CS 182 Lecture 4: Motion Planning

Professors: Ariel Procaccia and **Stephanie Gil**

Email: sgil@seas.harvard.edu

Prof. Gil Office hours: Wednesdays 2:30-3:30p

Last Time:

- Informed search
  - Greedy best-first search
  - A*
  - Optimality of A*

This Time:

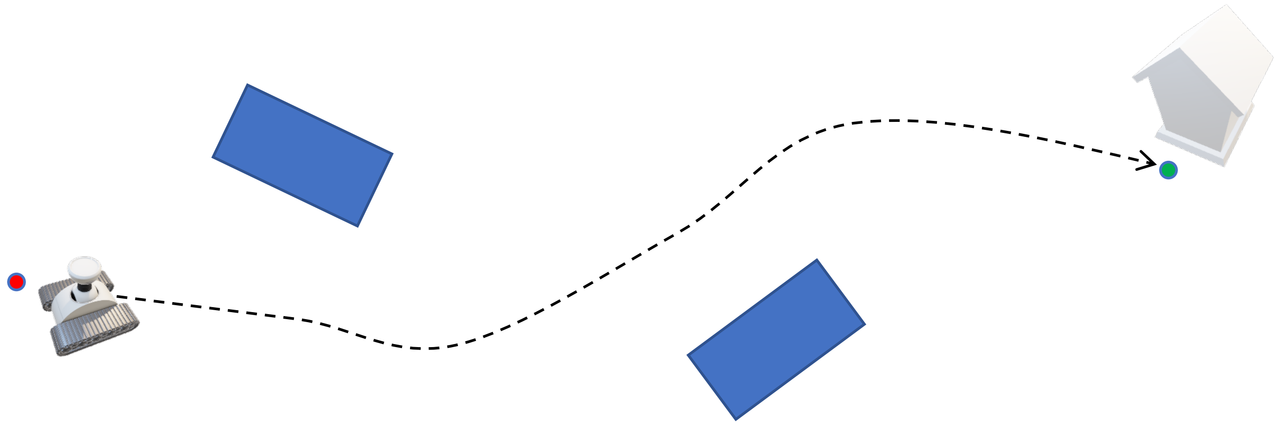
- Motion planning
  - Discretizing an environment
  - Searching for a path from start state to goal state (tie to previous lecture)
  - Questions of optimality

- Practical motion planning
  - Higher dimensions
  - Some real-world problems and examples

- Reference reading: “Planning Algorithms” by Steven LaValle Ch. 5&6 (available online)

The Motion Planning Problem

- What series of motions will get the robot to the goal?



- Are there cases where there is no solution to this problem?
  - Why would this happen?
  - Is there a way to know this is the case?

Practicalities: Modeling the Problem

- Problem: find a series of *valid* configurations that move the object from source to destination

- Validity?

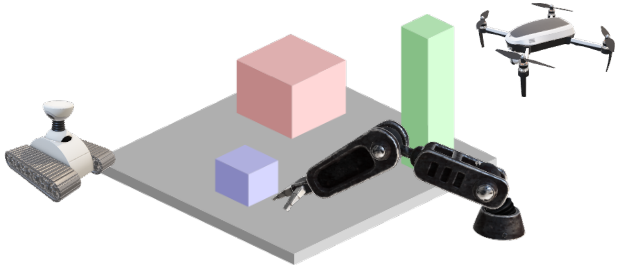
Environment imposed:
Obstacle-free

Dynamics imposed:
Can my agent do that?

Configuration Space

- Idea: focus on point robots – a point represents a *state* (i.e. pose, manipulator position)

Planning Space

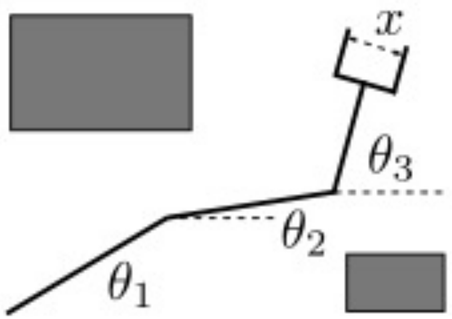


Configuration Space

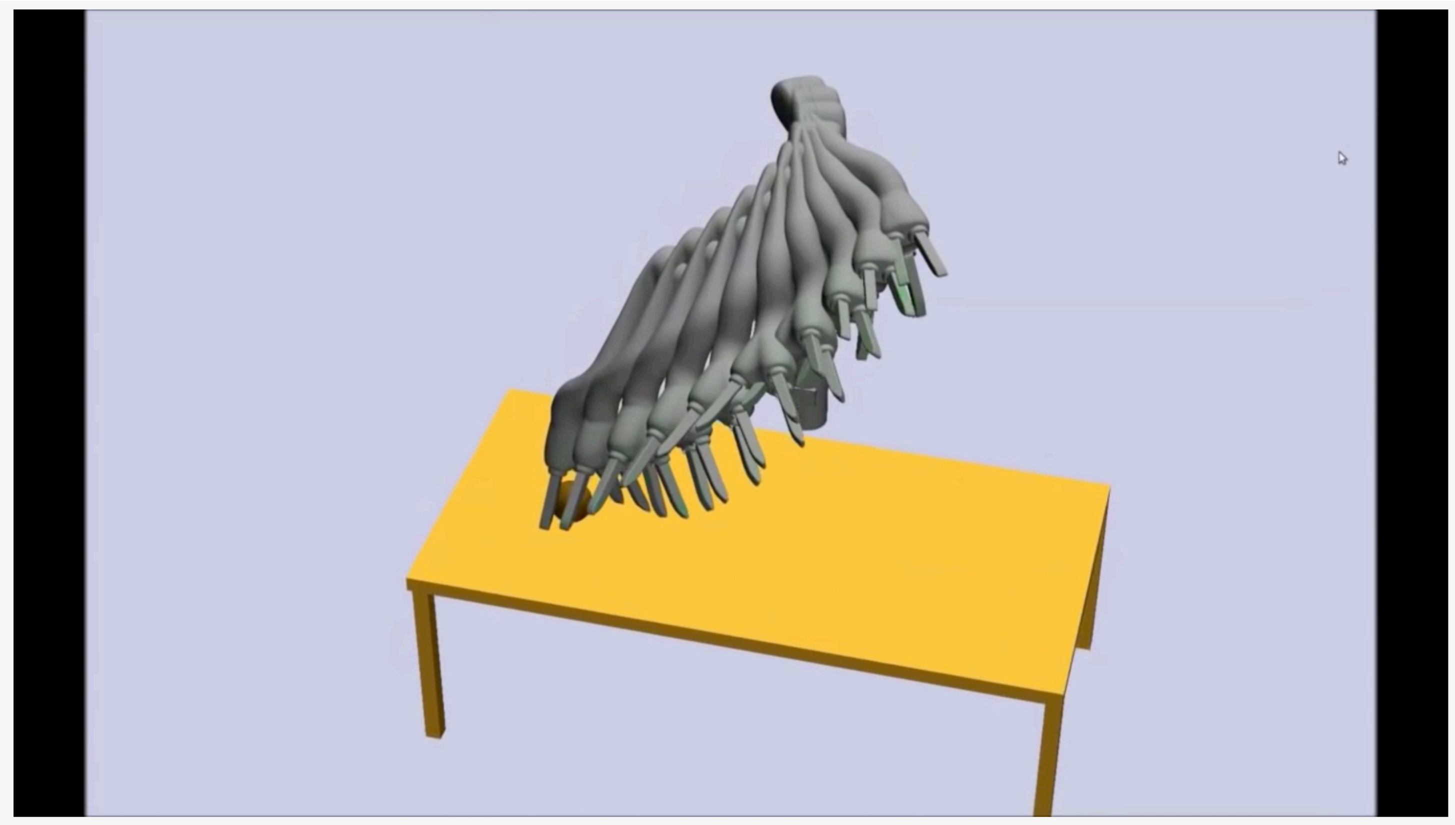
Location (x,y)
Orientation (θ)

Location (x,y,z)
Orientation (ρ, γ, θ)

N-revolute joints

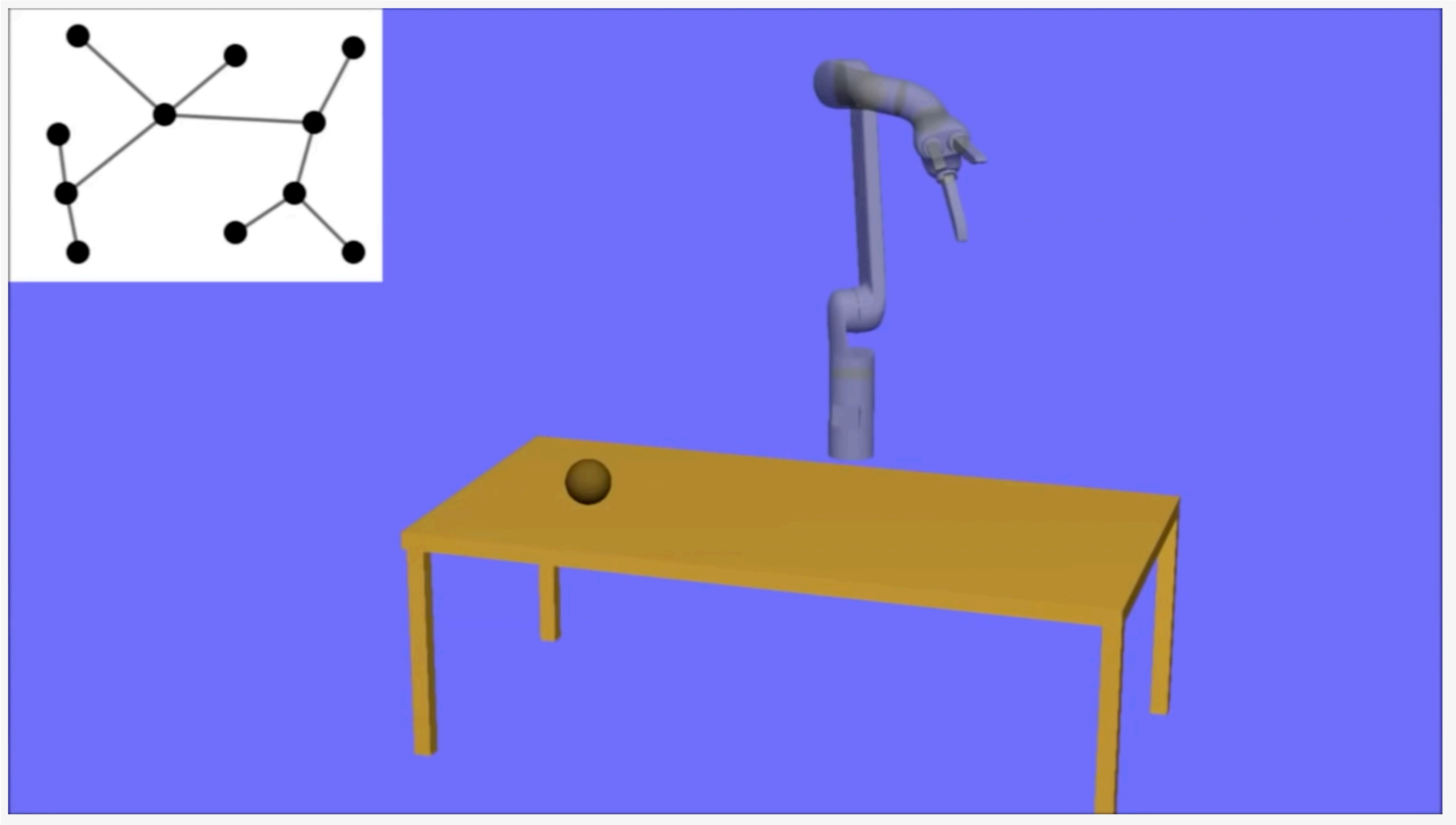


Example: Configuration Space



[Video: Duke robotics]

Example: Manipulator Arm



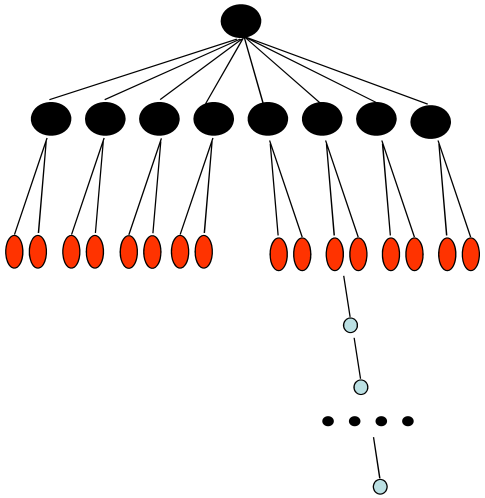
Modeling the Problem (cont.)

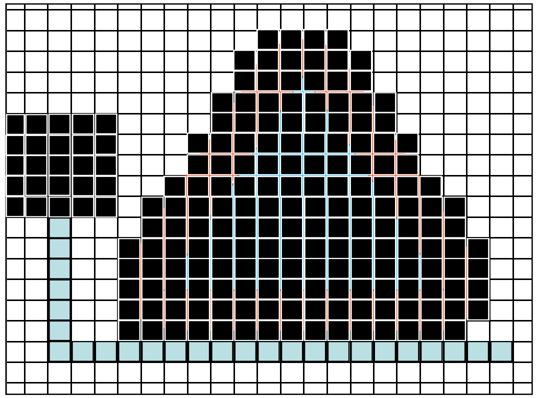
- Given:
  - A robot with configuration space C
  - The set of obstacles: C_{obs}
  - An initial configuration: q_{init}
  - A goal configuration q_{goal}

- Problem: Find a path x:[0,1]--> C (a continuous function) such that the path follows the requirements of
  - Starts from the initial configuration $x(0)=q_{init}$
  - Reaches the goal configuration $x(1)=q_{goal}$
  - Avoids collision with obstacles $x(s) \notin C_{obs}$ for all $s \in [0,1]$

Overview: State-space Search Methods

- Similar to the search problems that you are already familiar with!
- Depth-first search
- Breadth-first search
- A* search
- **Performance guarantees?**





These approaches rely on an assumption, what is it?

Common Performance Guarantee

- **Performance guarantee of interest:**

- An algorithm is *complete* if
    1) it terminates in finite time and
    2) it returns a solution when one exists or it returns failure otherwise

Discretizing the Space

Cell decomposition → going from free space to graph search

- Must satisfy the following properties:

1. **Accessibility -** Computing a path from one point to another *inside a cell* must be trivially easy
    - For example, if every cell is convex then any pair of points in a cell can be connected by a line segment
2. **Representation –** Adjacency information for the cells can be easily extracted to build the roadmap
3. **Querying –** For any two points q_1 and q_G it should be efficient to determine which cells contain them

Vertical Cell Decomposition

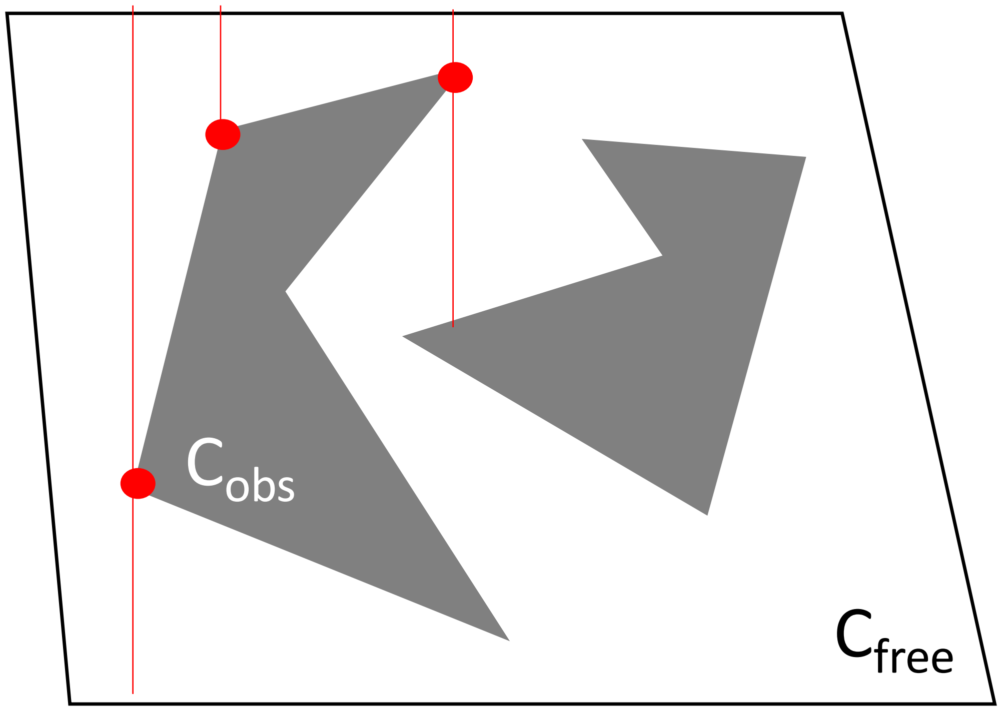
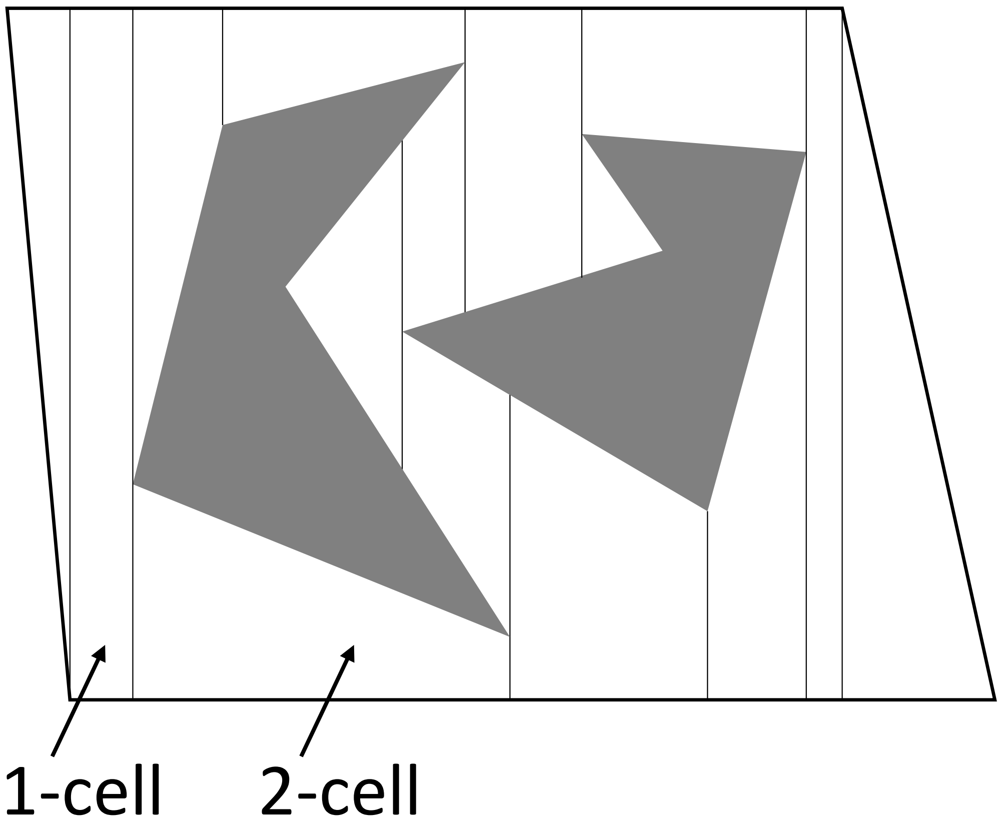


Fig. 6.3 from LaValle text



Each cell is convex so accessibility condition is satisfied

Preserves the connectivity of C_{free}

From Cells to a Graph

- Choose any vertex inside of a cell and connect adjacent cells in a way that accessibility is preserved
- This is called a “roadmap” in motion planning

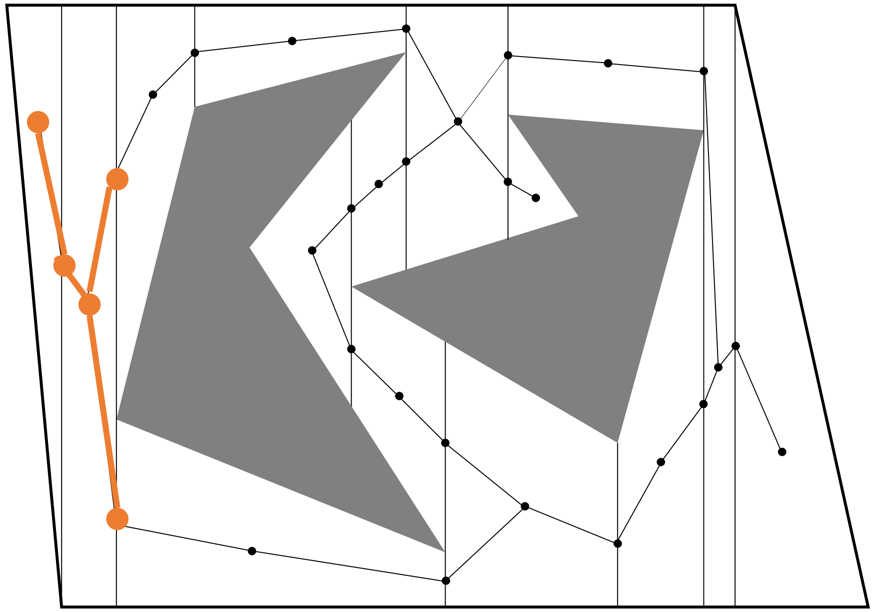


Fig. 6.4 from LaValle text

Solving for a Path from a Query

- Solving for a path between two query points (q_1, q_G):

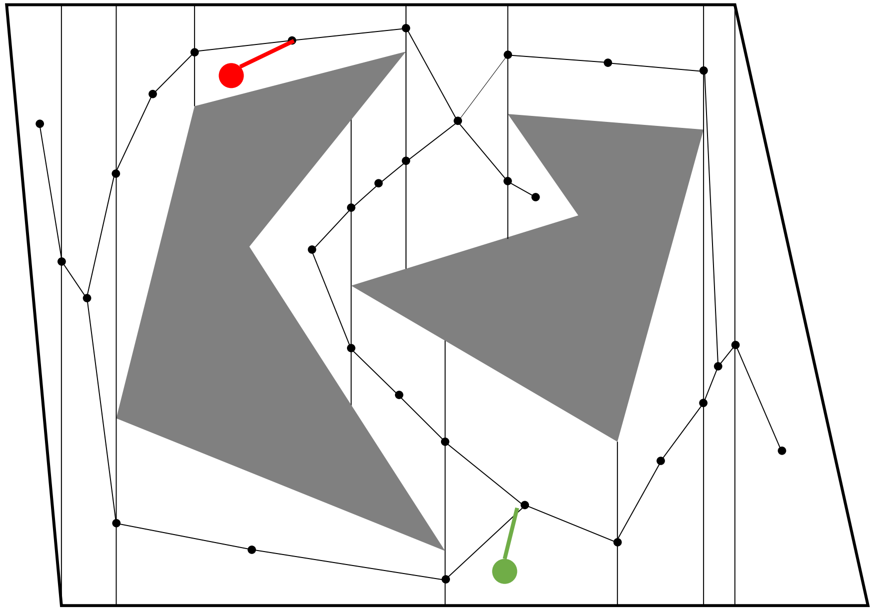


Fig. 6.4 from LaValle text

- Q1: Use 1) DFS expanding rightmost descendant first and 2) A* search using distance as edge cost to find a path from $q_1\ to\ q_G$

Solving for a Path from a Query

- Solving for a path between two query points (q_1, q_G):

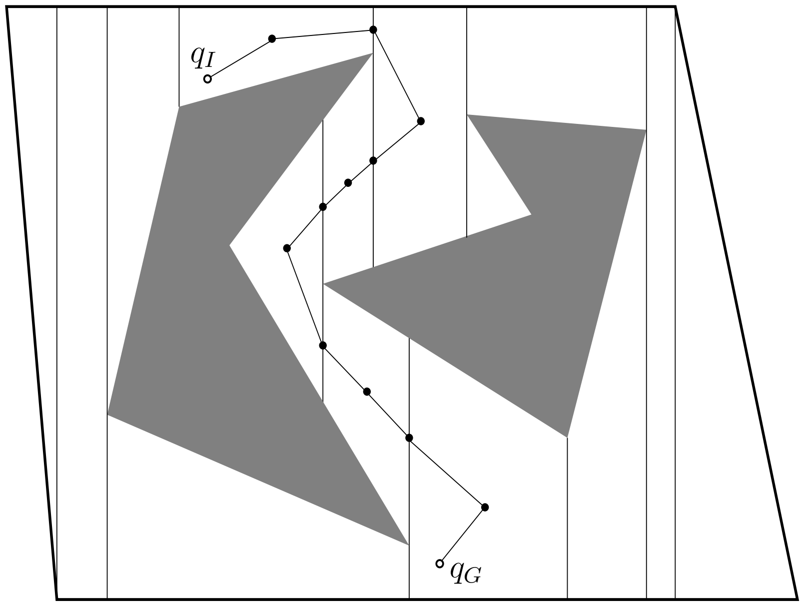
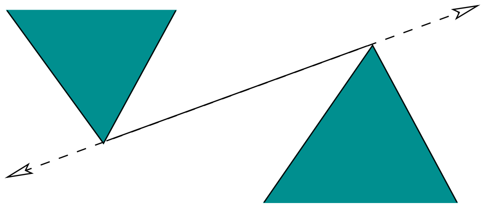


Fig. 6.5 from LaValle text

Visibility Graph – Another Type of Discretization

- A visibility graph results in the optimal *shortest path roadmap*
- Two edge types:
  - Reflex vertex - A polygonal vertex for which the interior angle (in C_{free}) is greater than π
    - Vertices of a convex polygon are reflex vertices – draw an edge between these
  - Bitangent edges – Mutually visible vertices are connected



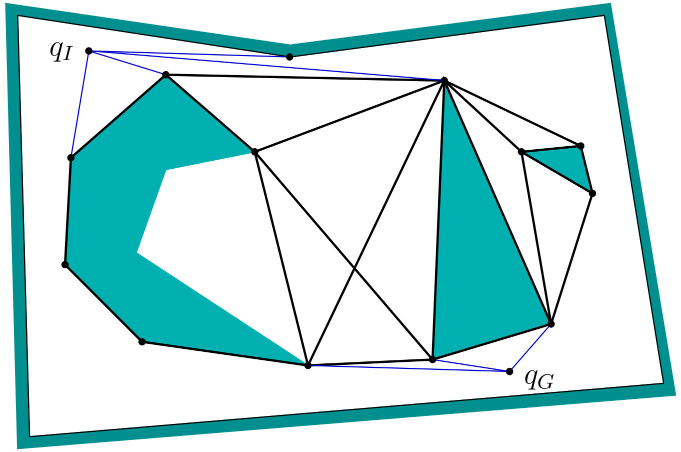


Fig. 6.12 from LaValle text

Shortest-Path Roadmap

- This idea of the shortest-path roadmap may be the first example of a motion planning algorithm!

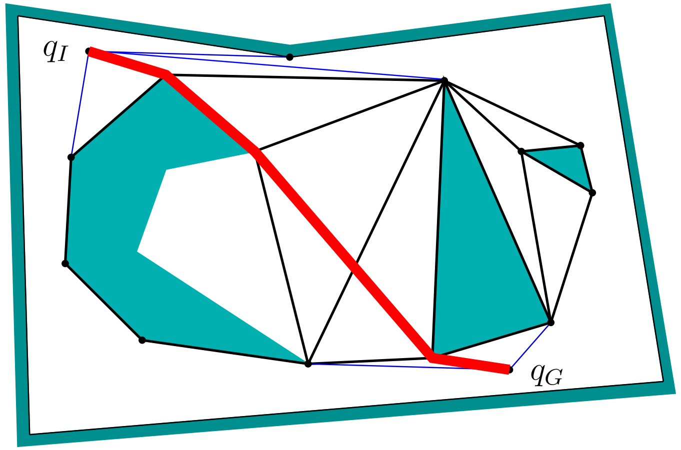


Fig. 6.13 from LaValle text

# of edges in the roadmap

- Construction complexity (bitangent tests) is O(nlgn+m)
  - This can be faster for certain types of environments

Question of Optimality?

- For any path x:[0,1]--> C_{free}, it is always possible to find a shorter path → therefore the shortest path problem in C_{free} is ill-poised!

- Consider the problem of determining the shortest path in cl(C_{free})
  - i.e. the agent is allowed to touch or graze obstacles

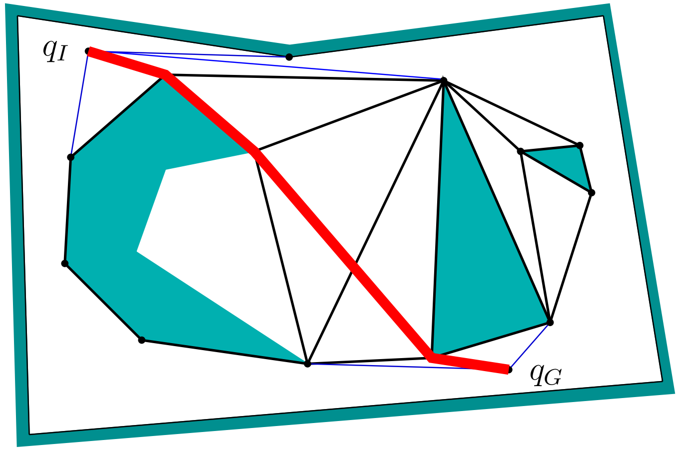


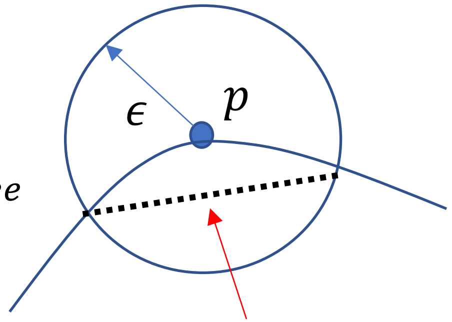
Fig. 6.13 from LaValle text

Optimality Proof

<u>Lemma</u> - Consider a set S of disjoint polygonal obstacles, and start and goal positions q_1 and q_G respectively. Any shortest path between q_1 and q_G is a polygonal path where the inner vertices are vertices of S

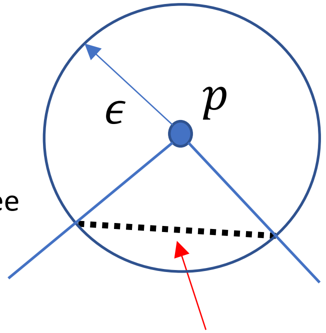
<u>Proof by contradiction</u> – Assume that this is not the case and that the shortest path goes through a point p in (the interior) C_{free}

$\exists\, \epsilon > 0\ s.t.$
$ball(\epsilon) \in C_{free}$



A shorter path!

Try a
polygonal
path in C_{free}



Same argument
applies

Visibility Graph-based Path Finding

- Recap

Algorithm:

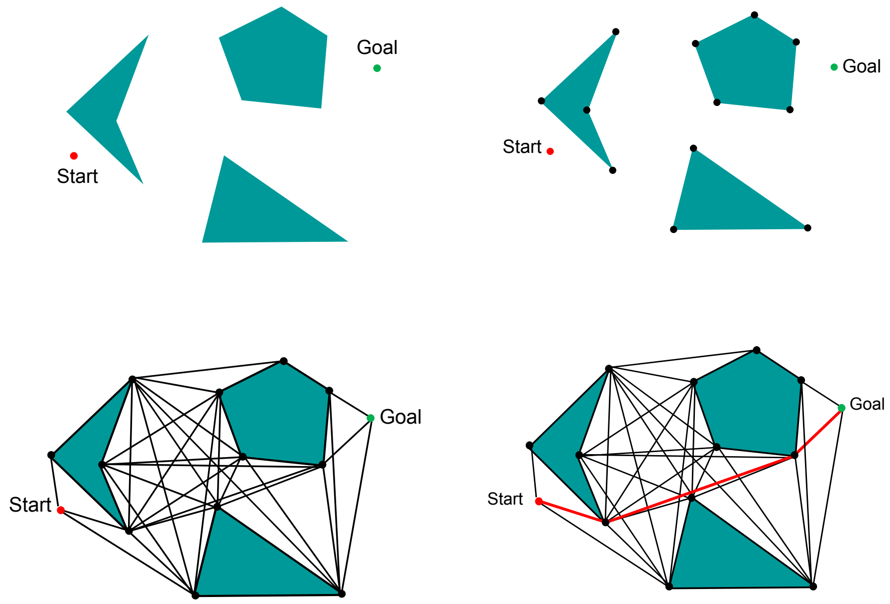
➢ Create a graph:

➢ <u>Vertex set:</u> all vertices on polygonal obstacles

➢ <u>Edge set:</u> all vertex pairs that can be connected by a straight collision free path

➢ Return the shortest path on this graph.

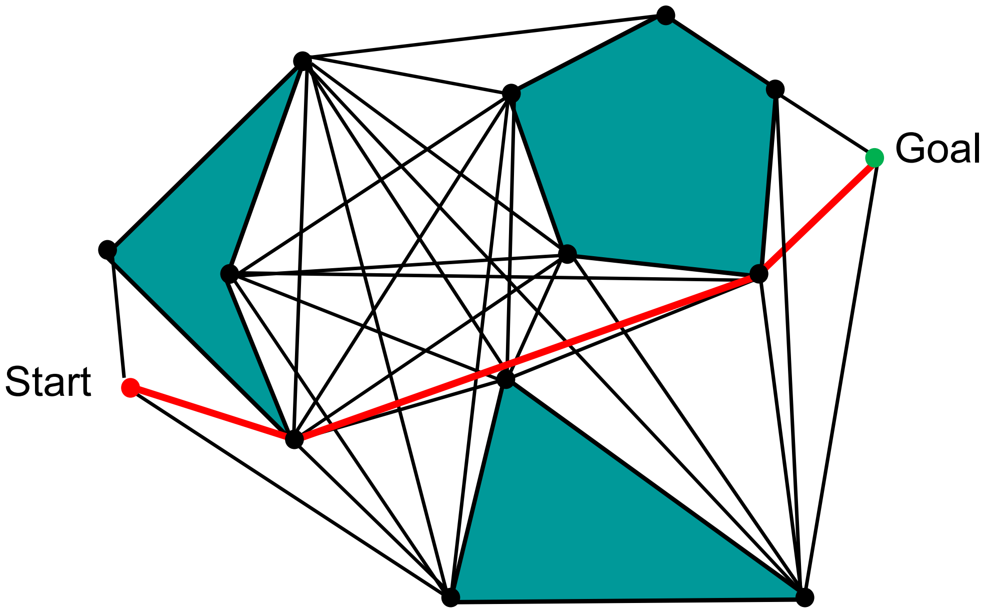
Visibility Graph Approach



This algorithm returns the shortest path (in 2D)!

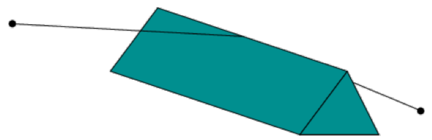
Practicality of using Visibility Graphs?

- **Q2** (polls everywhere): Can we scale-up to 3D?



Navigation in 3D is NP-hard

- Visibility graph only applies to 2D Euclidean spaces
  - Intuition: in 3D, the optimal path does not have to go through vertices, and might go through edges instead…



  - Optimal path planning in 3D is NP-hard
  - Extensive computational issues (# obstacles, dimension of the workspace, dynamics, etc.)

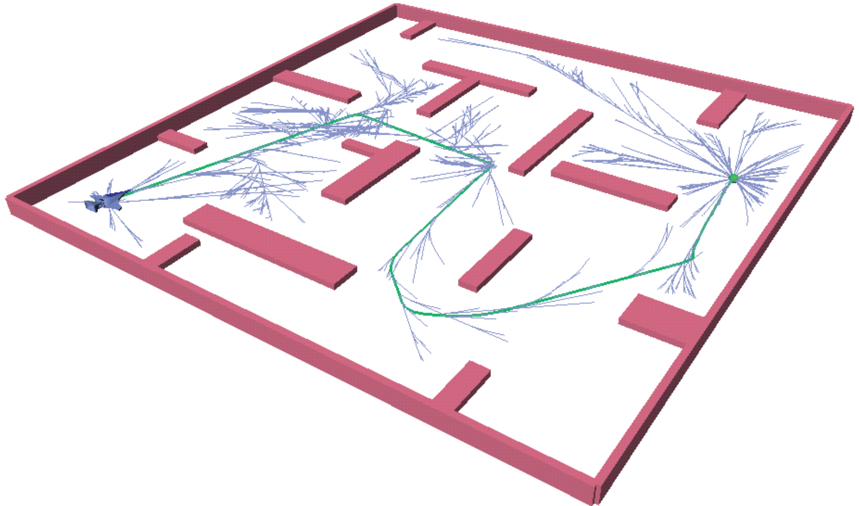
Recap on State-space Search Methods

- Rely on discretization of the configuration space
- Guarantees resolution completeness: the algorithm returns a solution when one exists, if the *resolution parameter* is fine enough
- Exponential running time with increasing degrees of freedom

Sampling-based Methods

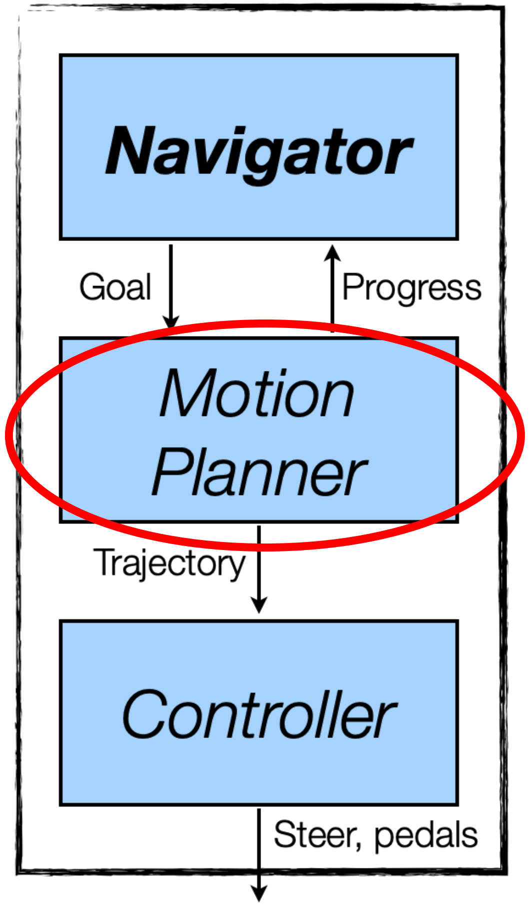
- **Key idea:**
  - Sample the configuration space
  - Connect samples with trajectories to infer the connectivity of the free space

- Probabilistic RoadMap (PRM) and Rapidly-exploring Random Trees (RRT) are two widely used variants of this method



Some Practical Examples…

- Autonomous vehicles

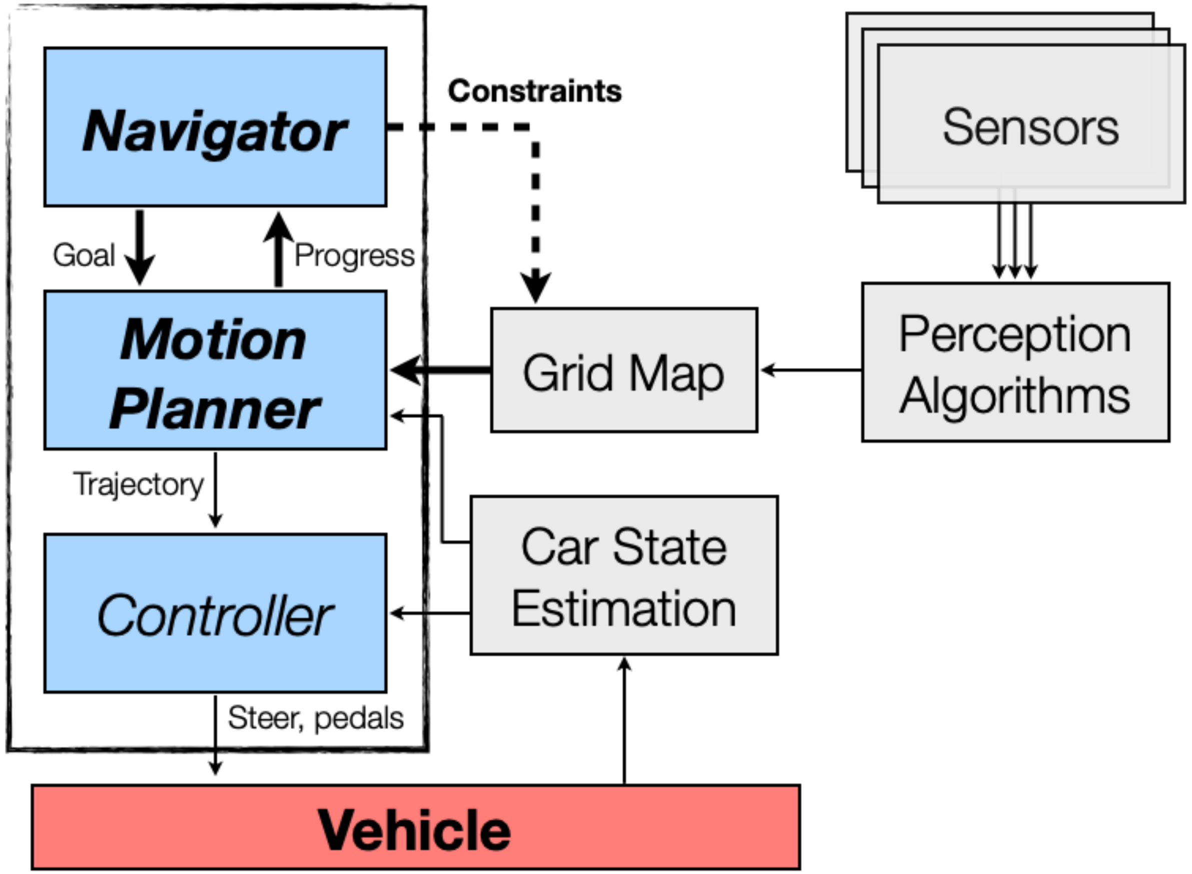


A* Algorithm for Navigating through the Road Network

[Slide credit: Sertac Karaman, MIT DARPA Urban Challenge]

Some Practical Examples…

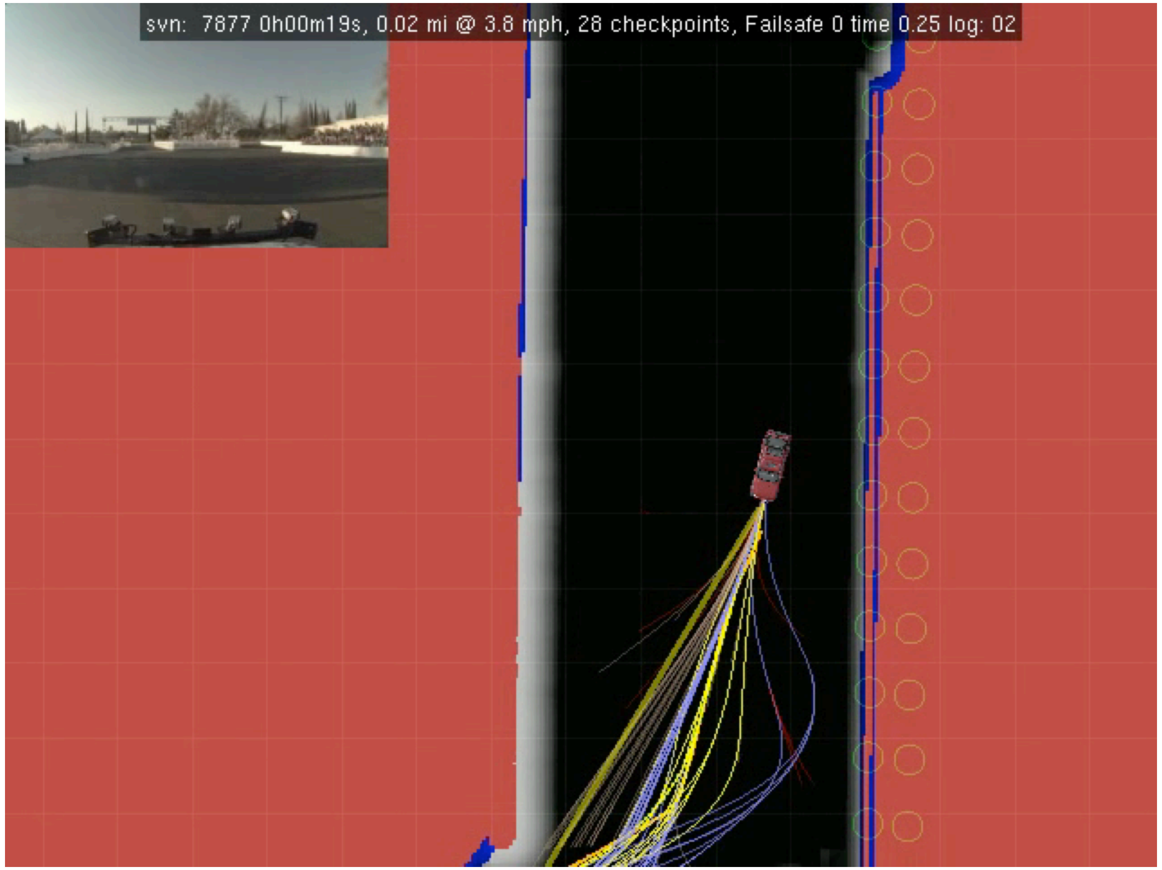
- Autonomous vehicles



[Slide credit: Sertac Karaman, MIT DARPA Urban Challenge]

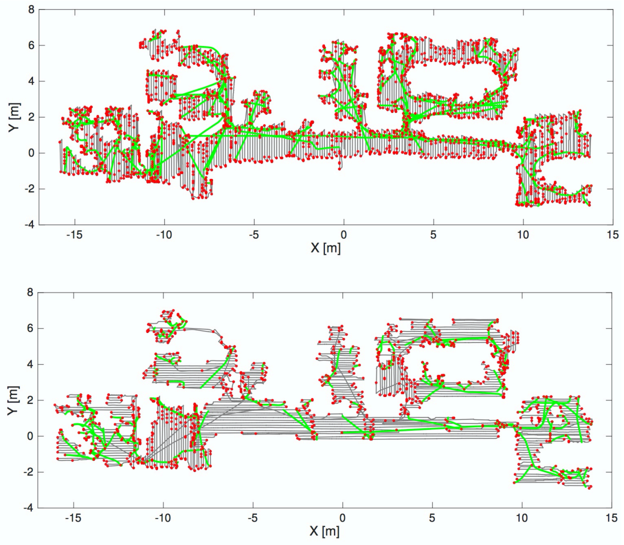
Some Practical Examples…

- Autonomous vehicles

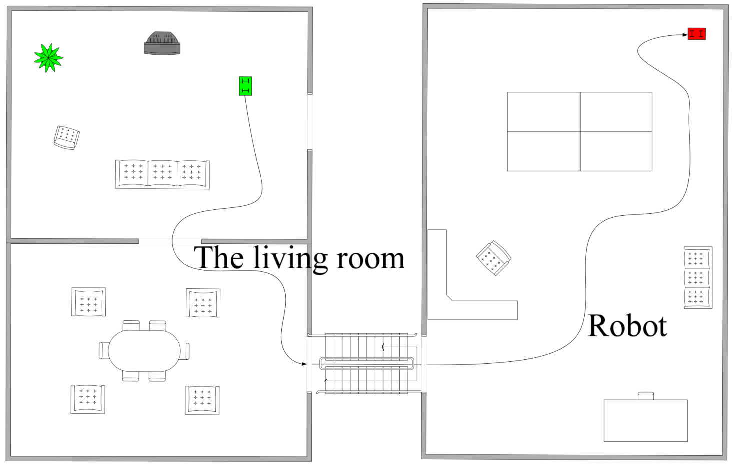


Some Practical Examples…

- Service robots



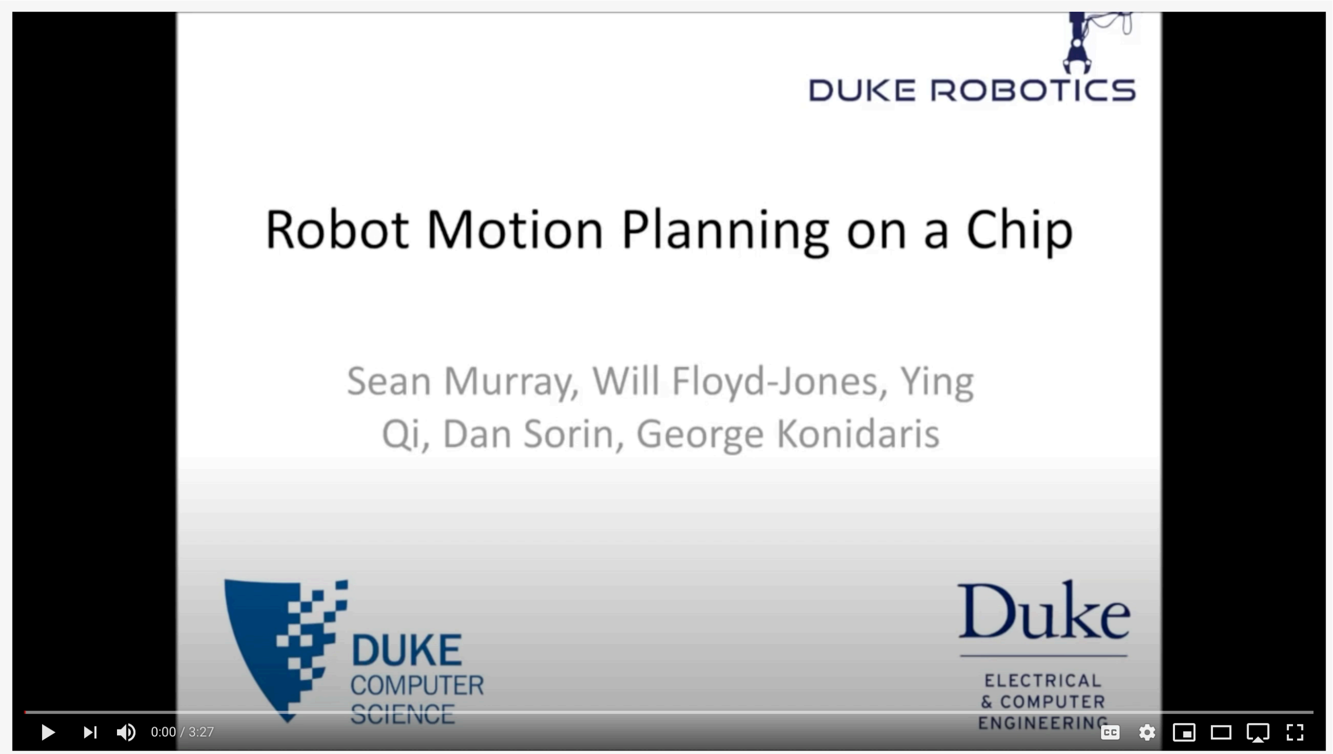
[Image credit: IEEE Spectrum]



[Image credit: SEAS UPenn]

Some Practical Examples…

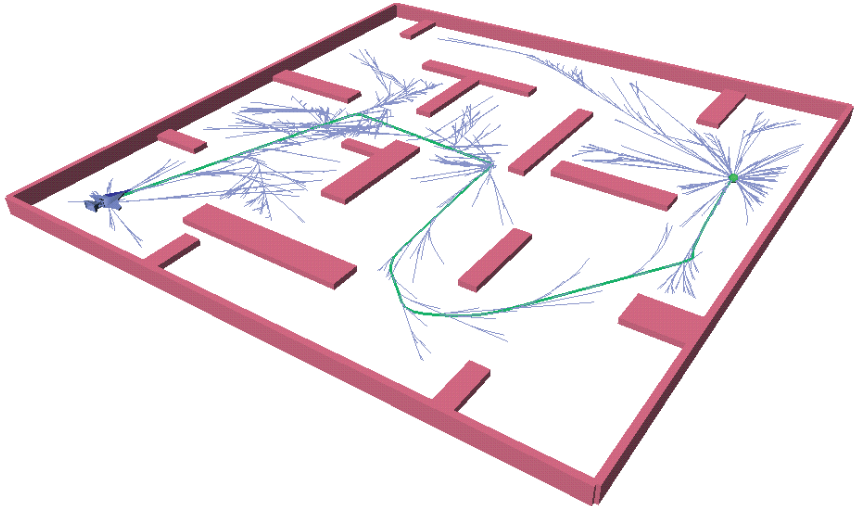
- Robot manipulators



[Video: Duke robotics]

Sampling-based Methods

- **Key challenges:**
  - *How* to best sample the configuration space?
  - Optimality? Completeness?



Probabilistic RoadMap (PRM Algorithm)

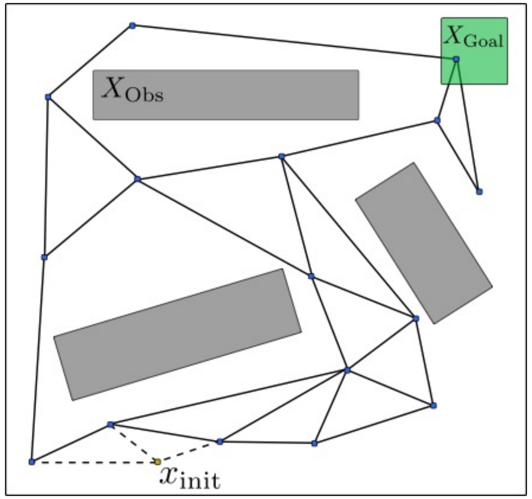
1. Sample N points uniformly at random from C

2. Connect each pair with a straight trajectory

3. Delete all vertices and edges that lie in the obstacle set C_{obs}

4. Return the remaining roadmap G=(V,E)

Probabilistically complete!



[Image credit: Sertac Karaman *RSS*]

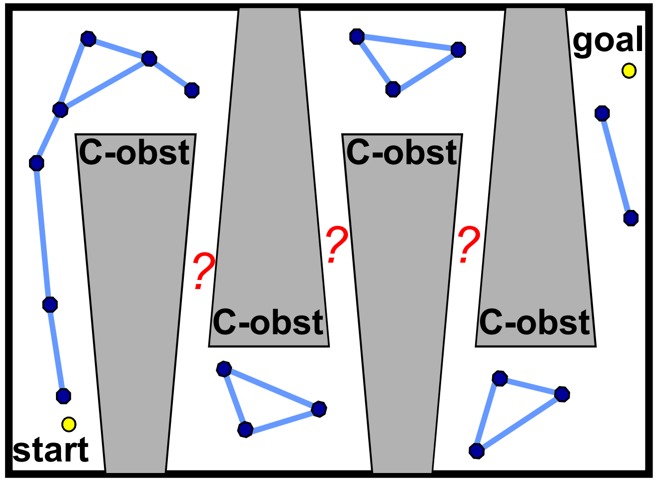
Dynamic environments?

Narrow passages?

PRM: Pros vs. Cons

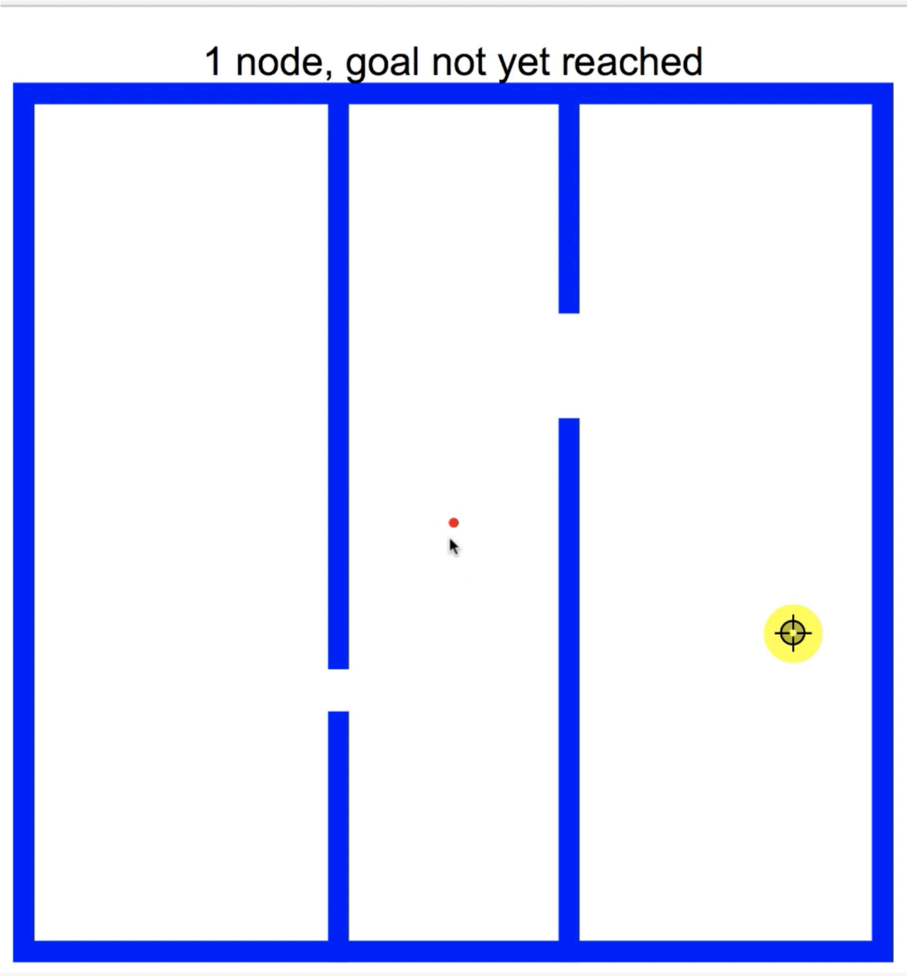
- Probabilistically complete
  - Probability of returning a solution approaches 1 as the number of samples increases

- But… performance (# samples needed) can be environment-dependent



[Image credit: Sertac Karaman *RSS*]

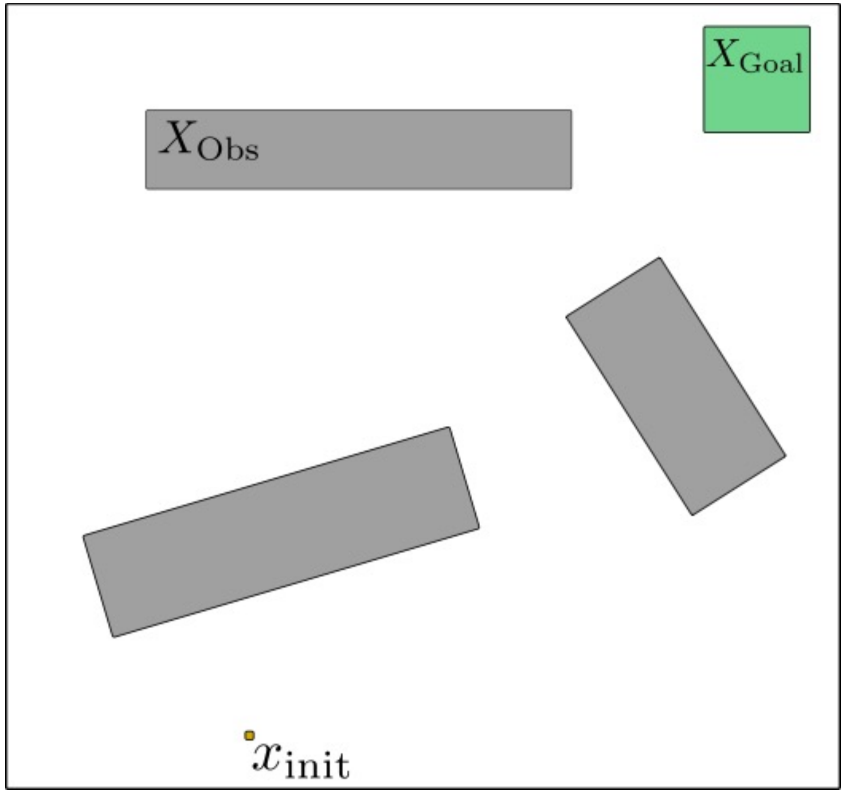
Incremental Sampling: Random Trees



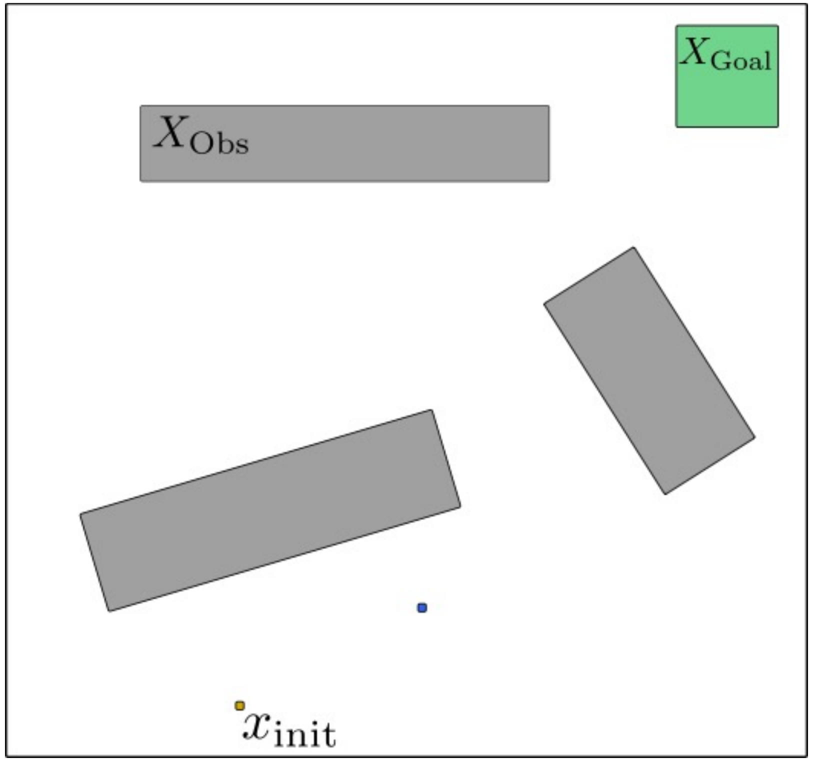
What went wrong?
How can we improve?

[Video source: Aaron Becker University of Houston]

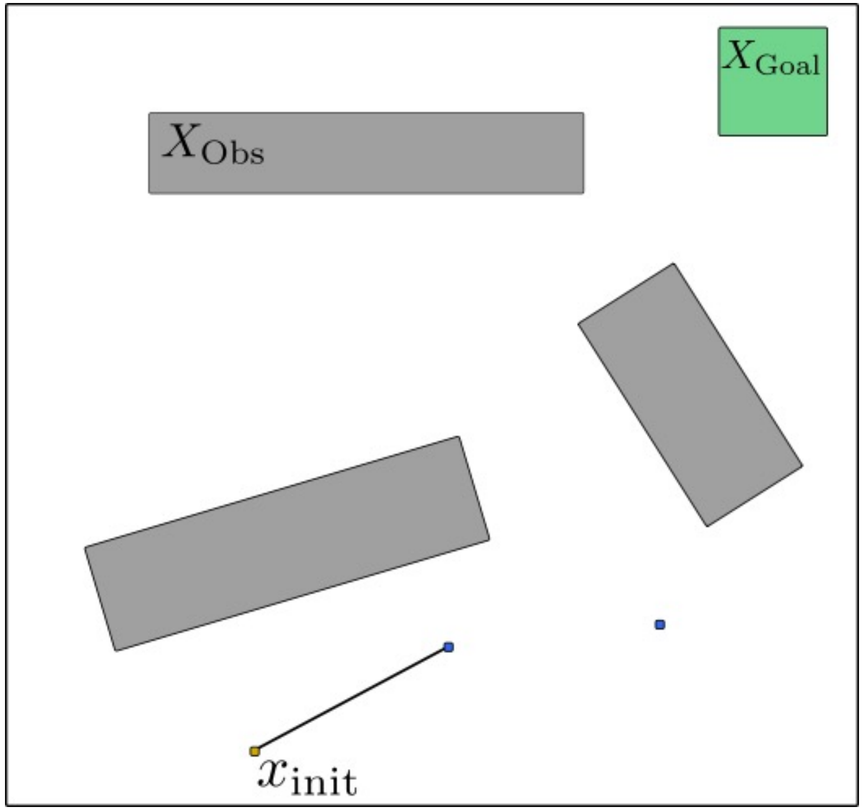
Rapidly-exploring Random Tree (RRT) Algorithm



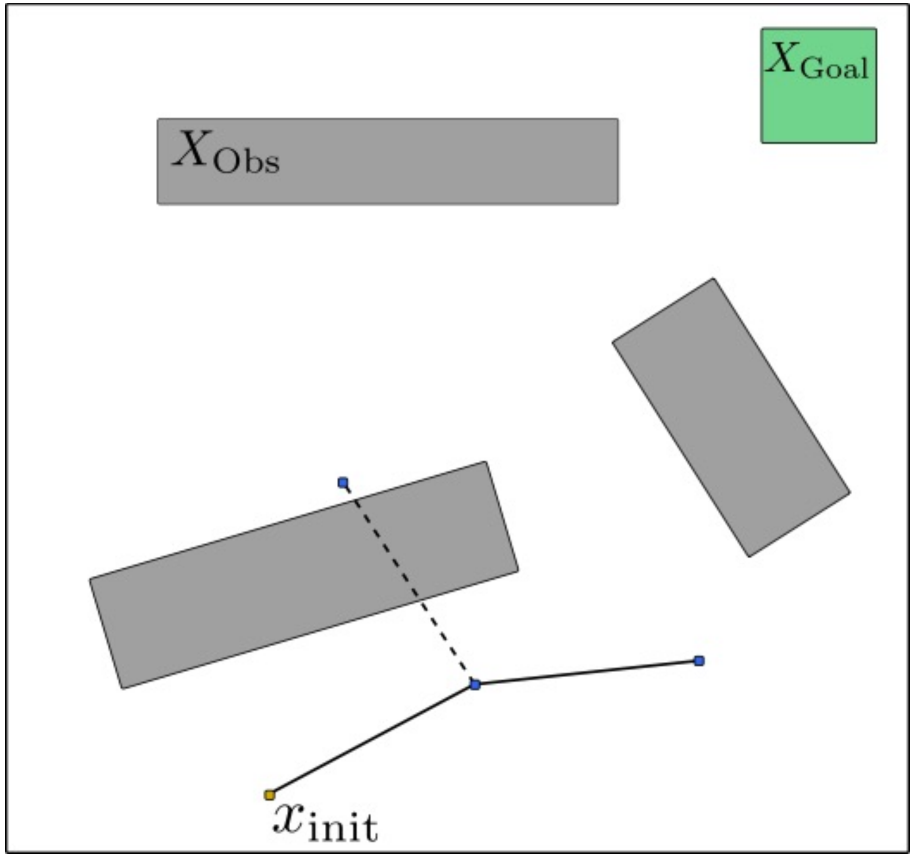
Rapidly-exploring Random Tree (RRT) Algorithm



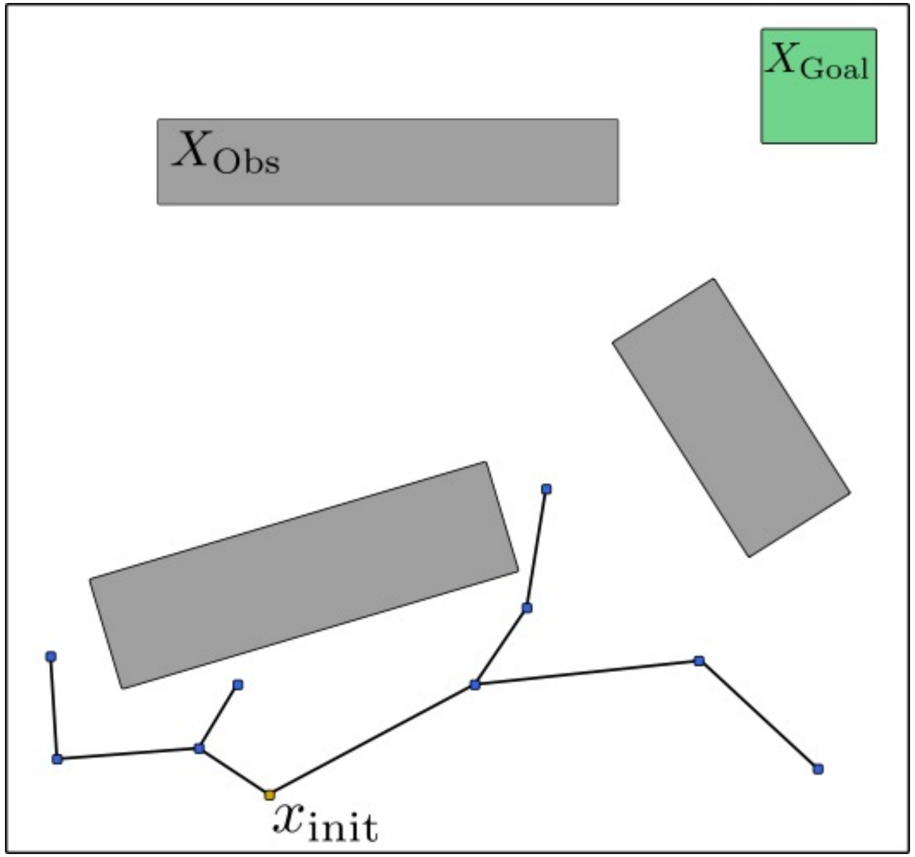
Rapidly-exploring Random Tree (RRT) Algorithm



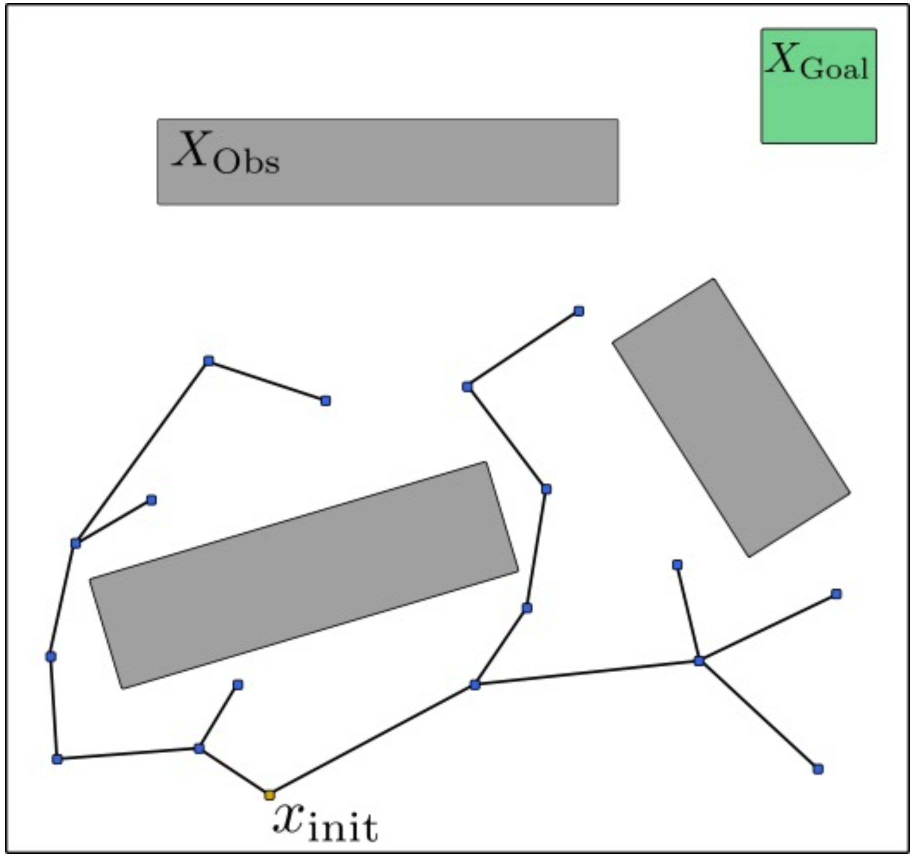
Rapidly-exploring Random Tree (RRT) Algorithm



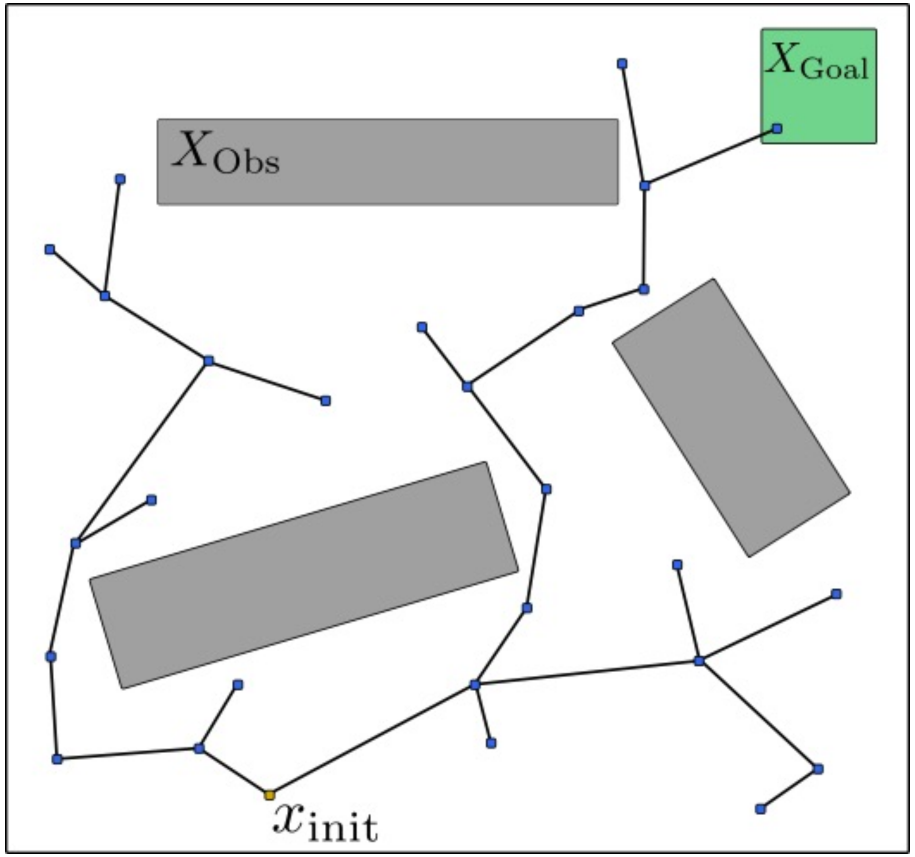
Rapidly-exploring Random Tree (RRT) Algorithm



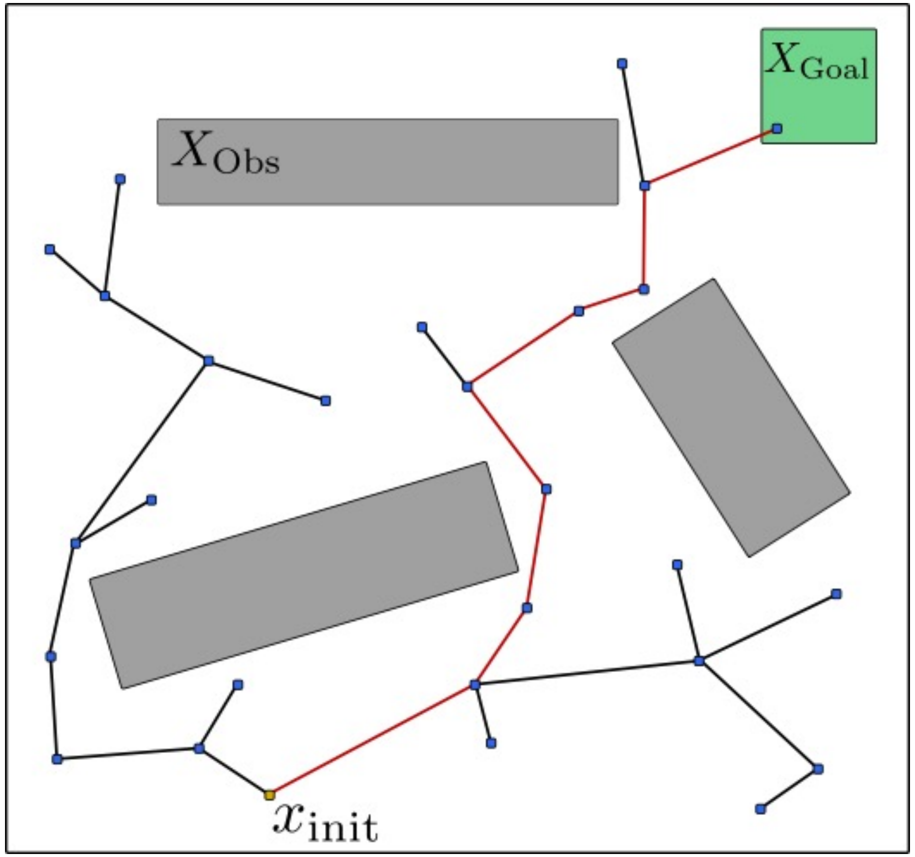
Rapidly-exploring Random Tree (RRT) Algorithm



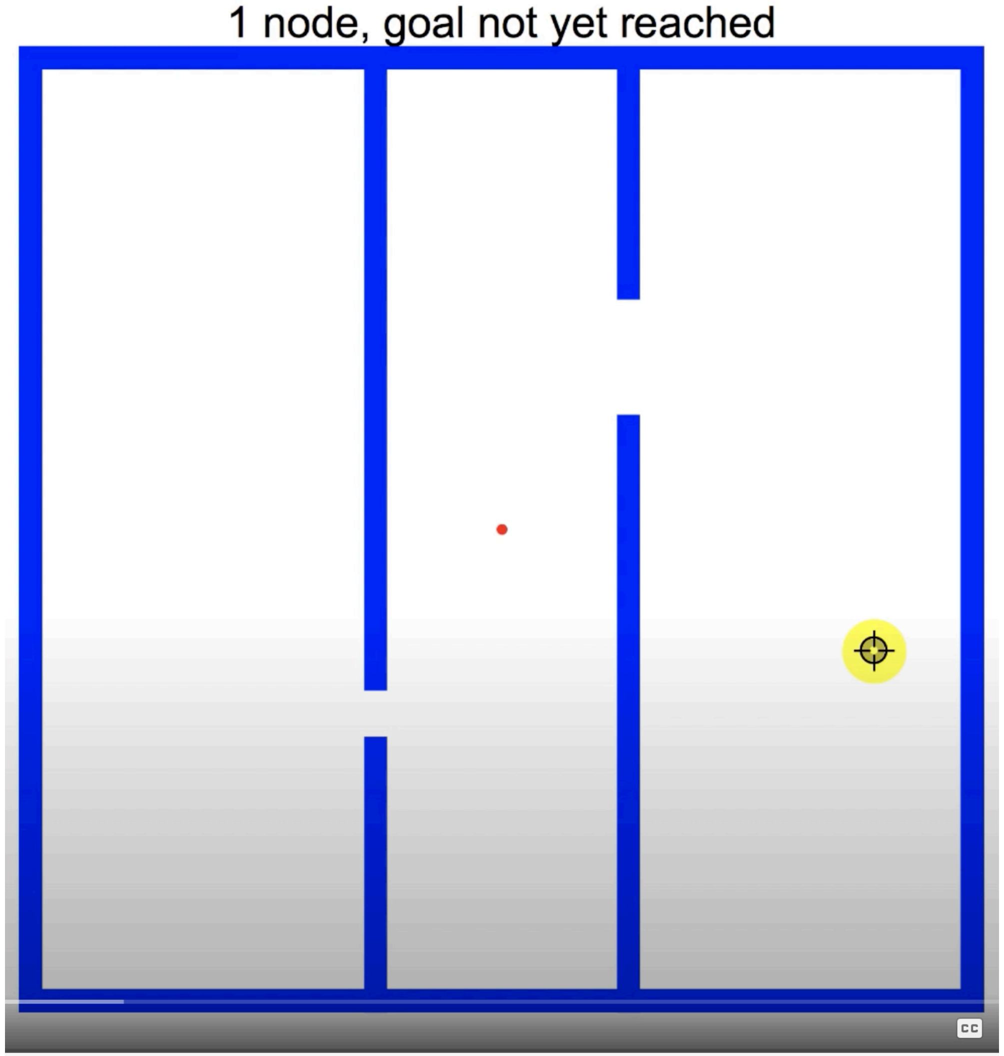
Rapidly-exploring Random Tree (RRT) Algorithm



Rapidly-exploring Random Tree (RRT) Algorithm



RRT Example



[Video source: Aaron Becker University of Houston]

Variants?
- Dynamically feasible paths
- Biasing the search

How might we improve?

Probabilistic Completeness vs. Optimality

- **Q3:** Is RRT *optimal* (i.e. does it produce the shortest path)?

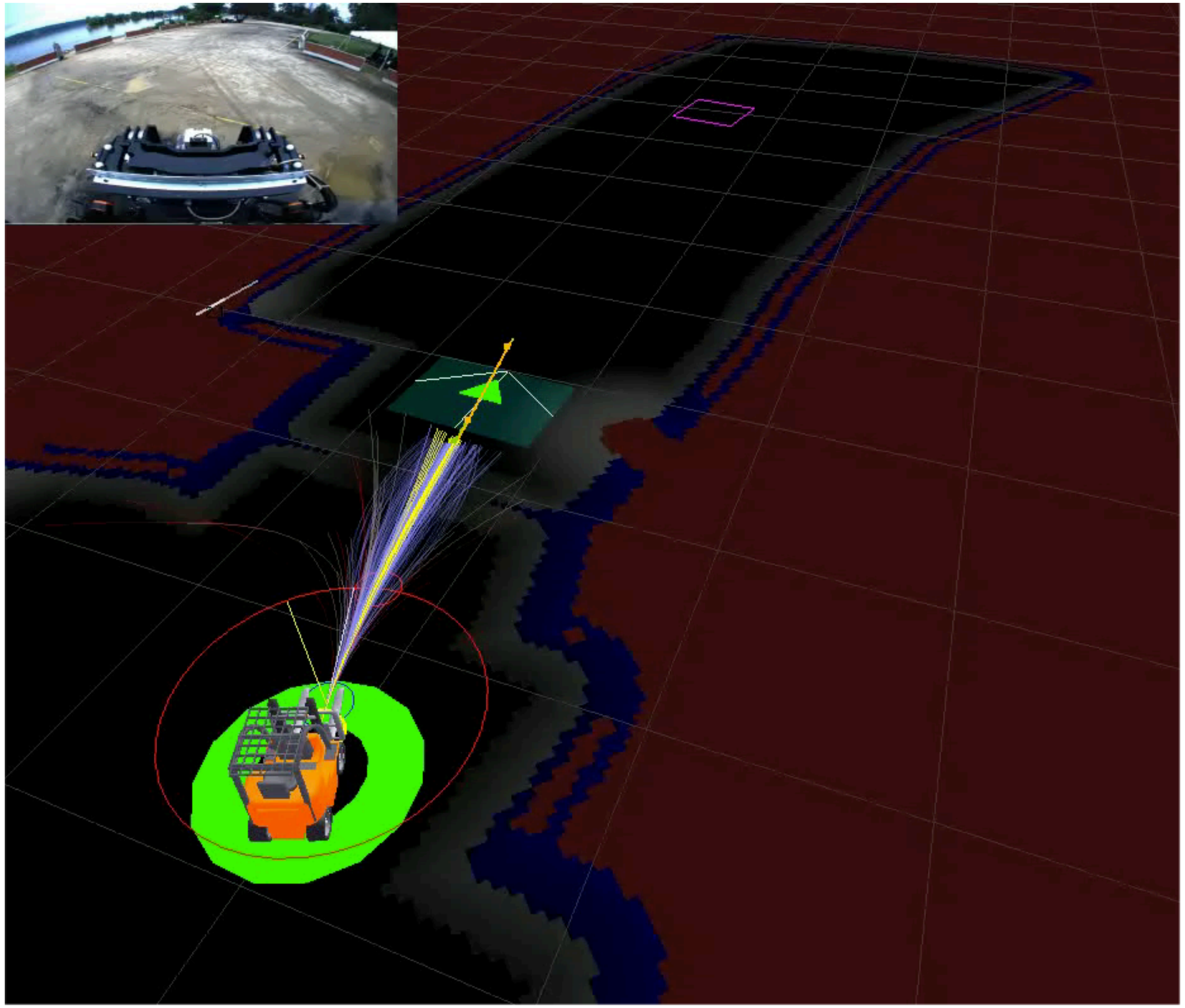
Probabilistic Completeness vs. Optimality

Re: new video

Looks great!

At around 6:03 the bot drives forward and right for an unknown reason -- any idea why?

[Image credit: Sertac Karaman]



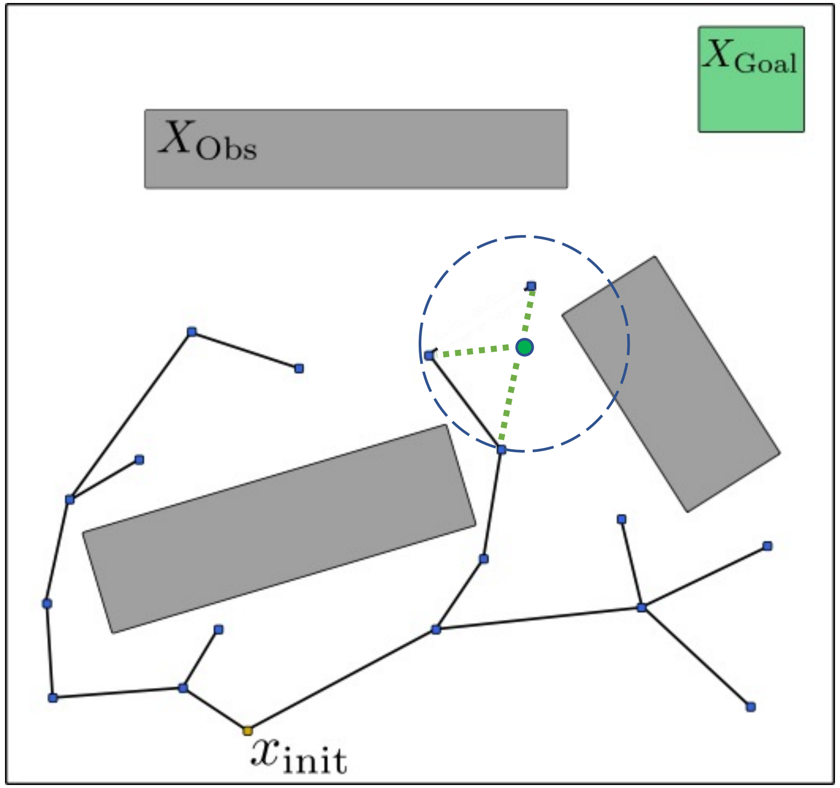
- [Sertac Karaman, PhD’12. Thesis Title: Sampling-based Algorithms for Optimal Path Planning Problems. (Advisor: Emilio Frazzoli)]
- [Brandon Luders, PhD’14. Thesis Title: Robust Sampling-based Motion Planning for Autonomous Vehicles in Uncertain Environments (Advisor: Jonathan How)]

RRT* Algorithm: Intuitive Explanation

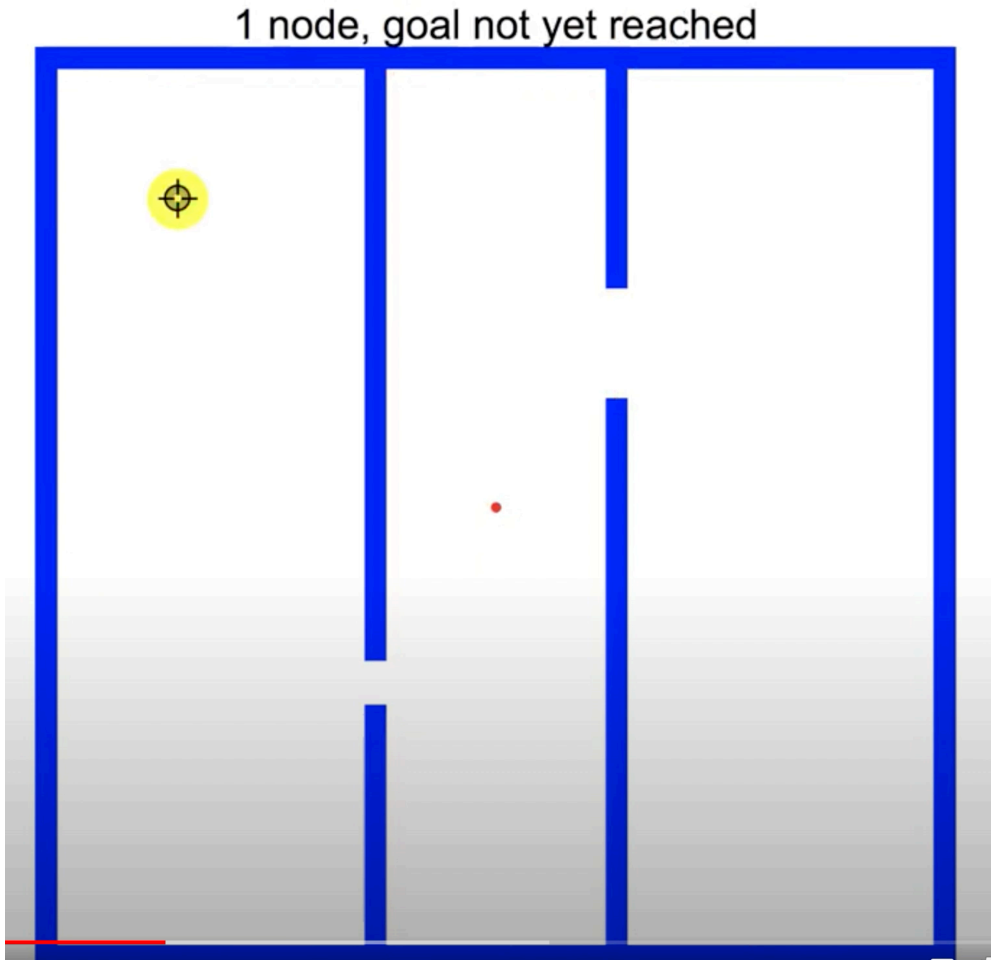
1. Sample
2. Select best parent node

   Minimize cost from the root

3. Find new lower cost paths
4. Rewire the tree

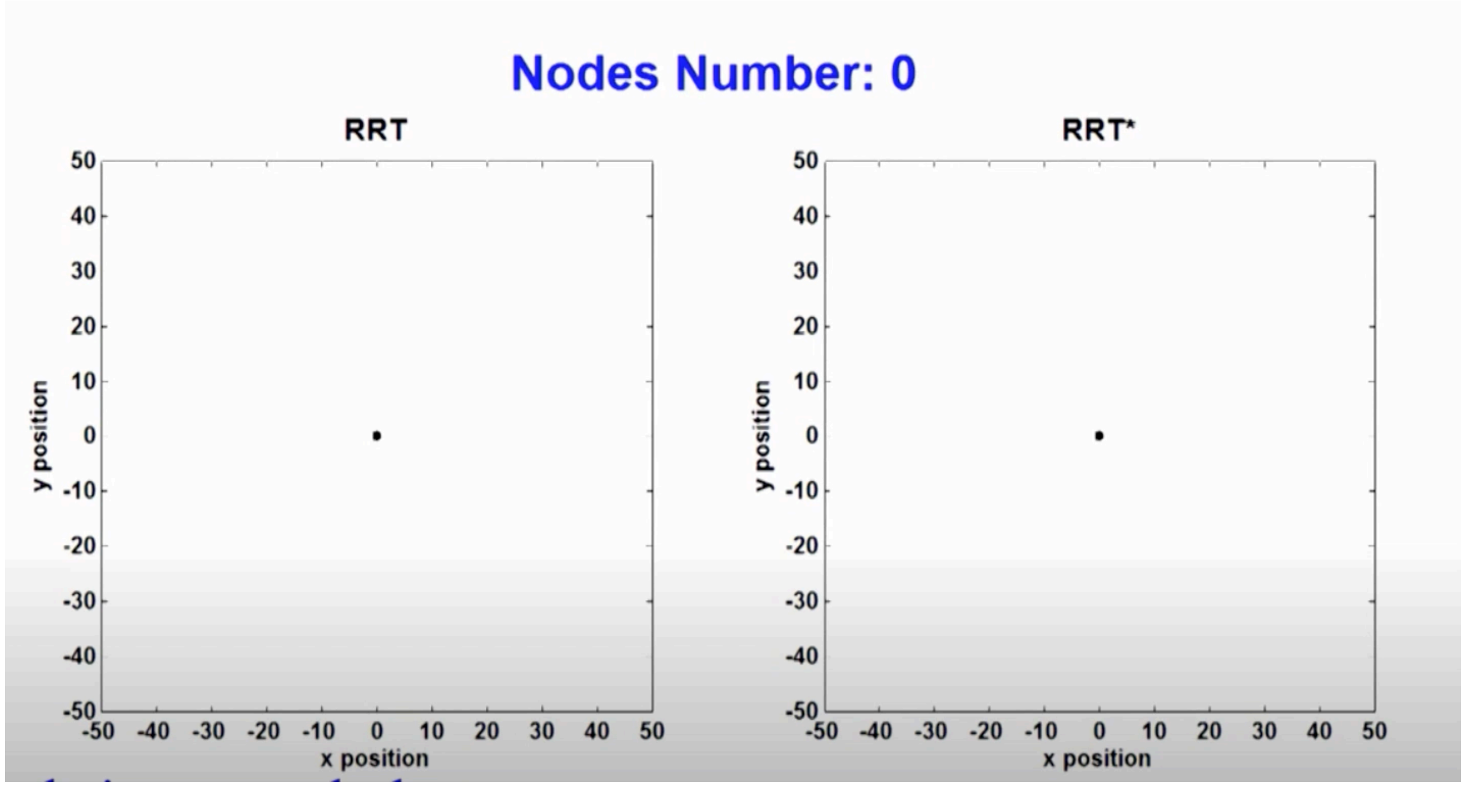


RRT* Example



[Video source: Aaron Becker University of Houston]

Iterations of RRT vs RRT*

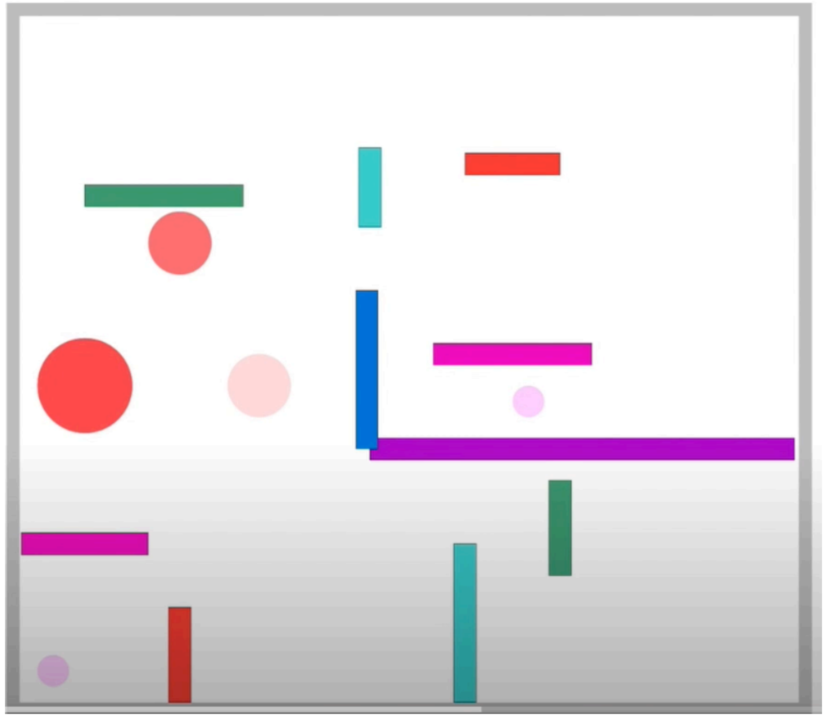


[Simulation by Yiqun Dong]

Variants

- Vehicle dynamics?

- Changing environments?



[Video credit: RRT* FND Advanced Robotics and Mechatronics Systems Laboratory (ARMS)]

Variants

- High DOF?

[Video credit: IROS ’11 Perez, Karaman, Shkolnik, Frazzoi, Teller, and Walter]

Next Time…

- Constraint Satisfaction Problems
  - Reference readings R&N Ch. 6 (Sec 6.1-6.3)