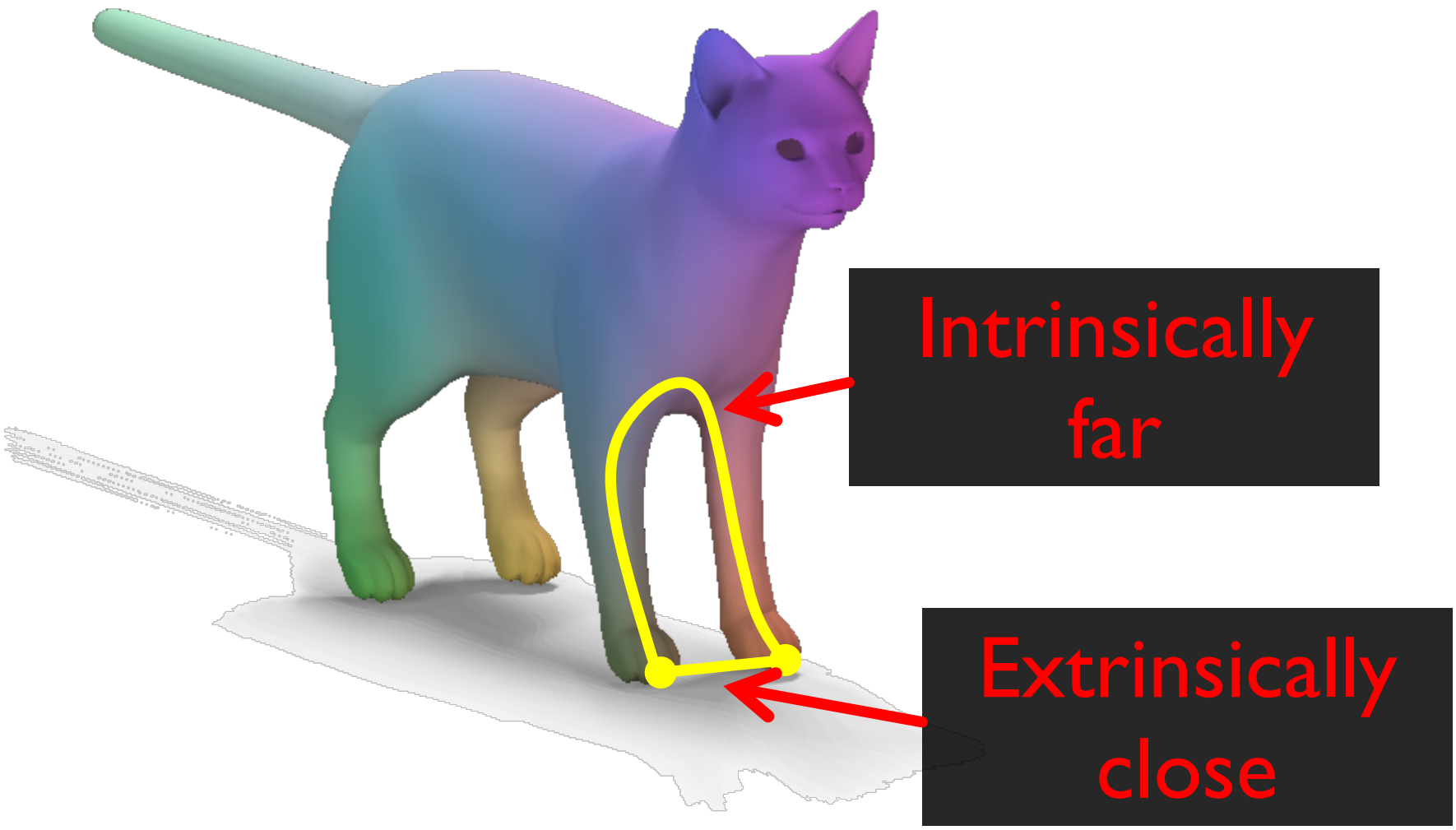
UC San Diego

CSE291-C00

Geodesics

Instructor: Hao Su

Geodesic Distances

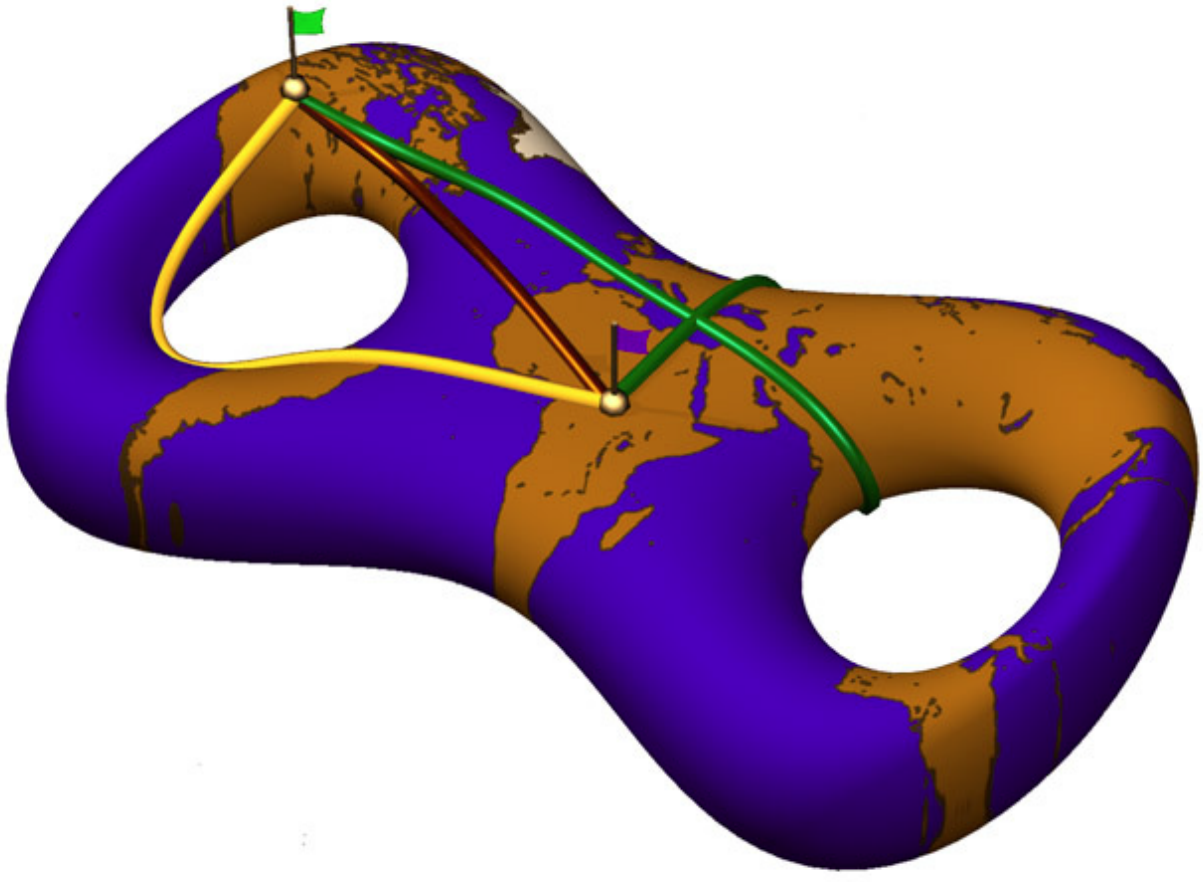


Geodesic distance

[jee-*uh*-**des**-ik **dis**-t*uh*-ns]:

Length of the shortest path, constrained not to leave the manifold.

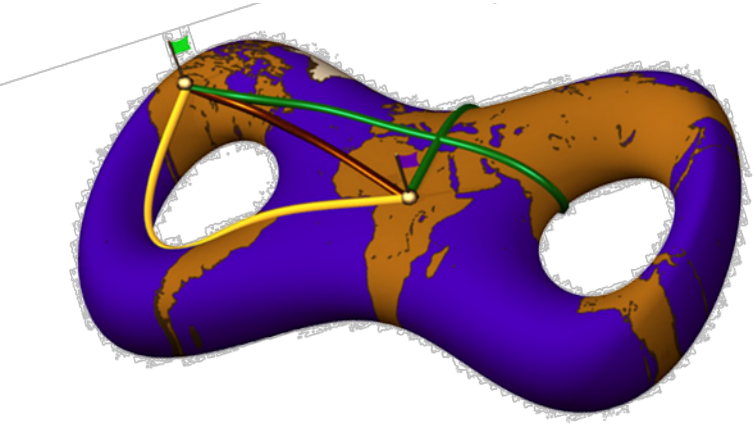
Complicated Problem



Straightest Geodesics on Polyhedral Surfaces (Polthier and Schmies)

Local minima

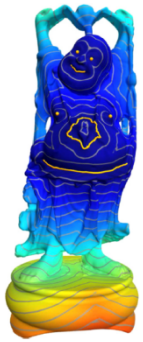
Related Queries



Locally OK

Single source

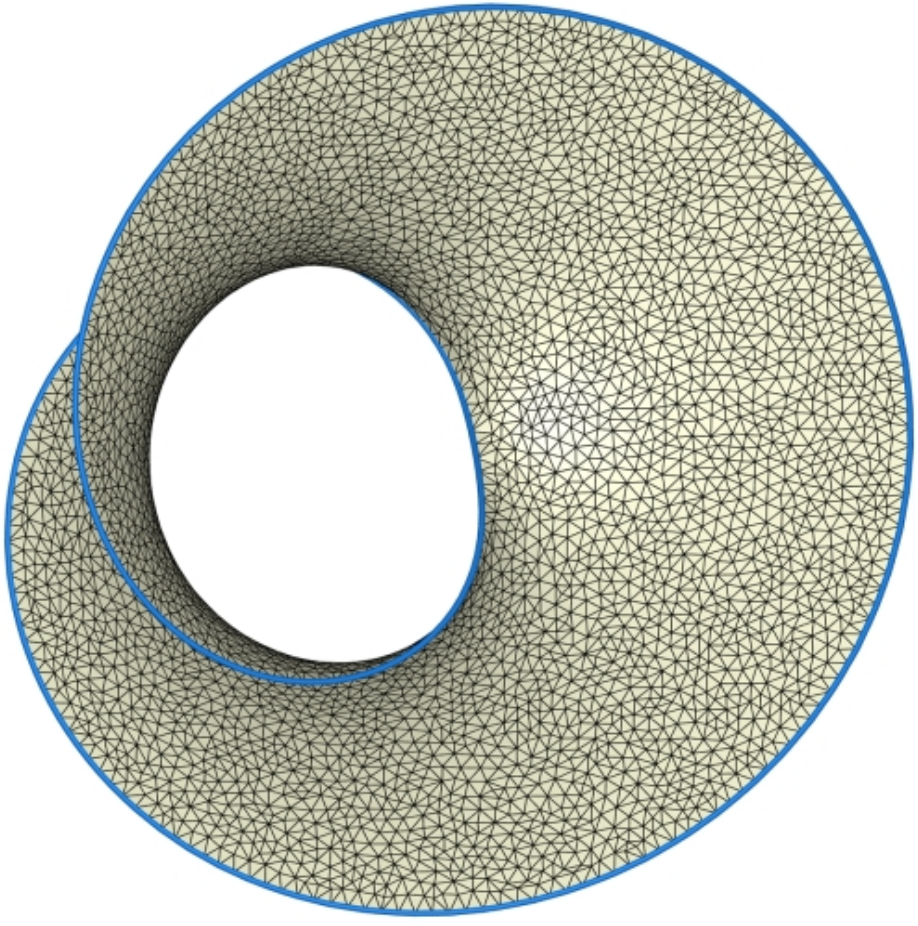
Multi-source



All-pairs

https://www.ceremade.dauphine.fr/~peyre/teaching/manifold/tp3.html http://www.sciencedirect.com/science/article/pii/S001044851100226

Computer Scientists' Approach

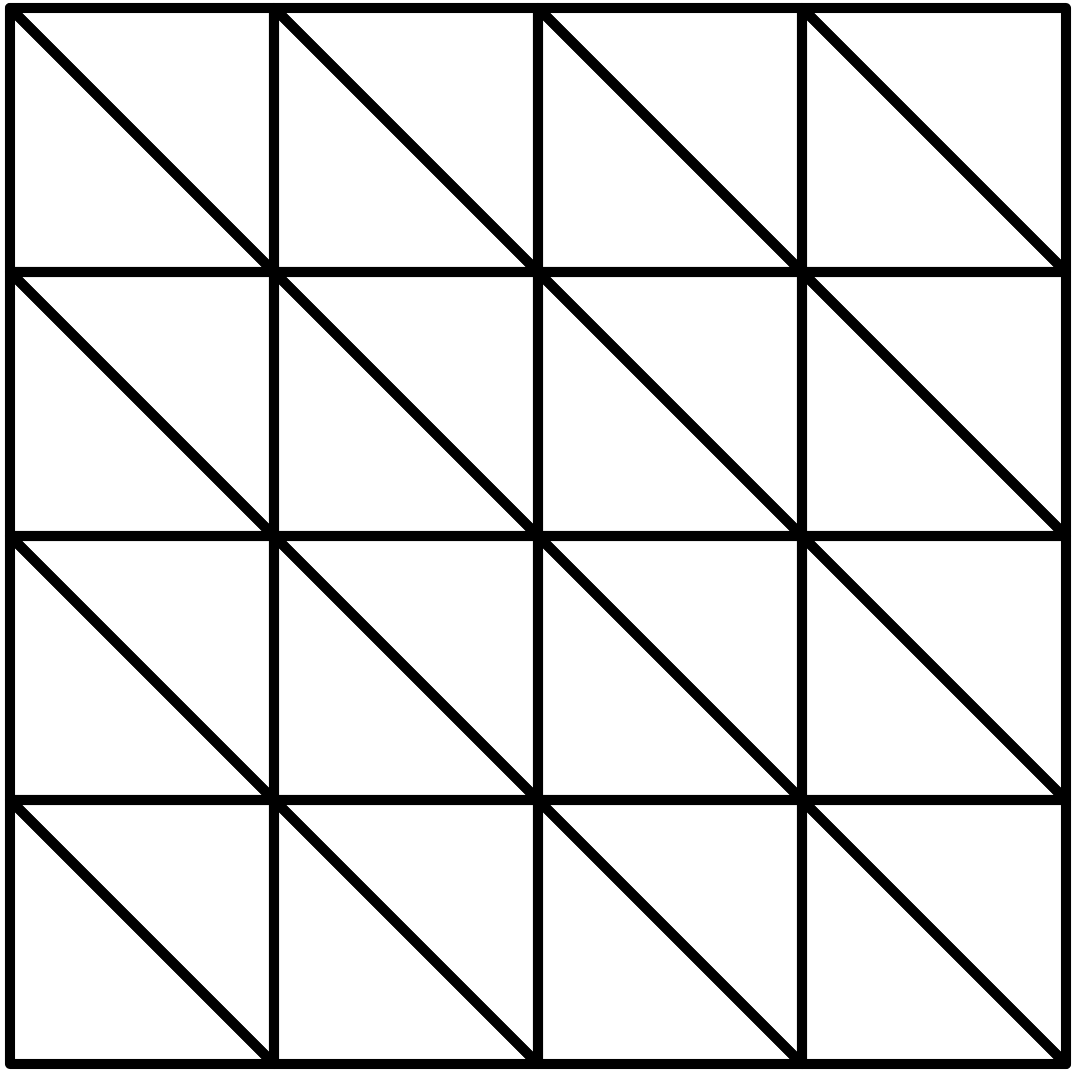


Approximate geodesics as paths along edges

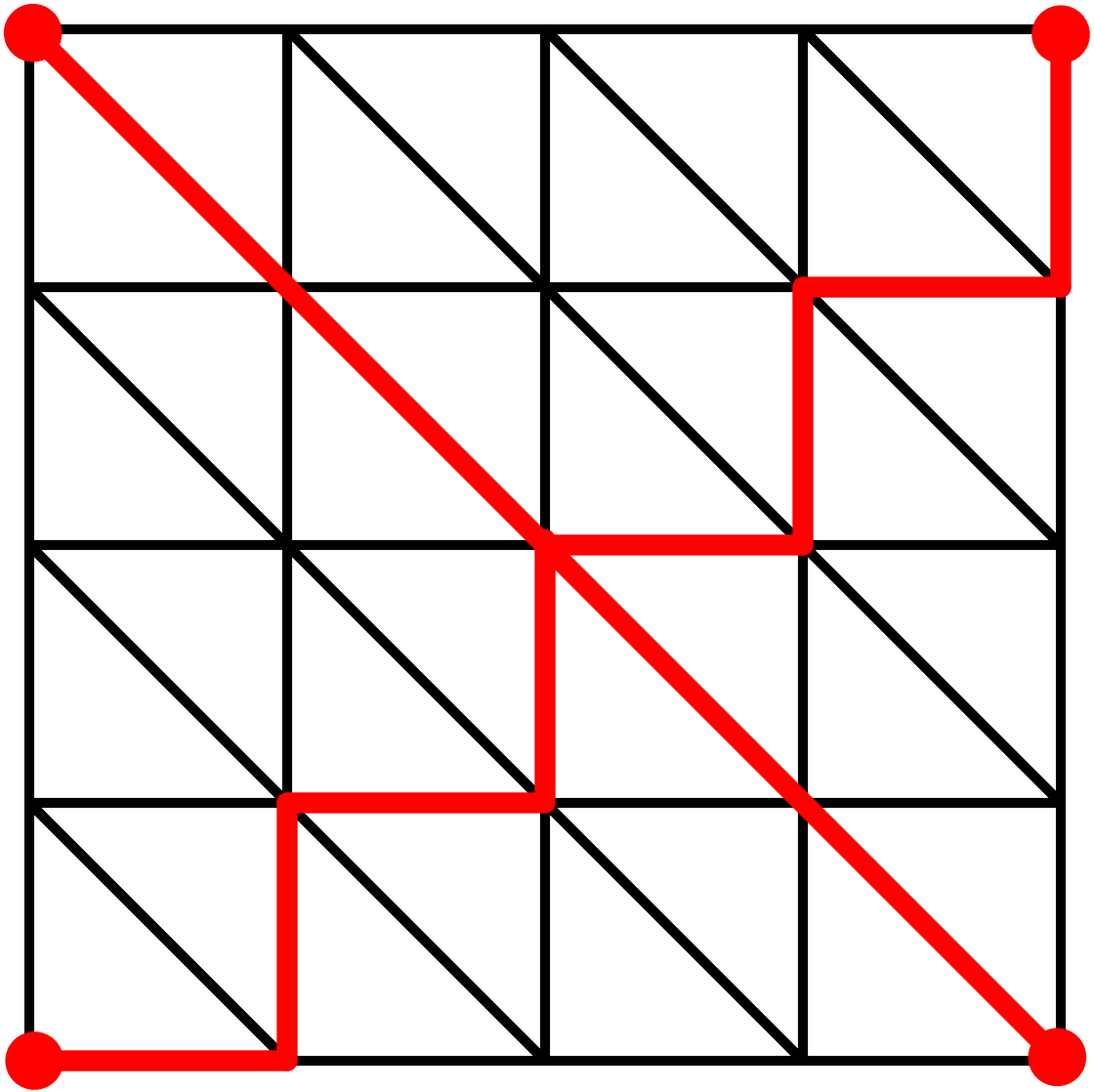
http://www.cse.ohio-state.edu/~tamaldey/isotopic.html

Meshes are graphs

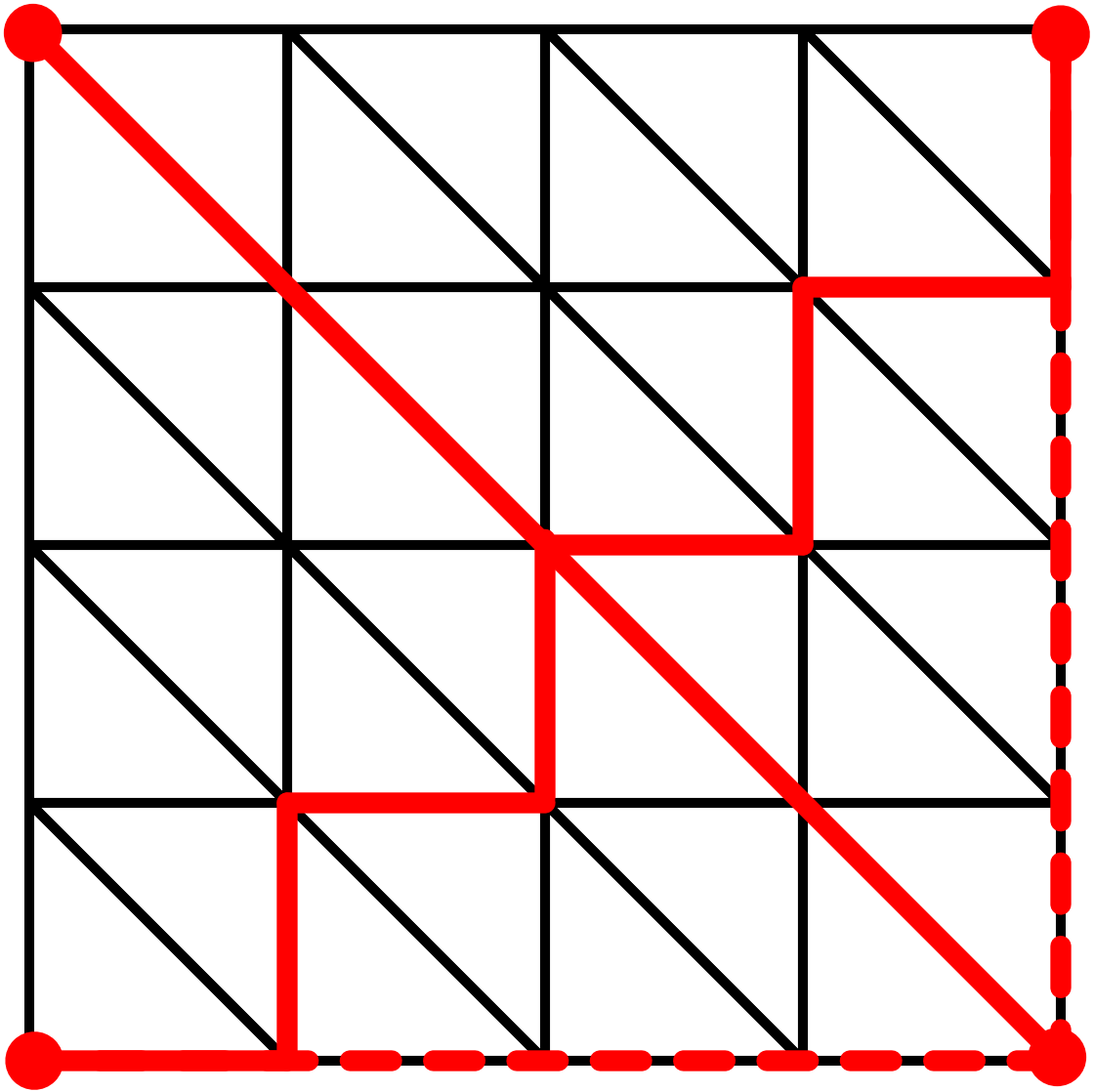
Pernicious Test Case



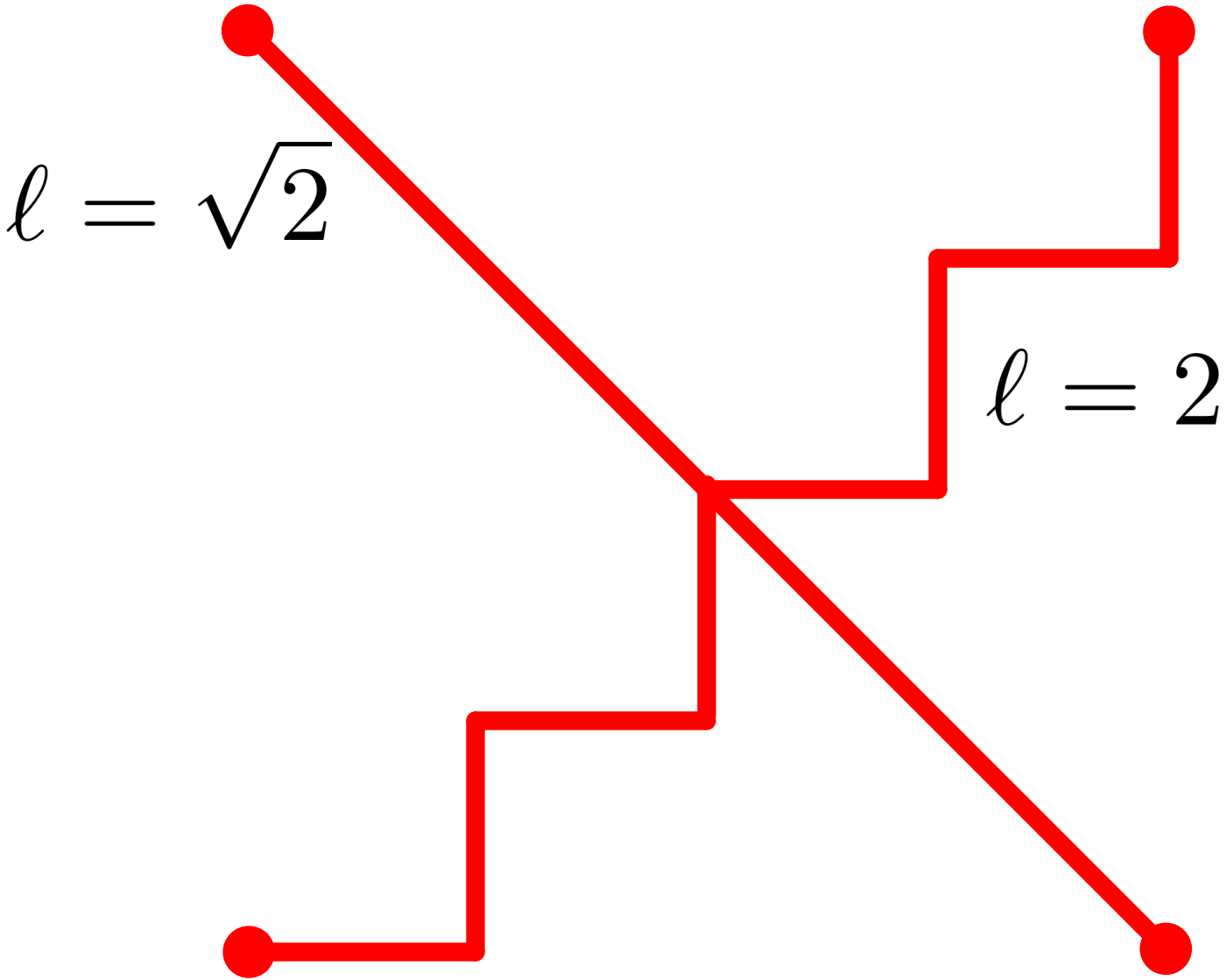
Pernicious Test Case



Pernicious Test Case



Distances



Conclusion 1

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

Often an acceptable approximation.

Conclusion 2

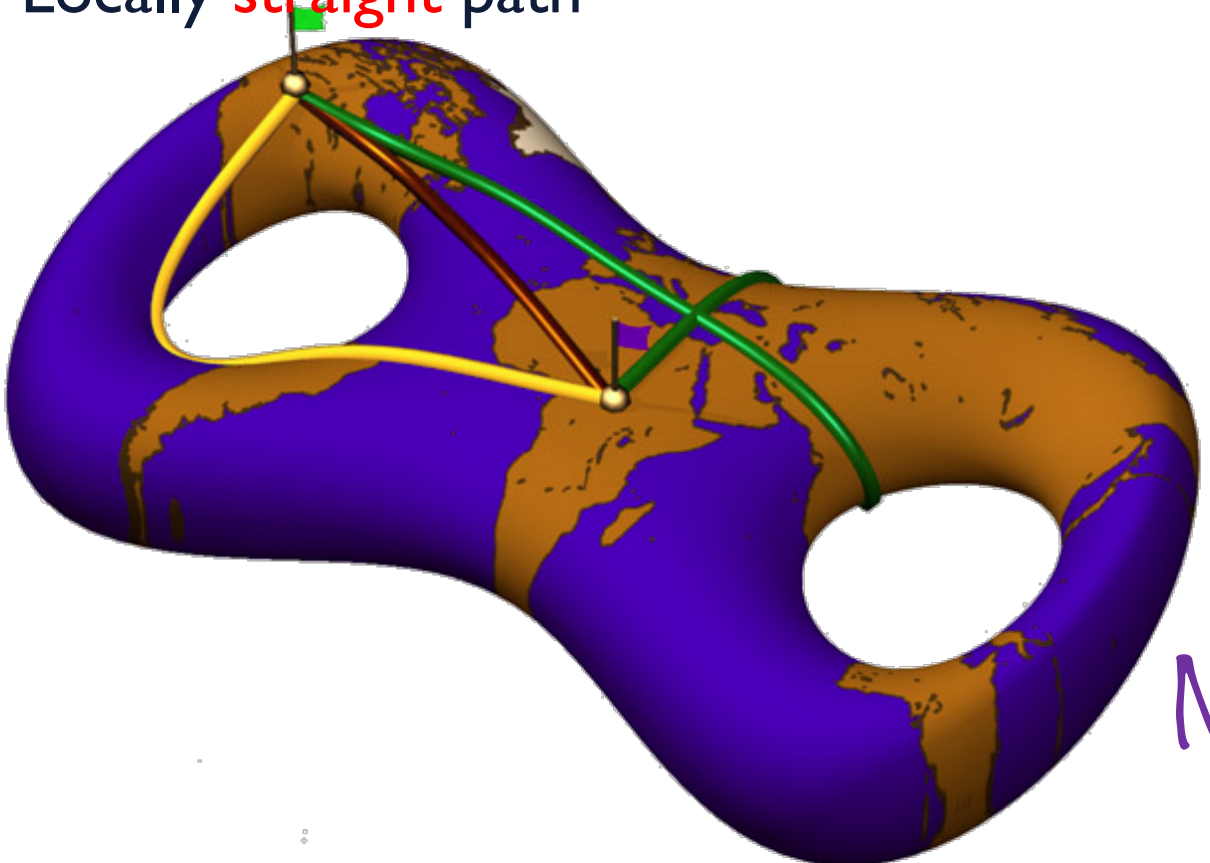
Geodesic distances need special discretization.

So, we need to understand the theory!

```
\begin{math}
```

Three Possible Definitions

- Globally shortest path
- Local minimizer of length
- Locally straight path



Not the same!

Recall: Arc Length

$$\int_a^b \|\gamma'(t)\|\, dt$$

Energy of a Curve

$$L[\gamma] := \int_a^b \|\gamma'(t)\|\, dt$$

Easier to work with:

$$E[\gamma] := \frac{1}{2}\int_a^b \|\gamma'(t)\|^2\, dt$$

Lemma: $L^2 \leq 2(b-a)E$

Equality exactly when parameterized by arc length. Proof on board.

First Variation of Arc Length

Lemma. Let γ be a family of curves with fixed endpoints in surface S; assume γ is parameterized by arc length at t=0. Then,

$$\left.\frac{d}{dt}E[\gamma_t]\right|_{t=0} = -\int_a^b \left(\frac{d\gamma_t(s)}{dt} \cdot \mathrm{proj}_{T_{\gamma_t(s)}S}\left[\gamma_t''(s)\right]\right) ds$$

Corollary. γ is a geodesic iff

$$\mathrm{proj}_{T_{\gamma(s)}S}\left[\gamma''(s)\right] = 0$$

Intuition

- The only acceleration is out of the surface
- No steering wheel!

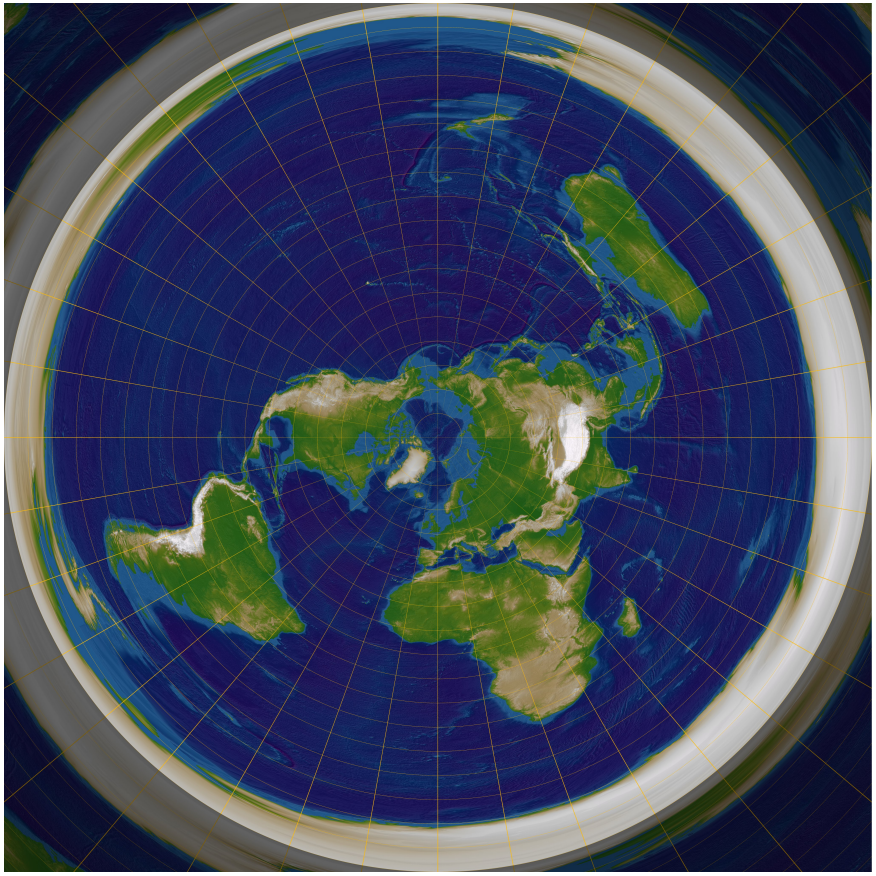
$$\mathrm{proj}_{T_{\gamma(s)}S}\left[\gamma''(s)\right] = 0$$

Two Local Perspectives

$$\mathrm{proj}_{T_{\gamma(s)}S}\left[\gamma''(s)\right] = 0$$

- **Boundary value problem**
  - **Given:** $\boldsymbol{\gamma(0), \gamma(1)}$
- **Initial value problem (ODE)**
  - **Given:** $\boldsymbol{\gamma(0), \gamma'(0)}$

Exponential Map

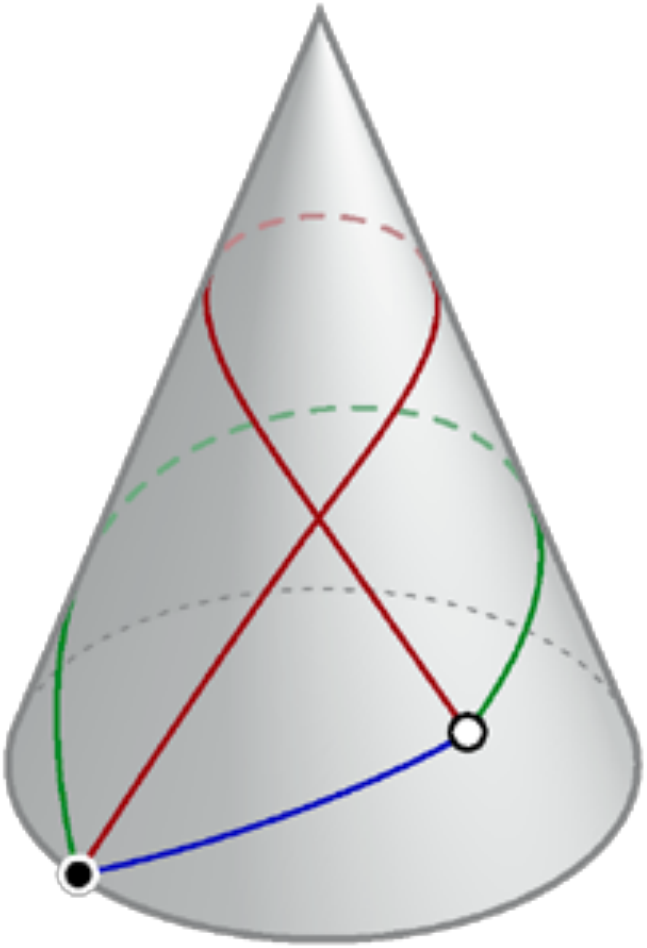


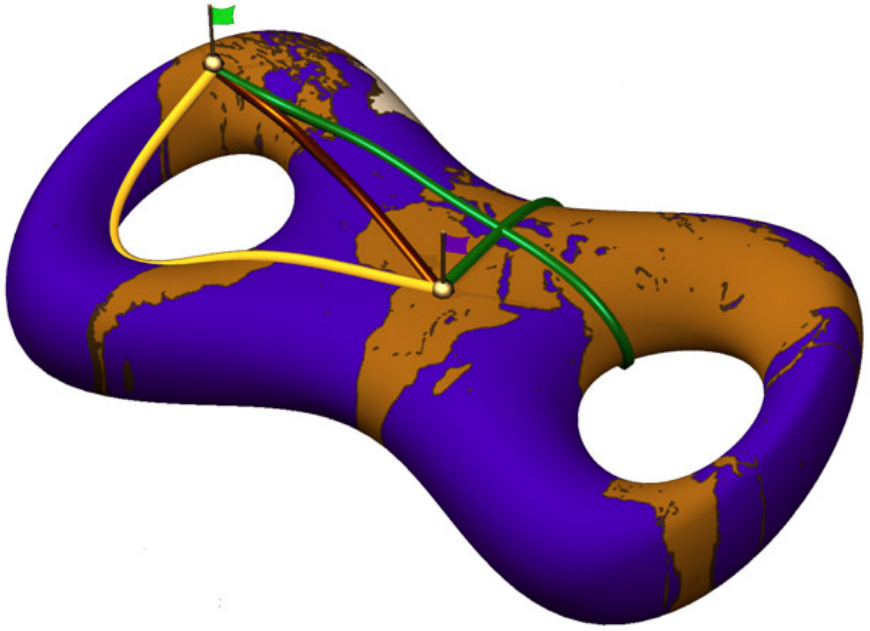
$$\exp_p(v) := \gamma_v(1)$$

where γ_v is (unique) geodesic from p with velocity v.

https://en.wikipedia.org/wiki/Exponential_map_(Riemannian_geometry)

Instability of Geodesics

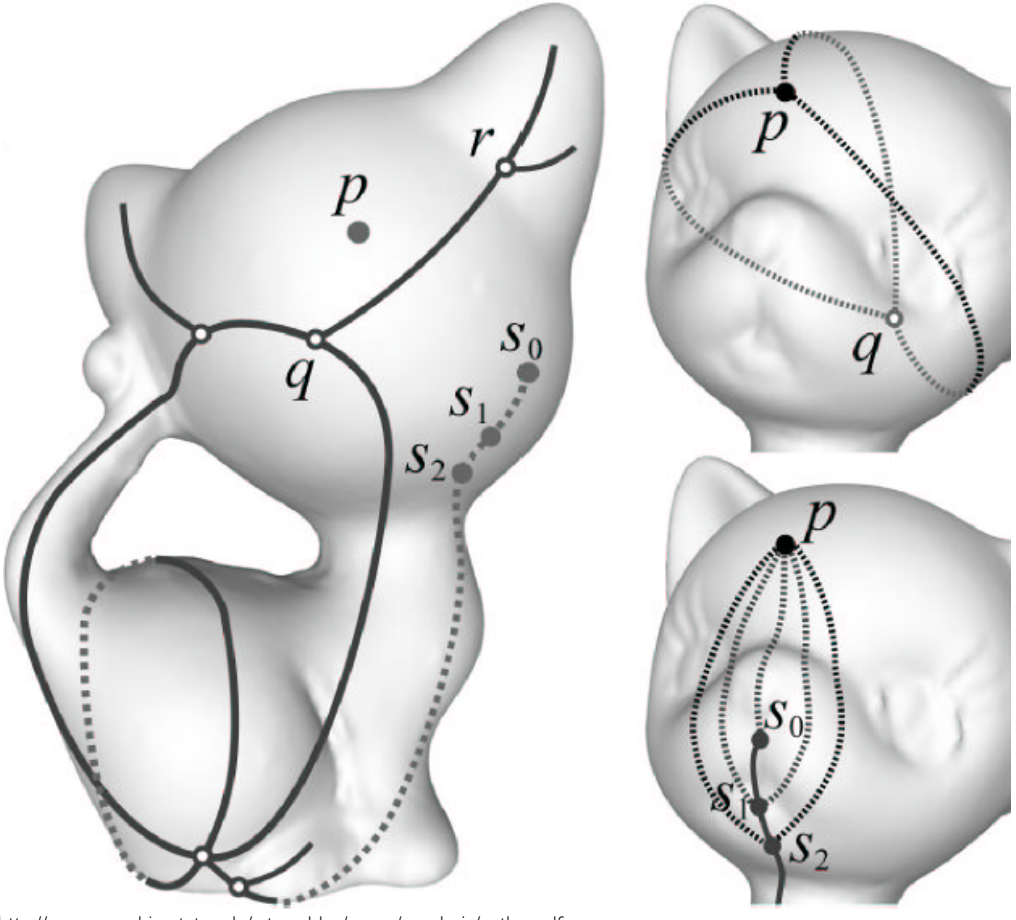




Locally minimizing distance is not enough to be a shortest path!

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

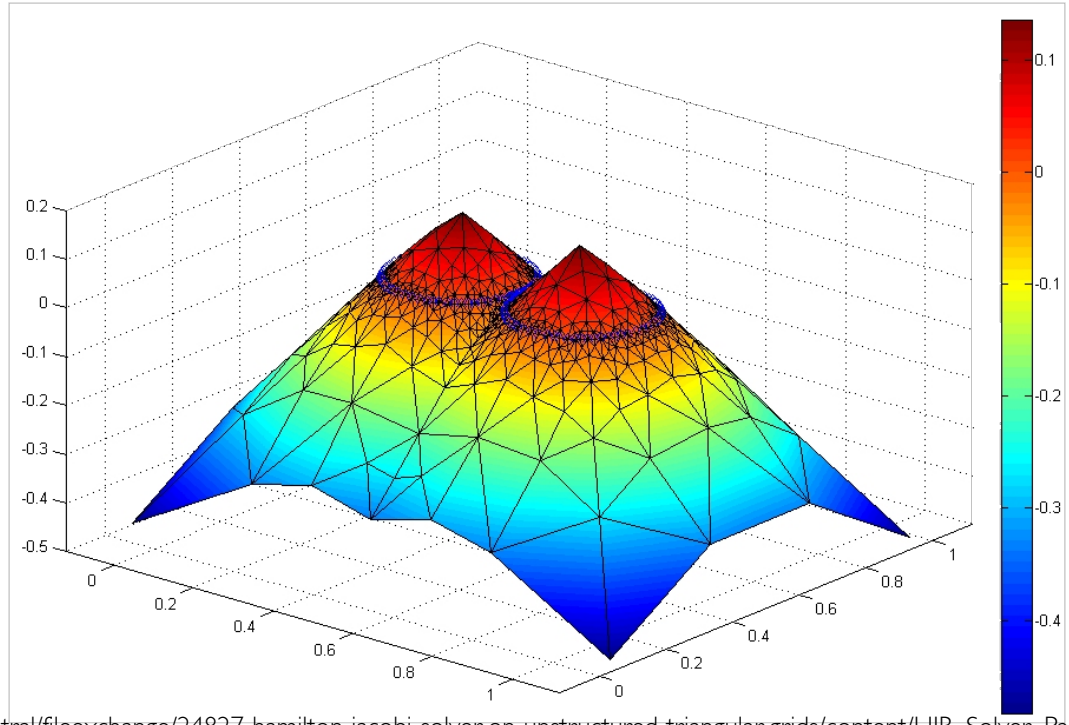
Cut point:

Point where geodesic ceases to be minimizing

Set of cut points from a source p

Eikonal Equation

$$\|\nabla u\|_2 = 1$$ (defer)



https://www.mathworks.com/matlabcentral/fileexchange/24827-hamilton-jacobi-solver-on-unstructured-triangular-grids/content/HJB_Solver_Package/@SolveEikonal/SolveEikonal.m

```
\end{math}
```

Starting Point for Algorithms

Graph shortest path algorithms are well-understood.

Can we use them (carefully) to compute geodesics?

Useful Principles

"Shortest path had to come from somewhere."

"All pieces 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*:

$$k = \arg \min_{v_k \in V \setminus S} d_k$$

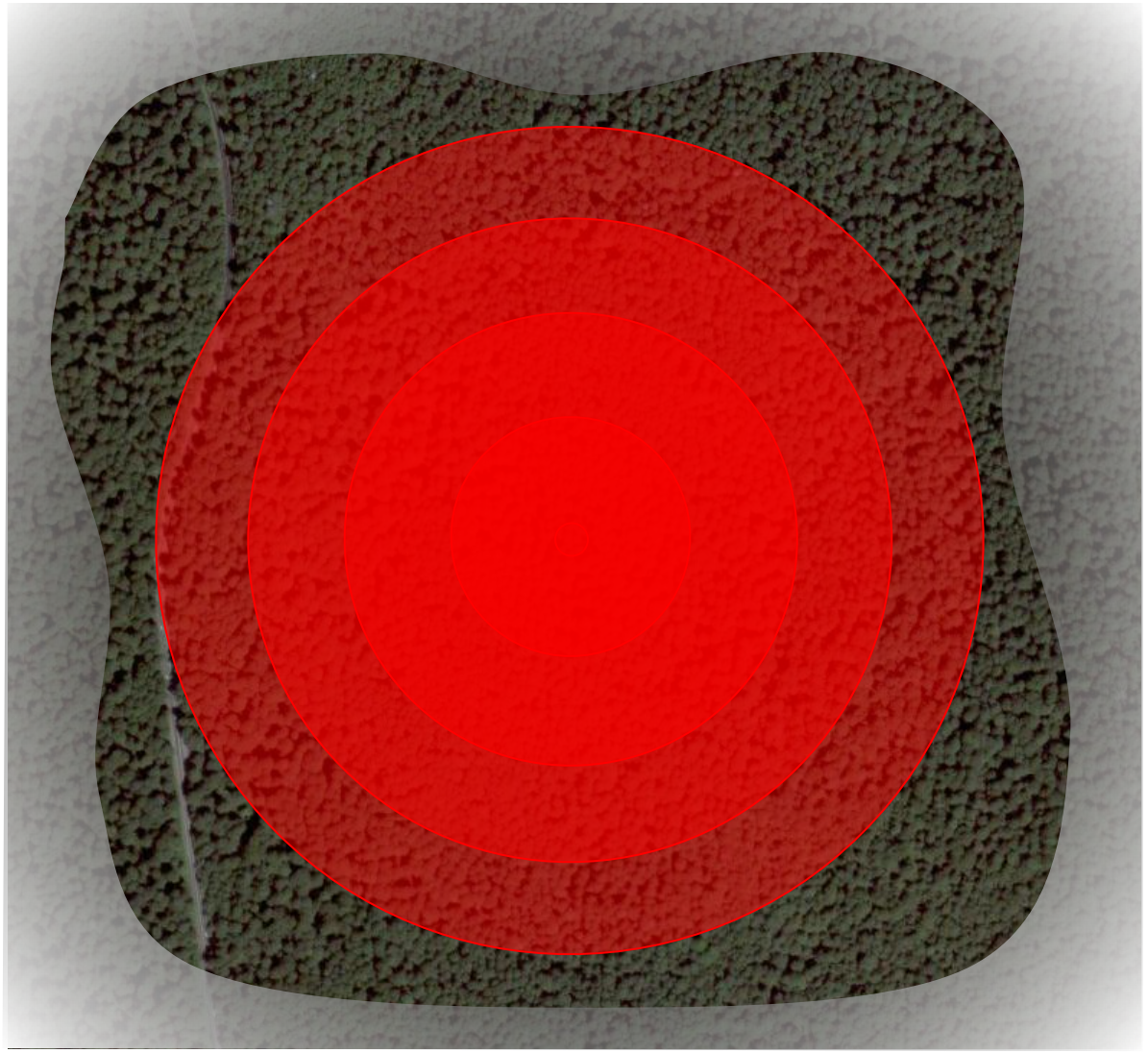
$$S \leftarrow v_k$$

$$d_\ell \leftarrow \min\{d_\ell, d_k + d_{k\ell}\} \; \forall \text{ neighbors } v_\ell \text{ of } v_k$$

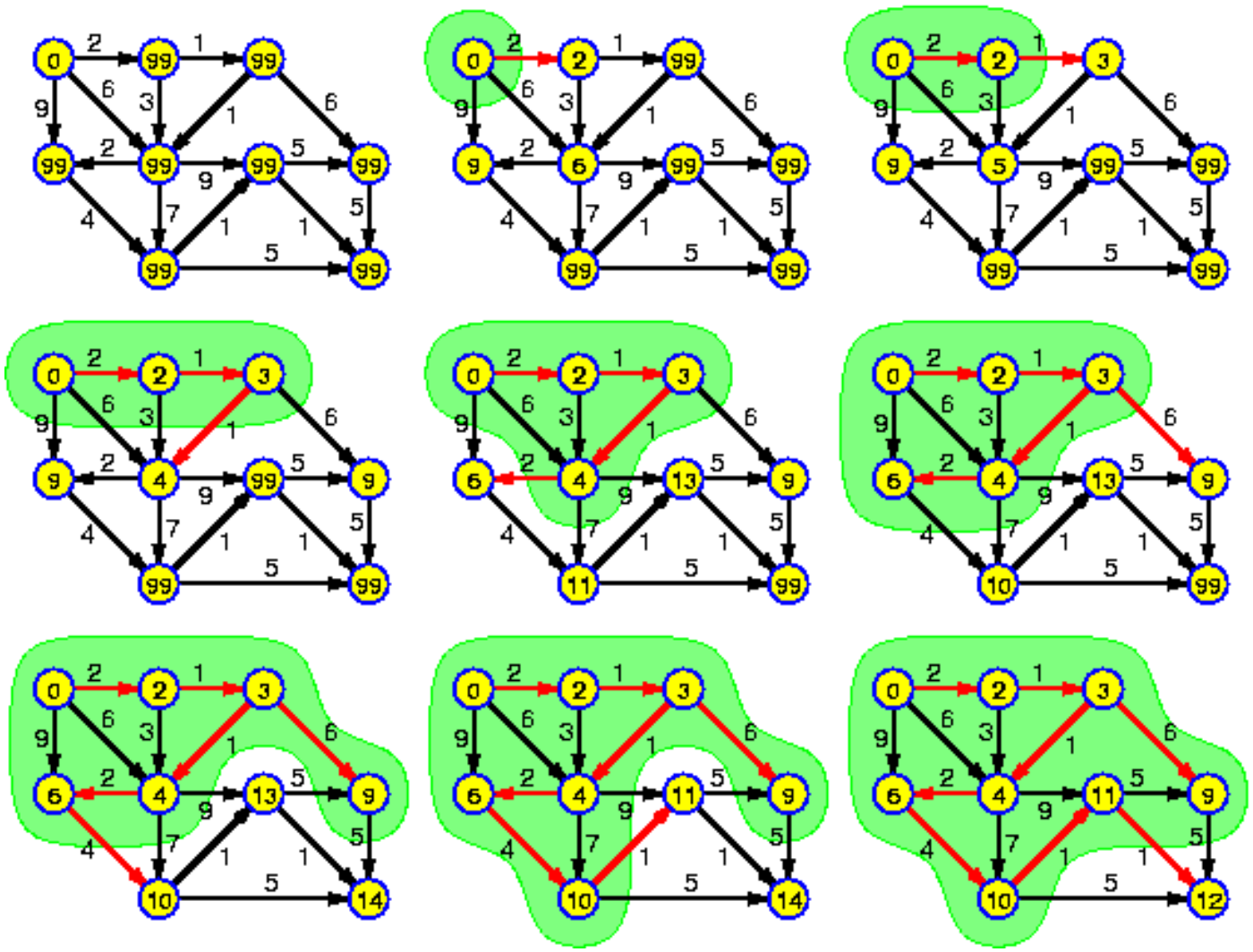
Inductive proof:

During each iteration, *S* remains optimal.

Advancing Fronts



Example

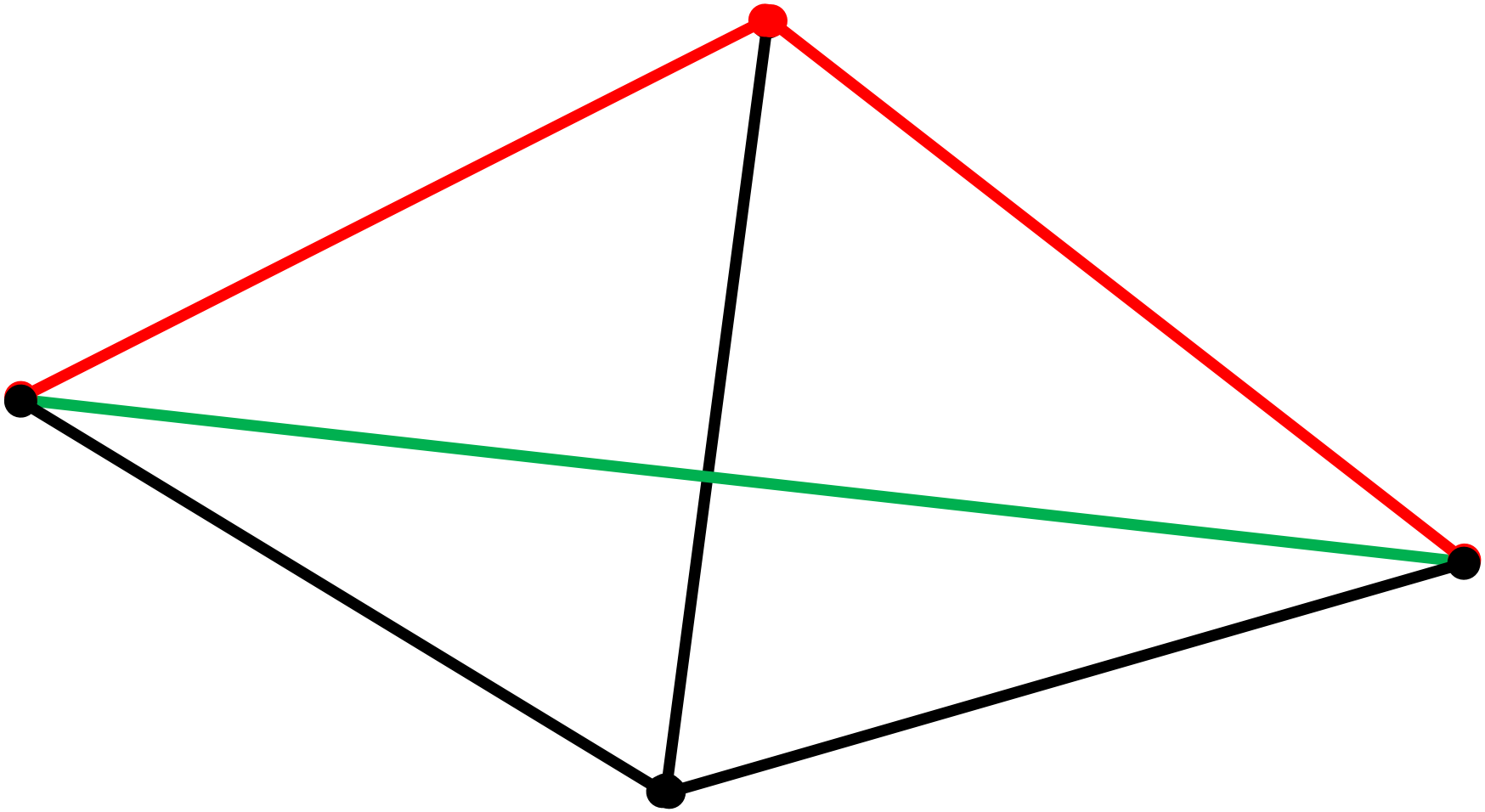


http://www.iekucukcay.com/wp-content/uploads/2011/09/dijkstra.gif

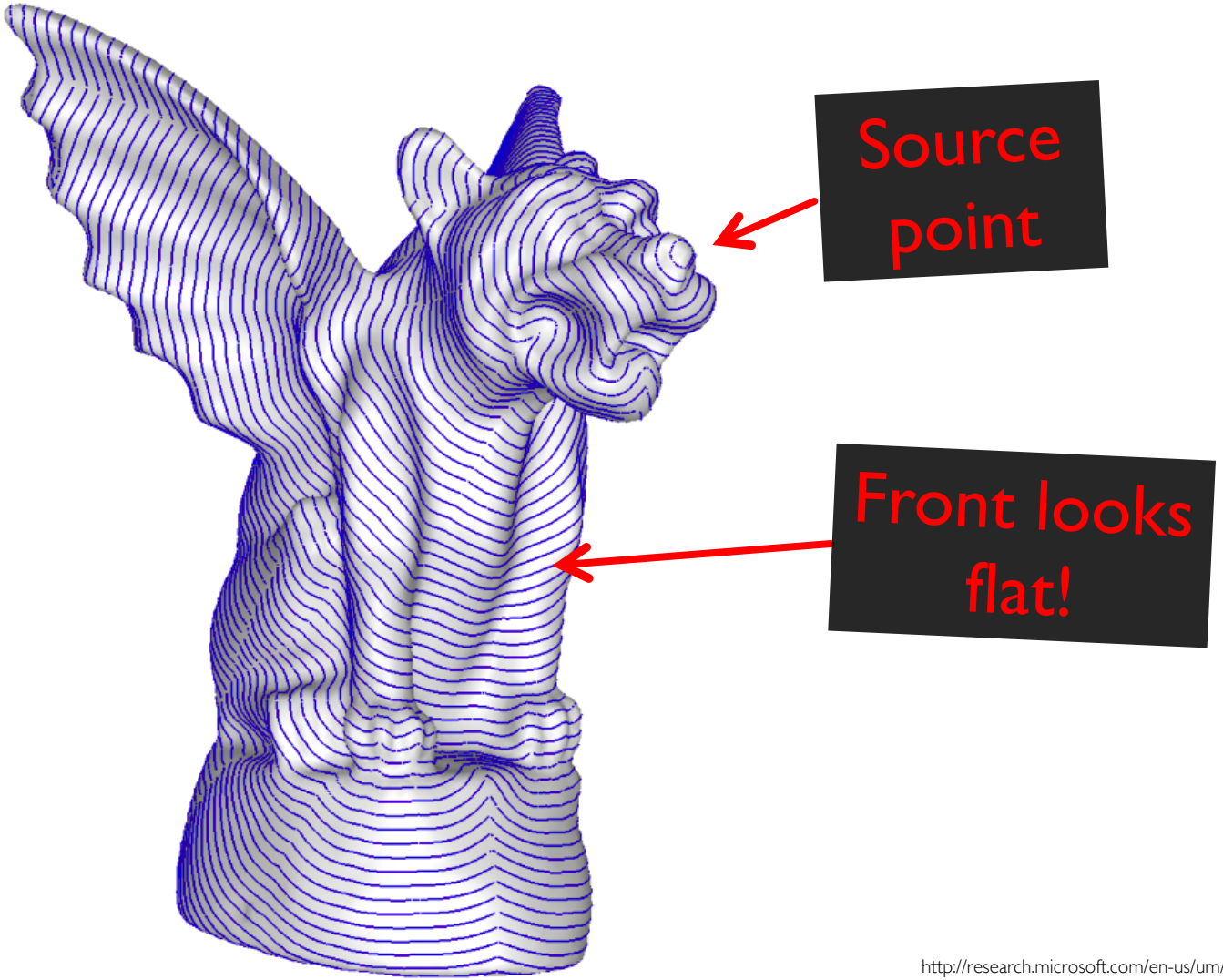
Fast Marching

Dijkstra's algorithm, modified to approximate geodesic distances.

Problem

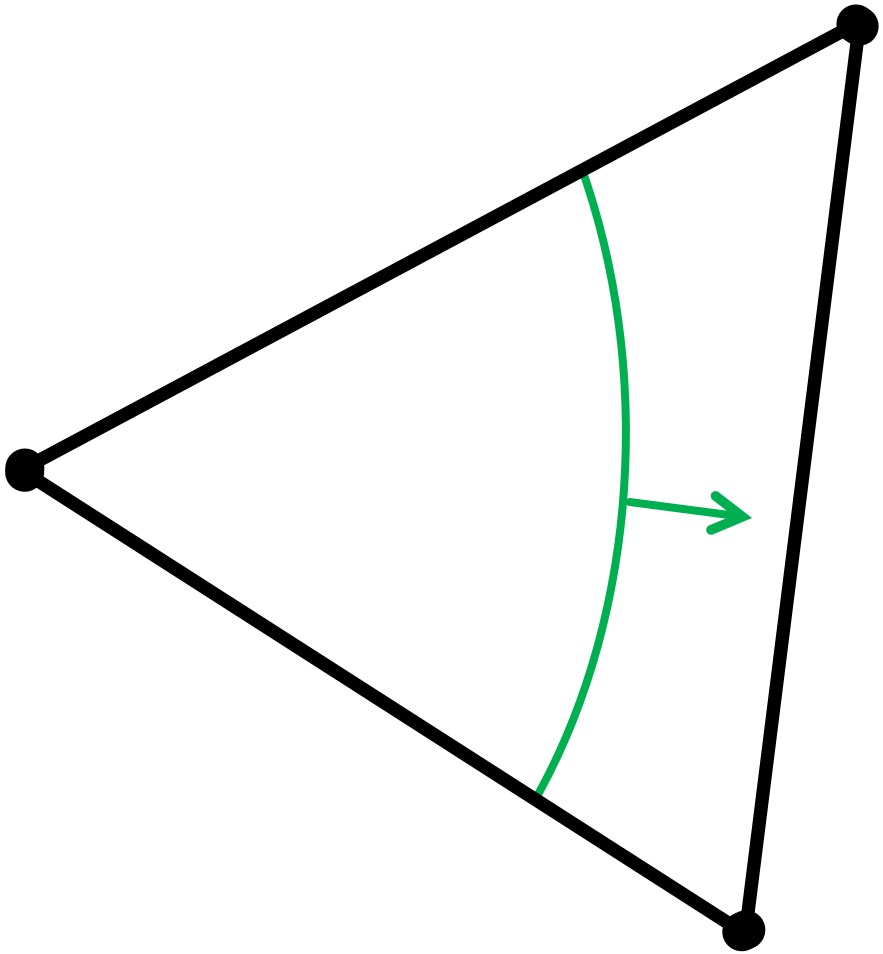


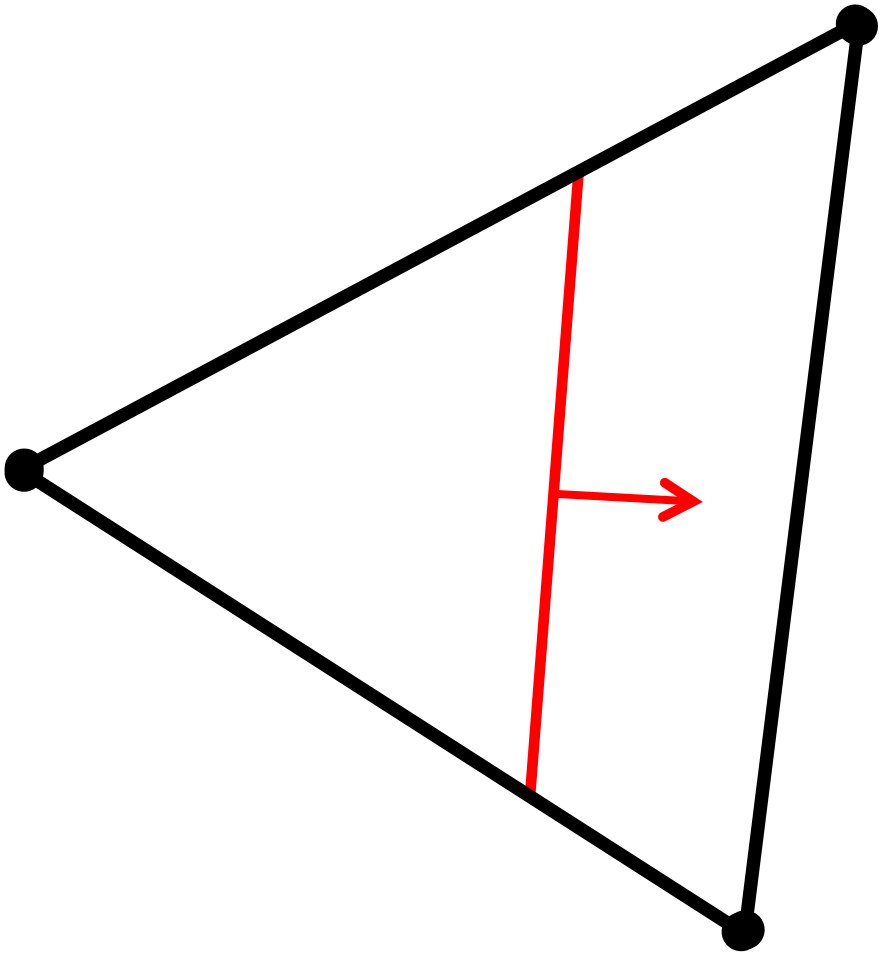
Planar Front Approximation



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

At Local Scale





Fast Marching vs. Dijkstra

- Modified update step

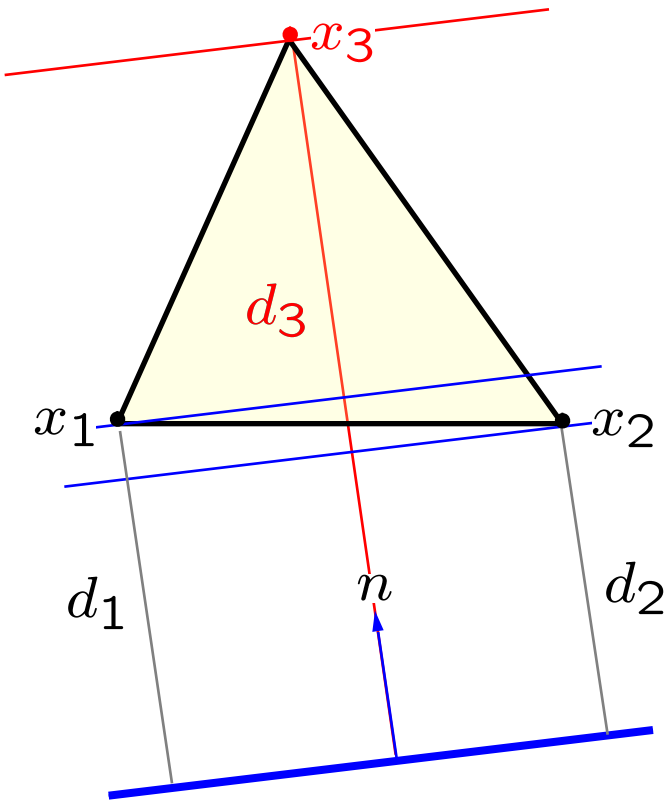
- Update all triangles adjacent to a given vertex

Fast marching update step

- Update x_3 from triangle (x_1, x_2, x_3)
- Compute $d(x_3)$ from $d_1 = d(x_1)$ and $d_2 = d(x_2)$
- Model **wave front** propagating from **planar source**

$$\langle x, n \rangle + p = 0$$

  - n unit **propagation direction**
  - p source **offset**
- Front hits x_1 at time d_1
- Hits x_2 at time d_2
- **When does the front arrive to x_3 ?**



Planar source

$$\langle x, n \rangle + p = 0$$

Fast marching update step

- Assume w.l.o.g. $x_1, x_2, x_3 \in \mathbb{R}^2$ and $x_3 = 0$.
- d_3 is given by the **point-to-plane distance**

$$d_3 \;=\; \langle x_3, n \rangle + p \;=\; p$$

- Solve for parameters n and p using the **point-to-plane distance**

$$\begin{aligned} \langle x_1, n \rangle + p &= d_1 \\ \langle x_2, n \rangle + p &= d_2 \end{aligned}$$

- In **vector notation** $V^\top n + p \cdot 1 \;=\; d$

where $V = (x_1, x_2)$, $d = (d_1, d_2)^\top$, and $1 = (1, 1)^\top$.

- In a **non-degenerate** triangle matrix V is **full-rank**

$$n \;=\; (V^\top)^{-1}(d - p \cdot 1) \;=\; V^{-\top}(d - p \cdot 1)$$

Fast marching update step

$$n \;=\; V^{-\mathsf{T}}(d - p \cdot 1)$$

- Apparently, we have **two equations** with **three variables**.
- However, n is a **unit vector**, hence $\|n\| = 1$.

$$\begin{aligned} 1 \;&=\; n^{\mathsf{T}} n \;=\; (d - p \cdot 1)^{\mathsf{T}} V^{-1} V^{-\mathsf{T}} (d - p \cdot 1) \\ &=\; (d - p \cdot 1)^{\mathsf{T}} (V^{\mathsf{T}} V)^{-1} (d - p \cdot 1) \\ &=\; p^2 \cdot 1^{\mathsf{T}} Q 1 - 2p \cdot 1^{\mathsf{T}} Q d + d^{\mathsf{T}} Q d \end{aligned}$$

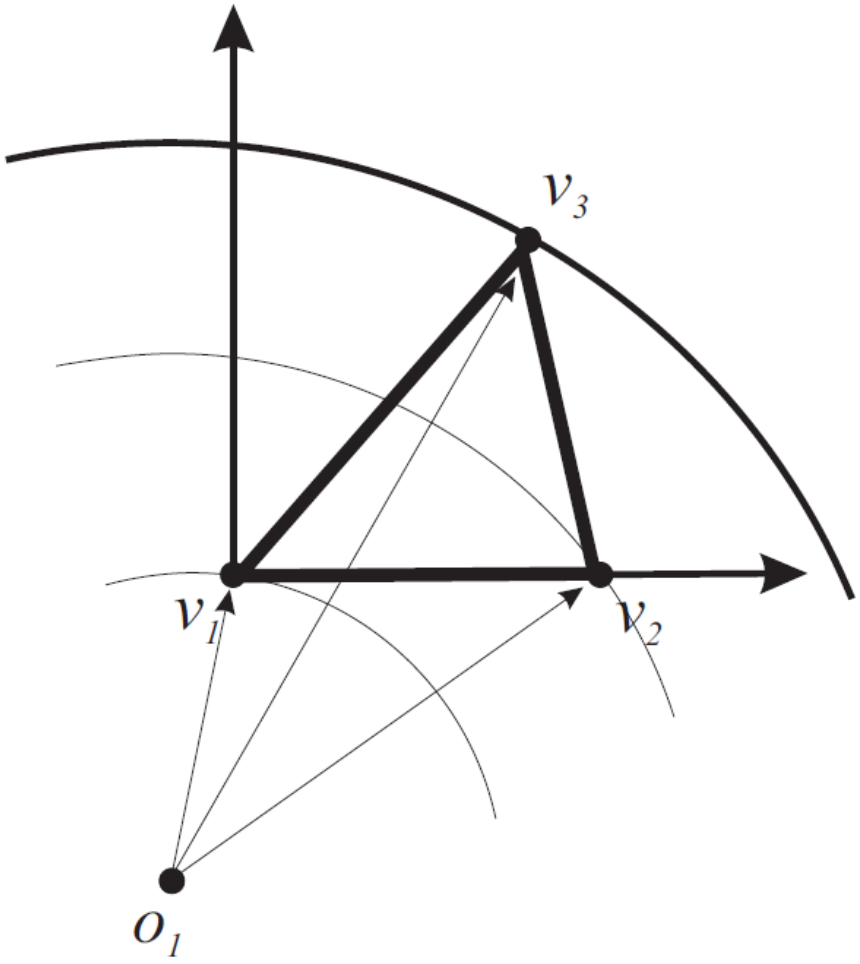
where $Q = (V^{\mathsf{T}} V)^{-1}$.

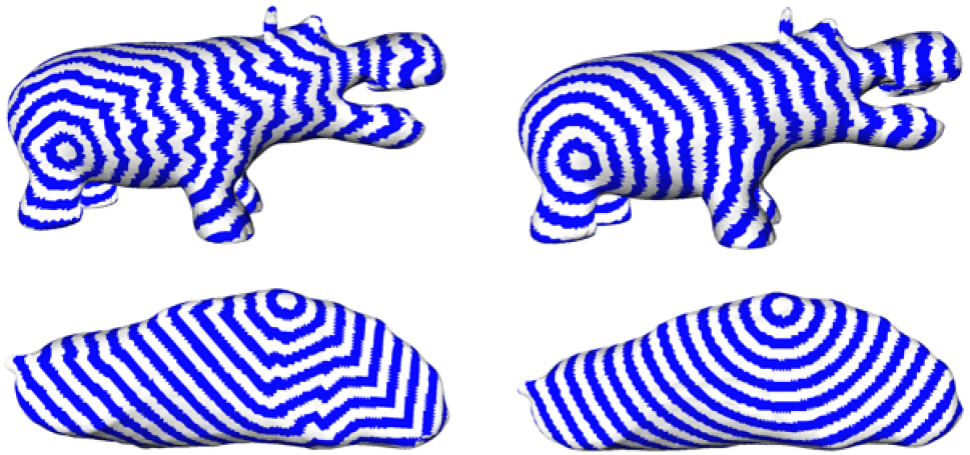
- Substitute $d_3 \;=\; p$ and obtain a **quadratic equation**

$$d_3^2 \cdot 1^{\mathsf{T}} Q 1 - 2 d_3 \cdot 1^{\mathsf{T}} Q d + d^{\mathsf{T}} Q d - 1 \;=\; 0$$

A much better one!

Modifying Fast Marching

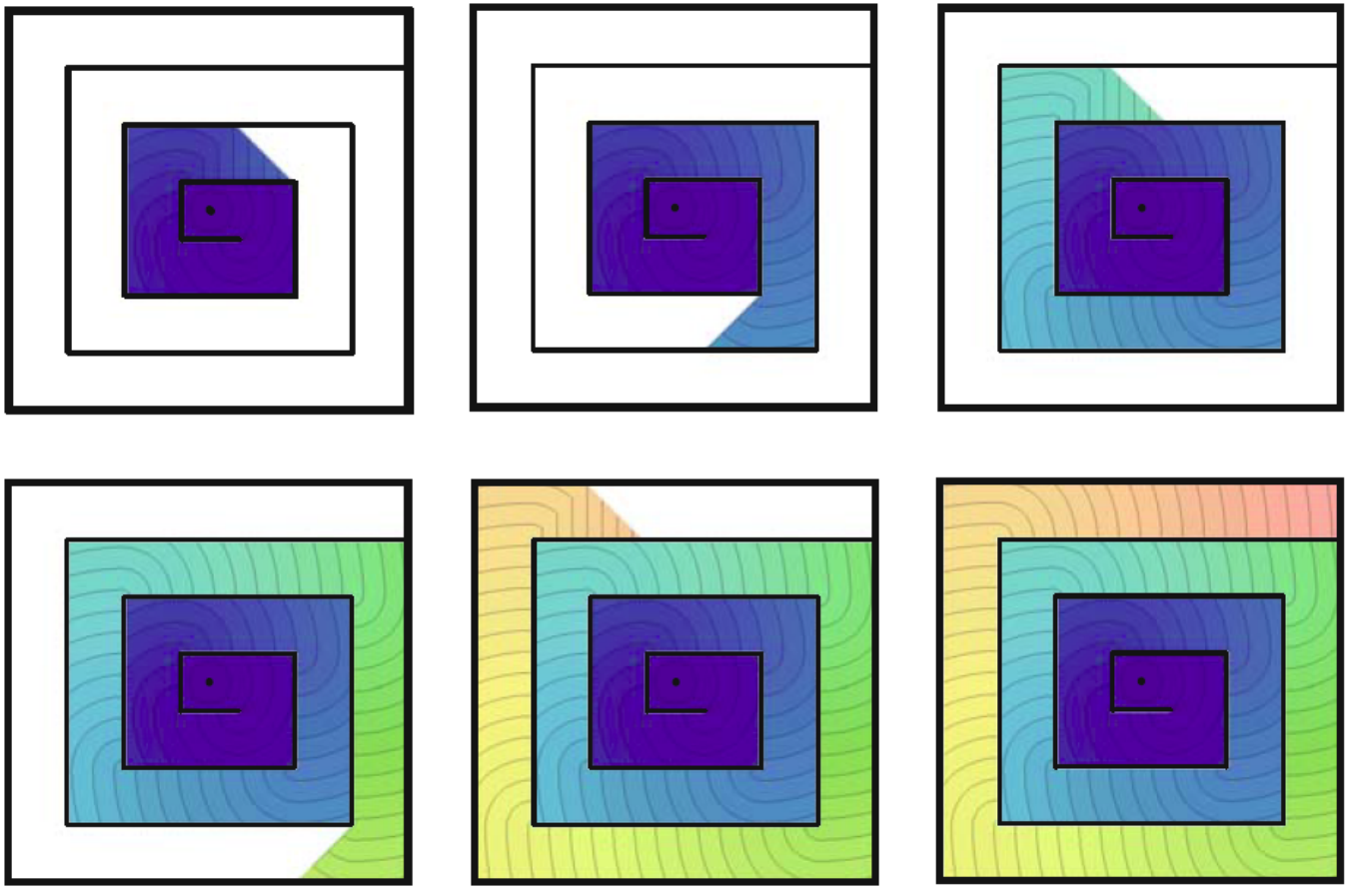




[Novotni and Klein 2002]:

Circular wavefront

Modifying Fast Marching



Raster scan and/or parallelize

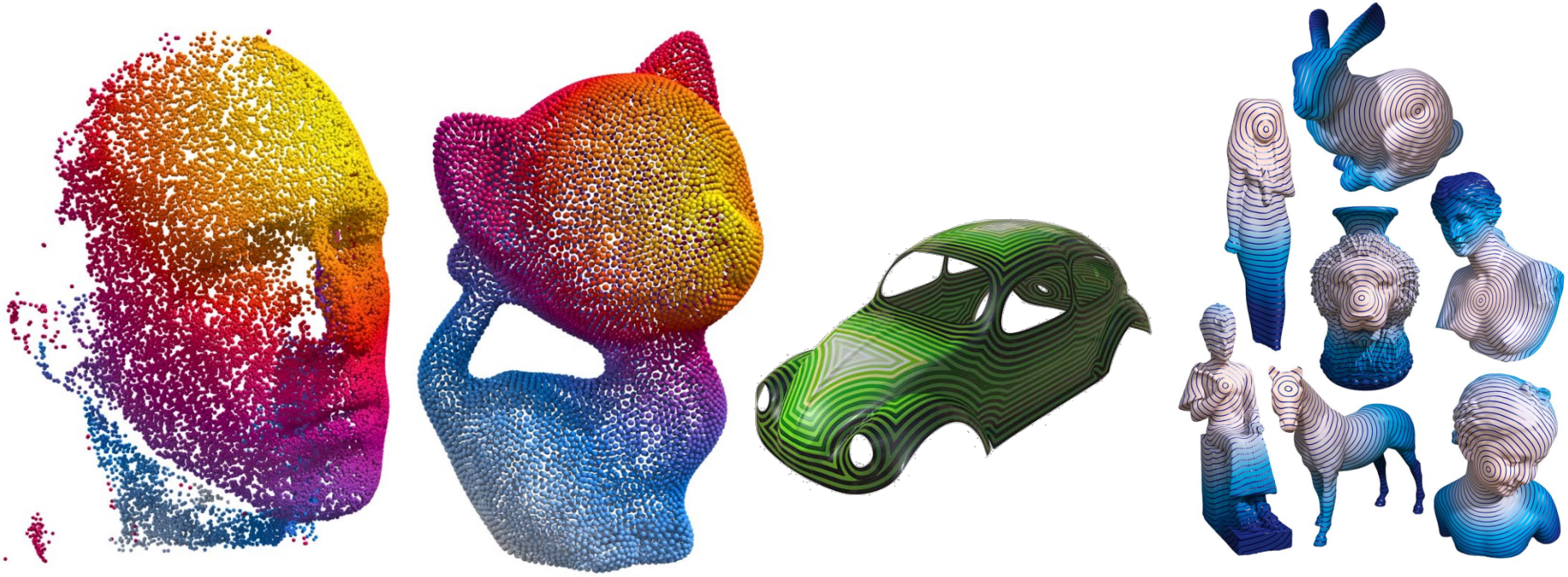
Bronstein, *Numerical Geometry of Nonrigid Shapes*

Grids and parameterized surfaces

Alternative to Eikonal Equation

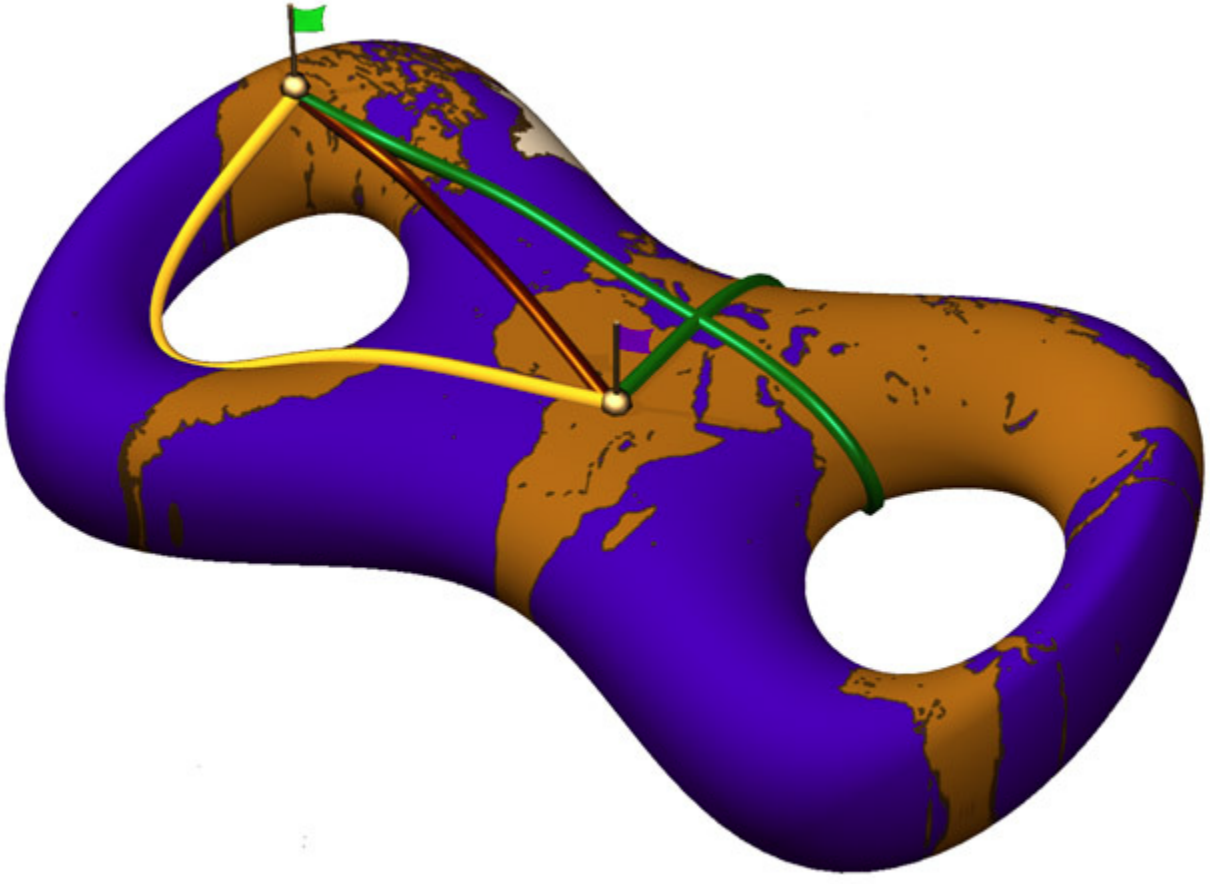
Algorithm 1 The Heat Method

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 2013.

Tracing Geodesic Curves



Trace gradient of distance function

Exact Geodesics

SIAM J. COMPUT.
Vol. 16, No. 4, August 1987
© 1987 Society for Industrial and Applied Mathematics
005

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

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 the result generally has less distortion and [illegible] if the cuts are geodesic. Geodesic paths [illegible] mesh into subparts, as done in [Kat[illegible]

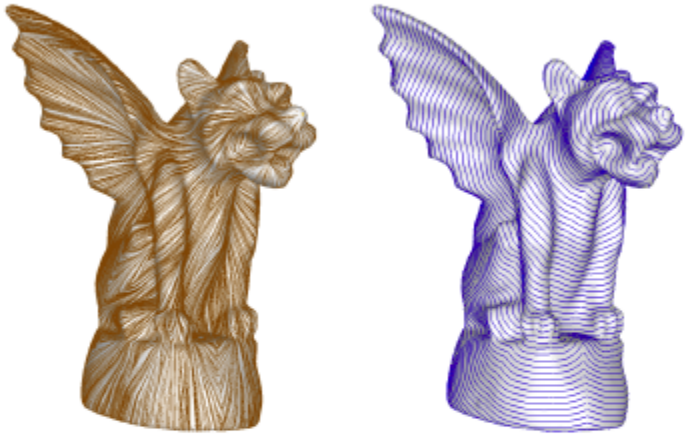


Figure 1: Geodesic paths from a source vertex, and isolines of the geodesic distance function.

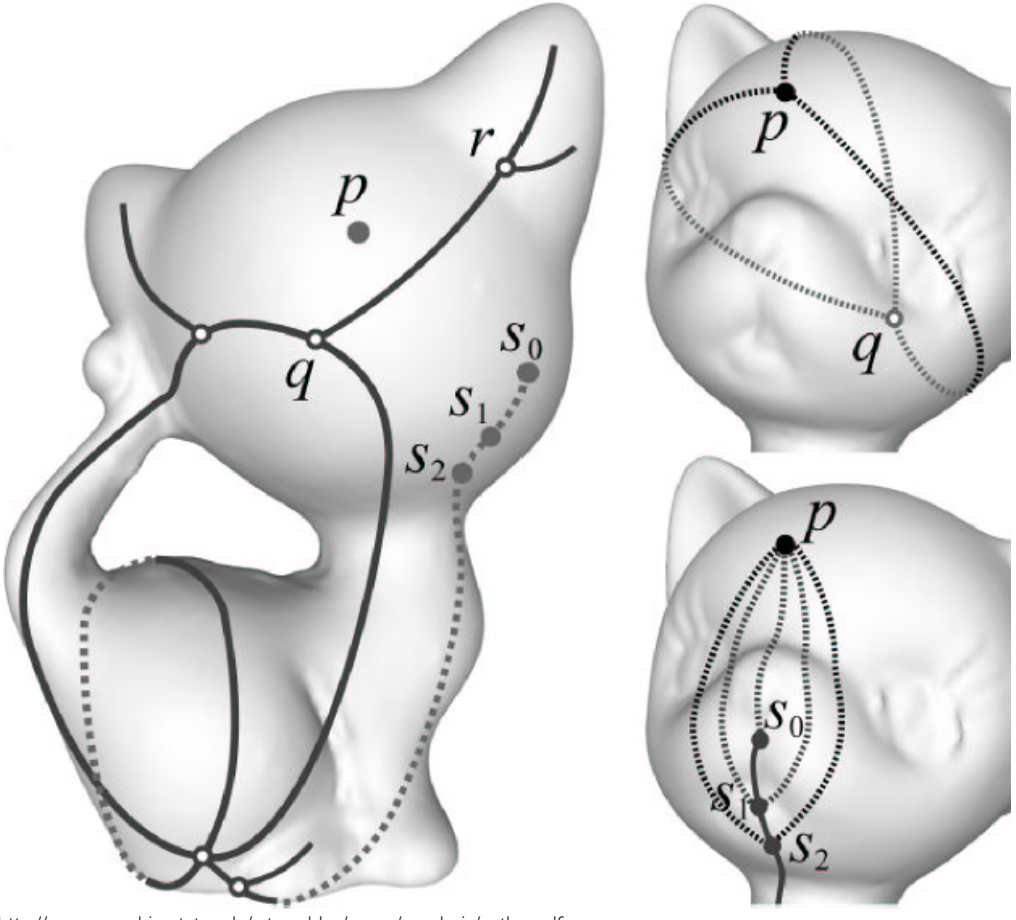
tance function over the edges, the implementation is actually practical even though, to our knowledge, it has never been done previously. We demonstrate that the algorithm's worst case running time of $O(n^2 \log n)$ is pessimistic, and that in practice, the algorithm runs in sub-quadratic time. For 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 to obtain computationally efficient and accurate approximations with *bounded* error. In practice, the algorithm runs in $O(n \log n)$ time even for small error thresholds.

Exact geodesic path between two points We show how the exact solution to the single source, single [illegible]

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

Recall: Cut Locus



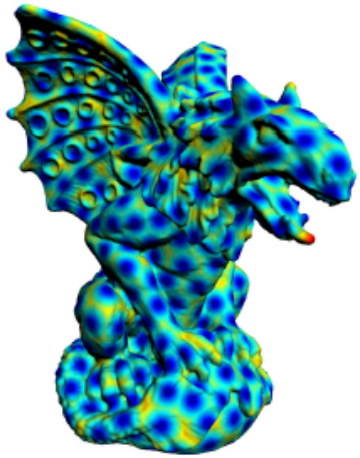
http://www.cse.ohio-state.edu/~tamaldey/paper/geodesic/cutloc.pdf

Cut point:
Point where geodesic ceases to be minimizing

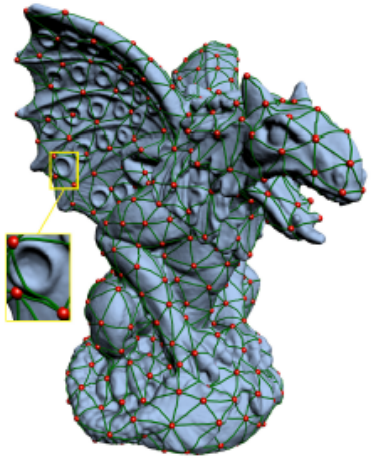
Set of cut points from a source p

All-Pairs Distances

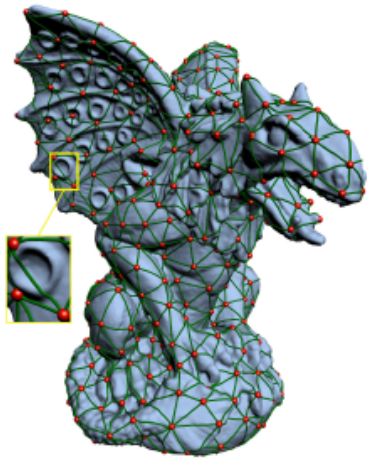
Sample points



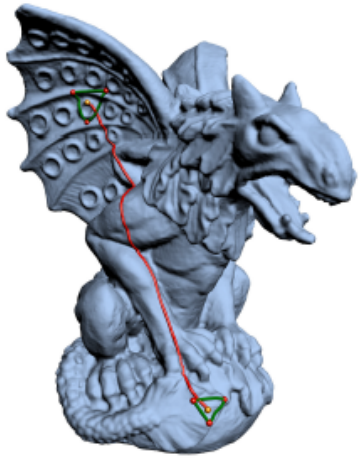
Geodesic field



Triangulate (Delaunay)



Fix edges



Query (planar embedding)

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

Geodesic Voronoi & Delaunay

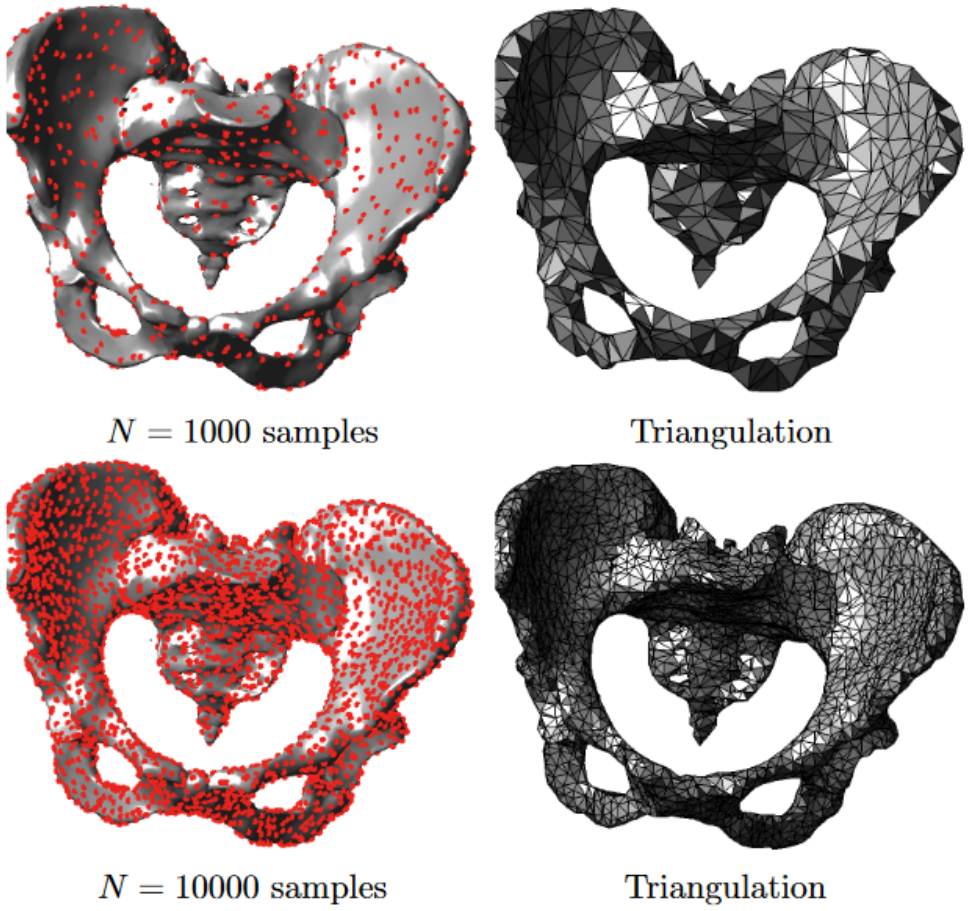


Fig. 4.12 *Geodesic remeshing with an increasing number of points.*

From *Geodesic Methods in Computer Vision and Graphics* (Peyré et al., FnT 2010)