Online Computation of Euclidean Shortest Paths in Two Dimensions

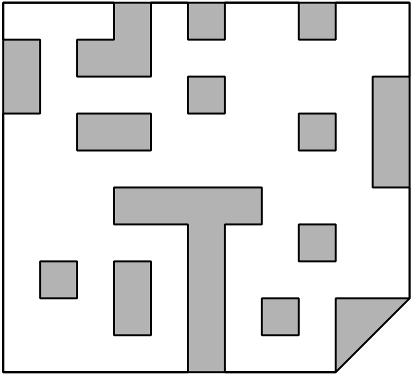
ICAPS 2020

Ryan Hechenberger, Peter J Stuckey, Daniel Harabor, Pierre Le Bodic, Muhammad Aamir Cheema

Monash University

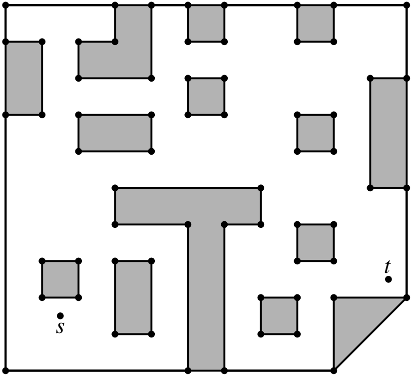
Euclidean Shortest Path

- Pathfinding on the Euclidean plane



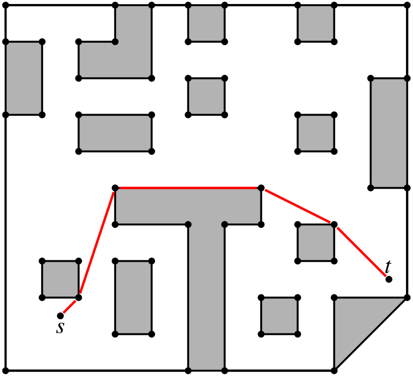
Euclidean Shortest Path

- Pathfinding on the Euclidean plane
- Takes as input start s, target t and a set of polygonal obstacles P



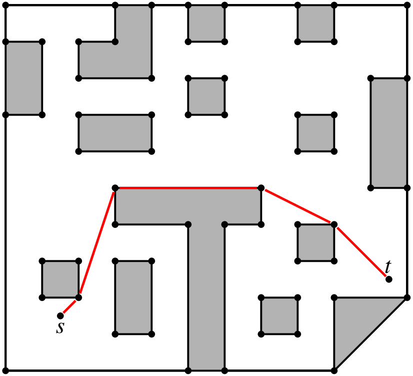
Euclidean Shortest Path

- Pathfinding on the Euclidean plane
- Takes as input start s, target t and a set of polygonal obstacles P
- Find shortest path from s to t without crossing through any obstacles P



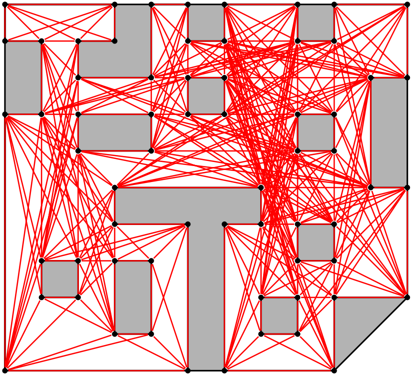
Euclidean Shortest Path

- Pathfinding on the Euclidean plane
- Takes as input start s, target t and a set of polygonal obstacles P
- Find shortest path from s to t without crossing through any obstacles P
- Distance is based off the Euclidean distance $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$



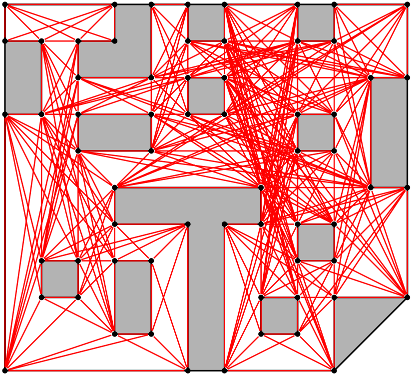
Visibility Graph

- Visibility graphs can solve the Euclidean shortest path (Lozano-Pérez et al. 1979)



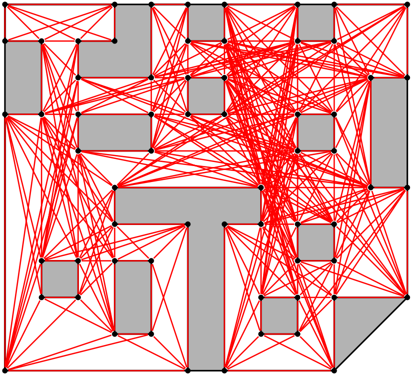
Visibility Graph

- Visibility graphs can solve the Euclidean shortest path (Lozano-Pérez et al. 1979)
- Generates graph $G = (V, E)$ where V is polygon vertices and E is edges that are free of all obstacles



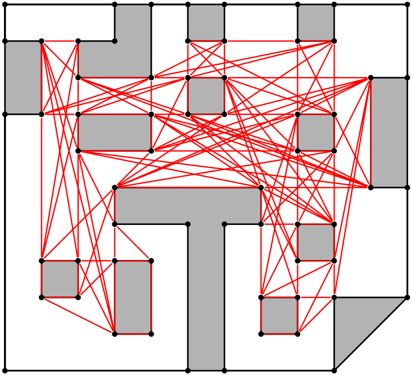
Visibility Graph

- Visibility graphs can solve the Euclidean shortest path (Lozano-Pérez et al. 1979)
- Generates graph $G = (V, E)$ where V is polygon vertices and E is edges that are free of all obstacles
- A full visibility graph uses V^2 edges complexity



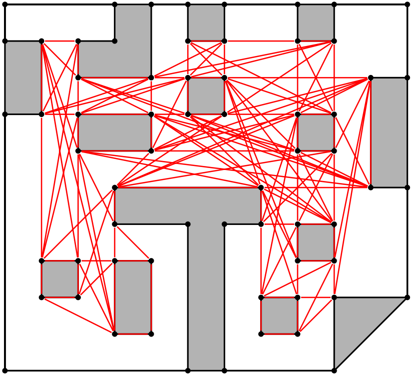
Sparse Visibility Graph

- A variant of the visibility graph (Oh et al., SoCS 2017)



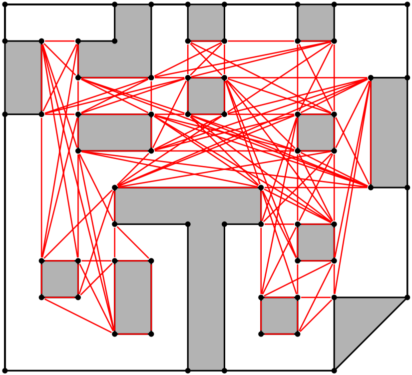
Sparse Visibility Graph

- A variant of the visibility graph (Oh et al., SoCS 2017)
- Keeps only edges that can form a shortest path



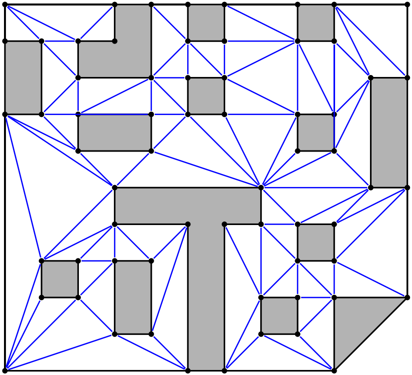
Sparse Visibility Graph

- A variant of the visibility graph (Oh et al., SoCS 2017)
- Keeps only edges that can form a shortest path
- Greatly reduces the density of edges



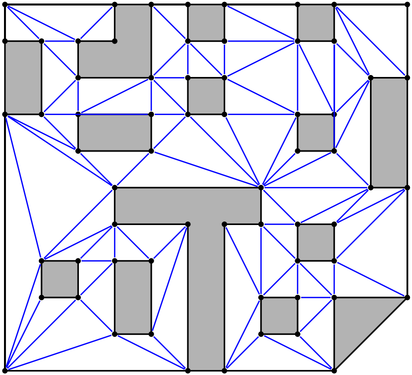
Navigation Mesh

- Inverts the search space from a set of non-traversable obstacles to traversable convex regions



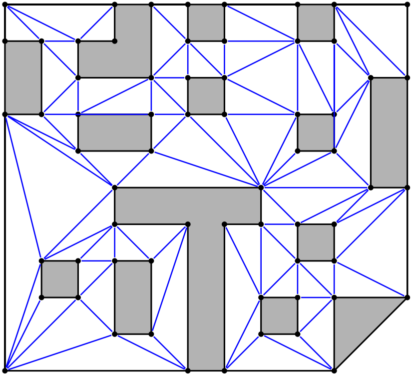
Navigation Mesh

- Inverts the search space from a set of non-traversable obstacles to traversable convex regions
- Has a state-of-the-art solver Polyanya (Cui et al., IJCAI 2017)



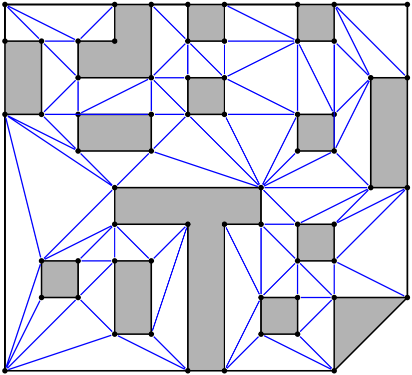
Navigation Mesh

- Inverts the search space from a set of non-traversable obstacles to traversable convex regions
- Has a state-of-the-art solver Polyanya (Cui et al., IJCAI 2017)
- Requires less memory and are easier to construct than visibility graphs



Navigation Mesh

- Inverts the search space from a set of non-traversable obstacles to traversable convex regions
- Has a state-of-the-art solver Polyanya (Cui et al., IJCAI 2017)
- Requires less memory and are easier to construct than visibility graphs
- Like visibility graphs, does not solve the problem directly



Introducing RayScan

- An online method for solving the Euclidean shortest path

Introducing RayScan

- An online method for solving the Euclidean shortest path
- Works as an on-the-fly partial visibility graph on a A* search

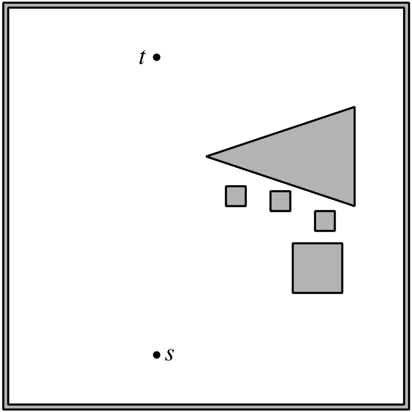
Introducing RayScan

- An online method for solving the Euclidean shortest path
- Works as an on-the-fly partial visibility graph on a A* search
- Generates edges for each expanding node; they can be discarded after being pushed to the priority queue

RayScan Concepts

Shooting Rays Direct the Search

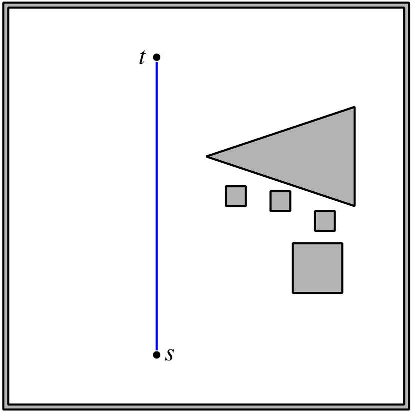
- Expand start node s



RayScan Concepts

Shooting Rays Direct the Search

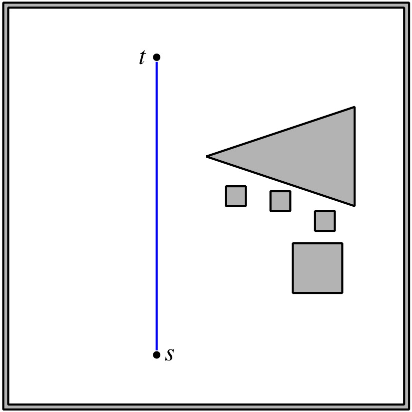
- Expand start node s
- Shoot a ray towards target point t



RayScan Concepts

Shooting Rays Direct the Search

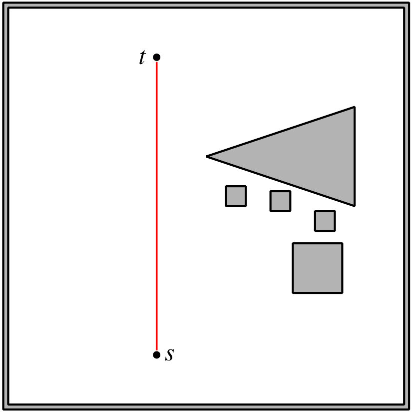
- Expand start node s
- Shoot a ray towards target point t
- If point is visible, it is a successor



RayScan Concepts

Shooting Rays Direct the Search

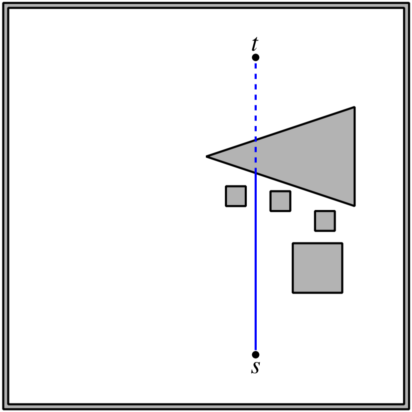
- Expand start node s
- Shoot a ray towards target point t
- If point is visible, it is a successor
- Direct line-of-sight from s to t is the simplest case



RayScan Concepts

Scanning

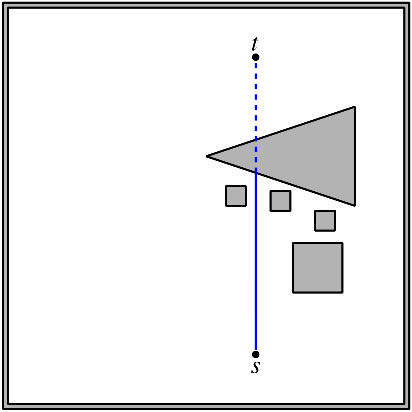
- Target t is not visible from start s



RayScan Concepts

Scanning

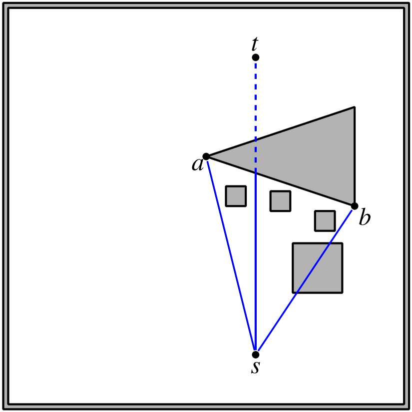
- Target t is not visible from start s
- First obstacle blocking target needs to be navigated around



RayScan Concepts

Scanning

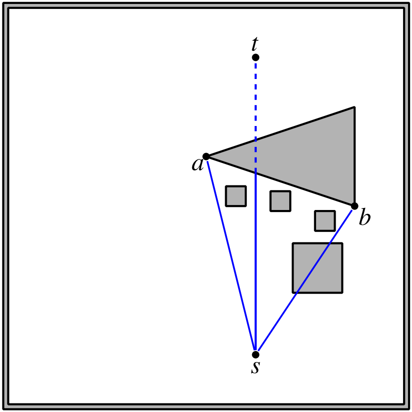
- Target t is not visible from start s
- First obstacle blocking target needs to be navigated around
- Scanning is a core concept to find methods of navigating around obstruction



RayScan Concepts

Scanning

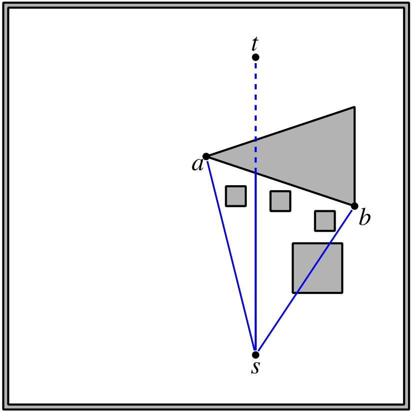
- Target t is not visible from start s
- First obstacle blocking target needs to be navigated around
- Scanning is a core concept to find methods of navigating around obstruction
- Find turning point a and b scanning CCW/CW that we can bend around



RayScan Concepts

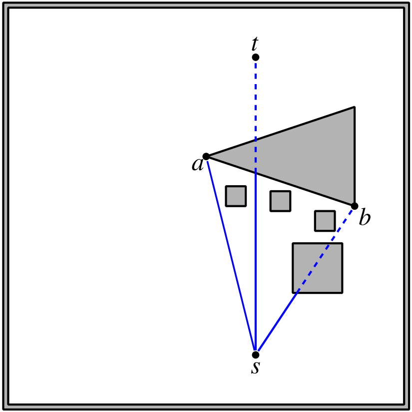
Recursion

- Turning point a is visible from s



RayScan Concepts
Recursion

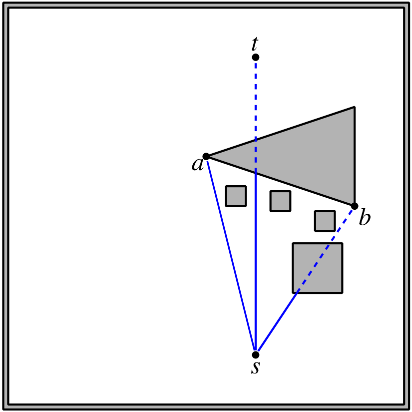
- Turning point *a* is visible from *s*
- Turning point *b* is blocked from *s*



RayScan Concepts

Recursion

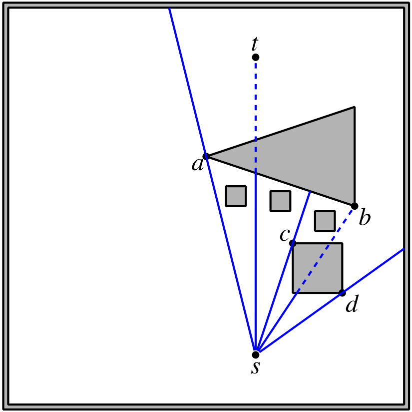
- Turning point a is visible from s
- Turning point b is blocked from s
- The ray shot to b directs the search to find successors that may lead to it



RayScan Concepts

Recursion

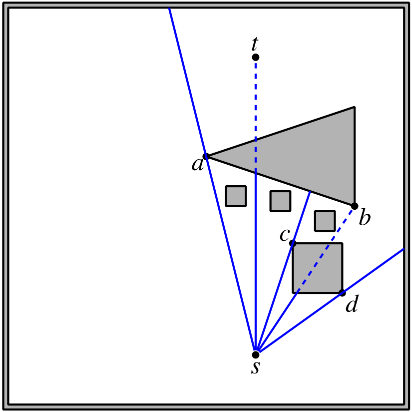
- Turning point *a* is visible from *s*
- Turning point *b* is blocked from *s*
- The ray shot to *b* directs the search to find successors that may lead to it
- Recurse the scan CCW/CW from obstruction intersection



RayScan Concepts

Recursion

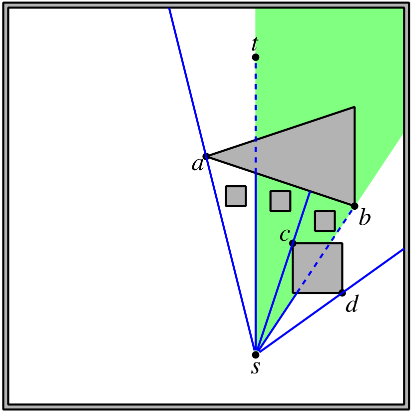
- Turning point *a* is visible from *s*
- Turning point *b* is blocked from *s*
- The ray shot to *b* directs the search to find successors that may lead to it
- Recurse the scan CCW/CW from obstruction intersection
- Successors *c* and *d* is found



RayScan Concepts

Angled Sectors

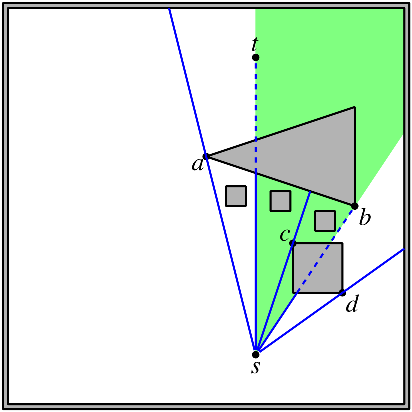
- Scan is restricted to a region called the angled sector



RayScan Concepts

Angled Sectors

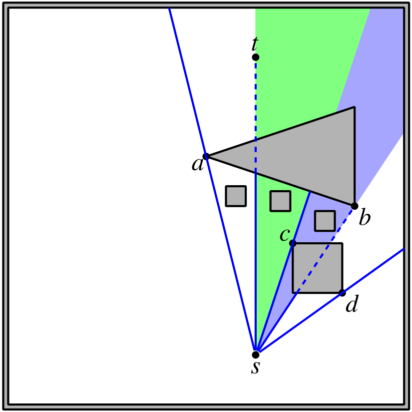
- Scan is restricted to a region called the angled sector
- Once the scan leaves the angled sector, it ends



RayScan Concepts

Angled Sectors

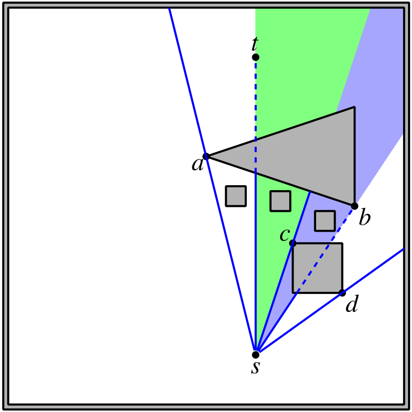
- Scan is restricted to a region called the angled sector
- Once the scan leaves the angled sector, it ends
- When recursing a scan CW/CCW, angled sector is also split



RayScan Concepts

Angled Sectors

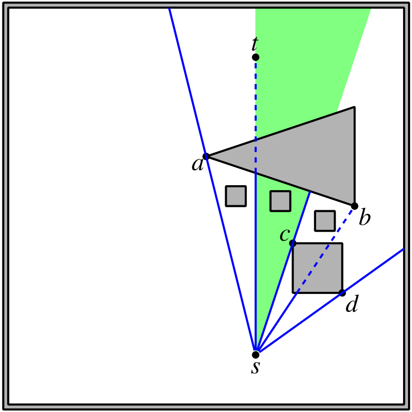
- Scan is restricted to a region called the angled sector
- Once the scan leaves the angled sector, it ends
- When recursing a scan CW/CCW, angled sector is also split
- After finding c, the angled sector gets splits



RayScan Concepts

Angled Sectors

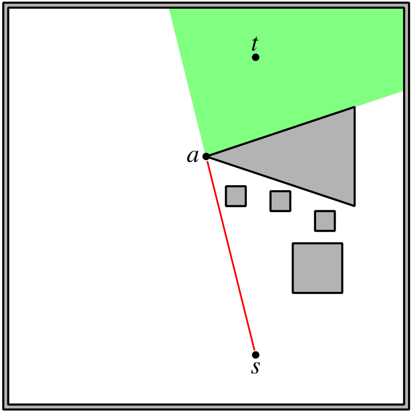
- Scan is restricted to a region called the angled sector
- Once the scan leaves the angled sector, it ends
- When recursing a scan CW/CCW, angled sector is also split
- After finding c, the angled sector gets splits
- The CCW scan uses the green split



RayScan Concepts

Next Expansion

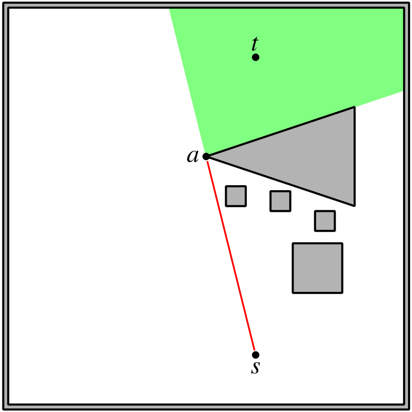
- A* search expands the smallest f-value a



RayScan Concepts

Next Expansion

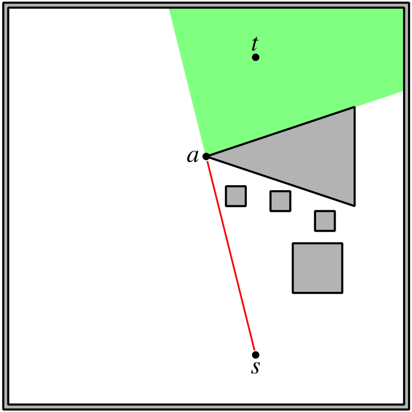
- A* search expands the smallest f-value a
- Each expanding node has a special angled sector known as the projection field



RayScan Concepts

Next Expansion

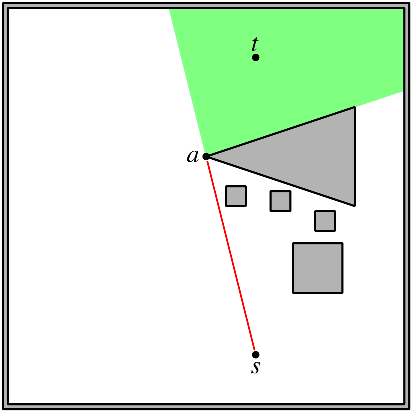
- A* search expands the smallest f-value a
- Each expanding node has a special angled sector known as the projection field
- All successor must fall within this field to ensure taut (bends around corner)



RayScan Concepts

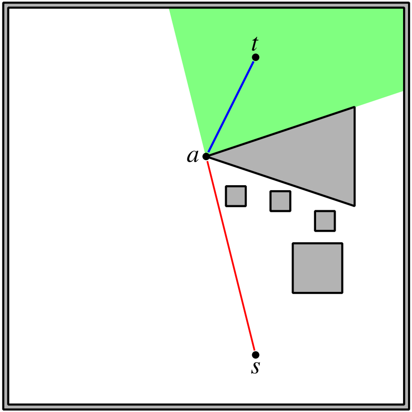
End Search

- Expanding a, target t is within the projection field



RayScan Concepts
End Search

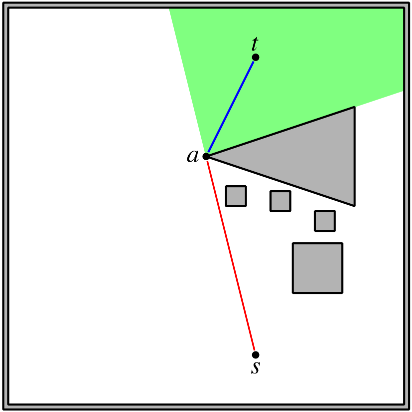
- Expanding a, target t is within the projection field
- Shoot ray to t



RayScan Concepts

End Search

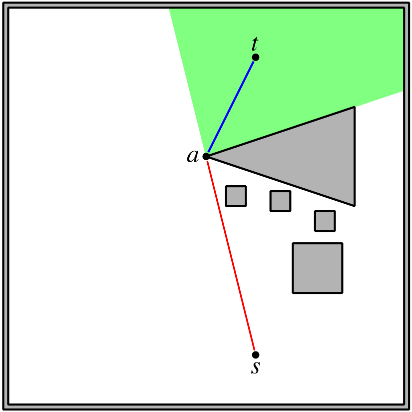
- Expanding a, target t is within the projection field
- Shoot ray to t
- Target is visible, successor is added



RayScan Concepts

End Search

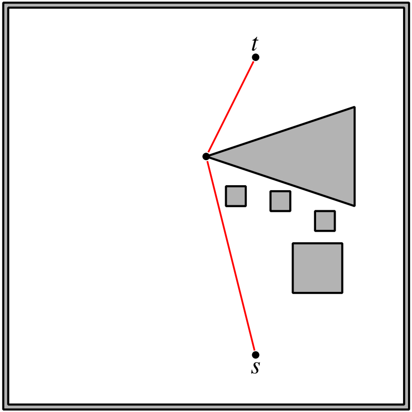
- Expanding a, target t is within the projection field
- Shoot ray to t
- Target is visible, successor is added
- Expansion of node ends since target is found



RayScan Concepts

End Search

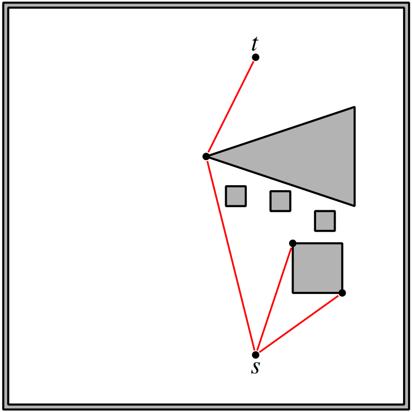
- Expanding a, target t is within the projection field
- Shoot ray to t
- Target is visible, successor is added
- Expansion of node ends since target is found
- If using a Euclidean heuristic value, search ends here



RayScan Concepts

Skipped Successors

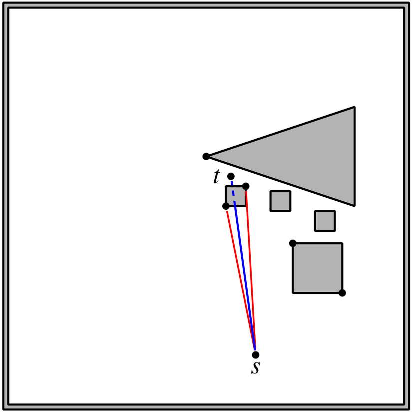
- Multiple squares are not considered



RaySan Concepts

Skipped Successors

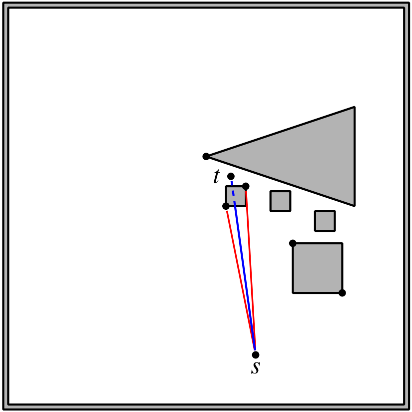
- Multiple squares are not considered
- Moving target t behind a square makes it relevant



RayScan Concepts

Skipped Successors

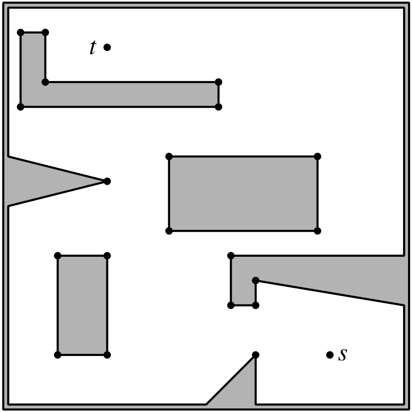
- Multiple squares are not considered
- Moving target t behind a square makes it relevant
- Since rays direct the search, this reduces the successors and number of rays shot



Comparisions to Visibility Graphs

RayScan

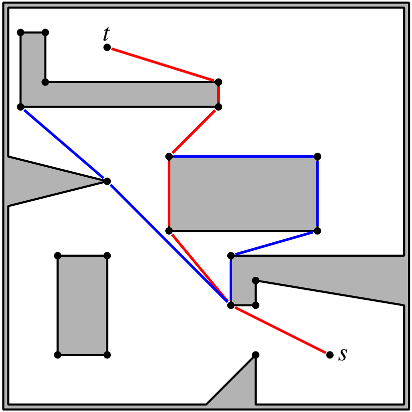
- Applying RayScan to a larger scenario



Comparisions to Visibility Graphs

RayScan

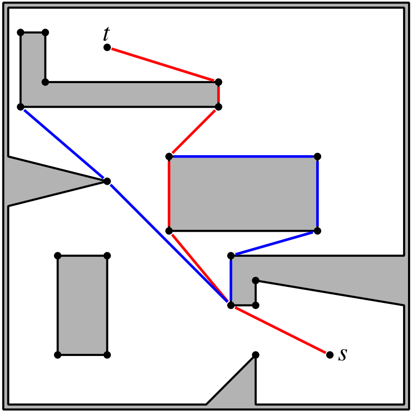
- Applying RayScan to a larger scenario
- Diagram shows RayScan search successors



Comparisions to Visibility Graphs

RayScan

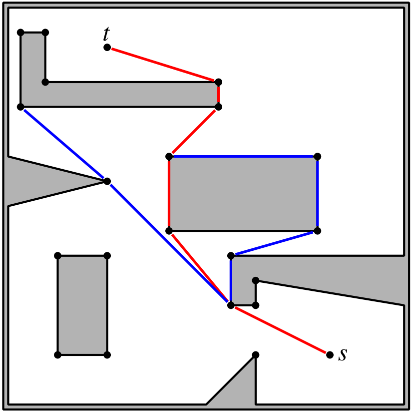
- Applying RayScan to a larger scenario
- Diagram shows RayScan search successors
- Red line is the shortest path



Comparisions to Visibility Graphs

RayScan

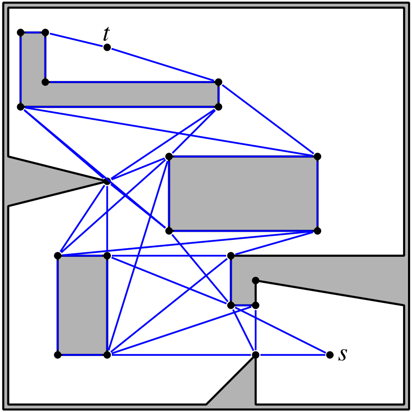
- Applying RayScan to a larger scenario
- Diagram shows RayScan search successors
- Red line is the shortest path
- Blue lines are other successors found during the search



Comparisions to Visibility Graphs

Sparse Visibility Graph

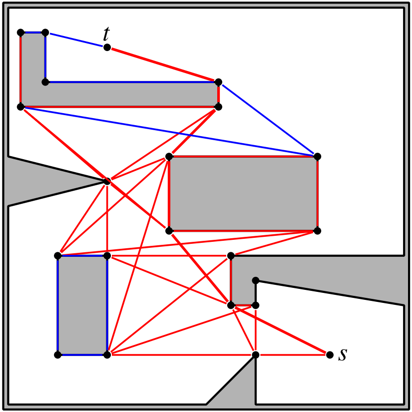
- Blue lines show the edges of the sparse visibility graph



Comparisions to Visibility Graphs

Sparse Visibility Graph

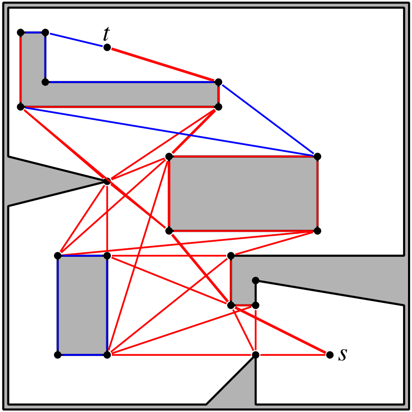
- Blue lines show the edges of the sparse visibility graph
- Red lines show the edges expanded during an A* search



Comparisions to Visibility Graphs

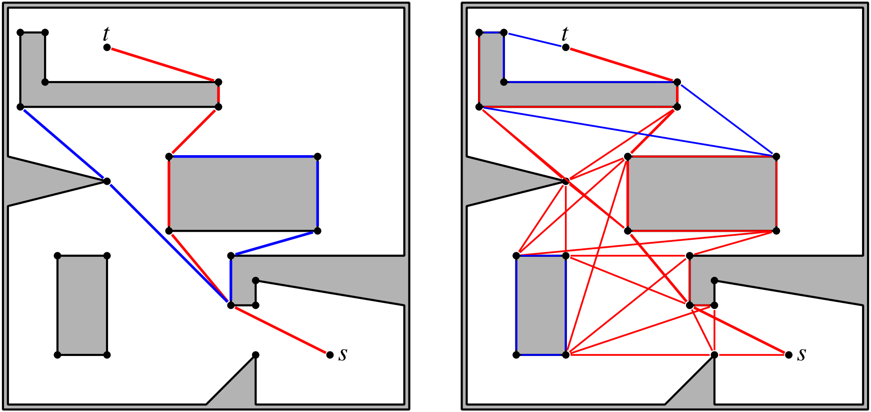
Sparse Visibility Graph

- Blue lines show the edges of the sparse visibility graph
- Red lines show the edges expanded during an A* search
- RayScan generates edges found on the sparse visibility graphs on-the-fly



Comparisions to Visibility Graphs

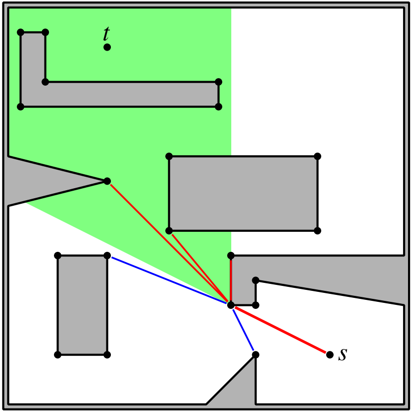
Sparse Visibility Graph Compare



Comparisions to Visibility Graphs

Visibility Graph with Projection Field

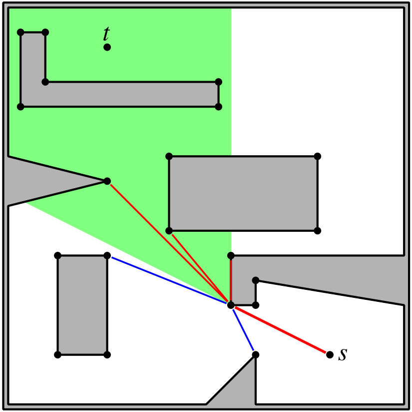
- The projection field concept can be applied to visibility graph on A*



Comparisions to Visibility Graphs

Visibility Graph with Projection Field

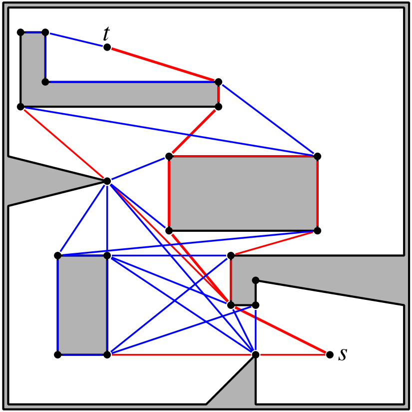
- The projection field concept can be applied to visibility graph on A*
- Add only the successors of A* within the projection field



Comparisions to Visibility Graphs

Visibility Graph with Projection Field

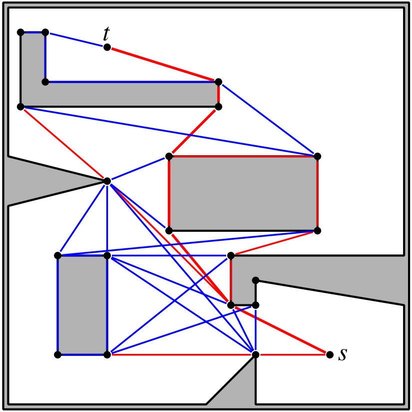
- The projection field concept can be applied to visibility graph on A*
- Add only the successors of A* within the projection field
- The search proceeds similarly to RayScan with this change



Comparisions to Visibility Graphs

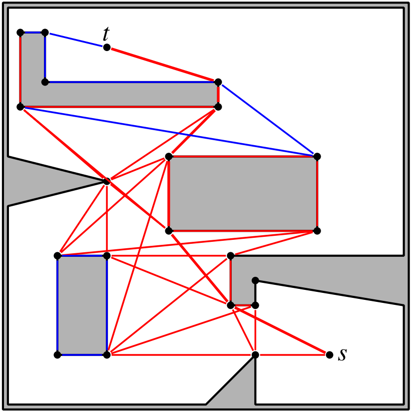
Visibility Graph with Projection Field

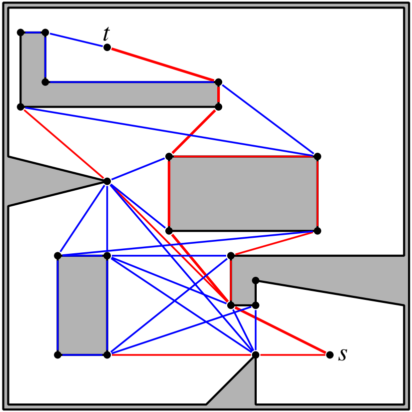
- The projection field concept can be applied to visibility graph on A*
- Add only the successors of A* within the projection field
- The search proceeds similarly to RayScan with this change
- This is referred as taut A* by Oh et al. (2017)



Comparisions to Visibility Graphs

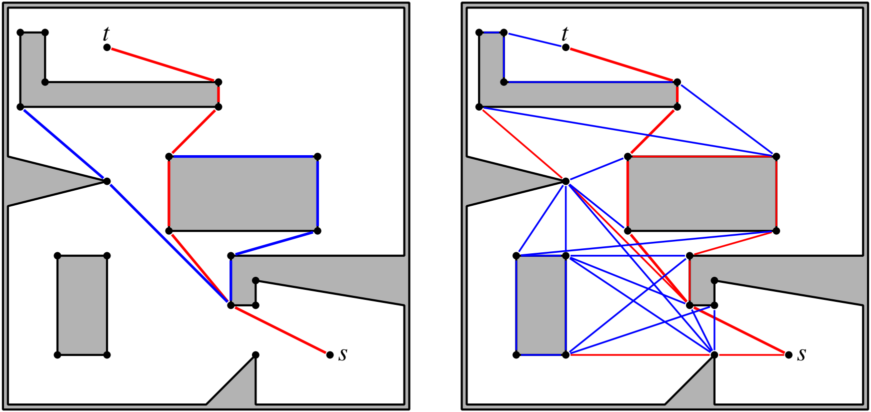
Visibility Graph with Projection Field Compare





Comparisions to Visibility Graphs

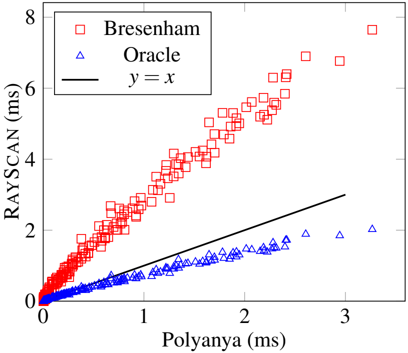
Visibility Graph with Projection Field Compare



Results Presented in Paper

- Testing against Moving AI Lab pathfinding benchmarks (Sturtevant 2012)

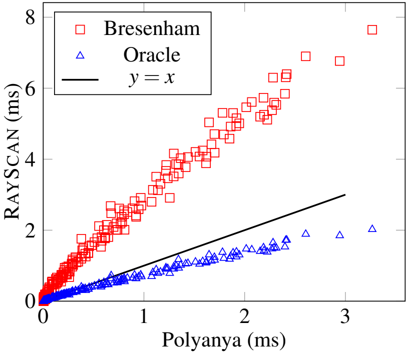
Starcraft Aurora Map



Results Presented in Paper

- Testing against Moving AI Lab pathfinding benchmarks (Sturtevant 2012)
- Comparing RayScan against Polyanya

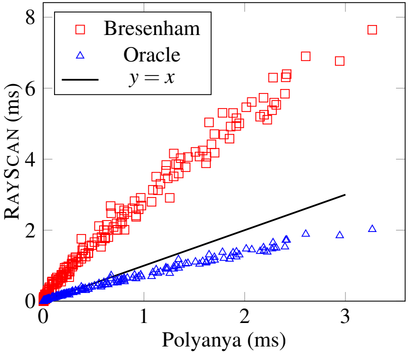
Starcraft Aurora Map



Results Presented in Paper

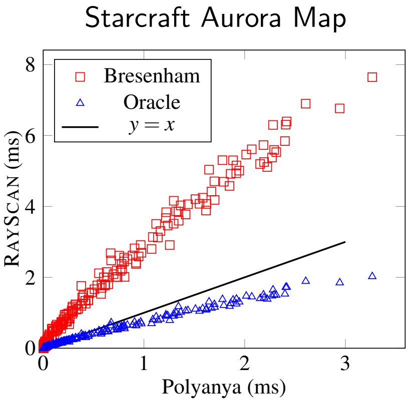
- Testing against Moving AI Lab pathfinding benchmarks (Sturtevant 2012)
- Comparing RayScan against Polyanya
- Bresenham is RayScan using a Bresenham line for the ray shooting (Bresenham 1965)

Starcraft Aurora Map



Results Presented in Paper

- Testing against Moving AI Lab pathfinding benchmarks (Sturtevant 2012)
- Comparing RayScan against Polyanya
- Bresenham is RayScan using a Bresenham line for the ray shooting (Bresenham 1965)
- Oracle uses a free ray shooter



Recap

- RayScan works directly on the environment

Recap

- RayScan works directly on the environment
- Well suited to single instance or dynamically changing maps

Recap

- RayScan works directly on the environment
- Well suited to single instance or dynamically changing maps
- Performance dependent on ray shooter (i.e. more than half runtime is shooting rays)

References

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, *22*(10), 560–570.

Oh, S., & Leong, H. W. (2017). Edge n-level sparse visibility graphs: Fast optimal any-angle pathfinding using hierarchical taut paths. *Proceedings of the Tenth International Symposium on Combinatorial Search (SoCS 2017)*.

Cui, M. L., Harabor, D., & Grastien, A. (2017). Compromise-free pathfinding on a navigation mesh. *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 496–502.

Sturtevant, N. (2012). Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games*, *4*(2), 144–148. http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems journal*, *4*(1), 25–30.