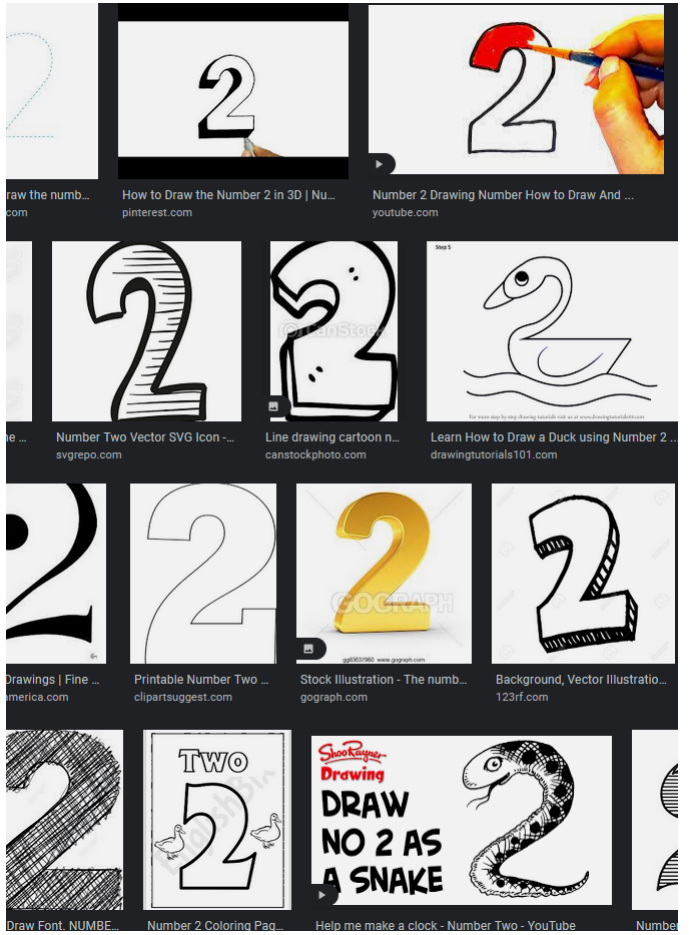
CSCI/ECEN 3302
Introduction to Robotics

Alessandro Roncone

aroncone@colorado.edu https://hiro-group.ronc.one

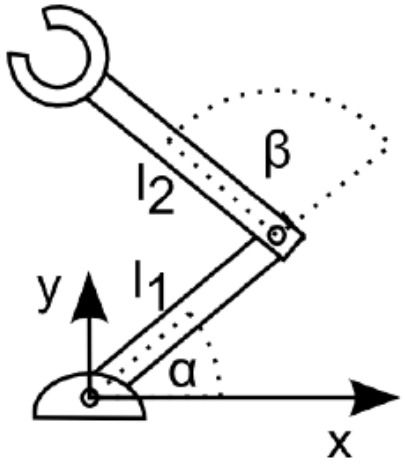
Administrivia

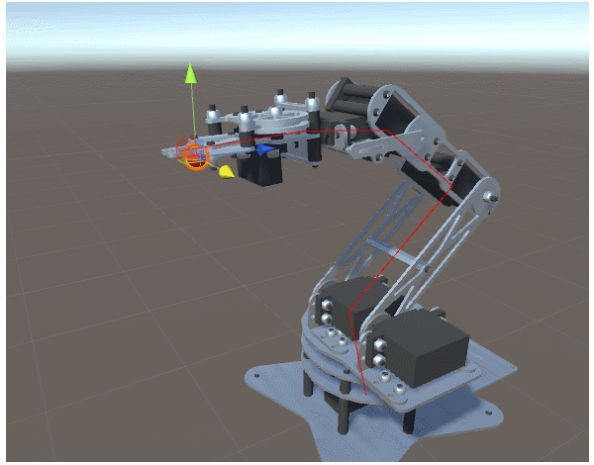
- Happy TwosDay! Tuesday, 2/2/22
- Lab 3
  - Deadline tonight!
  - See mega-thread on Piazza https://piazza.com/class/ky8w23o9yhn4yb?cid=75
  - World falling apart: https://piazza.com/class/ky8w23o9yhn4yb?cid=35
- Lab 4
  - Starts tomorrow
  - One week long
- Lab 5
  - Three weeks long
  - To be concluded before the Spring Break



Chapters 12-13

RECAP
Motion and Trajectory Planning





Motion Planning

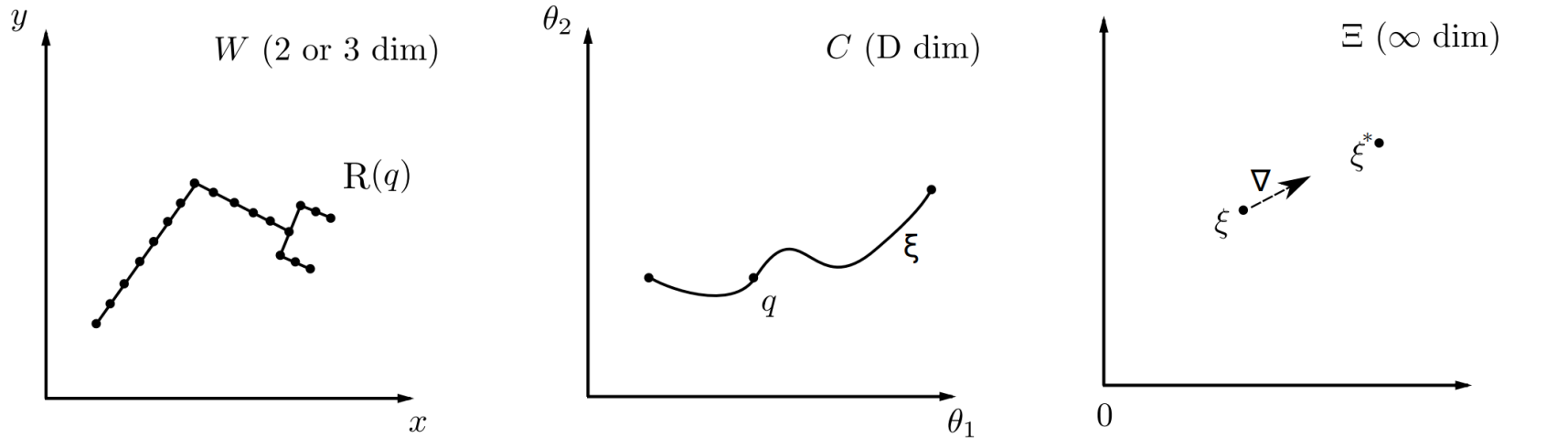
$\xi \rightarrow$ "xi"
$\Xi \rightarrow$ capital "xi"
Both pronounced "ksee"

- **Goal: Find a trajectory in configuration space from one point to another**
  - A trajectory is usually denoted as $\xi: t \in [0, T] \rightarrow C$
  - *(A function mapping time to robot configurations)*
  - The set of all trajectories is Ξ

Trajectory: execution of path r over time $r(t)$

- The Bad News:
  - All complete search algorithms scale exponentially!!
  - Motion planning problems are high dimensional, e.g., big exponent.

Problem Specification: Spaces



Operational/World Space W($\mathbb{R}^3 or\ \mathbb{R}^6$)	Configuration/Joint Space C($\#DoF$)	Trajectory Space $\Xi(\infty\ dim)$

Robot pose in World Space (set of points) → Single point in Configuration Space

Trajectory through Configuration Space (set of points) → Single point in Trajectory Space

Trajectory Optimization

Set of possible trajectories

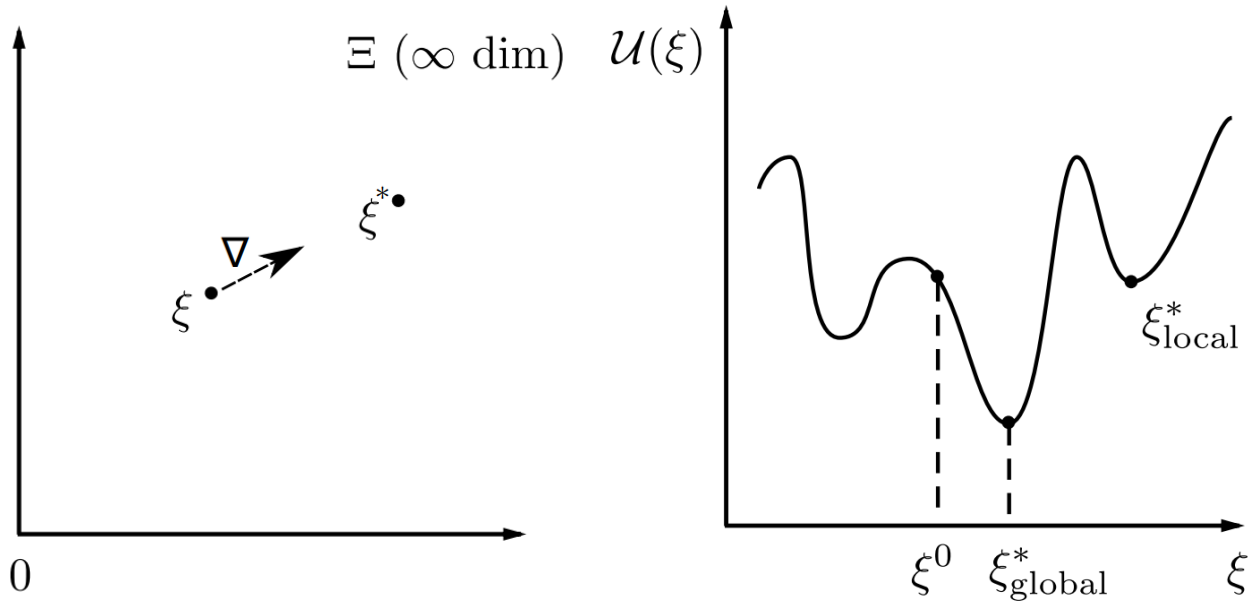
Problem Statement

- Trajectory $\xi: t \in [0, T] \rightarrow C$ Maps time to configurations
- Objective Function $U: \Xi \rightarrow \mathbb{R}^+$ Maps trajectories to scalars

- The objective U encodes traits we want our path to have
  - Path length
  - Efficiency
  - Obstacle avoidance
  - Legibility
  - Uncertainty reduction
  - Human comfort
- Goal: Leverage the benefits of randomized sampling with asymptotic optimality

Our goal: Use a feedback controller that minimizes path length for our low dimensional problem

Problem Specification: Optimization



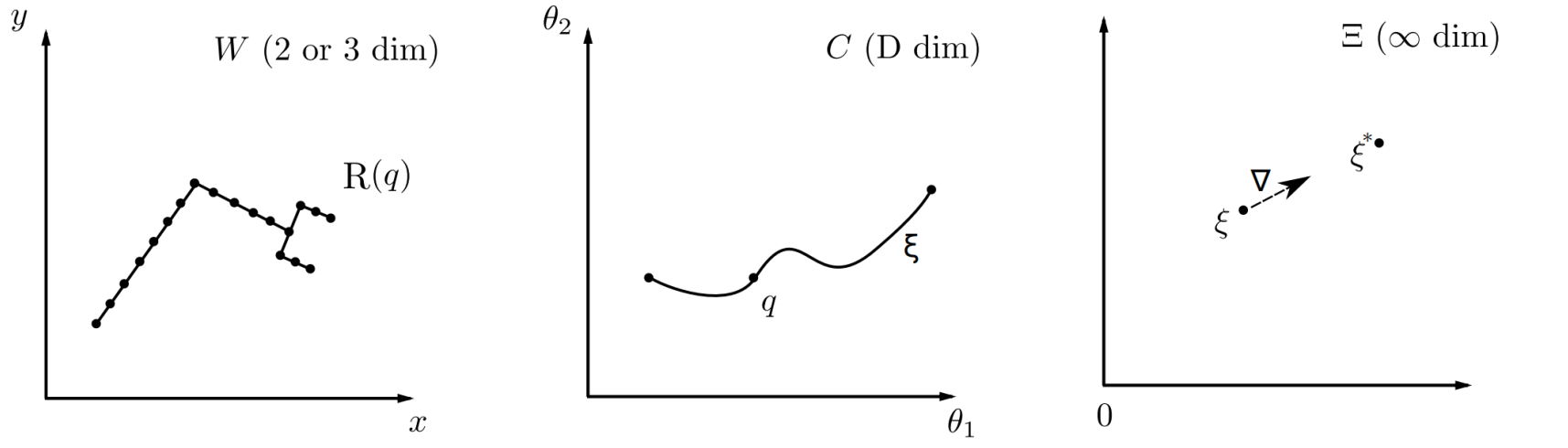
Points in the direction of greatest ascent for ξ

Gradient Descent is a popular technique for finding local minima

$$\xi_{t+1} = \xi_t - \frac{1}{\alpha} \nabla_\xi U(\xi_t)$$

Questions?

Problem Specification: Spaces

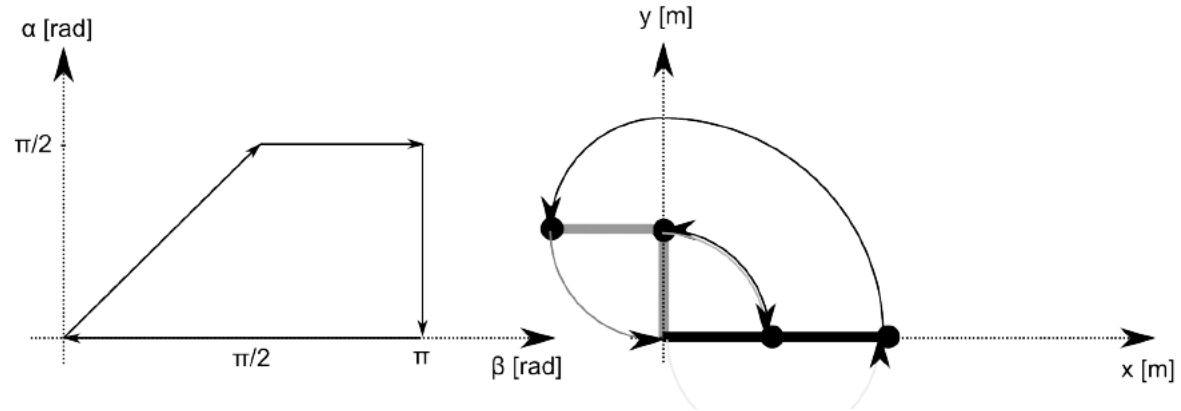


Operational/World Space W($\mathbb{R}^3 or\ \mathbb{R}^6$)	Configuration/Joint Space C($\#DoF$)	Trajectory Space $\Xi(\infty\ dim)$

Robot pose in World Space (set of points) → Single point in Configuration Space

Trajectory through Configuration Space (set of points) → Single point in Trajectory Space

Configuration Space



- Allows us to reduce robot positions to a single point
  - Very convenient for planning!
- One axis of configuration space for each degree of freedom of the robot
  - Convenient for planning as the space is the same as what the robot controls.
  - Removes the need to figure out how to get to a point in (X, Y)-space
- We still need to figure out how to put obstacles into configuration space!

Configuration Space

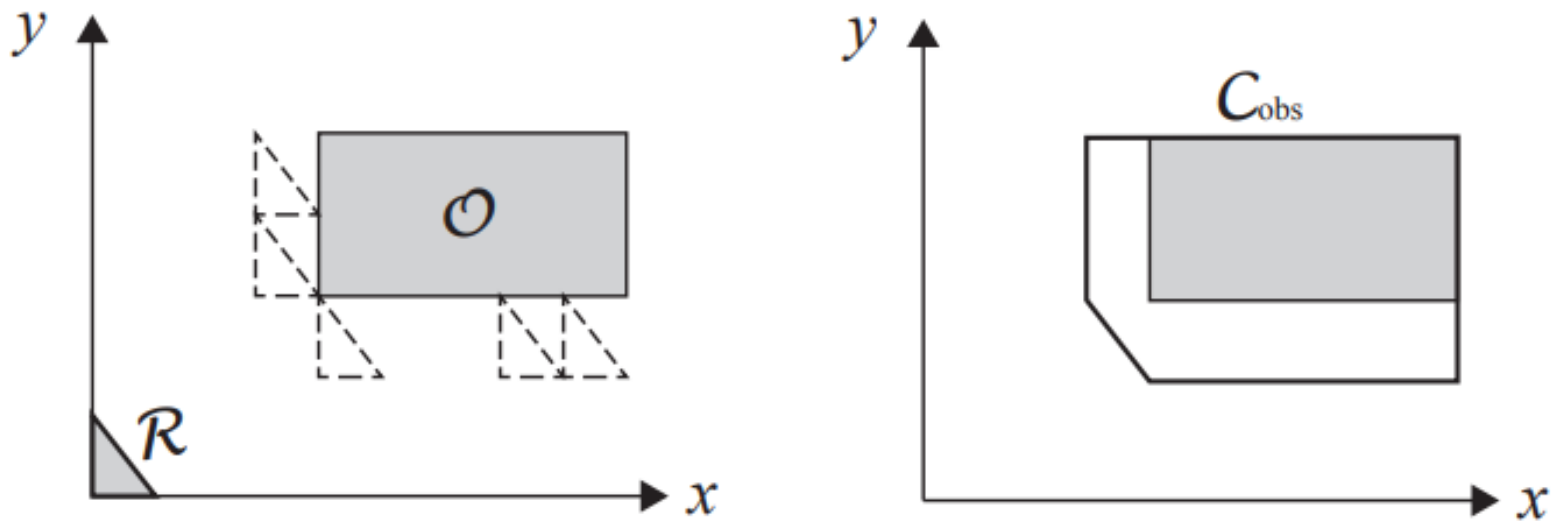
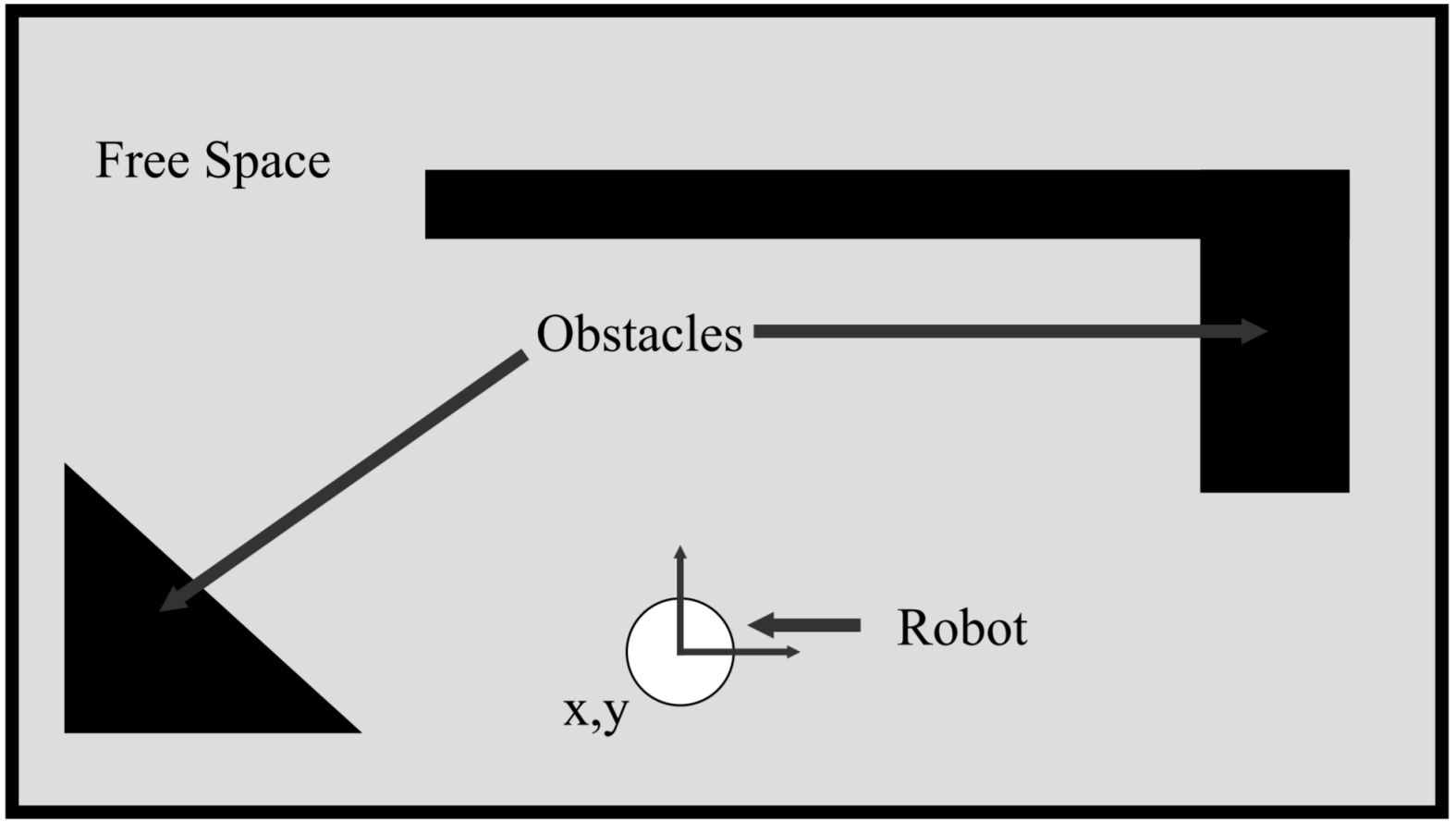


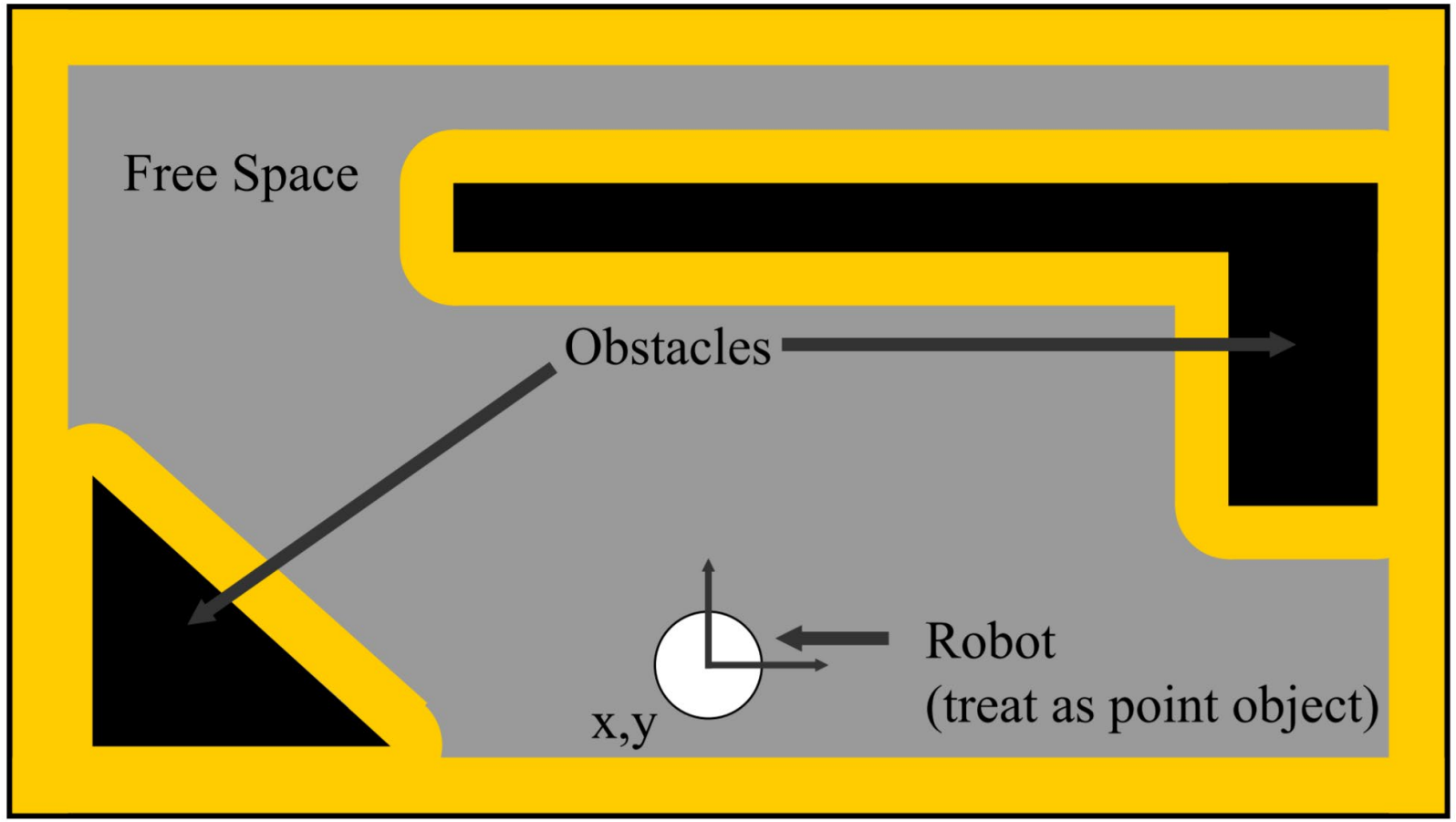
Figure 2.4: C_{obs} for a robot R that translates in x-y with a rectangle obstacle O

Example of taking a robot that can translate across (X, Y) and an obstacle and putting them in configuration space.

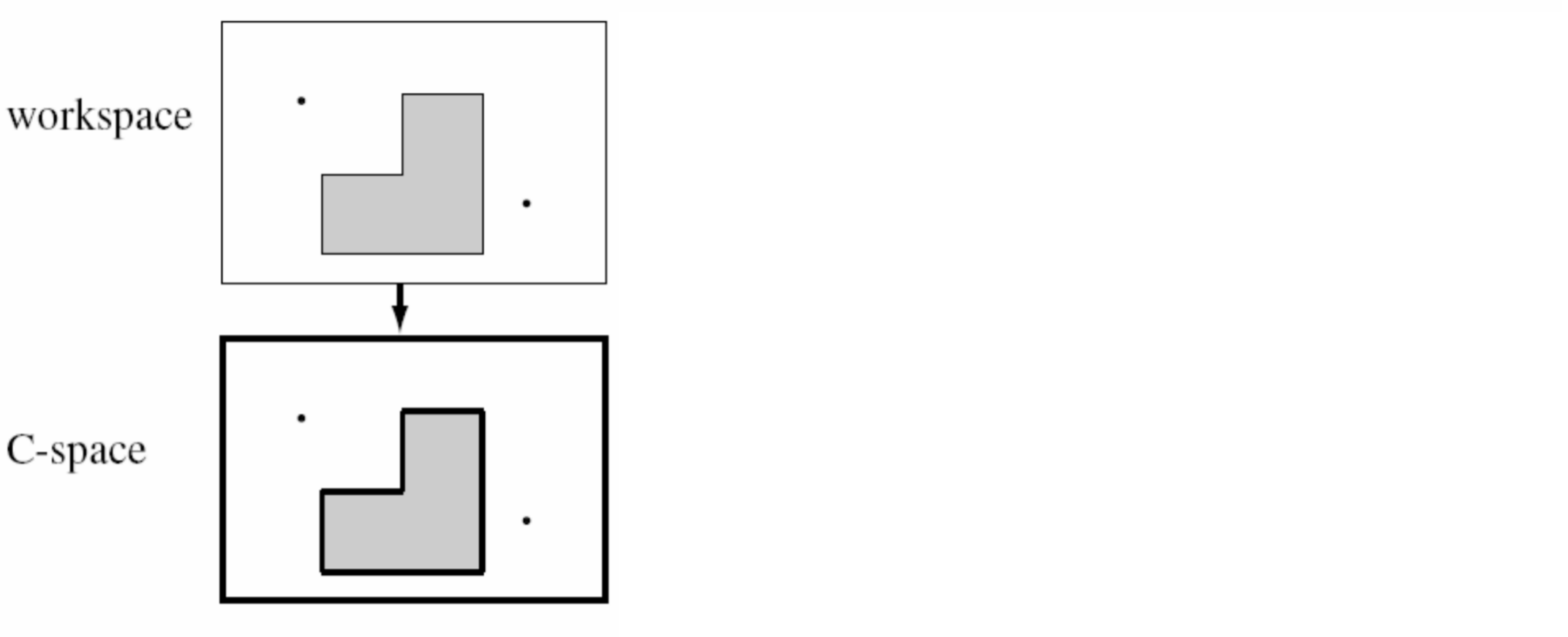
Configuration Space



Configuration Space

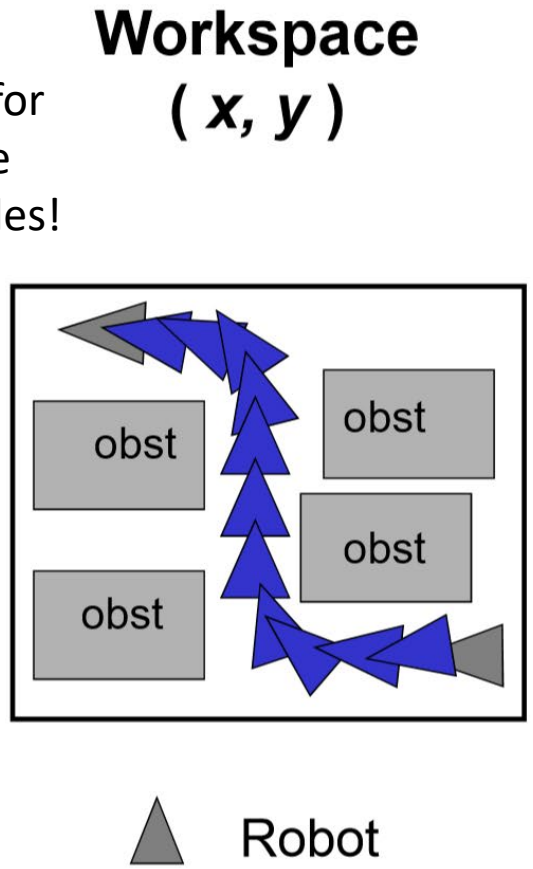


Configuration Space



Configuration Space

Each step has to check for collisions between the robot's body and obstacles!



The robot's path is just a path through space – as a single point, collision checks are easy!

Chapter 4

Module 2 – COMPUTATION

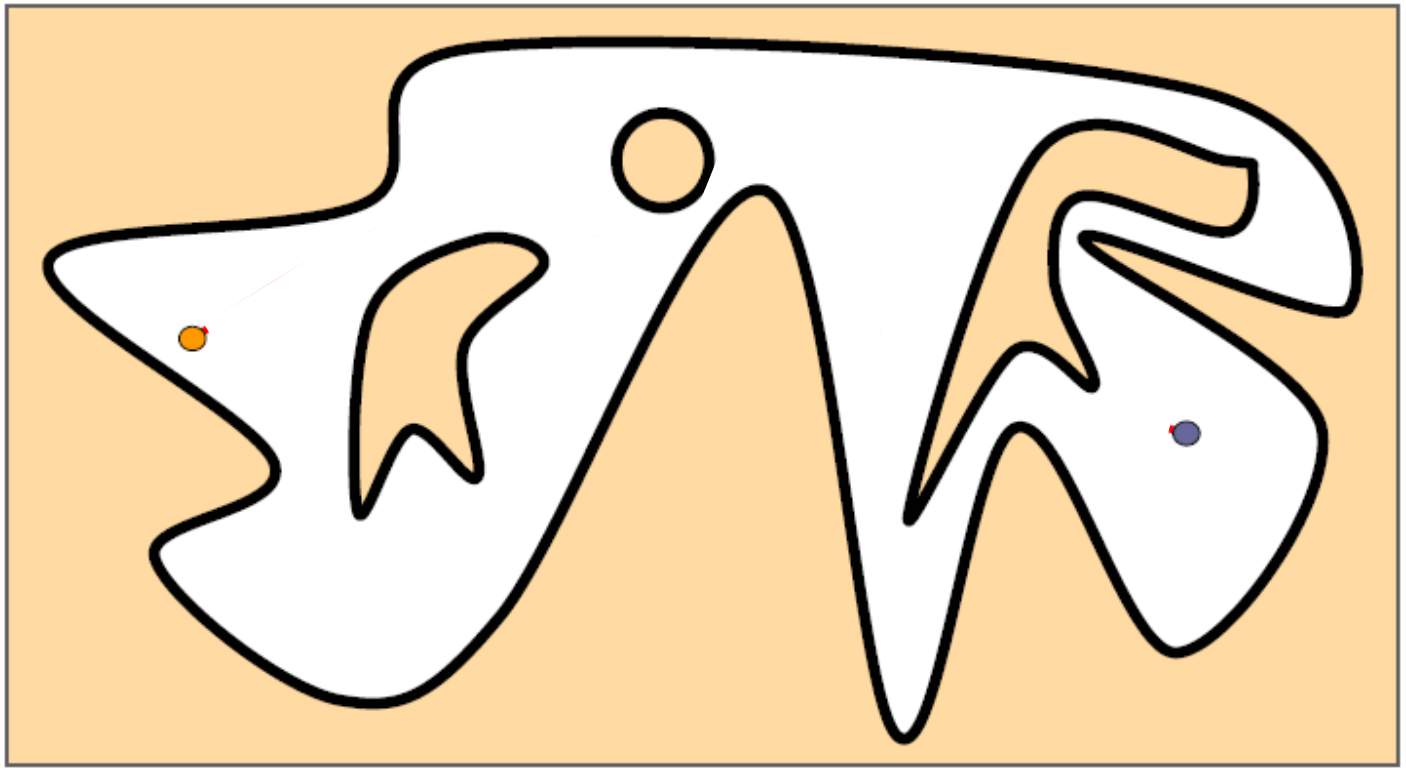
Part II –World representations

Objective: Compute a collision-free path for a mobile robot among static obstacles

- Inputs required
  - Geometry of the robot, any obstacles in the environment
  - Kinematics of the robot (DoF, Joint links and types)
  - Initial and goal robot configurations (positions & orientations)

- Expected Result
  - Continuous sequence of collision-free robot configurations
  - The sequence connects the initial and goal configurations
  - We can follow this sequence of configurations to traverse the path

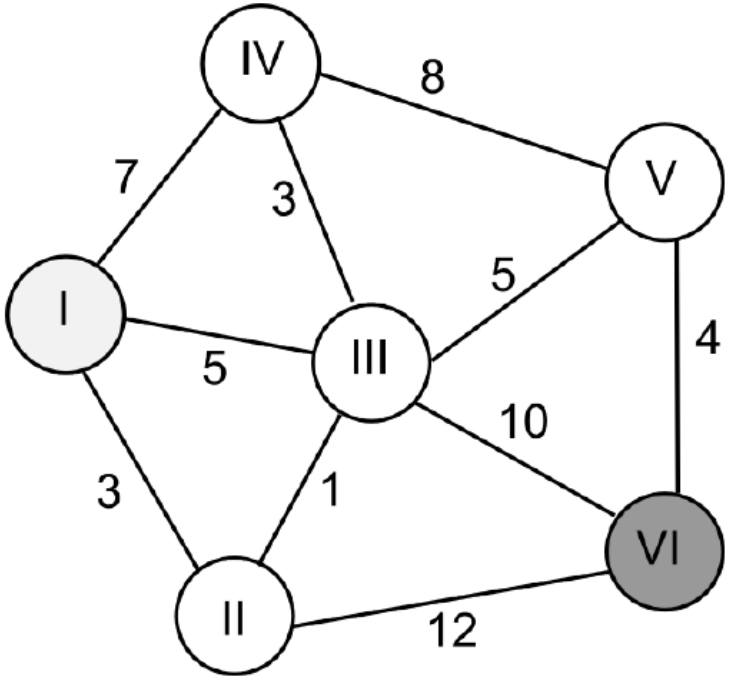
Moving from Start to Goal state



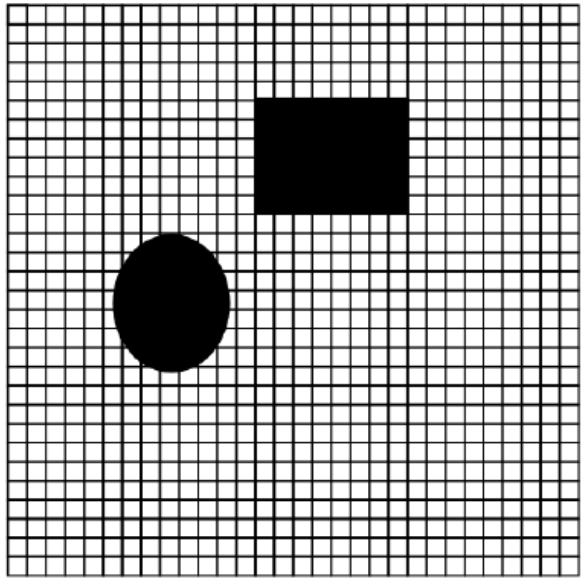
- General approach: Reduce what amounts to an intractable problem in continuous C-space to a tractable problem in a discrete space.
- Tools:
  - Environment Representations
  - Search Algorithms

Planning across length scales

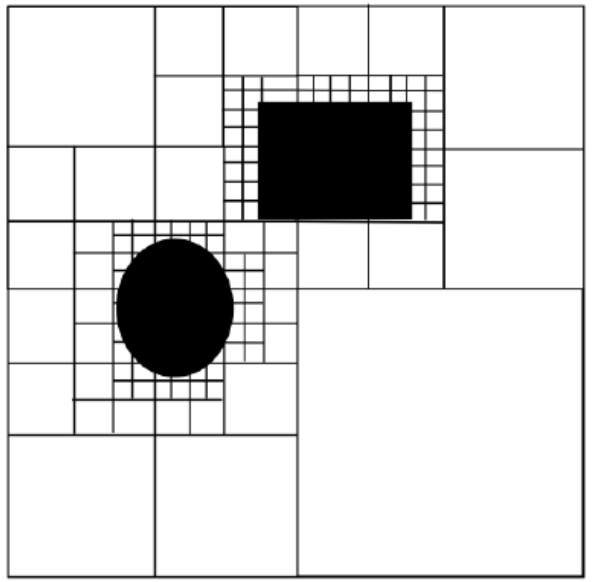
Map Representations



Topological Map
(Graph)



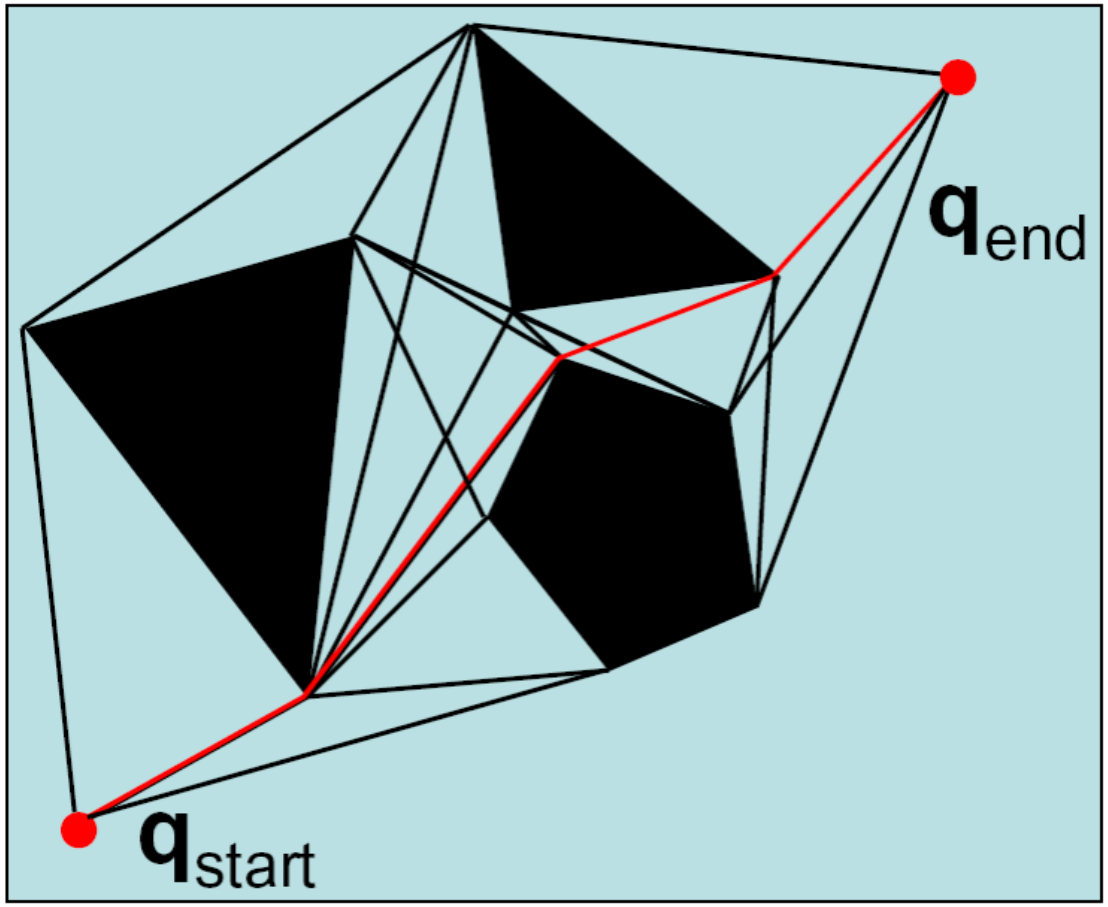
Grid Map
(Discrete Coordinates)



K-d Tree
(Quadtree)

Some Well-Known Representations

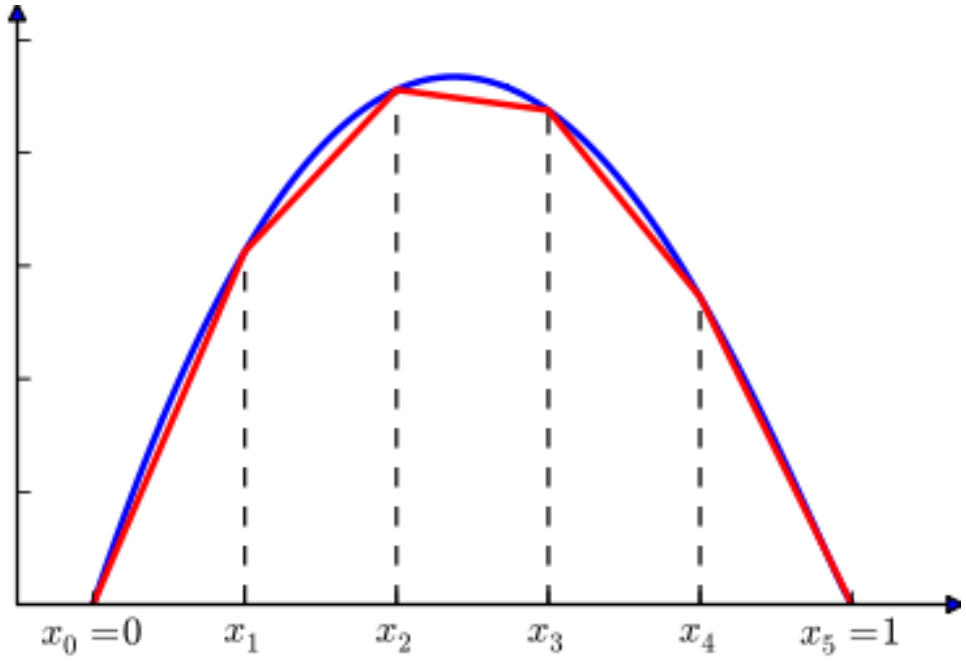
- Visibility Graphs
- Roadmap
- Cell Decomposition
- Potential Field



Visibility Graph Method

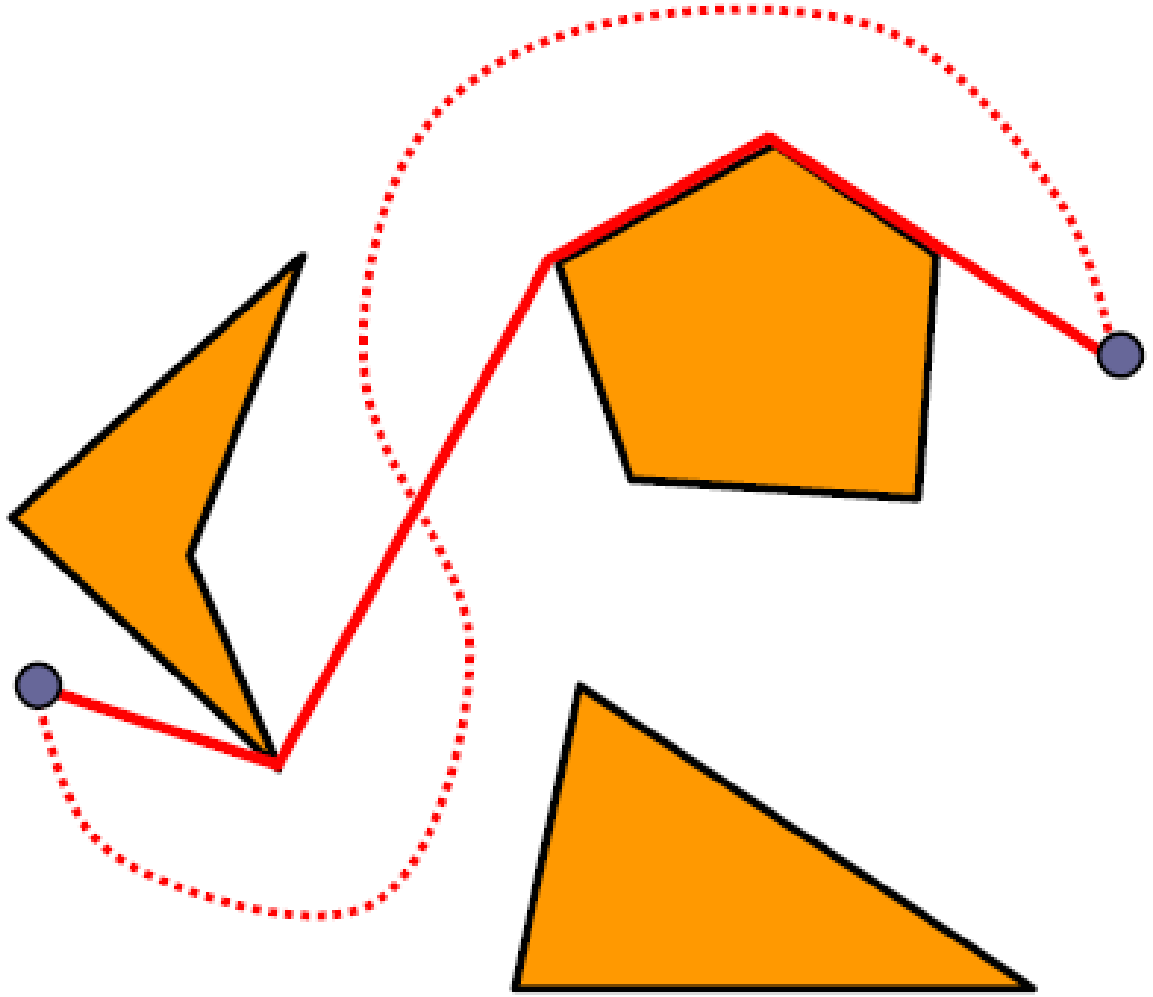
- If there is a collision-free path between two points, then there is a polygonal path that bends only at the obstacles vertices.

- A polygonal path is a piecewise linear curve:



Visibility Graph Method

- Solid path: Visibility Graph motion planning solution
- Dotted Path: Voronoi Roadmap planning solution



Visibility Graph

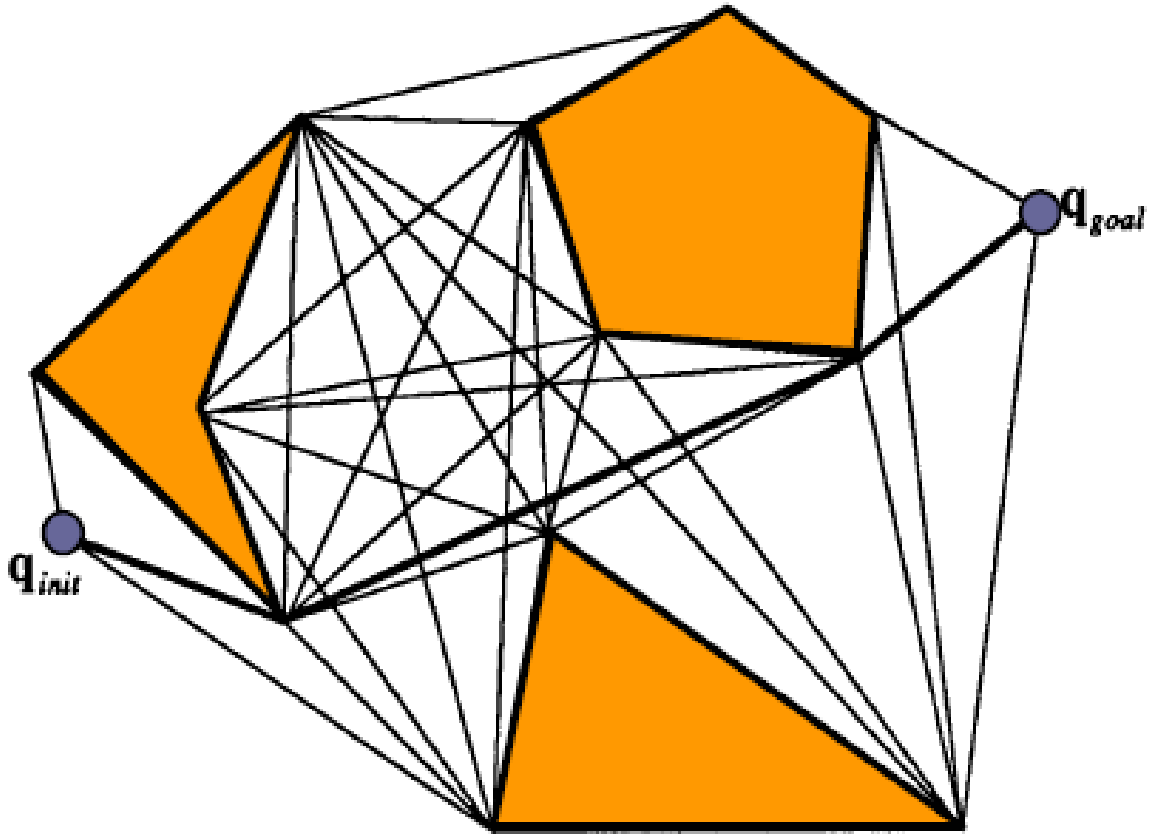
- A visibility graph is a graph such that
  - Nodes: q_{init}, q_{goal}, or an obstacle vertex.
  - Edges: An edge exists between nodes u and v if the line segment between u and v is an obstacle edge or it does not intersect the obstacles.

Visibility Graph Algorithm

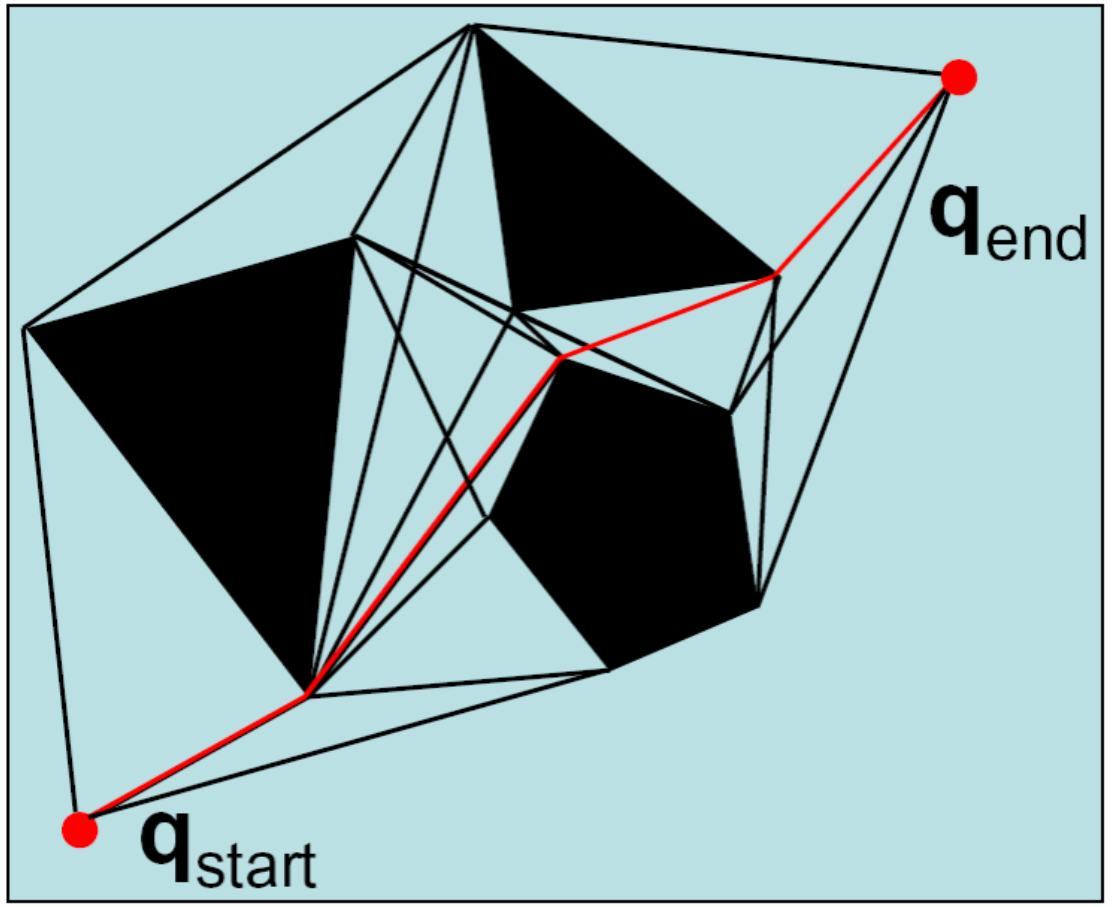
Input: q_{init}, q_{goal}, polygonal obstacles

Output: visibility graph G

```
1.    for every pair of nodes u,v
2.      if segment (u,v) is an obstacle edge then
3.        insert edge (u,v) into G;
4.      else
5.        for every obstacle edge e
6.          if segment (u,v) intersects e
7.            go to (1);
8.        insert edge (u,v) into G.
```

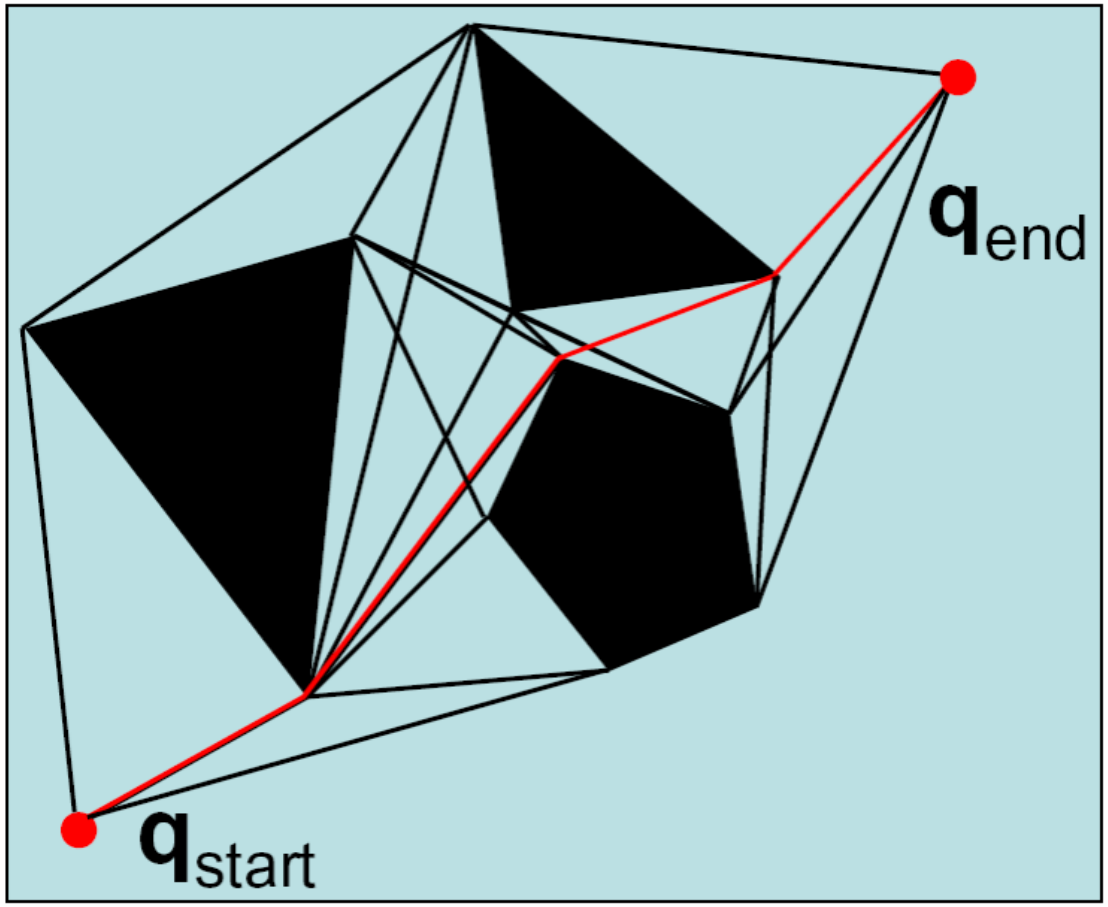


Visibility Graph: Strengths/Weaknesses?

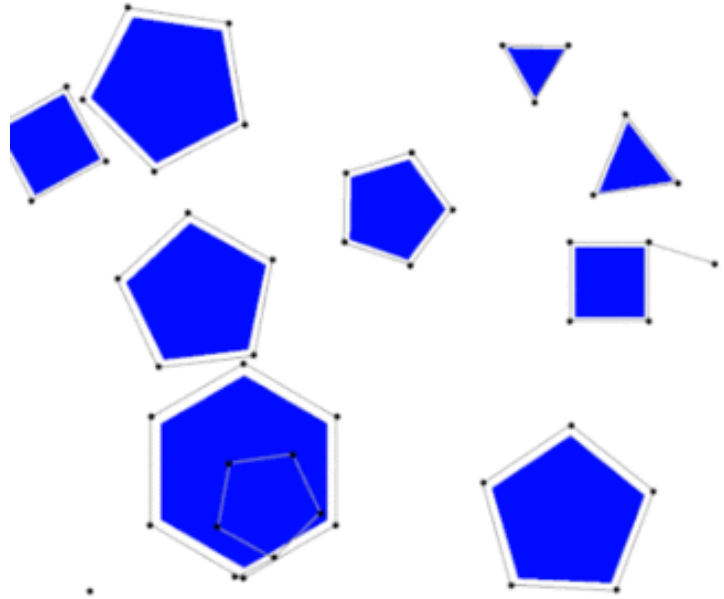


Some Well-Known Representations

- Visibility Graphs
- Roadmap
- Cell Decomposition
- Potential Field



Road mapping

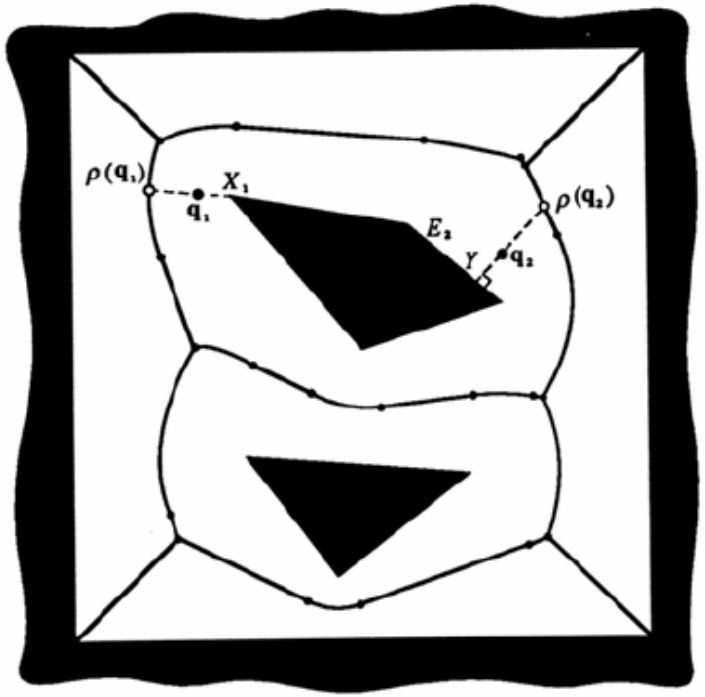


Probabilistic Road Mapping

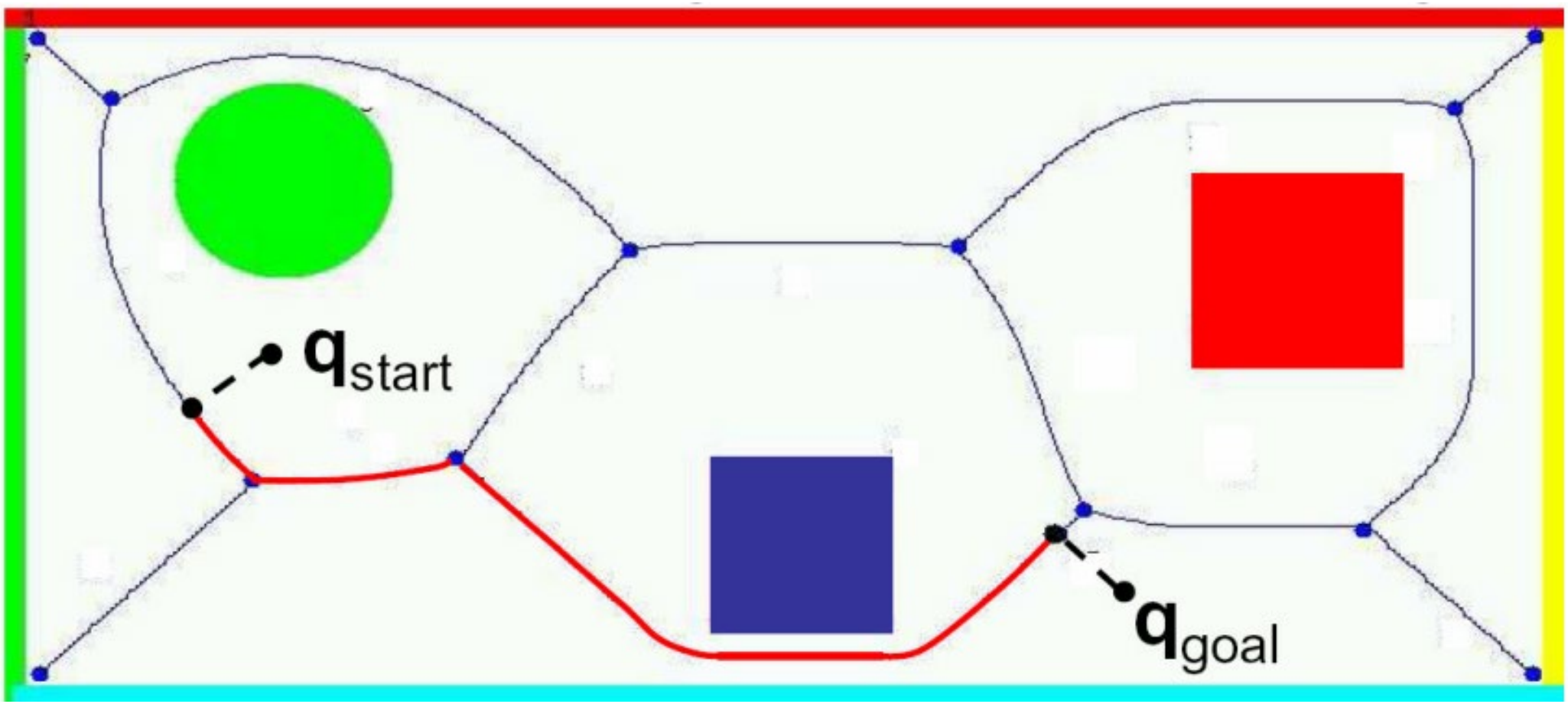
- Take random samples, test them to see they are in free space, and connect them via nearest neighbor
- Use a graph search algorithm to find the optimal path

Voronoi Road Mapping

- Introduced by computational geometry researchers.
- Generate paths that maximize clearance
- Applicable mostly to 2-D configuration spaces

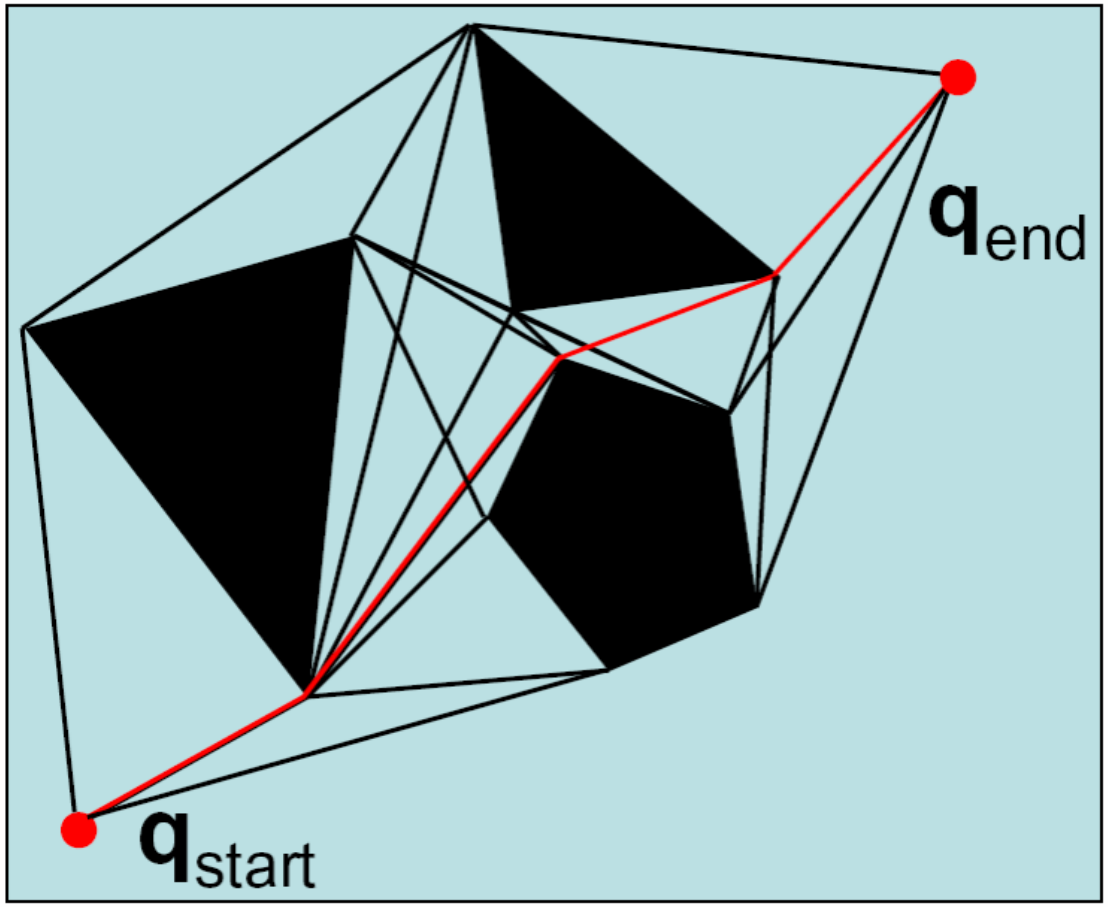


Voronoi Road Mapping: Strengths / Weaknesses

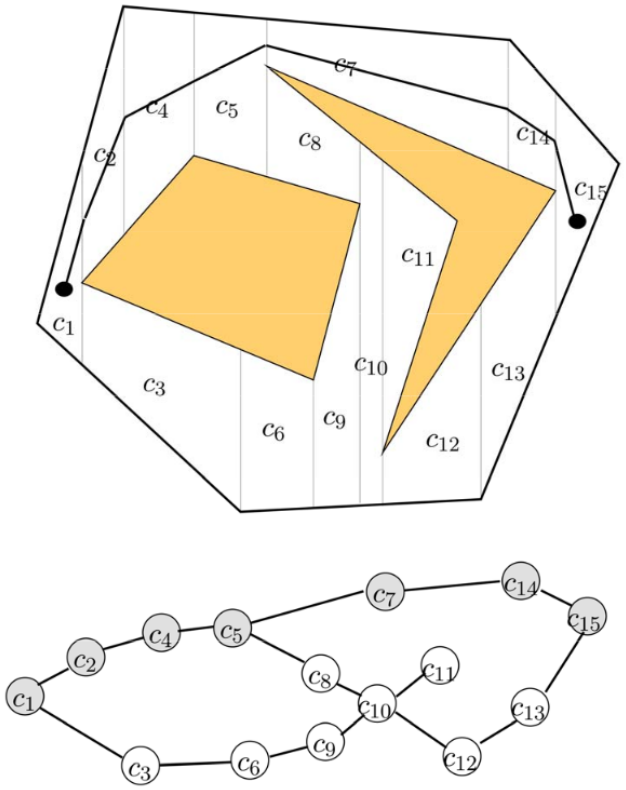


Some Well-Known Representations

- Visibility Graphs
- Roadmap
- Cell Decomposition
- Potential Field



Cell Decomposition



Exact cell decomposition

- Divides a space **F** precisely into sub-units

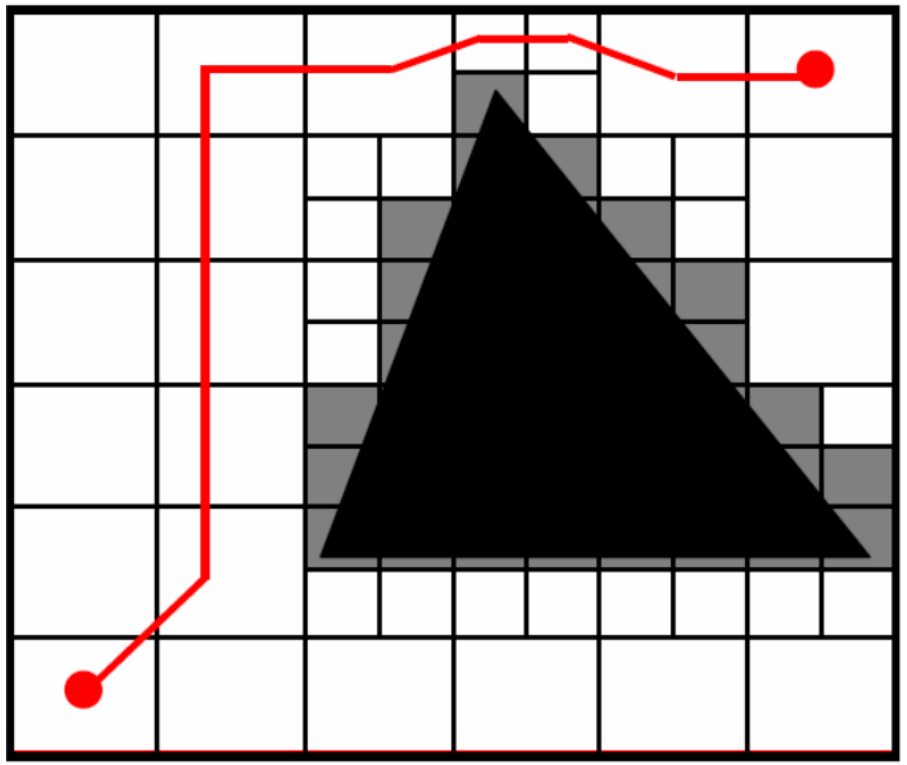
Approximate cell decomposition

- ***F*** is represented by a collection of non-overlapping cells whose union is contained in ***F*.**
- Cells usually have simple, regular shapes, e.g., rectangles, squares.
- Facilitates hierarchical space decomposition

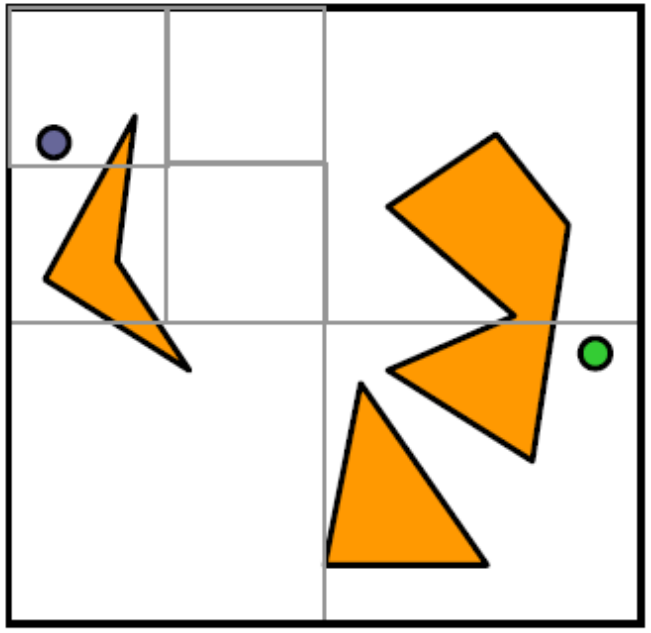
Cell Decomposition

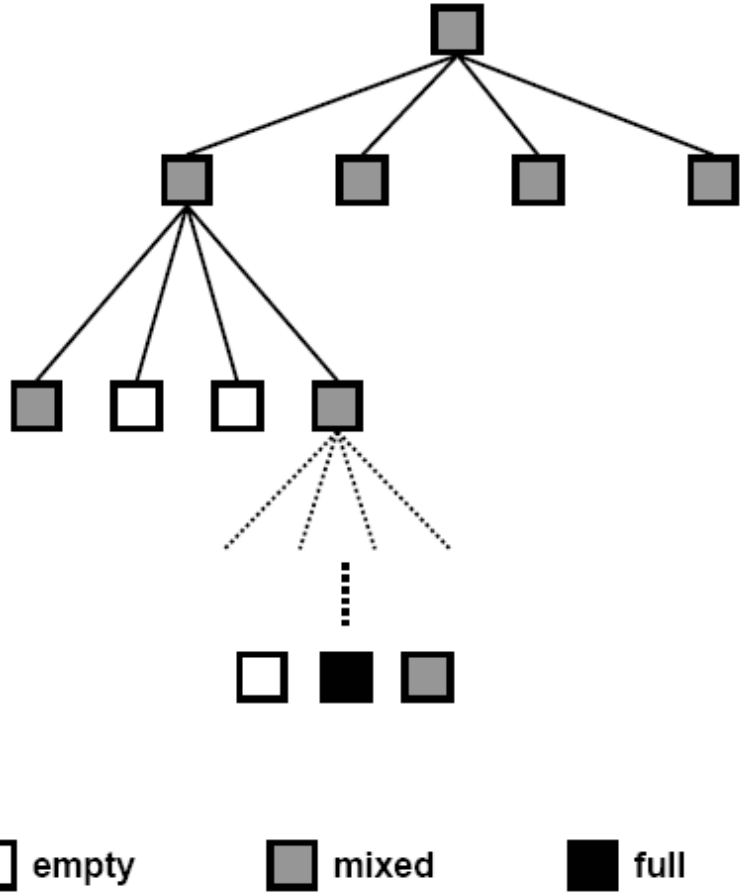
Not necessarily *complete*

(Complete: If a solution exists, it will eventually be found)

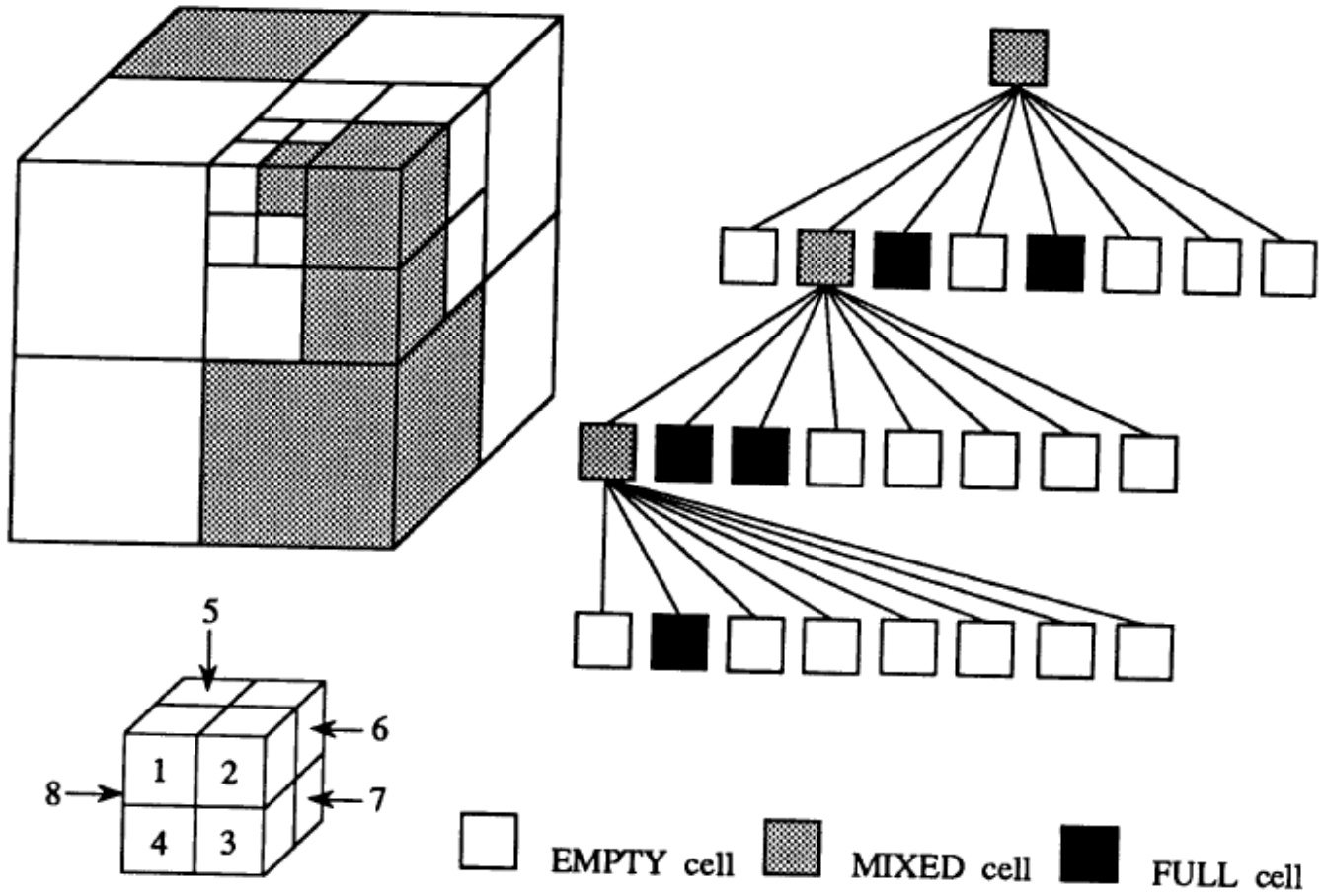


Quadtree Decomposition



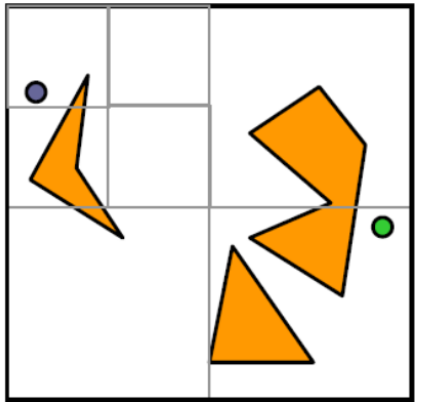


Octree Decomposition

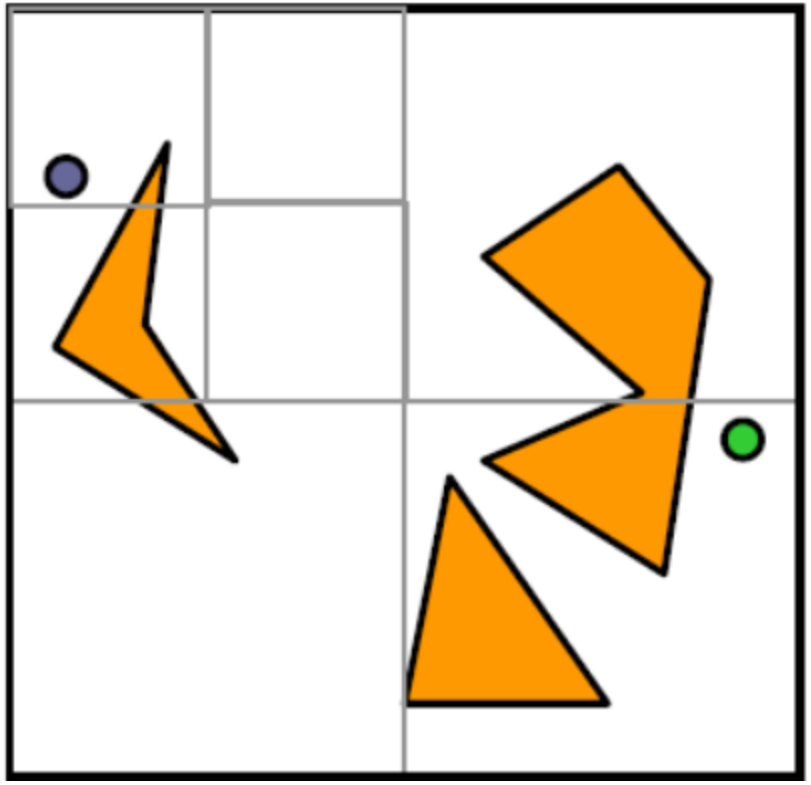


Cell Decomposition Path Planning Algorithm Outline

1. Decompose the free space F into cells.
2. Search for a sequence of **mixed** or **free** cells that connect the initial and goal positions.
3. Further decompose the mixed.
4. Repeat (2) and (3) until a sequence of **free** cells is found.

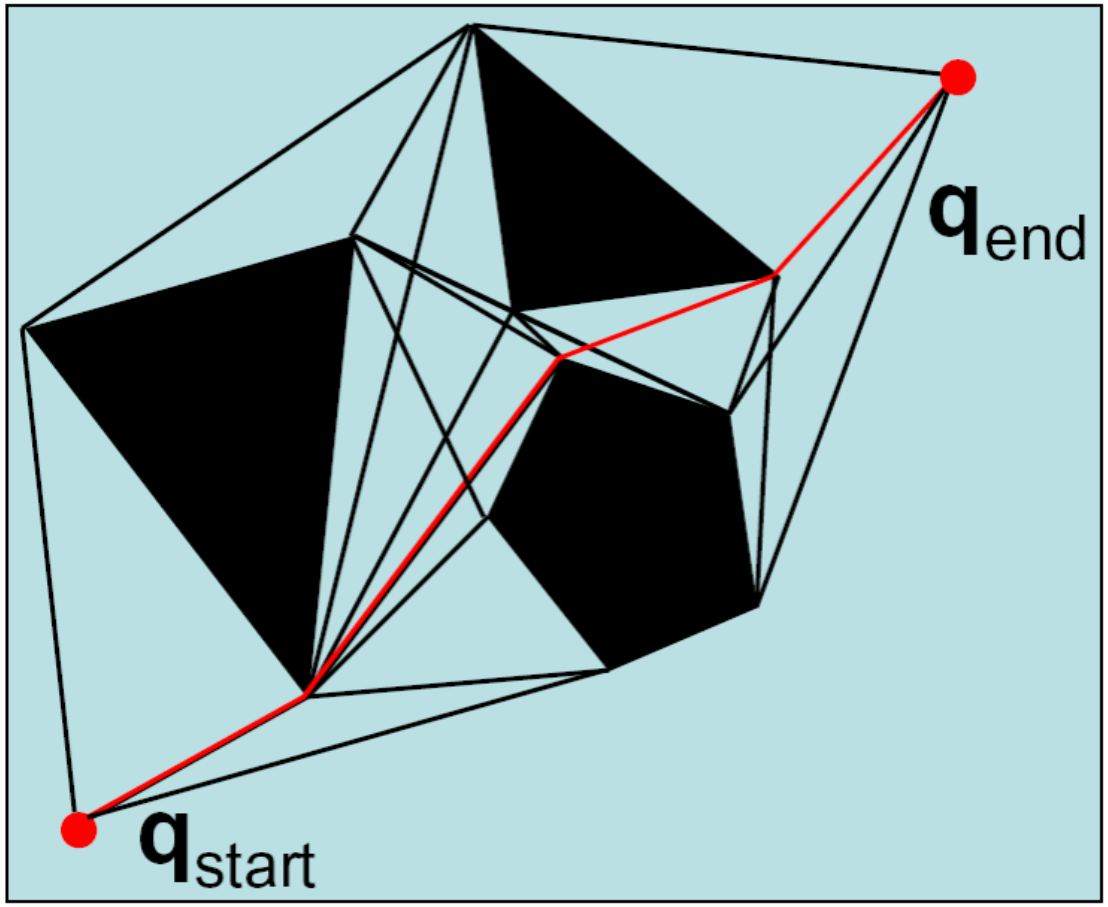


Cell Decomposition: Strengths/Weaknesses



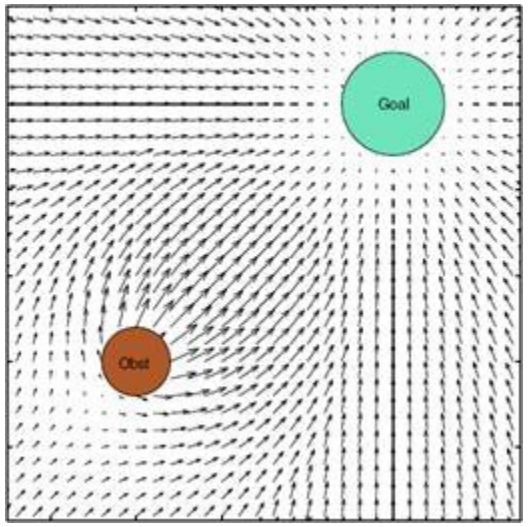
Some Well-Known Representations

- Visibility Graphs
- Roadmap
- Cell Decomposition
- Potential Field

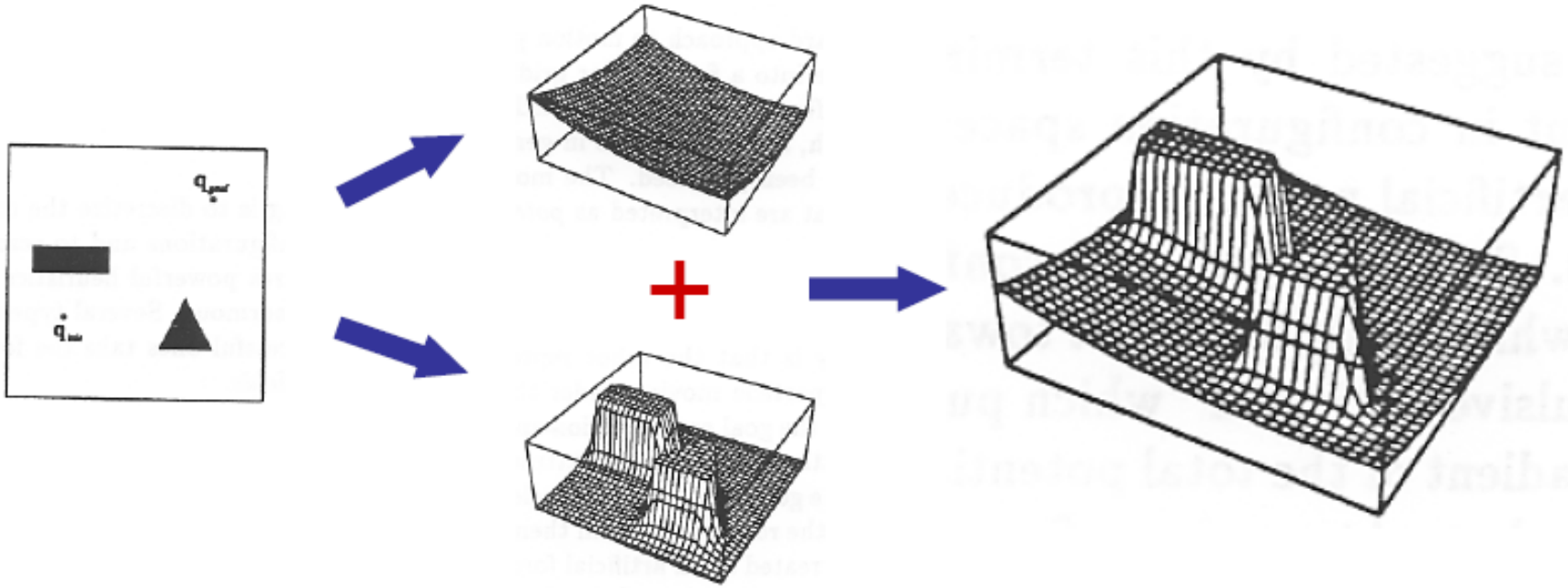


Potential Fields

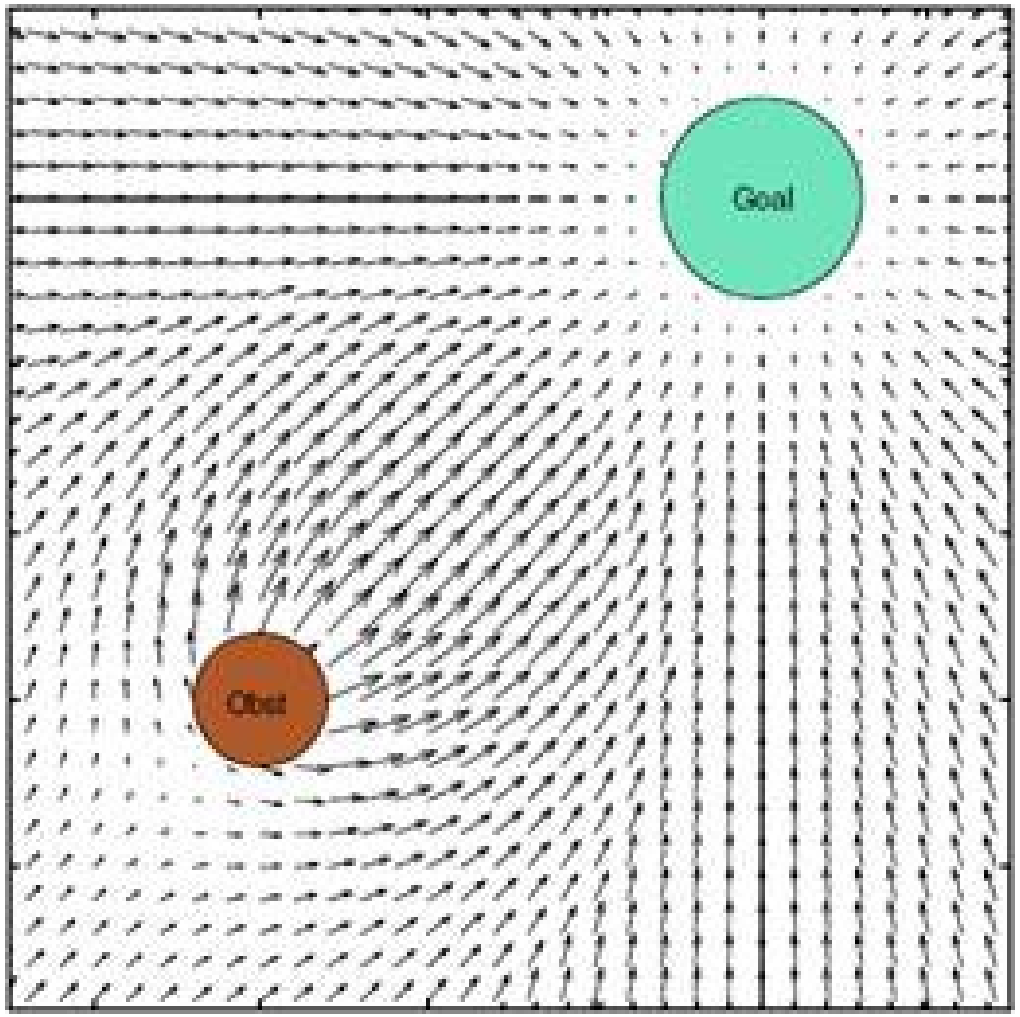
- Initially proposed for real-time collision avoidance [Khatib 1986].
- A potential field is a scalar function over the free space.
- To navigate, the robot applies a force proportional to the negated gradient of the potential field.
- A navigation function is an ideal potential field that
  - has global minimum at the goal
  - has no local minima
  - grows to infinity near obstacles
  - is smooth



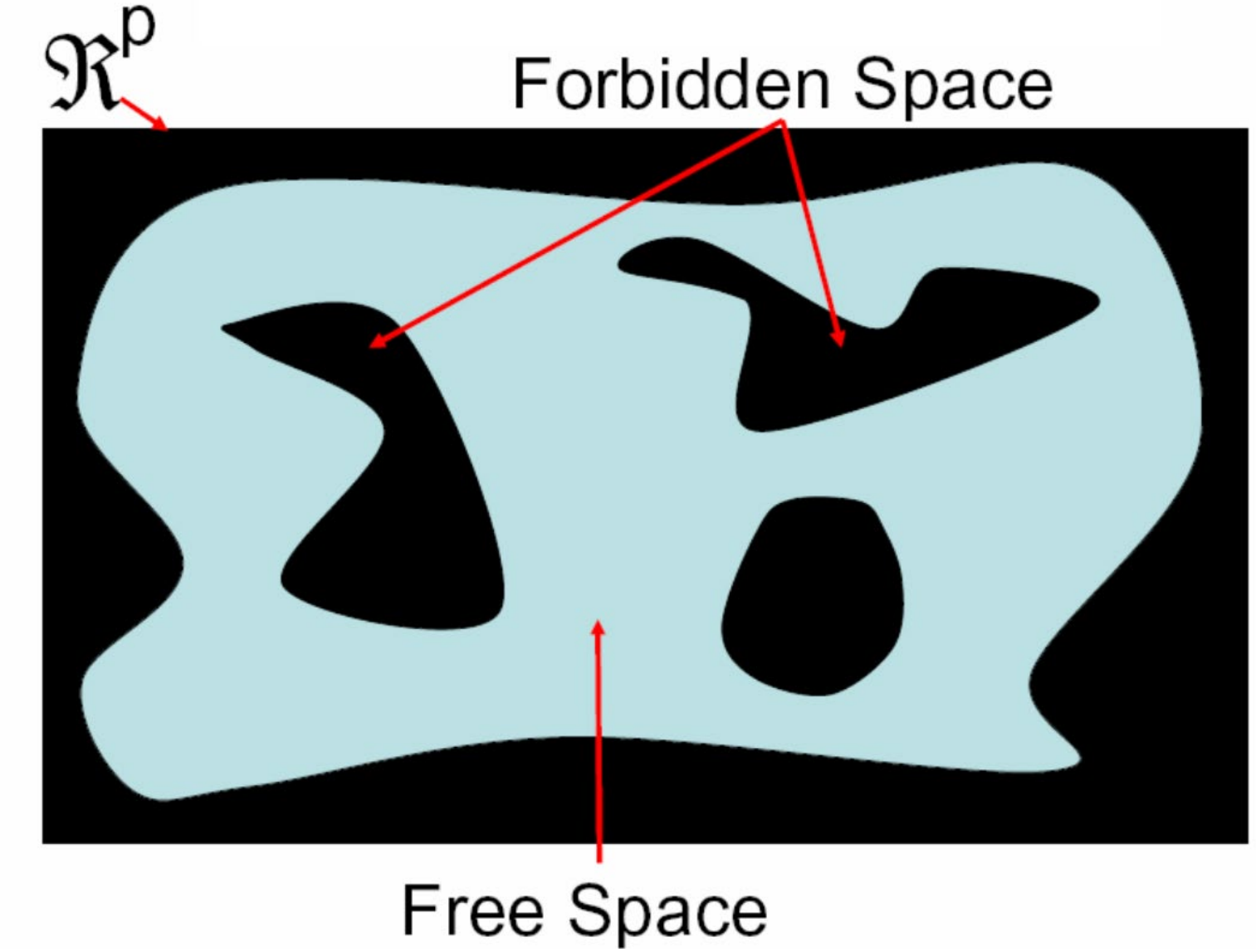
How Does It Work?



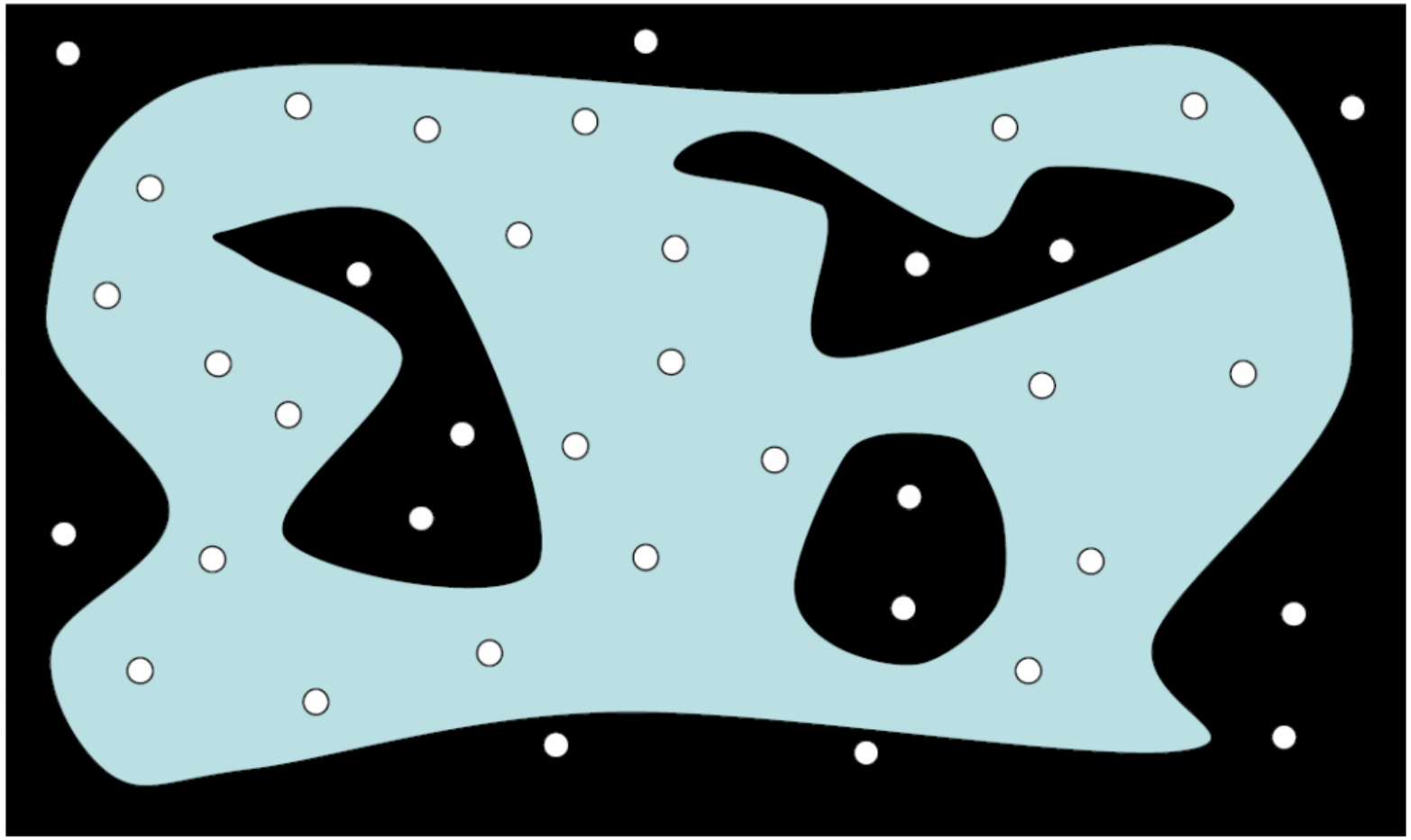
Potential Fields: Strengths/Weaknesses



Probabilistic Road Map (PRM)

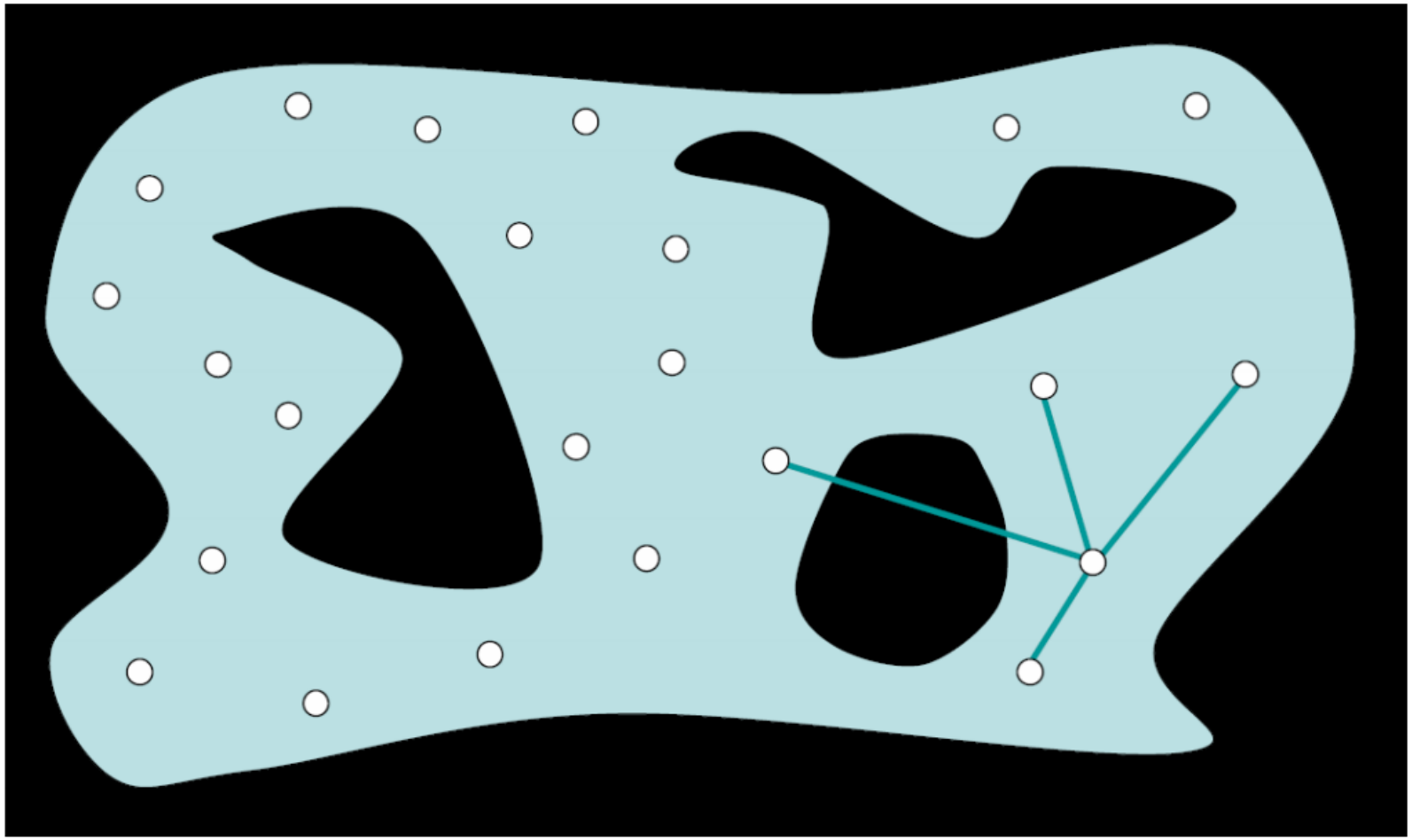


Probabilistic Road Map (PRM)



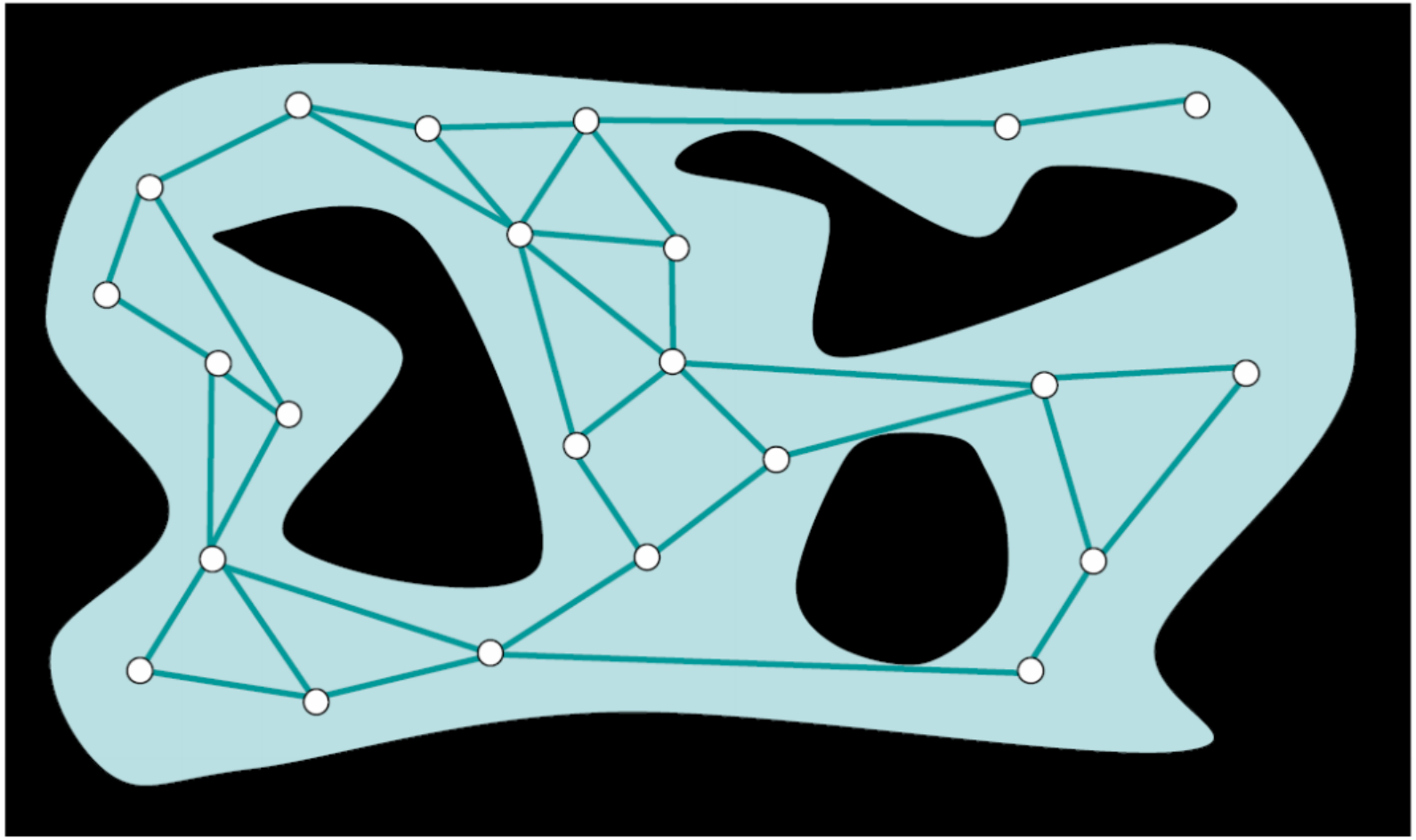
Sample random locations!

Probabilistic Road Map (PRM)



Remove points in forbidden areas
Link each point to its K nearest neighbors

Probabilistic Road Map (PRM)

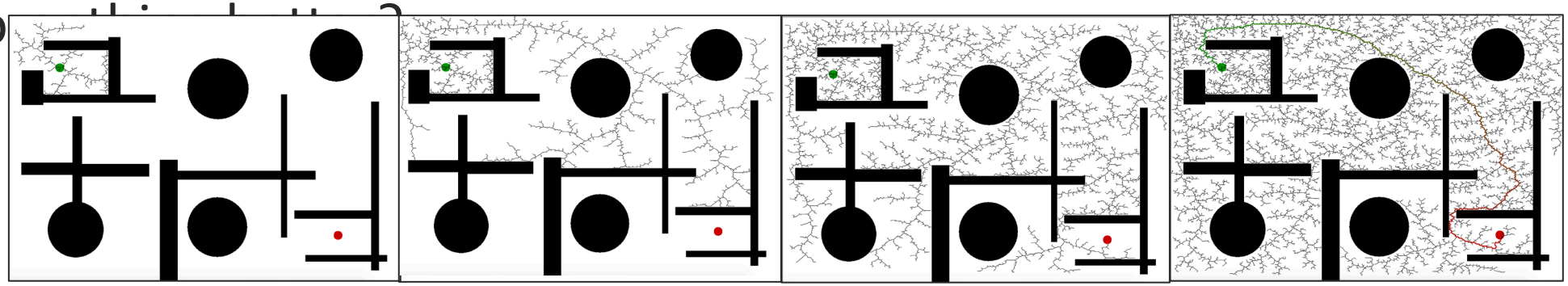


Remove edges crossing forbidden areas

How to sample points?

- Uniformly randomly
- Sample more near places with few neighbors
- Bias samples to exist near obstacles
- Use human-provided waypoints
- So

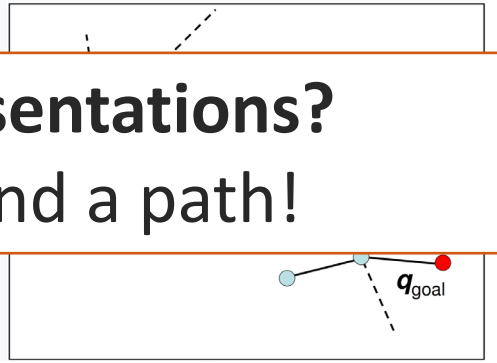
Rapidly-exploring Random Trees (RRT)



Rapidly-exploring Random Trees

```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    G.add_edge(q_near, q_new)
  return G
```



How can we make use of these representations?

Search algorithms provide a way to find a path!

Rapidly Exploring Random Trees [RRT]

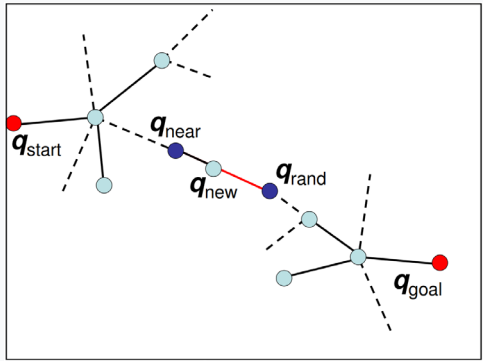
Basic idea:

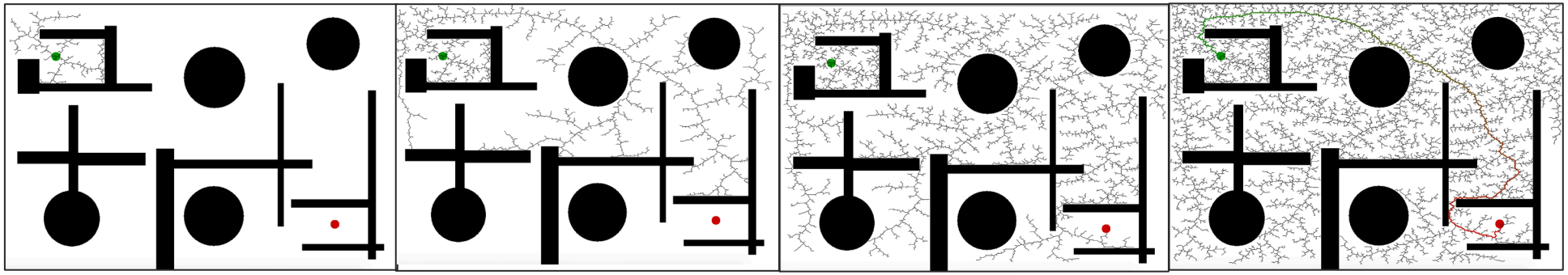
- Build up a tree through generating “next states” in the tree by executing random controls
- However: to ensure good coverage, we need something better than the above

Rapidly-exploring Random Trees

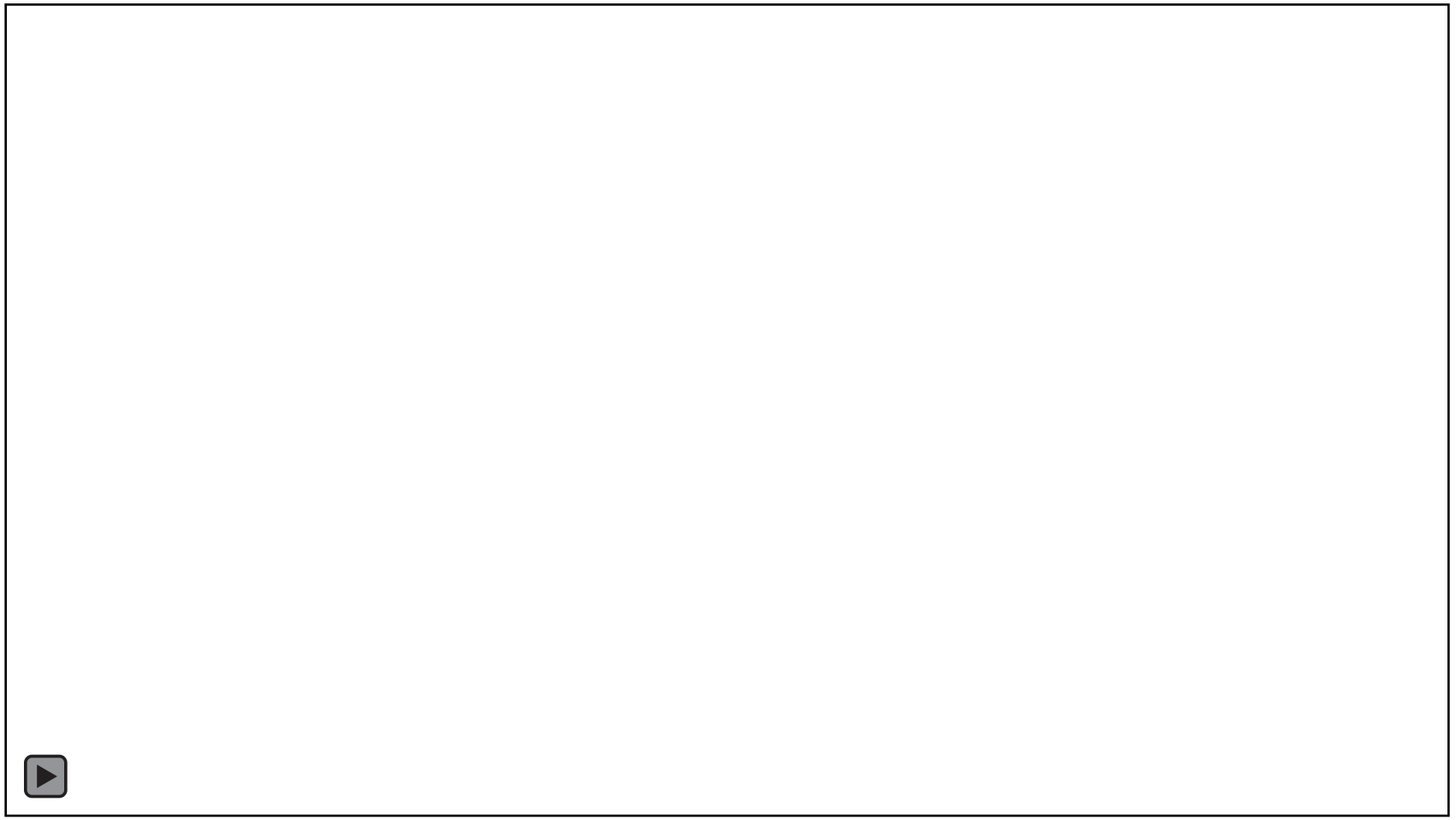
Algorithm BuildRRT
  Input: Initial configuration q_{init}, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_{init})
  for k = 1 **to** K
    q_{rand} ← RAND_CONF()
    q_{near} ← NEAREST_VERTEX(q_{rand}, G)
    q_{new} ← NEW_CONF(q_{near}, q_{rand}, Δq)
    G.add_vertex(q_{new})
    G.add_edge(q_{near}, q_{new})
  return G

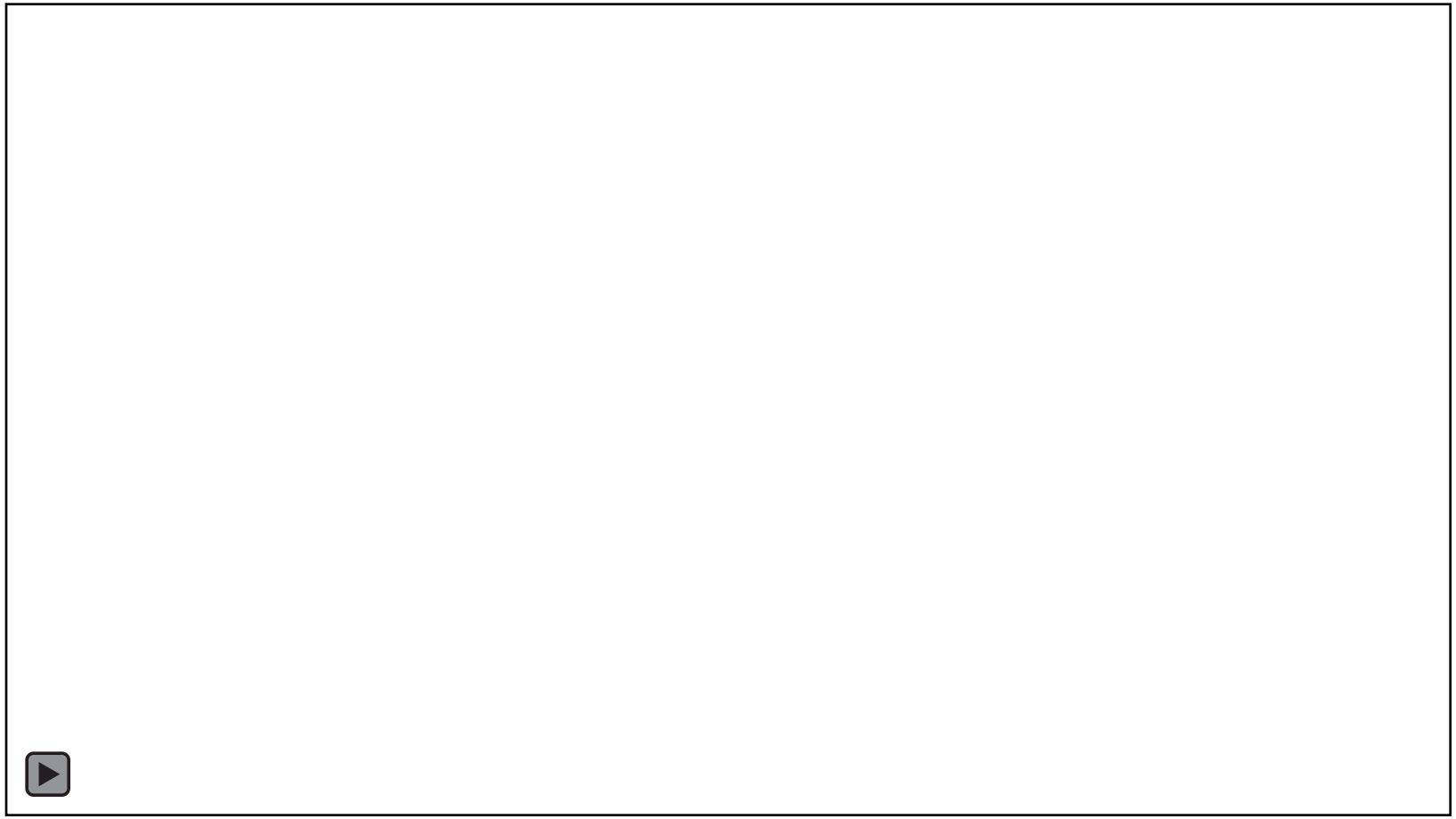




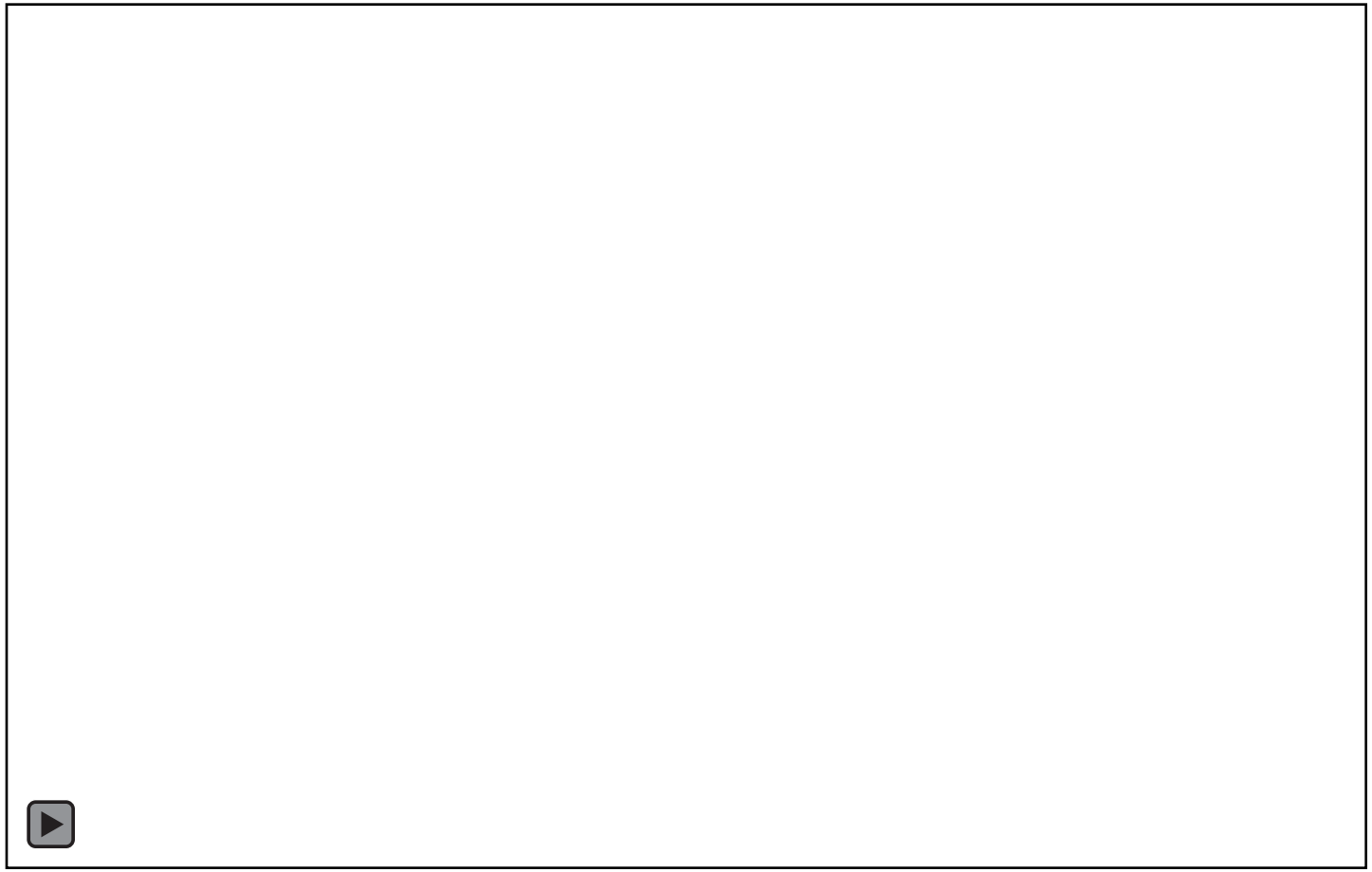
State-of-the-art Work: RRT*



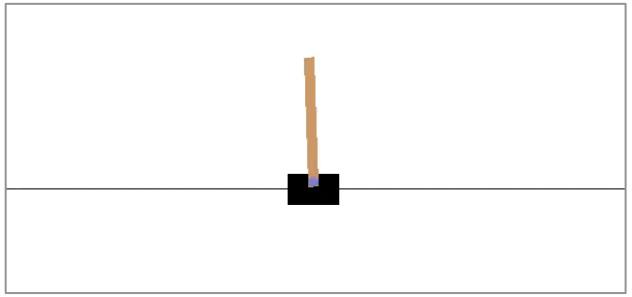
Dynamic Motion Planning

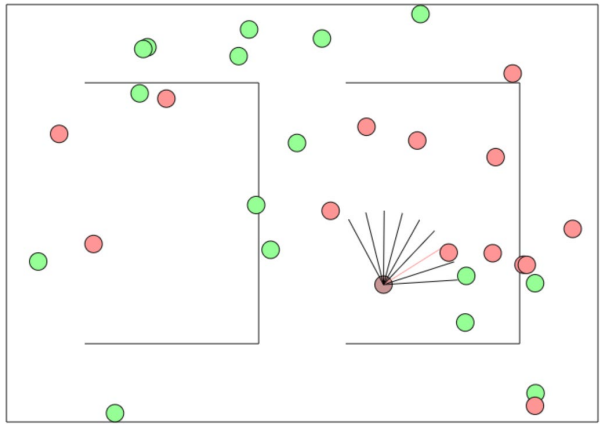


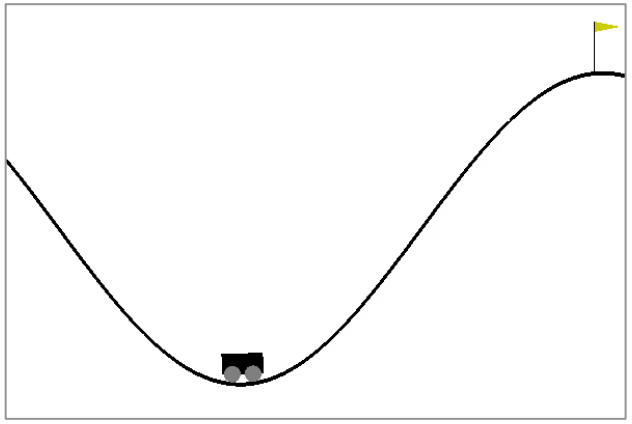
RRT with multiple goal states

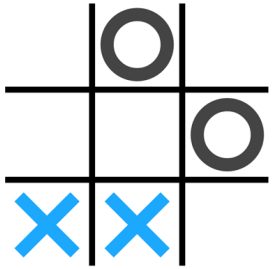


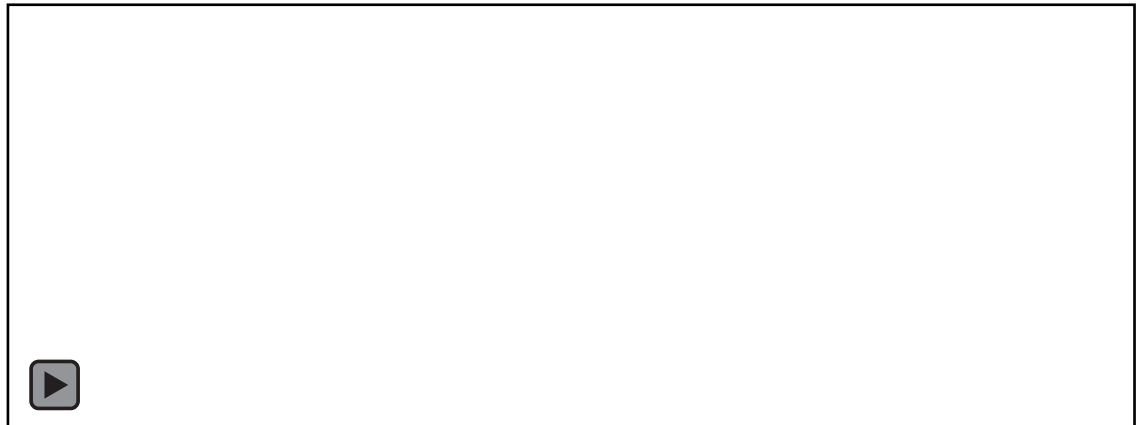
State Representation is Critical





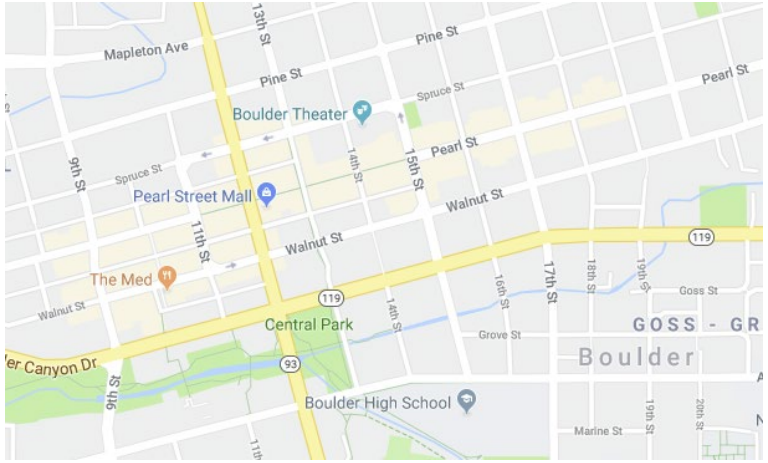


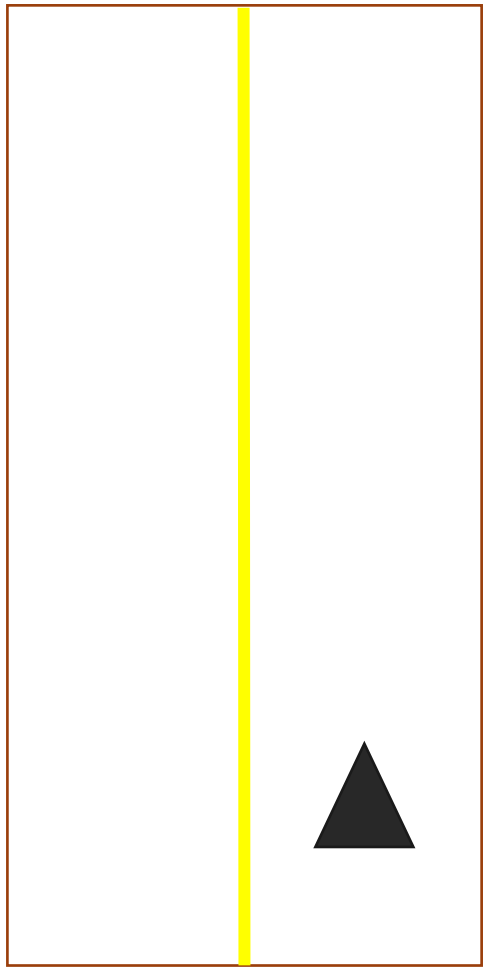




Exercise: Building an Autonomous Car

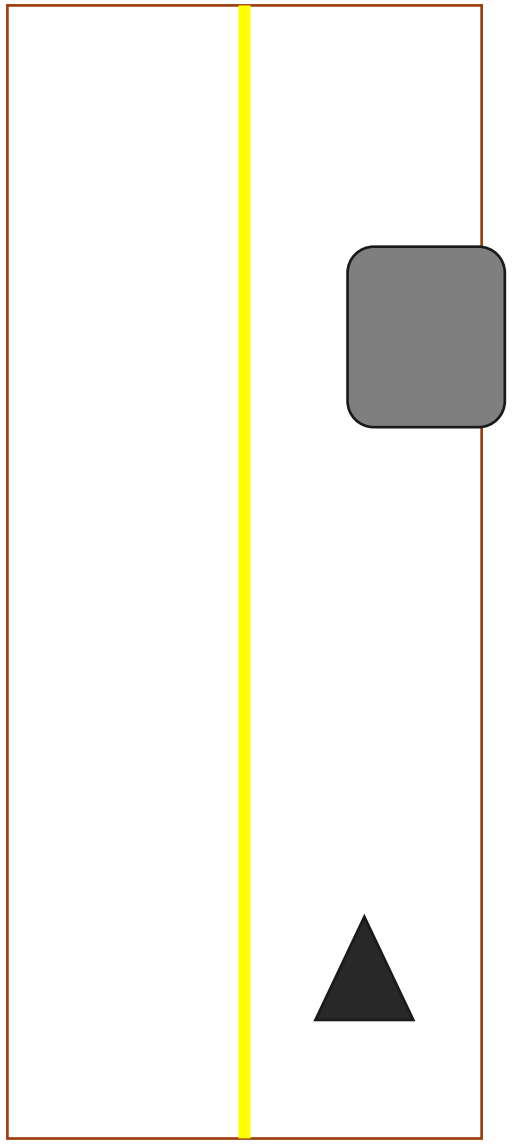
1. World Representation
2. Planning System



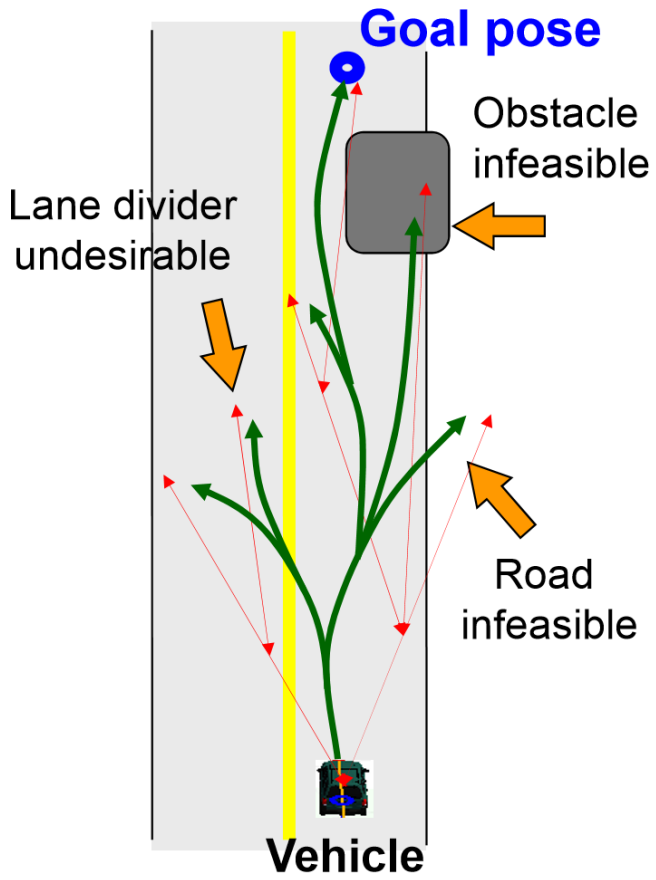


Exercise: Building an Autonomous Car

How can we get our agent to accommodate obstacles?



Exercise: Building an Autonomous Car



RRT for Autonomous Parking

How can we make use of these representations?

Search algorithms provide a way to find a path!