CS 468 (Spring 2013) — Discrete Differential Geometry

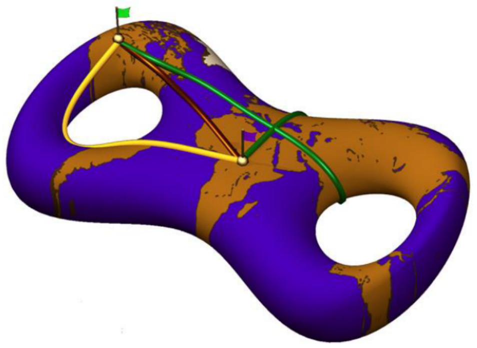
Lecture 10: Computing Geodesics
Justin Solomon and Adrian Butscher
Scribe: Trent Lukaczyk

1 Background

A Geodesic is a curve constrained to a surface which locally minimizes the intrinsic distance between two points, such that it satisfies the equation:

$$0 = \frac{d}{d\epsilon} \int_I ||\dot{\gamma}_\epsilon(t)||dt \Big|_{\epsilon=0} \tag{1}$$

This is can be posed as a optimization problem, however we are not able to expect that local minima are globaly optimum because we are able to trace more than one geodesic. Tracing the geodesics between the poles of a sphere would be an example of how we can draw multiple geodesics. Another example is shown below.



Straightest Geodesics on Polyhedral Surfaces (Polthier and Schmies)

Figure 1: Multiple Local Minima

Before calculating geodesics, we need to distinguish between extrinsic and intrinsic distance. Intrinsic distance is measured as an ant would walk along the surface. Extrinsic distance is defined as the three-space L2 norm between two points. The figure below shows that these two measures are often very different.

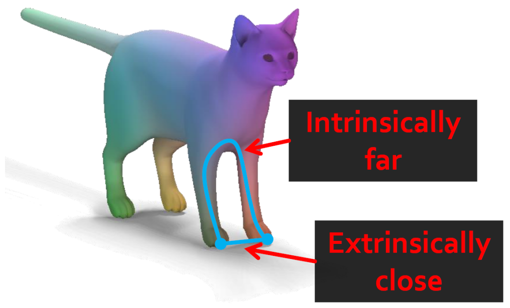


Figure 2: Intrinsic vs. Extrinsic Distance

We can calculate distances rigorously, or approximate them with the extrinsic distance in some cases. Additionally we can use marching algorithms to build level sets of distances from a source point. The methods presented in this lecture will address the issue of solving for distances and tracing the paths of geodesics.

2 Graph Shortest Distance

In this approach we calculate geodesics of a surface mesh by finding the length minimizing graph of edges. This is of course vulnerable to convergence issues.

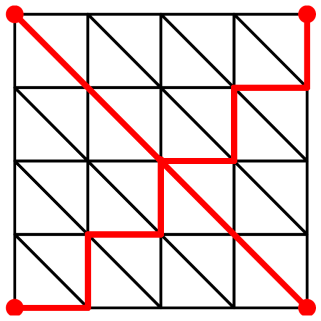


Figure 3: Graph-Based Distances

Convergence aside, this approach can be an acceptable approximation in many applications. It is also easily approached with a well known algorithm, Dijkstra's Algorithm.

2.1 Dijkstra's Algorithm

This is a simple and standard algorithm for graph-based geodesic calculations on a triangle surface mesh like the one shown in Figure 2. Two important principles behind this algorithm are -

- A shortest path had to come from somewhere,
- All steps of a shortest path are optimal.

The algorithm uses these principles to build an advancing front of paths with equal length to a source point, storing the optimal distance for each point as the front passes over it. The detailed algorithm is reproduced below.

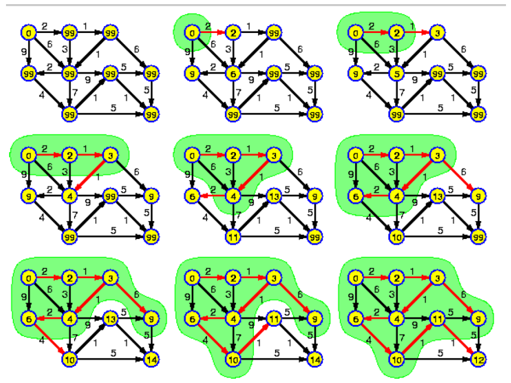
Listing 1: Dijkstra's Algorithm

```
$v_0$= initial vertex
$d_i$ = current distance to vertex $i$
S = verticies with known optimal distance

# initialize
$d_0 = 0$
$d_i$ = [inf for $d$ in $d_i$]
$S = \{\}$

for each iteration $k$:
    # update
    $k$ = argmin($d_k$), for $v_k$ not in $S$
    $S$.append($v_k$)
    for neighbors index $v_l$ of $v_k$:
        $d_l$ = min([$d_l$, $d_k$ + $d_{kl}$])
```

We can observe that at each iteration any new vertex of the advancing front is assigned a locally optimal distance to the initial vertex.



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

Figure 4: Example Dijkstra's Algorithm Evolution

3 Fast Marching Algorithm

This algorithm tries to address the limitation of graph shortest distance by advancing a planar front through each triangle. This is still an approximation, however.

The idea of fast marching algorithms is to propagate a wavefront of constant geodesic distances away from a source point. Near the source point this wavefront will be approximately circular (exactly circular on a plane). However, far away from the point, we can approximate the front as a line. The construction of this approach is shown below. The distance from an unknown source point (the red dot) and a vertex of interest (x_3) is measured as the length projected along a vector normal to a chosen wavefront line. Key here is that the distance from the source point and direction of the normal vector are unknowns for which we are solving.

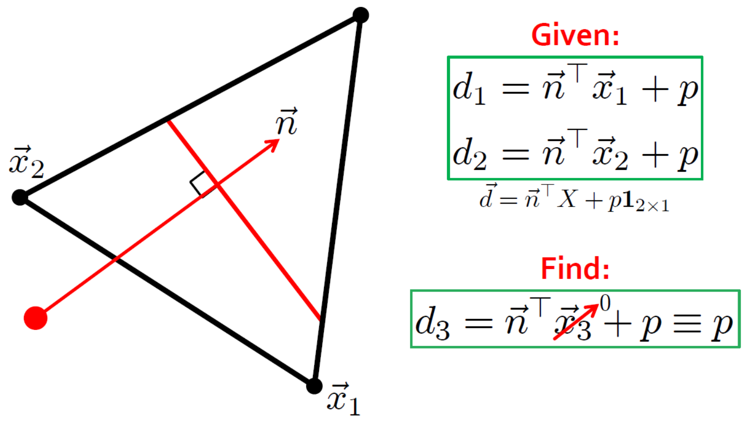

$\vec{x}_2$
$\vec{n}$
$\vec{x}_1$
Given:
$d_1 = \vec{n}^\top \vec{x}_1 + p$
$d_2 = \vec{n}^\top \vec{x}_2 + p$
$\vec{d} = \vec{n}^\top X + p\mathbf{1}_{2\times 1}$
Find:
$d_3 = \vec{n}^\top \overset{0}{\cancel{\vec{x}_3}} + p \equiv p$

Figure 5: Fast Marching Algorithm Geometry

We can formulate the distances from the unknown source point given three projections reproduced below.

$$d_1 = \vec{n}^\top \vec{x}_1 + p \tag{2}$$

$$d_2 = \vec{n}^\top \vec{x}_2 + p \tag{3}$$

$$d_3 = \vec{n}^\top \vec{x}_3 + p \equiv p \tag{4}$$

In equation 4 we have chosen x_3 as origin for convenience. We can formulate a system of linear equations

$$\vec{d} = \vec{n}^\top X + p\vec{1} \tag{5}$$

where the columns of matrix X contain x_1 and x_2, and the vector $\vec{d}$ contains distances d_1 and d_2. Solving for the normal vector yields

$$\vec{n} = X^{-\top}(\vec{d} - p\vec{1}). \tag{6}$$

Expressing the construction of a unit normal vector, we can derive a quadratic equation for the distance p -

$$\begin{aligned} 1 &= \vec{n}^\top \vec{n} \\ &= (\vec{d} - p\vec{1})^\top X^{-1} X^{-\top} (\vec{d} - p\vec{1}) \\ &= p^2 \cdot \vec{1}^\top Q \vec{1} - 2p \cdot \vec{1}^\top Q \vec{d} + \vec{d}^\top Q \vec{d} \\ &= d_3^2 \cdot \vec{1}^\top Q \vec{1} - 2d_3 \cdot \vec{1}^\top Q \vec{d} + \vec{d}^\top Q \vec{d} \end{aligned} \tag{7}$$

Solving for the distance d_3 is thus a quadratic problem, which has two roots. These two roots correspond to two different orientations of the normal vector - one that points from vertex x_3 into the triangle, and the other that points from the edge opposite x_3 towards x_3. We are concerned with the larger root, which makes an obtuse angle with x_3 because it is consistent with our model of the advancing front.

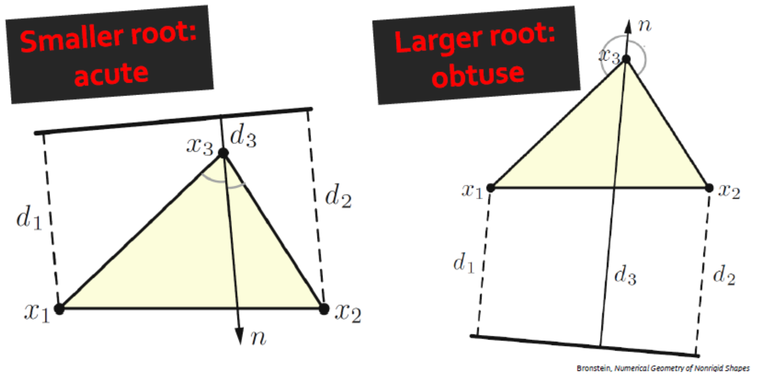


Figure 6: Two Roots of Fast Marching Normal Vector

We must also beware of the two cases in which the normal vector is not valid for the current triangle. The first case is when the normal vector does not pass through the triangle. The second case is when the wavefront passes through x_3 before another point. These cases requires us to fall back to Dijkstra's algorithm by updating the distance along the shortest edge. In the latter case we can also try splitting the triangle and re-attempting the algorithm.

The implementation of Fast Marching modifies Dijkstra's algorithm with a new update step for triangles instead of edges. This is shown in Listing 2 on the next page.

A take away from fast marching algorithms is that they are solving the Eikonal Equation,

$$||\nabla d|| = 1, \tag{8}$$

where d is the geodesic distance from the source point. This is a PDE that solves for geodesics. Note that Dijkstra's algorithm and Fast Marching are only approximate the Eikonal Equation and geodesic distance.

Listing 2: Fast Marching Algorithm

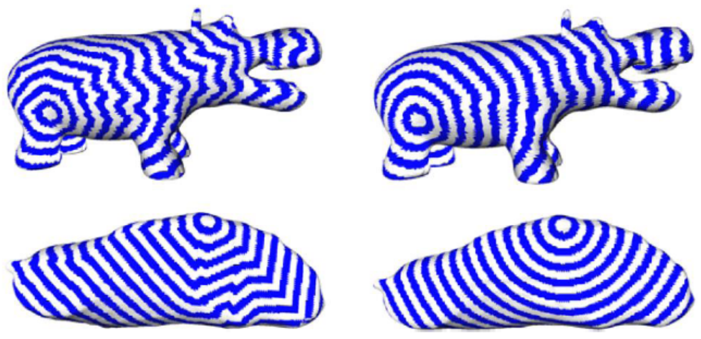
```
# update step
solve $p = \frac{1_{2\times1}^T Qd + \sqrt{(1_{2\times1}^T Qd)^2 - 1_{2\times1}^T Q 1_{2\times1} \cdot (d^T Qd - 1)}}{1_{2\times1}^T Q 1_{2\times1}}$
where $V = (x_1 - x_3, x_2 - x_3)$, and $d = (d_1, d_2)^T$
compute front propagation direction $n = V^{-T}(d - p \cdot 1_{2\times1})$

# check problematic cases
if $(V^T V)^{-1} V^T n < 0$ then
    # outside triangle
    $d_3 \leftarrow \min\{d_3, p\}$
else
    # obtuse triangle
    $d_3 \leftarrow \min\{d_3, d_1 + ||x_1||, d_2 + ||x_2||\}$
end
```

4 Other Single-Source Geodesic Approximations

4.1 Circular Wave Front

There are many other ways to approach the marching wavefront approximation to calculating geodesics. One is to construct circular wavefronts instead of planar wavefronts.



Novotni and Klein 2002

Figure 7: Circular Wave Front

4.2 Heat Equation

Another way is to integrate forward the heat equation, with an initial condition of a heat source at the initial vertex.

Listing 3: Heat Equation Geodesic Algorithim

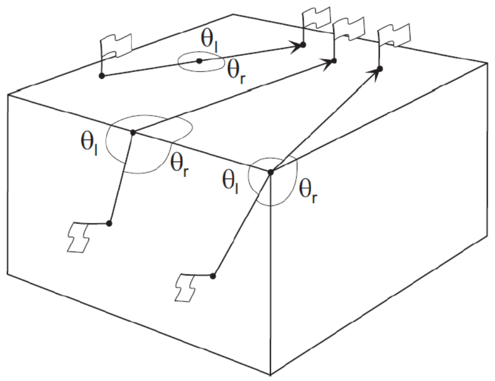
```
# integrate heat flow
$\dot{u} = \Delta u$ for time $t$

# evaluate vector field
$X = -\nabla u / |\nabla u|$

# solve the Poisson equation
$\Delta \phi = \nabla \cdot X$
```

5 Tracing Geodesics

An alternative to solving for a geodesic is to trace it forward from an initial point. This is in a sense solving an ODE instead of a PDE. A simple algorithm can be built on a triangle mesh by tracing a path around edges. Since geodesics trace a path of shortest distance between points, then we should expect that paths between points on two adjacent facets maintain this locally. Observing that the path traced across the adjoining edge of two facets is isometric (we can fold the facets around this edge without changing the path), we can reasonably say that a straight line along the facets between these two points is geodesic. Thus tracing a geodesic forward can be accomplished using an update rule that enforces equal angles to the left and right of the paths on either face.



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

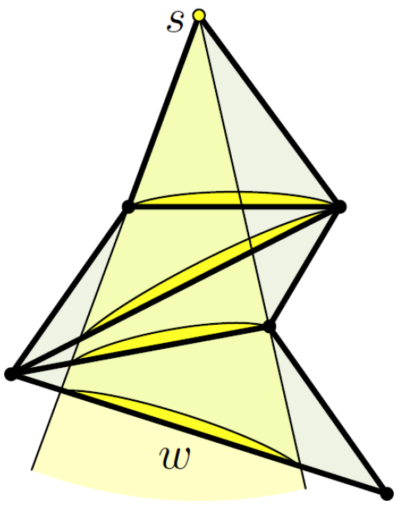
Figure 8: Geodesic Tracing

One boundary condition to beware of is when the path intersects a vertex. In this case angles across extra faces must be considered. It is also important to note that the geodesics traced in this way are merely locally minimizing. This leaves open the possibility that there are shorter paths between the two points.

6 Other Algorithms

6.1 MMP Algorithm

This algorithm attempts to calculate 'exact' geodesics by propagating 'windows' that follow back to the source point. The cost of this algorithm is $\mathcal{O}(n^2 \log n)$.



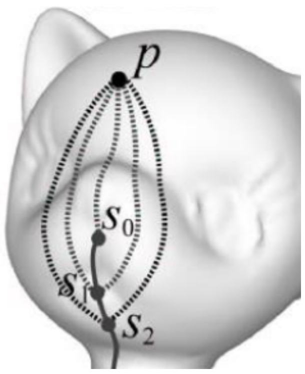
Surazhsky et al. "Fast Exact and Approximate Geodesics on Meshes." SIGGRAPH 2005.

Figure 9: MMP Marching Windows Algorithm

6.2 Cut Locus

Geodesics can be unstable, such that there is more than one geodesic between two points. One interesting application of this is identifying the cut locus.

In the example below, several curves trace from source point p. In the case of target point s_0, the geodesic between these points is length-minimizing. However, we are able to trace two geodesics of equal length to point s_1 or s_2. These are called cut points. The line connecting points s_{0-2} shows the cut locus, or collection of points for which the geodesics issued from point p are no longer length-minimizing.



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

Figure 10: Cut Locus Example

6.3 Fuzzy Geodesics

This approach represents the geodesic stochastically by building a probability distribution map on the surface. One way to approach this is to construct a measure of the difficulty required to move from source point p to target point q through some point x on the surface,

$$G^{\sigma}_{p,q}(x) \equiv \exp(-\left|d_M(p,x) + d_M(x,q) - d_M(p,q)\right|/\sigma), \tag{9}$$

where $d(\cdot,\cdot)$ is the distance along the surface between to points. An example of the probability density function on a surface is shown below. We can see that the interpretation of the geodesic is no longer a line, but a region through which a geodesic could pass with high likelihood.

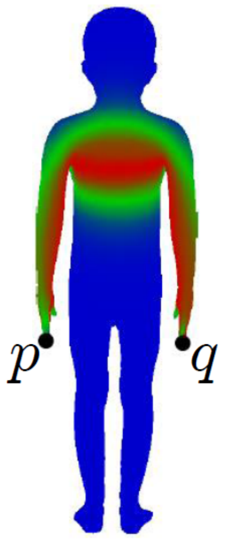
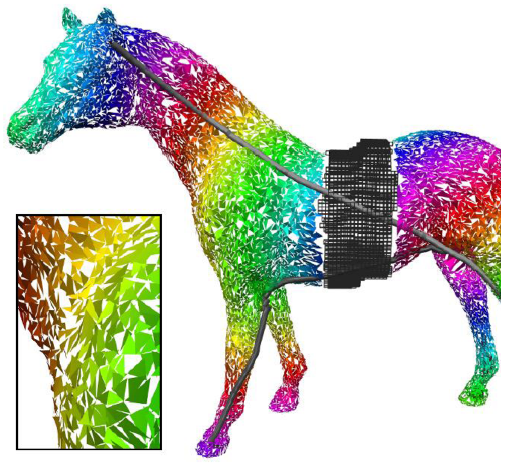


Figure 11: Fuzzy Geodesic Example

This approach though not exact, is very stable. It also admits a nice property that the intersection of two geodesics can be expressed as a point-wise multiplication.

6.4 Morphological Scaffolding

Finally, this approach builds a background lattice on the surface that allows geodesics to be approximated across surface holes (ie from poor 3D scanning) or through triangle soup meshes.



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

Figure 12: Morphological Scaffolding Example