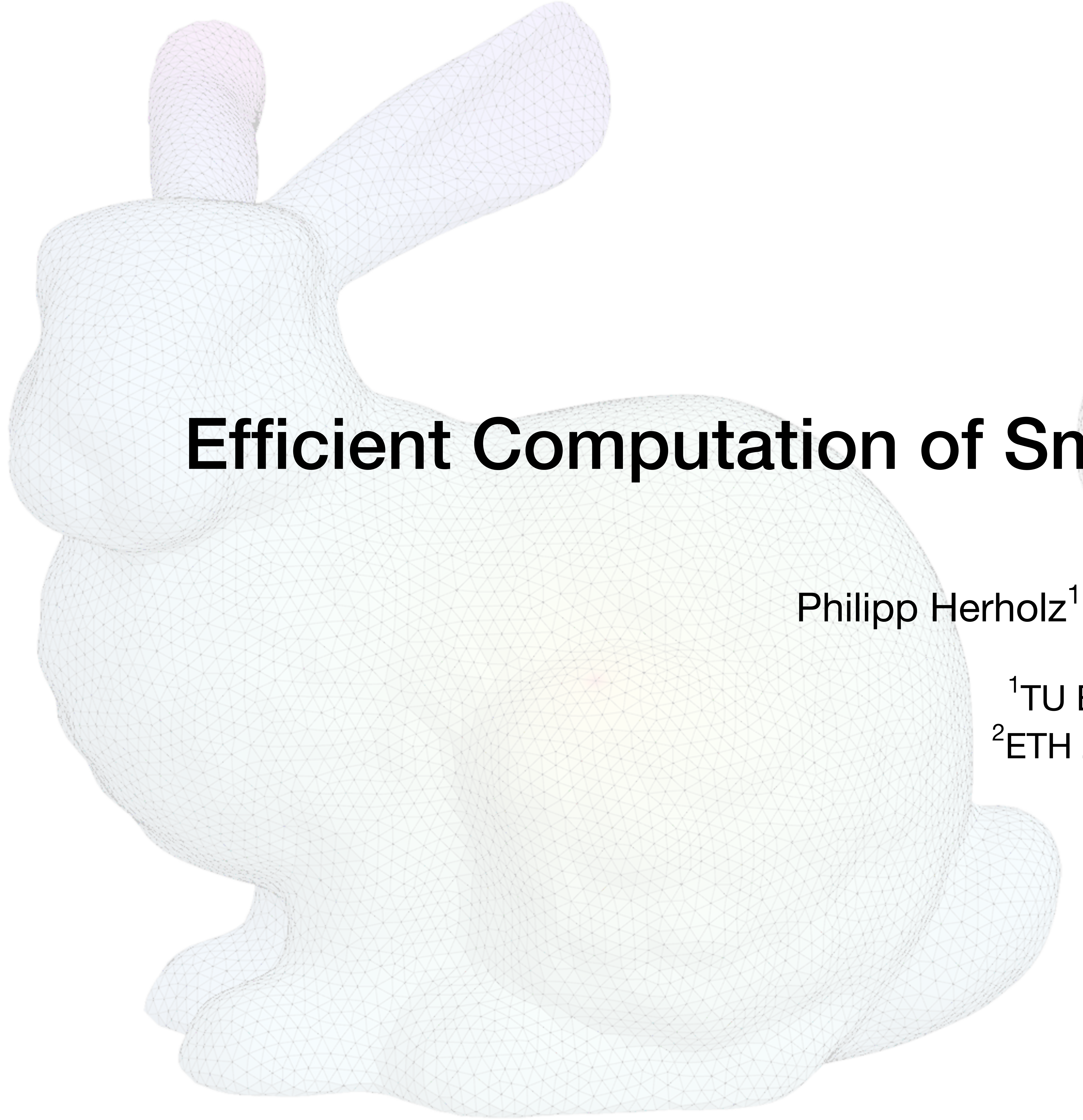
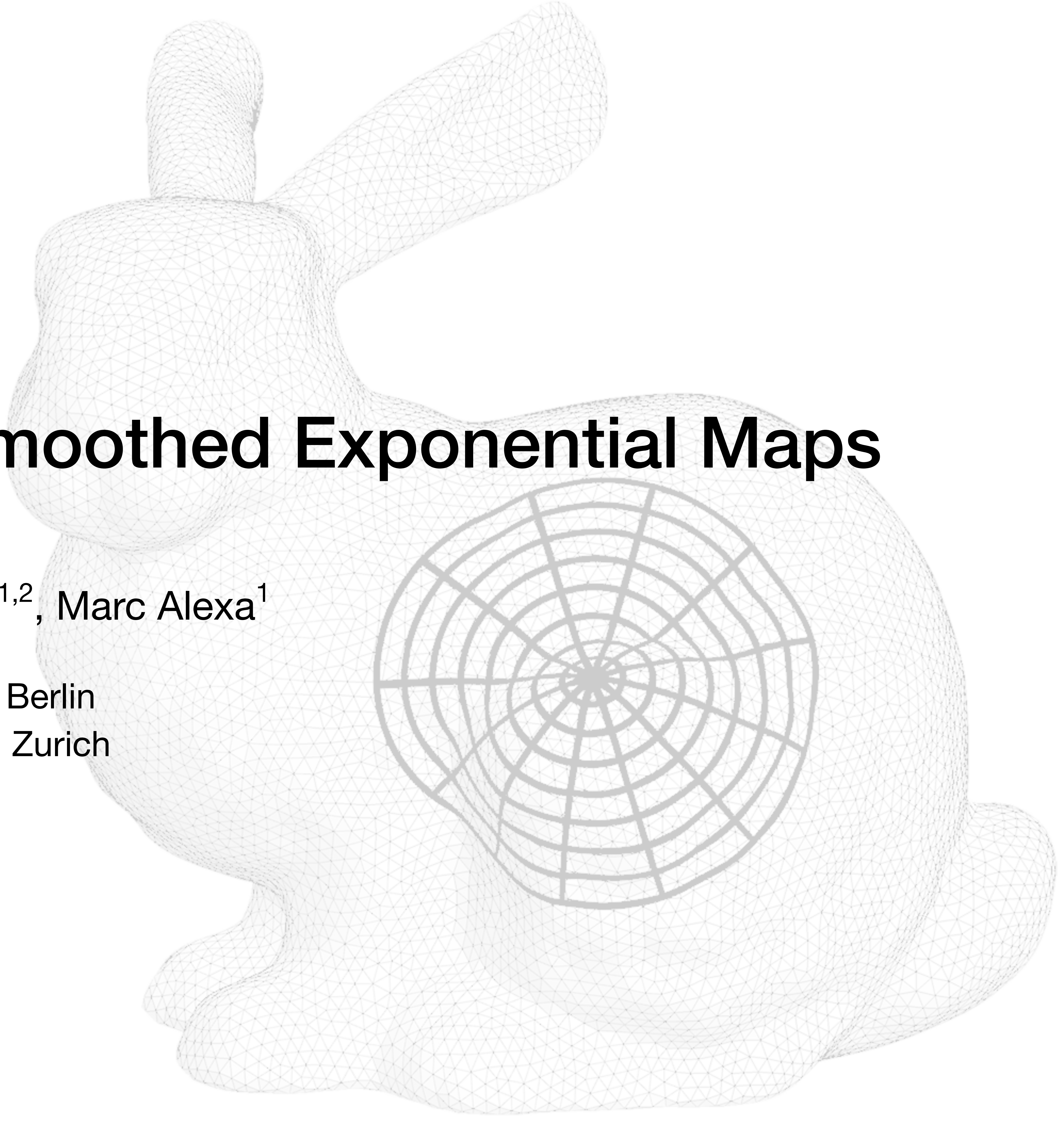
Efficient Computation of Smoothed Exponential Maps

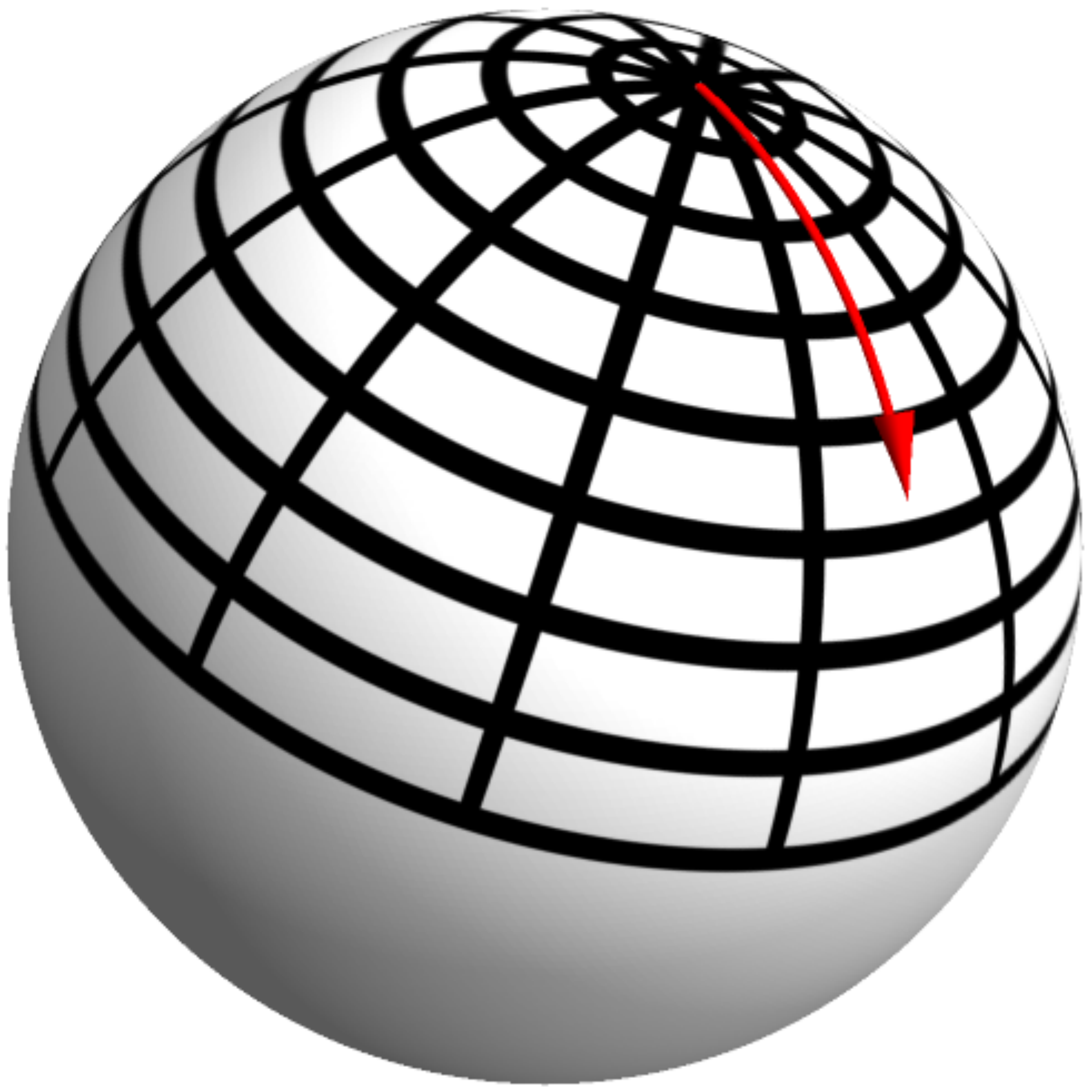


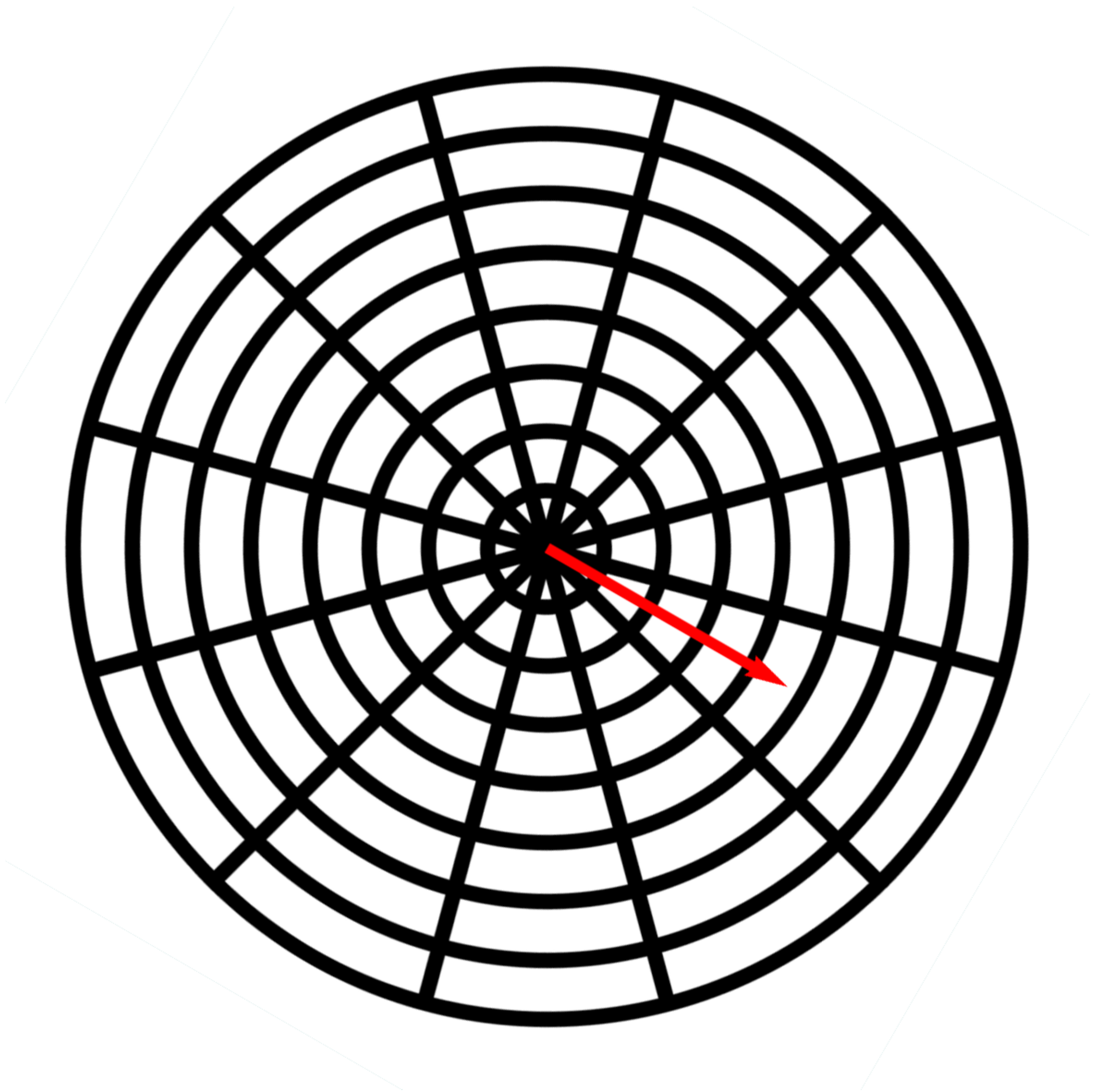


Philipp Herholz[1,2], Marc Alexa[1]

[1]TU Berlin
[2]ETH Zurich

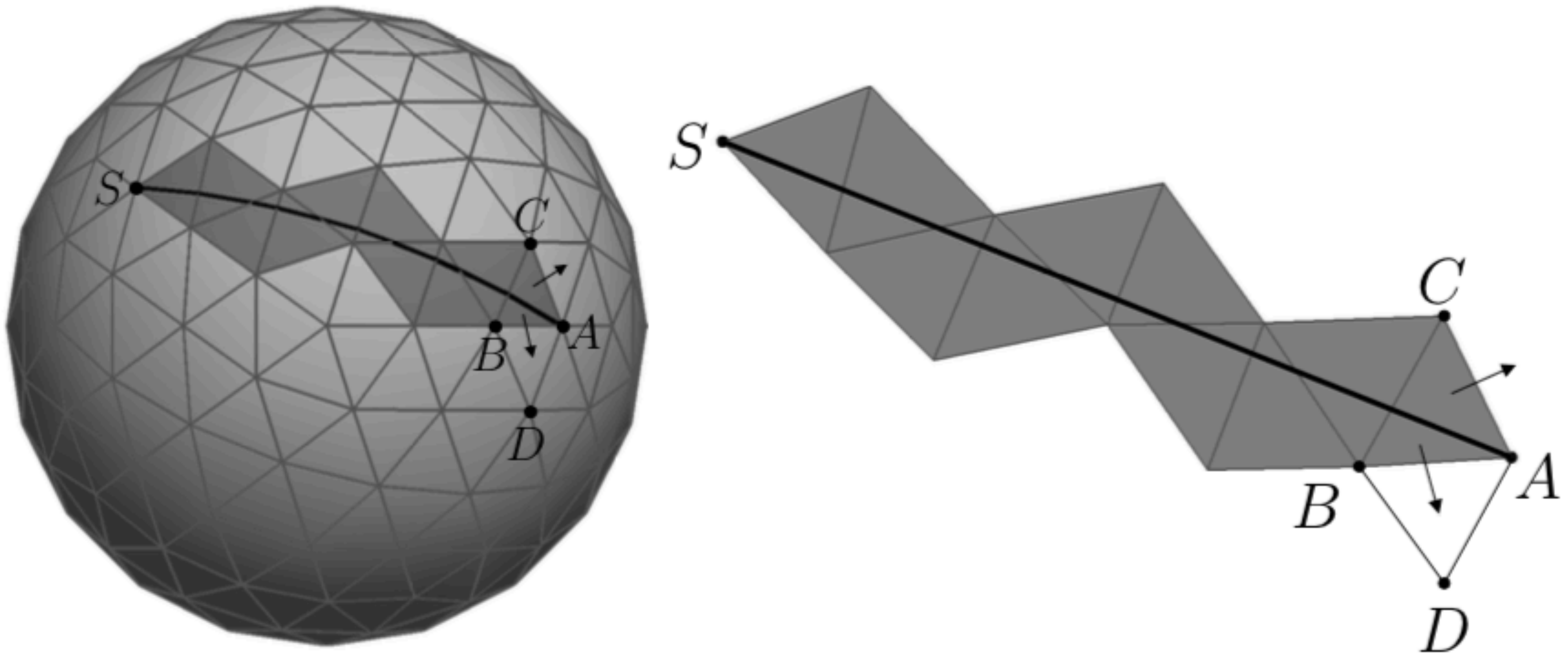
Exponential Maps





Previous Approaches

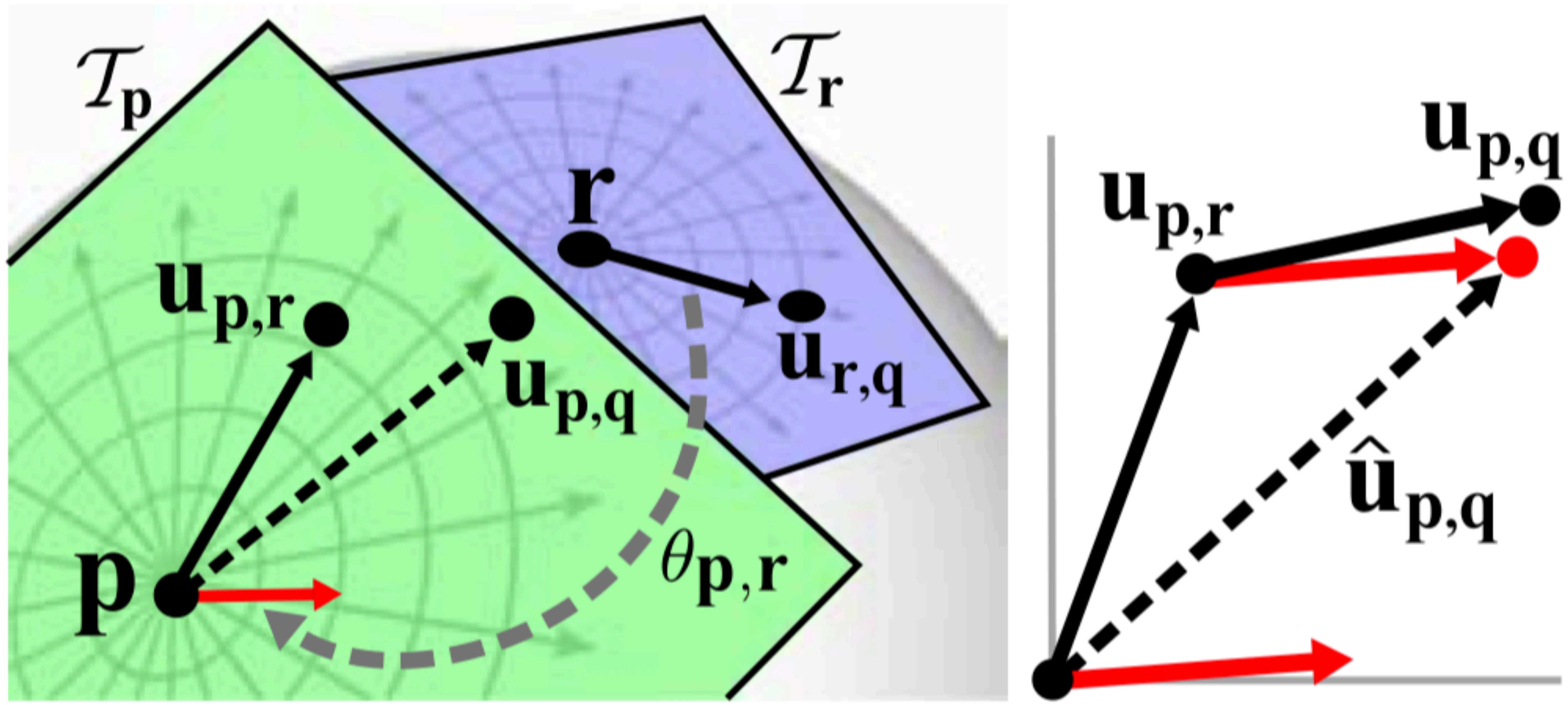
Exact polyhedral geodesics: Accurate but slow.



eg.: [Wang et al., 2017] [Qin et al. 2016] [Ying et al., 2013]
[Xin et al., 2009] [Surazhsky et al., 2005]

Previous Approaches

Dijkstra based propagation: Inaccurate but fast.



eg.: [Melvær et al., 2012] [Schmidt et al., 2006]

Previous Approaches

Dijkstra based propagation: Inaccurate but fast.

eg.: [Melvær et al., 2012] [Schmidt et al., 2006]

Previous Work: Geodesics in Heat

Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow

KEENAN CRANE
Caltech
CLARISSE WEISCHEDEL, MAX WARDETZKY
University of Göttingen

We introduce the *heat method* for computing the geodesic distance to a specified subset (*e.g.*, point or curve) of a given domain. The heat method is robust, efficient, and simple to implement since it is based on solving a pair of standard linear elliptic problems. The resulting systems can be prefactored once and subsequently solved in near-linear time. In practice, distance is updated an order of magnitude faster than with state-of-the-art methods, while maintaining a comparable level of accuracy. The method requires only standard differential operators and can hence be applied on a wide variety of domains (grids, triangle meshes, point clouds, *etc.*). We provide numerical evidence that the method converges to the exact distance in the limit of refinement; we also explore smoothed approximations of distance suitable for applications where greater regularity is required.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms: Algorithms

Additional Key Words and Phrases: digital geometry processing, discrete differential geometry, geodesic distance, distance transform, heat kernel

ACM Reference Format:

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph.* 28, 4, Article 106 (September 2009), 10 pages.
DOI:http://dx.doi.org/10.1145/XXXXXXX.YYYYYYY

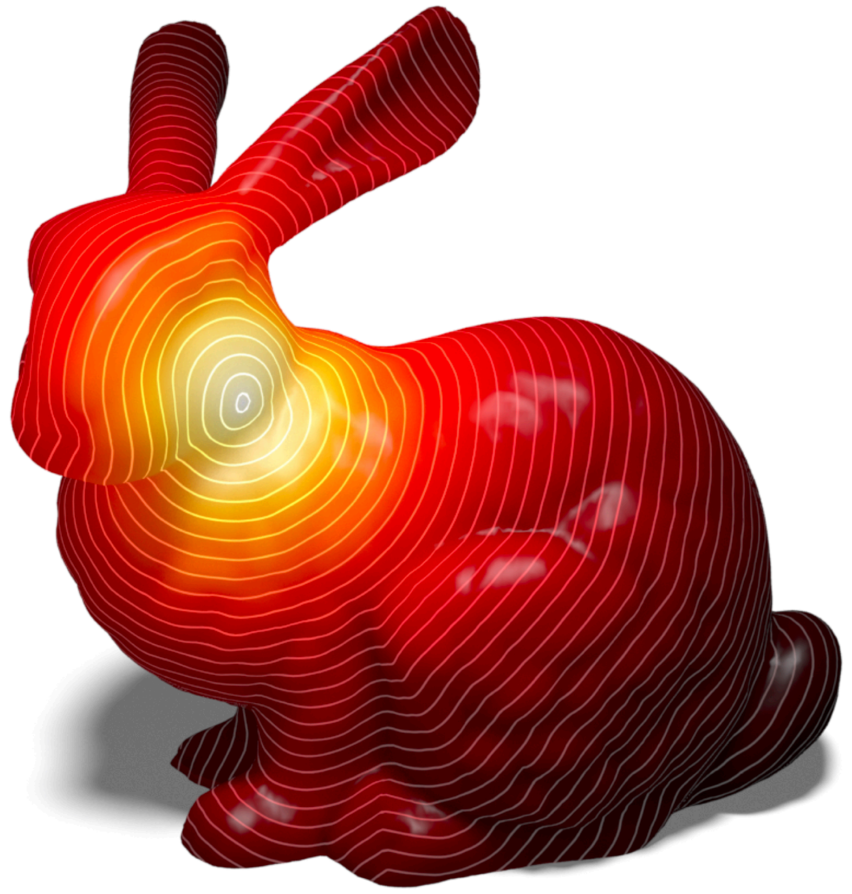


Fig. 1. Geodesic distance from a single point on a surface. The heat method allows distance to be rapidly updated for new source points or curves.
Bunny mesh courtesy Stanford Computer Graphics Laboratory.

1. INTRODUCTION

Imagine touching a scorching hot needle to a single point on a surface. Over time heat spreads out over the rest of the domain and can be described by a function $k_{t,x}(y)$ called the *heat kernel*, which measures the heat transferred from a source x to a destination y

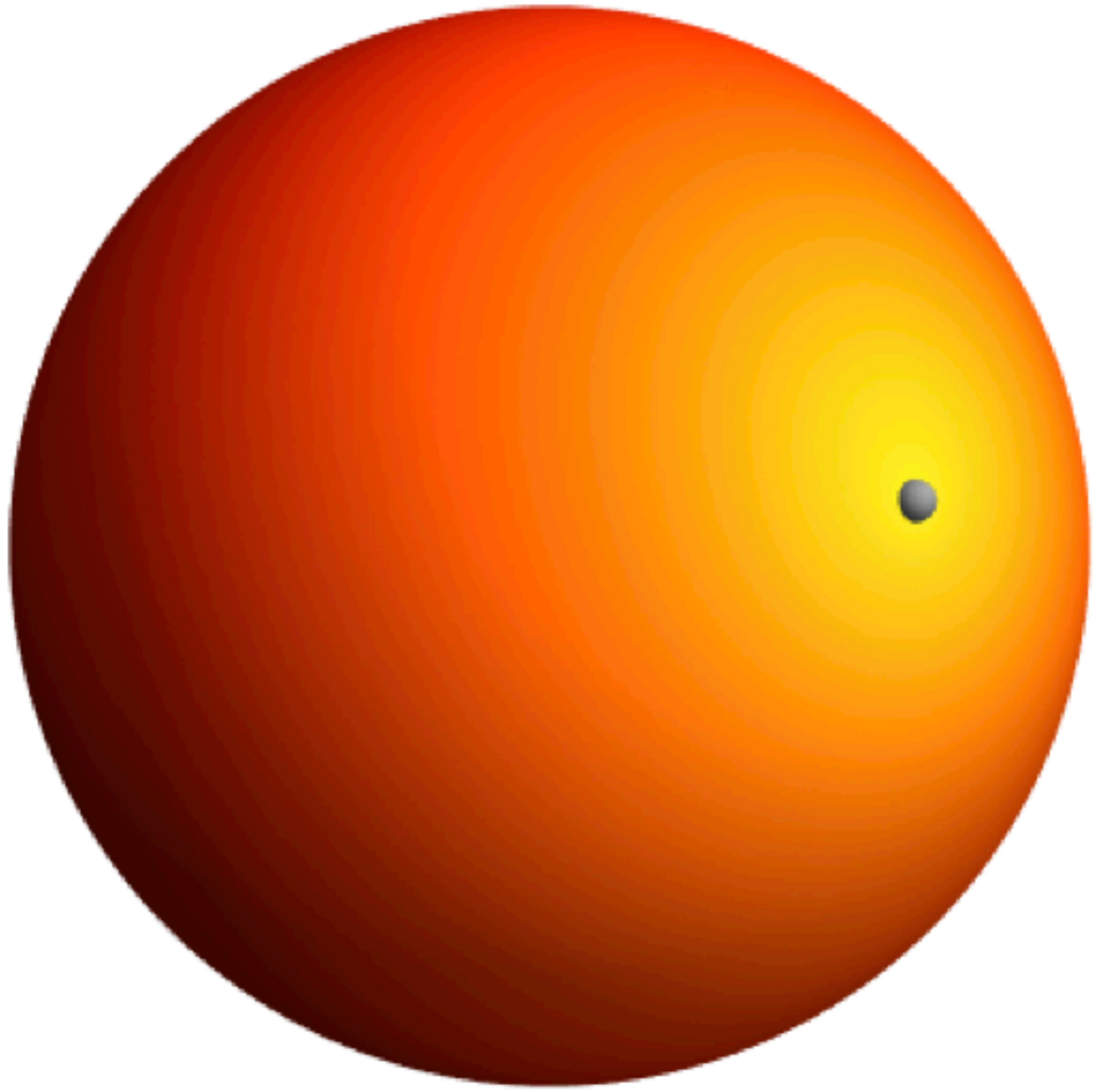
is Varadhan's formula [1967], which says that the geodesic distance ϕ between any pair of points x, y on a Riemannian manifold can be recovered via a simple pointwise transformation of the heat kernel:

$$\phi(x,y) = \lim_{t \to 0} \sqrt{-4t \log k_{t,x}(y)}. \tag{1}$$

The intuition behind this behavior stems from the fact that heat diffusion can be modeled as a large collection of hot particles taking random walks starting at x: any particle that reaches a distant point y after a small time t has had little time to deviate from the shortest possible path. To date, however, this relationship has not been exploited by numerical algorithms that compute geodesic distance.

Why has Varadhan's formula been overlooked in this context? The main reason, perhaps, is that it requires a precise numerical reconstruction of the heat kernel, which is difficult to obtain – applying the formula to a mere approximation of $k_{t,x}$ does not yield

Previous Work: Geodesics in Heat



$$(\mathbf{M} - t\mathbf{L})\mathbf{h} = \mathbf{e}_i$$

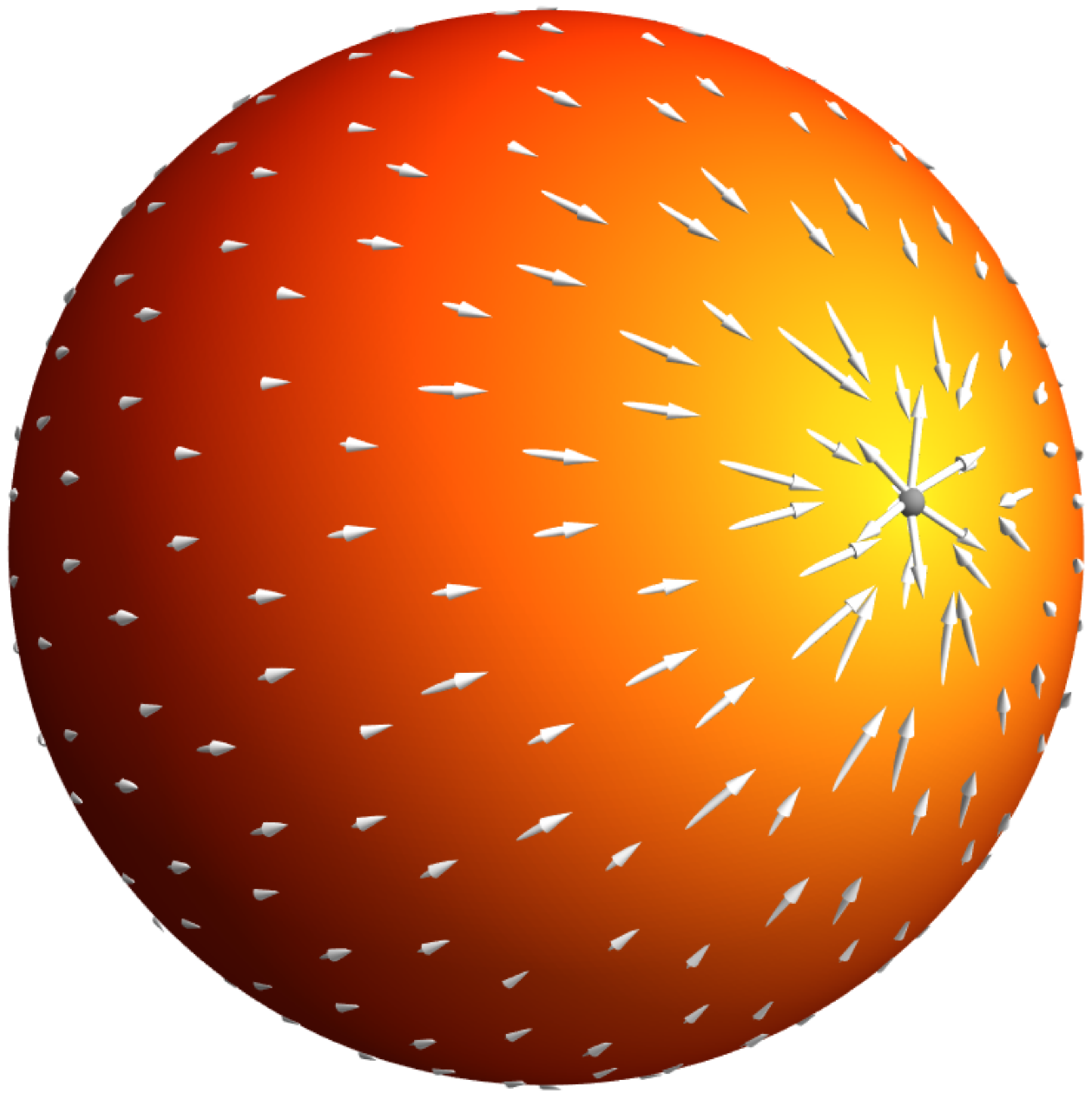
$\mathbf{L}$: cotan Laplacian

$\mathbf{M}$: mass matrix

$\mathbf{e}_i$: unit vector of source vertex

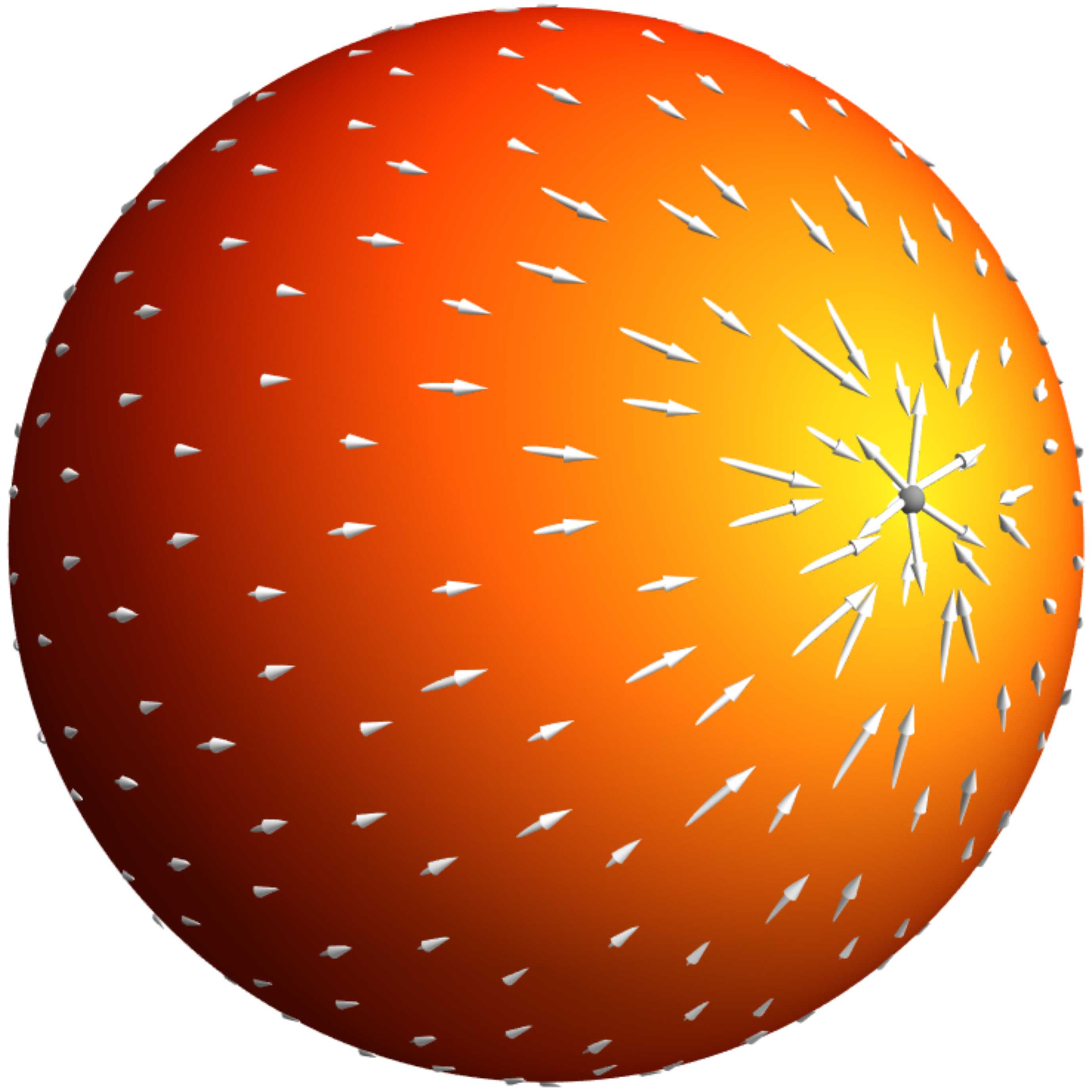
$\mathbf{h}$: “heat values” (color coded)

Previous Work: Geodesics in Heat



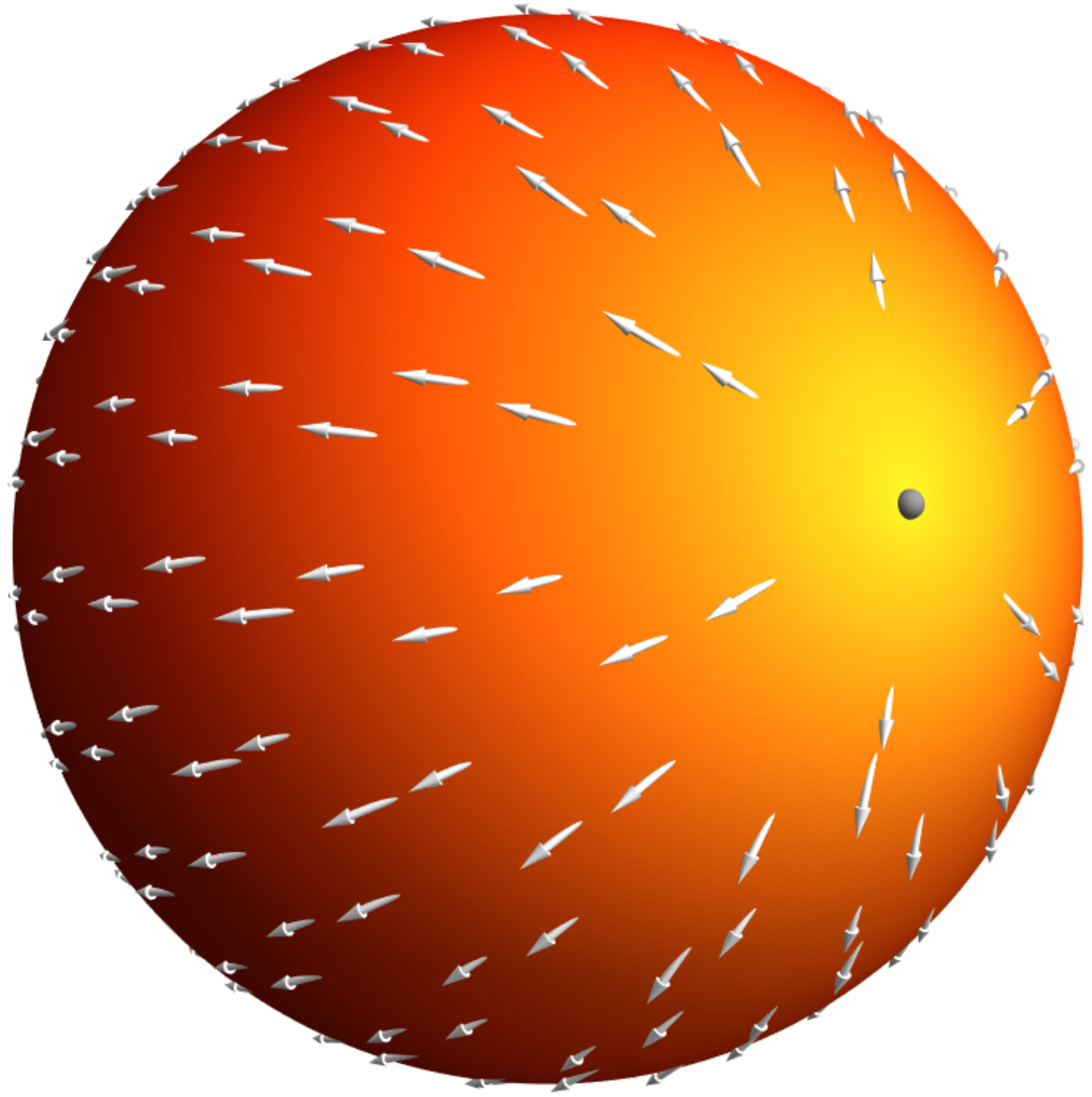
- Compute per triangle gradient of heat values.

Previous Work: Geodesics in Heat



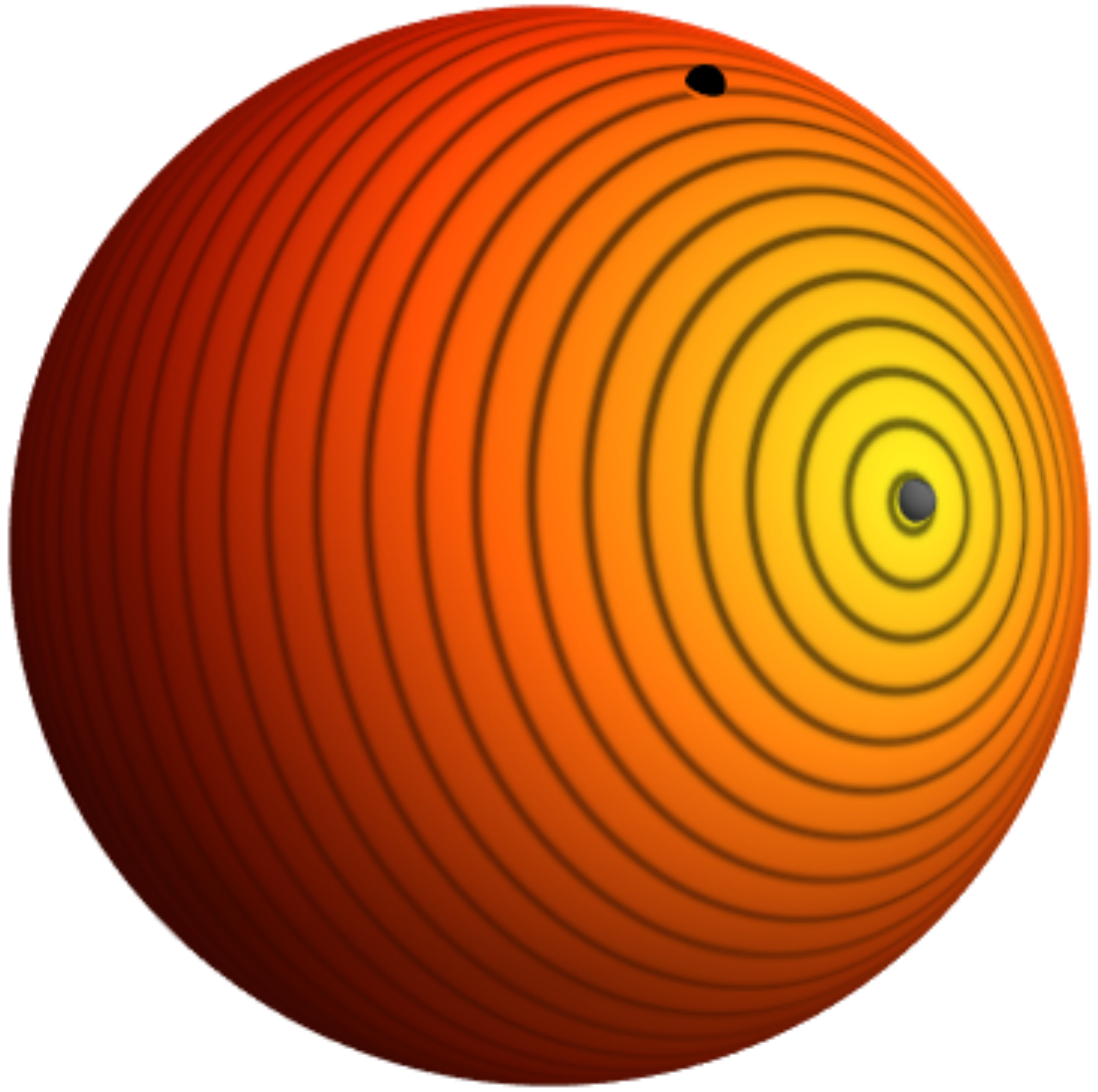
- Compute per triangle gradient of heat values.
- *Observation*: Gradient directions match those of geodesic distance quite exactly.

Previous Work: Geodesics in Heat



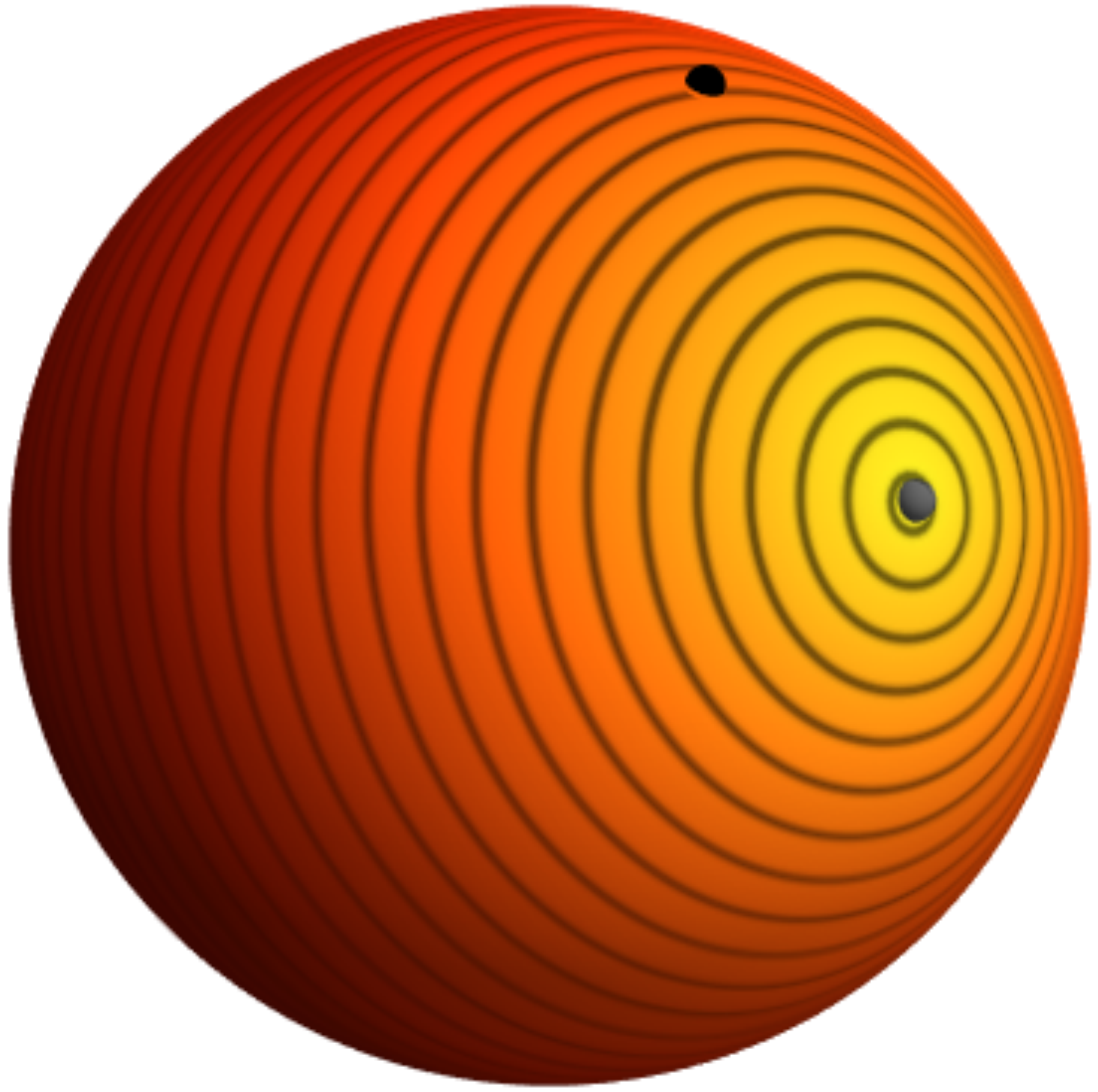
- Compute per triangle gradient of heat values.
- *Observation*: Gradient directions match those of geodesic distance quite exactly.
- *Idea*: Normalize and invert gradients, then integrate their divergence (solve a Poisson problem).

Previous Work: Geodesics in Heat



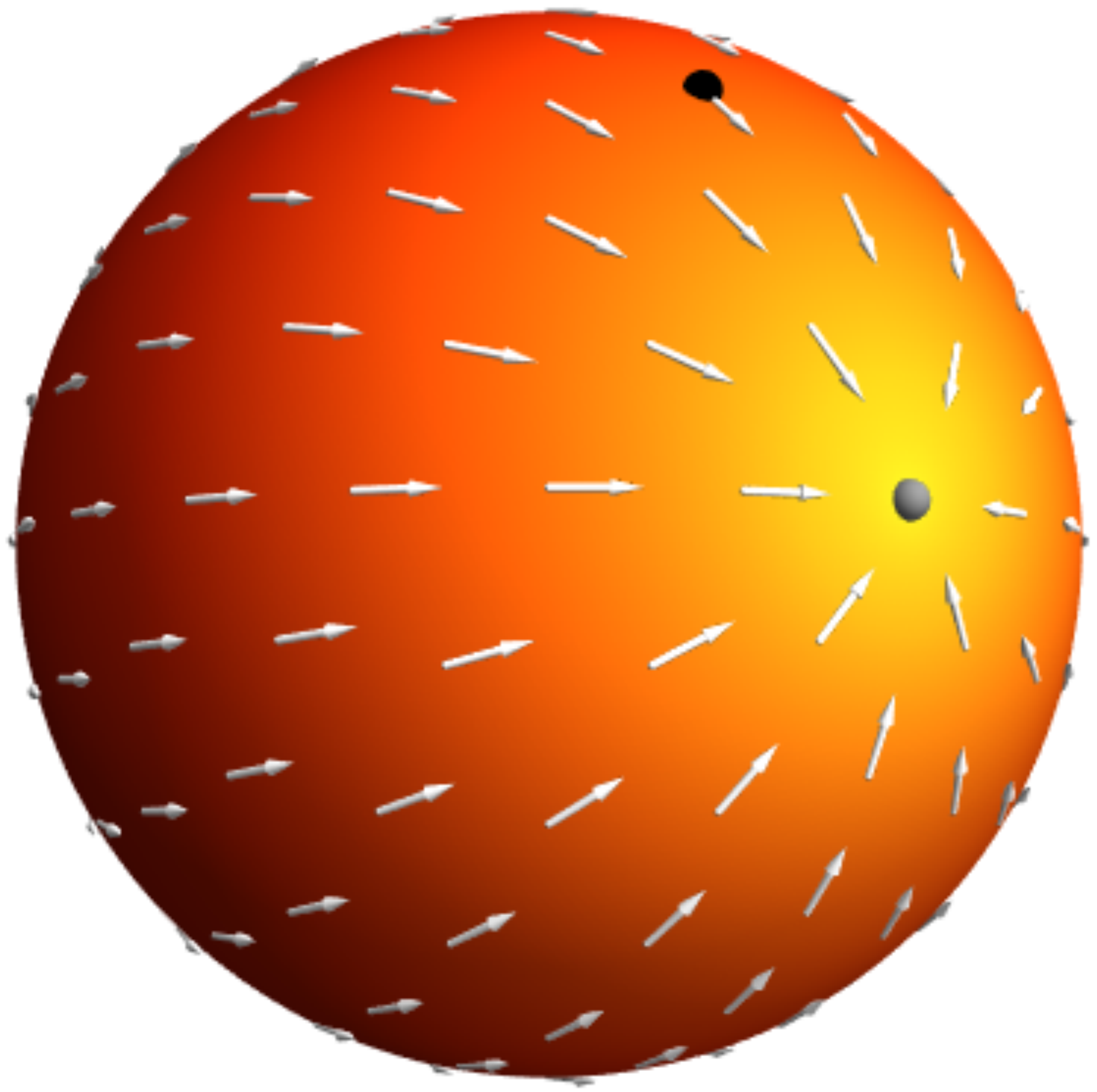
- Compute per triangle gradient of heat values.
- *Observation*: Gradient directions match those of geodesic distance quite exactly.
- *Idea*: Normalize and invert gradients, then integrate their divergence (solve a Poisson problem).

Angles in Heat



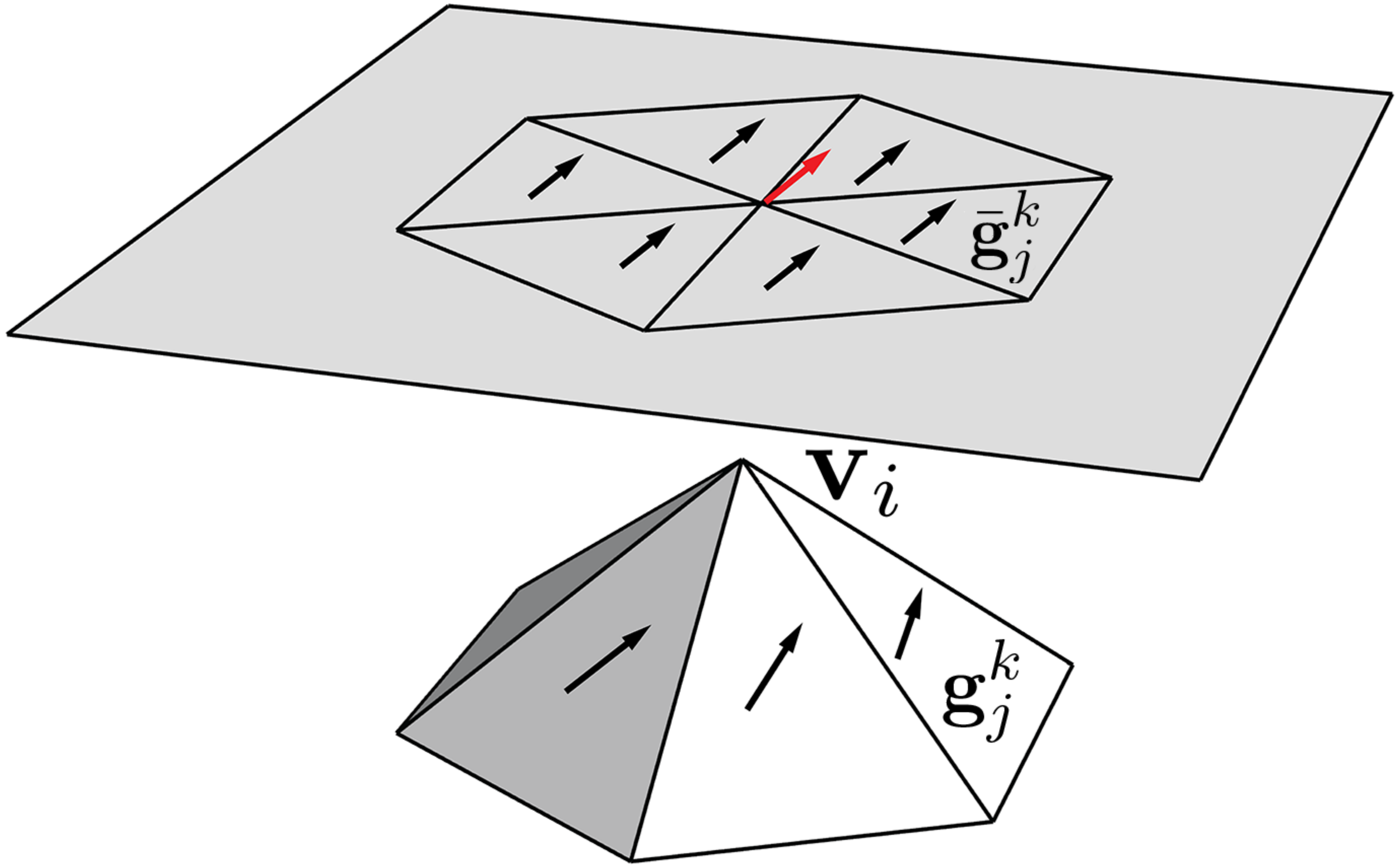
- How can we compute geodesic angles besides distances?

Angles in Heat



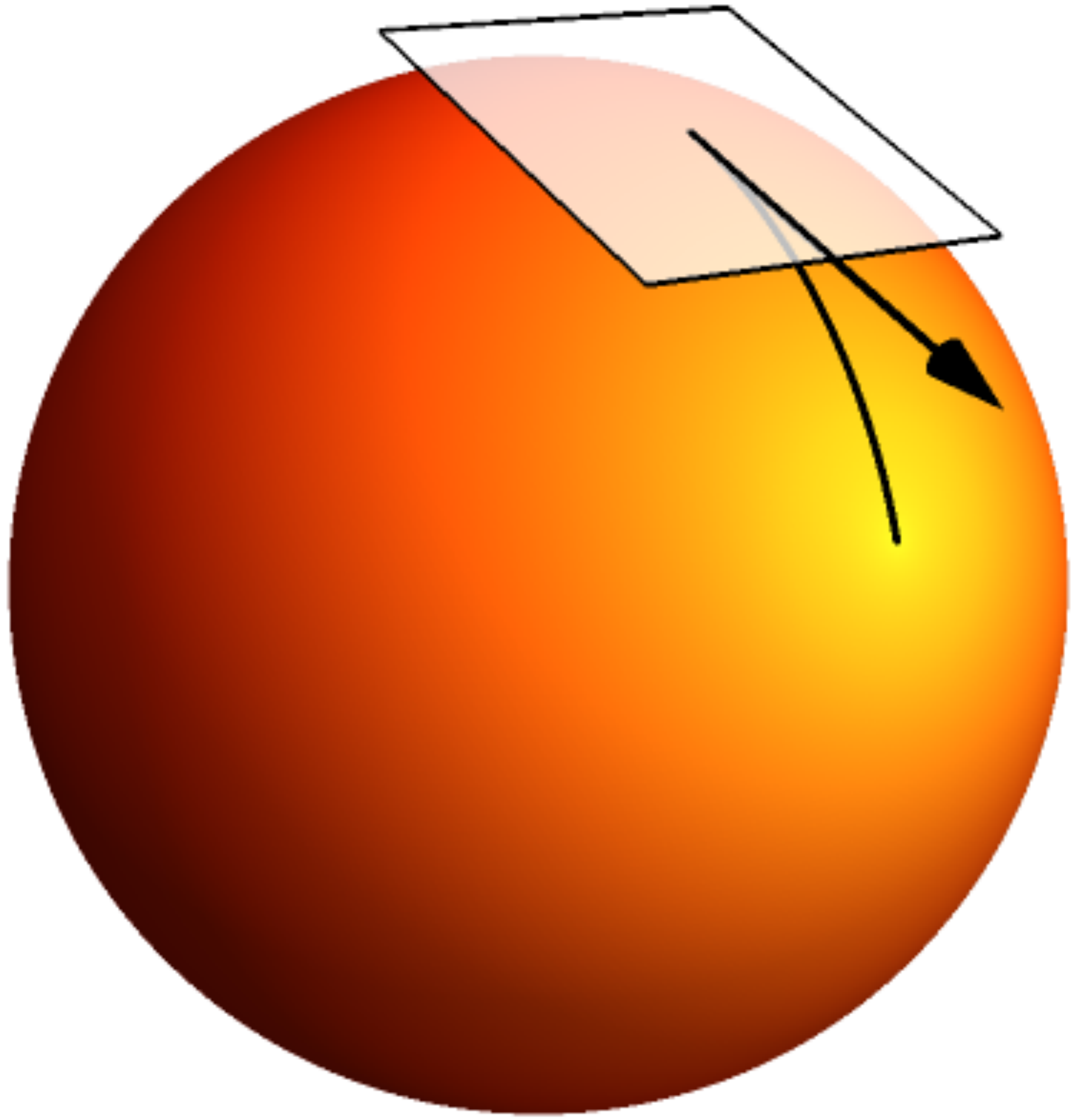
- How can we compute geodesic angles besides distances?
- *Observation*: Gradients directions of heat match those of geodesic distance quite exactly.

Angle interpolation



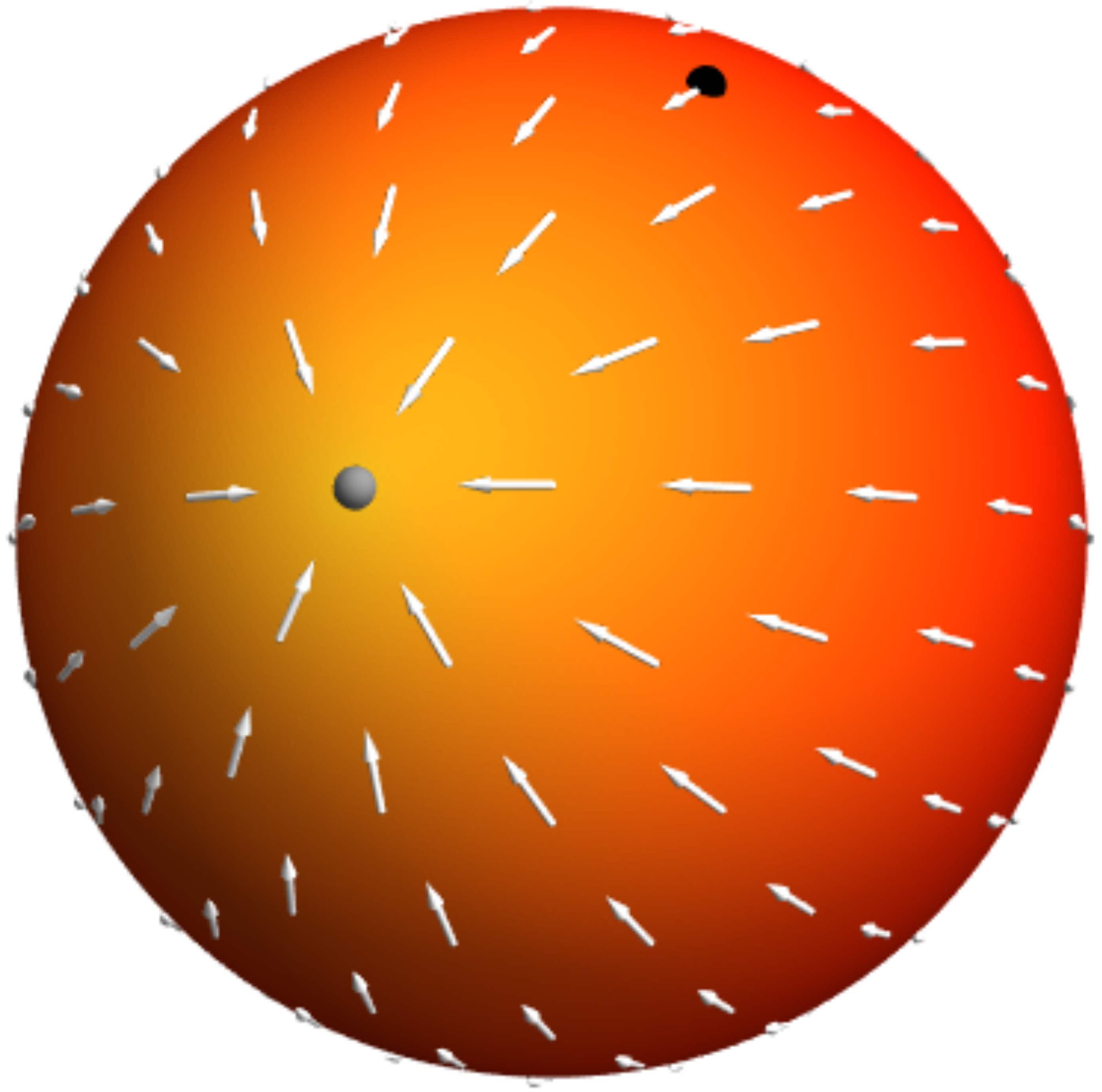
$$\bar{\mathbf{g}}_j^k = \mathbf{g}_j^k - \mathbf{n}_i \mathbf{n}_i^{\mathrm{T}} \mathbf{g}_j^k \qquad \mathbf{g}_j^i = \sum_k A_k \bar{\mathbf{g}}_j^k$$

Angles in Heat



- How can we compute geodesic angles besides distances?
- *Observation*: Gradients directions of heat match those of geodesic distance quite exactly.

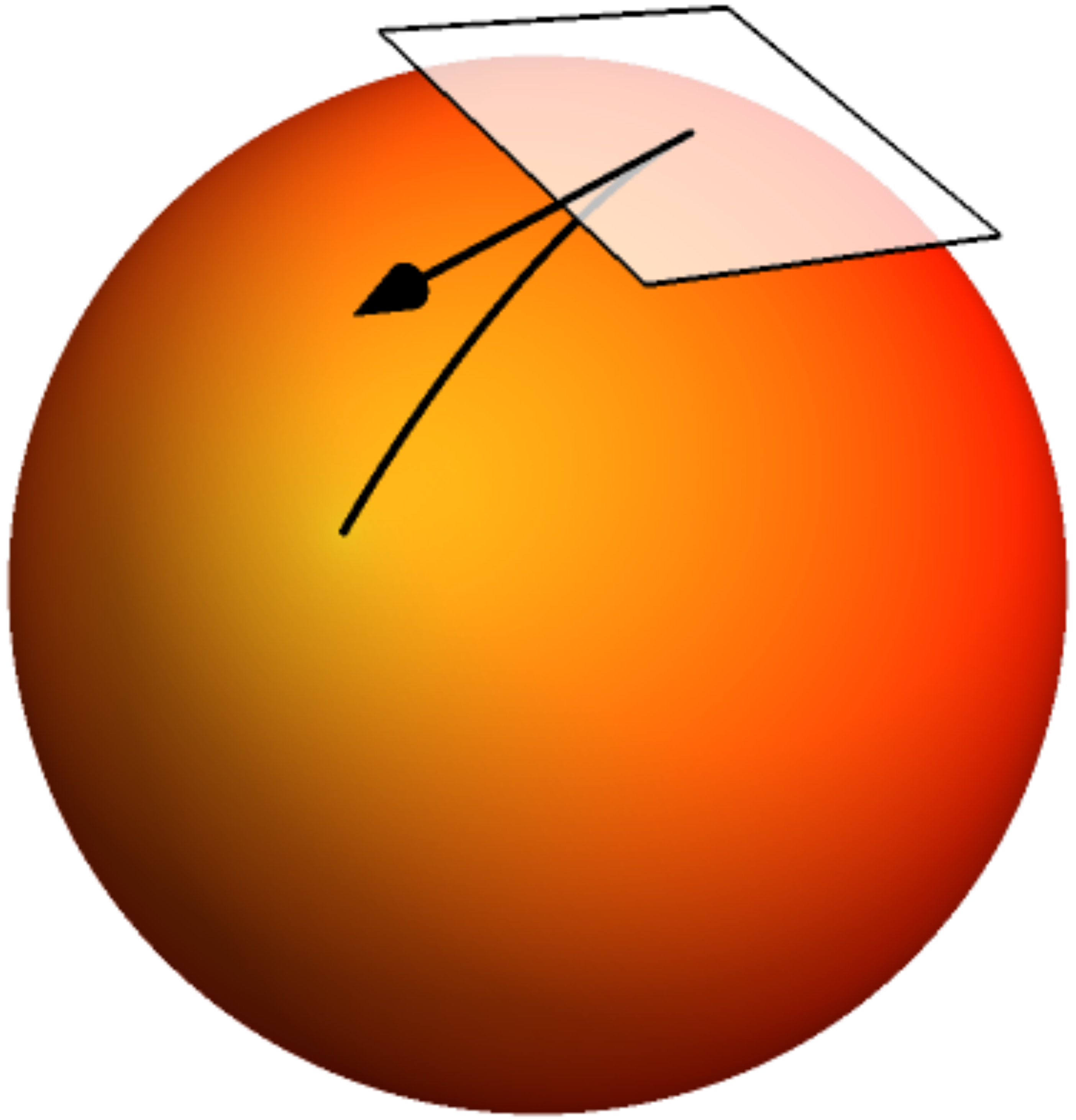
Angles in Heat



- How can we compute geodesic angles besides distances?

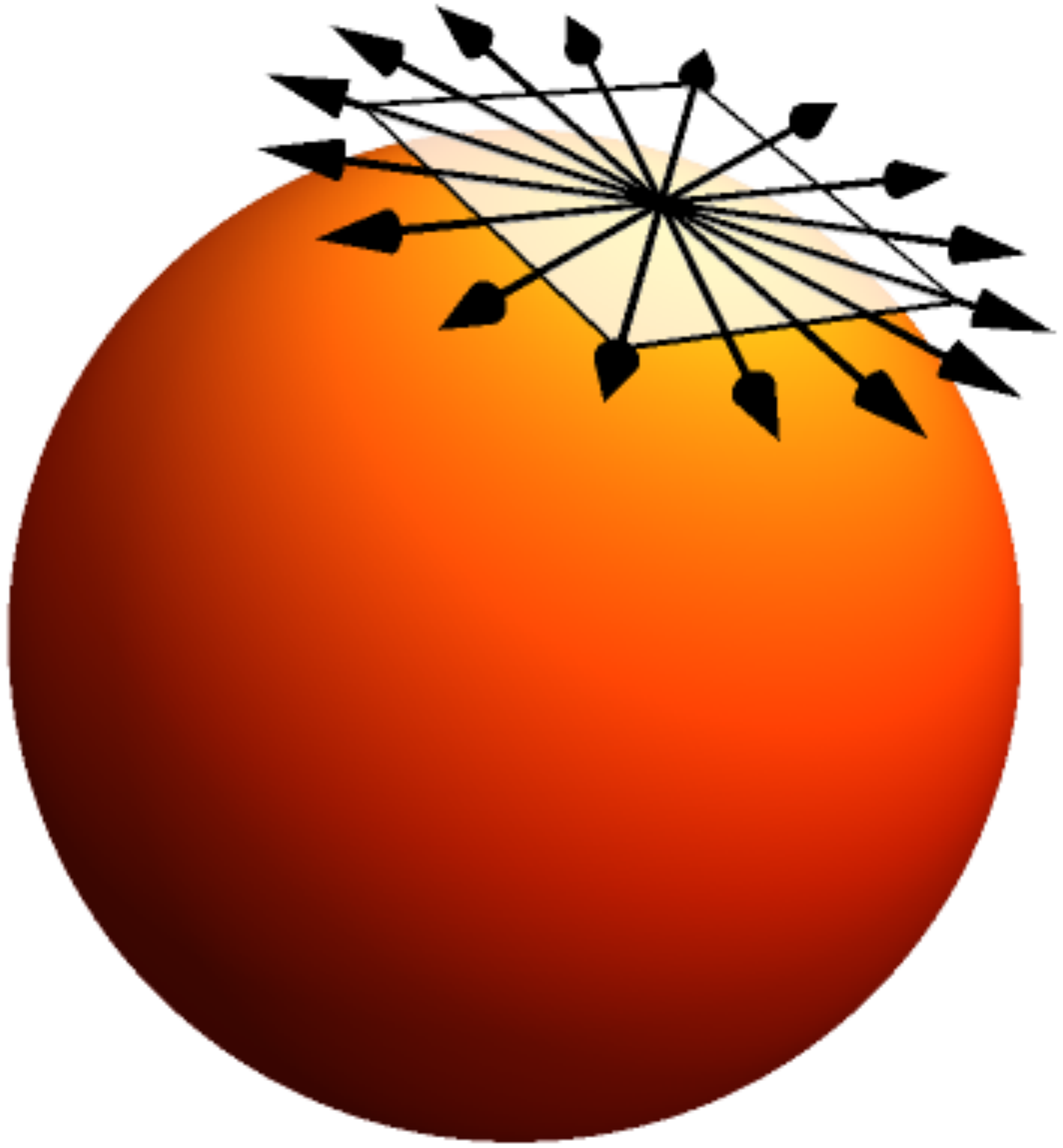
- *Observation*: Gradients directions of heat match those of geodesic distance quite exactly.

Angles in Heat



- How can we compute geodesic angles besides distances?
- *Observation*: Gradients directions of heat match those of geodesic distance quite exactly.

Angles in Heat



- How can we compute geodesic angles besides distances?
- *Observation*: Gradients directions of heat match those of geodesic distance quite exactly.
- *Problem*: Need to evaluate heat for each vertex.

Angles in Heat

Solve for every vertex j :

$$\mathbf{A} = \mathbf{M} - t\mathbf{L}$$

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

Angles in Heat

Solve for every vertex j :

$$\mathbf{A} = \mathbf{M} - t\mathbf{L}$$

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

Compute gradient of every $\mathbf{h}_j$ at center vertex i .

Angles in Heat

Solve for every vertex j :

$$\mathbf{A} = \mathbf{M} - t\mathbf{L}$$

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

Compute gradient of every $\mathbf{h}_j$ at center vertex i .

Observation 1: $\mathbf{h}_j$ is the j -th column of $\mathbf{A}^{-1}$. $\mathbf{A} * \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} = \mathbf{I}$

Angles in Heat

Solve for every vertex j :

$$\mathbf{A} = \mathbf{M} - t\mathbf{L}$$

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

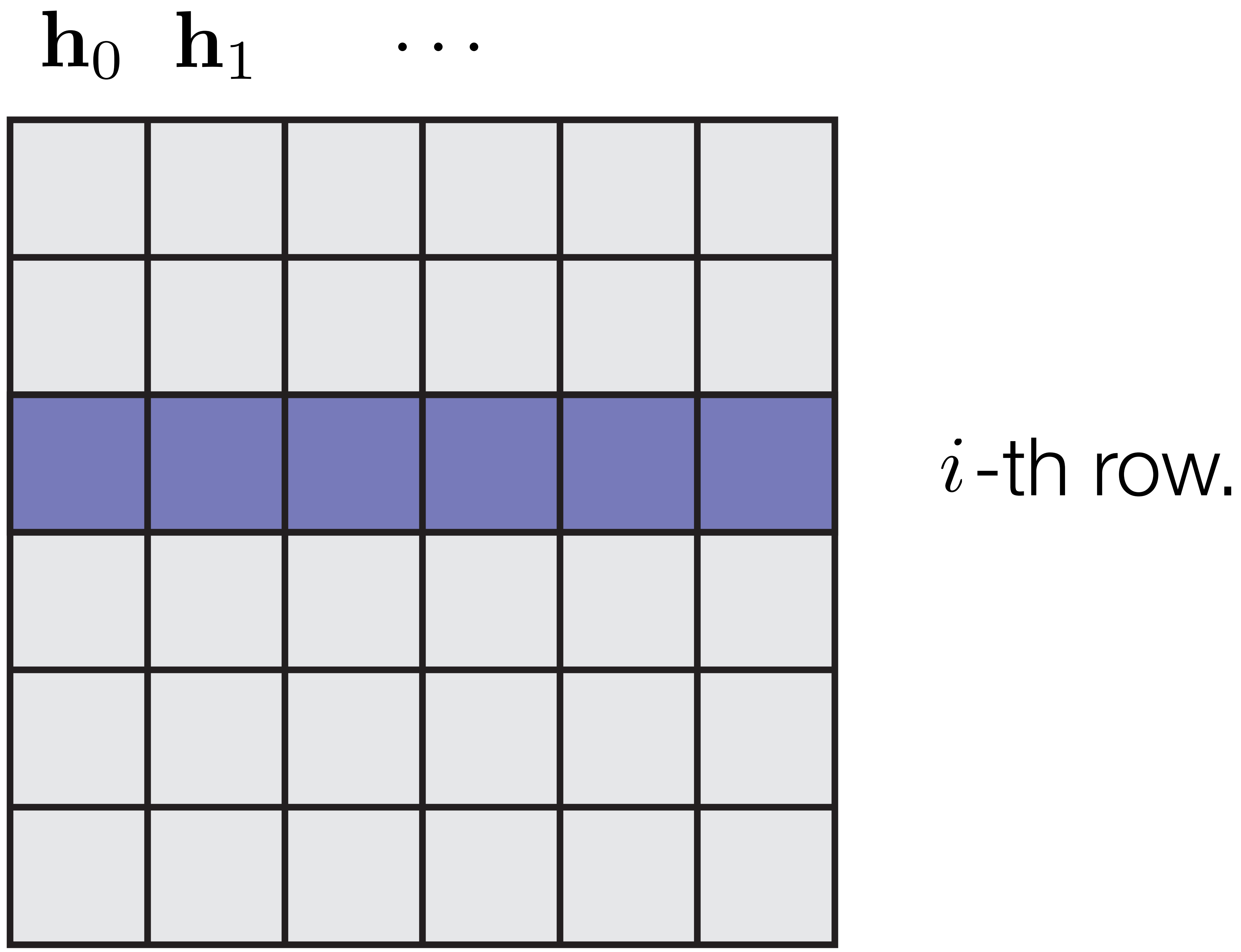
Compute gradient of every $\mathbf{h}_j$ at center vertex i .

Observation 1: $\mathbf{h}_j$ is the j-th column of $\mathbf{A}^{-1}$. $\quad \mathbf{A} * \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} = \mathbf{I}$

Observation 2: To evaluate the gradient at vertex i, we only need a few rows of $\mathbf{h}_j$.

Angles in Heat

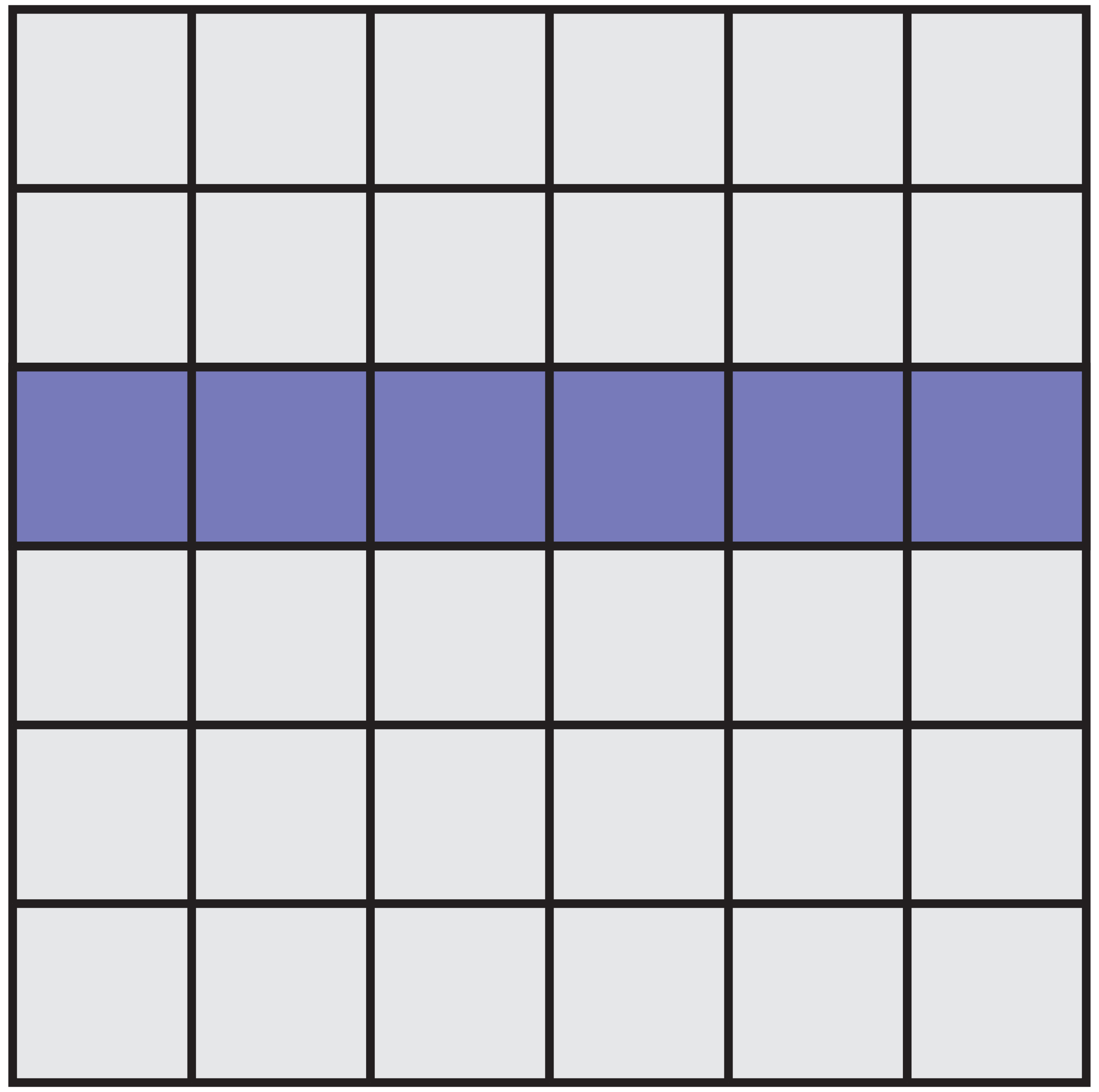
$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} =$$



Angles in Heat

$\mathbf{h}_0 \quad \mathbf{h}_1 \quad \cdots$

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} =$$

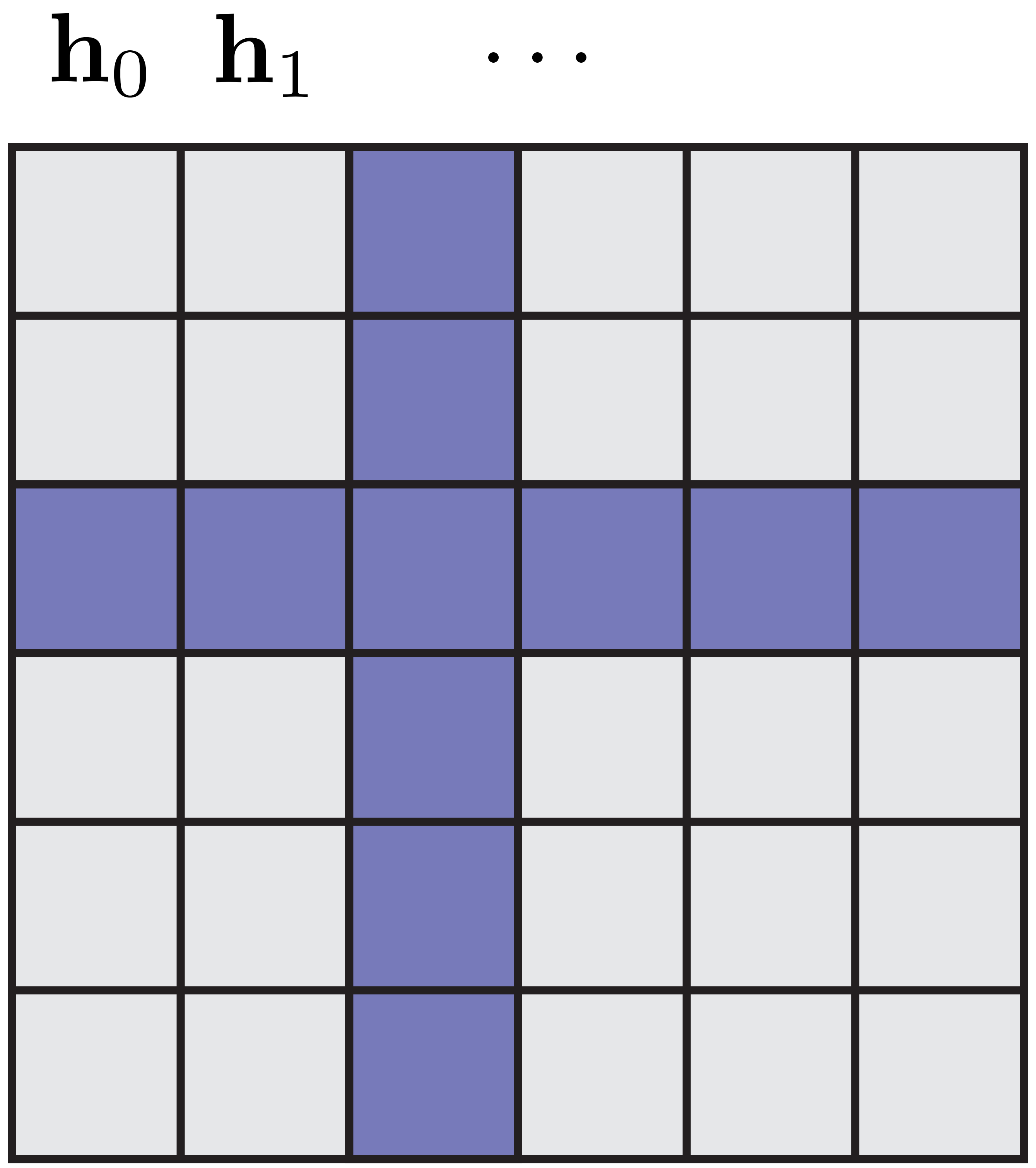


i-th row.

Idea: $\mathbf{A}$ is symmetric $\Rightarrow \mathbf{A}^{-1}$ is symmetric.

Angles in Heat

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} =$$

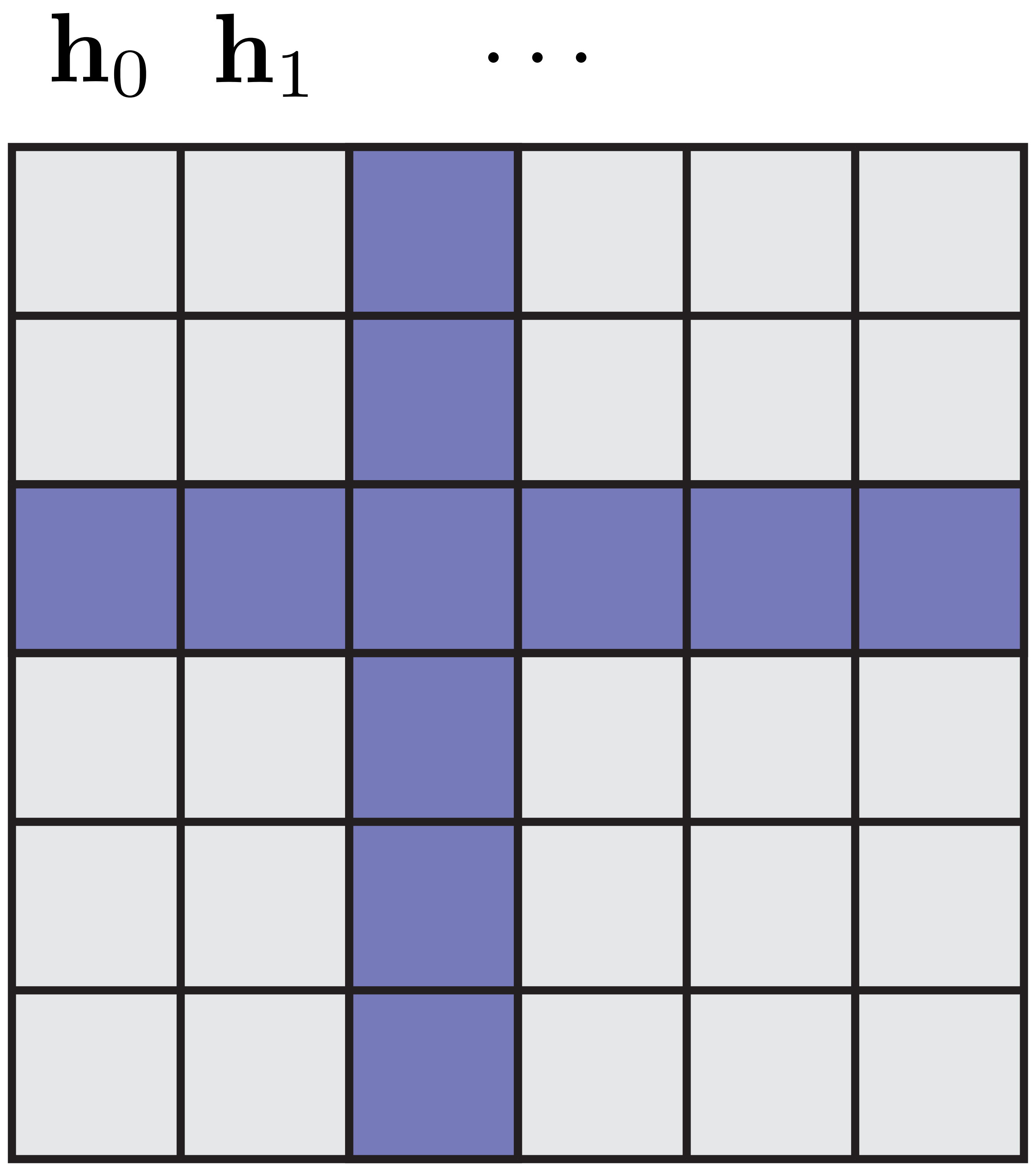


i-th row = $\mathbf{h}_i$.

Idea: $\mathbf{A}$ is symmetric $\Rightarrow \mathbf{A}^{-1}$ is symmetric.

Angles in Heat

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} =$$



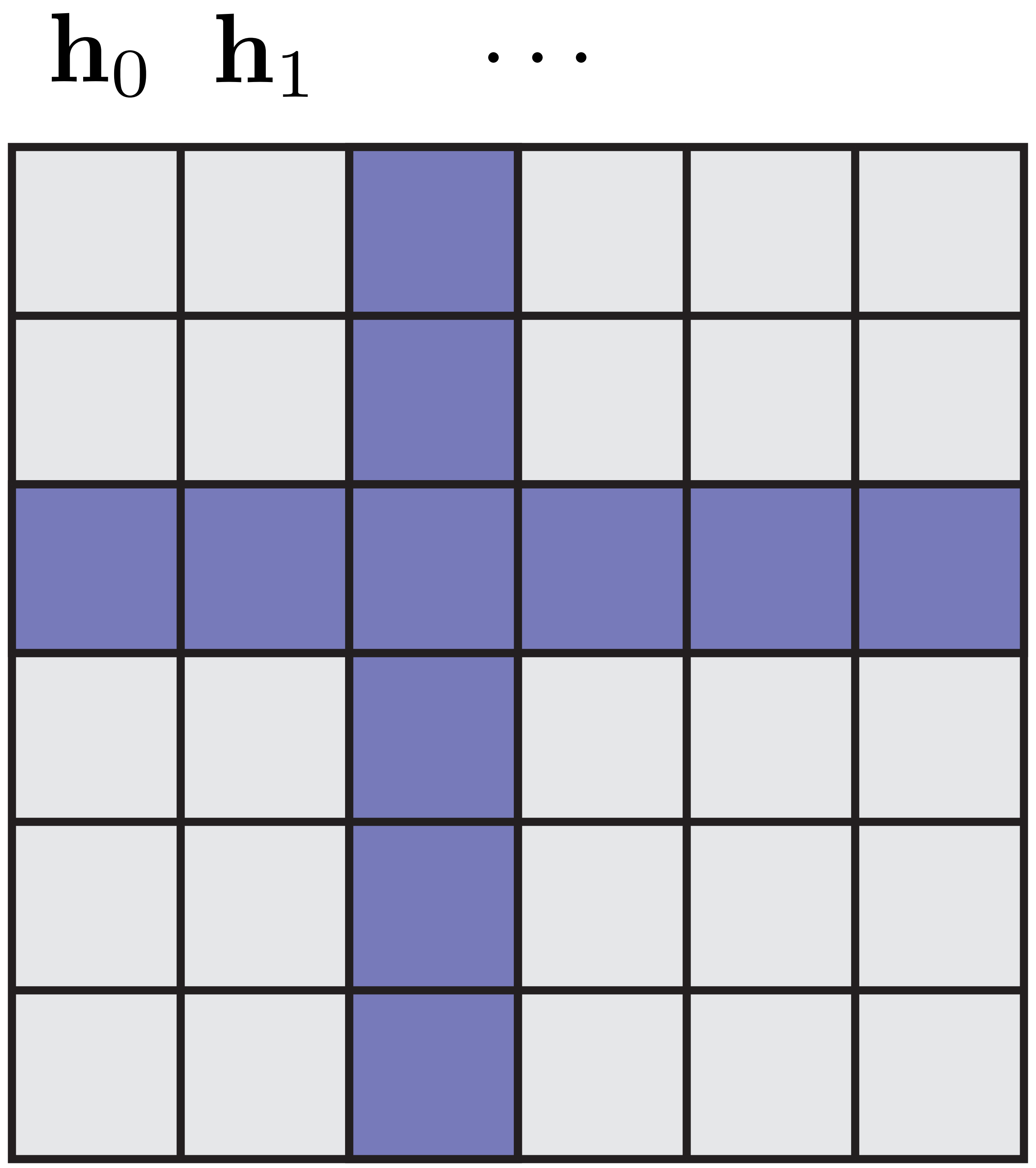
i-th row = $\mathbf{h}_i$.

Idea: $\mathbf{A}$ is symmetric $\Rightarrow \mathbf{A}^{-1}$ is symmetric.

We need to evaluate heat diffusion only for a few vertices (≈ 7) instead of all!

Angles in Heat

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \cdots \end{pmatrix} =$$



i-th row = $\mathbf{h}_i$.

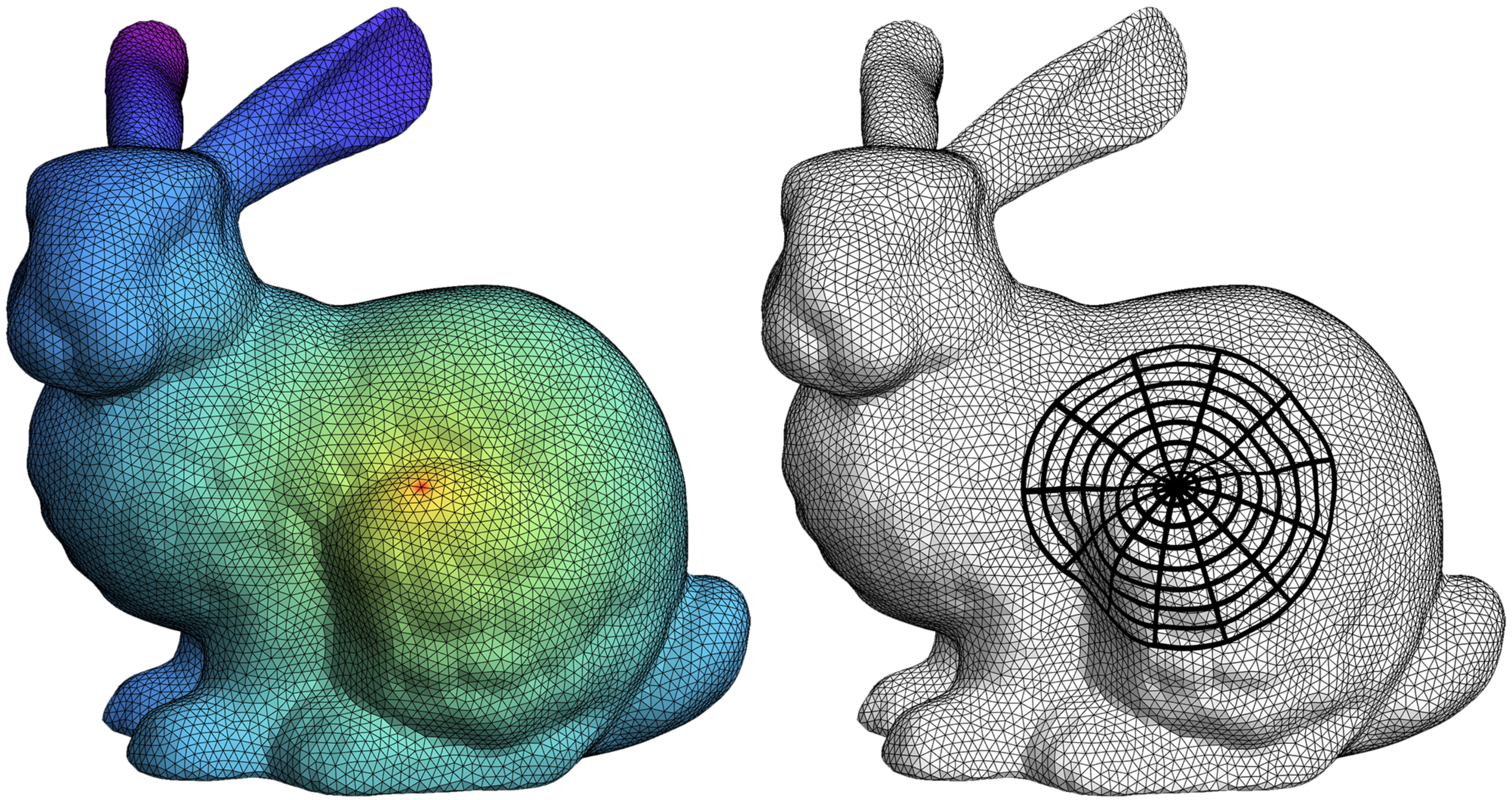
Idea: $\mathbf{A}$ is symmetric $\Rightarrow \mathbf{A}^{-1}$ is symmetric.

We need to evaluate heat diffusion only for a few vertices (≈ 7) instead of all!

Angles in Heat

We need to evaluate heat diffusion only for a few vertices ($\approx$ 7) instead of all!

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

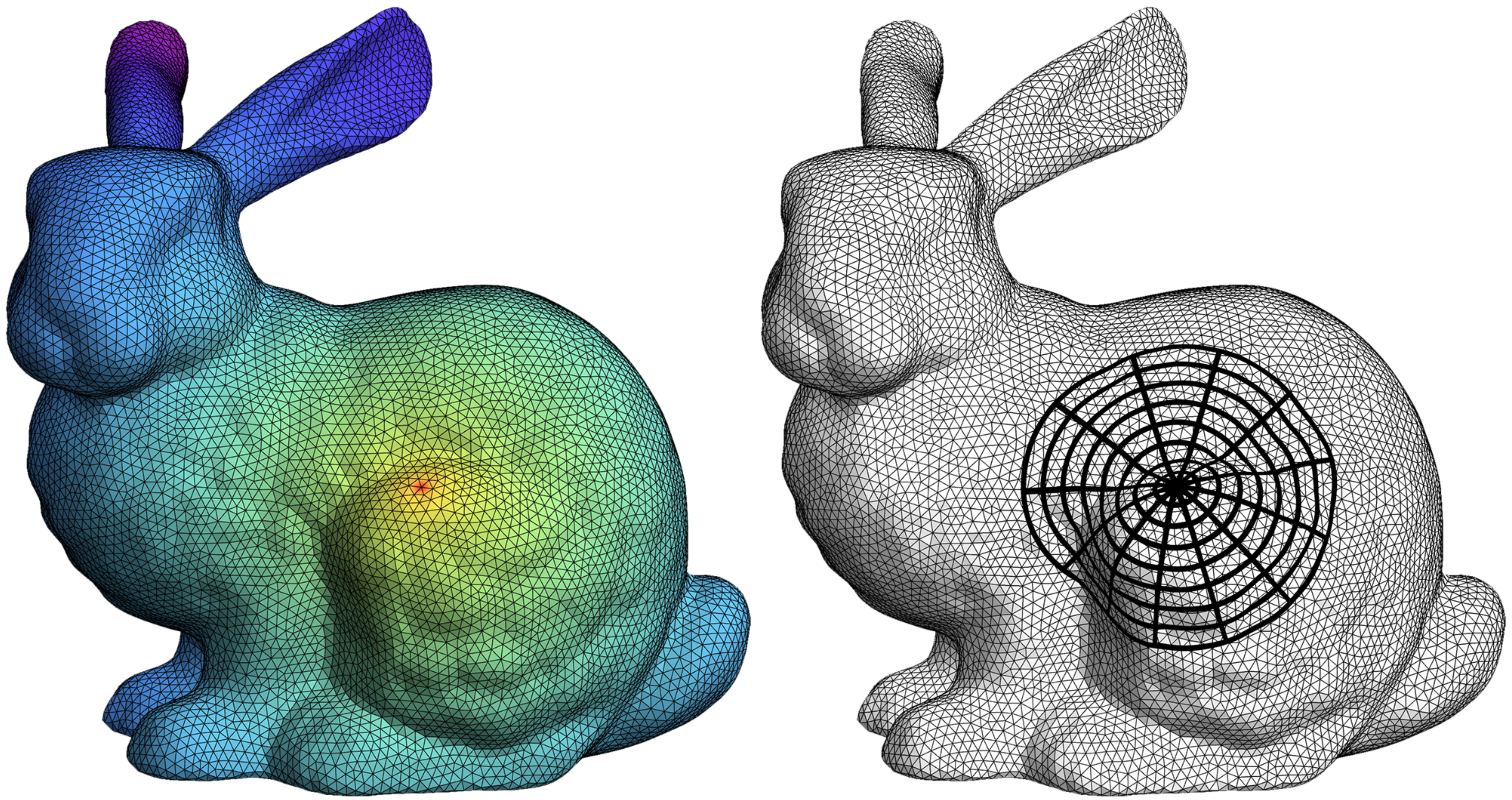


Angles in Heat

We need to evaluate heat diffusion only for a few vertices ($\approx$ 7) instead of all!

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

We get information for all vertices but usually only need a local exponential map.

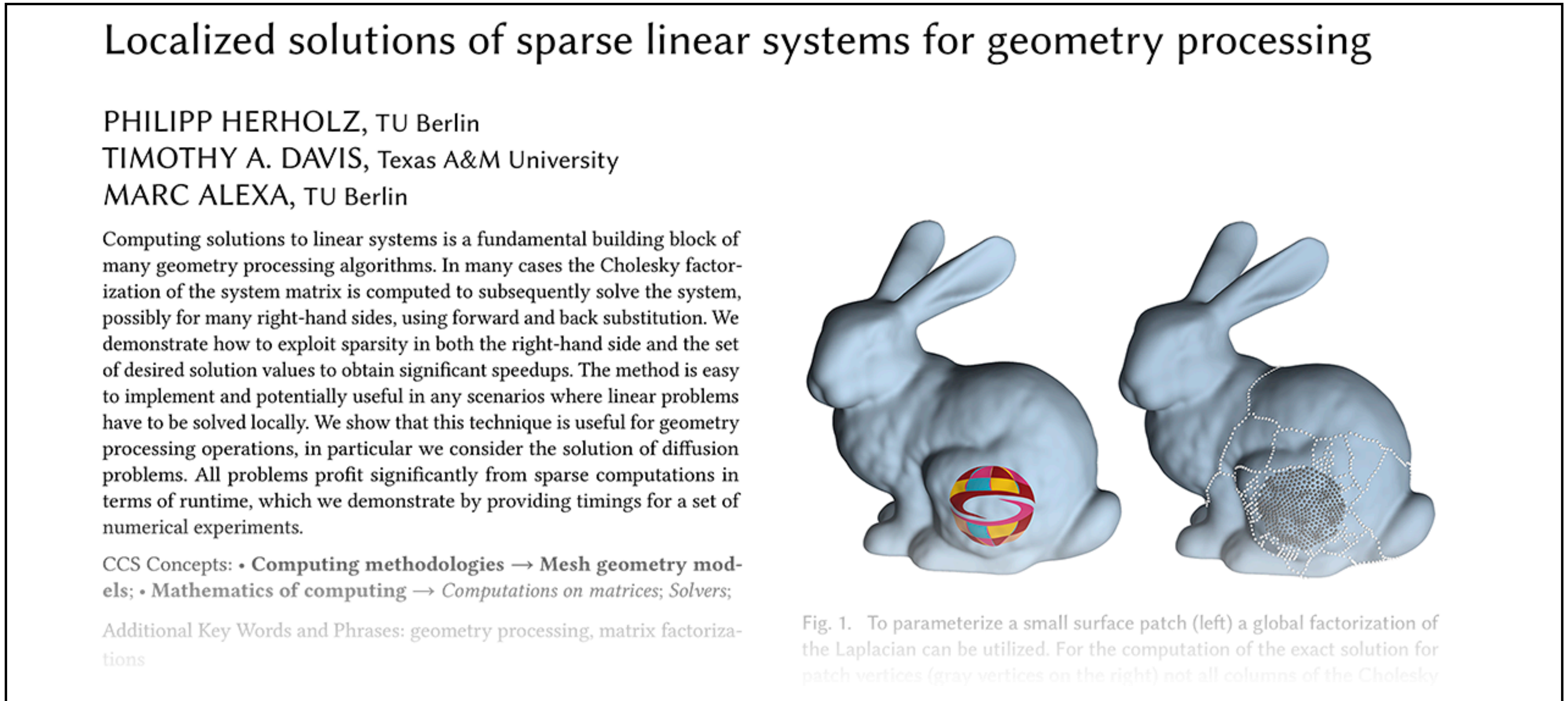


Localization

We need to evaluate heat diffusion only for a few vertices (≈ 7) instead of all!

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

We get information for all vertices but usually only need a local exponential map.



Localized solutions of sparse linear systems for geometry processing

PHILIPP HERHOLZ, TU Berlin
TIMOTHY A. DAVIS, Texas A&M University
MARC ALEXA, TU Berlin

Computing solutions to linear systems is a fundamental building block of many geometry processing algorithms. In many cases the Cholesky factorization of the system matrix is computed to subsequently solve the system, possibly for many right-hand sides, using forward and back substitution. We demonstrate how to exploit sparsity in both the right-hand side and the set of desired solution values to obtain significant speedups. The method is easy to implement and potentially useful in any scenarios where linear problems have to be solved locally. We show that this technique is useful for geometry processing operations, in particular we consider the solution of diffusion problems. All problems profit significantly from sparse computations in terms of runtime, which we demonstrate by providing timings for a set of numerical experiments.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; • **Mathematics of computing** → *Computations on matrices*; *Solvers*;

Additional Key Words and Phrases: geometry processing, matrix factorizations

Fig. 1. To parameterize a small surface patch (left) a global factorization of the Laplacian can be utilized. For the computation of the exact solution for patch vertices (gray vertices on the right) not all columns of the Cholesky

Cholesky factorization

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

Cholesky factorization

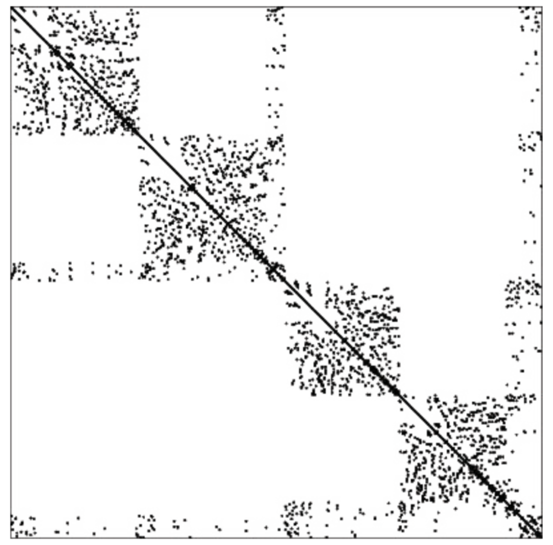
$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

$$\mathbf{L}\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{e}_j$$

Cholesky factorization

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

$$\mathbf{L}\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{e}_j$$

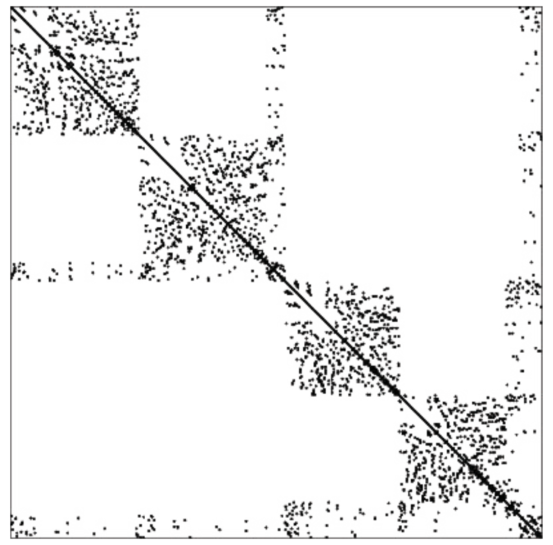
$\mathbf{A} =$ 

$\mathbf{L} =$ 

Cholesky factorization

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

$$\mathbf{L}\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{e}_j$$

$\mathbf{A} =$ 

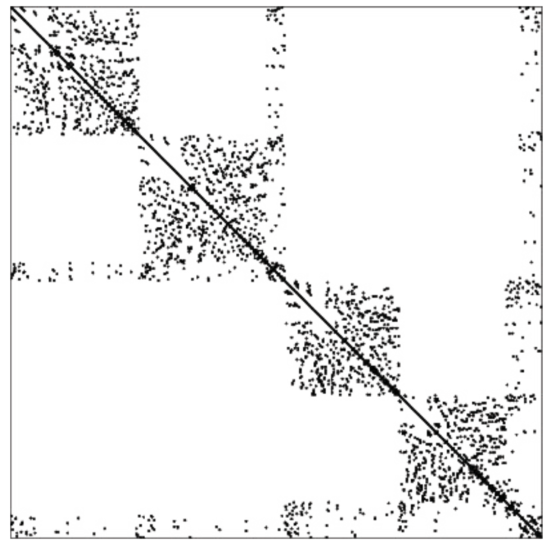
$\mathbf{L} =$ 

$$\mathbf{L}\mathbf{y} = \mathbf{e}_j$$

Cholesky factorization

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

$$\mathbf{L}\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{e}_j$$

$\mathbf{A} =$ 

$\mathbf{L} =$ 

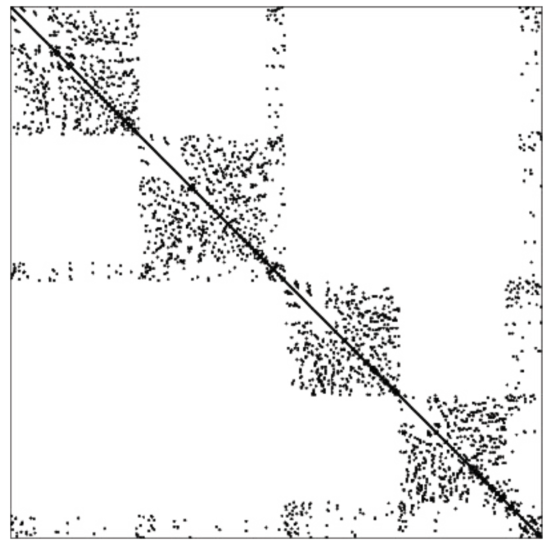
$$\mathbf{L}\mathbf{y} = \mathbf{e}_j$$

$$\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{y}$$

Cholesky factorization

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

$$\mathbf{L}\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{e}_j$$

$\mathbf{A} =$ 

$\mathbf{L} =$ 

$$\mathbf{L}\mathbf{y} = \mathbf{e}_j$$

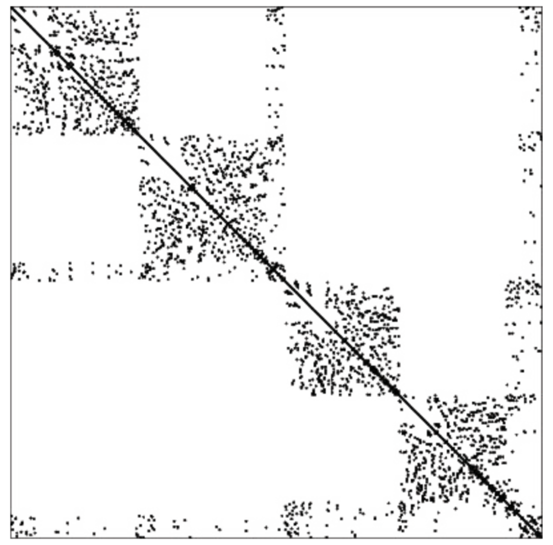
$\mathbf{y}$ is sparse because $\mathbf{e}_j$ is.

$$\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{y}$$

Cholesky factorization

$$\mathbf{A}\mathbf{h}_j = \mathbf{e}_j$$

$$\mathbf{L}\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{e}_j$$

$\mathbf{A} =$ 

$\mathbf{L} =$ 

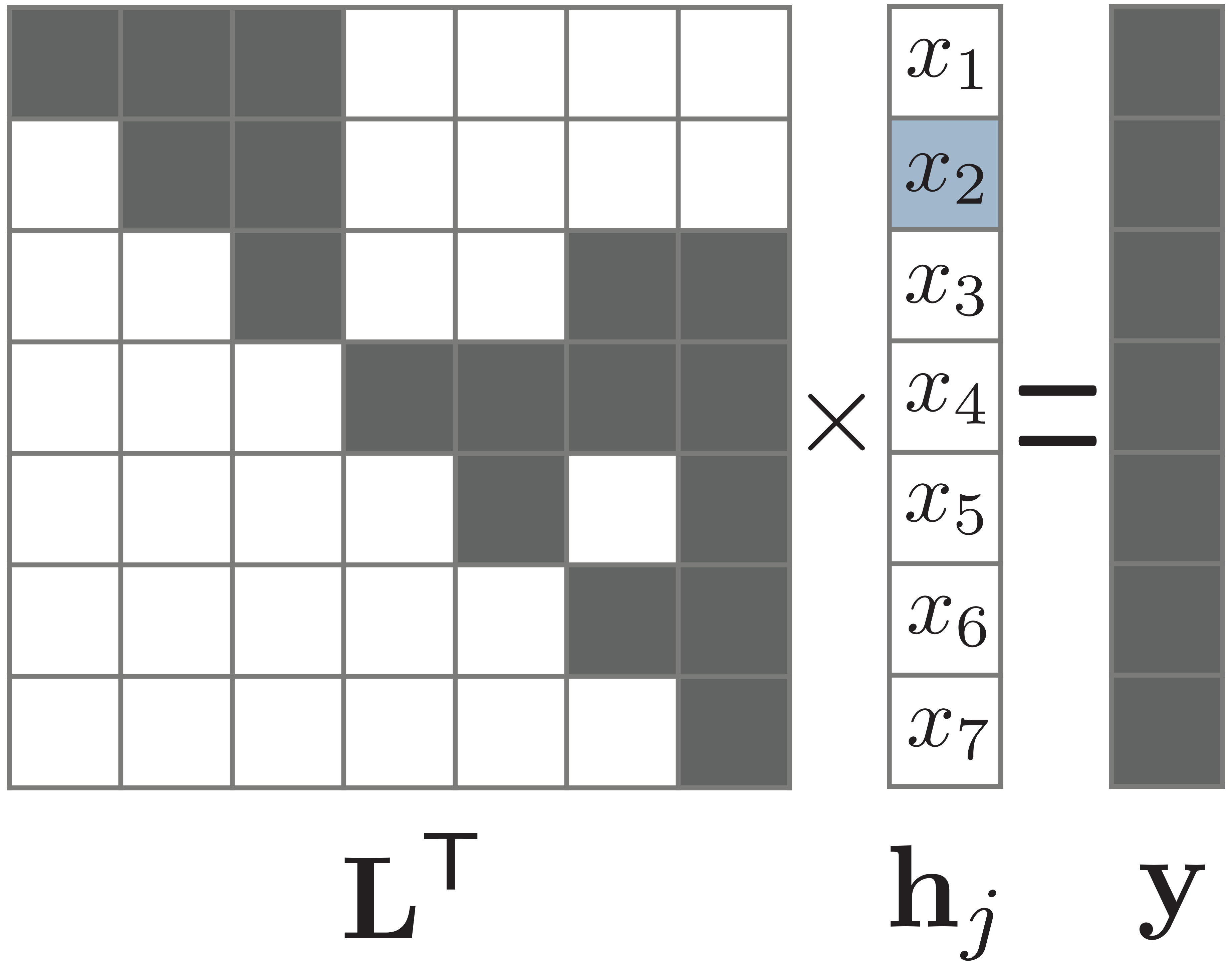
$$\mathbf{L}\mathbf{y} = \mathbf{e}_j$$

$\mathbf{y}$ is sparse because $\mathbf{e}_j$ is.

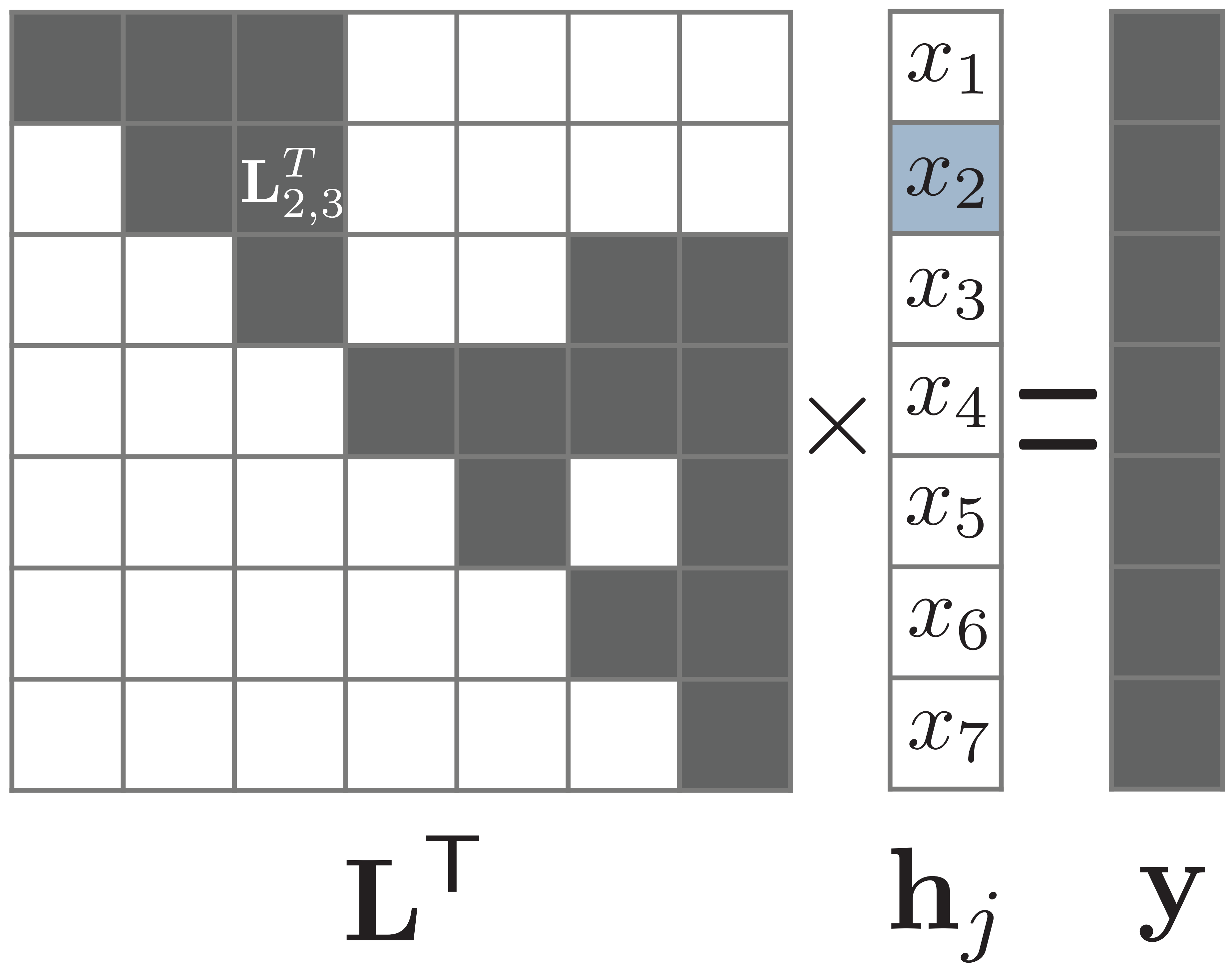
$$\mathbf{L}^{\mathsf{T}}\mathbf{h}_j = \mathbf{y}$$

We are interested in only a few values of $\mathbf{h}_j$.

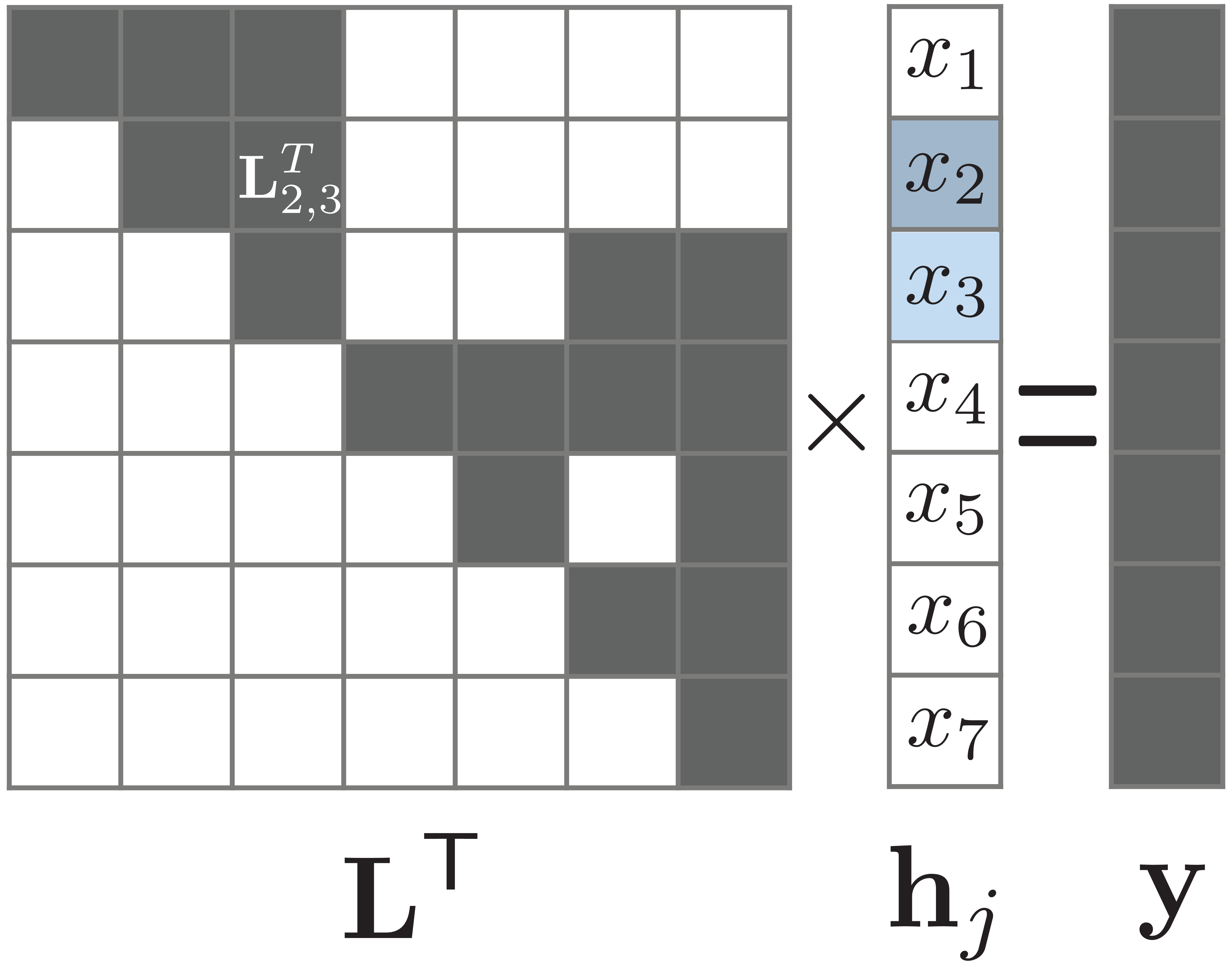
Localization



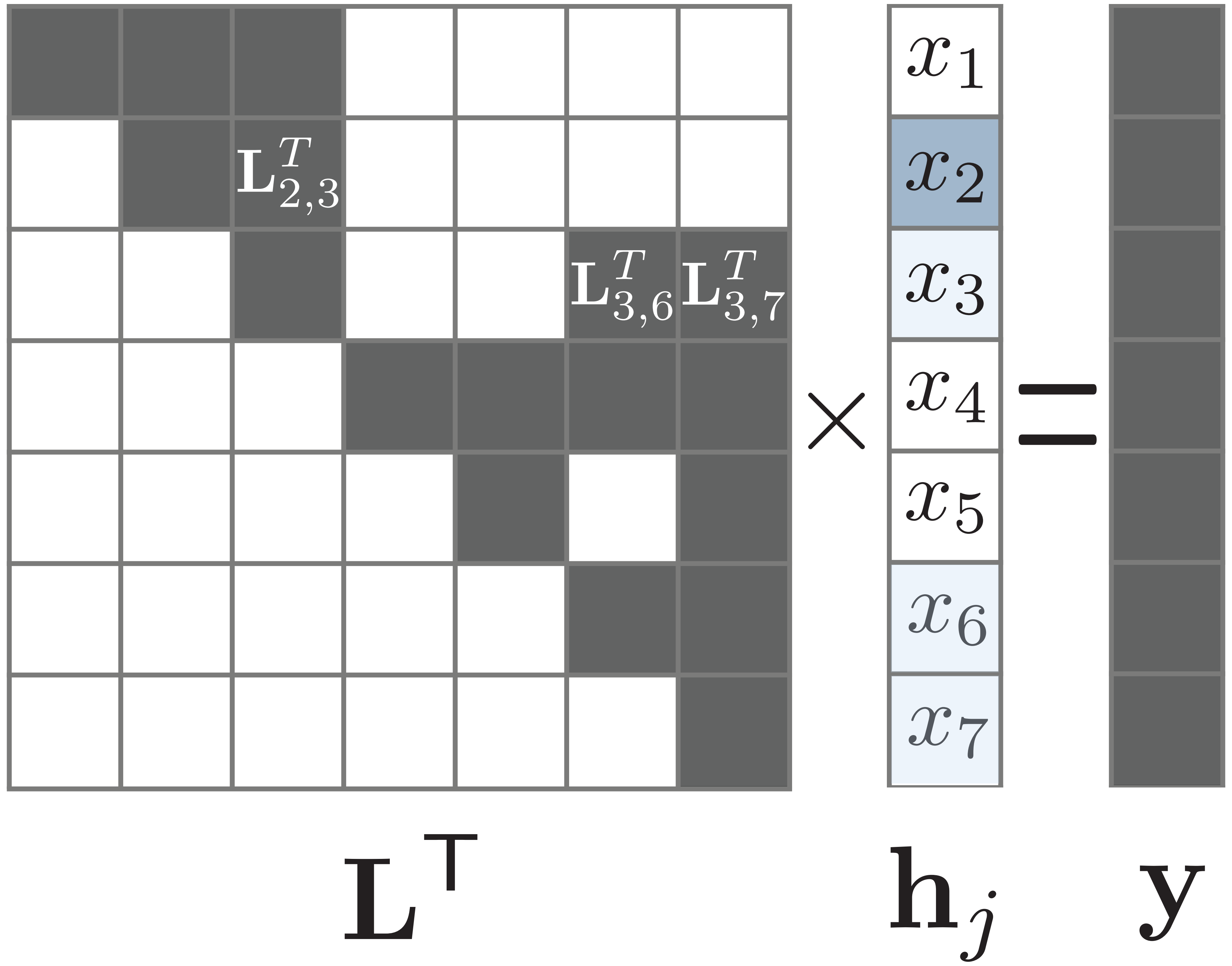
Localization



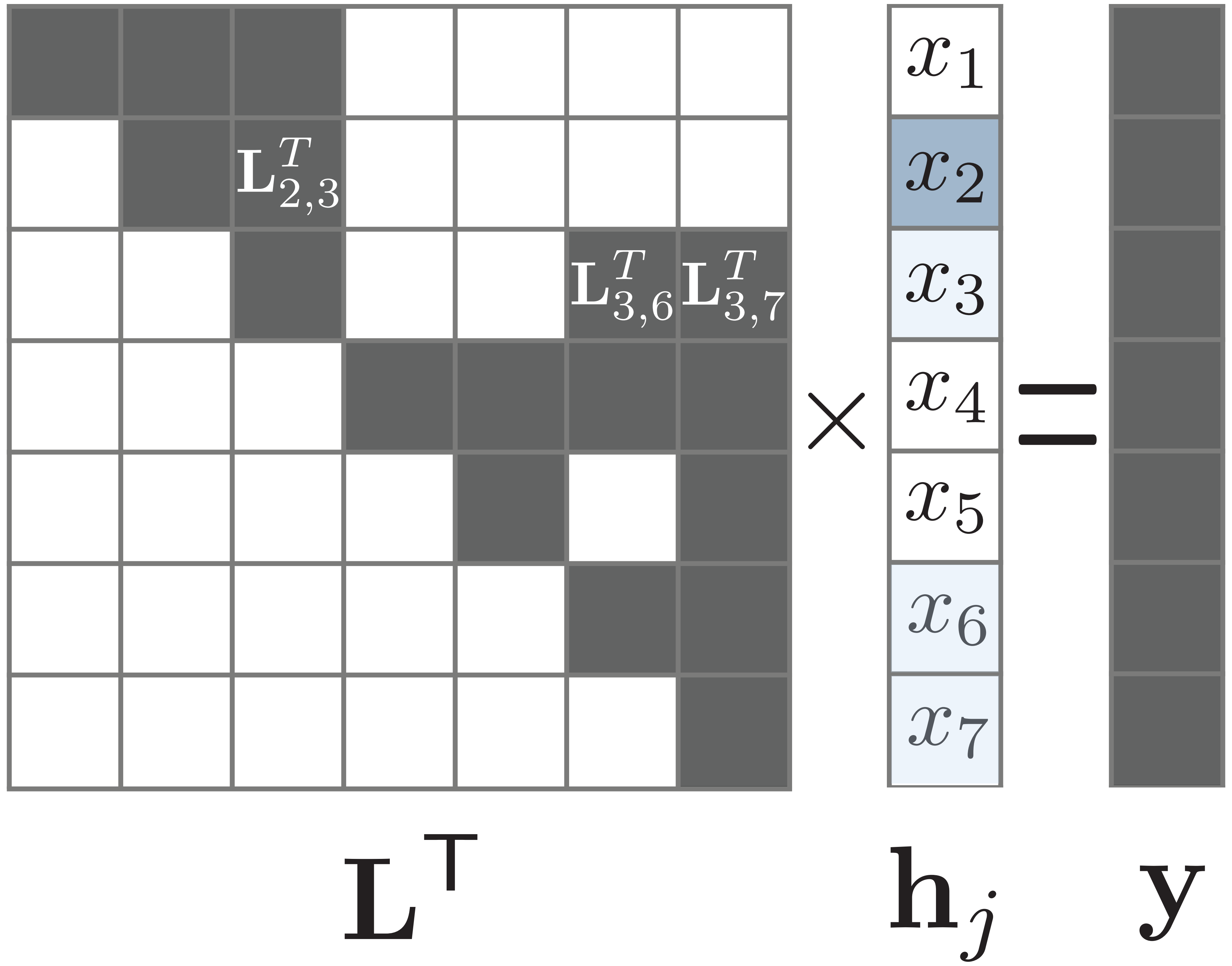
Localization



Localization

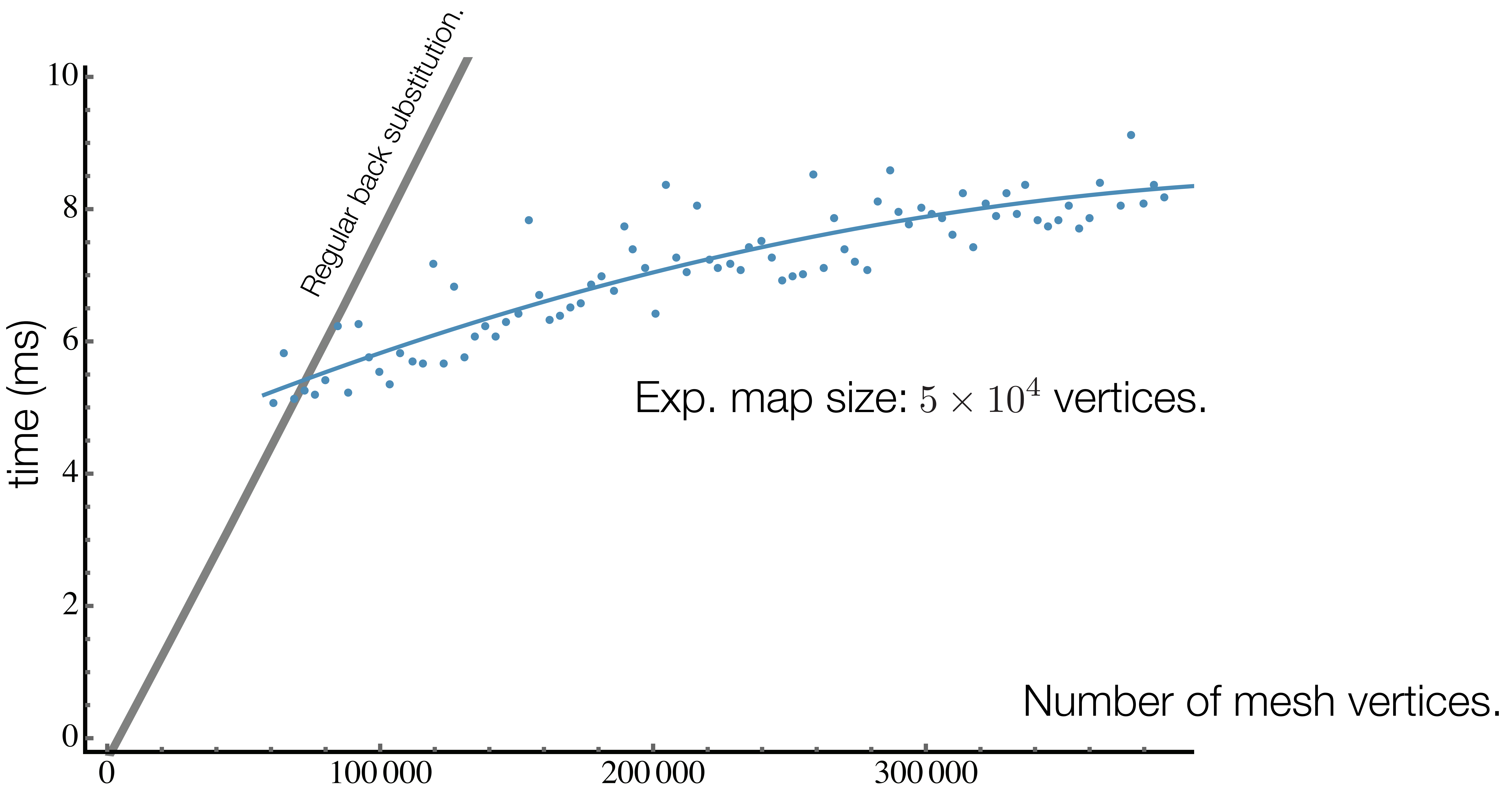


Localization

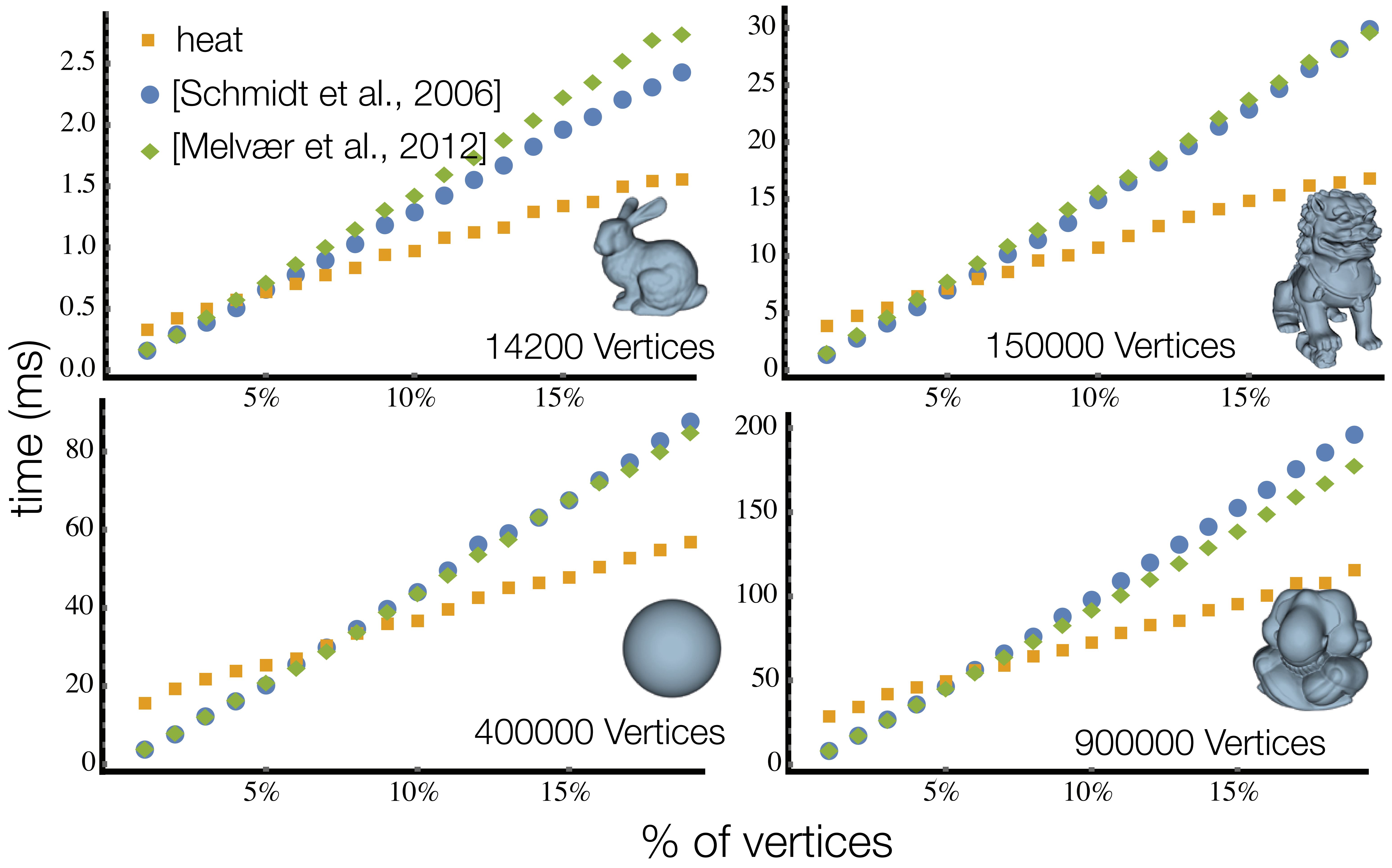


Compute x_7, x_6, x_3 and finally x_2. All other variables and rows remain unvisited.

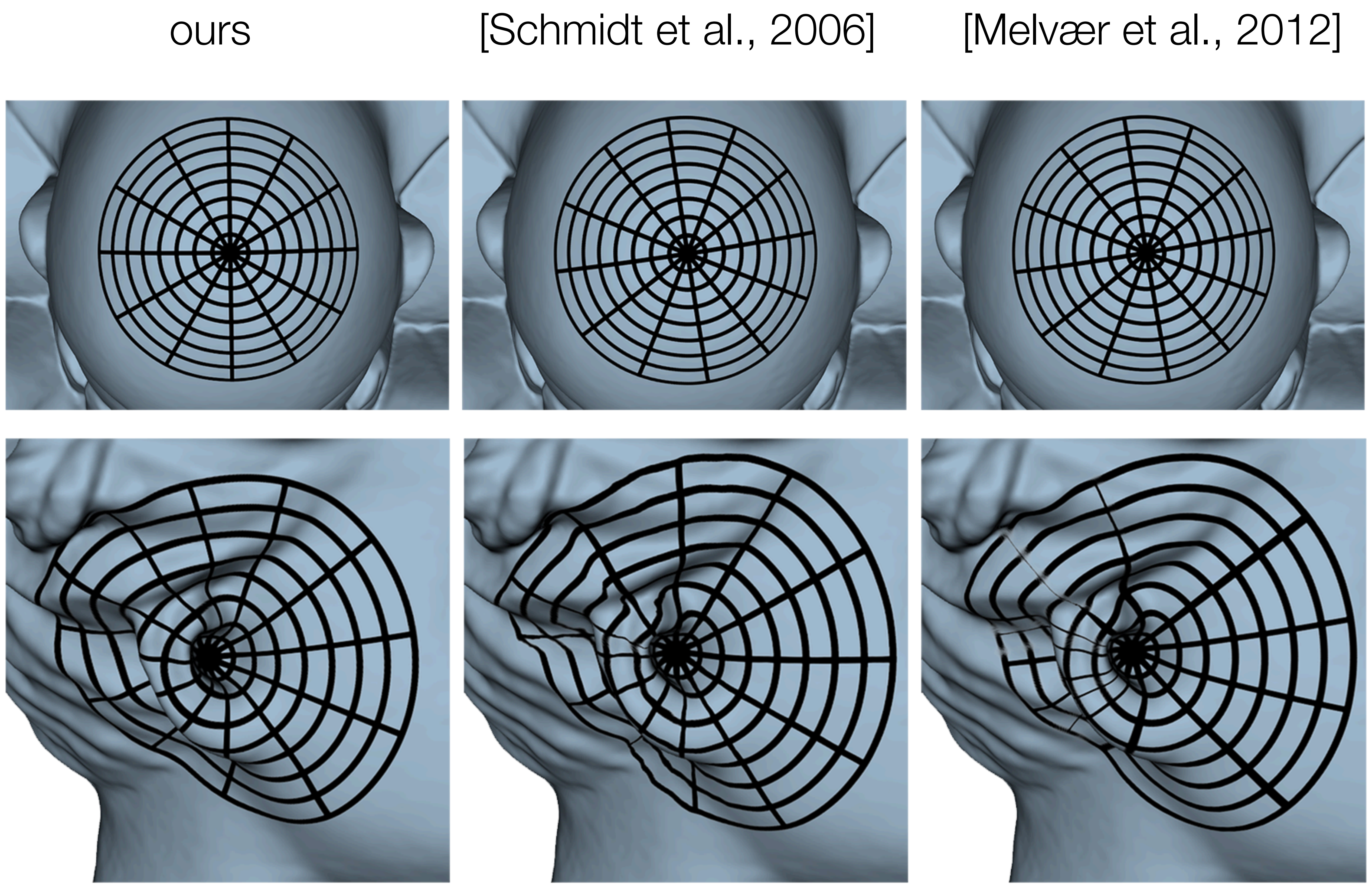
Localization: Performance

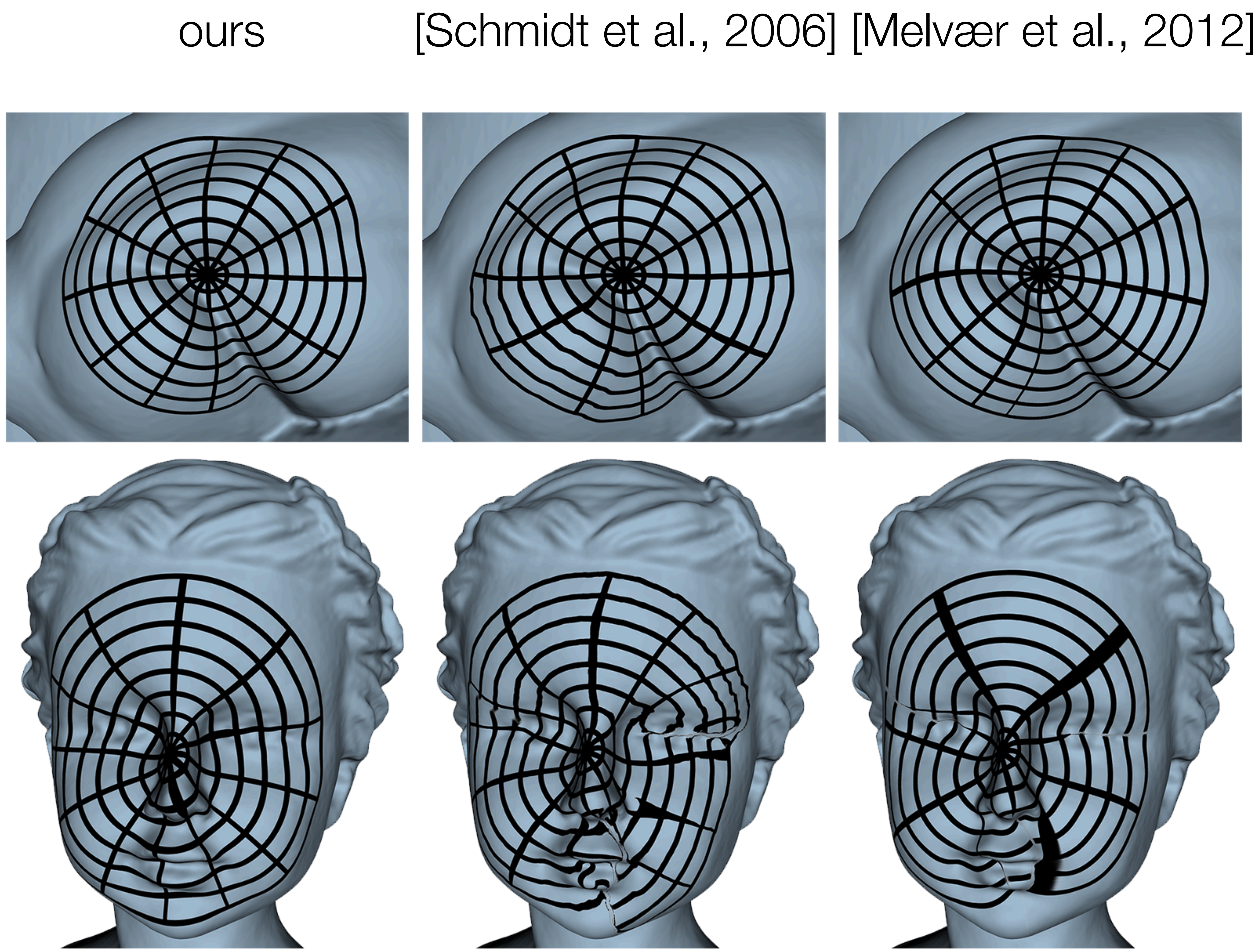


Performance comparison



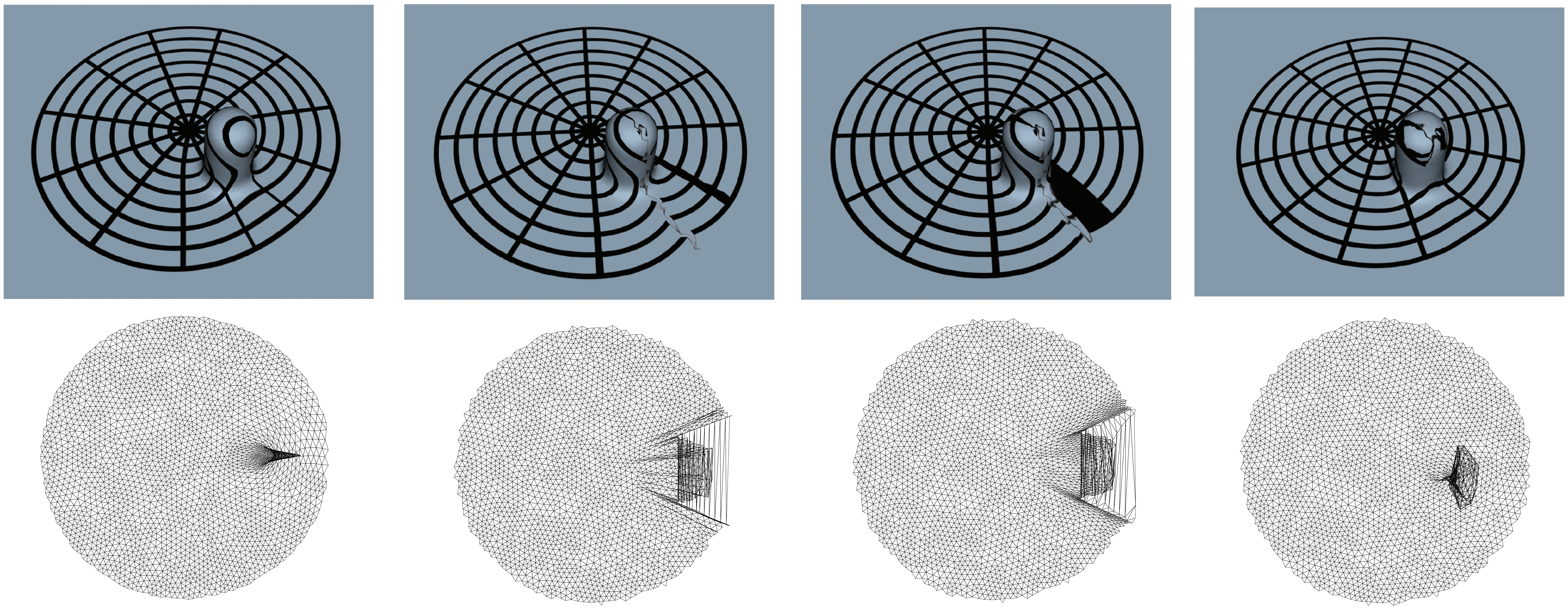
Quality comparison





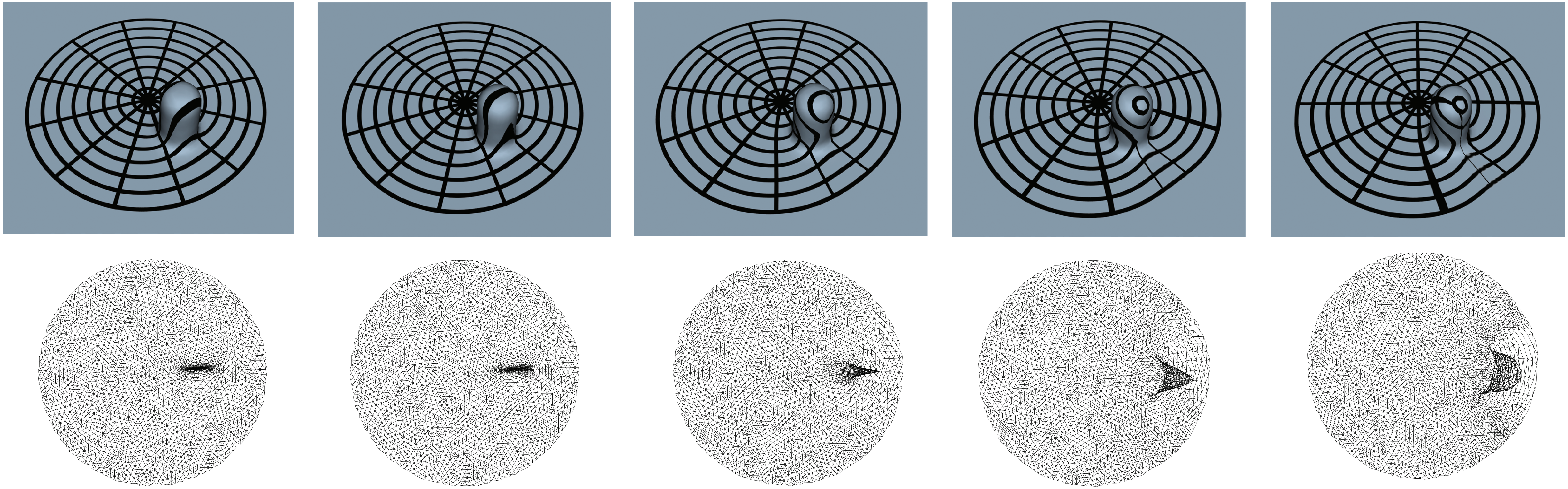
Quality comparison

ours | exact polyhedral geodesics | [Schmidt et al., 2006] | [Melvær et al., 2012]



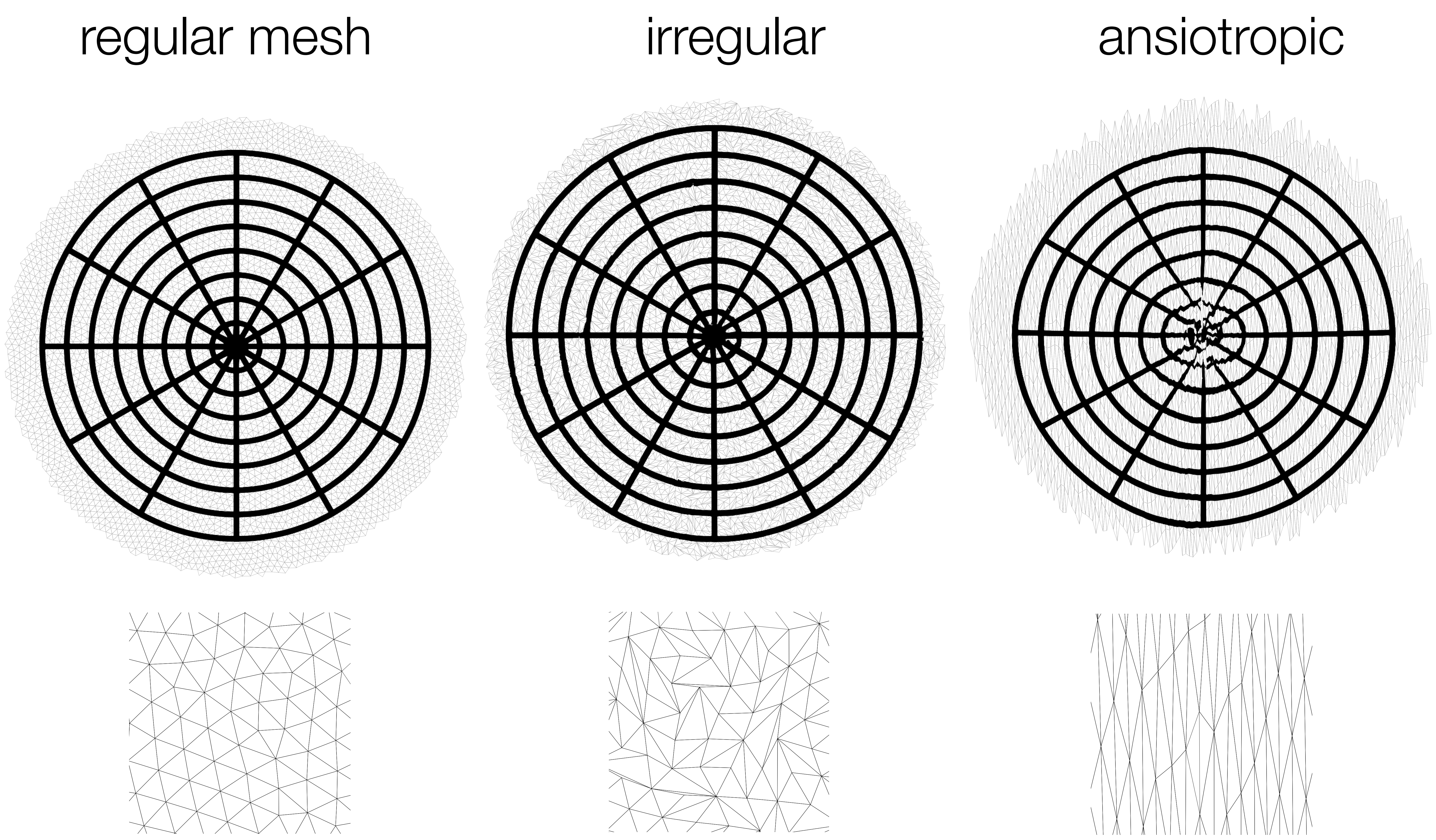
Smooth maps

$t = 10^2 h^2$ $t = 10 h^2$ $t = h^2$ $t = 10^{-1} h^2$ $t = 10^{-2} h^2$



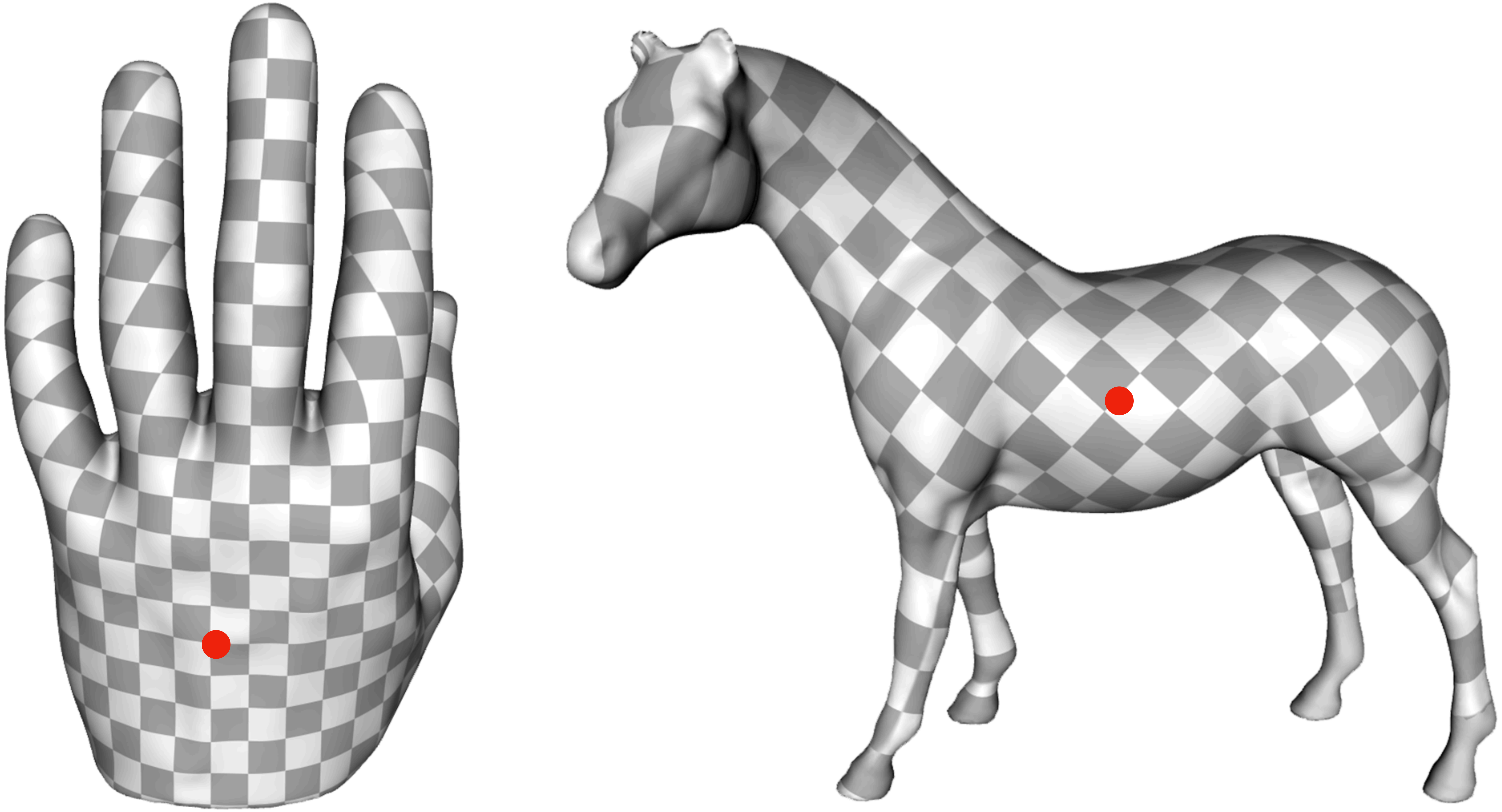
h : average edge length

Mesh quality



Using the intrinsic Delaunay Triangulation we can handle mesh degeneracies to a certain extend.

Global parameterization



Away from the cut locus the maps extend smoothly across the surface.

Summary

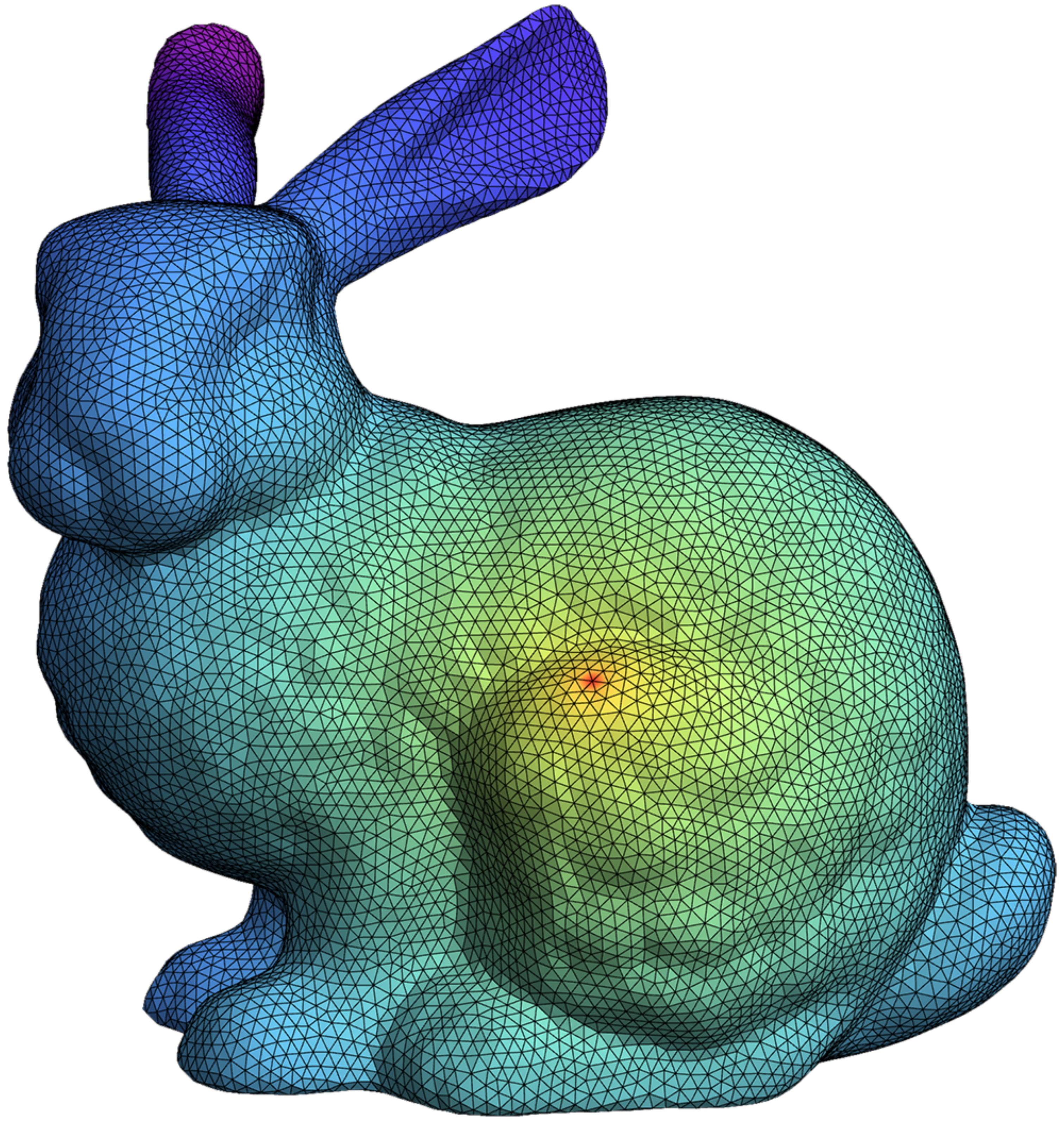
- We can extend diffusion based distance computation to also compute angles.

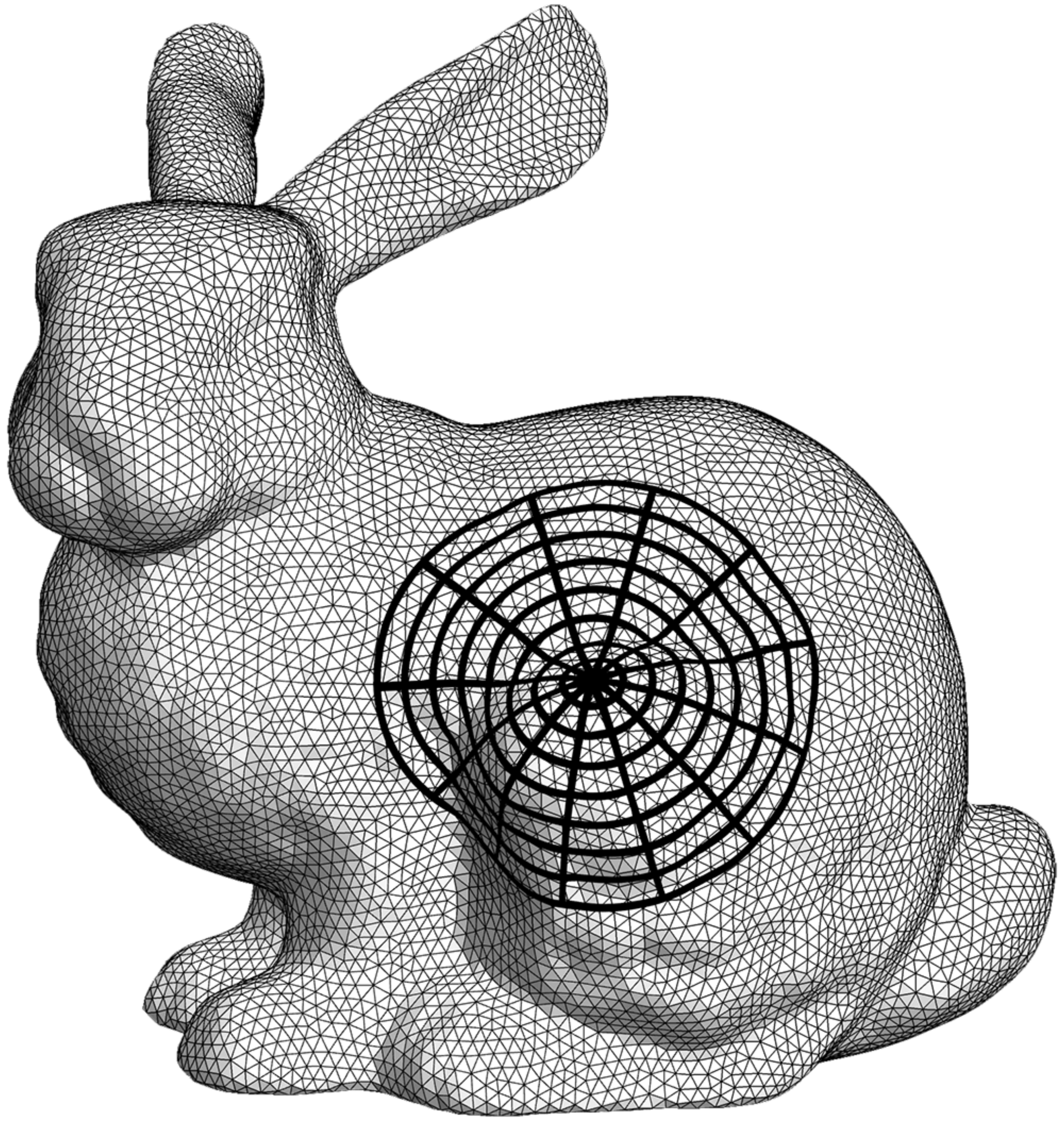
Summary

- We can extend diffusion based distance computation to also compute angles.
- The algorithm uses the same data structures as Geodesics in Heat.

Summary

- We can extend diffusion based distance computation to also compute angles.
- The algorithm uses the same data structures as Geodesics in Heat.
- The method yields smoother maps then Dijkstra based approaches while not being slower for medium sized patches.





Thank you for your attention!

Contact: philipp.herholz@inf.ethz.ch