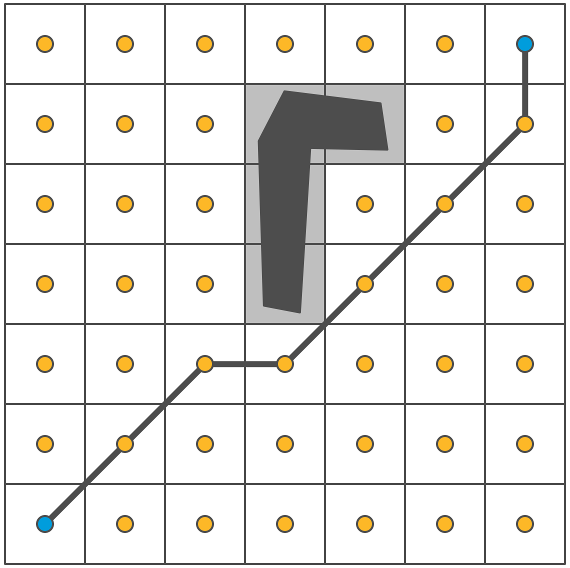
Fall 2021 | Lecture 4

Motion Planning

Ariel Procaccia | Harvard University

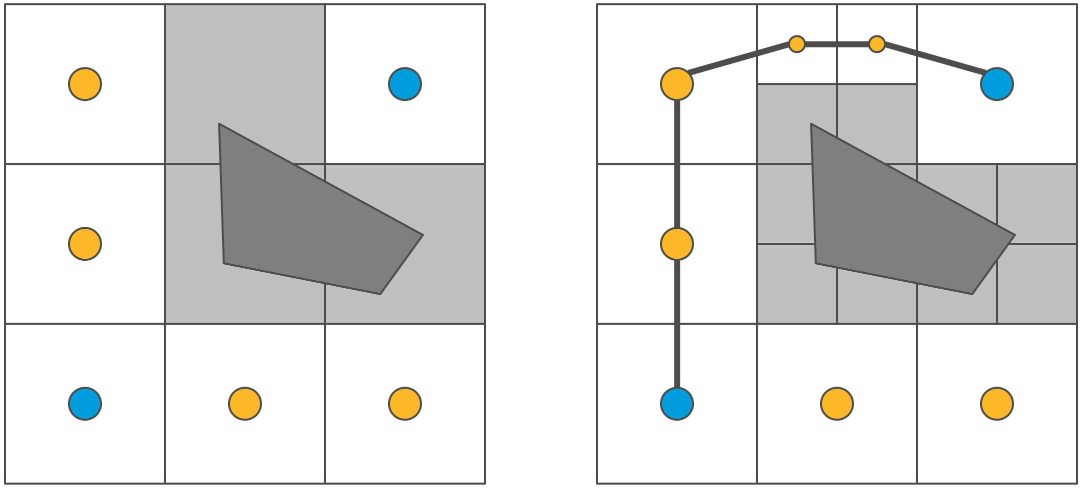
MOTION PLANNING

- Navigating between two points while avoiding obstacles
- A first approach: define a discrete grid
- Mark cells that intersect obstacles as blocked
- Find path through centers of remaining cells



CELL DECOMPOSITION

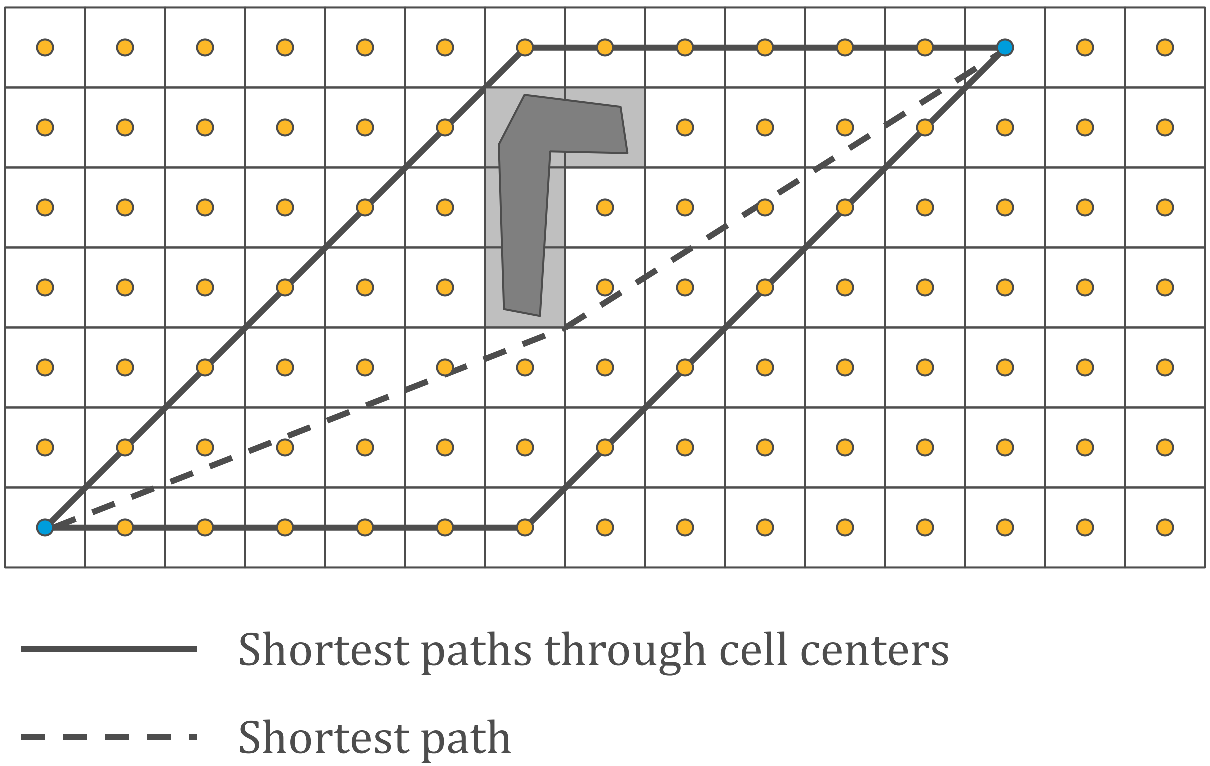
- Distinguish between
  - Cells that are contained in obstacles
  - Cells that intersect obstacles
- If no path found, subdivide the mixed cells



IS IT COMPLETE NOW?

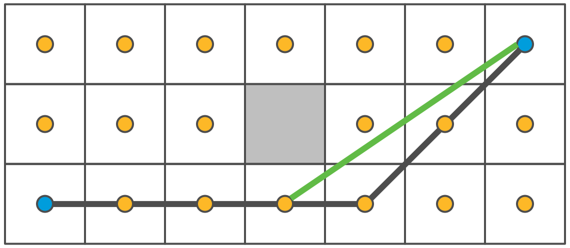
- An algorithm is resolution complete when:
  a. If a path exists, it finds it in finite time
  b. If a path does not exist, it returns in finite time
- Assume that there's a finite number of obstacles, each of which is a closed set
- Poll 1: Cell decomposition satisfies:
  1. a but not b ✓
  2. b but not a
  3. Both a and b
  4. Neither a nor b

CELL DECOMPOSITION

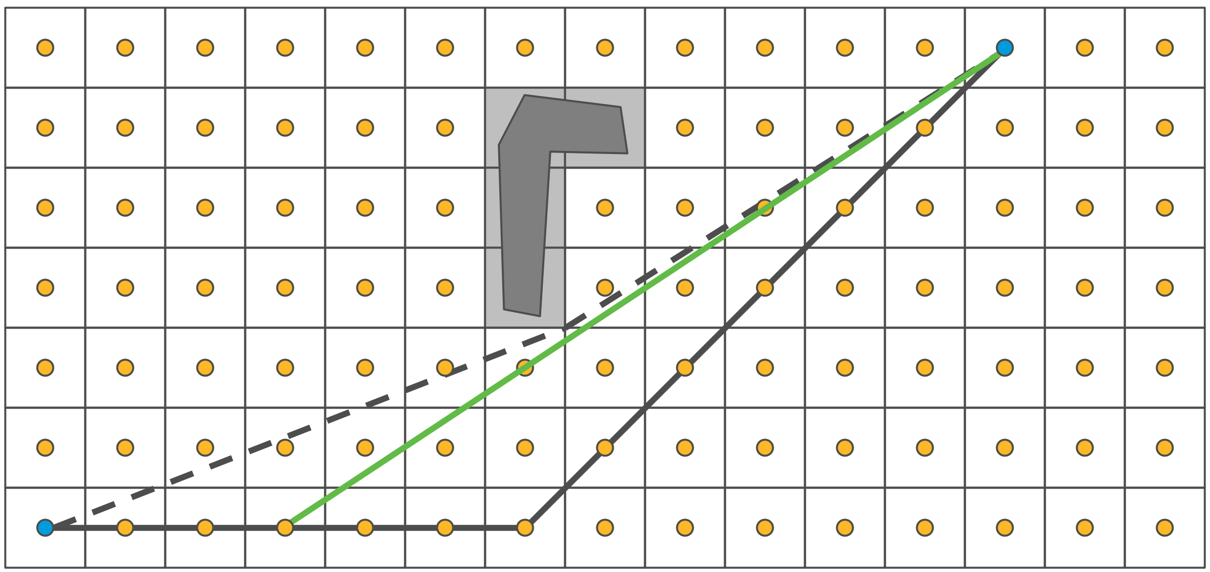


SOLUTION 1: A* SMOOTHING

- Allows connection to farther states than neighbors on the grid
- Key observation:
  - If $x_1, \dots, x_n$ is valid path
  - And x_k is visible from x_j
  - Then $x_1, \dots, x_j, x_k, \dots, x_n$ is a valid path



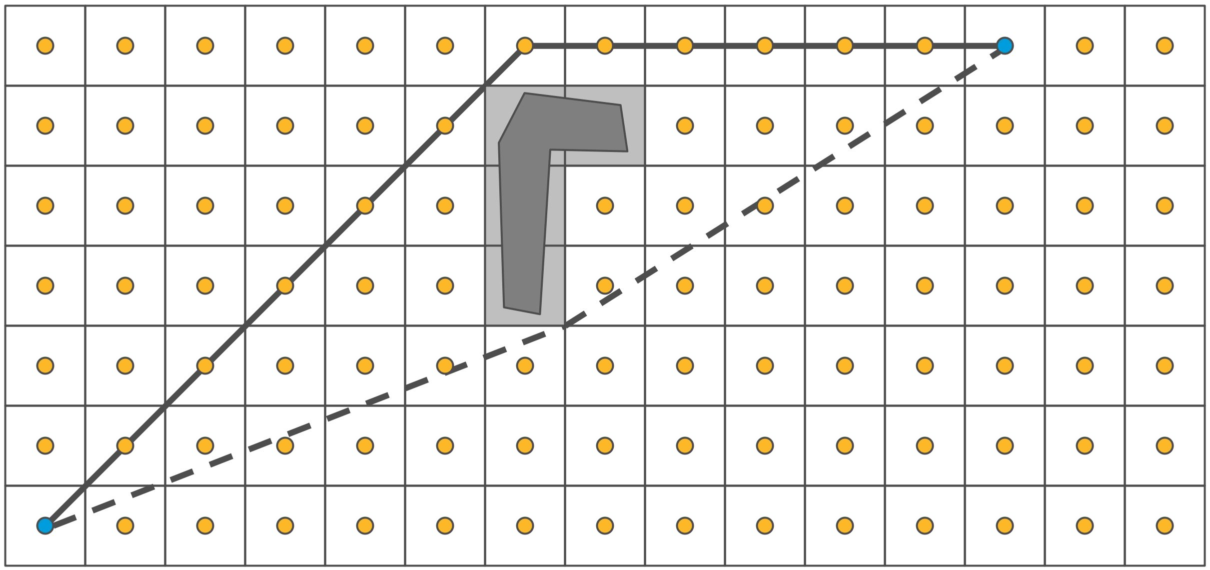
SMOOTHING WORKS!



Shortest paths through cell centers

Shortest path

SMOOTHING DOESN'T WORK!



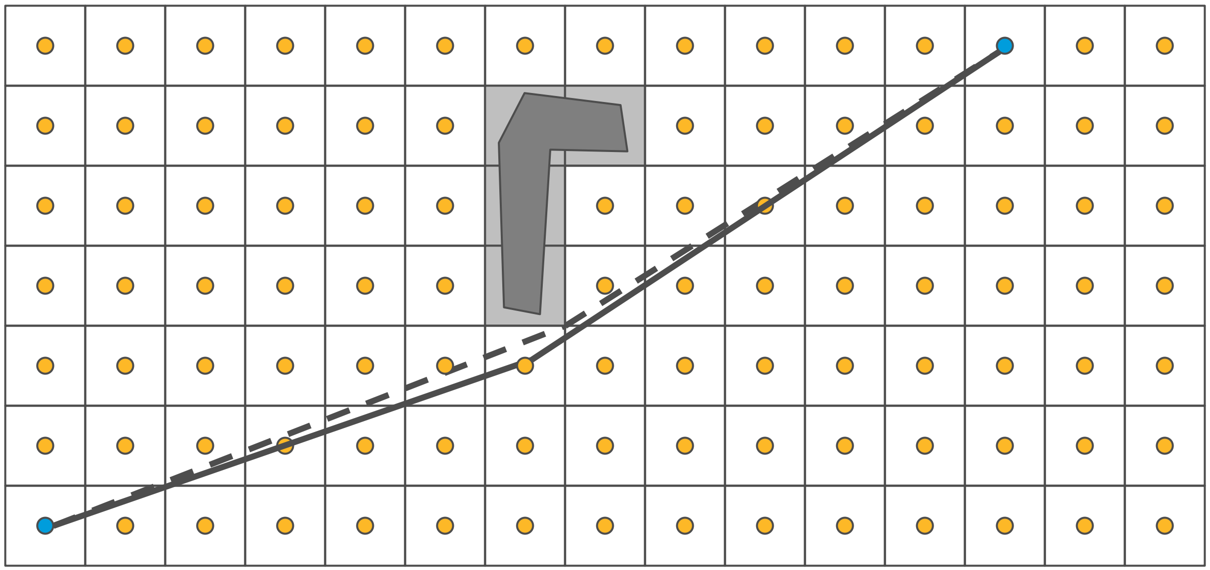
Shortest paths through cell centers

Shortest path

SOLUTION 2: THETA*

- Allow parents that are non-neighbors in the grid to be used during search
- Standard A*
  - $g(y) \ = \ g(x) + c(x, y)$
  - Insert y with estimate
    $$f(y) = g(x) + c(x, y) + h(y)$$
- Theta*
  - If parent(x) is visible from y, insert y with estimate
    $f(y) = g(\text{parent}(x)) + c(\text{parent}(x), y) + h(y)$

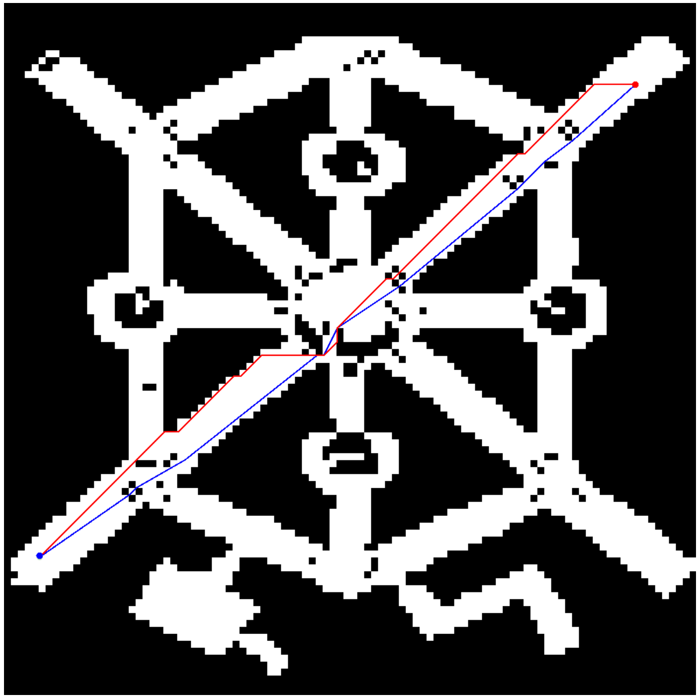
THETA* WORKS!



Theta* path, I think ☺

Shortest path

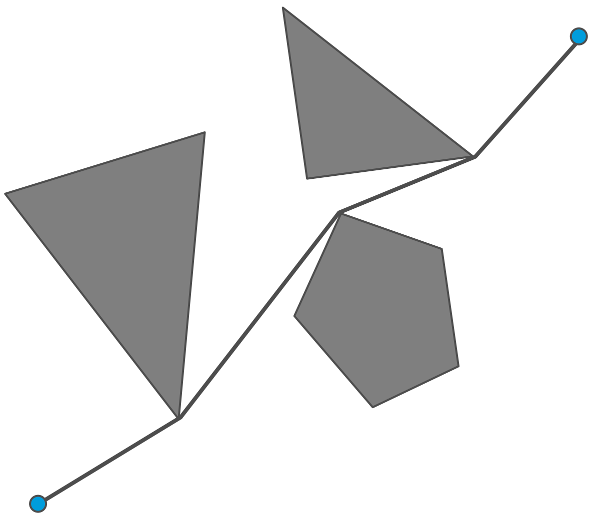
THETA* WORKS!



[Nash, AIGameDev 2010]

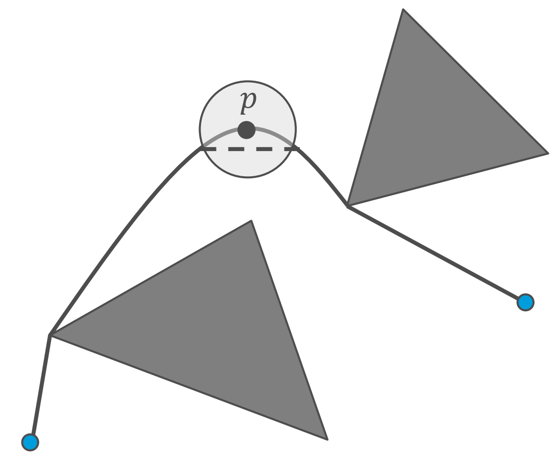
THE OPTIMAL PATH

- **Polygonal path:** sequence of connected straight lines
- **Inner vertex of polygonal path:** vertex that is not beginning or end
- **Theorem:** assuming open polygonal obstacles, shortest path is a polygonal path whose inner vertices are vertices of (the closure of) obstacles



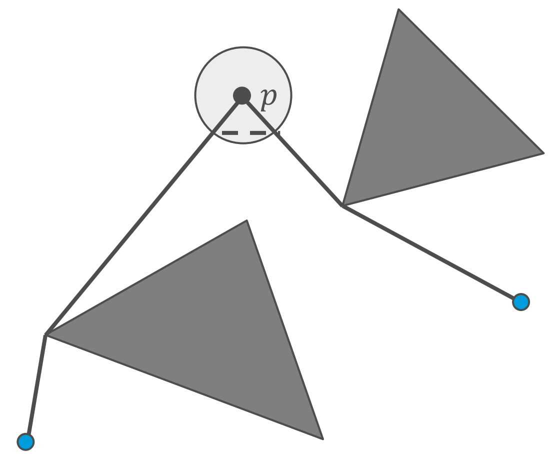
PROOF OF THEOREM

- Suppose for contradiction that shortest path is not polygonal
- Obstacles are polygonal ⇒ $\exists$point p in interior of free space such that "path through p is curved"
- $\exists$disc of free space around p
- Path through disc can be shortened by connecting points of entry and exit

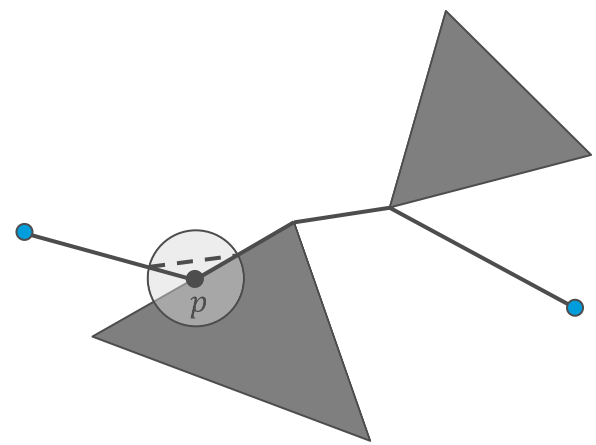


PROOF OF THEOREM

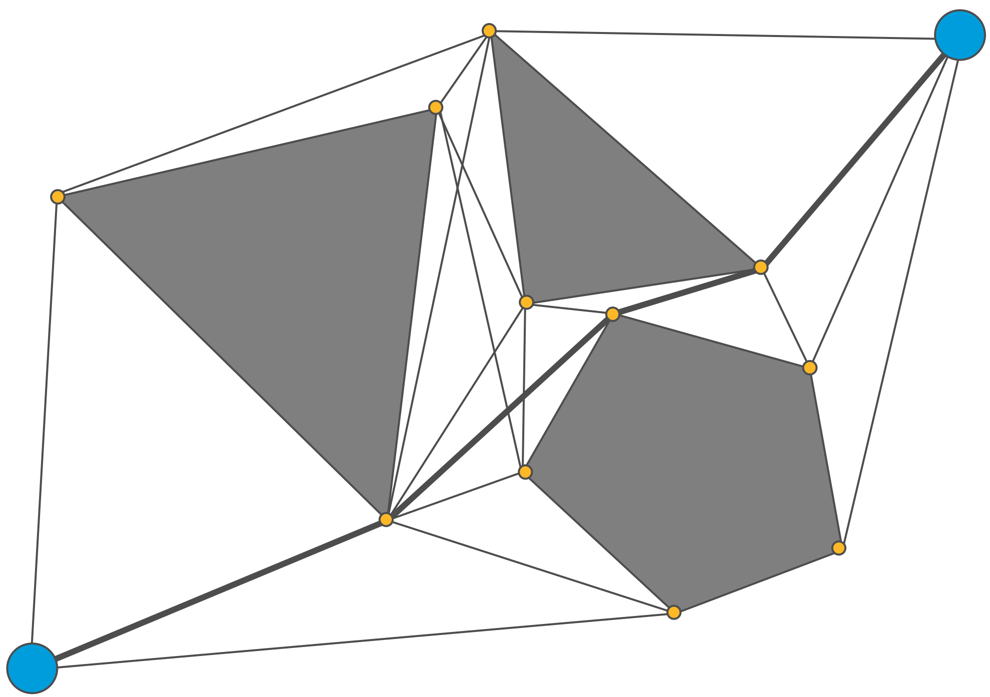
- Path is polygonal!
- Vertex cannot lie in interior of free space, otherwise we can do the same trick



- Vertex cannot lie on the interior of an edge, otherwise we can do the same trick ■



VISIBILITY GRAPH



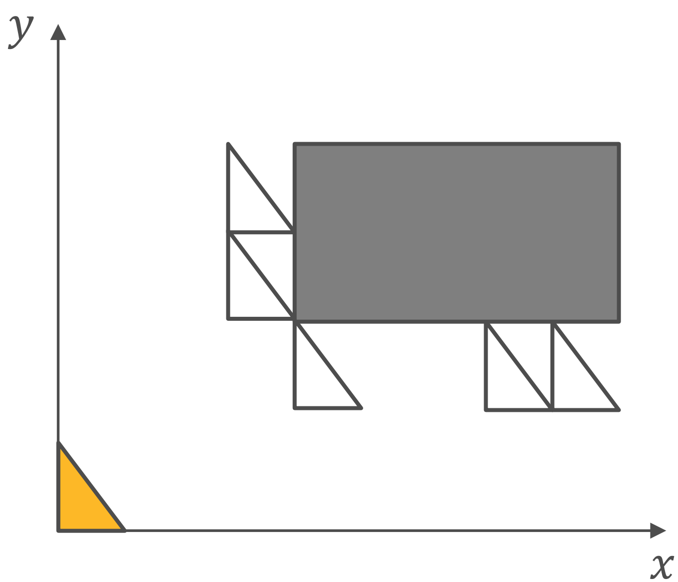
Vertices = vertices of polygons and s, t

Edges = all (x, y) such that y is visible from x

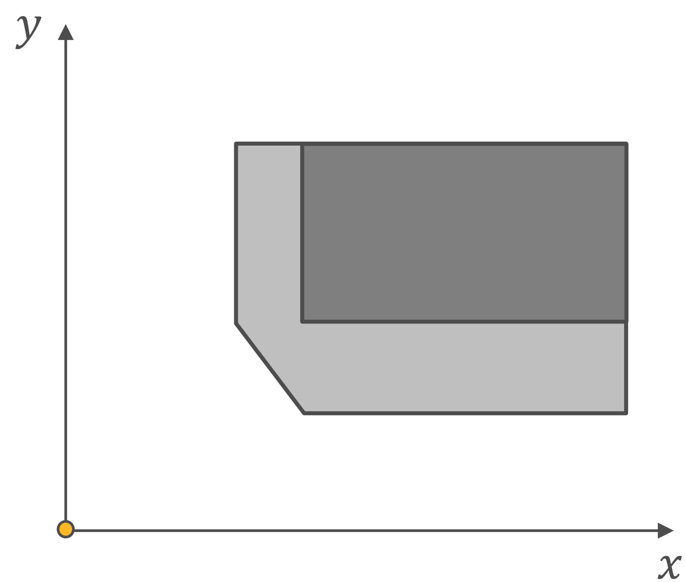
VISIBILITY GRAPH

- Poll 2: Let n be the total number of vertices of all polygons. How many edges will the optimal path in the visibility graph traverse in the worst case?
  1. $\Theta(\sqrt{n})$
  2. $\Theta(n)$ ✓
  3. $\Theta(n^2)$
  4. $\Theta(n^3)$

CONFIGURATION SPACE

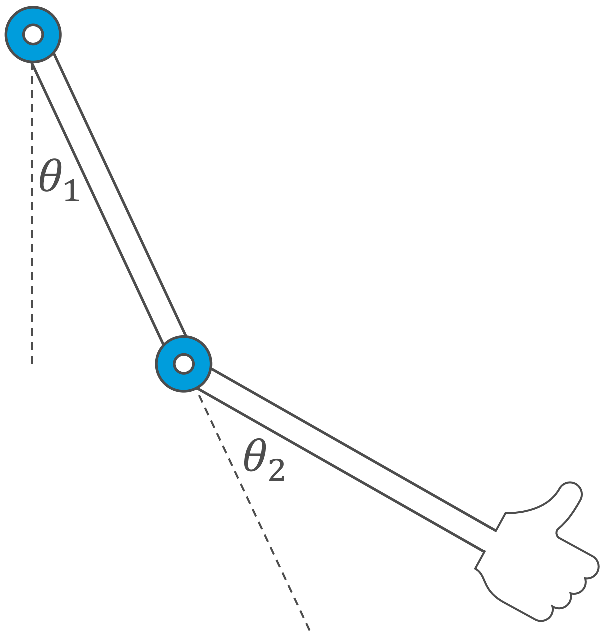


Physical space

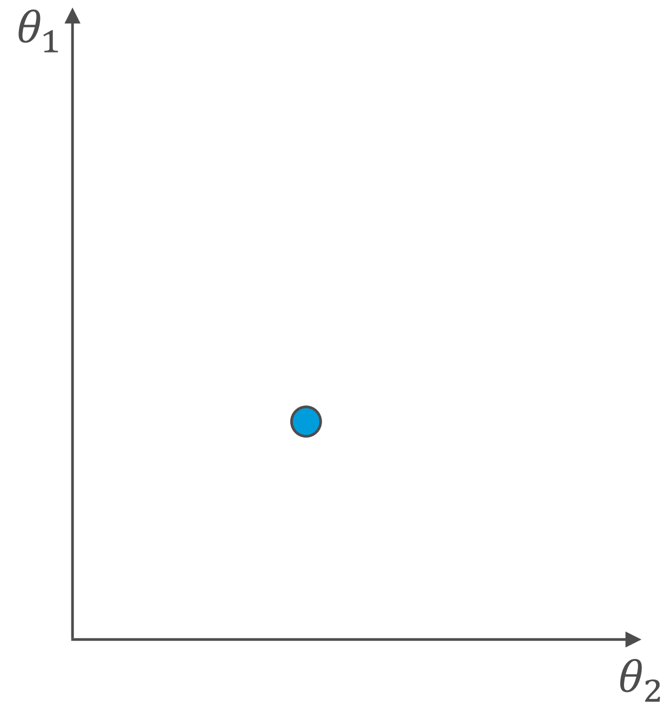


Configuration space

CONFIGURATION SPACE



Physical space



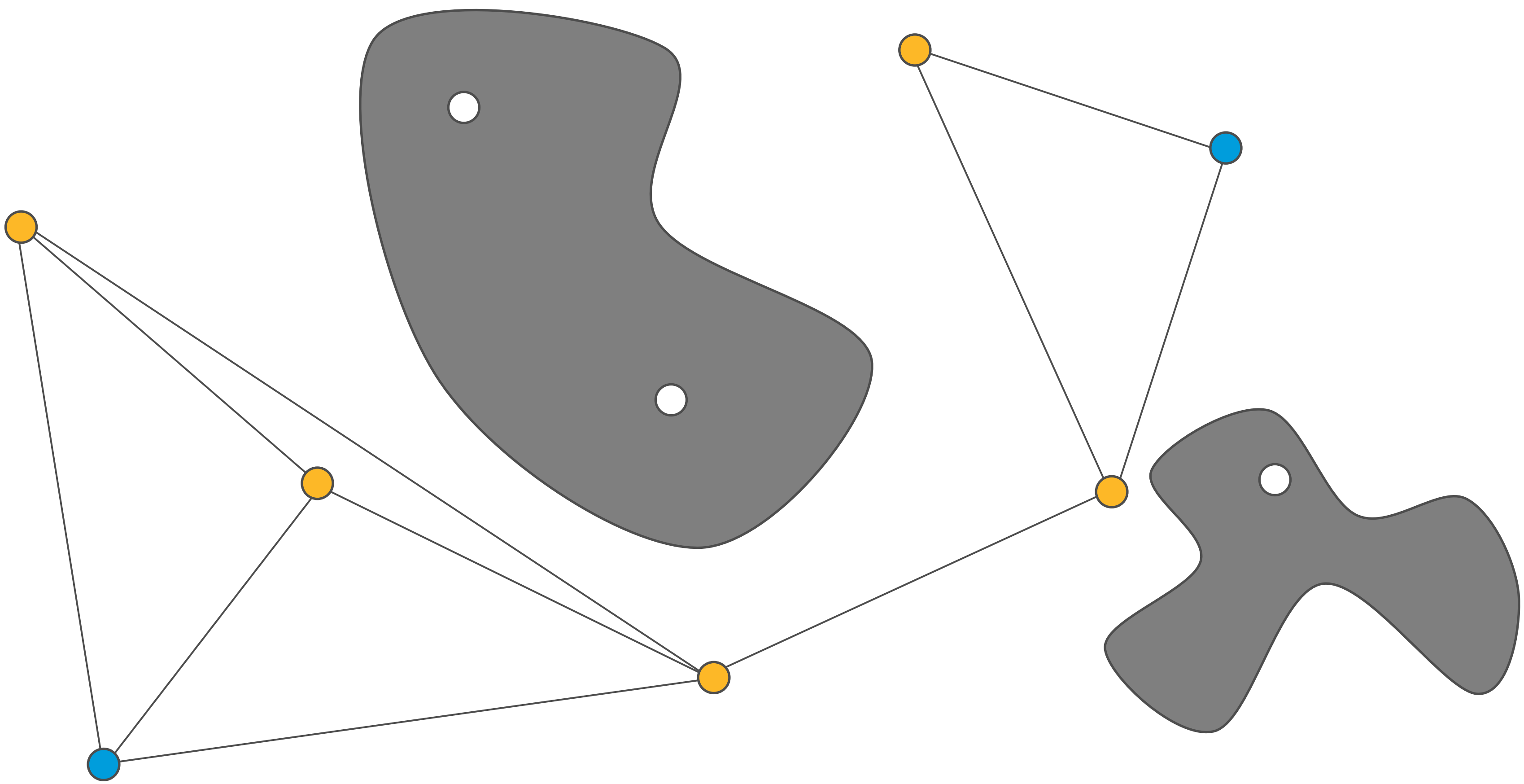
Configuration space

The configuration space can be high dimensional!

PROBABILISTIC ROADMAP

- Find M milestones by randomly sampling points in the configuration space and discarding the ones that are blocked
- Form edges by checking for collision-free lines between milestones
- Edges are checked between each milestone and its k nearest neighbors or in a ball of a given radius
- If there is a path from s to t in the resulting graph then terminate, otherwise add M more milestones

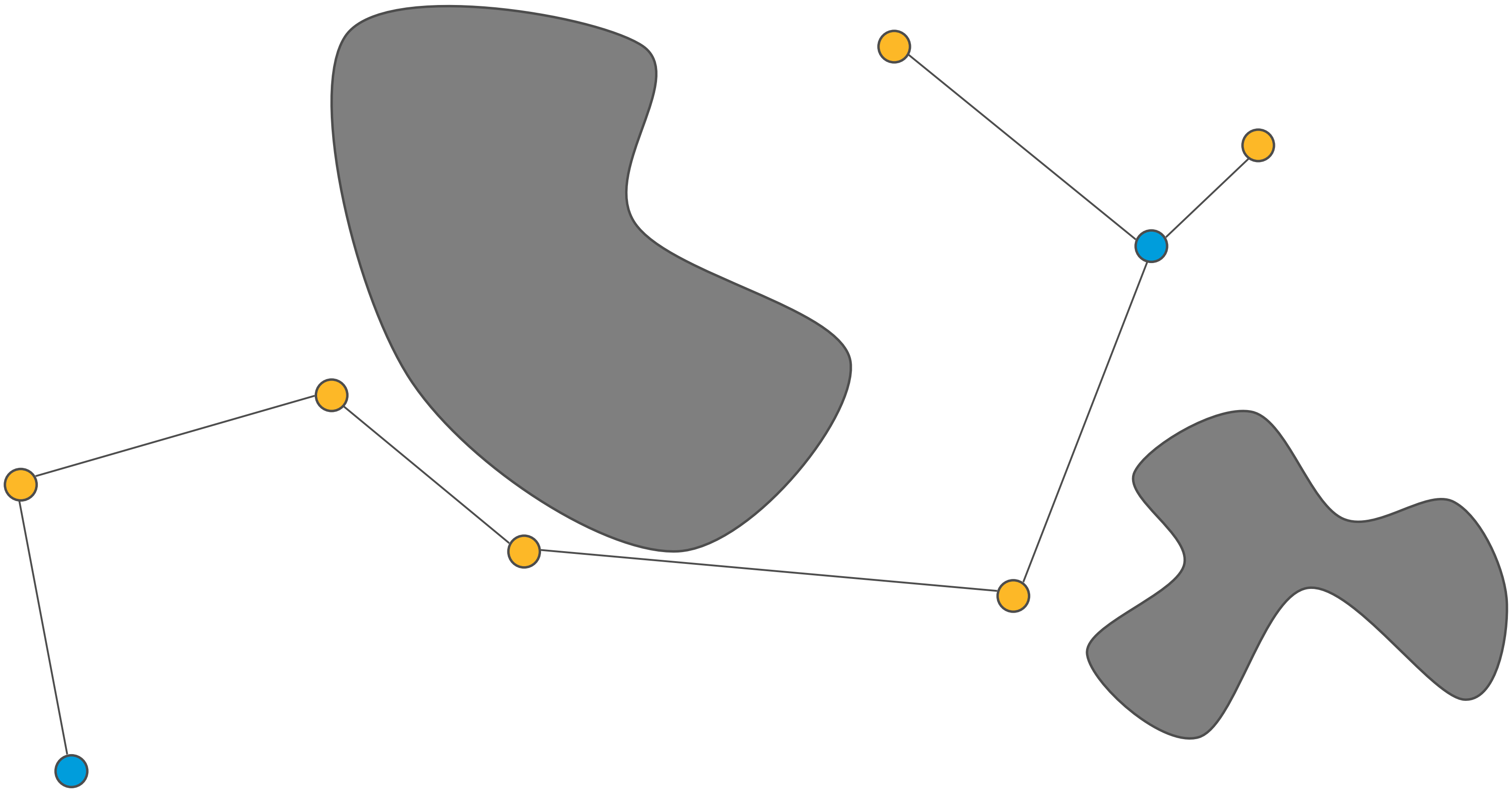
PROBABILISTIC ROADMAP



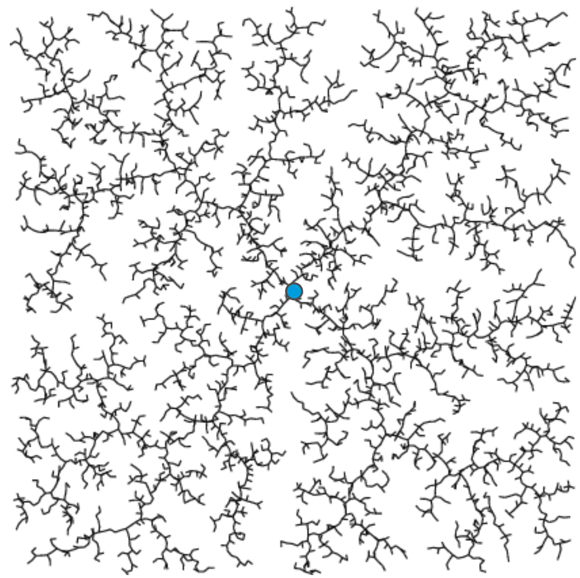
RAPIDLY-EXPLORING RANDOM TREES

- Incrementally build two trees with *s* and *t* as roots
- Every time a new milestone is added, connect it to the closest visible point in each tree
- If the new milestone connects to both trees then we're done

RAPIDLY-EXPLORING RANDOM TREES



RAPIDLY-EXPLORING RANDOM TREES



[Image from Lavalle]

In practice there's also a length parameter δ such that the new edge is cut after a distance of at most δ