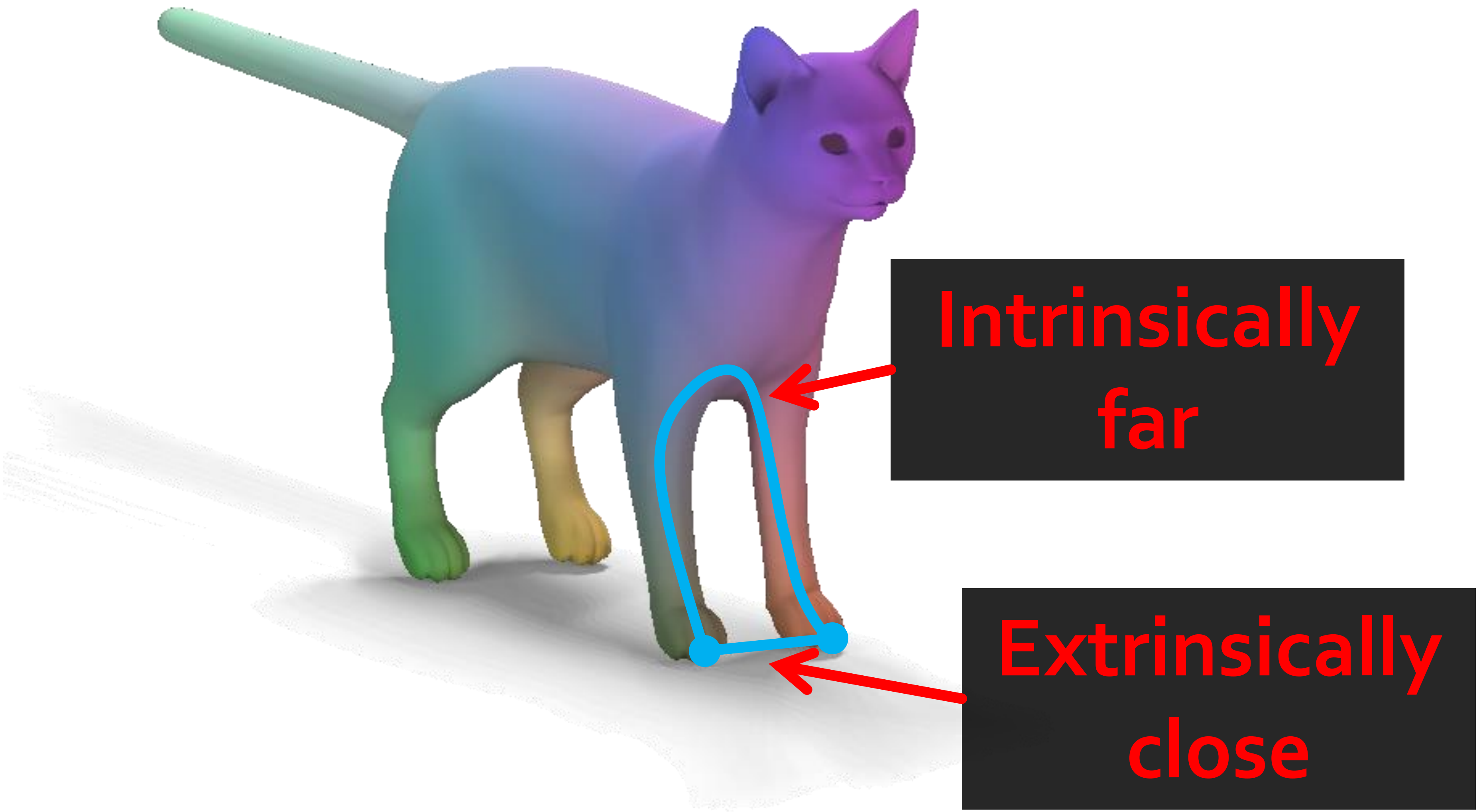
Computing Geodesics

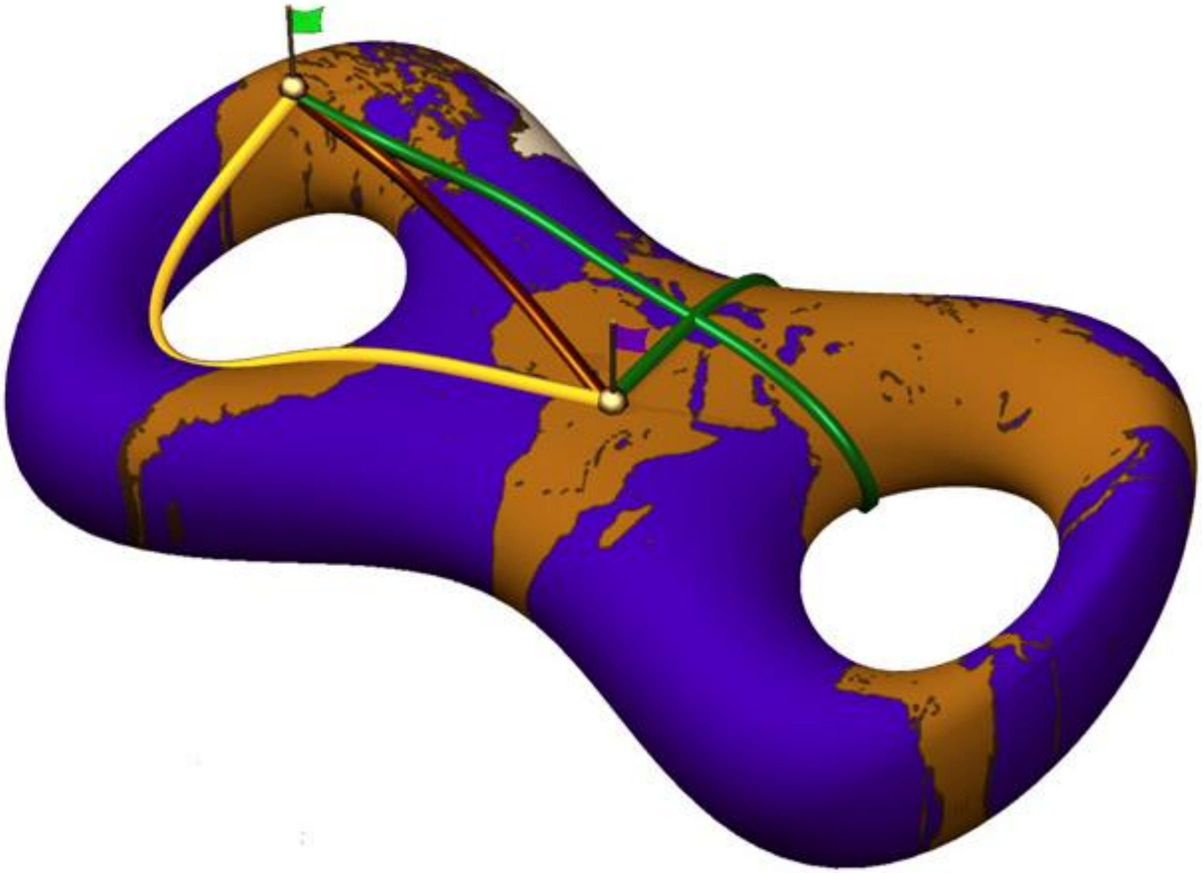
CS 468, Spring 2013
Differential Geometry for Computer Science

Justin Solomon and Adrian Butscher

The Issue



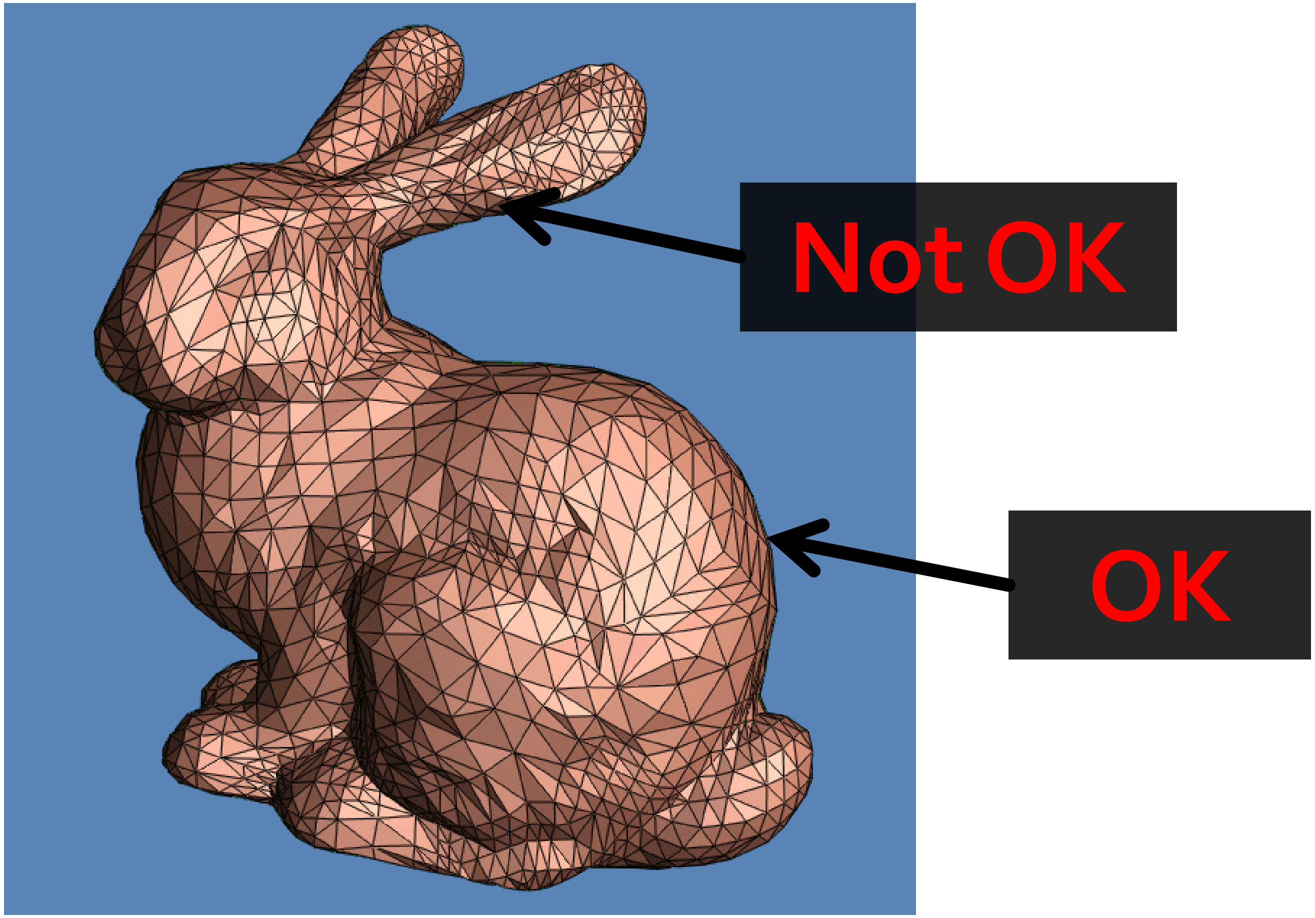
Complicating Factor



Straightest Geodesics on Polyhedral Surfaces (Polthier and Schmies)

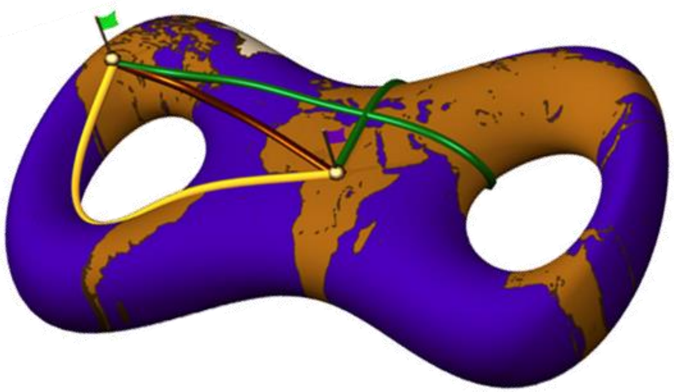
Local vs. global optimality

Reality Check



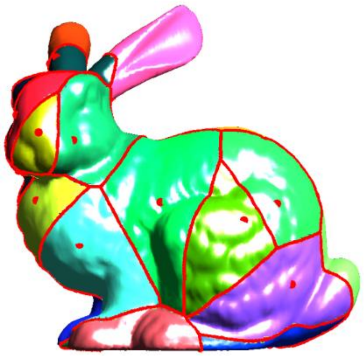
Extrinsic may suffice for near vs. far

Related Queries

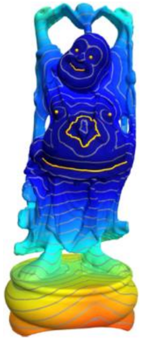


Locally OK

Single source



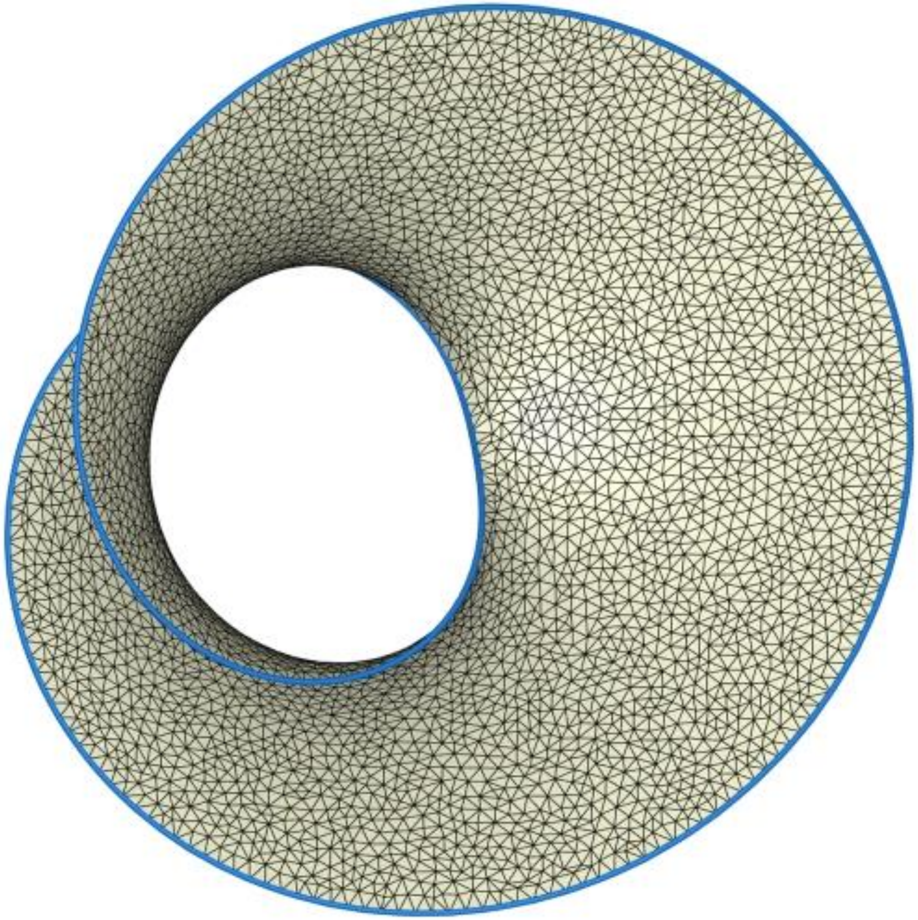
Multi-source



All-pairs

https://www.ceremade.dauphine.fr/~peyre/teaching/manifold/tp3.html http://www.sciencedirect.com/science/article/pii/S0010448511002260

Useful Approximation

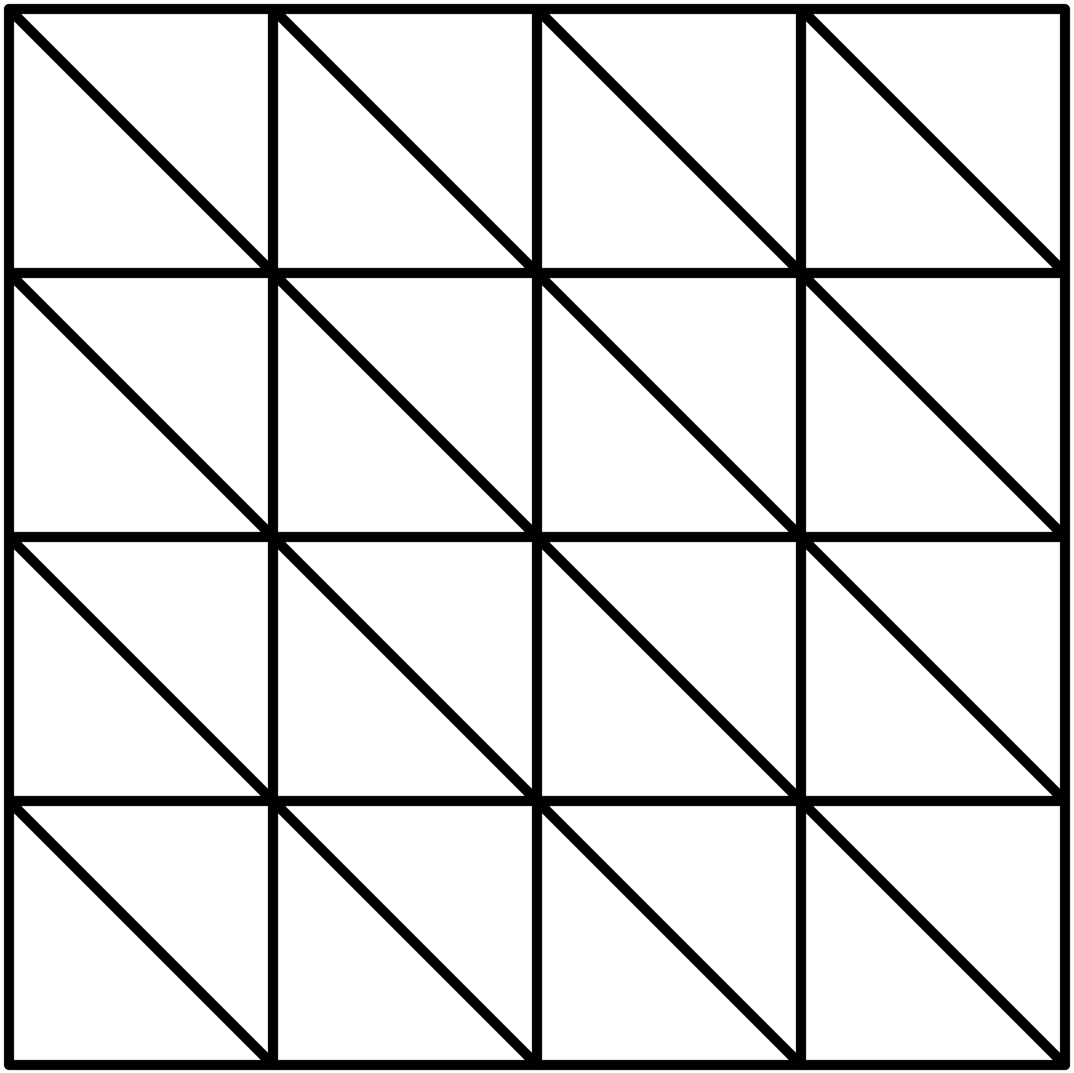


Approximate geodesics as paths along edges

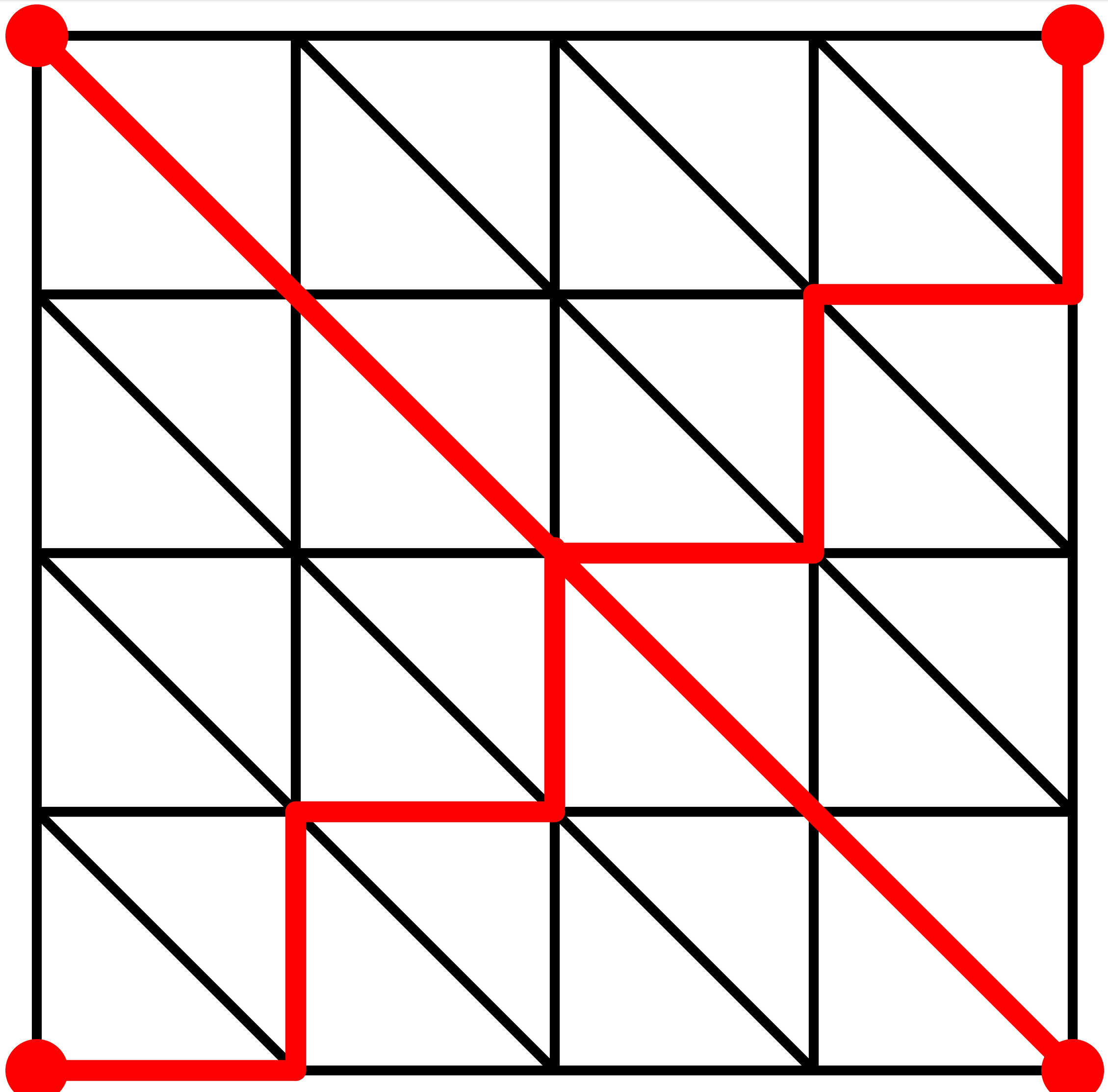
http://www.cse.ohio-state.edu/~tamaldey/isotopic.html

Meshes are graphs

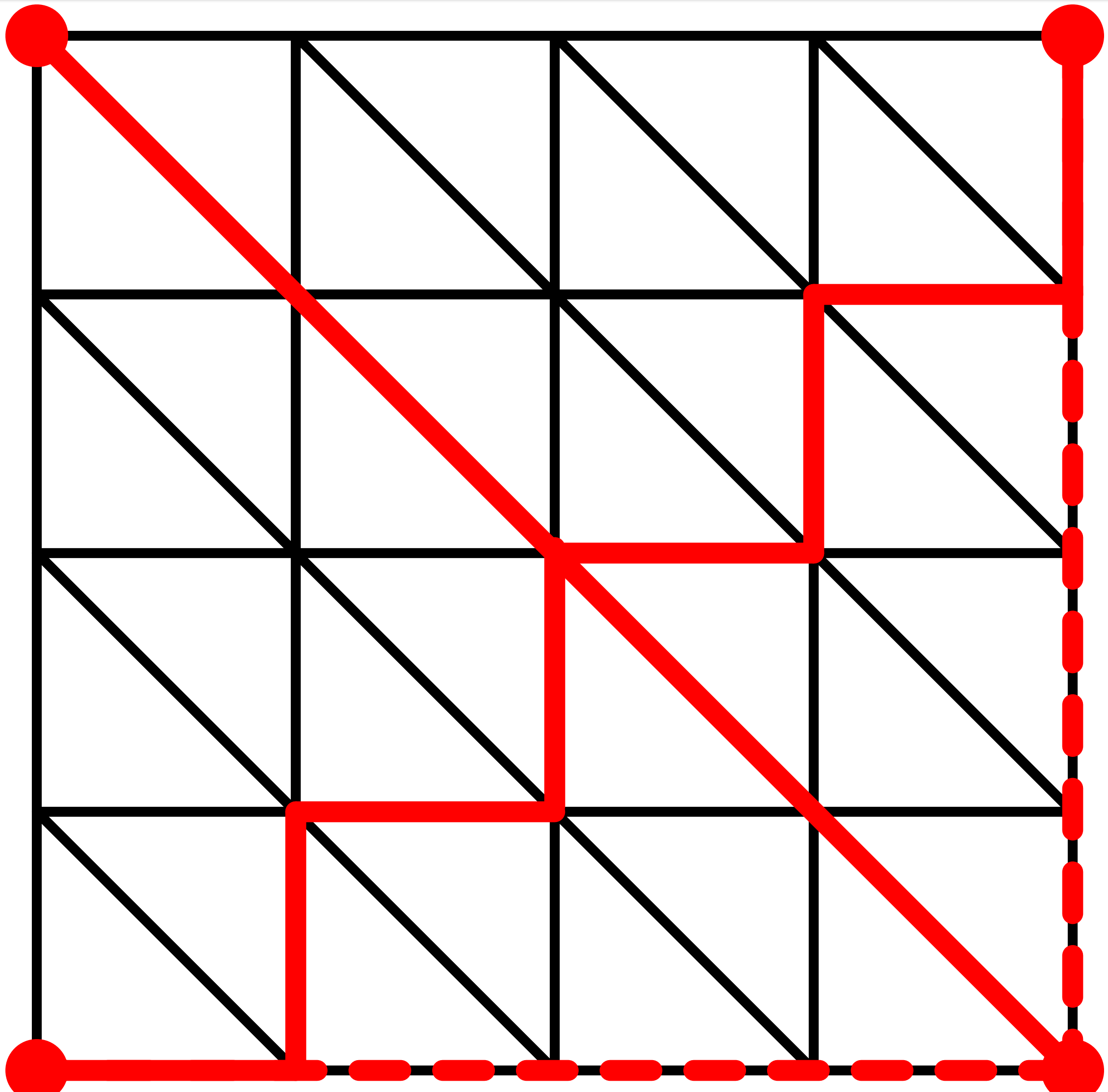
Pernicious Test Case



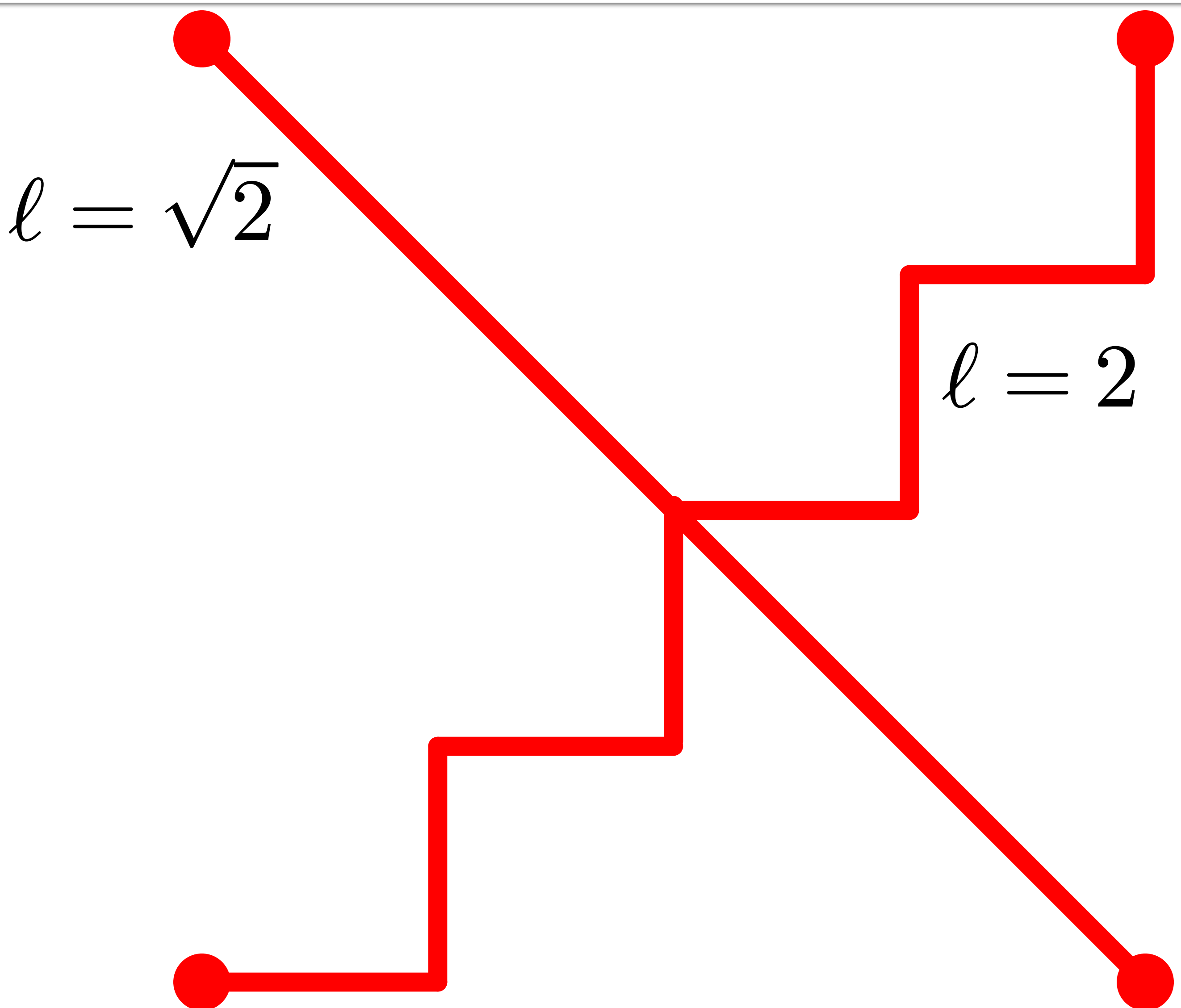
Pernicious Test Case



Pernicious Test Case



Distances



What Happened

- Asymmetric
- Anisotropic
- May not improve under refinement

Conclusion 1

Graph shortest-path does *not* always converge to geodesic distance.

Conclusion 1.5

Graph shortest-path does *not* always converge to geodesic distance.

Often an acceptable approximation.

Conclusion 2

Graph shortest path algorithms are well-understood.

Useful Principles

"Shortest path had to come from somewhere."

"All steps of a shortest path are optimal."

Dijkstra's Algorithm

$$v_0 = \text{Source vertex}$$
$$d_i = \text{Current distance to vertex } i$$
$$S = \text{Vertices with known optimal distance}$$

Initialization:

$$d_0 = 0$$
$$d_i = \infty \ \forall i > 0$$
$$S = \{\}$$

Dijkstra's Algorithm

$$v_0 = \text{Source vertex}$$
$$d_i = \text{Current distance to vertex } i$$
$$S = \text{Vertices with known optimal distance}$$

Iteration *k*:

$$k = \arg\min_{v_k \in V \backslash S} d_k$$
$$S \leftarrow v_k$$
$$d_\ell \leftarrow \min\{d_\ell, d_k + d_{k\ell}\}\ \forall \text{ neighbors } v_\ell \text{ of } v_k$$

Dijkstra's Algorithm

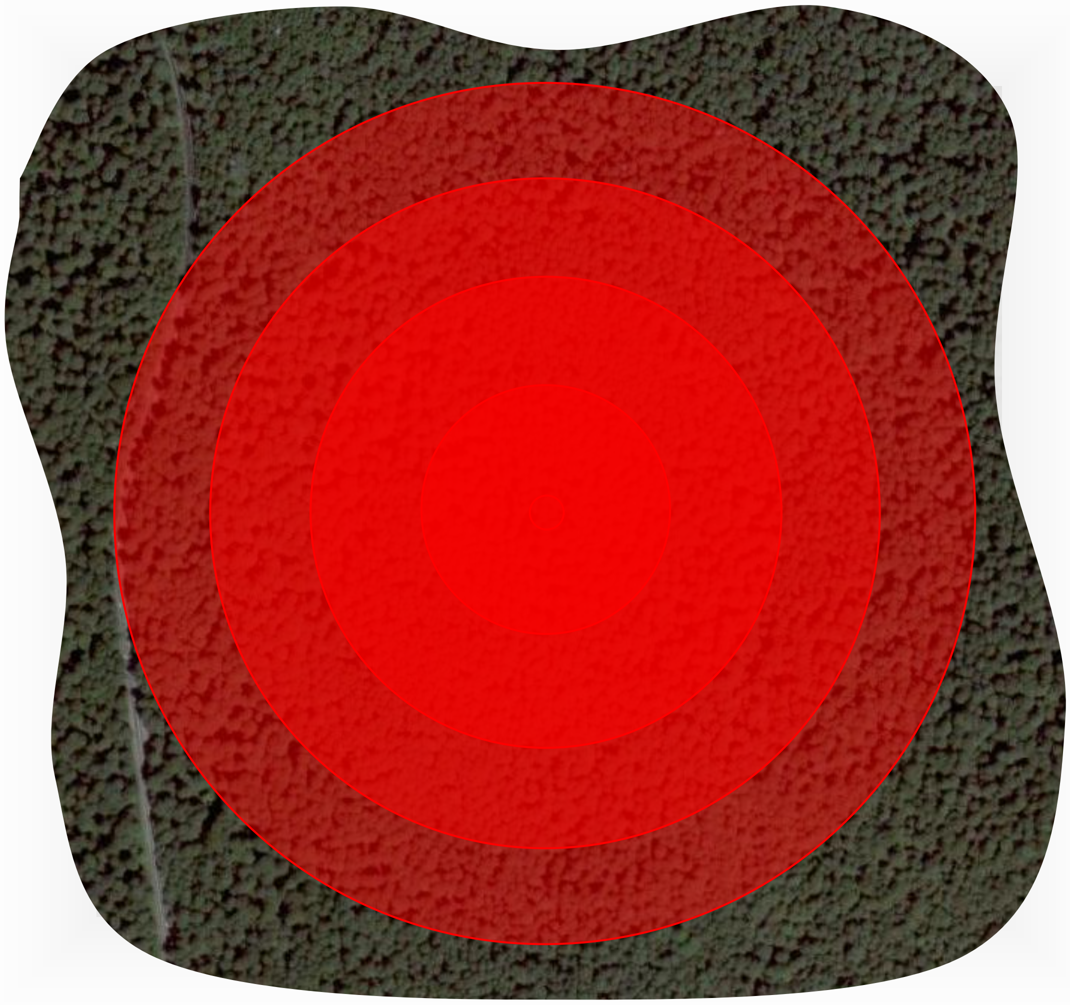
$$v_0 = \text{Source vertex}$$
$$d_i = \text{Current distance to vertex } i$$
$$S = \text{Vertices with known optimal distance}$$

Iteration *k*:

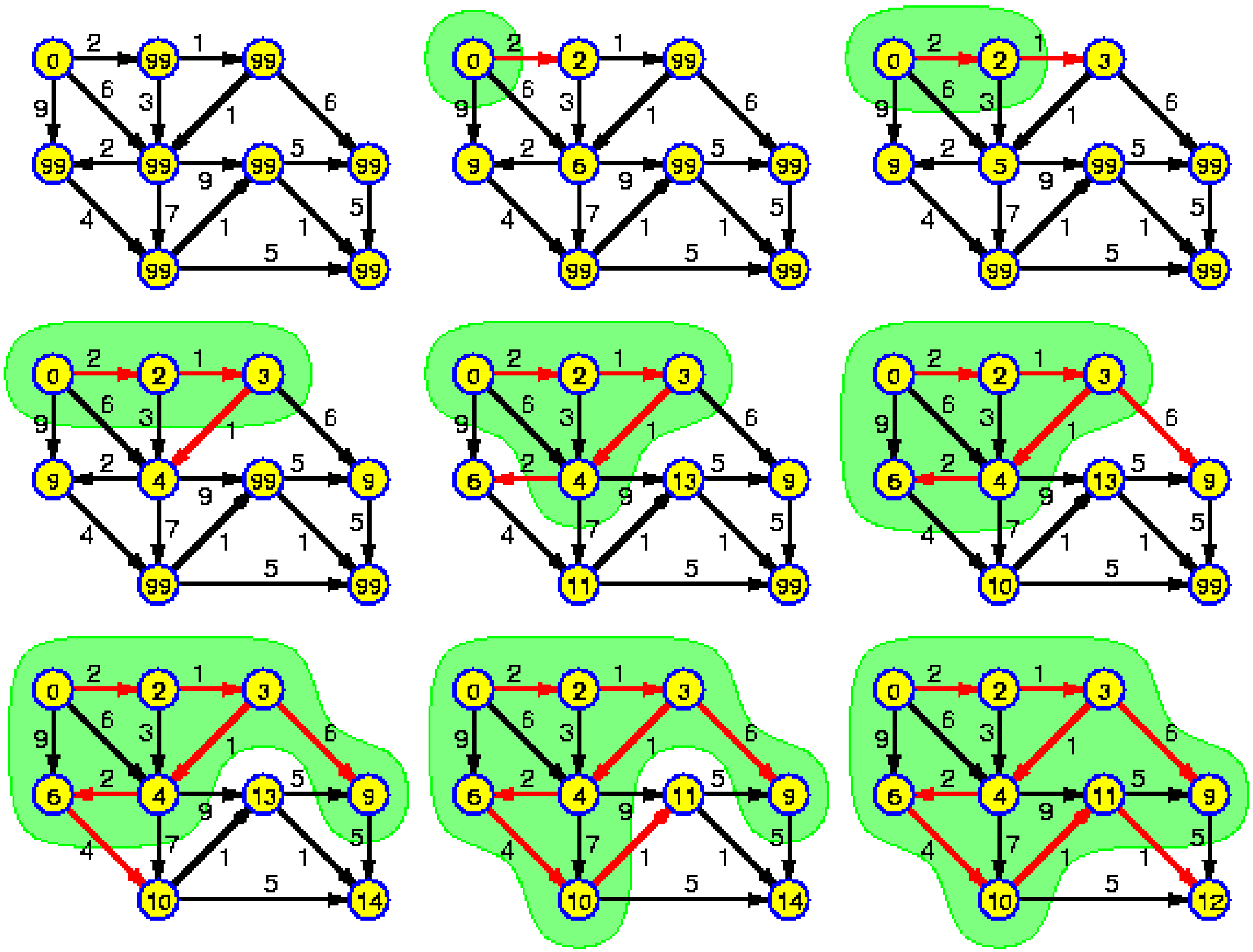
$$k = \arg\min_{v_k \in V \backslash S} d_k$$
$$S \leftarrow v_k$$
$$d_\ell \leftarrow \min\{d_\ell, d_k + d_{k\ell}\}\ \forall \text{ neighbors } v_\ell \text{ of } v_k$$

Inductive proof: **During each iteration, *S* remains optimal.**

Advancing Fronts



Example

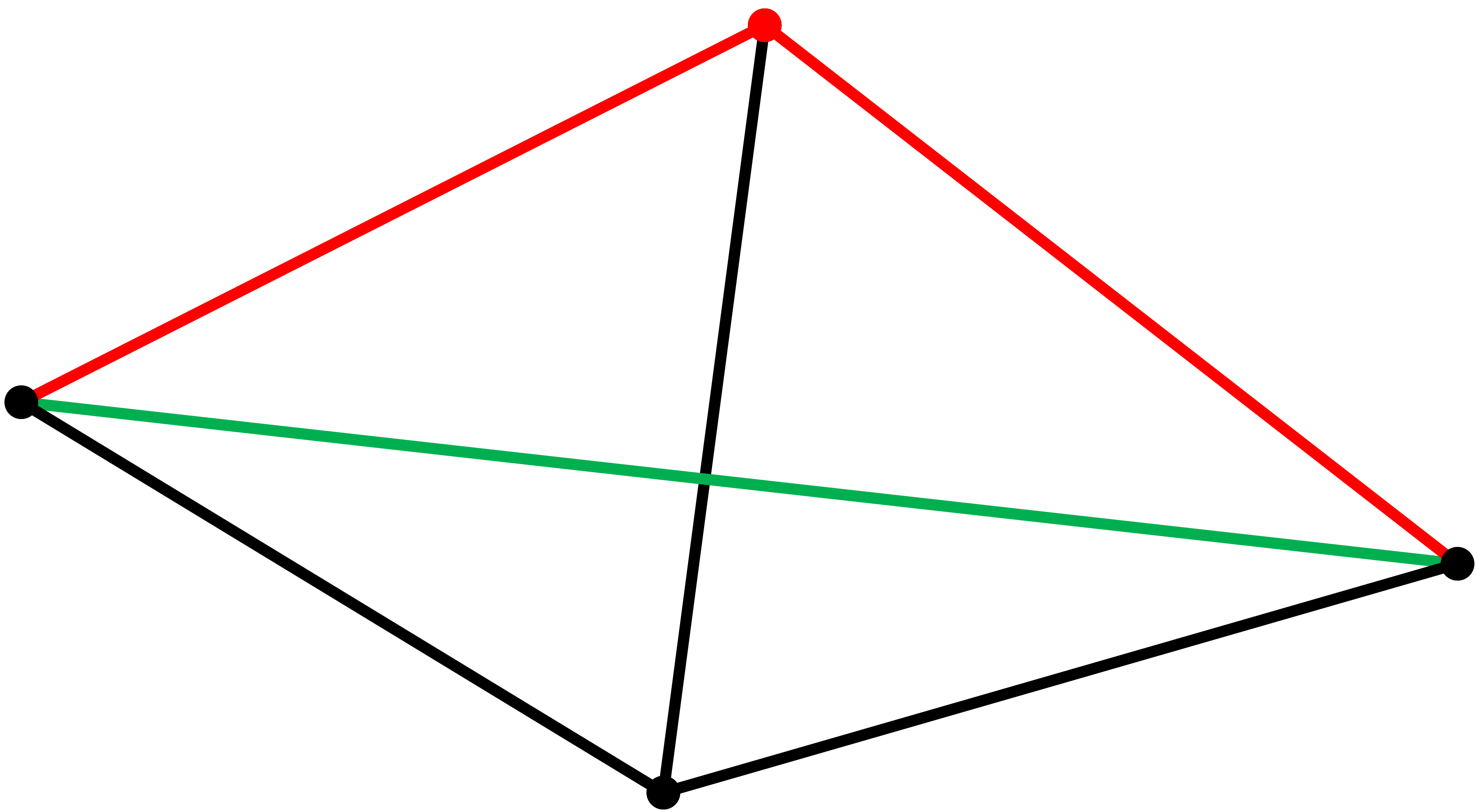


http://www.iekucukcay.com/wp-content/uploads/2011/09/dijkstra.gif

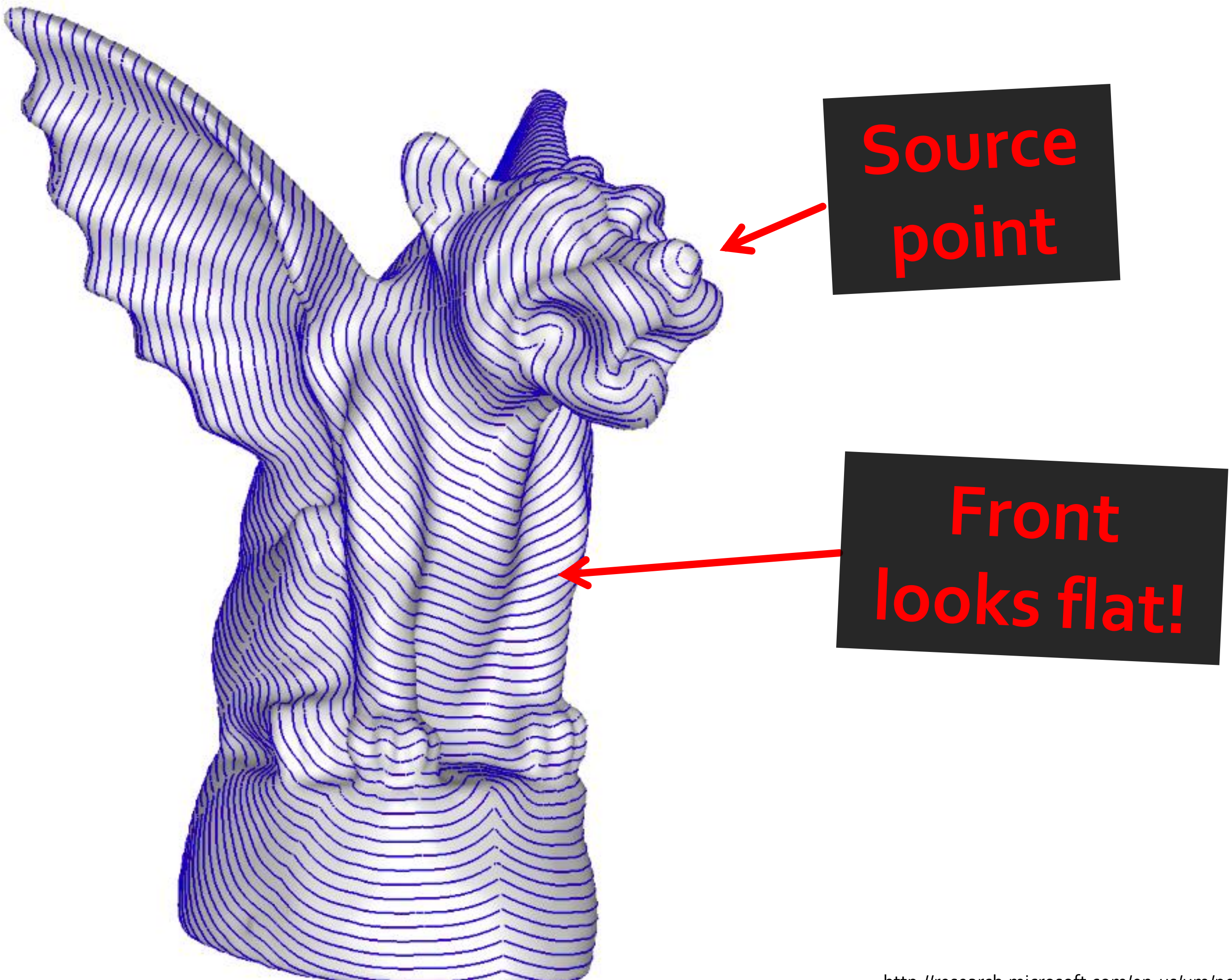
Fast Marching

Dijkstra’s algorithm, modified to approximate geodesic distances.

Problem

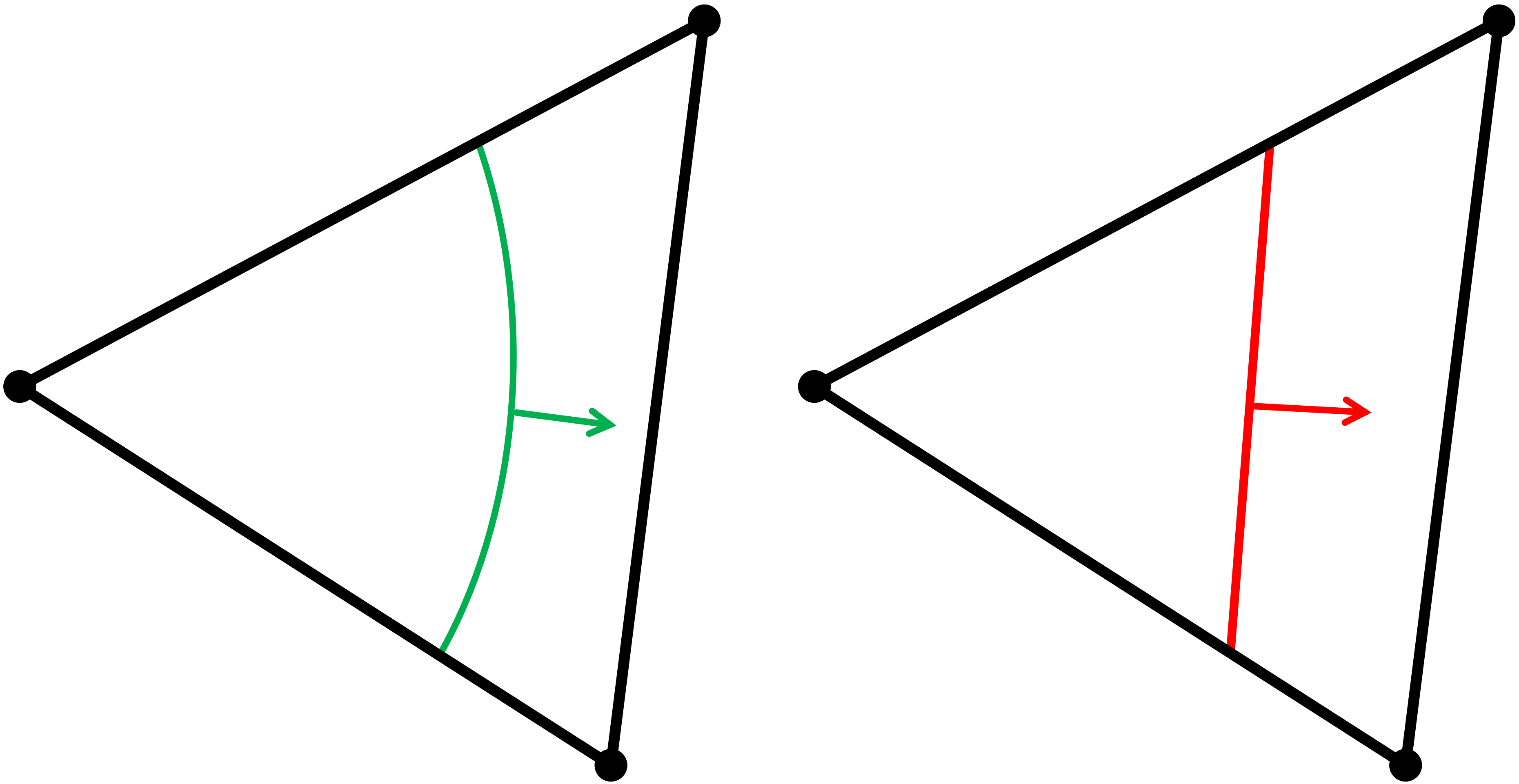


Planar Front Approximation

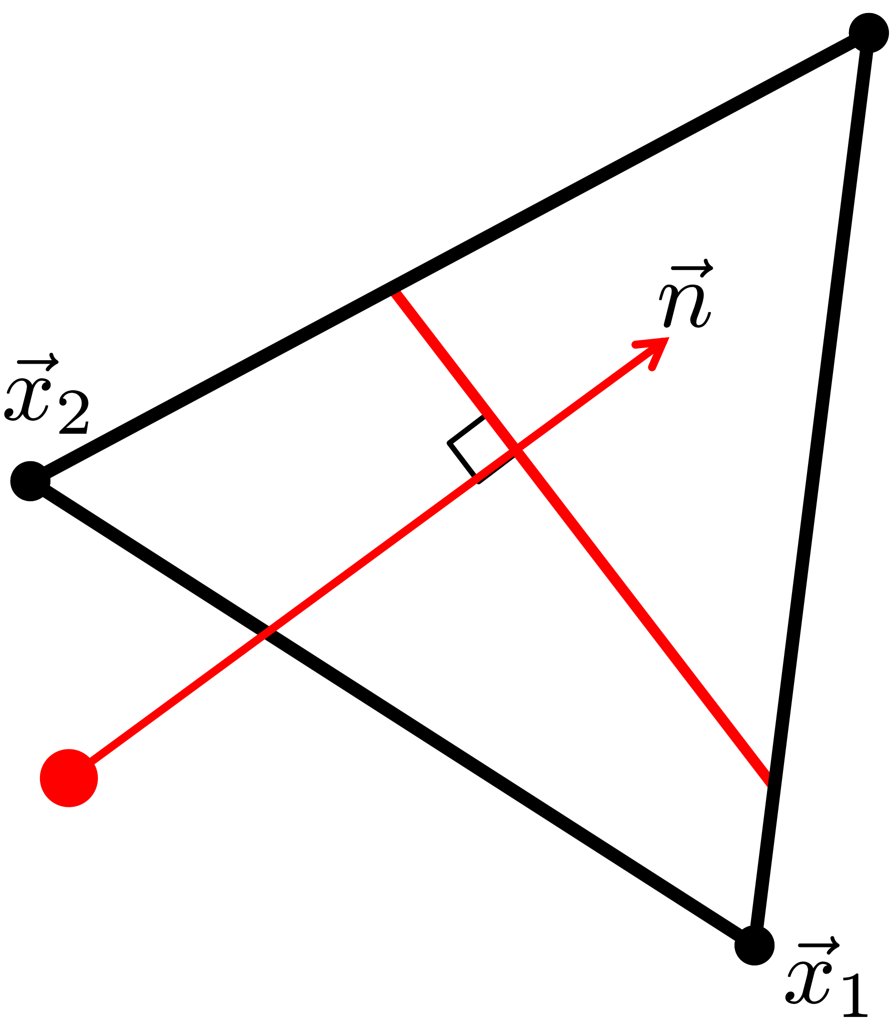


http://research.microsoft.com/en-us/um/people/hoppe/geodesics.pdf

At Local Scale



Planar Calculations



Given:

$$\boxed{\begin{aligned} d_1 &= \vec{n}^\top \vec{x}_1 + p \\ d_2 &= \vec{n}^\top \vec{x}_2 + p \end{aligned}}$$

$$\vec{d} = \vec{n}^\top X + p\mathbf{1}_{2\times 1}$$

Find:

$$\boxed{d_3 = \vec{n}^\top \cancelto{0}{\vec{x}_3} + p \equiv p}$$

Planar Calculations

$$\vec{d} = \vec{n}^\top X + p\mathbf{1}_{2\times 1}$$

$$\downarrow$$

$$\vec{n} = V^{-\top}(\vec{d} - p\mathbf{1}_{2\times 1})$$

$$\begin{aligned}
1 &= \vec{n}^\top \vec{n} \\
&= (\vec{d} - p\mathbf{1}_{2\times 1})^\top X^{-1} X^{-\top} (\vec{d} - p\mathbf{1}_{2\times 1}) \\
&= p^2 \cdot \mathbf{1}_{2\times 1}^\top Q \mathbf{1}_{2\times 1} - 2p \cdot \mathbf{1}_{2\times 1}^\top Q \vec{d} + \vec{d}^\top Q \vec{d}
\end{aligned}$$

$$Q \equiv (X^\top X)^{-1}$$

Planar Calculations

$$\boxed{1 = p^2 \cdot \mathbf{1}_{2\times 1}^\top Q \mathbf{1}_{2\times 1} - 2p \cdot \mathbf{1}_{2\times 1}^\top Q \vec{d} + \vec{d}^\top Q \vec{d}}$$

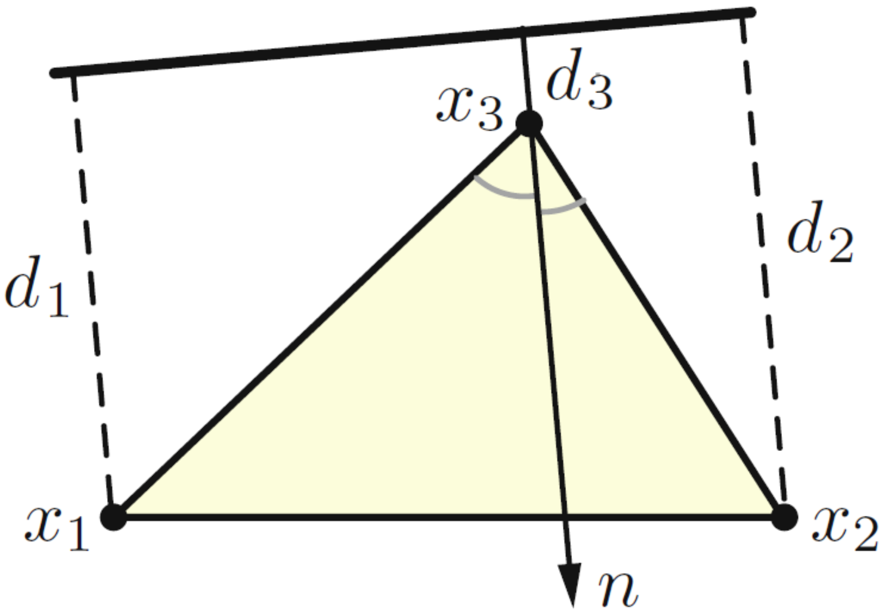
Quadratic equation for *p*

Find:

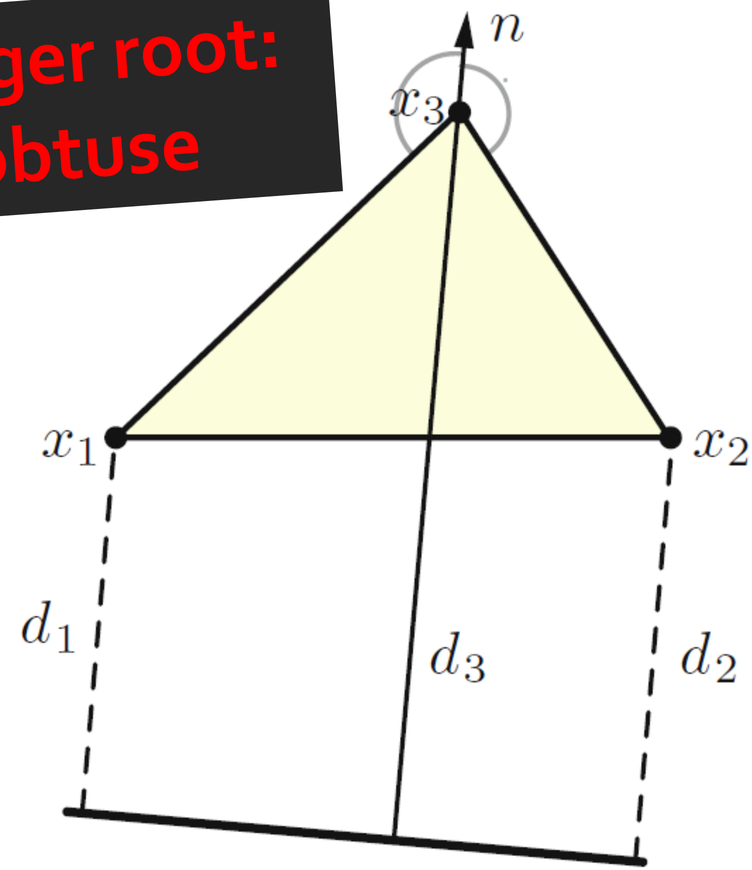
$$\boxed{d_3 = \vec{n}^\top \cancelto{0}{\vec{x}_3} + p \equiv p}$$

Two Roots

Smaller root: acute



Larger root: obtuse



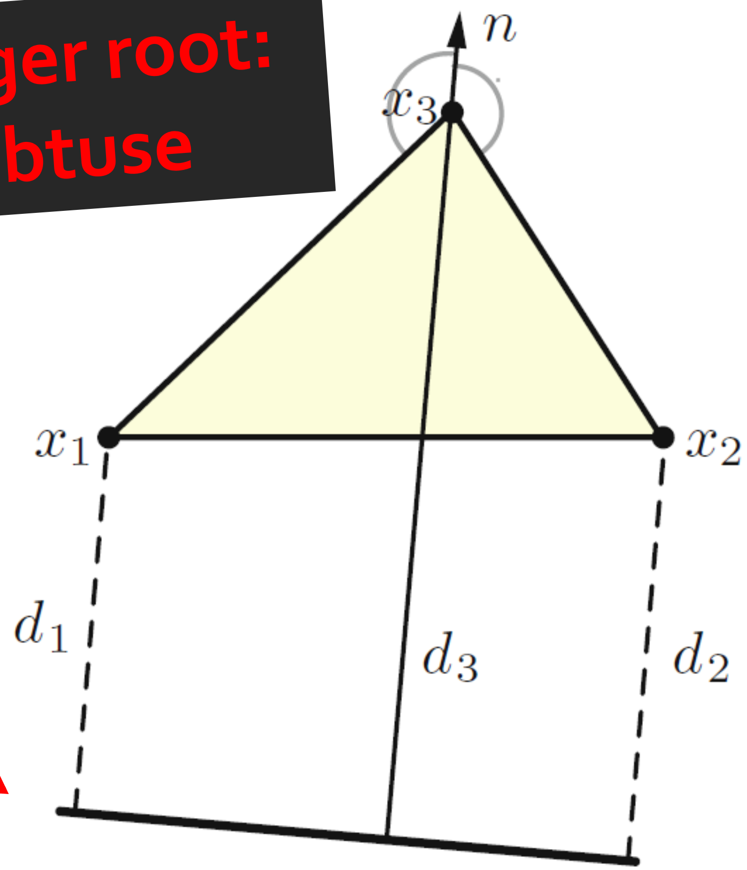
Bronstein, *Numerical Geometry of Nonrigid Shapes*

Two orientations for the normal

Larger Root: Consistent

Larger root: obtuse

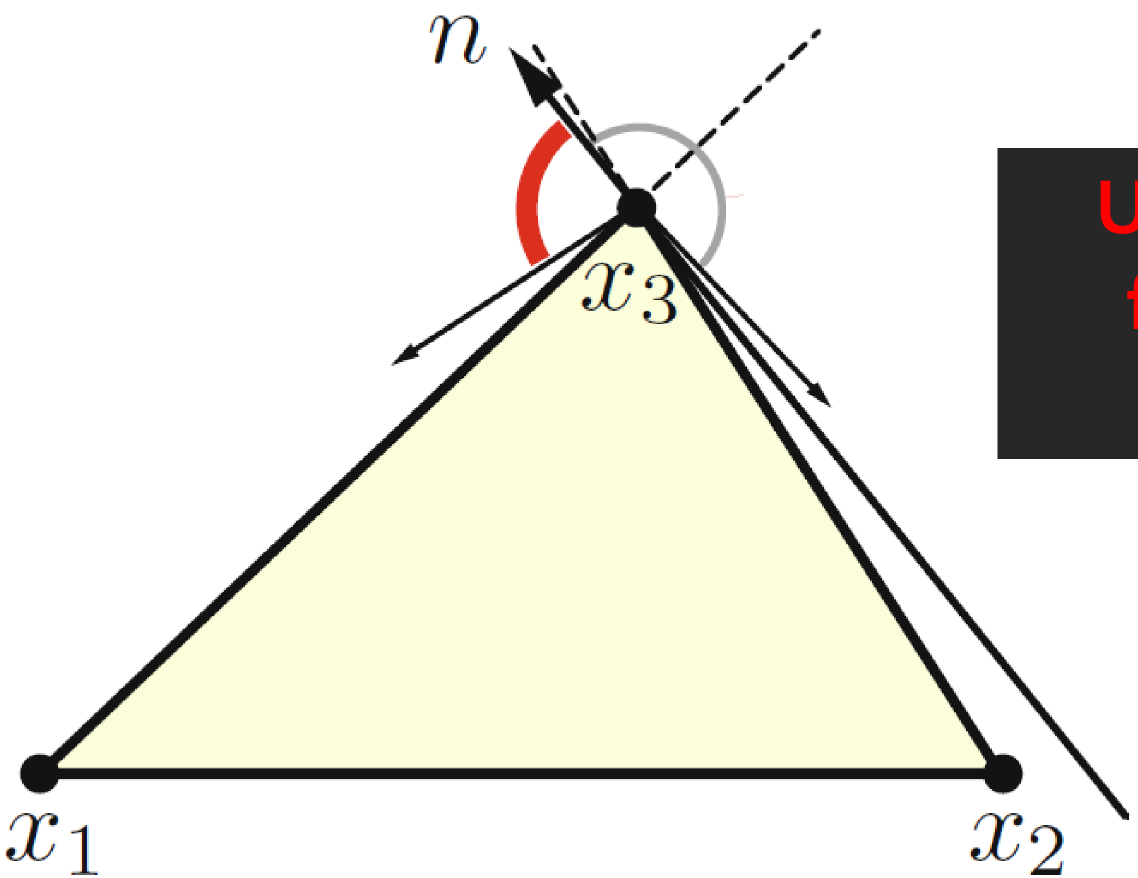
Reaches x_1, x_2 before x_3



Bronstein, *Numerical Geometry of Nonrigid Shapes*

Two orientations for the normal

Additional Issue

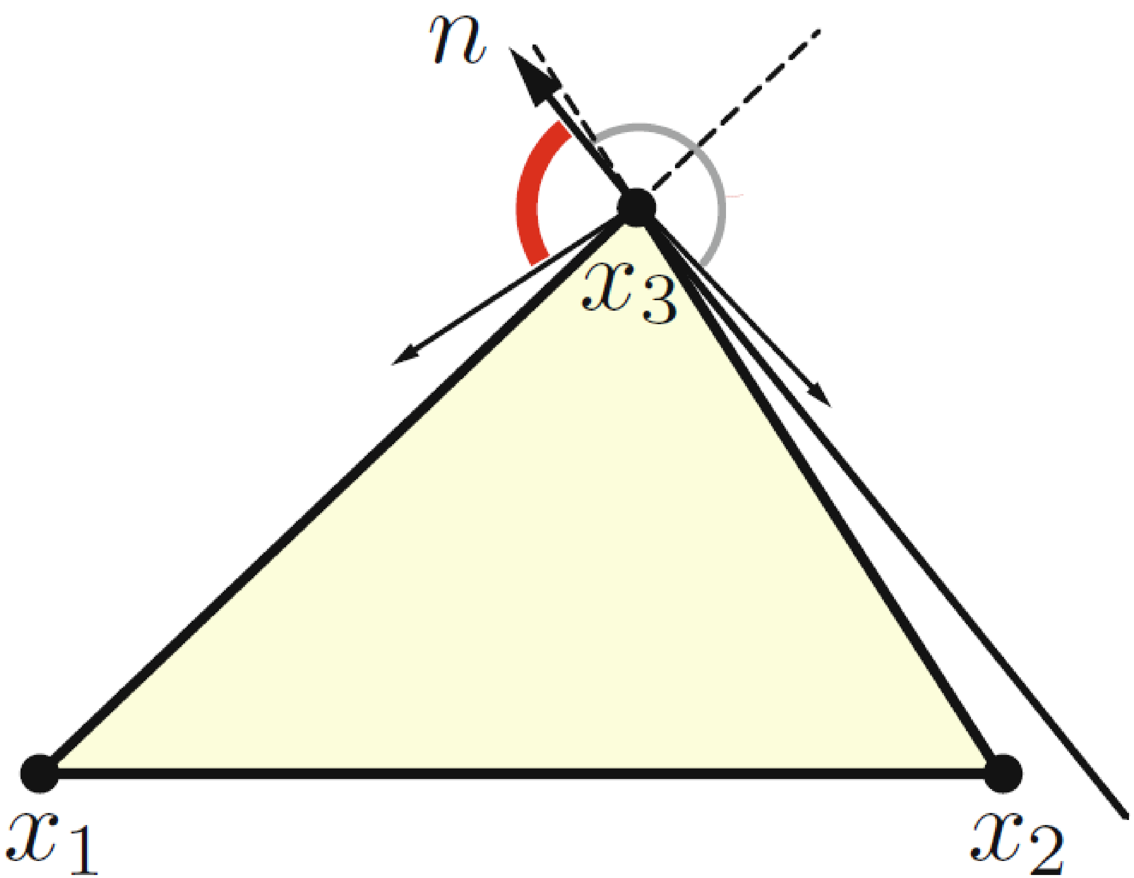


Update should be from a different triangle!

Bronstein, *Numerical Geometry of Nonrigid Shapes*

Front from outside the triangle

Condition for Front Direction



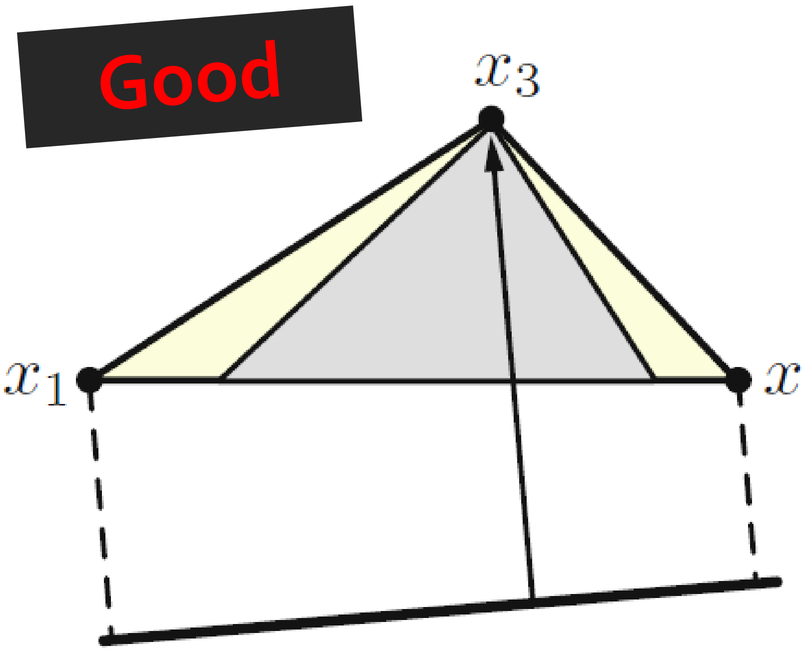
$$\boxed{QX^\top \vec{n} < 0}$$

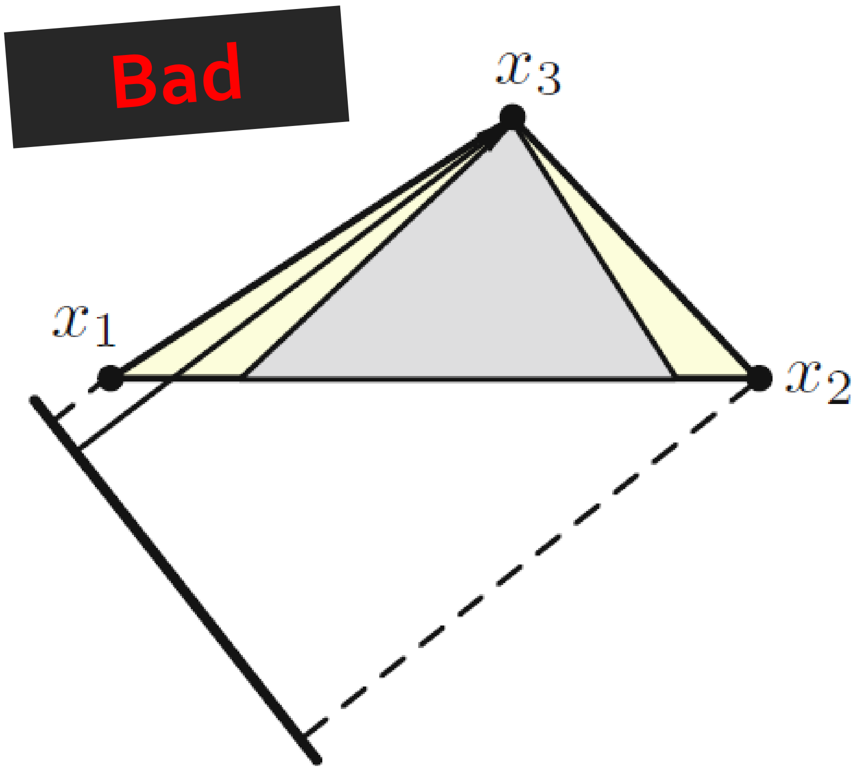
Homework!

Bronstein, *Numerical Geometry of Nonrigid Shapes*

Front from outside the triangle

Obtuse Triangles





Bronstein, *Numerical Geometry of Nonrigid Shapes*

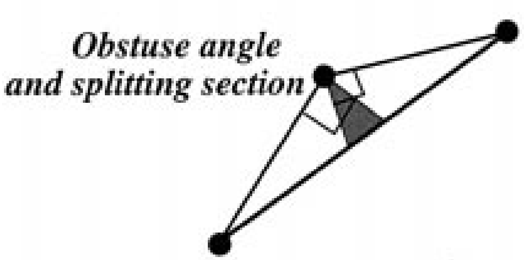
Must reach x_3 after x_1 and x_2

Fixing the Issues

- **Alternative edge-based update:**

$$d_3 \leftarrow \min\{d_3, d_1 + \|x_1\|, d_2 + \|x_2\|\}$$

- **Add connections as needed**
  [Kimmel and Sethian 1998]



Summary: Update Step

input : non-obtuse triangle with the vertices x_1, x_2, x_3, and the corresponding arrival times d_1, d_2, d_3

output : updated d_3

1 Solve the quadratic equation

$$p = \frac{1_{2\times1}^{\mathrm{T}} Q d + \sqrt{(1_{2\times1}^{\mathrm{T}} Q d)^2 - 1_{2\times1}^{\mathrm{T}} Q 1_{2\times1} \cdot (d^{\mathrm{T}} Q d - 1)}}{1_{2\times1}^{\mathrm{T}} Q 1_{2\times1}}.$$

where $V = (x_1 - x_3, x_2 - x_3)$, and $d = (d_1, d_2)^{\mathrm{T}}$.

2 Compute the front propagation direction $n = V^{-\mathrm{T}}(d - p \cdot 1_{2\times1})$

3 **if** $(V^{\mathrm{T}}V)^{-1}V^{\mathrm{T}}n < 0$ **then**

4 $\quad d_3 \longleftarrow \min\{d_3, p\}$

5 **else**

6 $\quad d_3 \longleftarrow \min\{d_3, d_1 + \|x_1\|, d_2 + \|x_2\|\}$

7 **end**

Fast Marching vs. Dijkstra

- Modified update step
- Update all triangles adjacent to a given vertex

Eikonal Equation

$$\boxed{\|\nabla d\| = 1}$$

Greek: "Image"

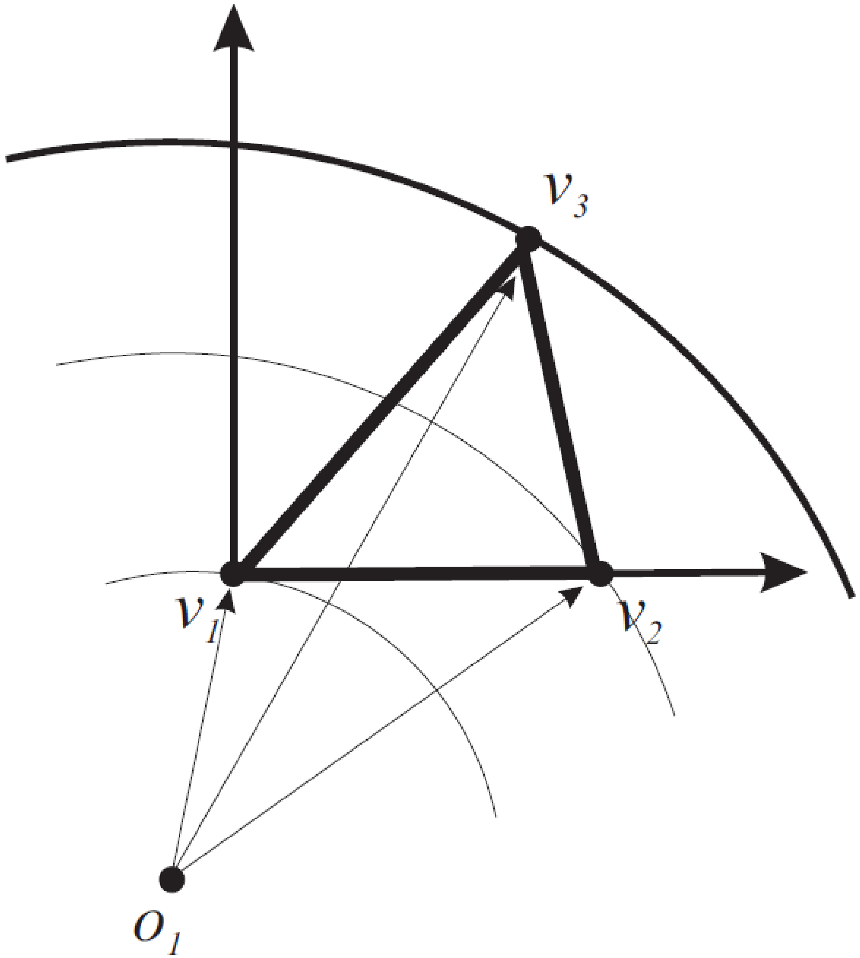
$$1 = \vec{n}^\top \vec{n}$$
$$= (\vec{d} - p\mathbf{1}_{2\times 1})^\top X^{-1} X^{-\top} (\vec{d} - p\mathbf{1}_{2\times 1})$$
$$= p^2 \cdot \mathbf{1}_{2\times 1}^\top Q \mathbf{1}_{2\times 1} - 2p \cdot \mathbf{1}_{2\times 1}^\top Q \vec{d} + \vec{d}^\top Q \vec{d}$$
$$Q \equiv (X^\top X)^{-1}$$

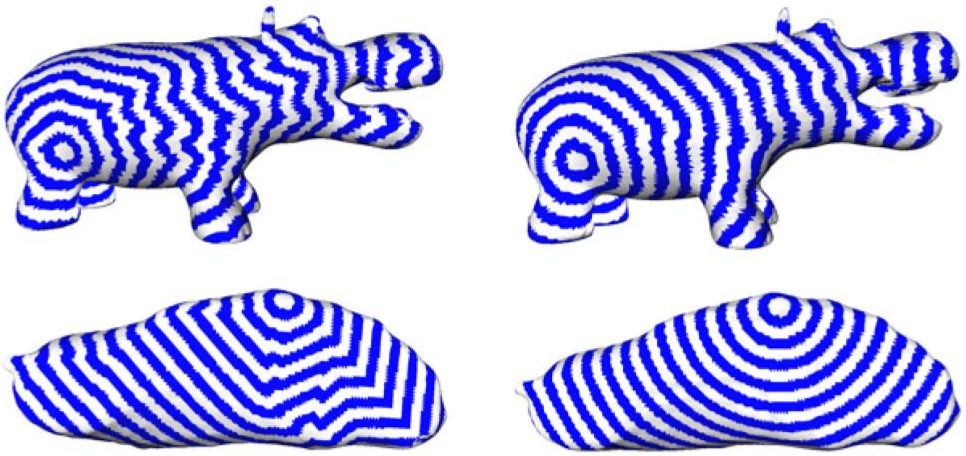
Solutions are geodesic distance


WARNING
STILL AN APPROXIMATION

A much better one!

Modifying Fast Marching

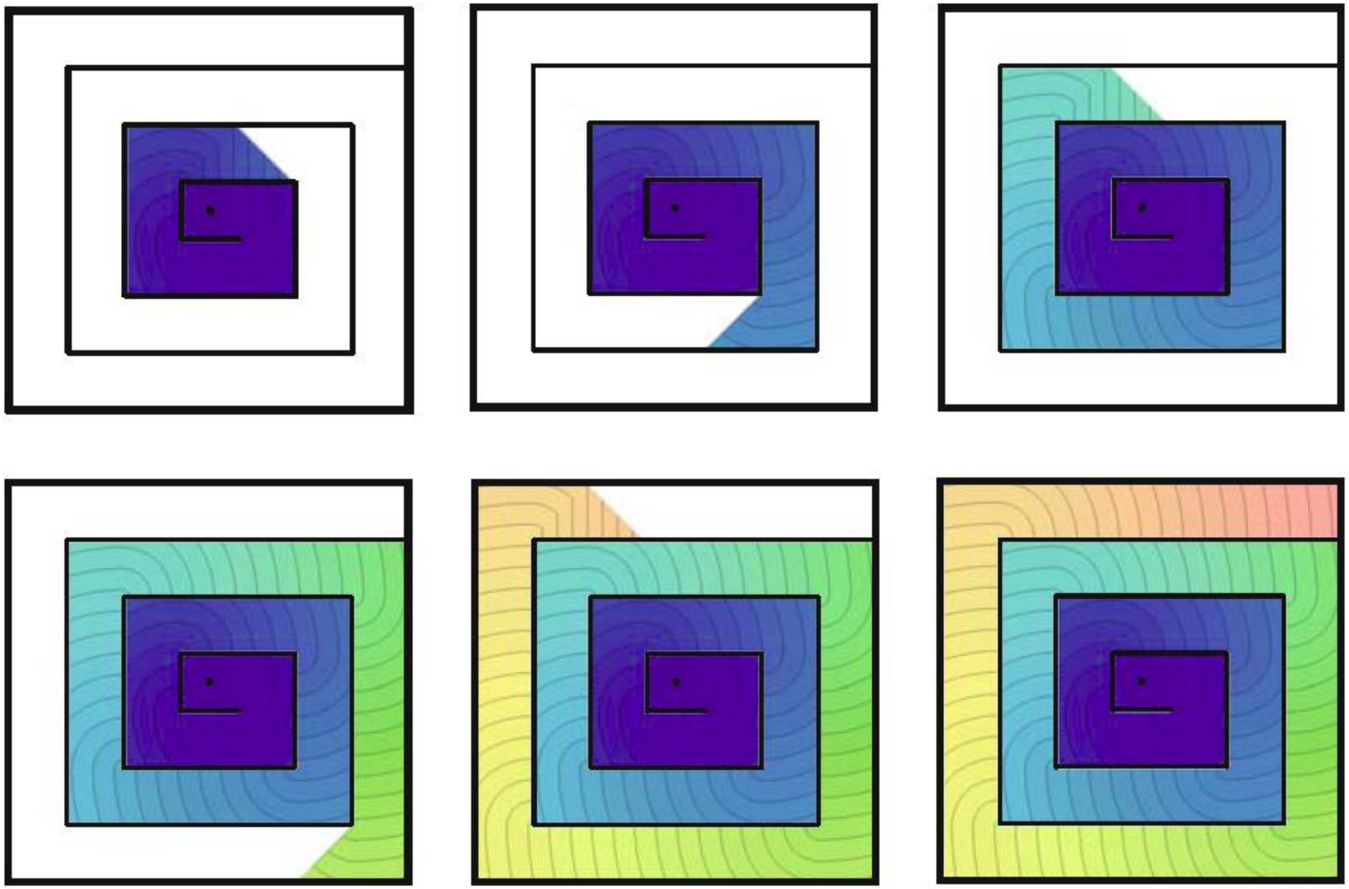




[Novotni and Klein 2002]:

Circular wavefront

Modifying Fast Marching



Raster scan and/or parallelize

Bronstein, *Numerical Geometry of Nonrigid Shapes*

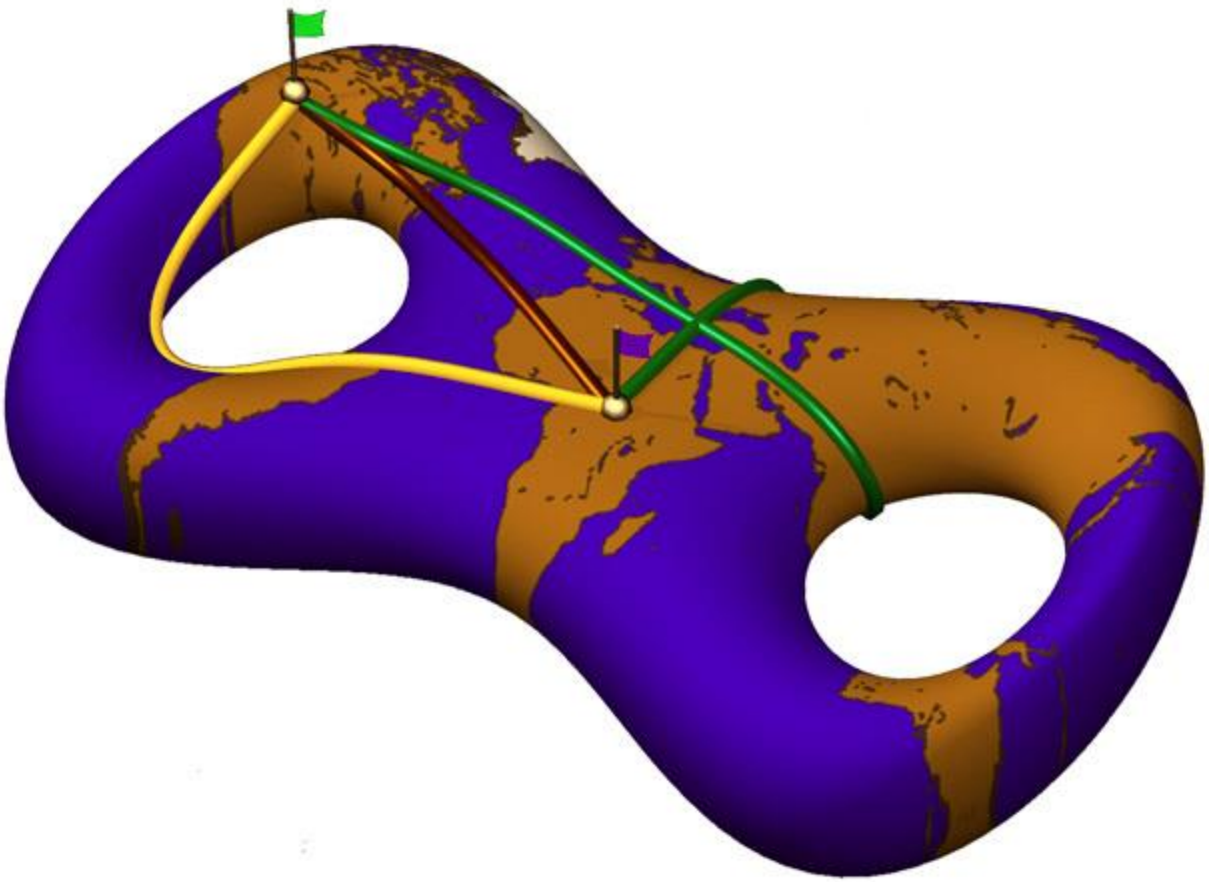
Alternative to Eikonal Equation

Algorithm 1 The Heat Method

I. Integrate the heat flow $\dot{u} = \Delta u$ for time t.
II. Evaluate the vector field $X = -\nabla u / |\nabla u|$.
III. Solve the Poisson equation $\Delta \phi = \nabla \cdot X$.

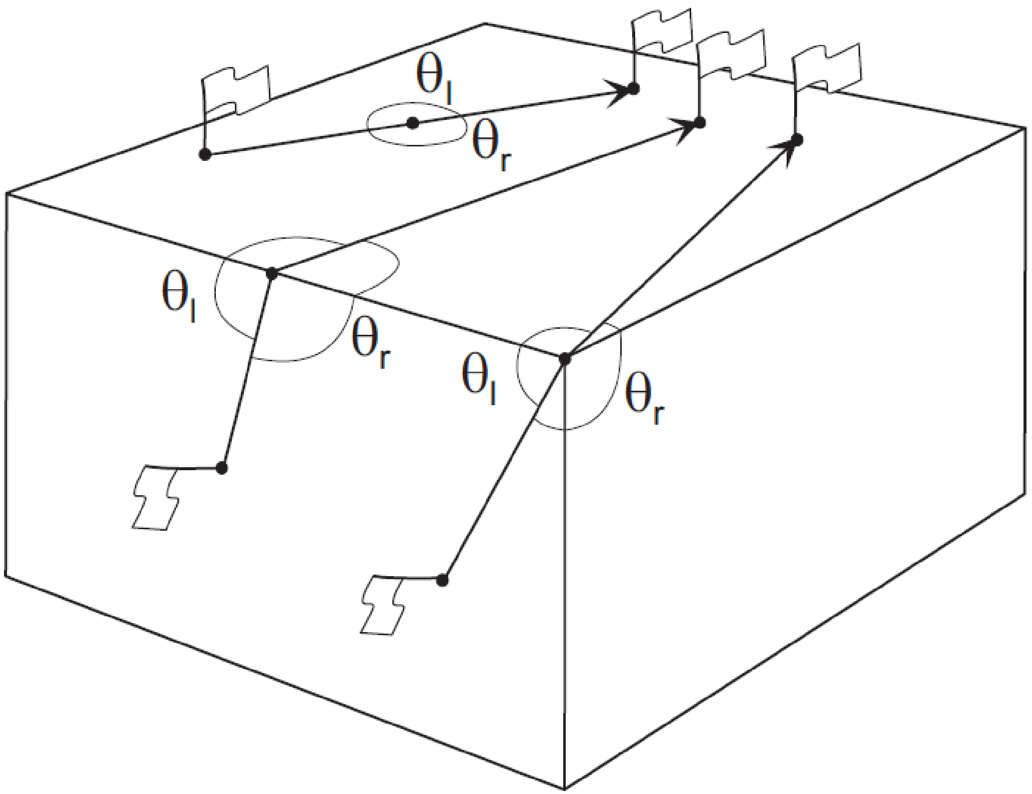
Crane, Weischedel, and Wardetzky. "Geodesics in Heat." TOG, to appear.

Tracing Geodesic Curves



Trace gradient of distance function

Initial Value Problem

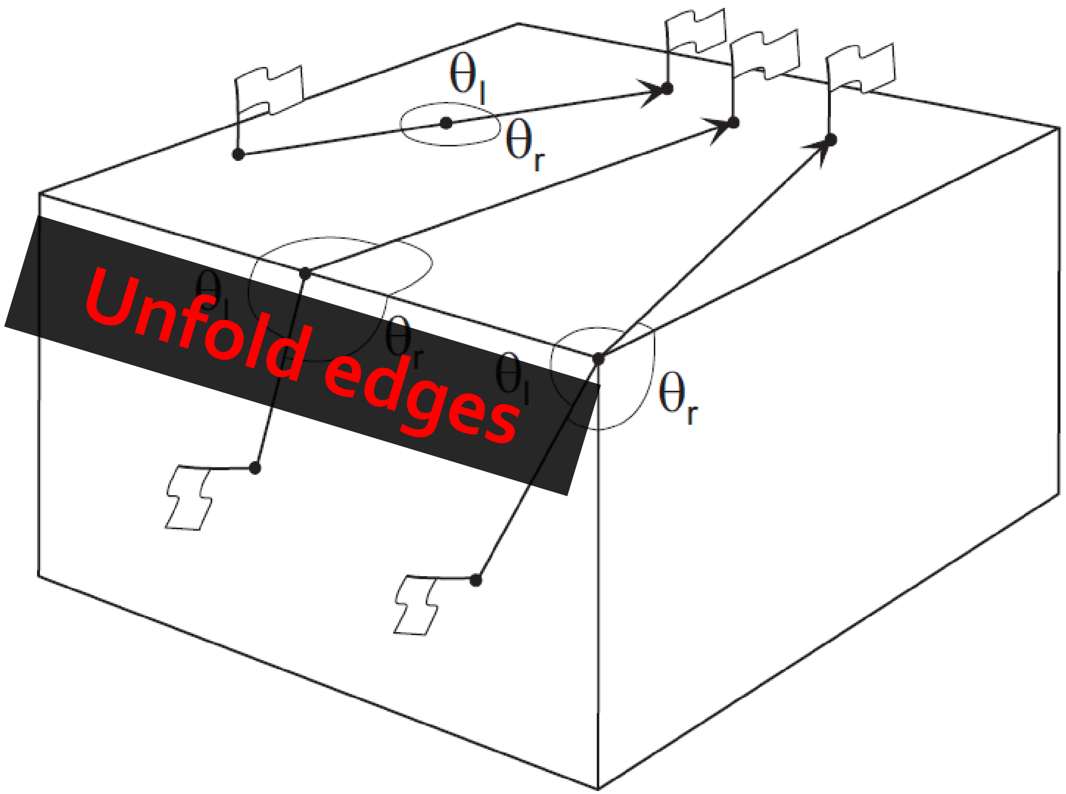


Equal left and right angles

Polthier and Schmies. "Shortest Geodesics on Polyhedral Surfaces." SIGGRAPH course notes 2006.

Trace a single geodesic exactly

Initial Value Problem



Equal left and right angles

Polthier and Schmies. "Shortest Geodesics on Polyhedral Surfaces." SIGGRAPH course notes 2006.

Trace a single geodesic exactly

Exact Geodesics

SIAM J. COMPUT.
Vol. 16, No. 4, August 1987

THE DISCRETE GEODESIC PROBLEM*

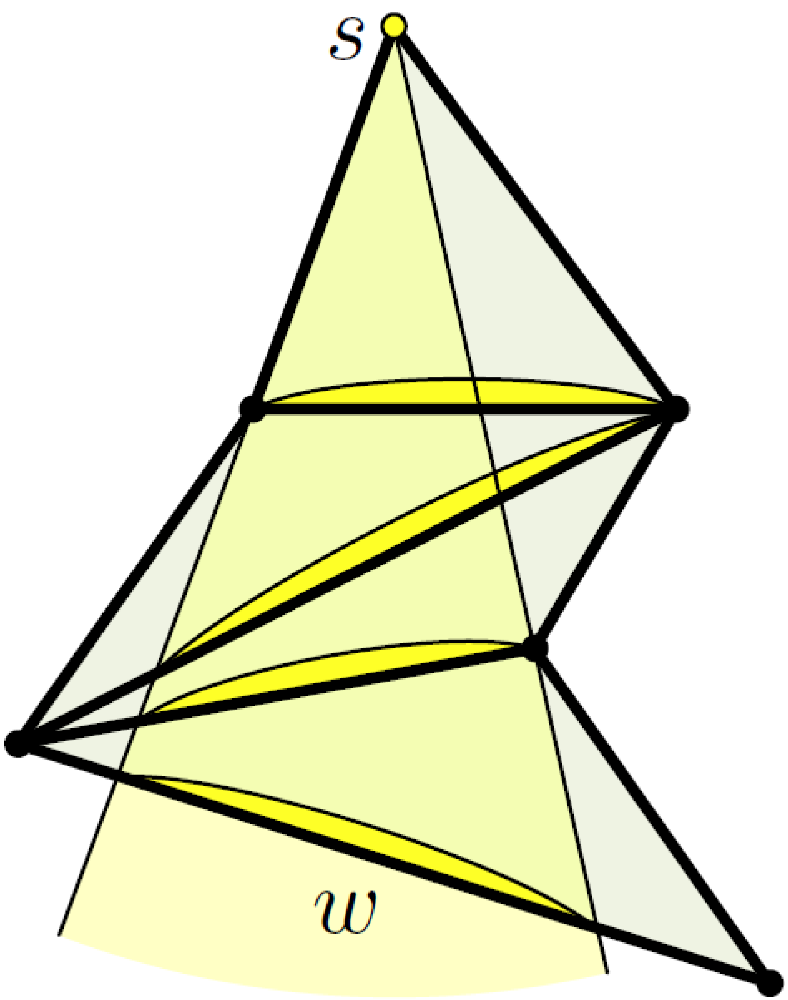
JOSEPH S. B. MITCHELL†, DAVID M. MOUNT‡ AND CHRISTOS H. PAPADIMITRIOU§

Abstract. We present an algorithm for determining the shortest path between a source and a destination on an arbitrary (possibly nonconvex) polyhedral surface. The path is constrained to lie on the surface, and distances are measured according to the Euclidean metric. Our algorithm runs in time $O(n^2 \log n)$ and requires $O(n^2)$ space, where n is the number of edges of the surface. After we run our algorithm, the distance from the source to any other destination may be determined using standard techniques in time $O(\log n)$ by locating the destination in the subdivision created by the algorithm. The actual shortest path from the source to a destination can be reported in time $O(k+\log n)$, where k is the number of faces crossed by the path. The algorithm generalizes to the case of multiple source points to build the Voronoi diagram on the surface, where n is now the maximum of the number of vertices and the number of sources.

Key words. shortest paths, computational geometry, geodesics, Dijkstra's algorithm

AMS(MOS) subject classification. 68E99

MMP Algorithm: Big Idea



Dijkstra-style front with *windows* explaining source.

Surazhsky et al. "Fast Exact and Approximate Geodesics on Meshes." SIGGRAPH 2005.

Practical Implementation

Fast Exact and Approximate Geodesics on Meshes

Vitaly Surazhsky, University of Oslo
Tatiana Surazhsky, University of Oslo
Danil Kirsanov, Harvard University
Steven J. Gortler, Harvard University
Hugues Hoppe, Microsoft Research

Abstract

The computation of geodesic paths and distances on triangle meshes is a common operation in many computer graphics applications. We present several practical algorithms for computing such geodesics from a source point to one or all other points efficiently. First, we describe an implementation of the exact "single source, all destination" algorithm presented by Mitchell, Mount, and Papadimitriou (MMP). We show that the algorithm runs much faster in practice than suggested by worst case analysis. Next, we extend the algorithm with a merging operation to obtain computationally efficient and accurate approximations with bounded error. Finally, to compute the shortest path between two given points, we use a lower-bound property of our approximate geodesic algorithm to efficiently prune the frontier of the MMP algorithm, thereby obtaining an exact solution even more quickly.

Keywords: shortest path, geodesic distance.

1 Introduction

In this paper we present practical methods for computing both exact and approximate shortest (i.e. geodesic) paths on a triangle mesh. These geodesic paths typically cut across faces in the mesh and are therefore not found by the traditional graph-based Dijkstra algorithm for shortest paths.

The computation of geodesic paths is a common operation in many computer graphics applications. For example, parameterizing a mesh often involves cutting the mesh into one or more charts (e.g. [Krishnamurthy and Levoy 1996; Sander et al. 2003]), and

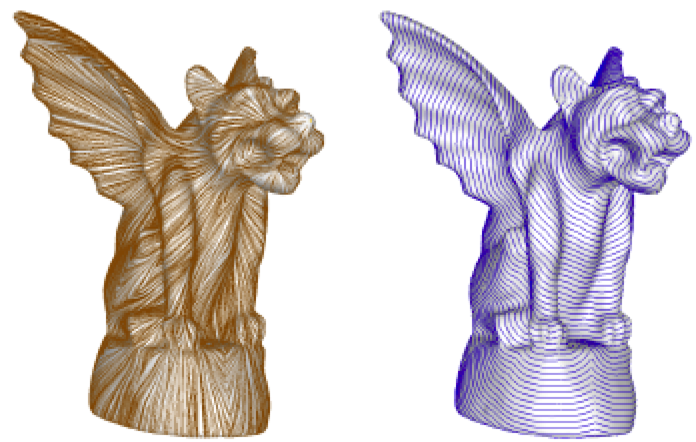


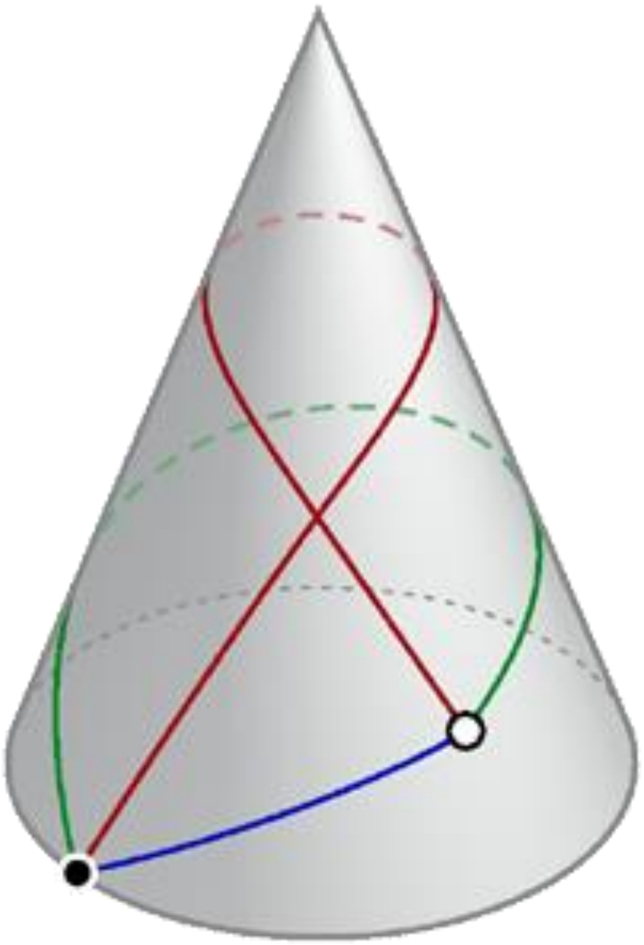
Figure 1: Geodesic paths from a source vertex, and isolines of the geodesic distance function.

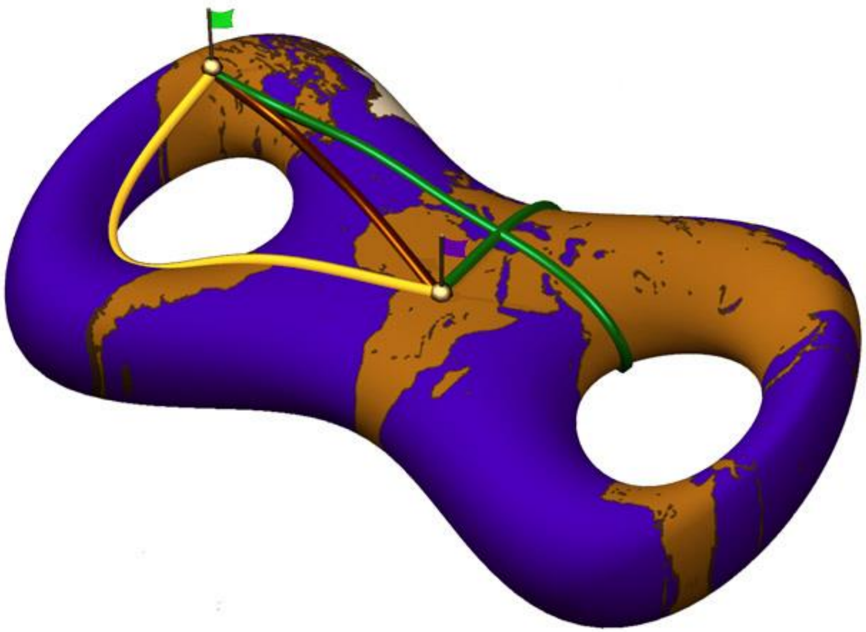
tance function over the edges, the implementation is actually practical even though, to our knowledge, it has never been done previously. We demonstrate that the algorithm's worst case running time of $O(n^2 \log n)$ is pessimistic, and that in practice, the algorithm runs in sub-quadratic time. For instance, we can compute the exact geodesic distance from a source point to all vertices of a 400K-triangle mesh in about one minute.

Approximation algorithm We extend the algorithm with a merging operation ... approximations with *bounded* error. In practice, the algorithm runs in $O(n \log n)$ time even for small error thresholds.

http://code.google.com/p/geodesic/

Instability of Geodesics

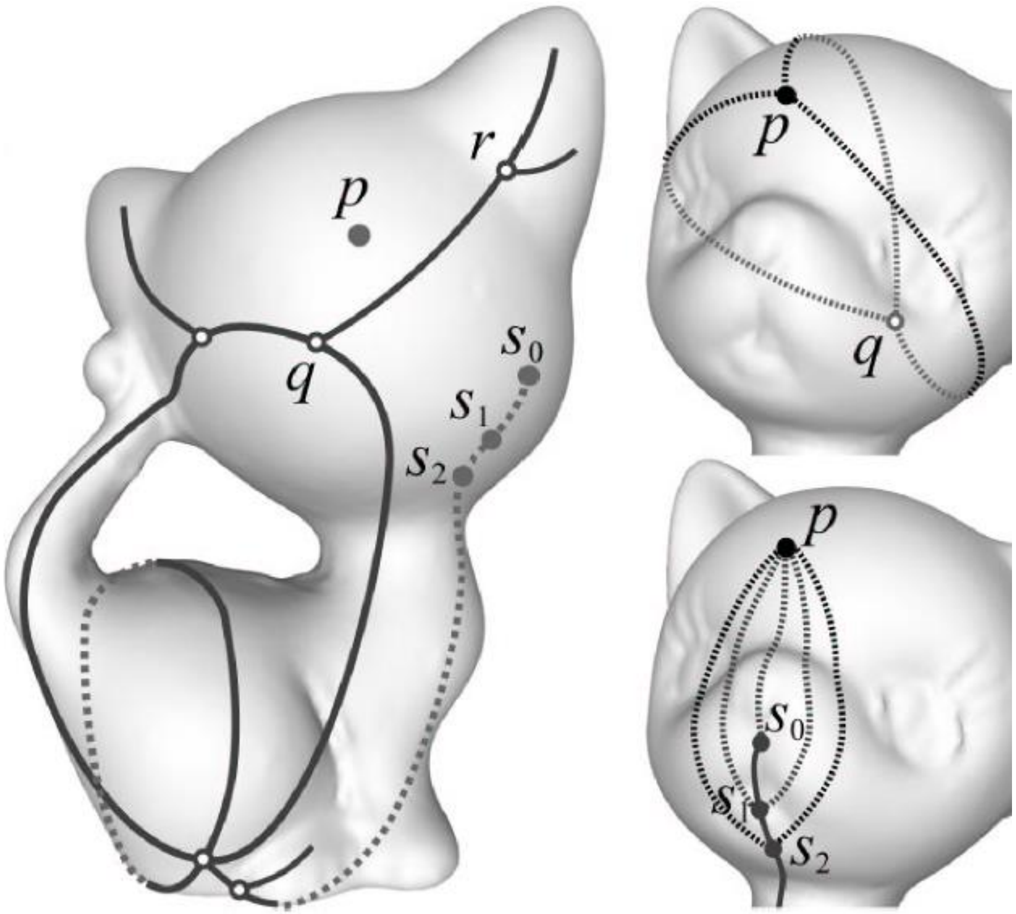




Locally minimizing distance is not enough to be a shortest path!

http://parametricwood2011.files.wordpress.com/2011/01/cone-with-three-geodesics.png

Cut Locus



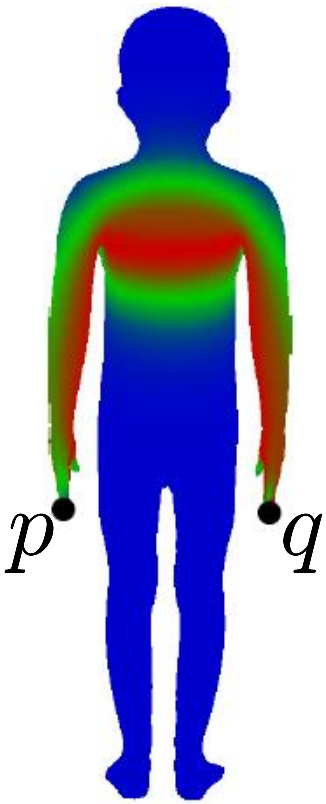
http://www.cse.ohio-state.edu/~tamaldey/paper/geodesic/cutloc.pdf

Cut point:
Point where geodesic ceases to be minimizing

Set of cut points from a source *p*

Fuzzy Geodesics

$$G^{\sigma}_{p,q}(x) \equiv \exp\left(-|d_M(p,x) + d_M(x,q) - d_M(p,q)|/\sigma\right)$$

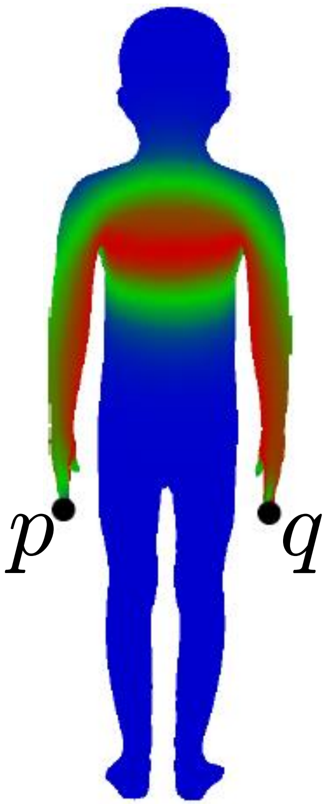


Function on surface expressing difference in triangle inequality

Stable version of geodesic distance

Fuzzy Geodesics

$$G^{\sigma}_{p,q}(x) \equiv \exp\left(-|d_M(p,x) + d_M(x,q) - d_M(p,q)|/\sigma\right)$$

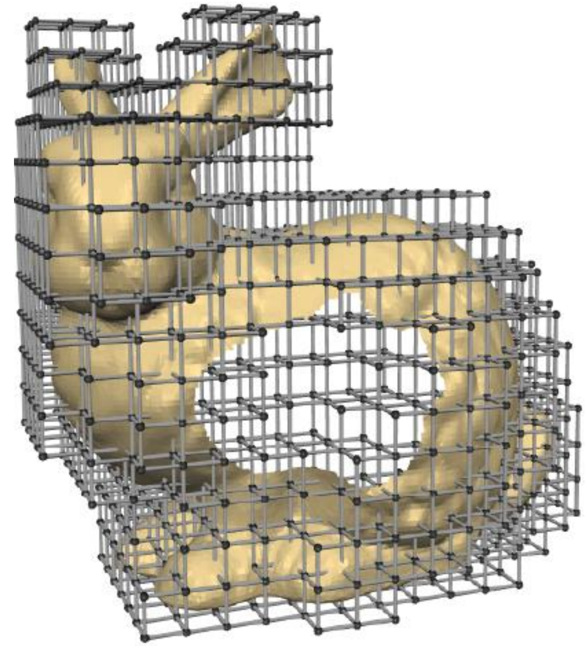


Function on surface expressing difference in triangle inequality

"Intersection" by pointwise multiplication

Stable version of geodesic distance

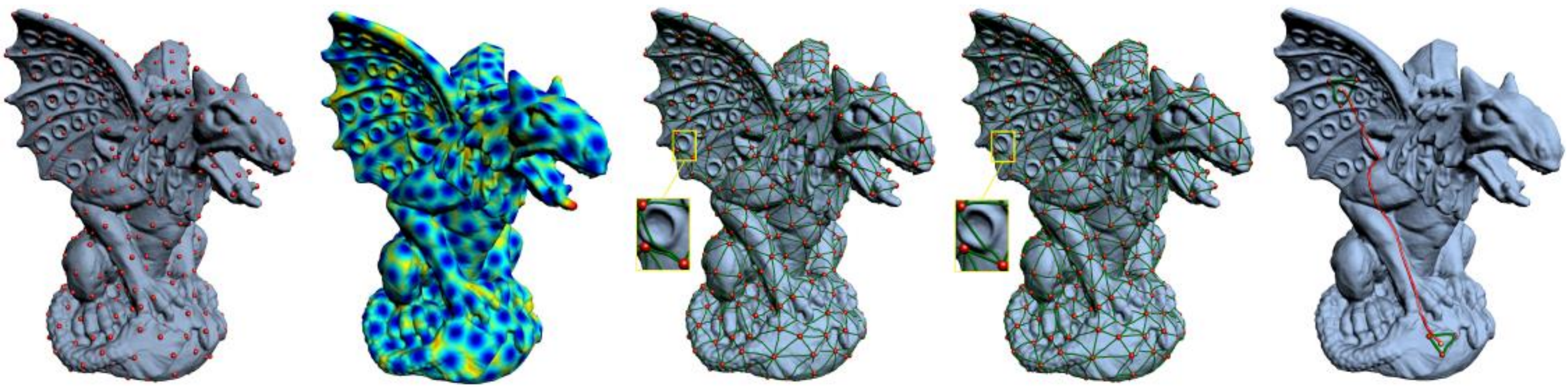
Alternative



Morphological operators to fill holes rather than remeshing

Campen and Kobbelt. "Walking On Broken Mesh: Defect-Tolerant Geodesic Distances and Parameterizations." Eurographics 2011.

All-Pairs Distances



Sample points | Geodesic field | Triangulate (Delaunay) | Fix edges | Query (planar embedding)

Xin, Ying, and He. "Constant-time all-pairs geodesic distance query on triangle meshes." I3D 2012.

Computing Geodesics

CS 468, Spring 2013
Differential Geometry for Computer Science

Justin Solomon and Adrian Butscher