Lecture 25: Geodesic Paths

COMPSCI/MATH 290-04

Chris Tralie, Duke University

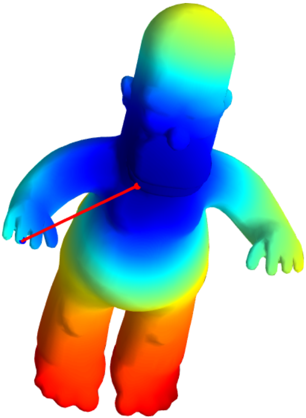
4/14/2016

Announcements

- Group Assignment 3 Out: First Deadline Monday 4/18. Final Deadline Tuesday 4/26
- Final Project Final Deadline 5/3 5:00 PM

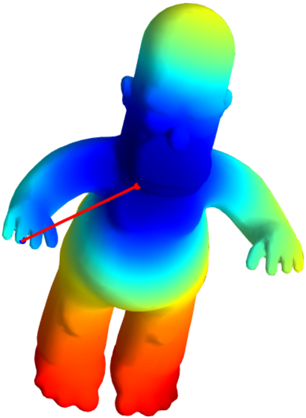
Table of Contents

Geodesic Paths

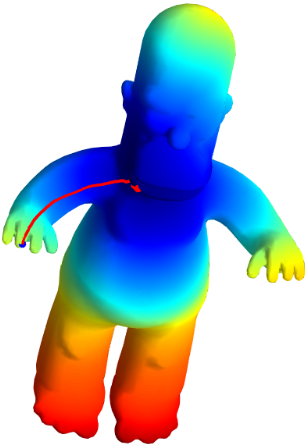


Euclidean Path (shortest path of flying fly)

Geodesic Paths

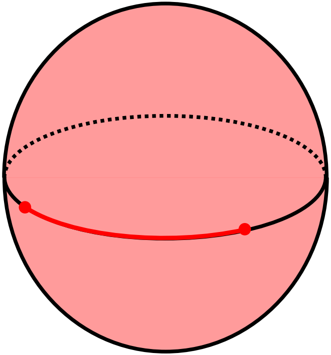


Euclidean Path (shortest path of flying fly)

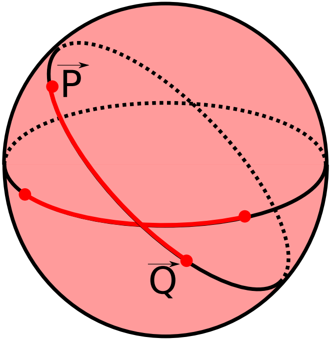


Geodesic Path (shortest path of crawling ant)

Geodesic Paths on Spheres

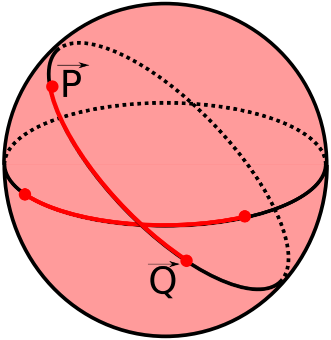


Geodesic Paths on Spheres



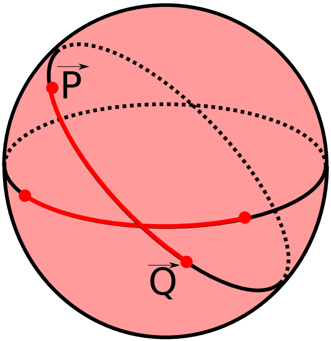
- ▷ Geodesic paths on spheres lie along great circles

Geodesic Paths on Spheres



- ▷ Geodesic paths on spheres lie along great circles
- ▷ Geodesic distance is the shortest geodesic path

Geodesic Paths on Spheres



- Geodesic paths on spheres lie along great circles
- Geodesic distance is the shortest geodesic path
- *What is the geodesic distance between two points $\vec{P}$ and $\vec{Q}$ on a sphere centered at the origin with radius R?*

Geodesic Paths on Spheres

What is the geodesic distance between two points $\vec{P}$ and $\vec{Q}$ on a sphere centered at the origin with radius R?

$$R\cos^{-1}\left(\frac{\vec{P}\cdot\vec{Q}}{||\vec{P}||||\vec{Q}||}\right) = R\cos^{-1}\left(\frac{\vec{P}\cdot\vec{Q}}{R^2}\right)$$

Remember SLERP??

Another Geodesic Mesh Example

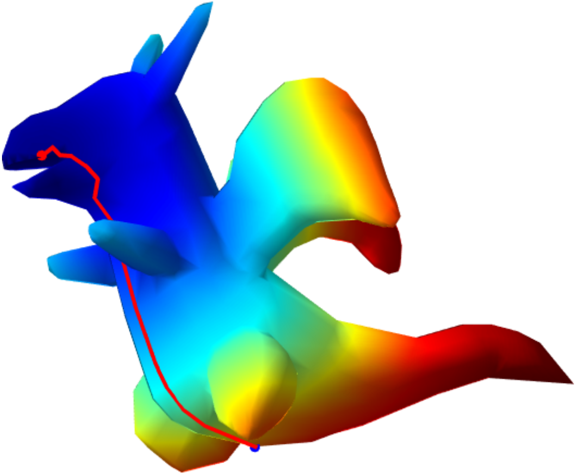


Table of Contents

Dijkstra’s Algorithm Review

```
def Dijsktra(Graph, source):
    list dists
    list prev
    dist[source] = 0
    Queue Q
    for vertex v in Graph:
        if v not source:
            dists[v] = Infinity
            prev[v] = Undefined
        Q.add(v, dists[v])
    while len(Q) > 0:
        u = Q.getMin()
        for v in neighbors(u):
            d = dists[u] + length(u, v)
            if d < dists[v]:
                dists[v] = d
                prev[v] = u
                Q.decreasePriority(v, d)
    return (dist, prev)
```

Dijkstra's Algorithm Review

```
def Dijsktra(Graph, source):
    list dists
    list prev
    dist[source] = 0
    Queue Q
    for vertex v in Graph:
        if v not source:
            dists[v] = Infinity
            prev[v] = Undefined
        Q.add(v, dists[v])
    while len(Q) > 0:
        u = Q.getMin()
        for v in neighbors(u):
            d = dists[u] + length(u, v)
            if d < dists[v]:
                dists[v] = d
                prev[v] = u
                Q.decreasePriority(v, d)
    return (dist, prev)
```

What is the worst case behavior for

- V vertices
- E edges

for a balanced min heap Q?

Dijkstra's Algorithm Review

```
def Dijsktra(Graph, source):
    list dists
    list prev
    dist[source] = 0
    Queue Q
    for vertex v in Graph:
        if v not source:
            dists[v] = Infinity
            prev[v] = Undefined
        Q.add(v, dists[v])
    while len(Q) > 0:
        u = Q.getMin()
        for v in neighbors(u):
            d = dists[u] + length(u, v)
            if d < dists[v]:
                dists[v] = d
                prev[v] = u
                Q.decreasePriority(v, d)
    return (dist, prev)
```

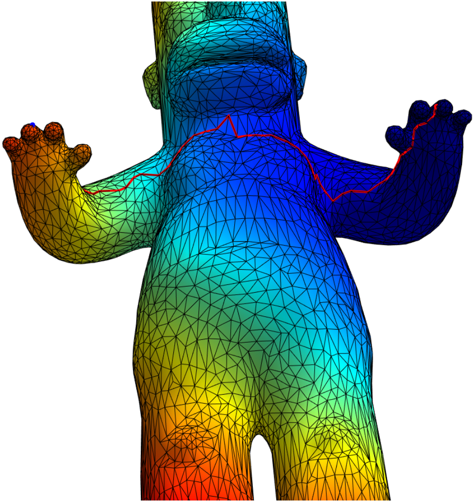
What is the worst case behavior for

- V vertices
- E edges

for a balanced min heap Q?

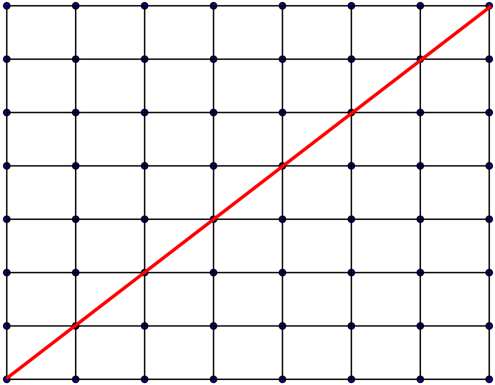
$O((E+V)\log(V))$

Dijkstra's Directly on Mesh Edges



Dijkstra's Directly on Mesh Edges

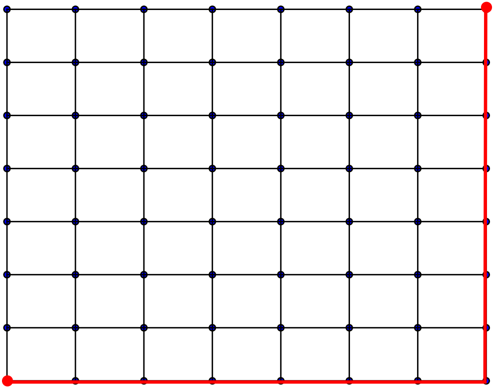
8x8 Cartesian Grid: Side Length 1



Shortest path along mesh is length $7\sqrt{2}$

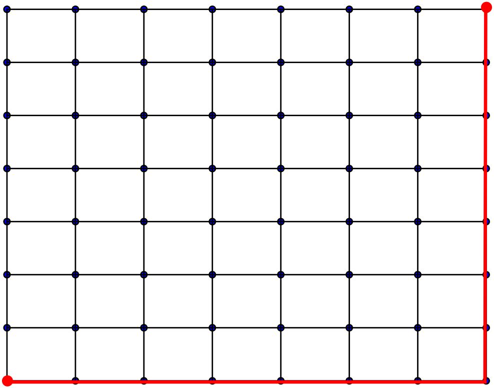
Dijkstra's Directly on Mesh Edges

8x8 Cartesian Grid: Side Length 1



Dijkstra's Directly on Mesh Edges

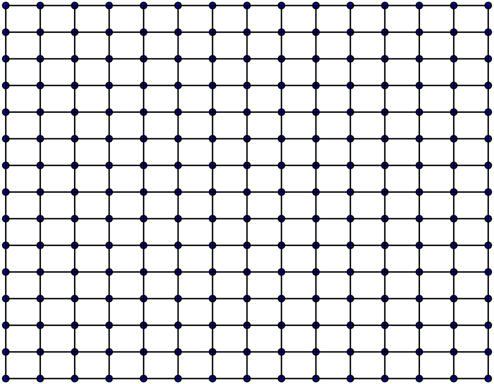
8x8 Cartesian Grid: Side Length 1



Shortest path along mesh is 14

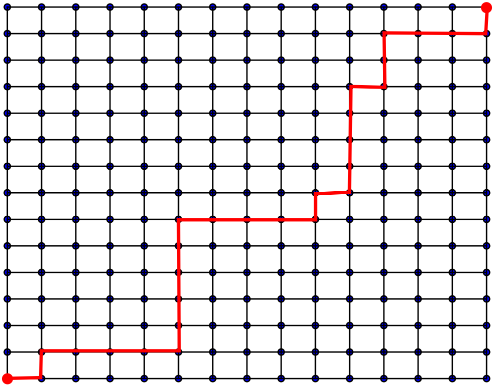
Dijkstra's Directly on Mesh Edges

Does refining the grid help?
15x15 Cartesian Grid: Side Length 0.5



Dijkstra’s Directly on Mesh Edges

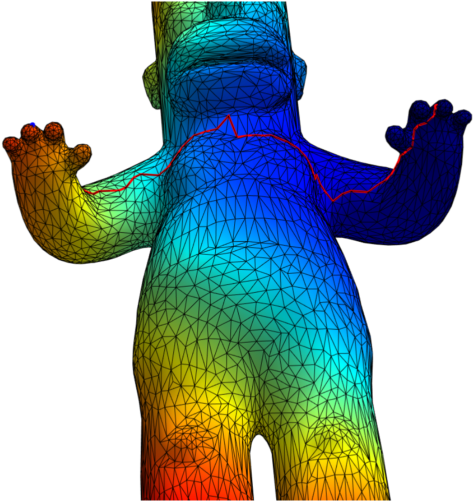
Does refining the grid help?
15x15 Cartesian Grid: Side Length 0.5



Nope!

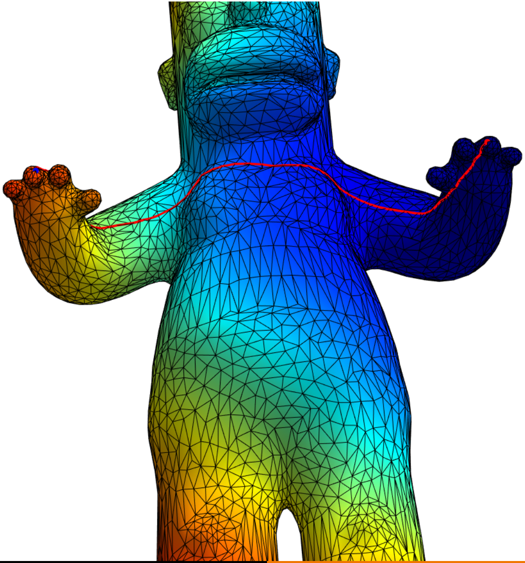
Dijkstra’s Directly on Mesh Edges

In general, mesh biases the solution!



Fast Marching

A modification of Dijkstra's algorithm to cut through triangles



Fast Marching

A modification of Dijkstra's algorithm to cut through triangles

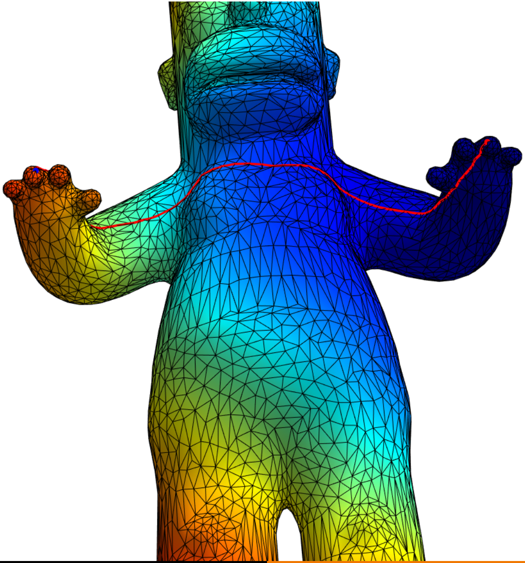
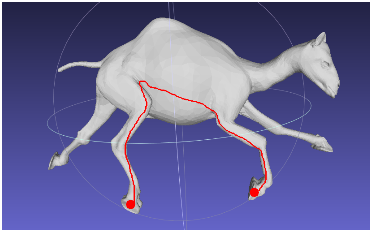
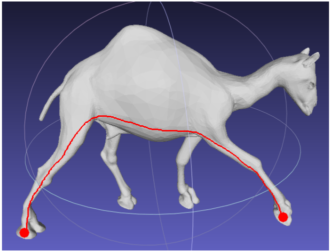


Table of Contents

Mesh Isomorphisms

An isomorphism preserves all pairwise geodesic distances





Mesh Isomorphisms

Contrast with Euclidean