Lecture 20

Path Planning Fundamentals

CS 3630

CMDragons '15

Manuela Veloso

Joydeep Biswas, Juan Pablo Mendoza

Danny Zhu, Richard Wang

Philip Cooksey, Steven Klee

Highlights from the RoboCup 2015 SSL Final:
CMDragons vs. MRL

School of Computer Science
Carnegie Mellon University

http://www.cs.cmu.edu/~robosoccer/small/

Fundamentals

- **Mobile robot path planning**:

  Identifying a sequence of actions that, when executed, will enable the robot to reach the goal location

- **Representation:**
  - State (state space)
  - Actions
  - Initial and goal states

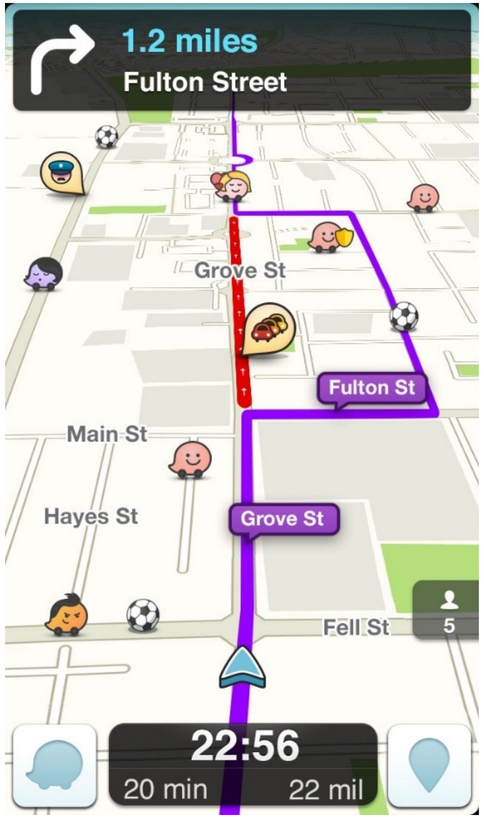
- **Plan**:

  Sequence of actions/states that achieve desired goal state.

Fundamental Questions

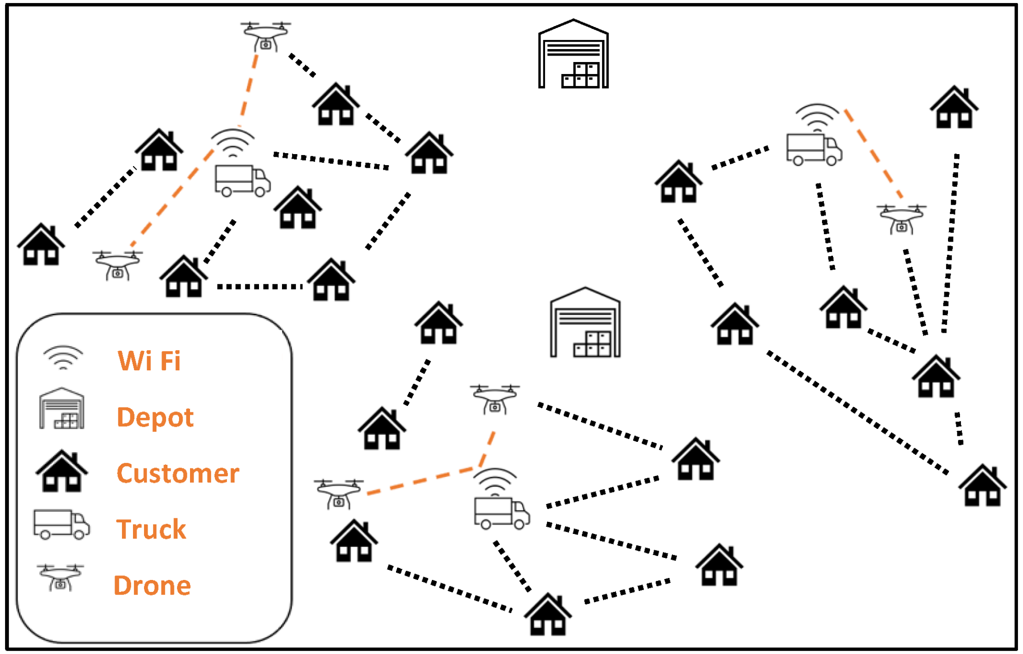
- What domain/application constraints do we need to consider?
- How is a plan computed?
- How is a plan represented/encoded?
- What does the plan achieve?
- How do we evaluate a plan's quality?

Broad range of applications



Route Planning

NPC Planning



Logistics Planning

Crop Coverage Planning

Representation

We need to represent two things:

The World

The Robot

The World consists of...

- **Obstacles**
  - Places where the robot can't (shouldn't) go
- **Free Space**
  - Unoccupied space within the world
- **Unknown**
  - Robots "might" be able to go here
    - There may be unknown obstacles
    - The state may be unreachable

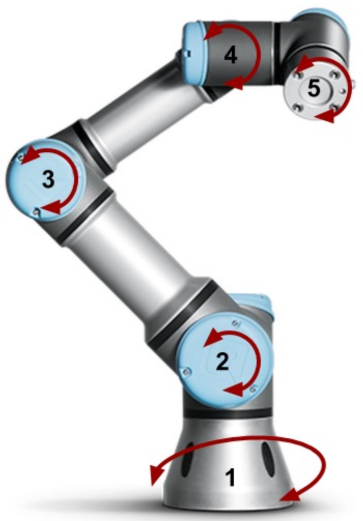
What happens if we assume the opposite?

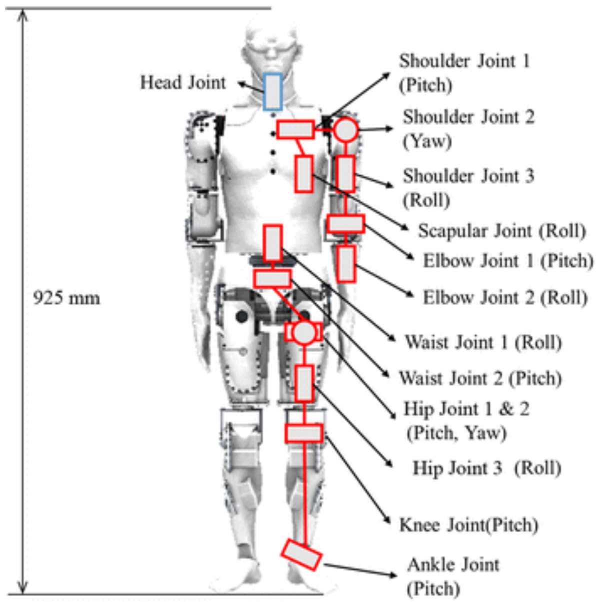
When planning, we typically assume that all unknown space is free space.

How do we represent the robot?

Degrees of Freedom

- Degrees of Freedom (DOF) is used to abstractly define the motion capabilities of a robot.
- The number of DOFs corresponds to the number of <u>moveable joints</u> the robot has
  - The higher the DOF, the more complex and adaptable the robot is… and typically the harder to control

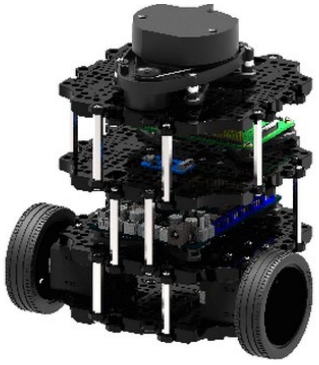




Mobile Robots

- Wheeled robots don't have joints
  - i.e. wheels≠joints
- Instead of DOF, the robot's capabilities can be described in terms of:
  - degree of mobility δ_m
  - degree of steerability δ_s
  - degree of maneuverability δ_M

$$\delta_M = \delta_m + \delta_s$$



Wheeled Robot Designs

Not a controllable wheel, just a ball castor

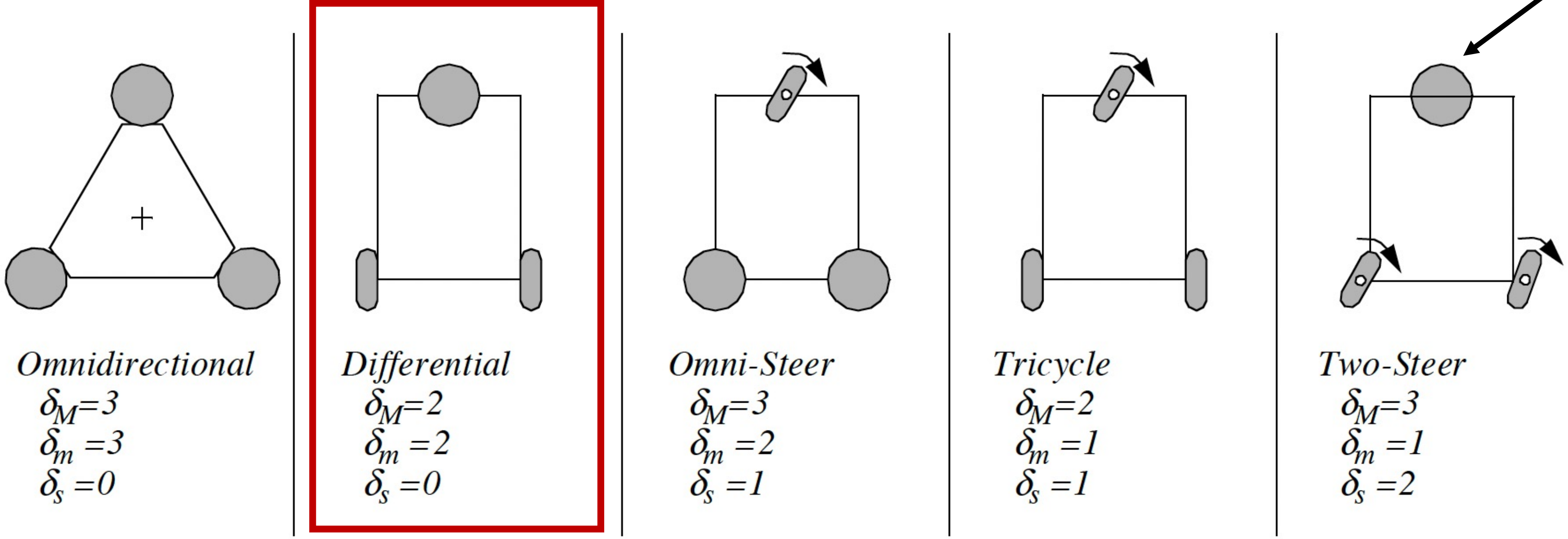


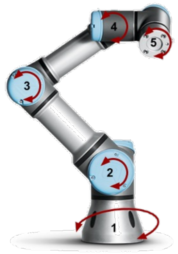
Fig 3.14 *The five basic types of three-wheel configurations. The spheric wheels can be replaced by castor or swedish wheels without influencing the maneuverability. More configurations with various numbers of wheels are found chapter 2.*

Introduction to Autonomous Mobile Robots book

Configuration of a Robot

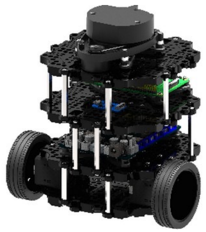
- The configuration of a robot is defined as the specification of the position of all points of the robot

- For a robotic arm, the configuration is defined by specifying the angle of each of the joints



  - dimensionality same as DOF

- For a differential drive robot, the configuration is specified by the robot's pose

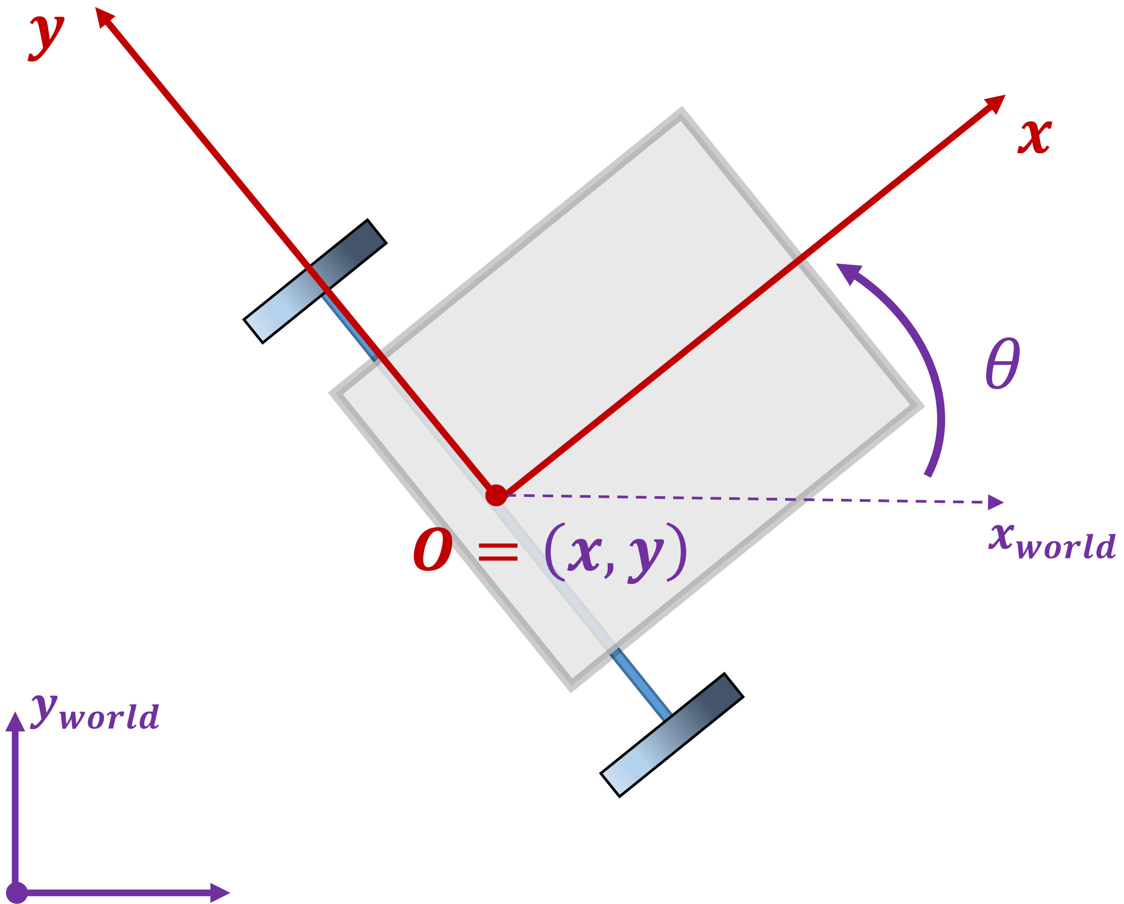


  - dimensionality is 3 (x, y, θ)

- Configuration space is the set of all possible configurations of a given robot in a given environment

Configuration Space

We use q to denote a point in a configuration space Q.



differential drive robot

Because our DDR can rotate in the plane, it is necessary to know both the position and the orientation of the body-attached frame to specify a configuration:

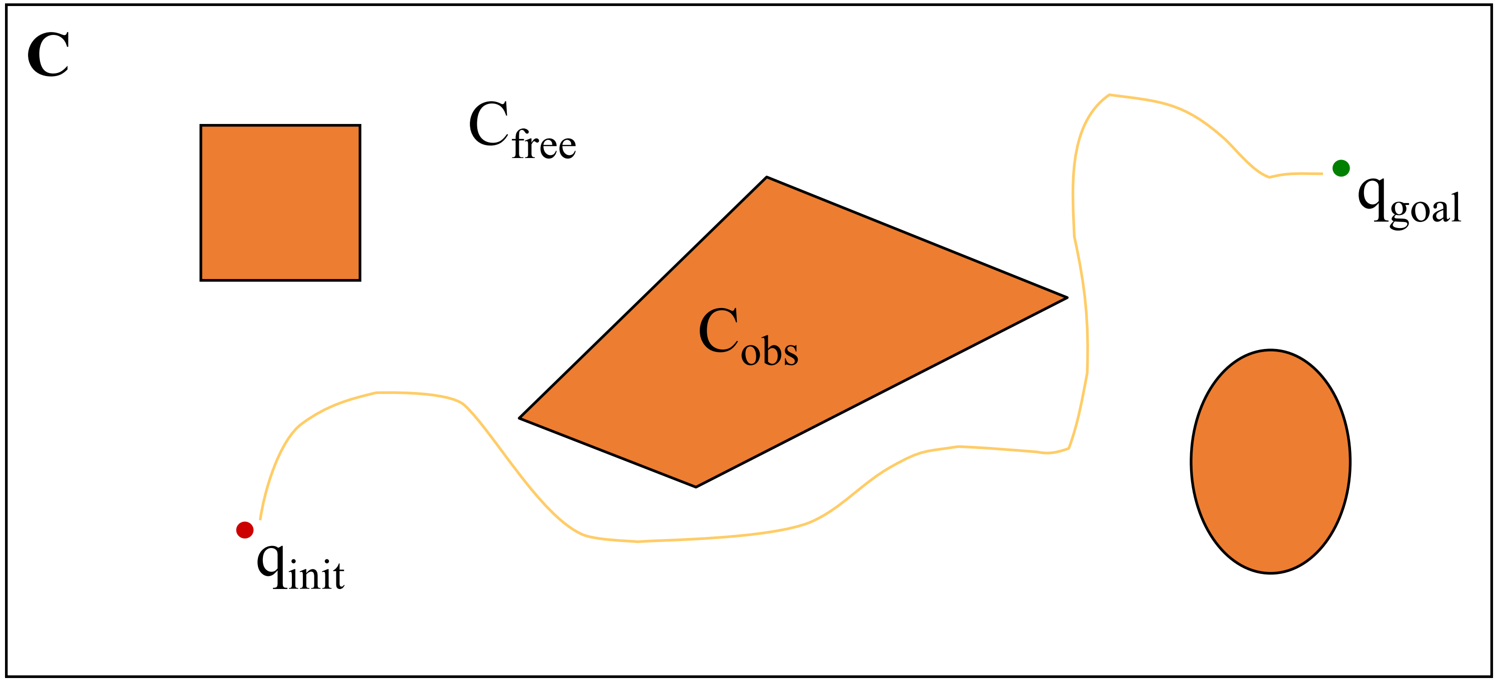
$$Q = \mathbb{R}^2 \times [0, 2\pi)$$

$$q = (x, y, \theta) \in Q$$

If we know the configuration, $q = (x, y, \theta)$, we can compute the location of any point on the robot.

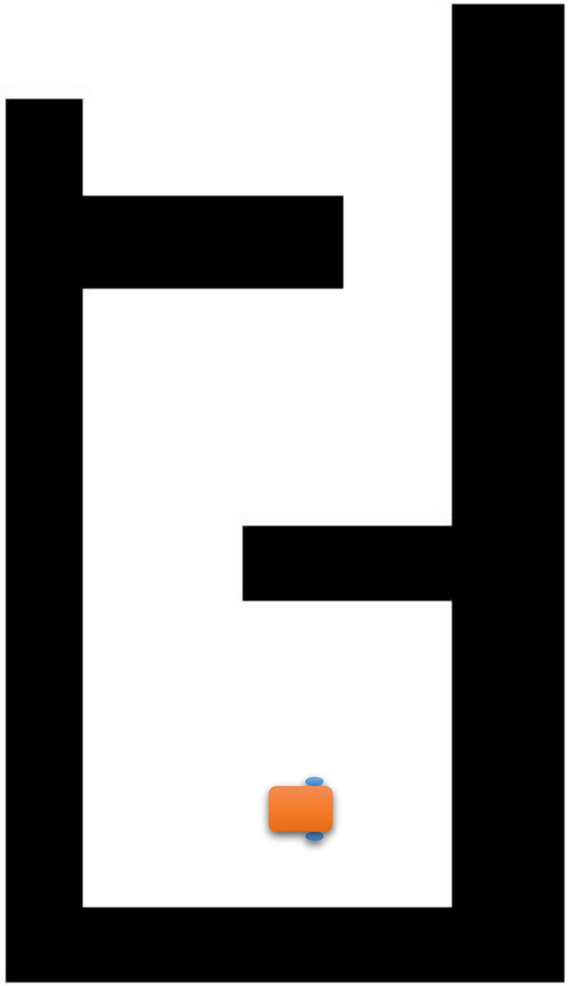
Example Configuration Space

Point robot (no constraints)

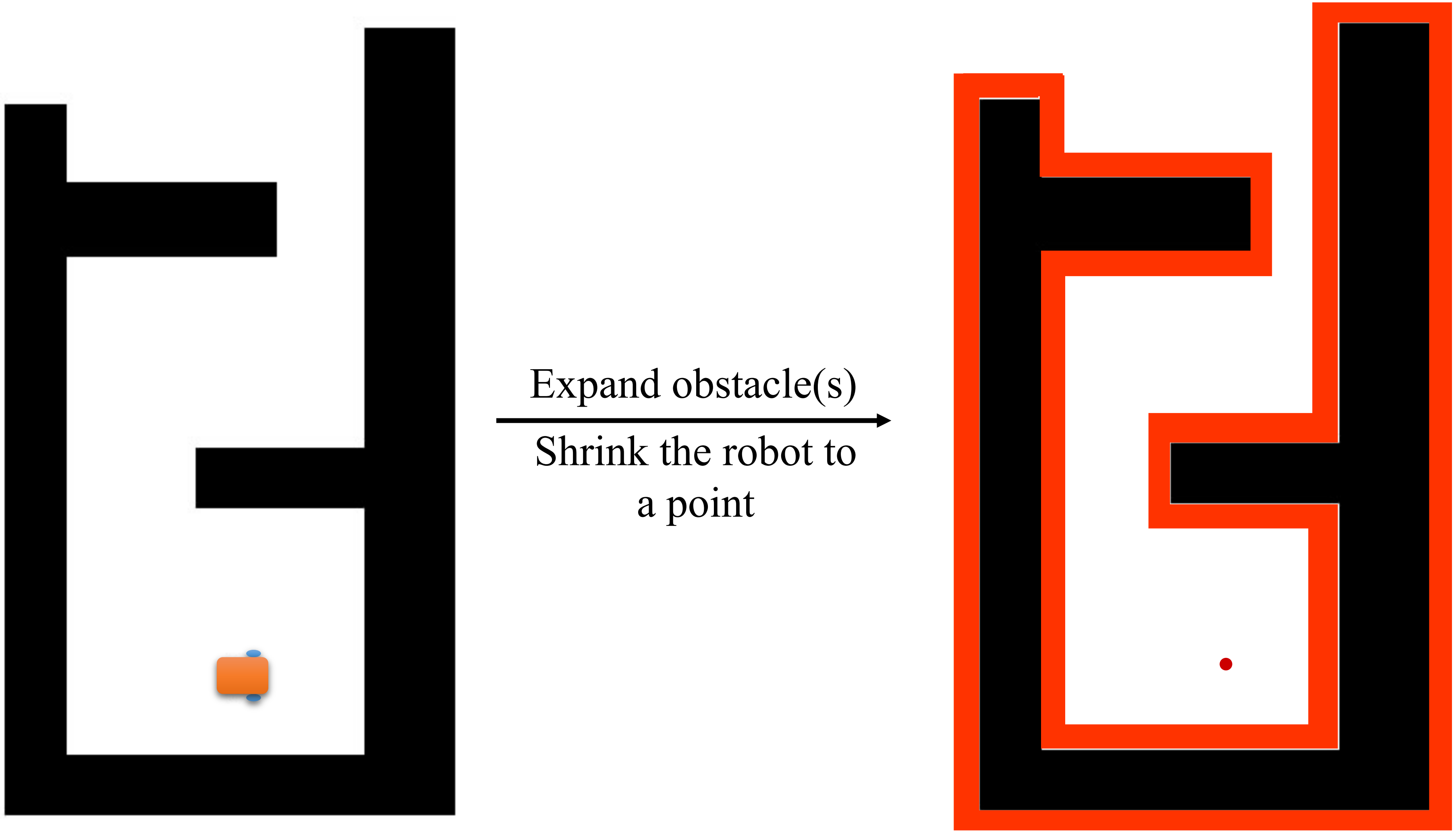


For a point robot moving in 2-D plane, C-space is in $\mathbb{R}^2$ => 2 DOF

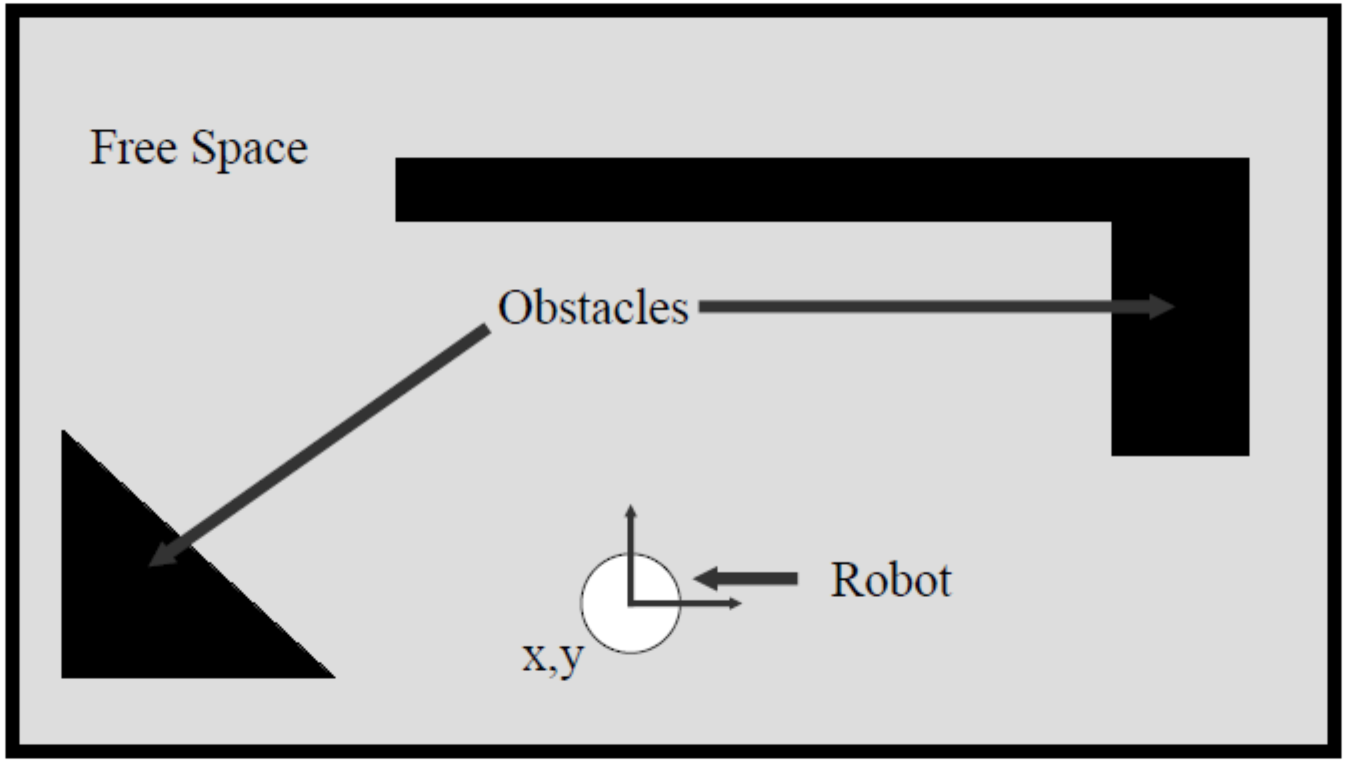
What if the robot is not a point?



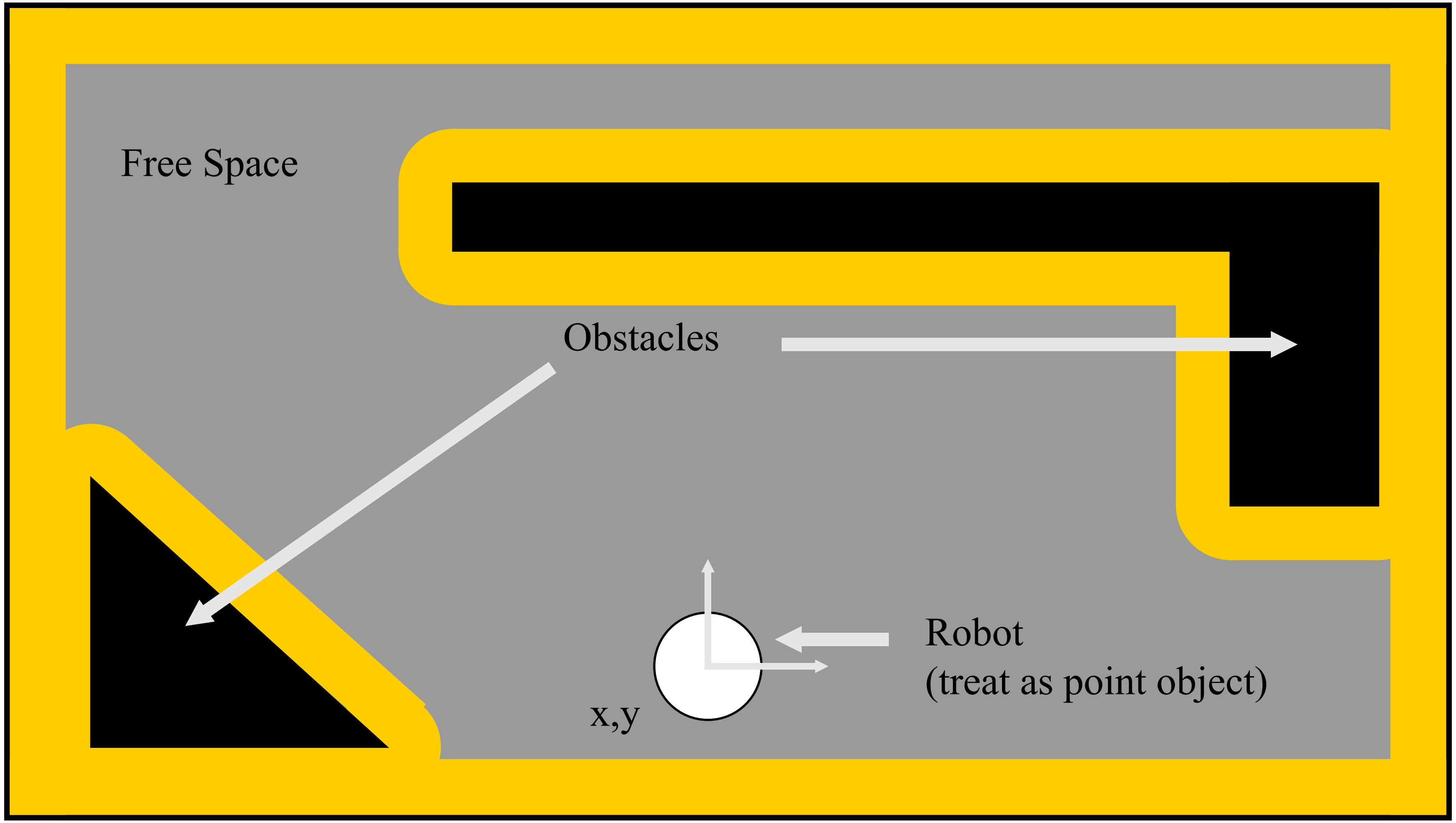
What if the robot is not a point?



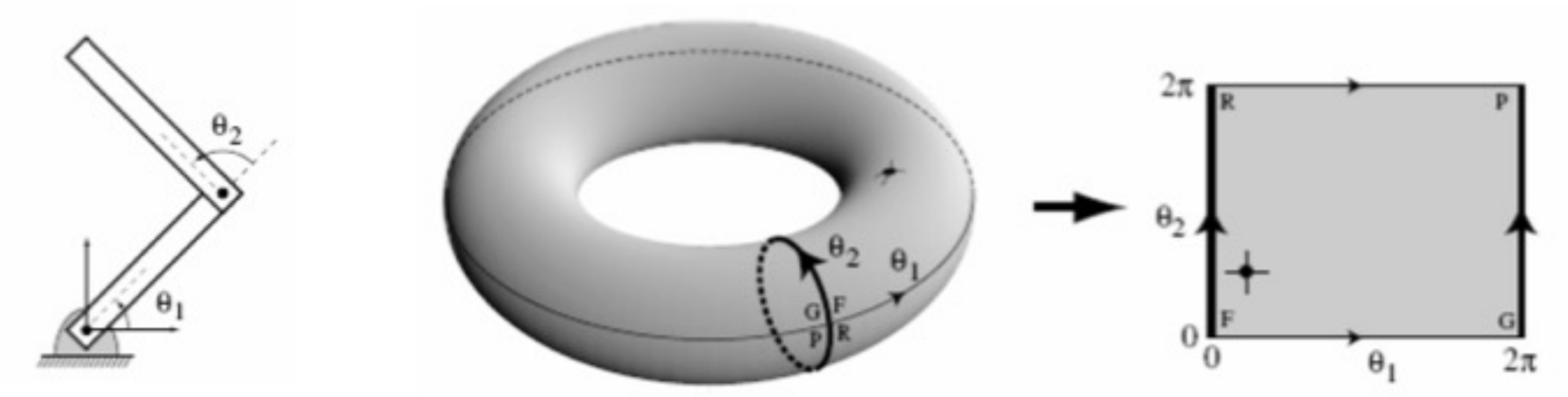
Example Workspace and Robot

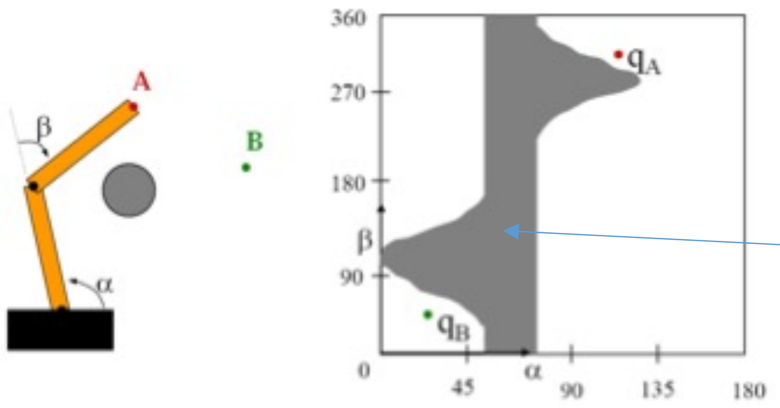


Configuration Space: Accommodate Robot Size



Configuration Space for Robot Arms





Configurations in which the robot would hit the obstacle

Source: L. Kavraki, Rice

Interactive demo:

https://www.cs.unc.edu/~jeffi/c-space/robot.xhtml

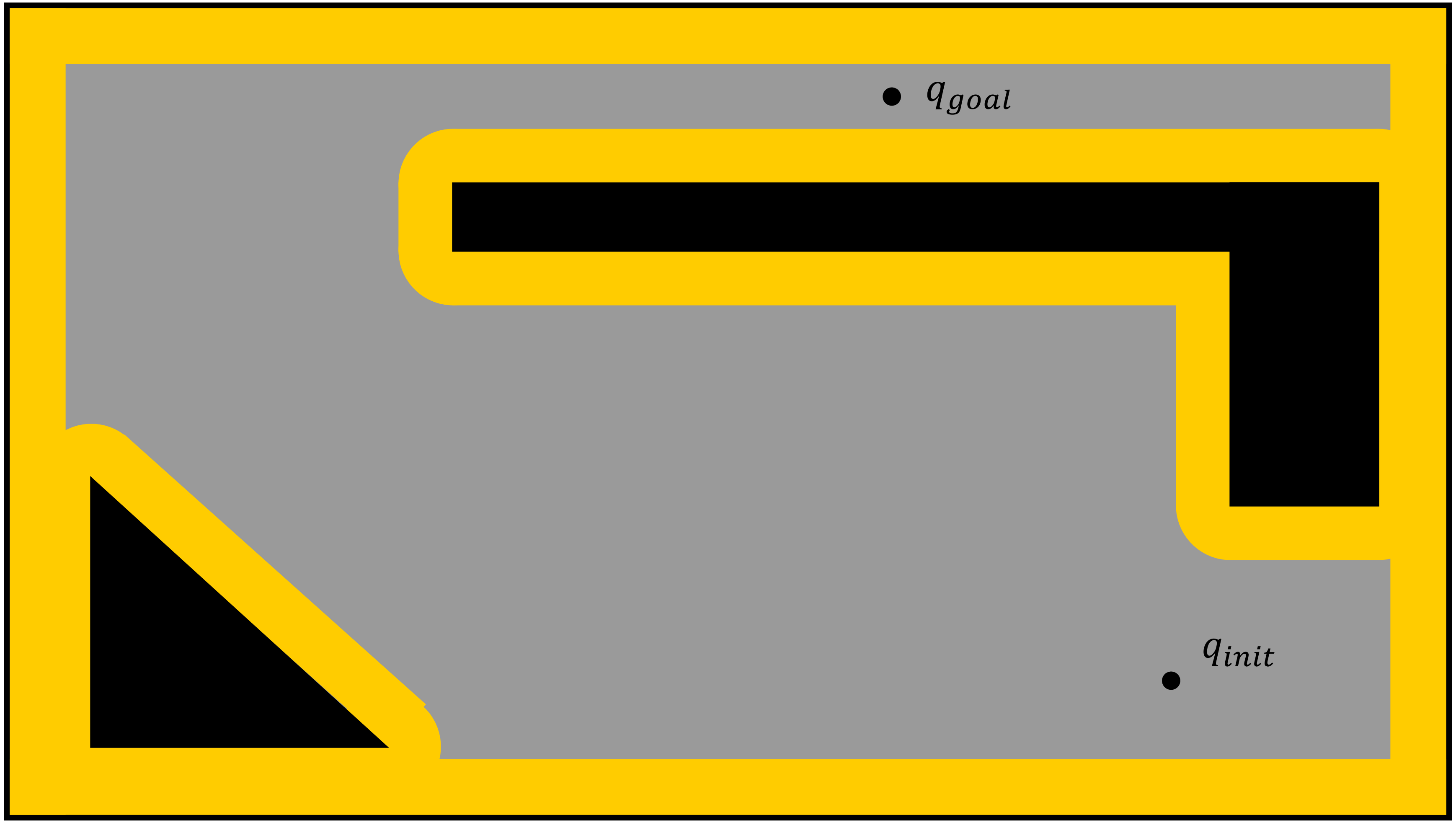
Path Planning

- Find **a collision-free path** from the starting configuration q_{init} to a goal configuration q_{goal}.
- Collision checking between the robot and obstacles can be computationally heavy, so we deal with this problem by mapping obstacles in the world to the robot's configuration space.
- In the configuration space, we now have the problem of finding a path for a single point (which represents the configuration of the robot).

- The above is fairly straightforward for 2D configuration spaces. Very challenging in high-dimensional spaces.

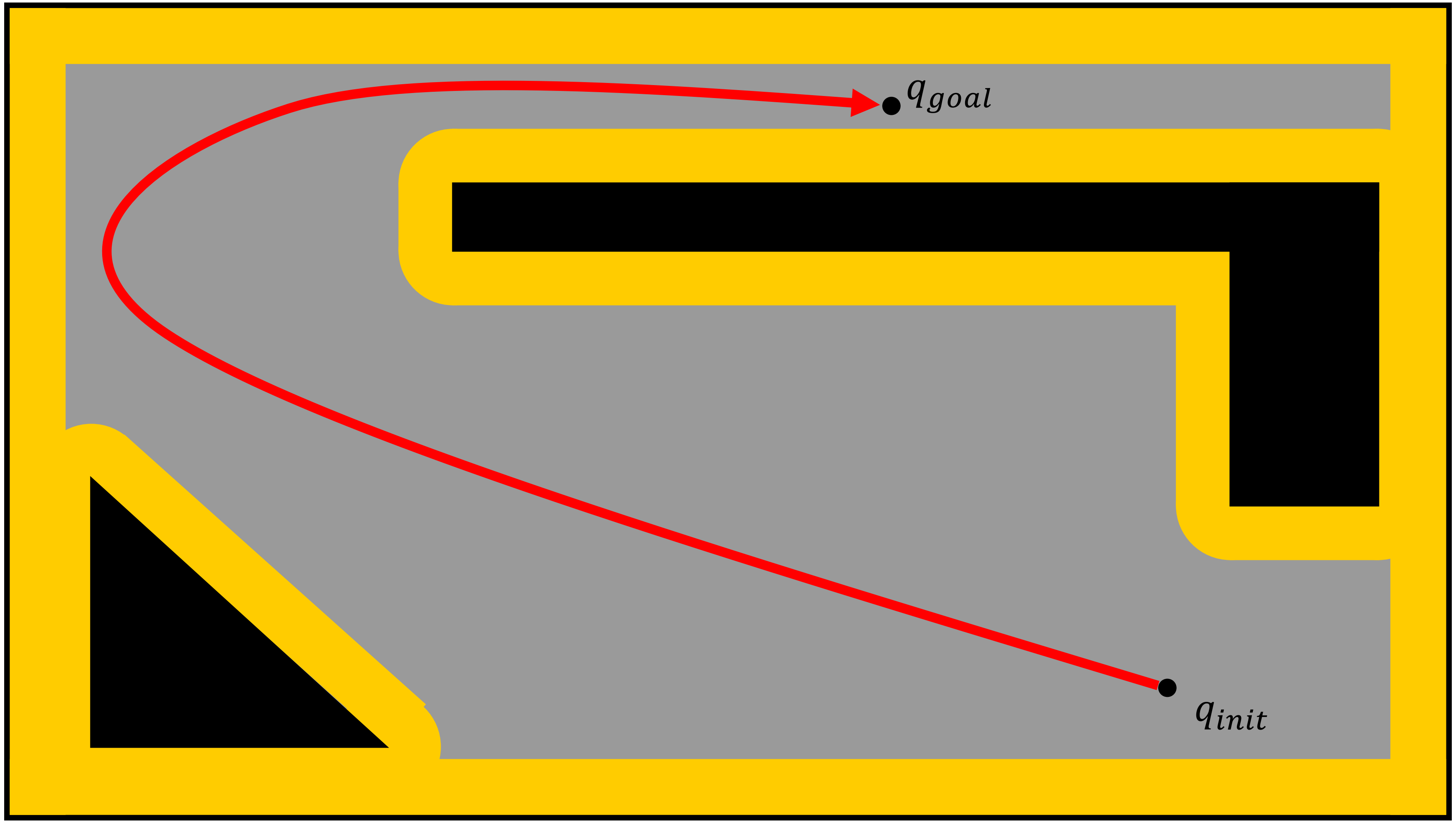
Planning a Collision-Free Path

Find a collision-free path from q_{init} to q_{goal}



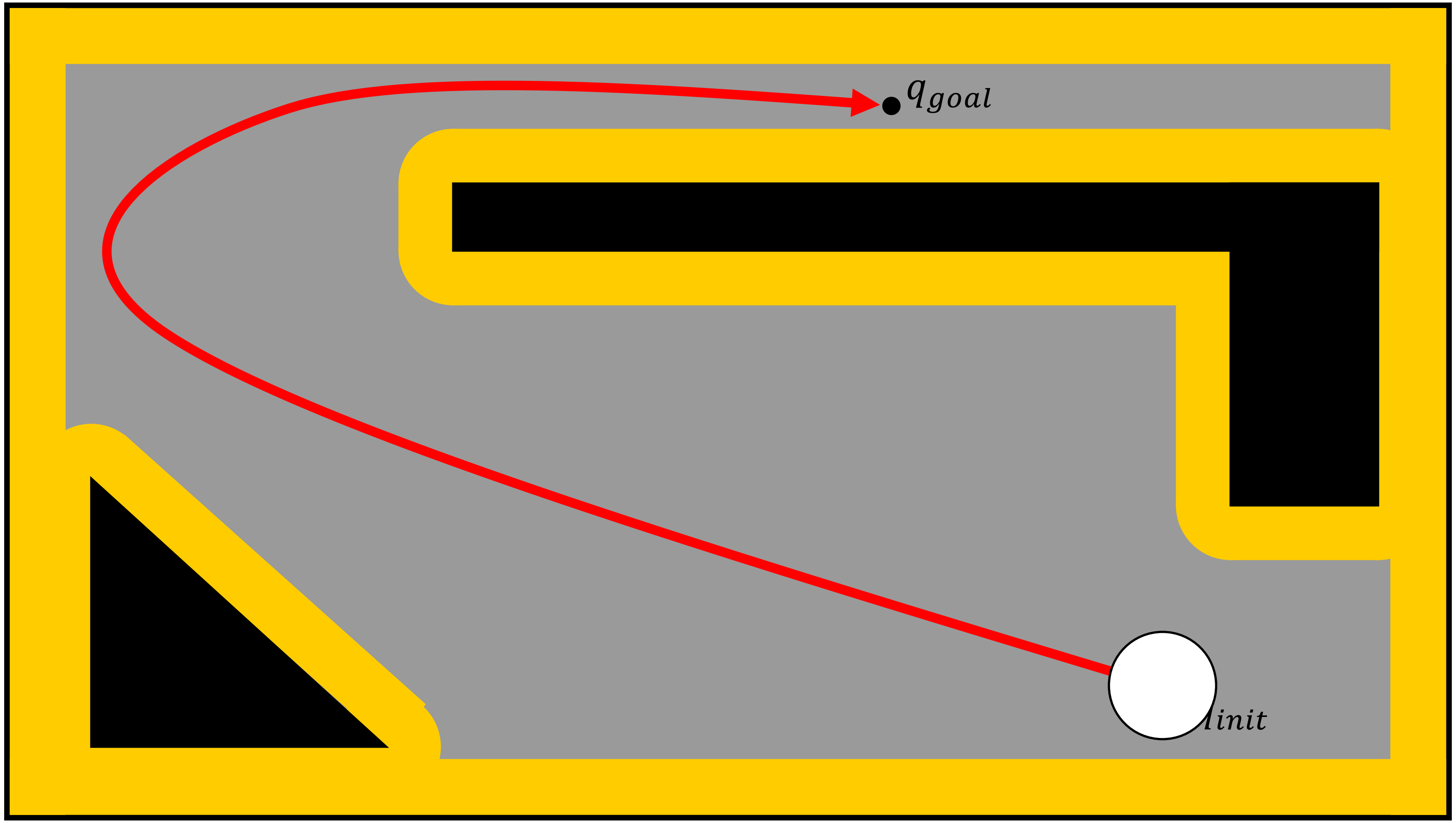
Planning a Collision-Free Path

Find a collision-free path from q_{init} to q_{goal}



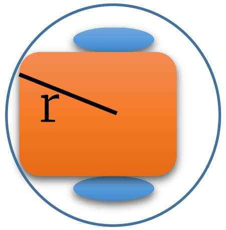
Planning a Collision-Free Path

Find a collision-free path from q_{init} to q_{goal}



For our application

- Our robot is close enough to being circular that it is fine to model it as a circle with a fixed radius.

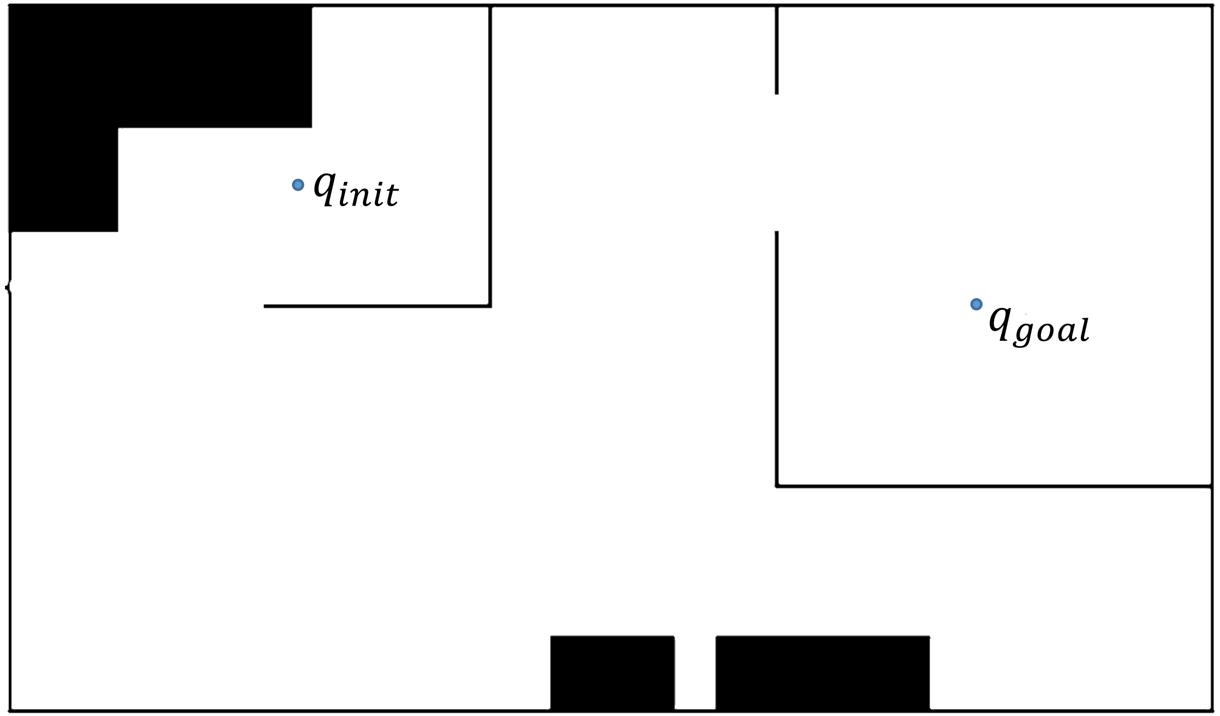


Path Planning…

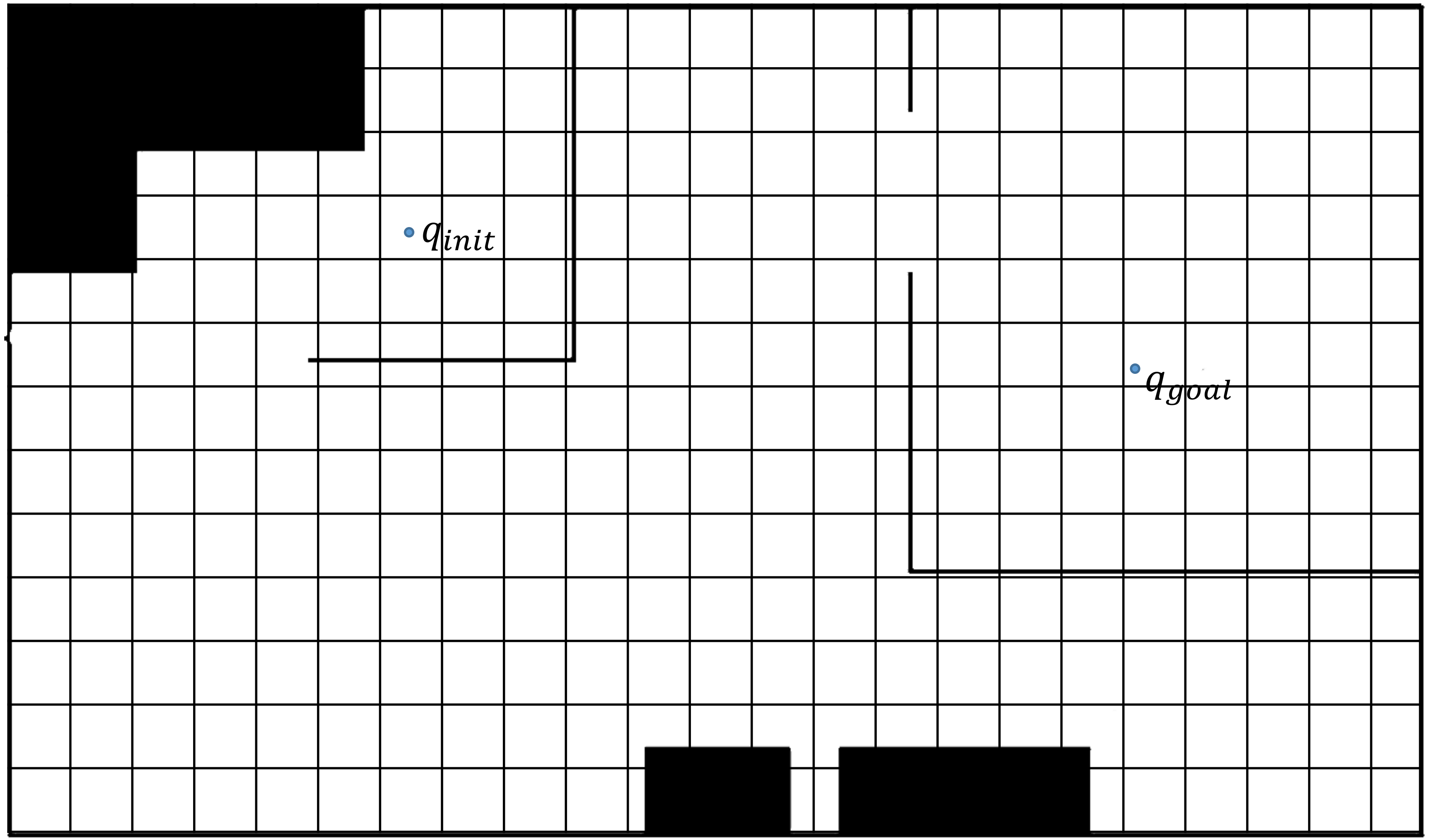
- Now let's assume that someone gave us a map. How do we plan a path from q_{init} to q_{goal}?

Fundamental Questions

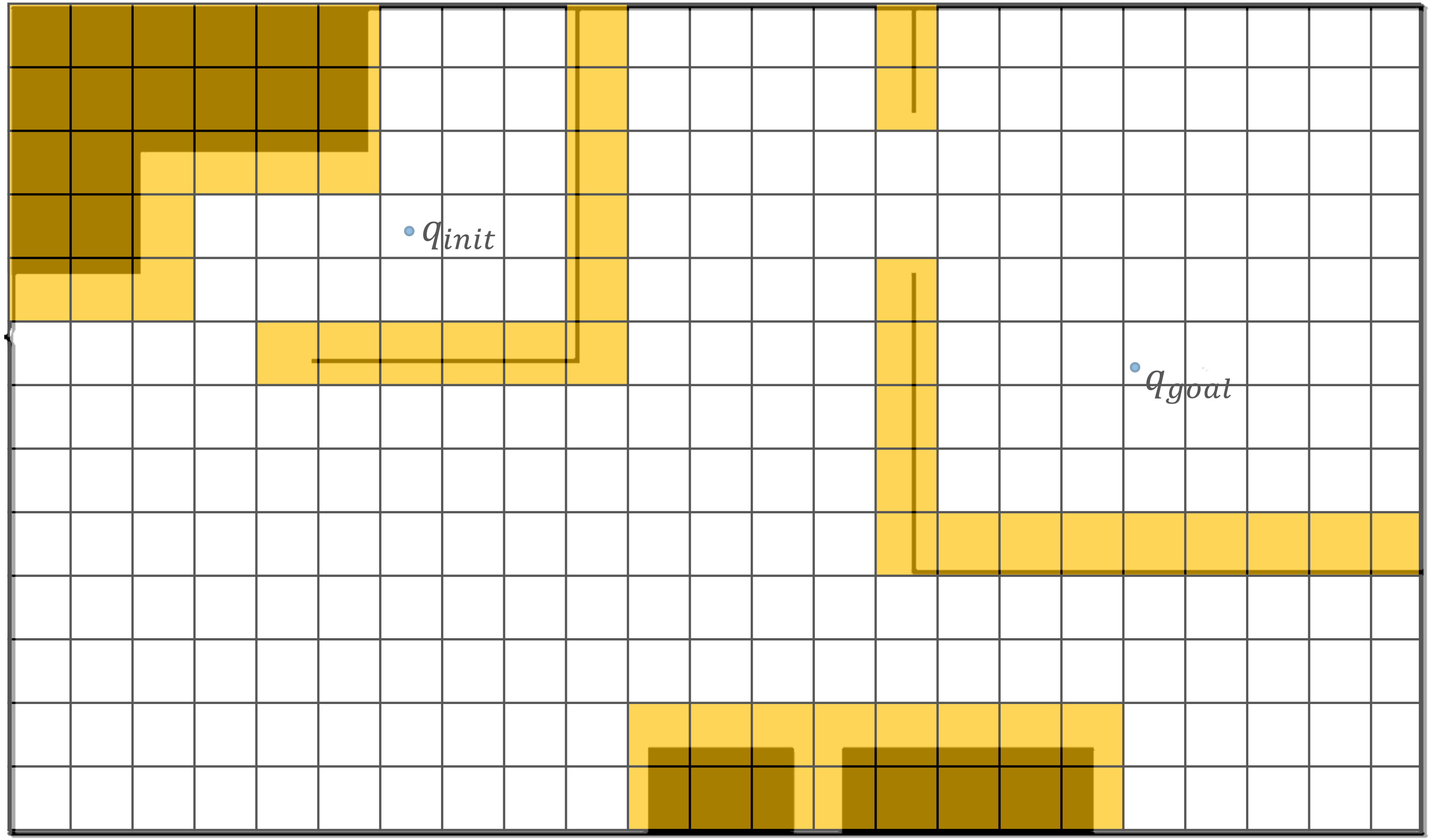
- How is a plan represented?
- How is a plan computed?
- What does the plan achieve?
- How do we evaluate a plan's quality?



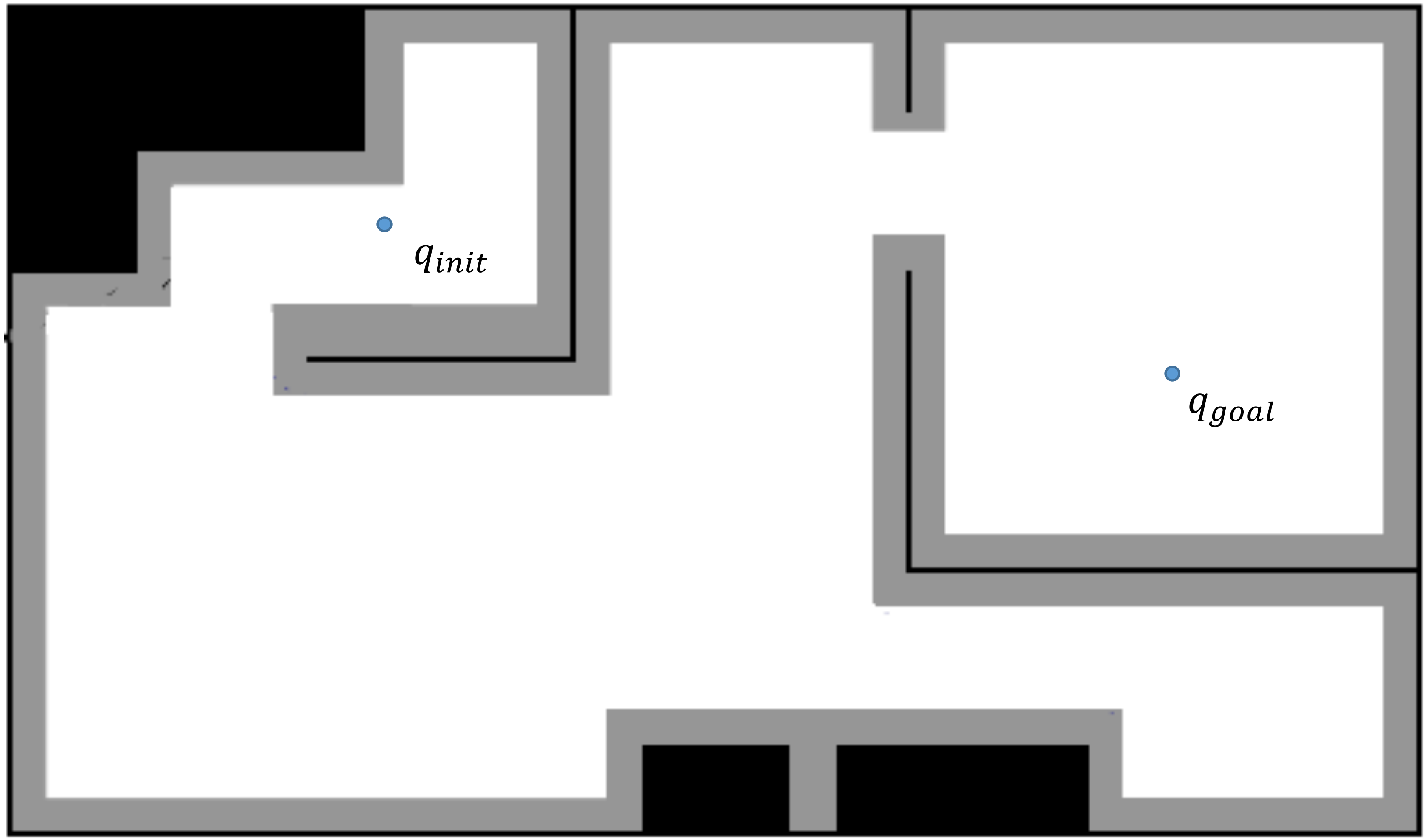
Occupancy Grid



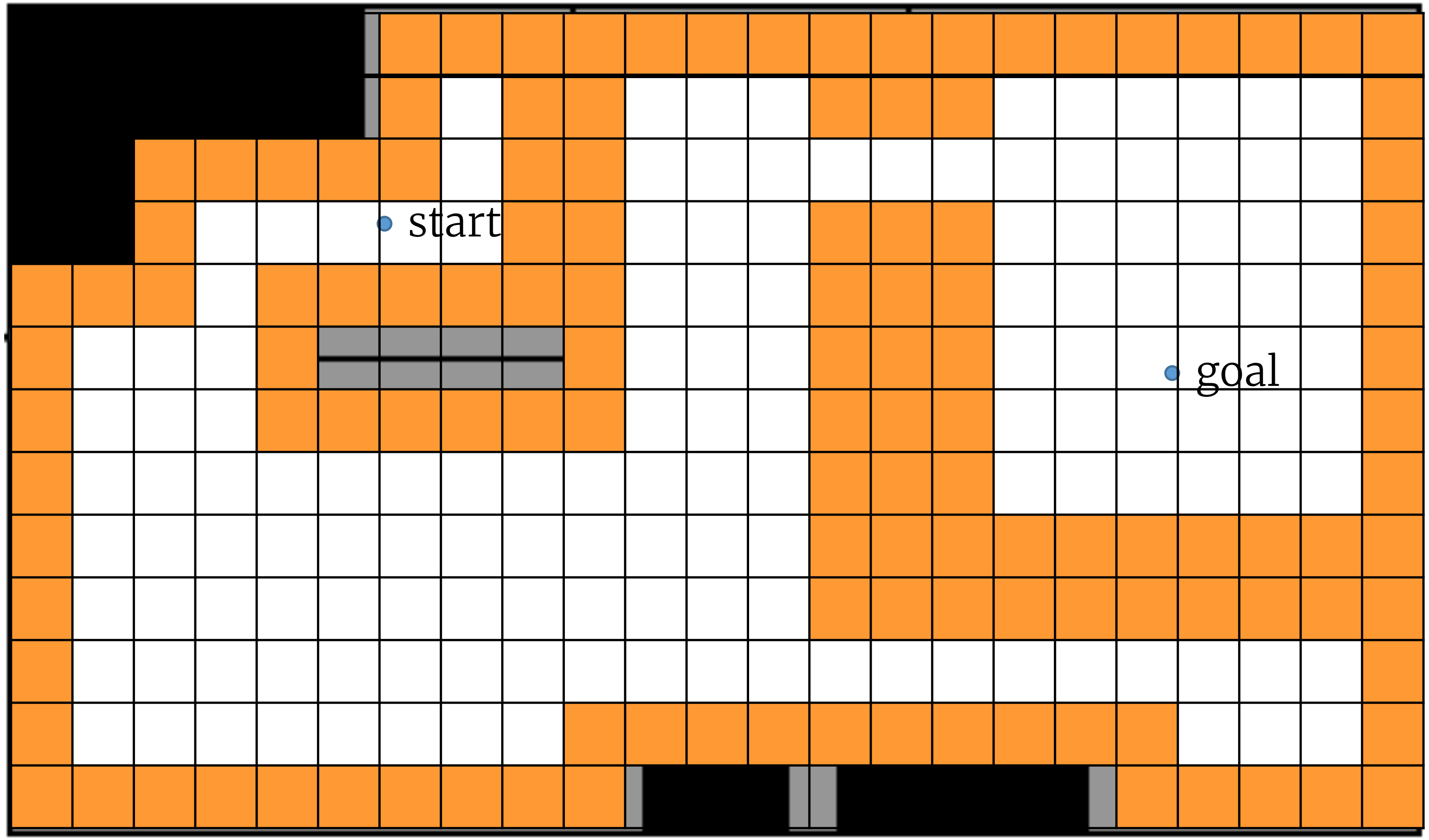
Occupancy Grid



What about the c-space?

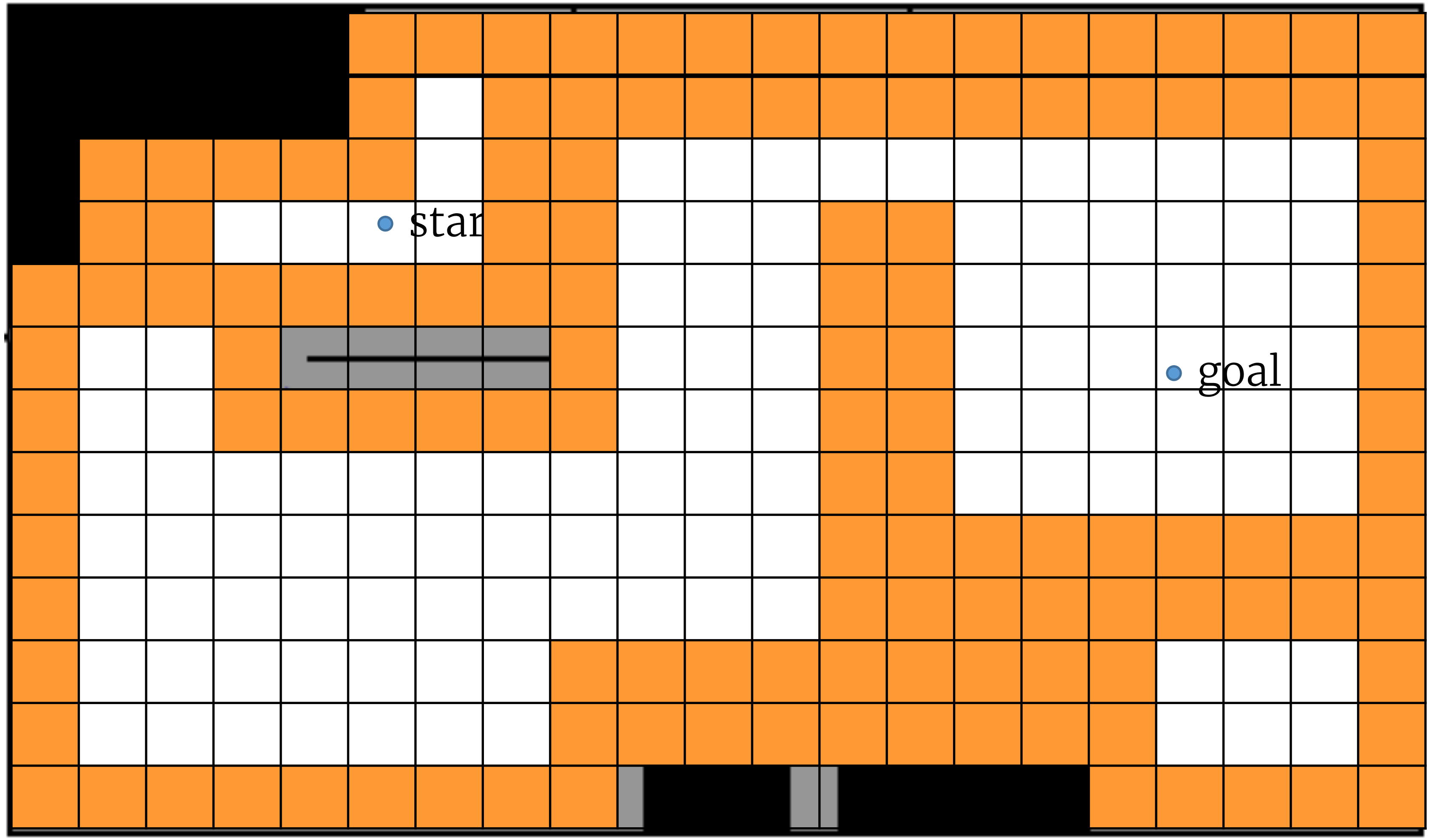


Occupancy Grid, accounting for C-Space



Occupancy Grid, accounting for C-Space

Slightly larger grid size can make the goal unreachable.



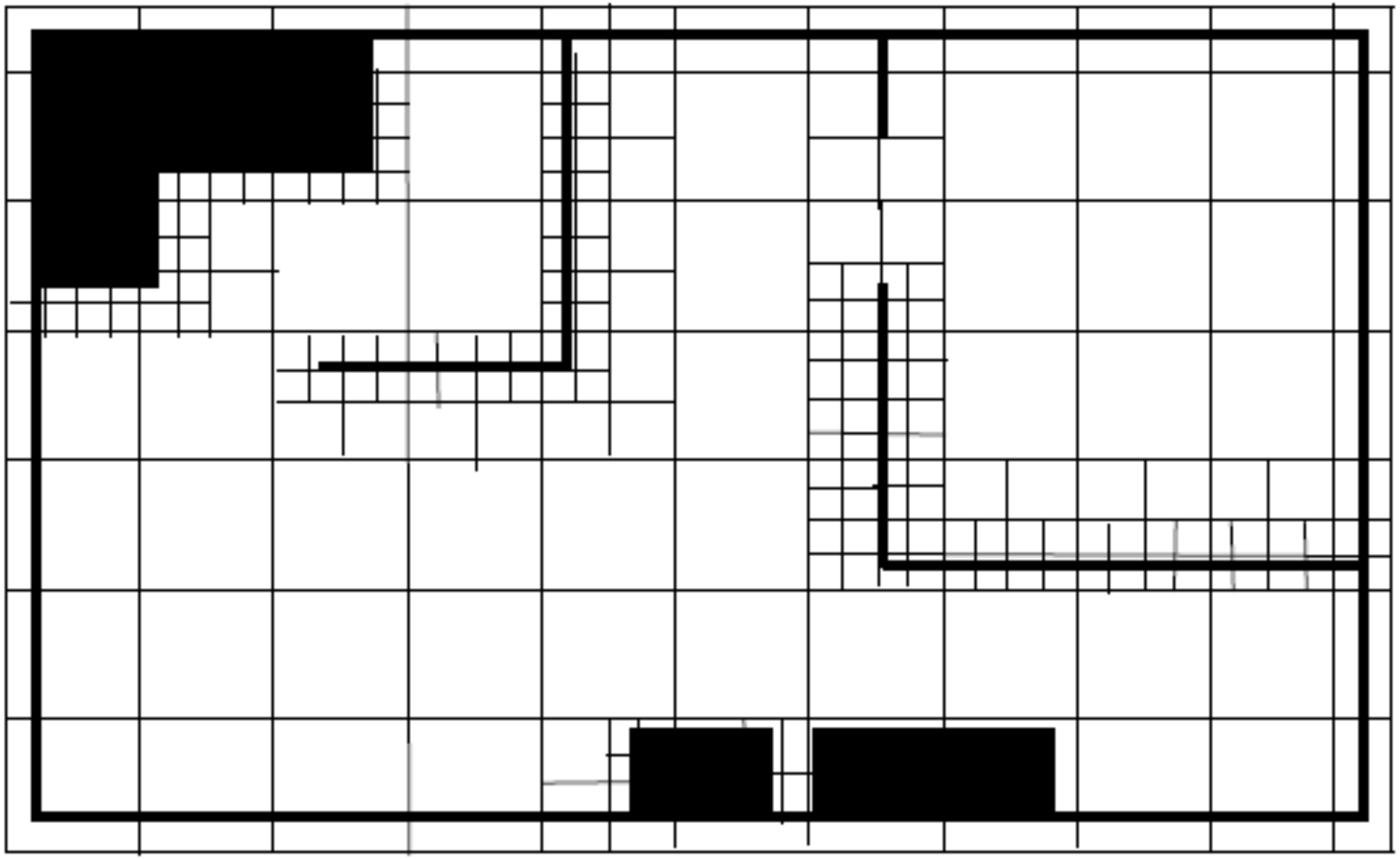
Alternative Grid-Based Methods

- Use variable size grids:
  - Quadtrees
  - Hierarchical cell decomposition

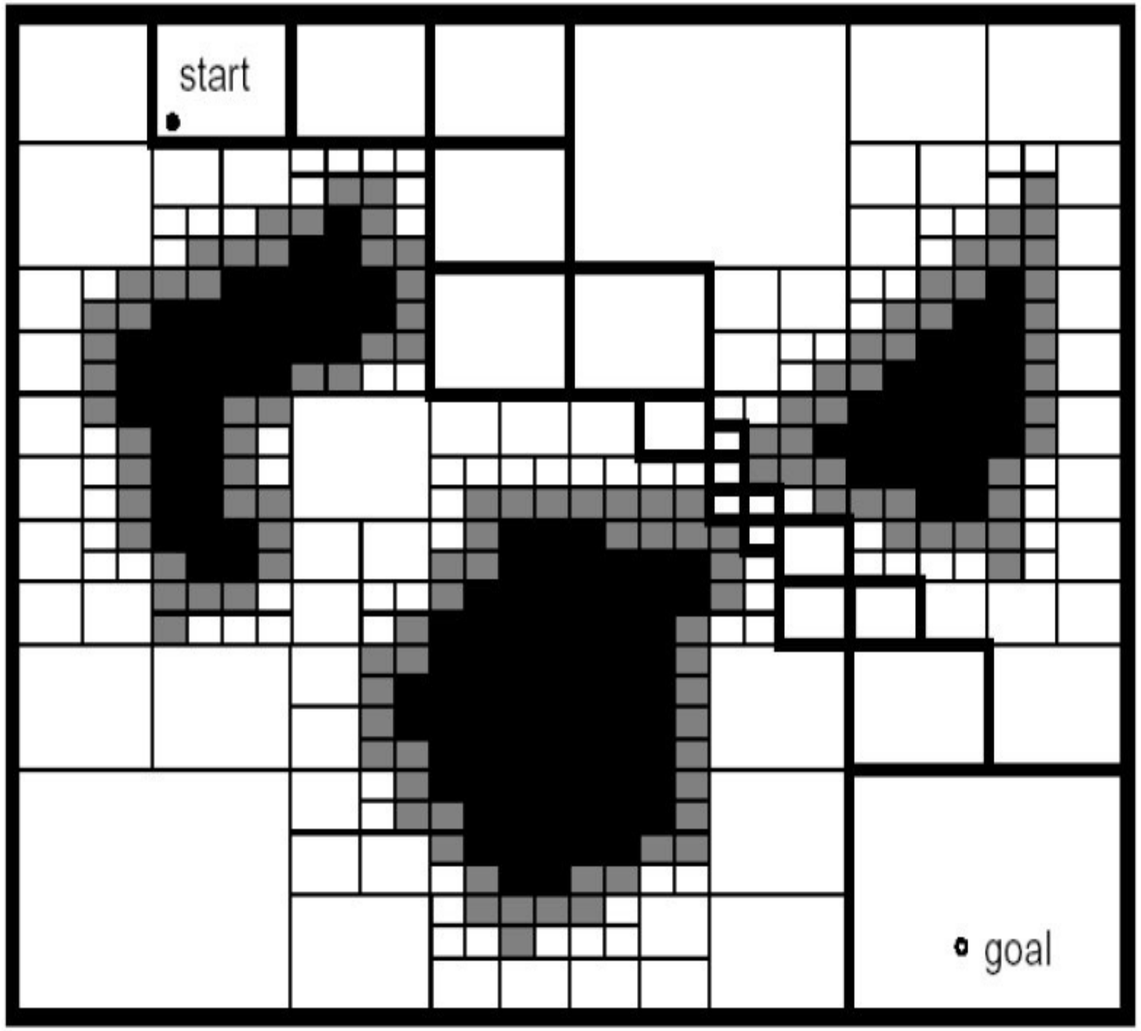
Quadtree

- Start with a large default grid size
- If *part* of the grid contains an obstacle, subdivide that cell into four quadrants
- Continue splitting partially filled cells until a minimum allowed grid size is reached
- Do not split cells entirely filled with obstacles

Quadtree: An Example



Quadtree: Another Example



Big Picture

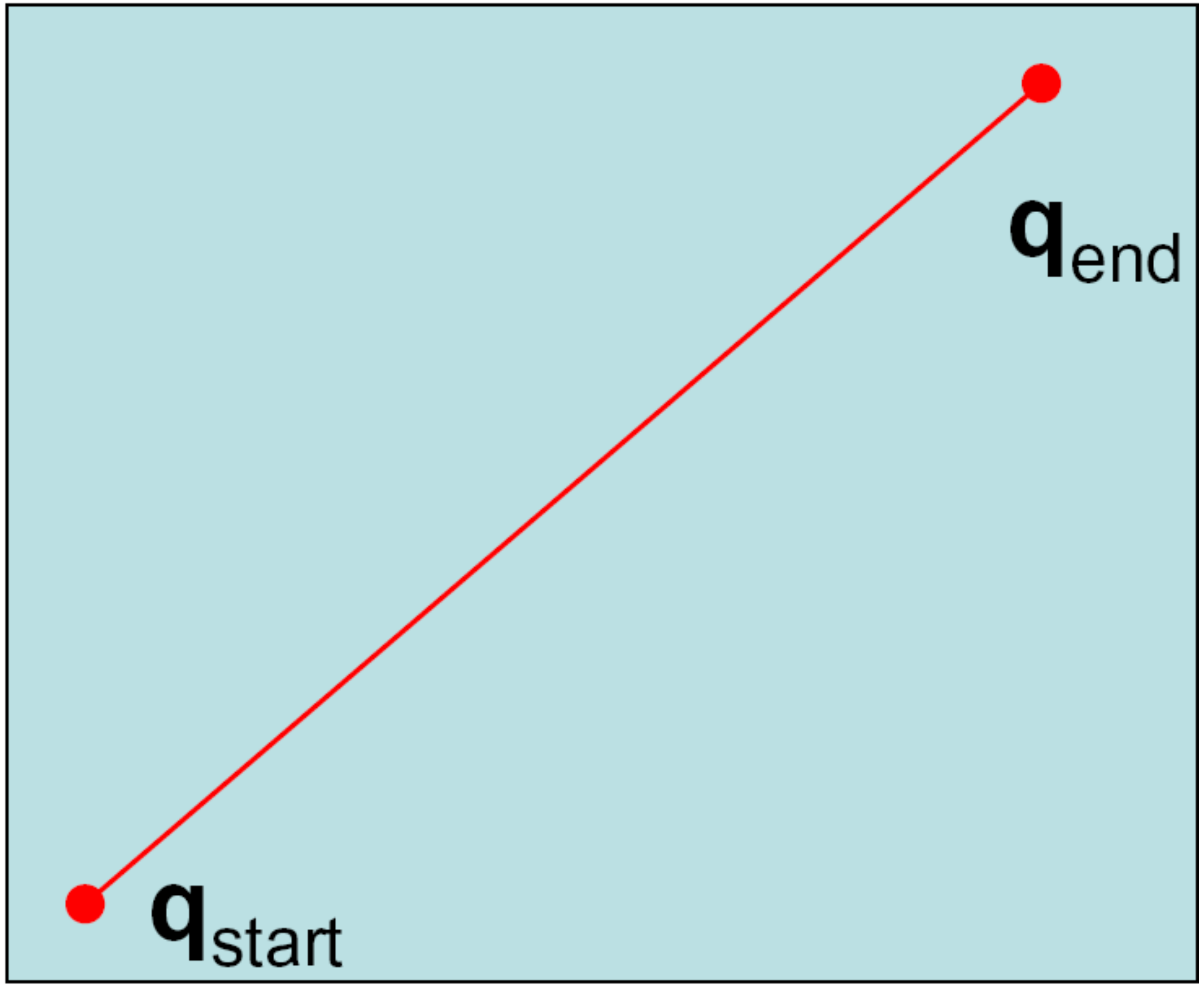
- Occupancy grids perform exhaustive search across the state space.
  - Represent and evaluate a far greater number of world states than the robot would actually need to traverse

- What can we do differently?

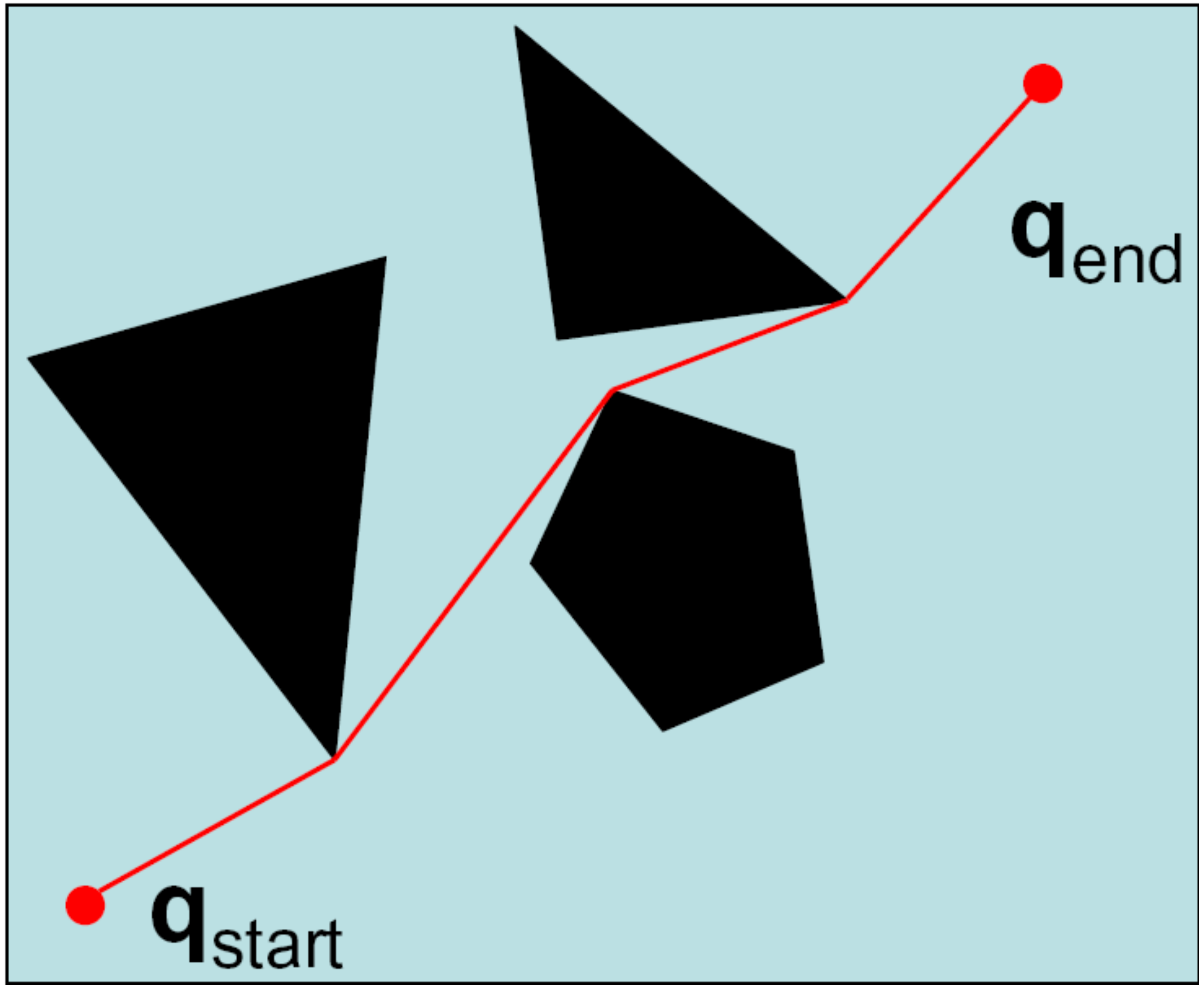
Roadmap Algorithms

- General idea:
  - Avoid searching the entire state space
  - Pre-compute a (hopefully) small graph (a.k.a. roadmap), such that staying on the "roads" will avoid obstacles
  - Find a path from q_{start} to the closest point on the roadmap, and then use the roadmap to find a path to q_{goal}

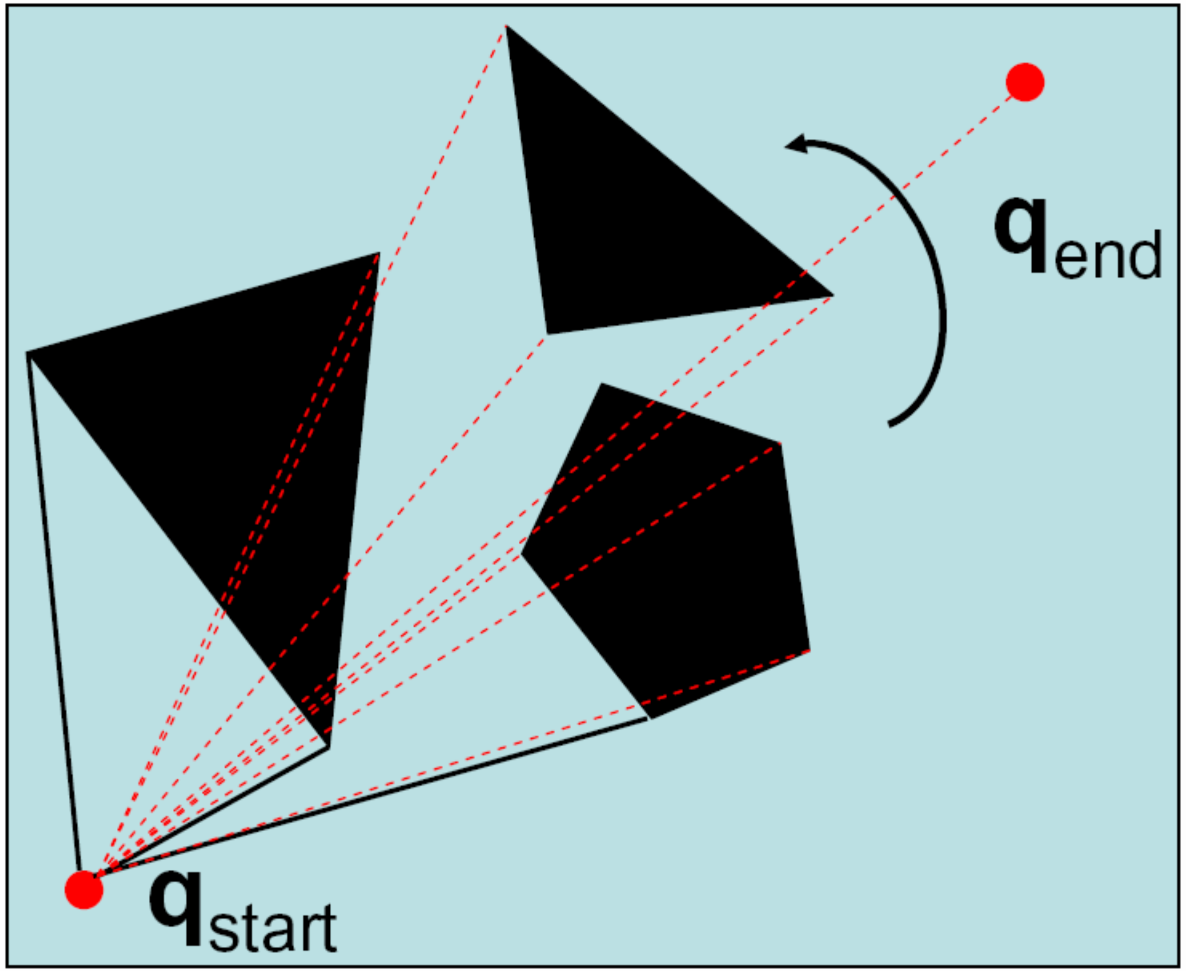
Shortest Path between Two Points



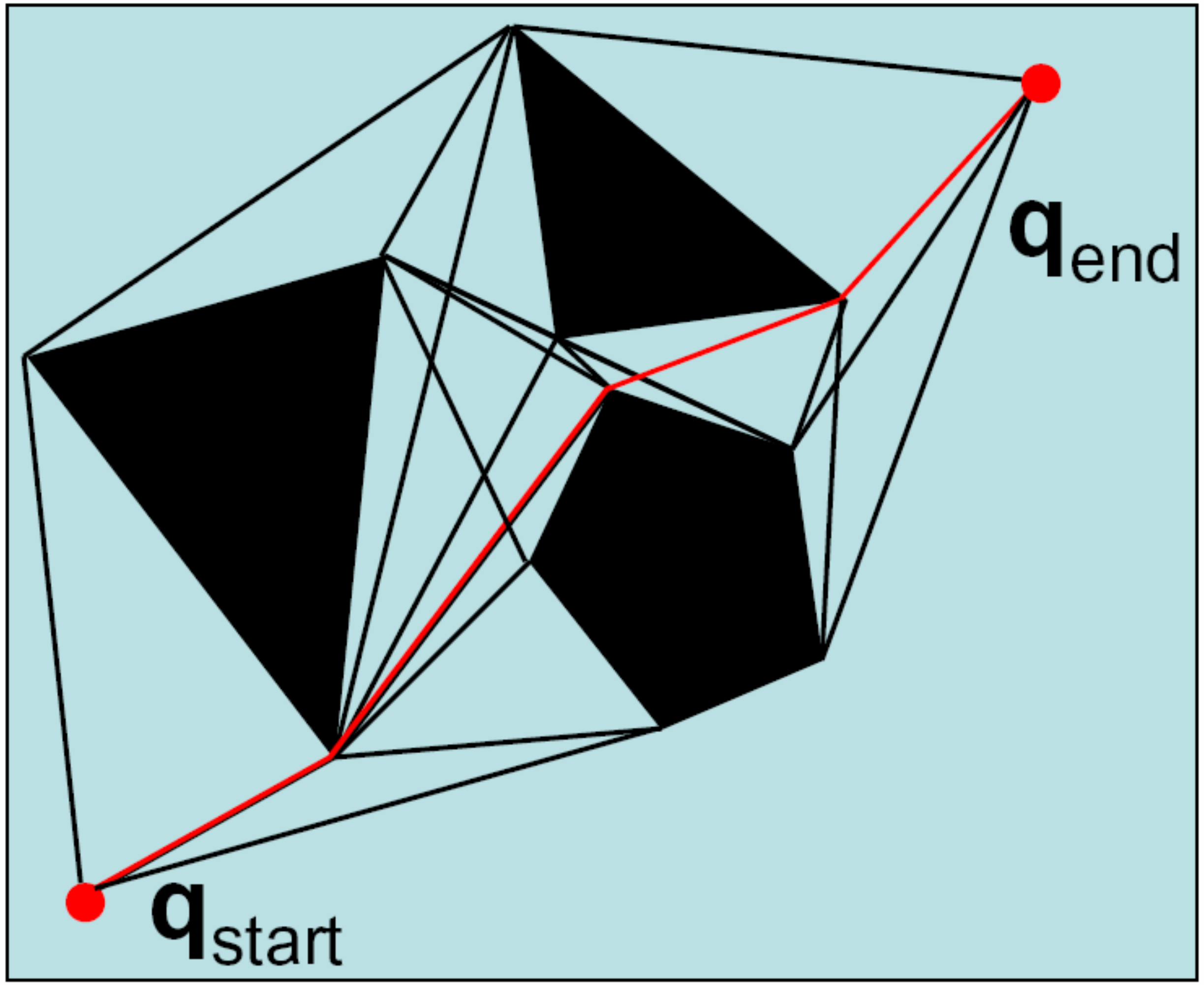
Shortest Path with Obstacles



The Sweep Algorithm



Visibility Graph



Visibility Graphs

- Shortest path but…
  - Stays as close as possible to the obstacles
  - Deviations from path can lead to collisions
  - Requires polygonal obstacles

Other Alternatives…

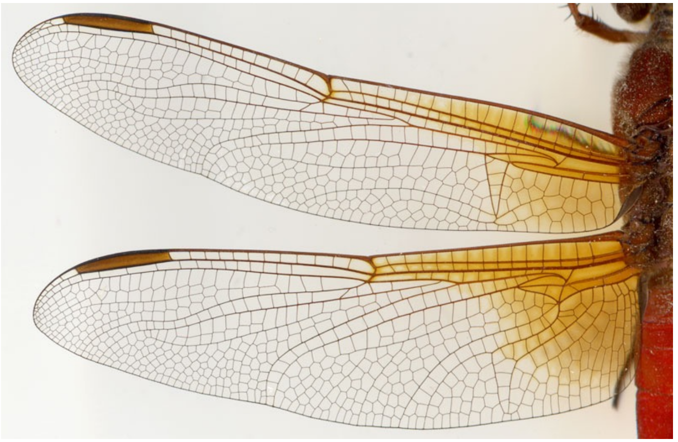
- What if we care more about keeping our robot safe than about optimality?

- The other extreme: stay away from obstacles as far as possible.

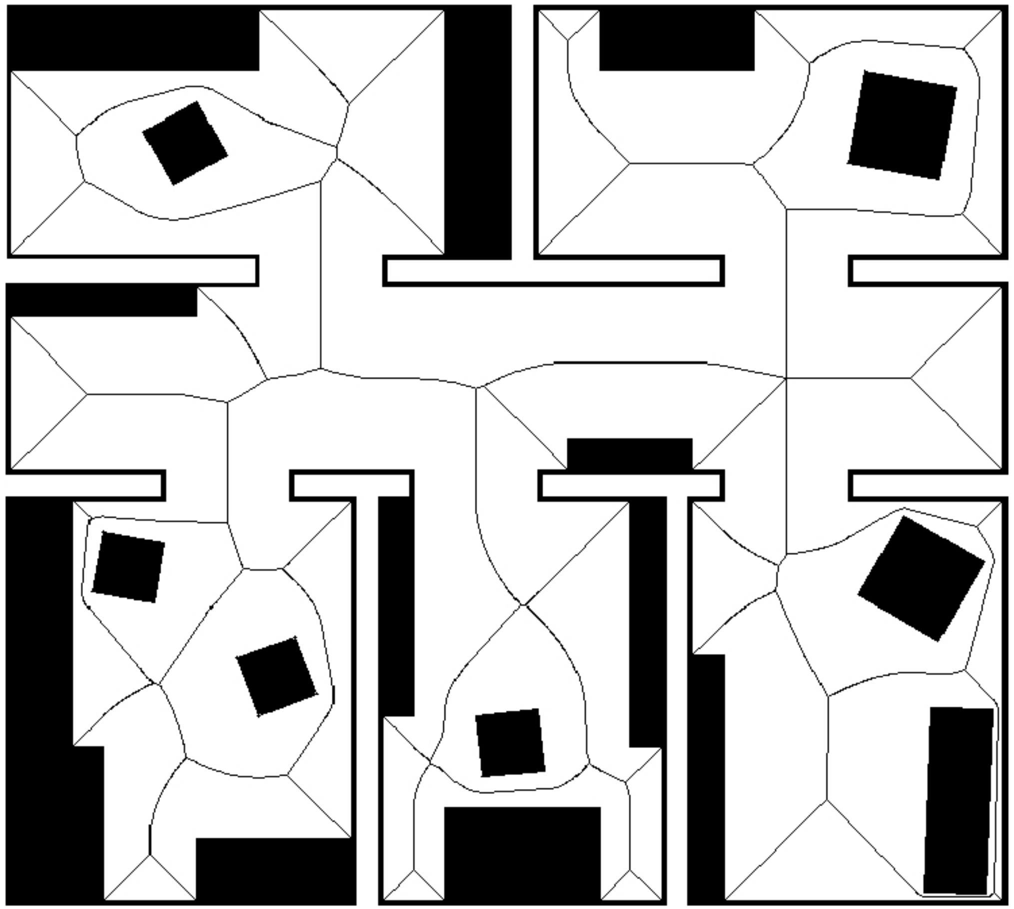
… voronoi diagrams

Voronoi Diagram

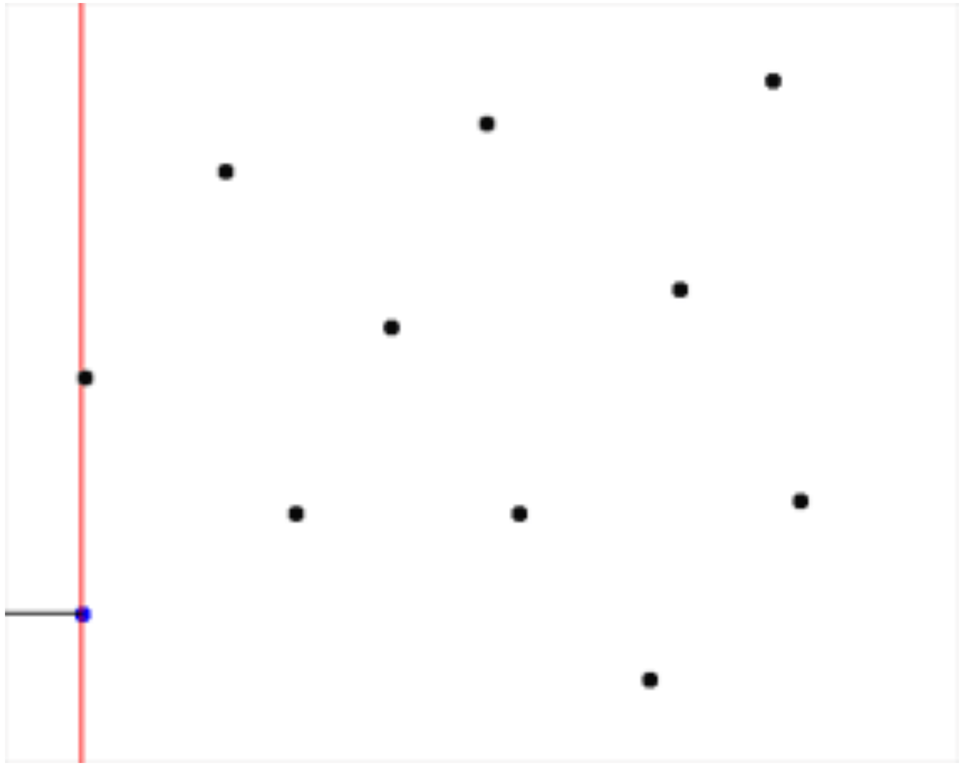
Voronoi Diagram: in nature




Voronoi Diagram: for robots



Voronoi Diagram: how to compute?

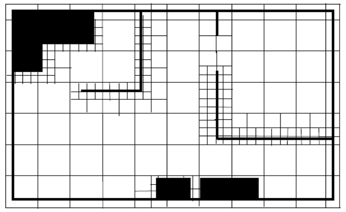


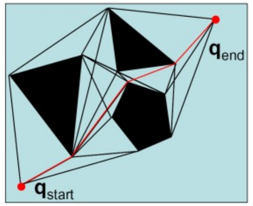
Fortune's Algorithm

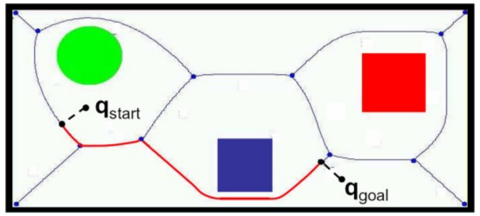
Voronoi Diagrams: issues

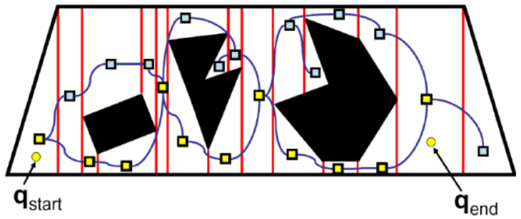
- Difficult to compute in higher dimensions
- Staying away from obstacles is not necessarily the best heuristic
  - Can lead to paths that are much too conservative
  - Can be unstable -- small changes in obstacle configuration can lead to large changes in the diagram

Common Underlying Structure for Search









All of these approaches result in a graph structure that we need to search...

Can be solved with traditional search methods:

Informed Search	Uninformed Search
A* Tree Search	Breadth-first Search
Pure Heuristic	Uniform cost Search
A* Graph Search	Depth-first Search
Best First or 'Greedy'	Depth-limited Search

Most common solution is to use A* with an admissible heuristic of straight-line distance to the goal.