Shortest Paths with Arbitrary Clearance from Navigation Meshes

Marcelo Kallmann

University of California, Merced

Motivation

- Path planning is important for navigation
  - For finding paths for autonomous agents
  - For determining if locations are reachable
  - For obtaining distances to possible targets
  - etc
- Useful paths are not that easy to compute
  - Guaranteed clearance is often needed
  - Paths have to be short and smooth
  - Efficiently computed
  - etc

Euclidean Shortest Paths

- Globally shortest paths are difficult to find
  - Is a global problem in 2D
  - Ex: the shortest path in the medial axis graph of a given environment may not be continuously deformable to the shortest path in the environment
    - Medial axis can only give "locally shortest paths"
- The problem can be solved in O (n log n)
  - n = num. input segments
  - Continuous Dijkstra approach solves it
  - Difficult to implement and to consider clearance

Considering Clearance

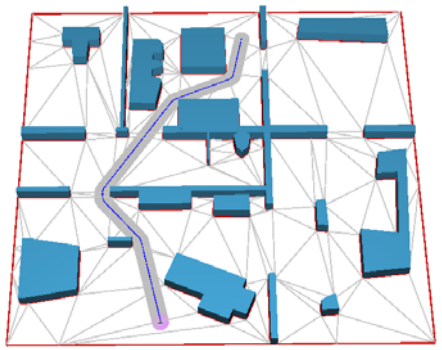
- Preprocessing for a specific clearance value
  - Lozano-Pérez and Wesley 1979
  - Chew 1985
    - First dilates the environment, then uses visibility graph of tangents
    - Precomputation: O (n^2 log n), produces graph with O (n^2) elements
    - Query: O (n^2 log n)
- Clearance-independent preprocessing possible
  - Wein, van den Berg and Halperin, "the visibility–Voronoi complex and its applications", 2007
    - Preprocessing: O (n^2 log n)
    - Query time: O *(n* log *n* + m *)* = *O (n^2)*
    - *Probably the best practical method for global optimality*

Related Work in Animation

- Multi Agents
  - Elastic roadmaps
    - Gayle, Sud, Andersen, Guy, Lin and Manocha, 2009
  - Multi agent navigation graphs
    - Sud, Andersen, Curtis, Lin and Manocha, 2008
  - Velocity Obstacles
    - van den Berg, Lin and Manocha, 2008
- Methods based on the medial axis or Voronoi diagrams
  - Hardware acceleration also used
    - Hoff, Culver, Keyser, Lin and Manocha 2000
  - Corridor Maps are very related
    - Geraerts and Overmars, 2007
    - Geraerts, 2010
- Methods based on randomized planners

Proposed Work

- New navigation mesh that enables the efficient computation of locally shortest paths with clearance



Navigation Meshes

- Navigation meshes are very popular
  - Google results: “navigation mesh” 14.7M, “visibility graph” 5.1M
    - *Update of Jul 12: now Google shows: 4.1M and 4.2M*
  - Meshes are directly available from environment geometry
  - Meshes are known structures
- However
  - Navigation meshes are not well defined, many techniques used
  - Difficult to obtain paths with clearance
  - Difficult to guarantee optimality aspects
- Reality
  - Globally shortest paths (with or without clearance) cannot be easily extracted from navigation meshes

Navigation Meshes

- Only few previous work based on triangulated meshes
  - Kallmann et al 2003
  - Lamarche and Donikian 2004
  - Demyen and Buro 2006
- However
  - Optimality not (or not well) addressed
  - Clearance not (or not correctly) addressed

Method Overview

- Summary of algorithms
  - 1. Computation of the Navigation Mesh
  - 2. Channel Search
  - 3. Locally Shortest Path Extraction from Channel
  - 4. Extended Search for Globally Optimal Path

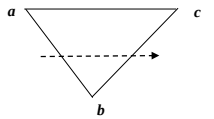
1. Mesh Computation

Which Navigation Mesh to Use?

- Triangulations
  - Easier to work with triangle primitives
  - Triangulation of n segments usually have O(n) triangles
- Starting point: CDTs
  - Obstacles segments become constraints
  - Can be computed in O (n log n)

Triangle Clearance

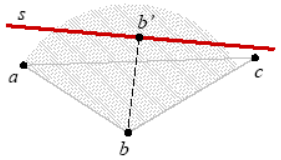
- Definition: Triangle Traversal τ_{abc}



- Definition: Traversal clearance

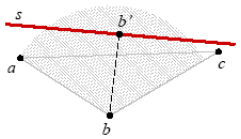
$cl(a,b,c)$ = Distance (b, closest vertex or constraint)

$$cl(a,b,c) \leq min\{dist(b,a), dist(b,c)\}$$



Clearances are Precomputed

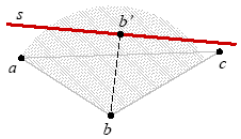
- For every traversal abc :
  - Compute its clearance and store it in the triangulation
  - Two values stored per edge



- Path with clearance r may pass traversal abc if:
  r ≤ $cl(a,b,c)$

Clearances are Precomputed

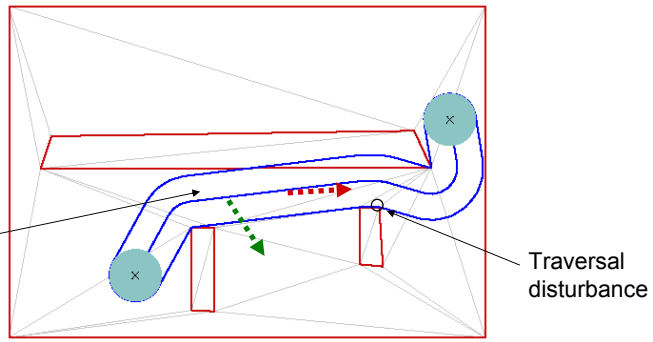
- For every traversal abc :
  - Compute its clearance and store it in the triangulation
  - Two values stored per edge



- Path with clearance r may pass traversal abc if:
  r ≤ $cl(a,b,c)$

CDTs Are Not Enough

- The subdivision has to be able to allow local clearance tests

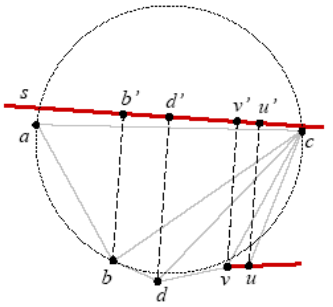


Detecting Disturbances

- Given a traversal τ_{abc}
  - traversal τ_{bcd} possible
  - v is connected to c

v is a disturbance to traversal τ_{abc} if:

1. v can be orthogonally projected on ac,
2. v is not shared by two collinear constraints,
3. $dist(v,s) < cl(a,b,c)$, and
4. $dist(v,s) < dist(v,c)$.

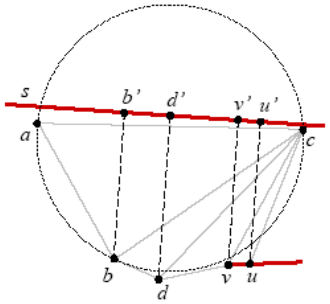


Detecting Disturbances

- Given a traversal τ_{abc}
  - traversal τ_{bcd} possible
  - v is connected to c

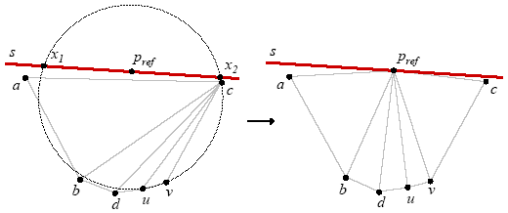
v is a disturbance to traversal τ_{abc} if:

1. v can be orthogonally projected on ac,
2. v is not shared by two collinear constraints,
3. $dist(v,s) < cl(a,b,c)$, and
4. $dist(v,s) < dist(v,c)$.



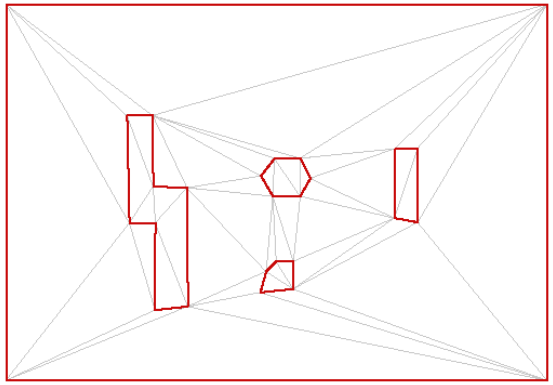
Computing LCTs

- Check all traversals
- Every time a disturbance is found
  - Subdivide traversal
  - One good refinement point is the mid point between x_1 and x_2, which are the intersections of s and circle (uvc)



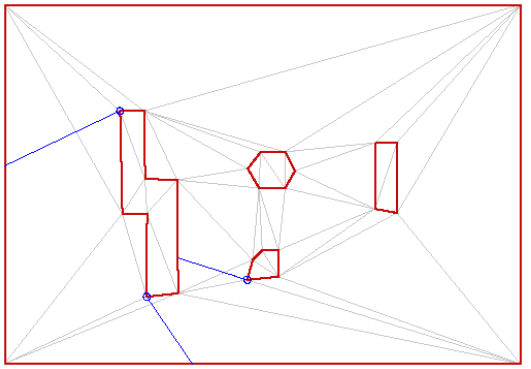
Computing LCTs

- Starting Point: CDT



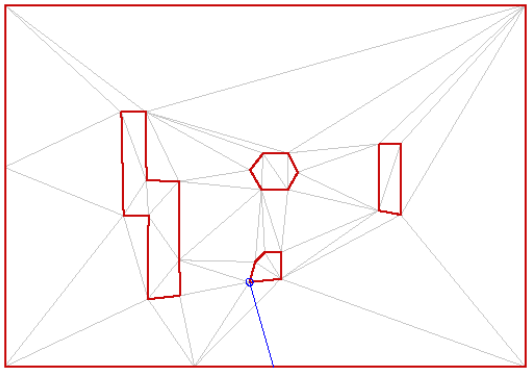
Computing LCTs

- Iteration 1: Detect all disturbances and refine them



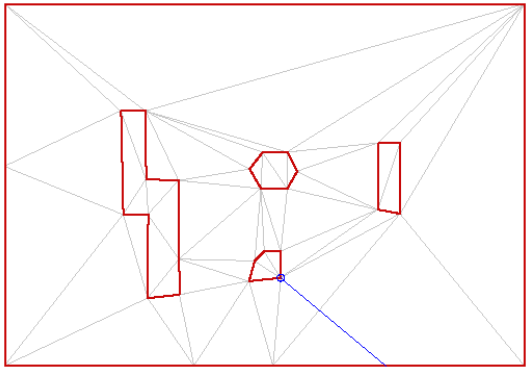
Computing LCTs

- Iteration 2: Repeat until no more disturbances



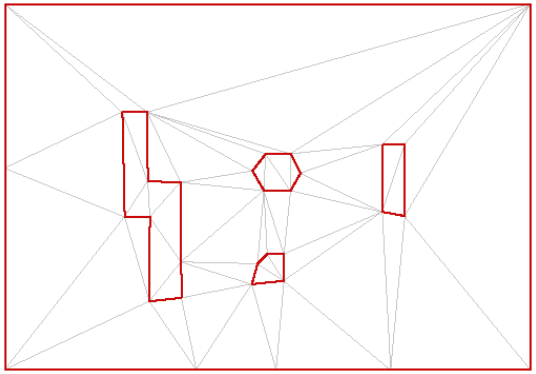
Computing LCTs

- Iteration 3: Repeat until no more disturbances



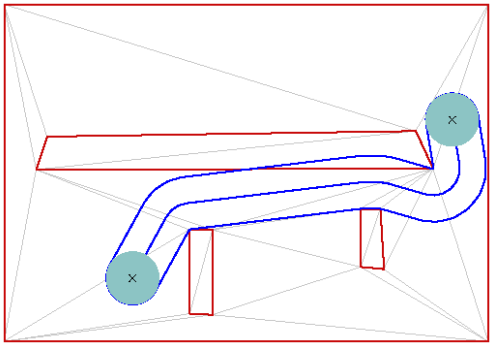
Computing LCTs

- Iteration 4: no more disturbances: LCT found



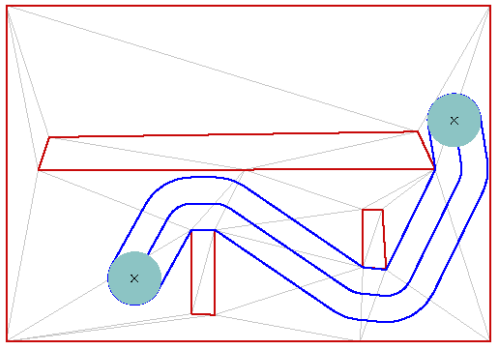
CDT

- Example 1



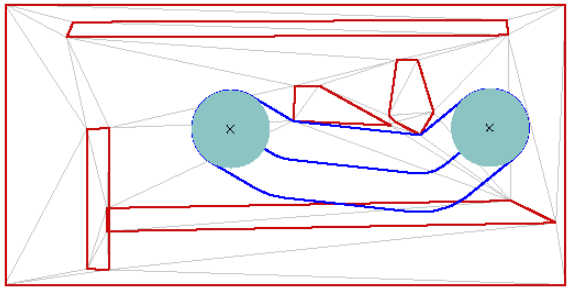
LCT

- Local clearance tests work in LCTs



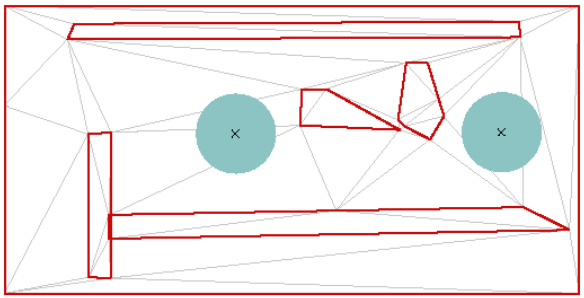
CDT

- Example 2



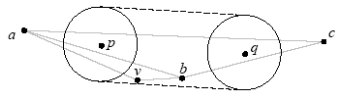
LCT

- Local clearance tests work in LCTs: no path

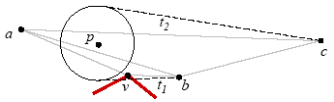


There are Special Cases

- Clearance tests assume crossing entrance and exit edges
- Two special cases to handle:
  - Capsule tests



  - Departure/arrival tests



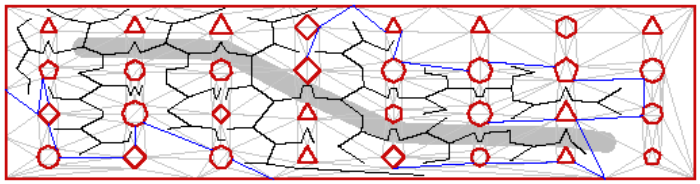
2. Channel Search

Channel Search

- First, locate initial point p
  - If not already known, use oriented walk method
- Graph search on the adjacency graph
  - For each triangle traversed, check clearance
  - O (n log n) if priority queue used
  - O (n) possible
    - By exploiting the planarity of the graph
    - Linear-time shortest path algorithm available
      - Henzinger, Klein, Rao, Subramanian, "Faster Shortest-Path Algorithms for Planar Graphs" 1977, 1994

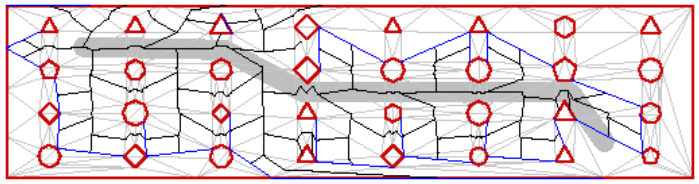
Adjacency Graph

- Metric 1: Use center of triangles
  - Not good: too many zig zags



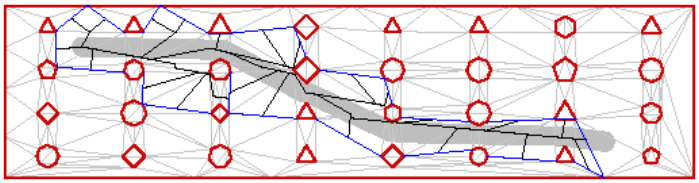
Adjacency Graph

- Metric 2: Use center of edges
  - Much better



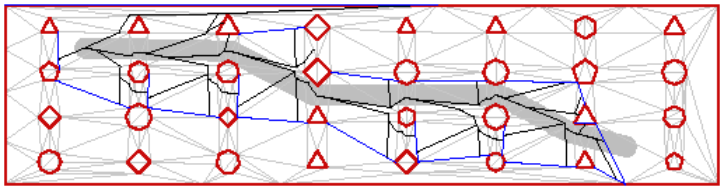
Adjacency Graph

- Metric 3: adds a visibility criterion to the goal to not miss straight line solution paths
  - It is always important to not miss straight line solutions



Adjacency Graph

- Metric 4: improves previous metric by considering clearance value
  - A good metric without considering more time-consuming strategies (like backtracking)

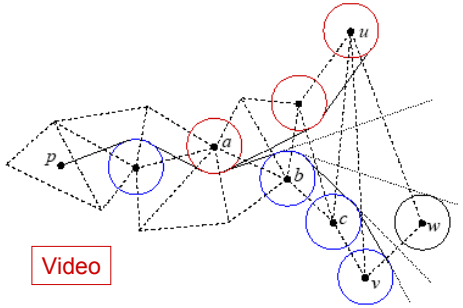


Video

3. Paths from Channels

Paths from Channels

- Given a channel connecting p and q
  - Navigation mesh already gives channel triangulated
- Find Shortest path in the channel
  - Funnel algorithm is efficient: O(h), h length of channel



Video

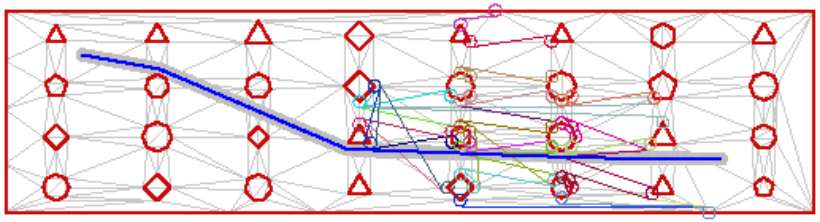
4. Optimal Search

Optimal Search

- Extended algorithm needed for finding the global optimal
  - First find the locally shortest path, use its length as a current solution upper bound
  - Then run new search, where each front maintains a funnel
    - Each front maintains lower and upper bounds for all path lengths passing by the front's funnel
  - Allow fronts to overlap when needed, a front dies when it may not lower current solution length

Optimal Search

- Example: environment with high branching factor: O(n) junctions
  - Many overlaps may occur



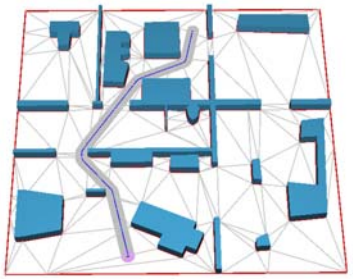
Optimal Search

- A more standard environment: very few overlaps

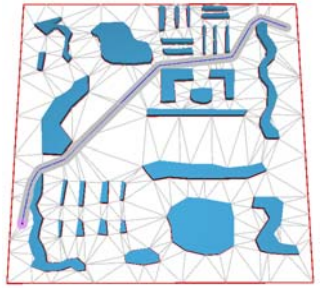
Results

Test Environments

env1, n=171

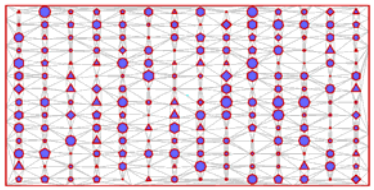


env2, n=313

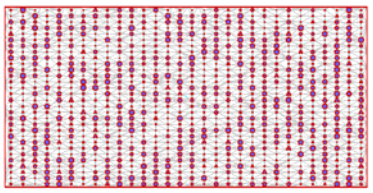


Test Environments

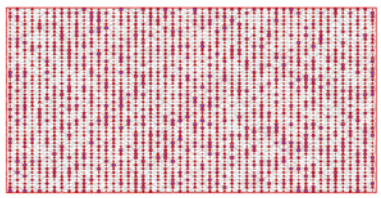
obst1k



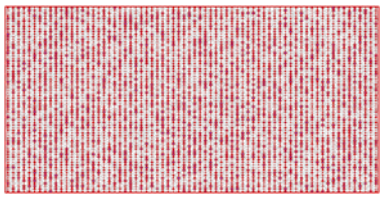
obst5k



obst15k

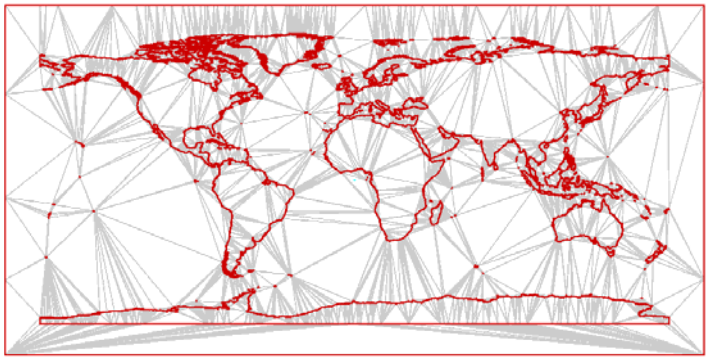


obst30k



Test Environments

World map with large variation of triangle sizes, n=63K



Performance

environment	n	refinements	LCT triangles	t_{local} (ms)	t_{global} (ms)	len_{local}	len_{global}
env1	171	16	333	0.09	0.22	8.75	8.74
env2	313	29	687	0.11	0.29	55.8	55.7
obst1k	1207	19	2455	0.19	1.48	159.6	159.4
obst5k	5550	44	11191	0.49	13.51	156.4	156.1
obst15k	14963	88	30105	1.49	62.88	164.5	164.0
obst30k	29198	105	58609	2.28	206.99	168.1	167.6
world map	63426	2198	131247	3.11	401.92	212.0	210.1

(Intel Quad Core Q 9450 @ 2.66 GHz)

Performance

environment	n	refinements	LCT triangles	t_{local} (ms)	t_{global} (ms)	len_{local}	len_{global}
env1	171	16	333	0.09	0.22	8.75	8.74
env2	313	29	687	0.11	0.29	55.8	55.7
obst1k	1207	19	2455	0.19	1.48	159.6	159.4
obst5k	5550	44	11191	0.49	13.51	156.4	156.1
obst15k	14963	88	30105	1.49	62.88	164.5	164.0
obst30k	29198	105	58609	2.28	206.99	168.1	167.6
world map	63426	2198	131247	3.11	401.92	212.0	210.1

Relatively few LCT refinements needed

Performance

environment	n	refinements	LCT triangles	t_{local} (ms)	t_{global} (ms)	len_{local}	len_{global}
env1	171	16	333	0.09	0.22	8.75	8.74
env2	313	29	687	0.11	0.29	55.8	55.7
obst1k	1207	19	2455	0.19	1.48	159.6	159.4
obst5k	5550	44	11191	0.49	13.51	156.4	156.1
obst15k	14963	88	30105	1.49	62.88	164.5	164.0
obst30k	29198	105	58609	2.28	206.99	168.1	167.6
world map	63426	2198	131247	3.11	401.92	212.0	210.1

Locally shortest paths computed very efficiently

Performance

environment	n	refinements	LCT triangles	t_{local} (ms)	t_{global} (ms)	len_{local}	len_{global}
env1	171	16	333	0.09	0.22	8.75	8.74
env2	313	29	687	0.11	0.29	55.8	55.7
obst1k	1207	19	2455	0.19	1.48	159.6	159.4
obst5k	5550	44	11191	0.49	13.51	156.4	156.1
obst15k	14963	88	30105	1.49	62.88	164.5	164.0
obst30k	29198	105	58609	2.28	206.99	168.1	167.6
world map	63426	2198	131247	3.11	401.92	212.0	210.1

Globally shortest paths require more computation time

Performance

environment	n	refinements	LCT triangles	t_{local} (ms)	t_{global} (ms)	len_{local}	len_{global}
env1	171	16	333	0.09	0.22	8.75	8.74
env2	313	29	687	0.11	0.29	55.8	55.7
obst1k	1207	19	2455	0.19	1.48	159.6	159.4
obst5k	5550	44	11191	0.49	13.51	156.4	156.1
obst15k	14963	88	30105	1.49	62.88	164.5	164.0
obst30k	29198	105	58609	2.28	206.99	168.1	167.6
world map	63426	2198	131247	3.11	401.92	212.0	210.1

Locally shortest paths are very close to the globally shortest ones

Performance

environment	n	refinements	LCT triangles	t_{local} (ms)	t_{global} (ms)	len_{local}	len_{global}
env1	171	16	333	0.09	0.22	8.75	8.74
env2	313	29	687	0.11	0.29	55.8	55.7
obst1k	1207	19	2455	0.19	1.48	159.6	159.4
obst5k	5550	44	11191	0.49	13.51	156.4	156.1
obst15k	14963	88	30105	1.49	62.88	164.5	164.0
obst30k	29198	105	58609	2.28	206.99	168.1	167.6
world map	63426	2198	131247	3.11	401.92	212.0	210.1

More variation observed when triangles have “irregular sizes”

Conclusions and Final Remarks

Main Conclusions

- LCTs give a correct methodology for extracting paths of arbitrary clearance directly from the LCT mesh
- Locally shortest paths are obtained efficiently and in optimal times
  - Preprocessing: O (n^2), maybe O (n log n)
  - Query time: O (n), current implementation O (n log n)

Proofs will be Available Soon

- T1: Marking cells may not block corridors
  - CDT property guarantees it
- T2: Refinement indeed solves disturbance
  - Due location of p_{ref} and CDT properties
- T3: LCT remains with O(n) triangles
  - Bounded internal angle and prevention of propagations
- T4: Local clearance tests and special cases are enough
  - Concatenation of traversals shows it

Extensions and Variations

- Many extensions possible
  - Corridors without junctions can be reduced to one link in the adjacency graph
  - Handling additional properties: non-flat terrains, multi-layers, etc
  - Reactive behaviors based on path queries
- Navigation Meshes are very useful
  - Ray sensors easily computed
  - Intersection tests for placement of disc-agents, etc
- Many of these tools already implemented
  - Implementation code to be available soon
  - Please keep checking our web site

Final Notes

- Not all references in these slides are in the paper
- Project page will maintain latest info
  - http://graphics.ucmerced.edu

Video

Thank You