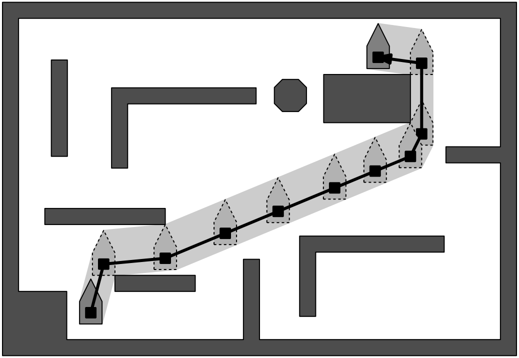
Euclidean Shortest Path Planning (Chapter 15)

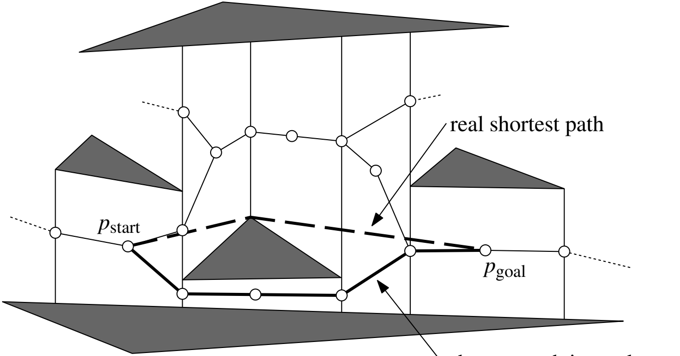
Elisha Sacks

Shortest Path Planning



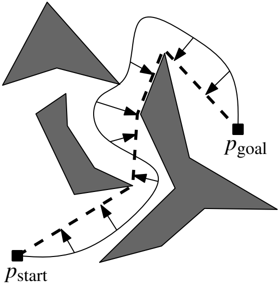
- A polygonal robot translates amidst polygonal obstacles.
- Task: compute a shortest path between start and goal points.
- Work in configuration space.

Roadmap



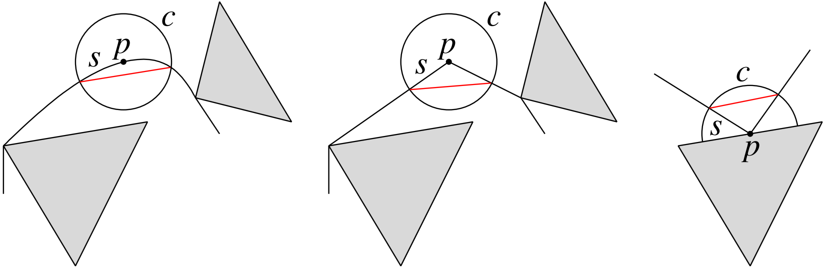
The shortest path is rarely in the roadmap even if p_{start} and p_{goal} are roadmap vertices.

Shortest Path Intuition



- Connect p_{start} and p_{goal} with a string.
- Tighten the string as much as possible.
- This path is polygonal.
- Do this for every way of navigating the obstacles.
- One way yields the shortest path.

Polygonal Path

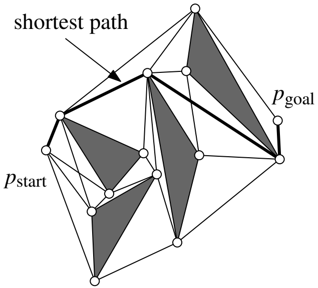


1 2 3

Claim The shortest path is polygonal.
Proof If the path is curved at a point p in free space, it intersects a circle c centered at p in a curved segment s (1). The path can be shortened by replacing s by a line segment.

Claim The inner vertices are obstacle vertices.
Proof A vertex cannot be in free space as above (2). It cannot be on an obstacle edge by a similar argument using a semicircle (3).

Visibility Graph



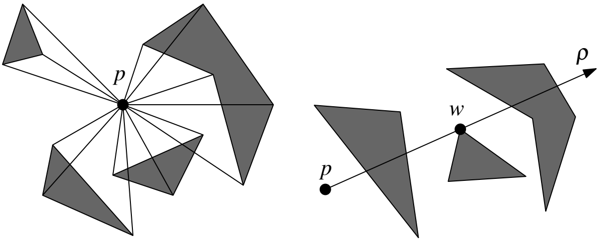
- The vertices are the obstacle vertices, p_{start}, and p_{goal}.
- If vertices v and w are mutually visible, vw is an edge.
- In particular, the obstacle edges are in the visibility graph.
- Shortest path algorithm: construct the visibility graph and invoke Dijkstra's algorithm.

Visibility Graph Construction

Algorithm $\text{VisibilityGraph}(S)$
Input: A set S of disjoint polygonal obstacles and vertices.
Output: The visibility graph $G = (V, E)$ of S.

1. Set V to the vertices of S; set $E = \emptyset$.
2. for all vertices $p \in V$
   2.1 Set $W \leftarrow \text{VisibleVertices}(p, S)$
   2.2 For every vertex $q \in W$, add the arc (p, q) to E.
3. return G.

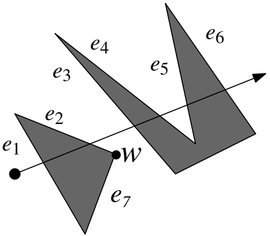
Visibility Test

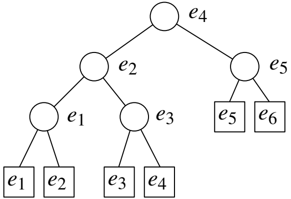


If p and w bound the same obstacle, w is visible from p if pw is disjoint from the interior of the obstacle.

Otherwise, w is visible from p if it is closer to p than the first edge that intersects the ray ρ from p to w.

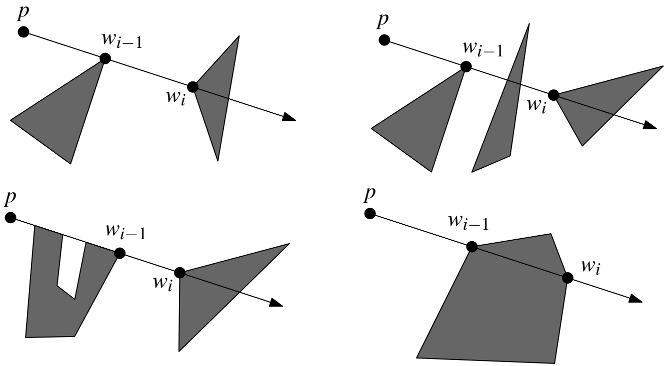
Strategy





- Process each vertex w in clockwise order.
- Maintain a list of edges that intersect ρ in distance order.
- Check if w is visible using the first edge in the list.
- Remove the edges wu with u counterclockwise from w.
- Insert the edges wv with v clockwise from w.
- Example: remove e_2 and add e_7.

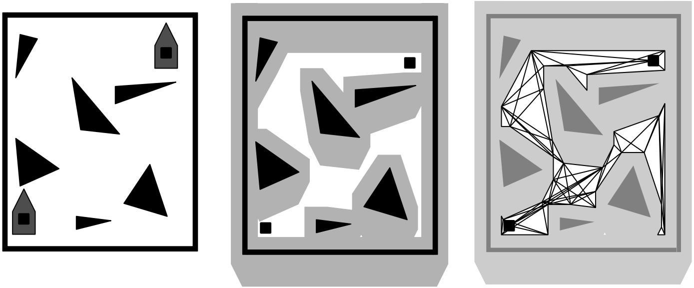
Degenerate Cases



Three collinear vertices create degenerate cases.

Path Planning Summary

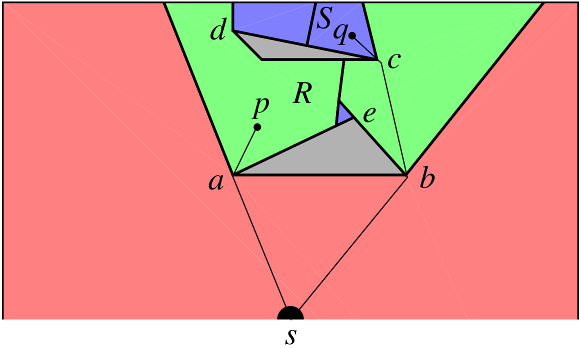
work space | configuration space | visibility graph



Computational Complexity

- The time complexity of VisibleVertices is $O(n \log n)$.
- The time complexity of VisibilityGraph is $O(n^2 \log n)$.
- This dominates the time complexity of Djikstra's algorithm.
- The visibility graph bound is close to optimal.
- The optimal shortest path algorithm is $O(n \log n)$.
- We will look at the strategy; the details are complicated.

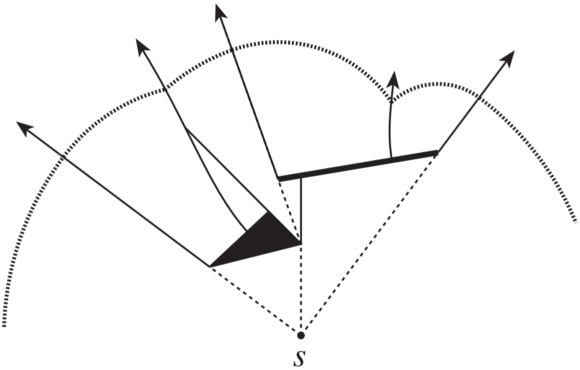
Shortest Path Map Algorithm



A shortest path map (SPM) for a point s and disjoint polygonal obstacles O is a planar subdivision where all the points in a face have the same sequence of O vertices on their shortest paths to s.

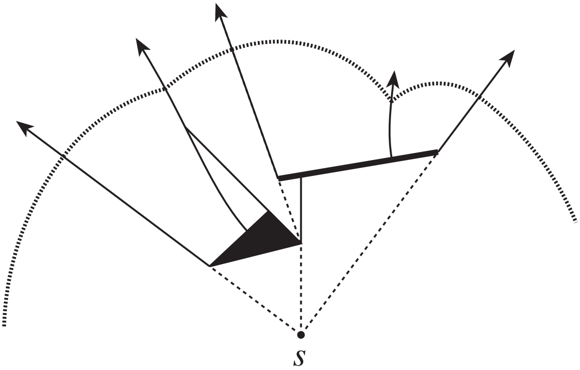
The SPM is constructed by propagating a unit-velocity wavefront from v through the free space.

Wavelets



- The wavefront consists of circular arcs called wavelets.
- The initial wavelet is centered at s.
- When the wavefront hits an obstacle vertex o, a wavelet centered at o is born.
- A wavelet dies when it hits an O edge or when it collapses.

SPM Edges



- The endpoints of the incident wavelets trace the SPM edges.
- SPM edges from mutually visible O vertices are straight.
- The other SPM edges are hyperbolic.
- Three SPM edges meet at an SPM vertex.

SPM Algorithm

- There are $O(n)$ wavefront events for n vertices in O.
- Naive event handling is $O(n)$.
- Hershberger and Suri achieve $O(\log n)$ with two ideas.
- They decompose the plane into $O(n)$ simple cells and propagate the wavefront between cells.
- They propagate an approximate wavefront that accurately detects the wavelet collisions then compute the exact collision points with a Voronoi technique.

What about 3D?

- Internal vertices of the shortest path can be on obstacle edges.
- Path planning is NP-hard.
- There is an exponential time algorithm.
- There are polynomial time approximate algorithms.
- Shortest path planning with rotation is even harder.