Algorithmic Human-Robot Interaction

Motion Planning II

CSCI 7000

Prof. Brad Hayes

Computer Science Department

University of Colorado Boulder

Today: Motion Planning & Projects

Making Robots Move

- Algorithms for motion planning

Research Projects

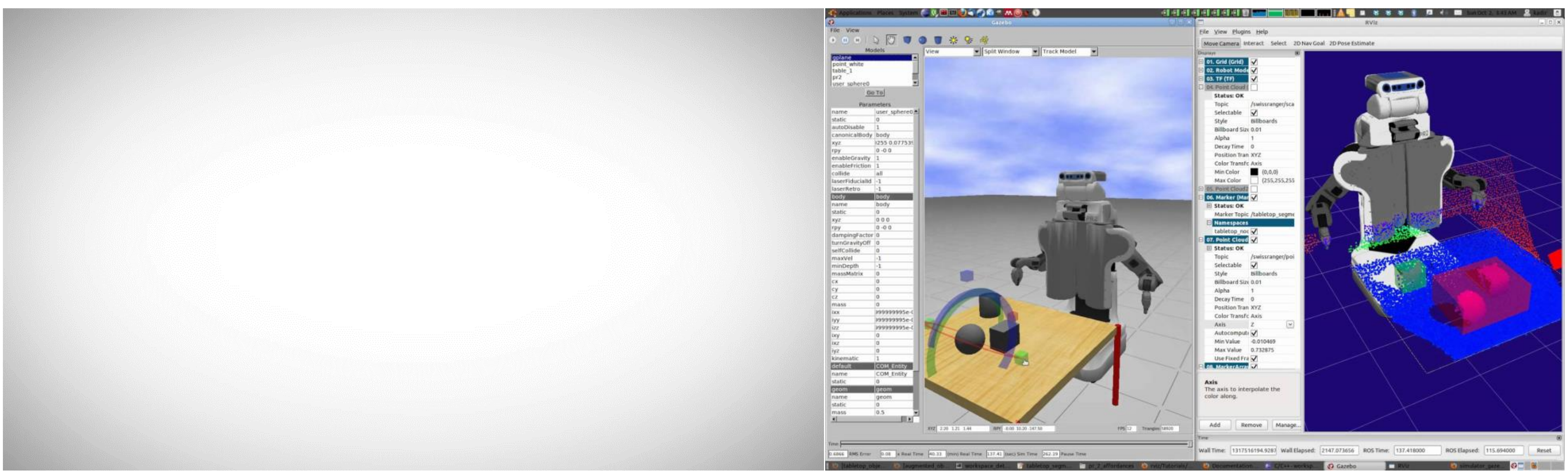
- Broad overview of some open HRI problems

ROS Installation

Any problems?

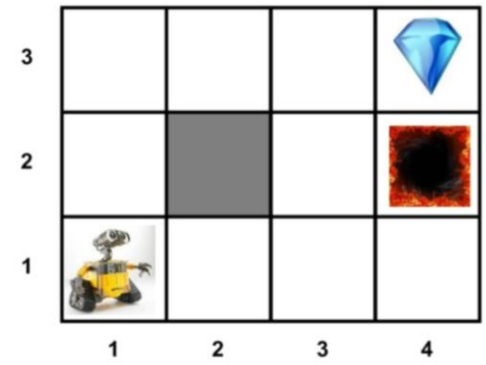
Next steps:

http://moveit.ros.org/ http://gazebosim.org/



Last Time…

Sample Problem Terminology



A **state** is a representation of the world

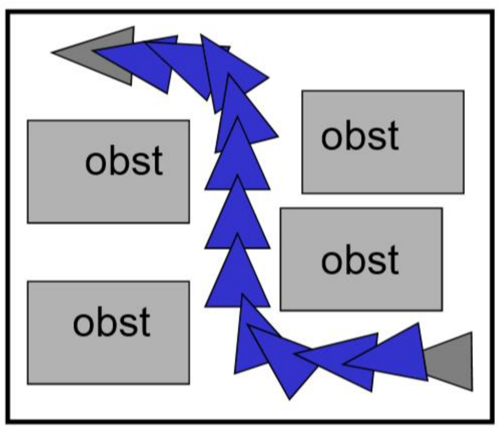
An **action** is something that transitions you from one state to another (can also be a self-transition!)

A **transition function** T(s,a,s’) provides the probability that a particular action **a** taken in a particular state **s** will bring the system to state **s’**

A **reward function** R(s, a) provides the value of taking a particular action **a** in state **s**

Non-trivial Robots in C-Space

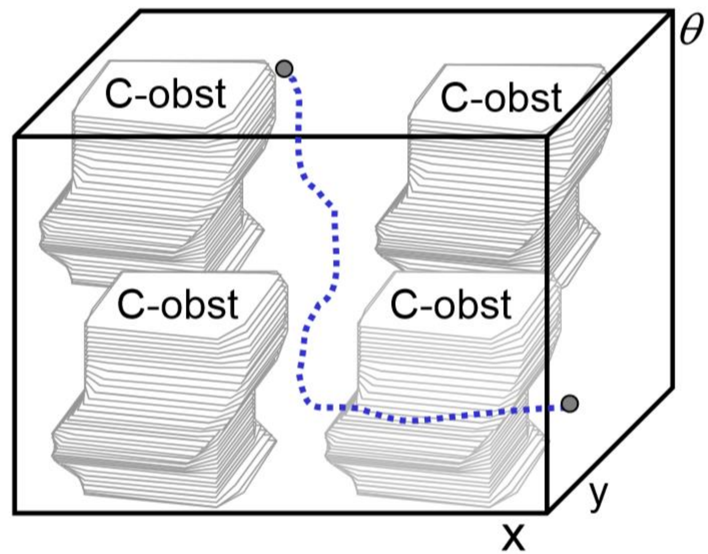
Workspace
(x, y)



Robot

Path is hard to express

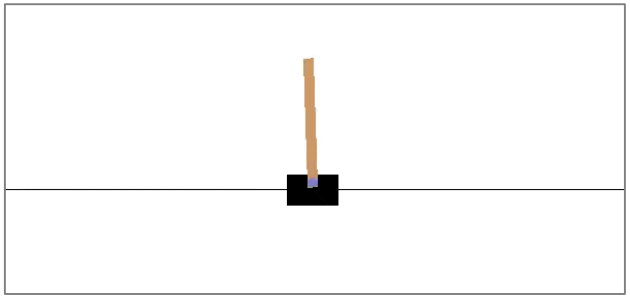
C-space
(x, y, θ)

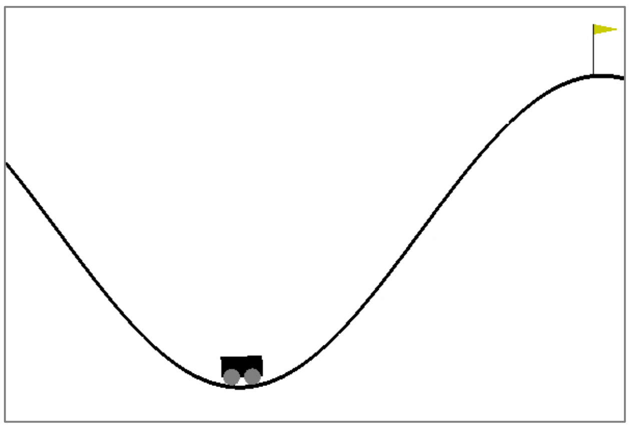


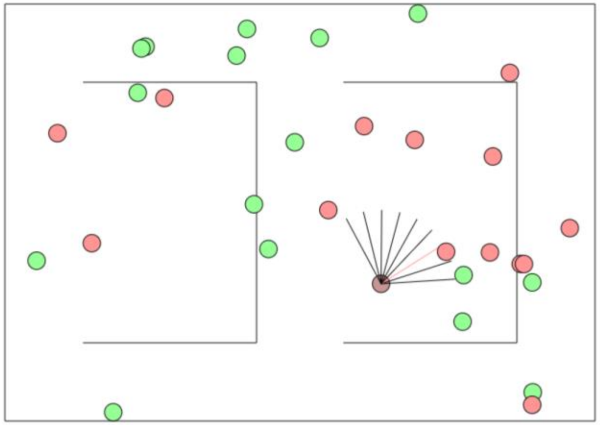
Robot

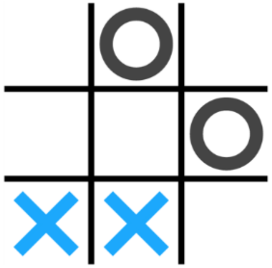
Path is just a space curve

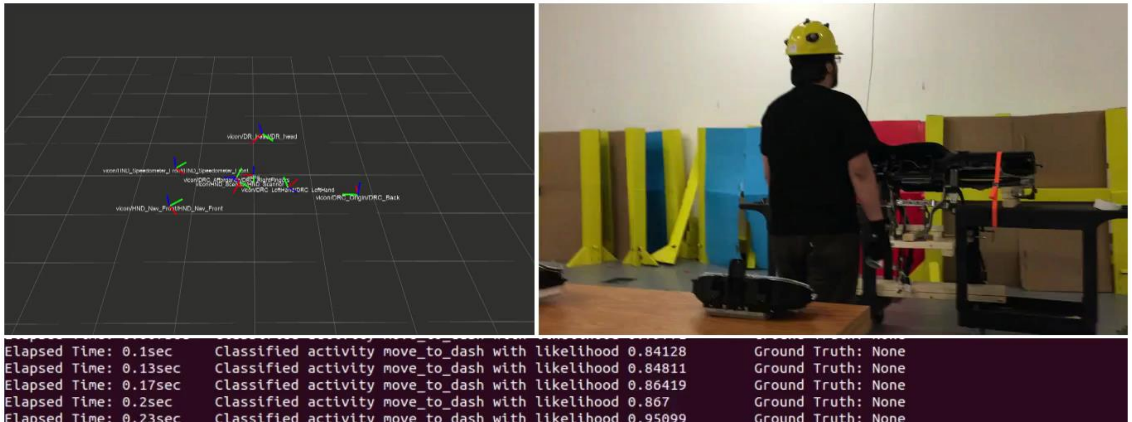
State Representation is Critical



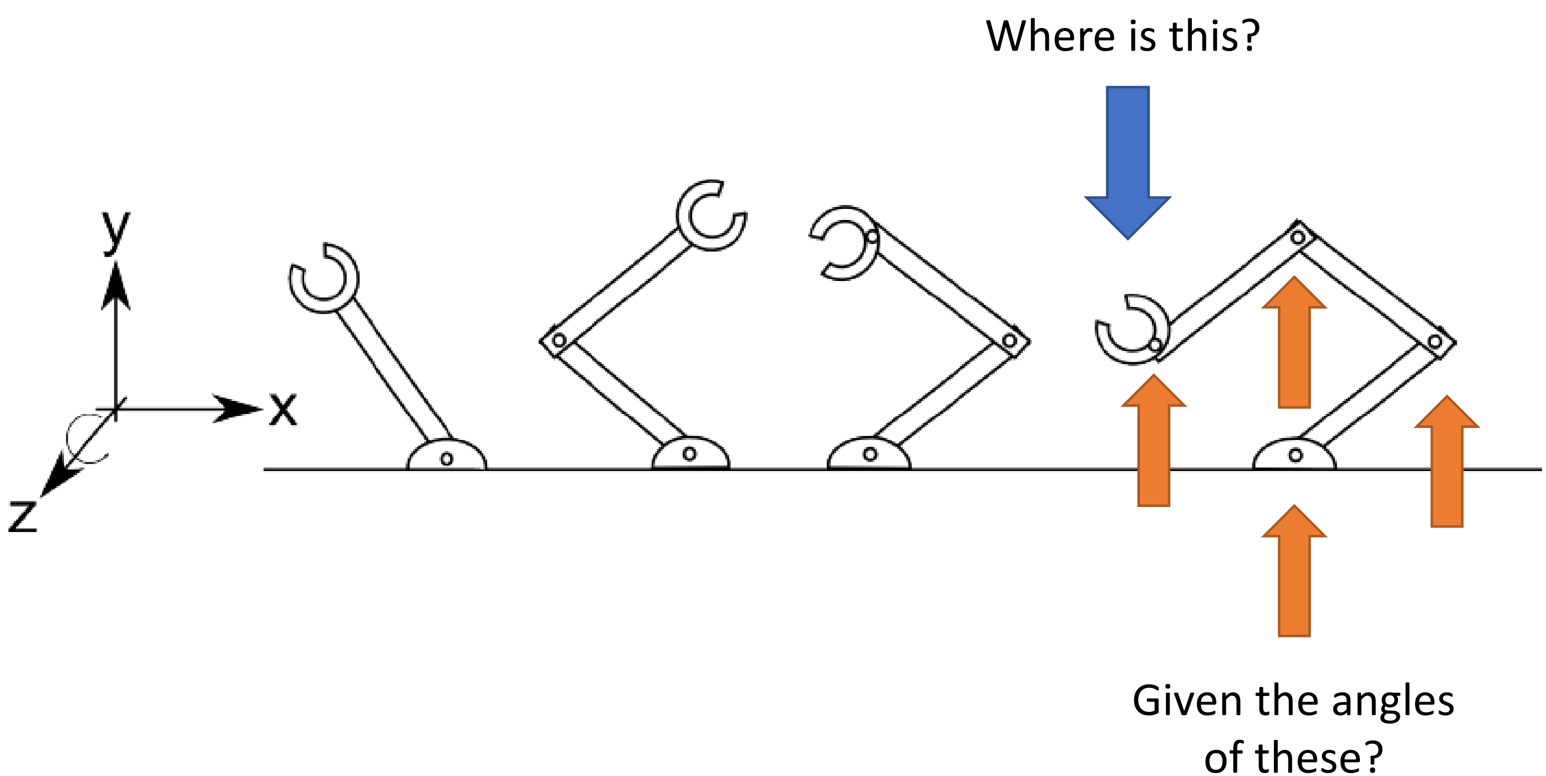




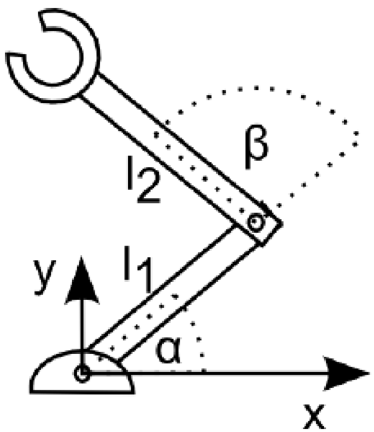


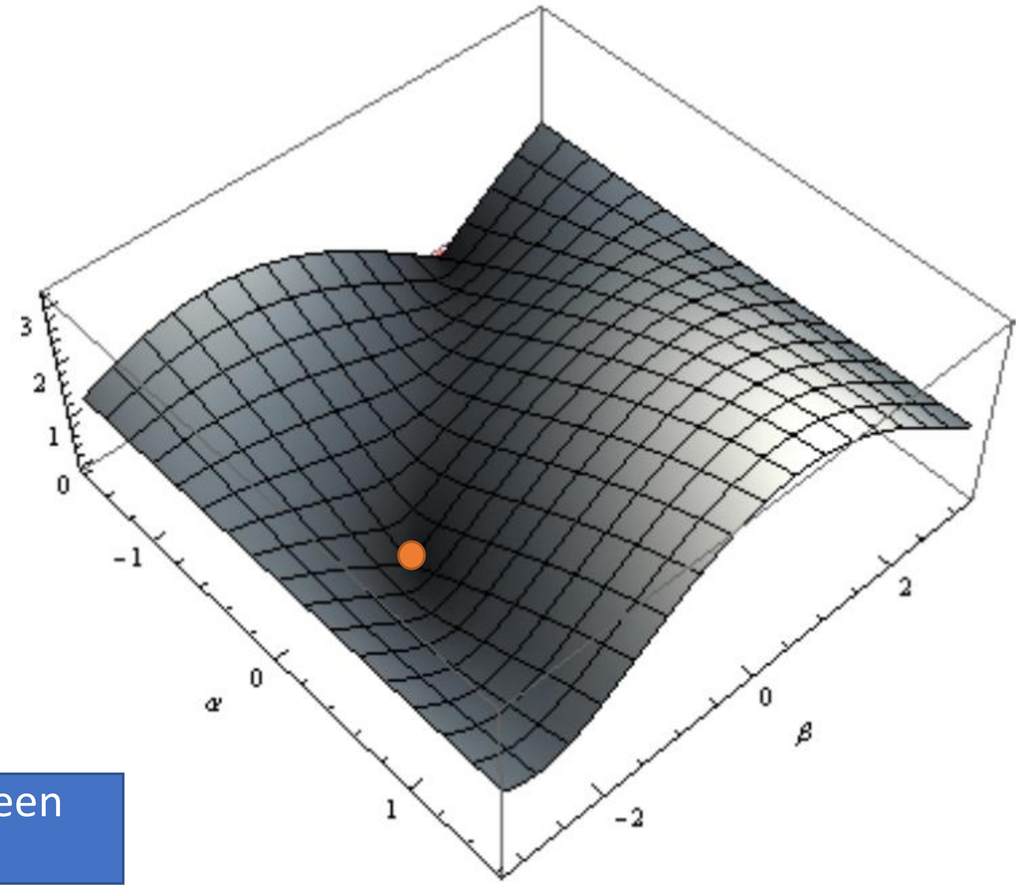


Forward Kinematics



Easier ways to solve the IK problems

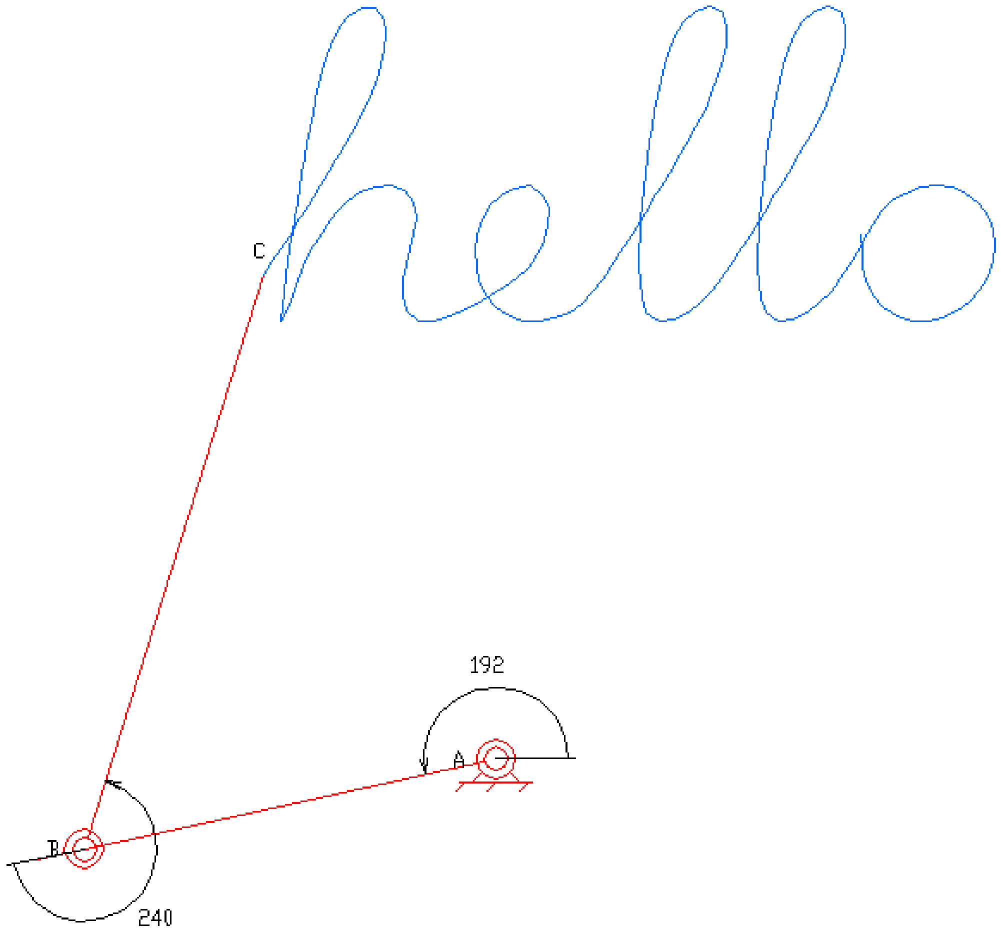




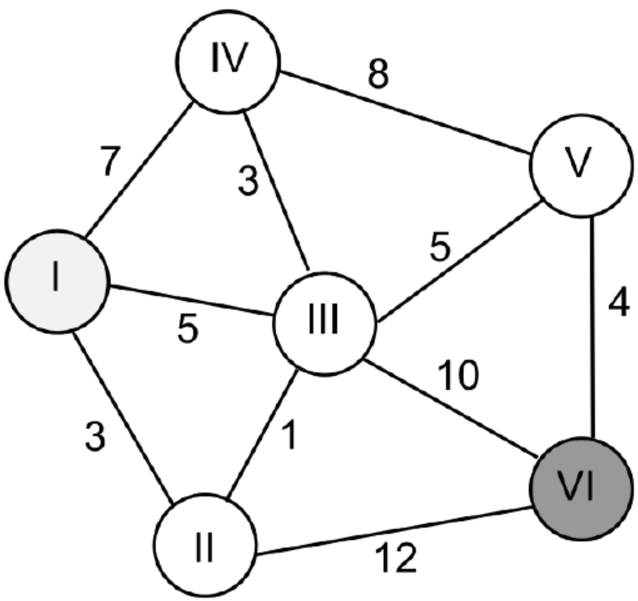
Just the Euclidean distance between two vectors!

$$f_{x,y}(\alpha,\beta) = \sqrt{(\sin(\alpha+\beta)l_2 + \sin(\alpha)l_1 - y)^2 + (\cos(\alpha+\beta)l_2 + cos(\alpha)l_1 - x)^2}$$

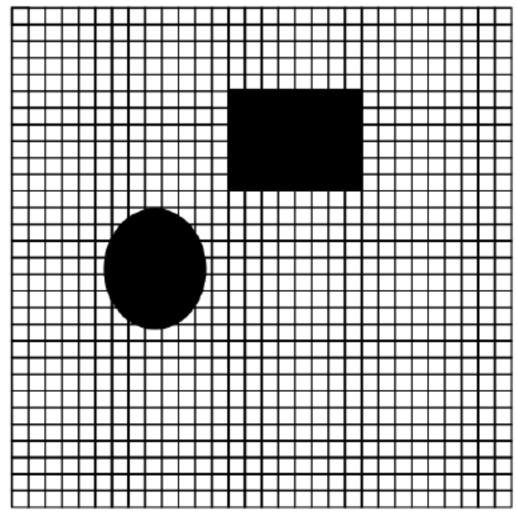
Motivating Problem



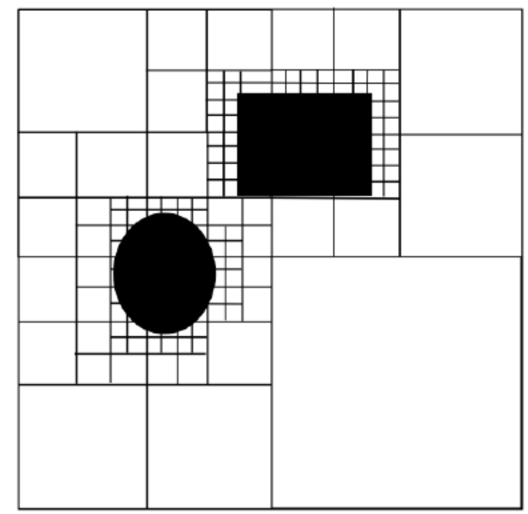
Map Representations



Topological Map
(Continuous Coordinates)



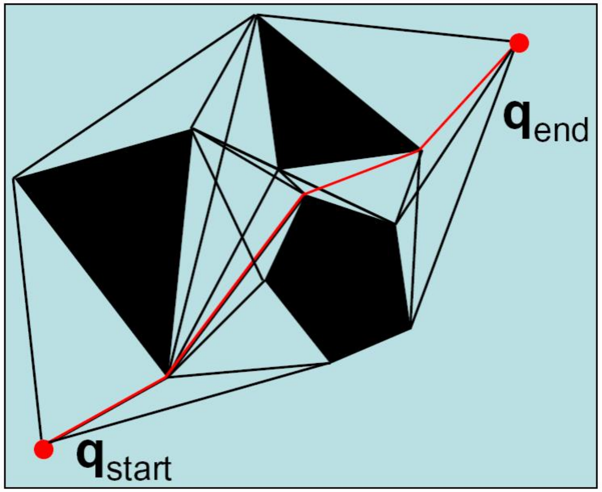
Grid Map
(Discrete Coordinates)



K-d Tree Map (Quadtree)

Some Well-Known Representations

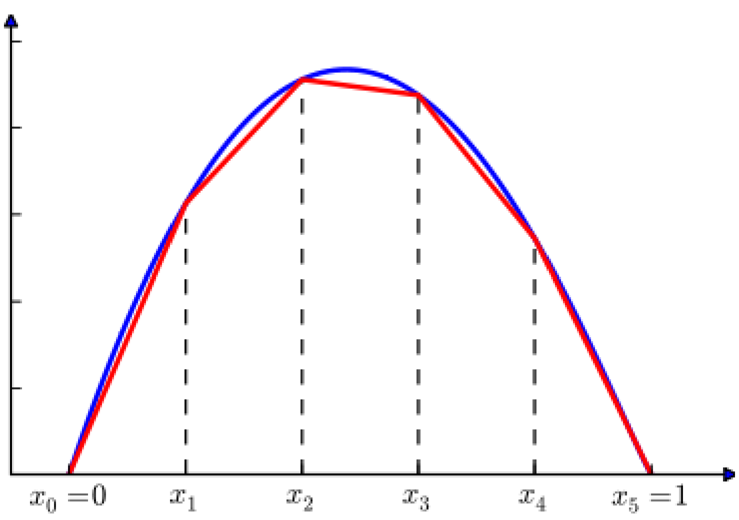
- Visibility Graphs
- Roadmap
- Cell Decomposition
- Potential Field



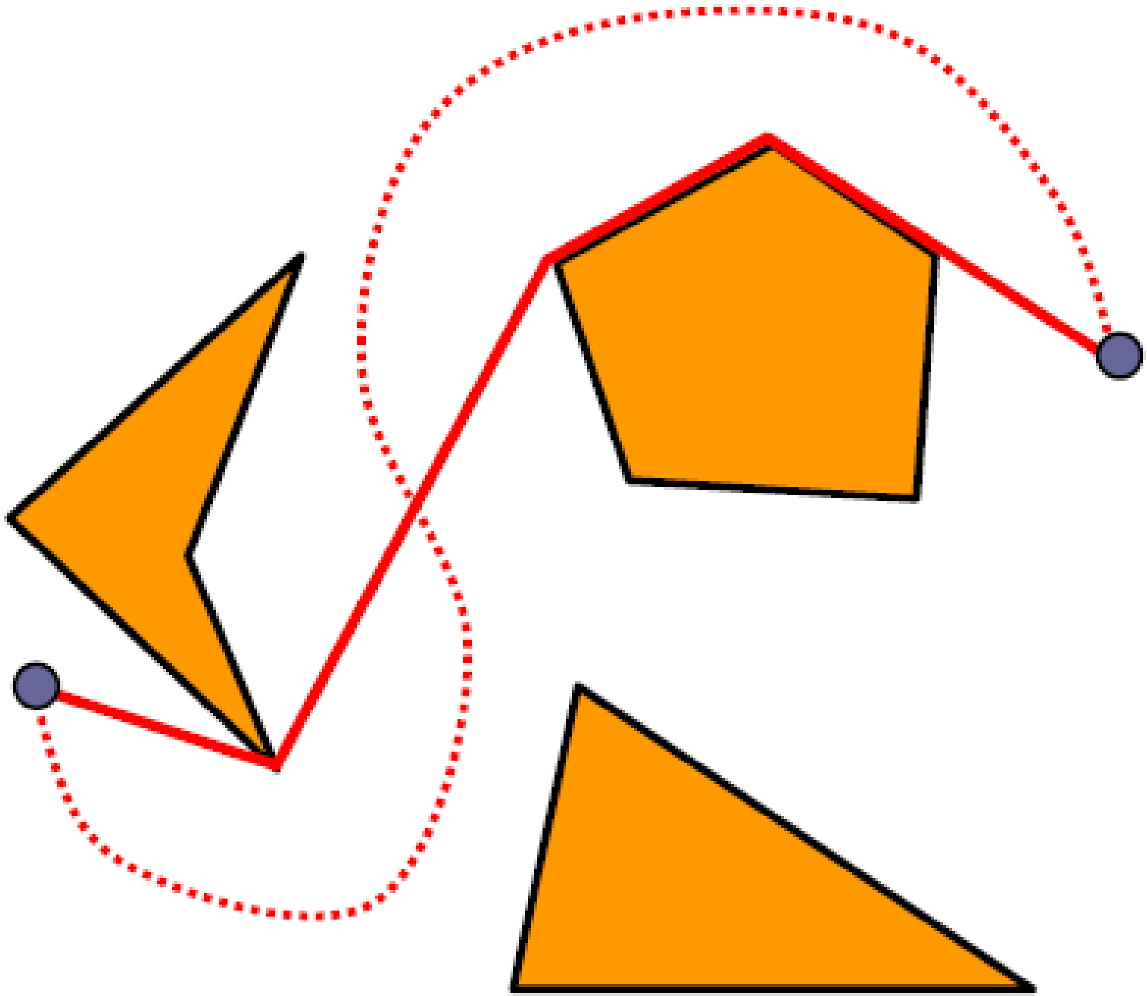
Visibility Graph Method

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:



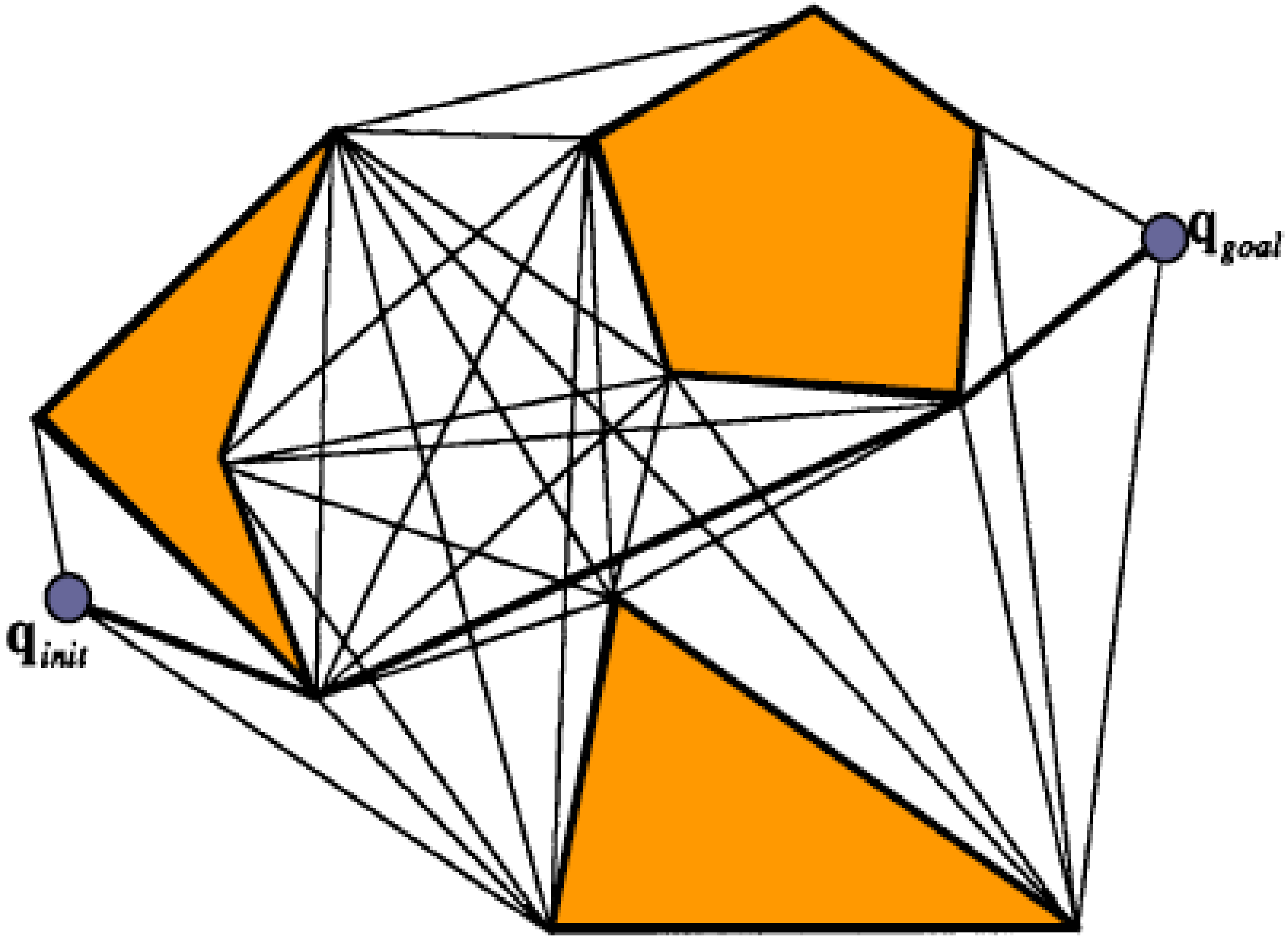
Visibility Graph Method



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.



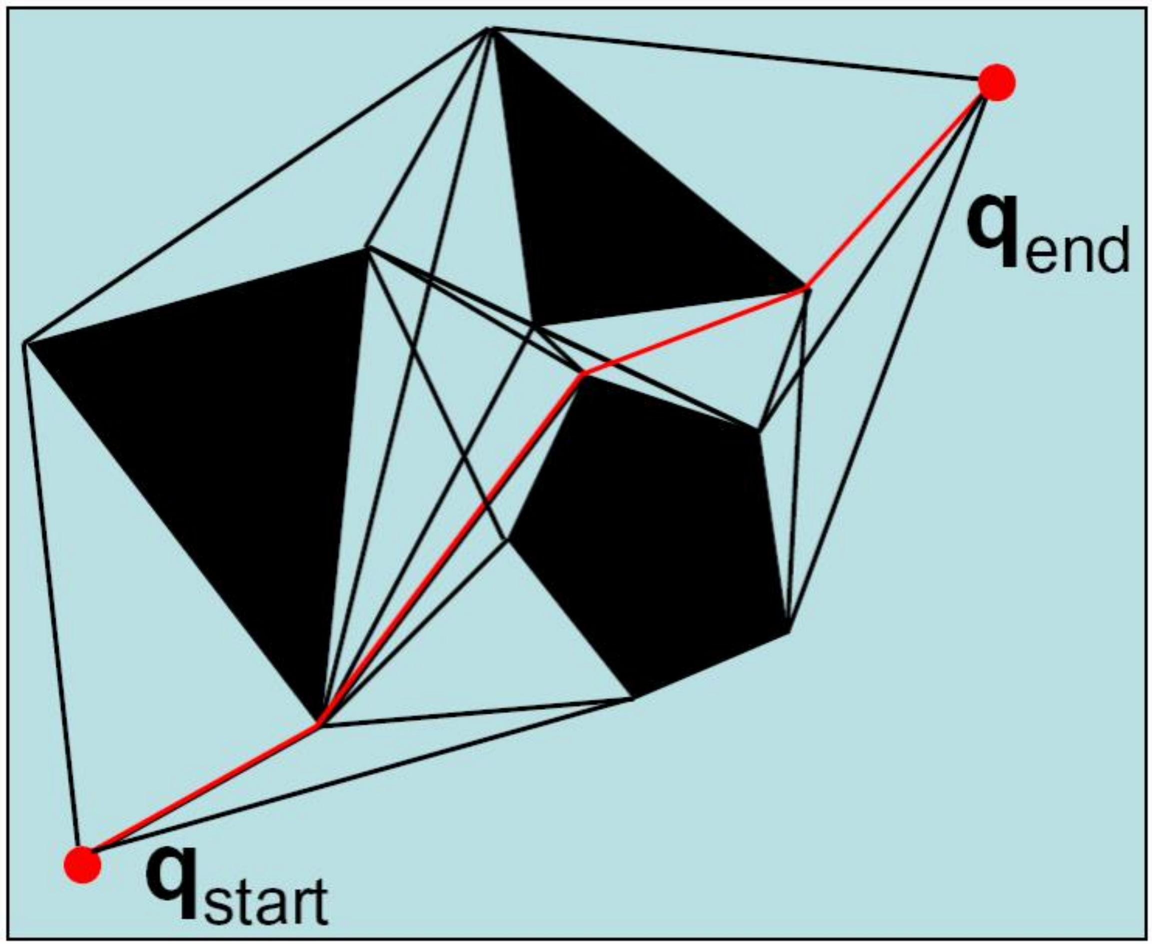
$\mathbf{q}_{goal}$
$\mathbf{q}_{init}$

Visibility Graph Example

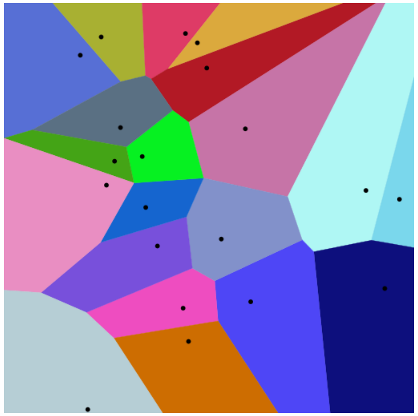
Input: q_{init}, q_{goal}, polygonal obstacles
Output: visibility graph G

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

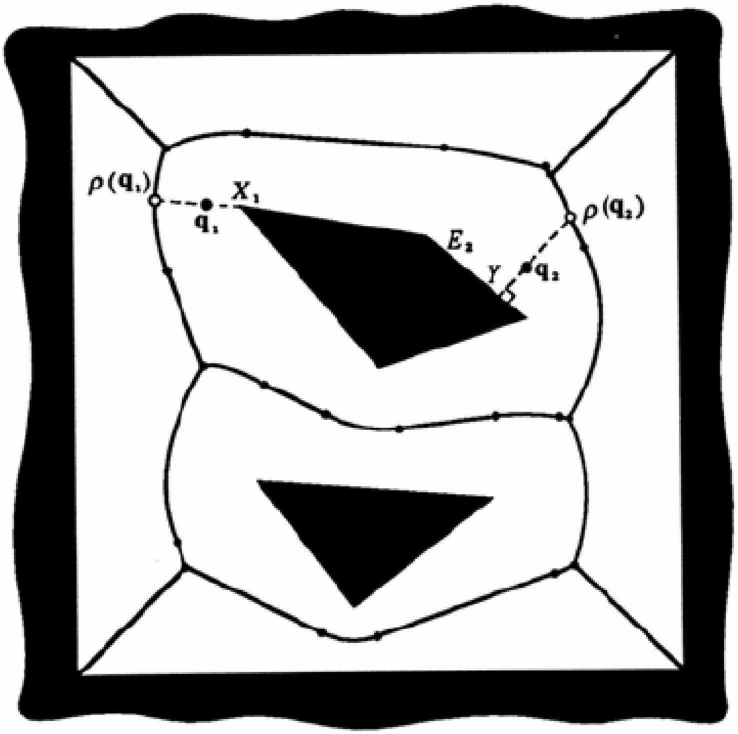
Visibility Graph: Strengths/Weaknesses?



Road mapping Technique

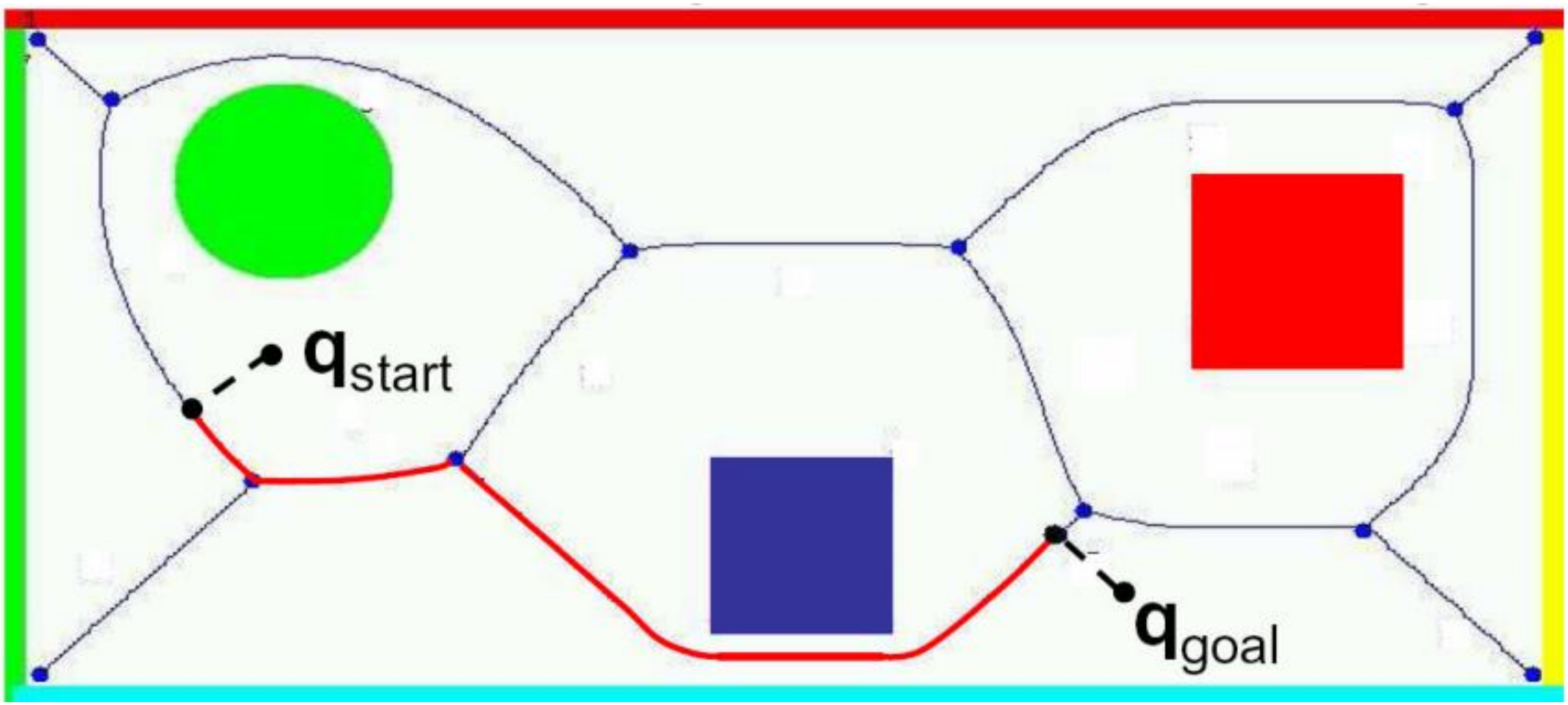


Voronoi Diagram



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

Voronoi Road Mapping: Strengths / Weaknesses



Exact cell decomposition

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

Approximate cell decomposition

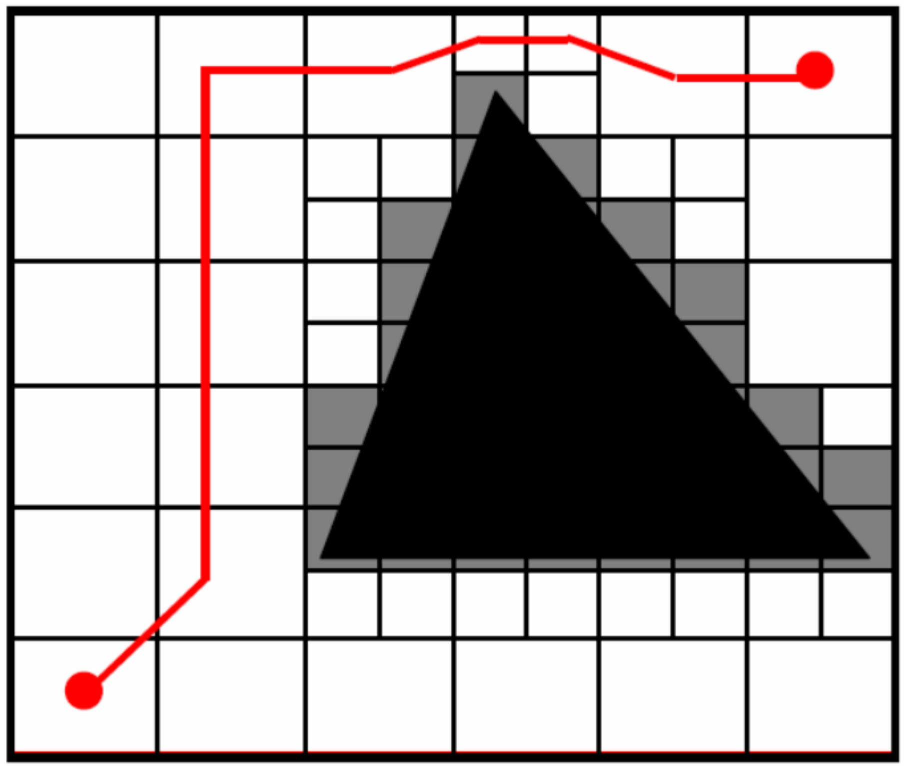
- ***F*** is represented by a collection of non-overlapping cells whose union is contained in ***F*.**
- Cells usually have simple, regular shapes, e.g., rectangles, squares.
- Facilitates hierarchical space decomposition

Cell Decomposition Methods

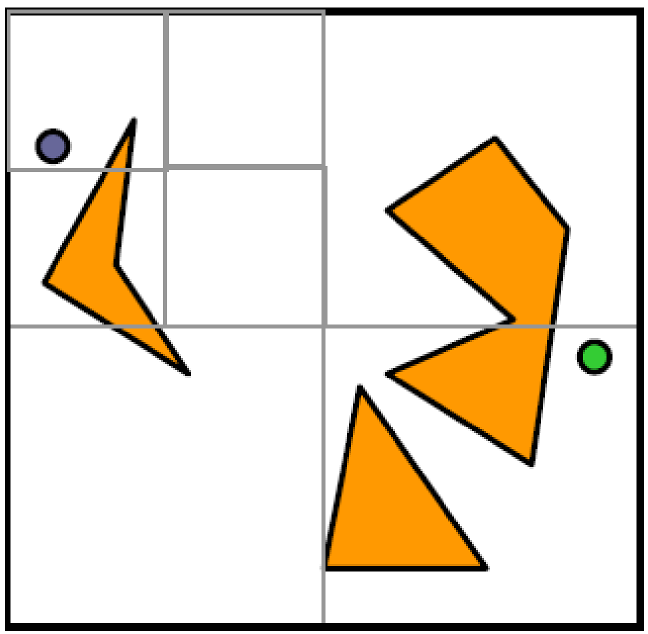
Cell Decomposition

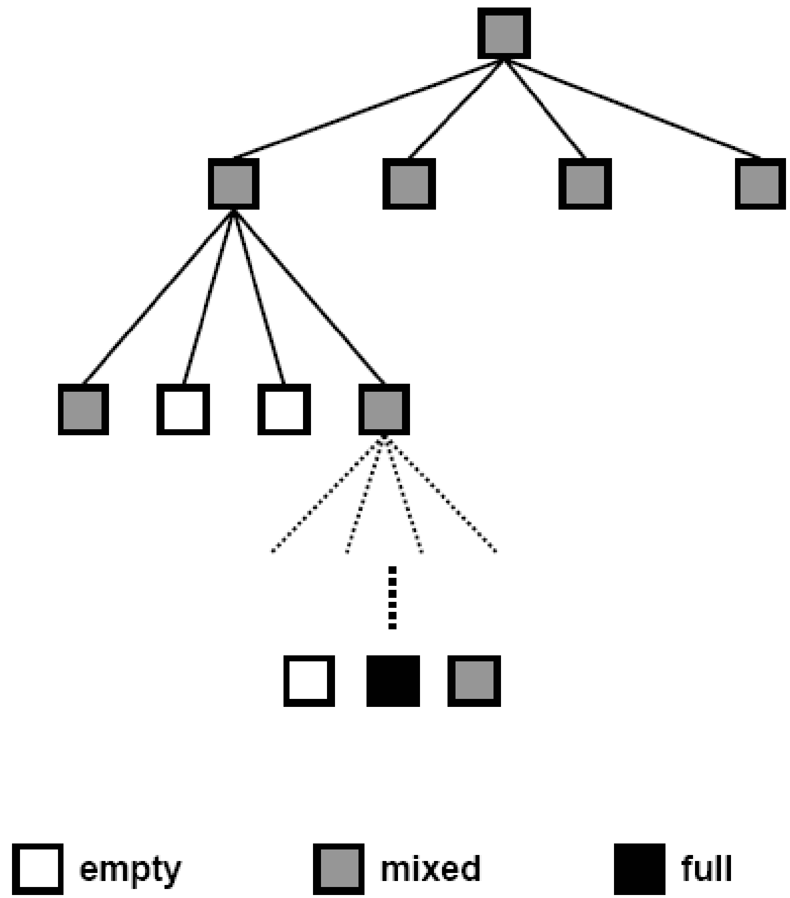
Not necessarily *complete*

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

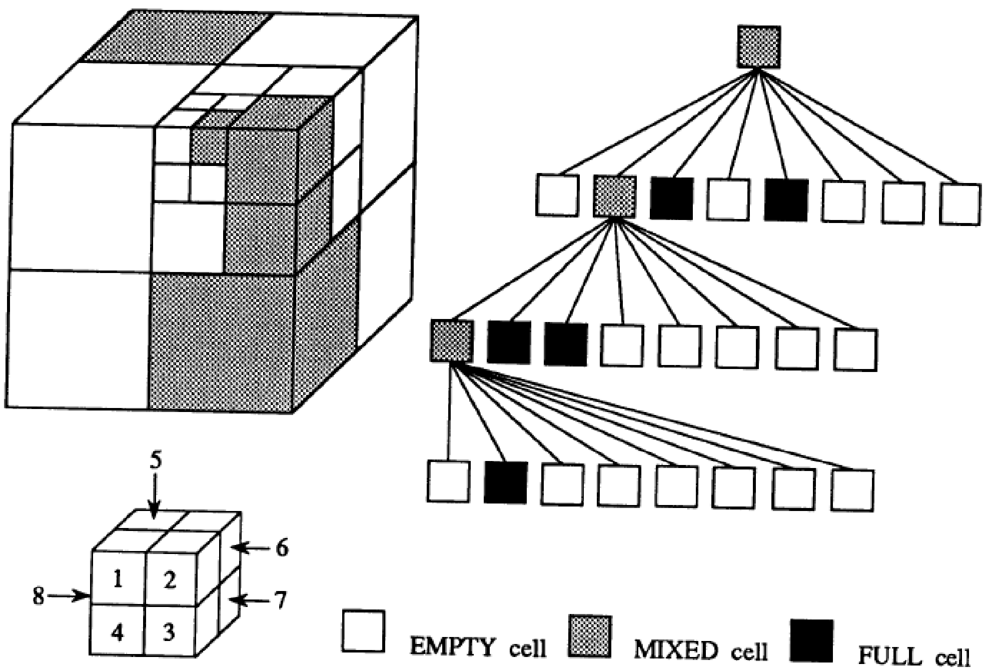


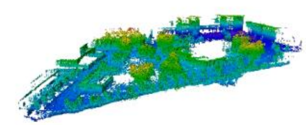
Quadtree Decomposition





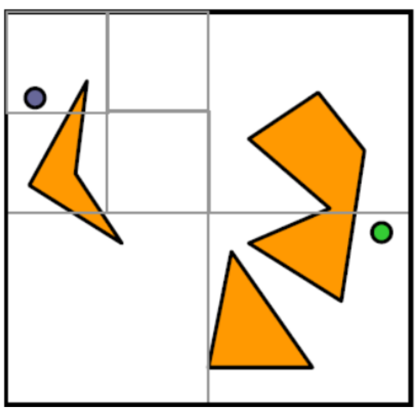
Octree Decomposition



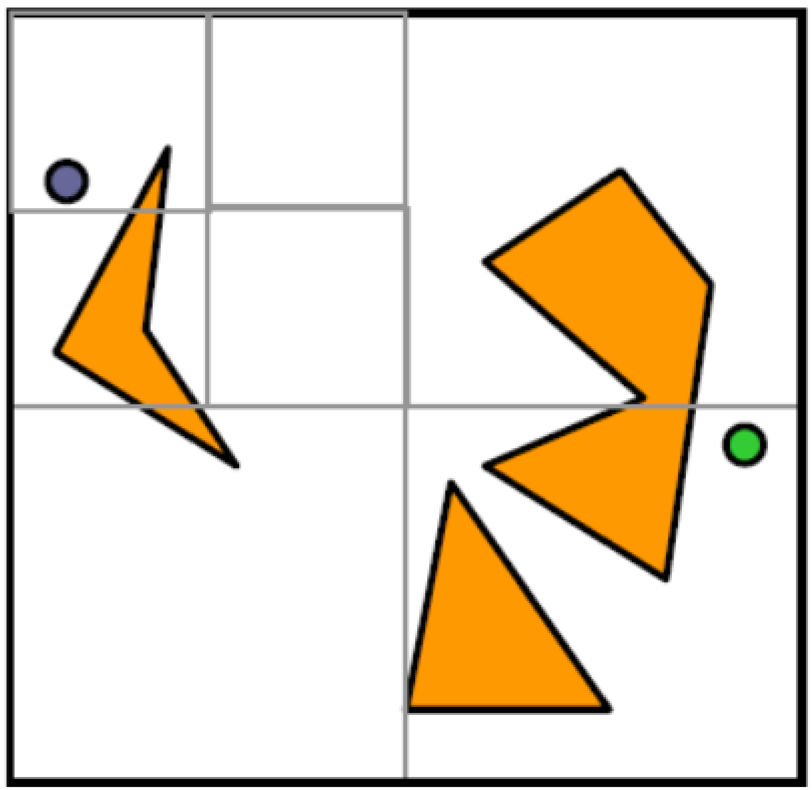


Cell Decomposition Path Planning Algorithm Outline

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.

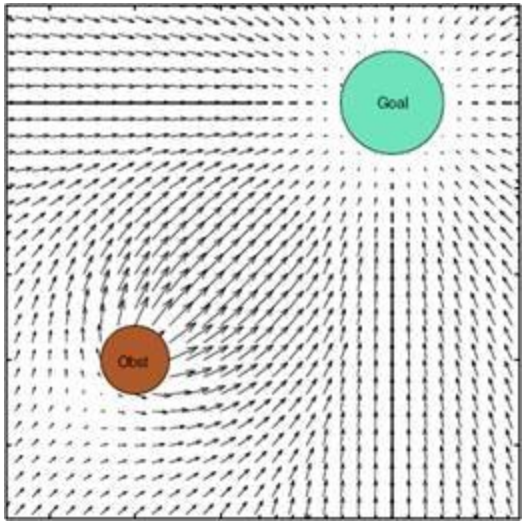


Cell Decomposition: Strengths/Weaknesses

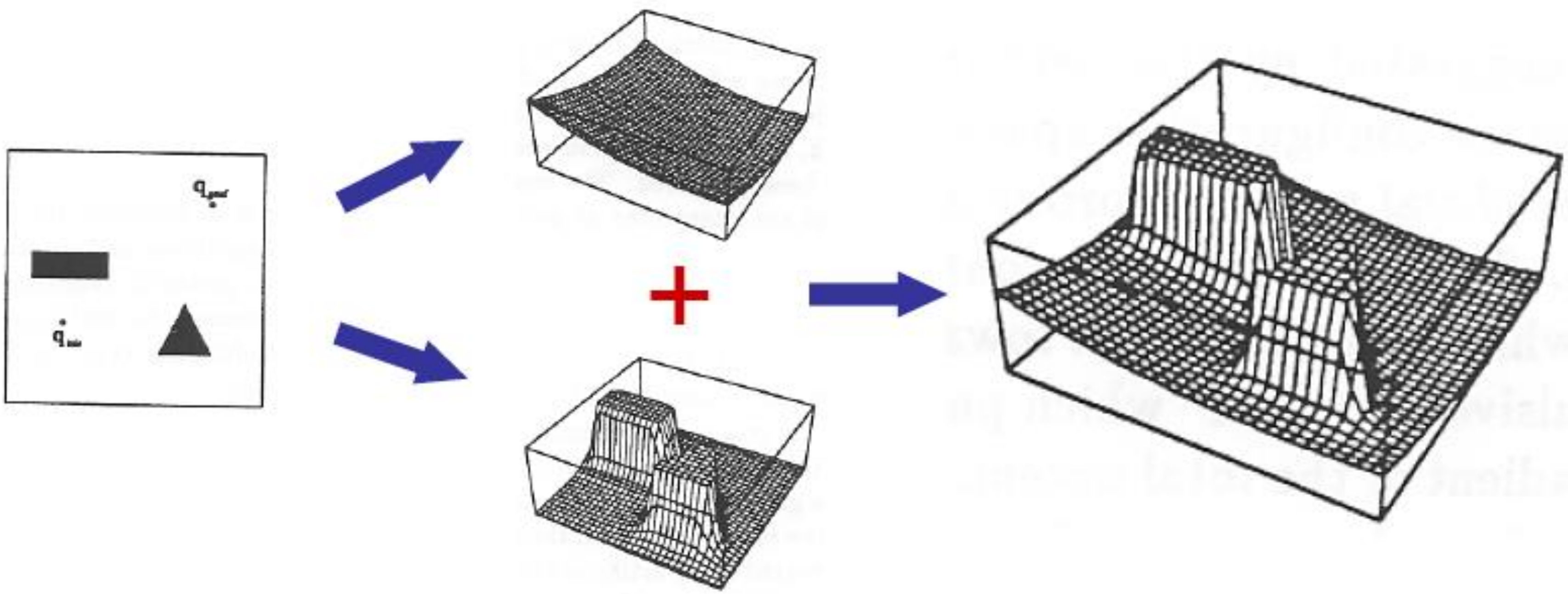


Potential Fields

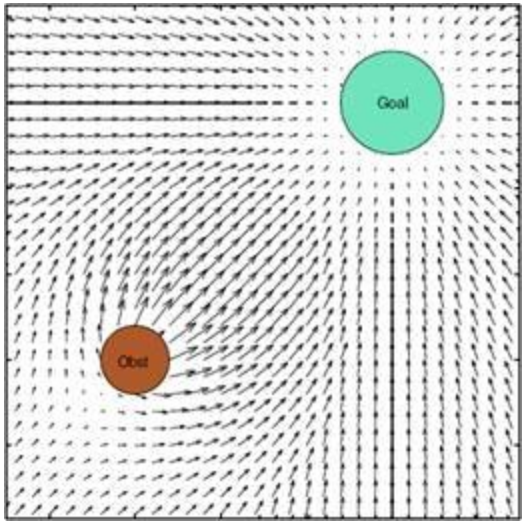
- Initially proposed for real-time collision avoidance [Khatib 1986].
- A potential field is a scalar function over the free space.
- To navigate, the robot applies a force proportional to the negated gradient of the potential field.
- A navigation function is an ideal potential field that
  - has global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



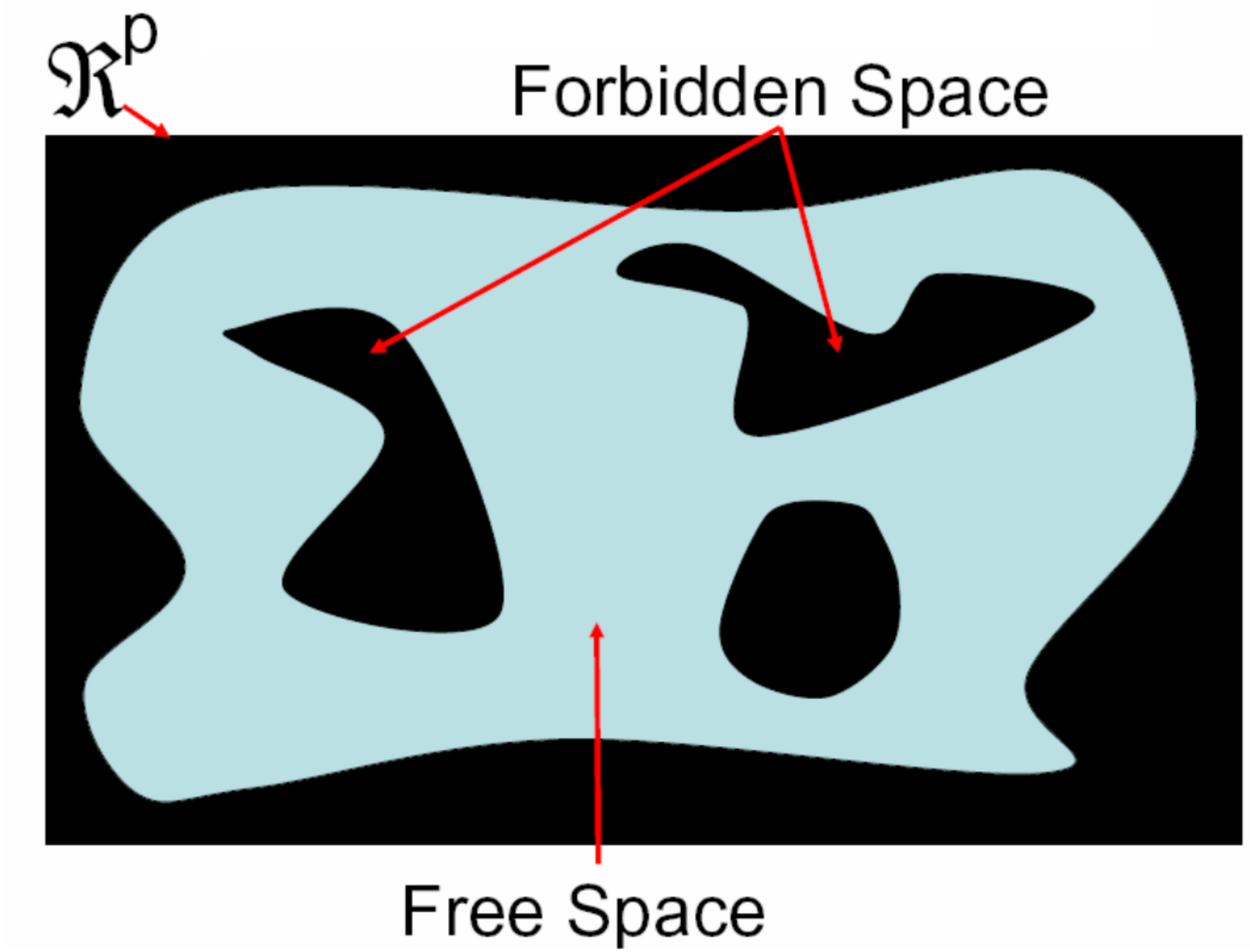
How Does It Work?



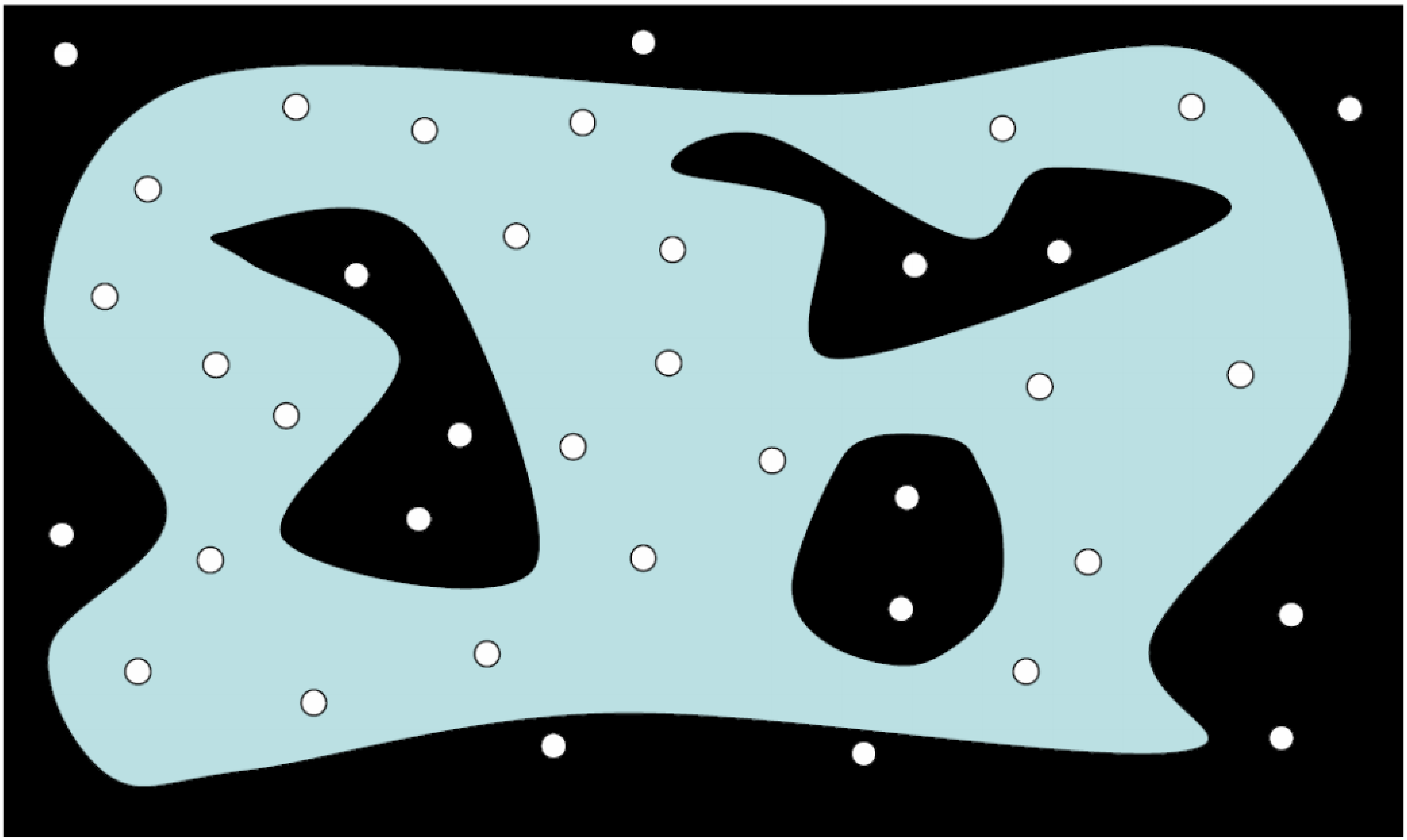
Potential Fields: Strengths/Weaknesses



Probabilistic Road Map (PRM)

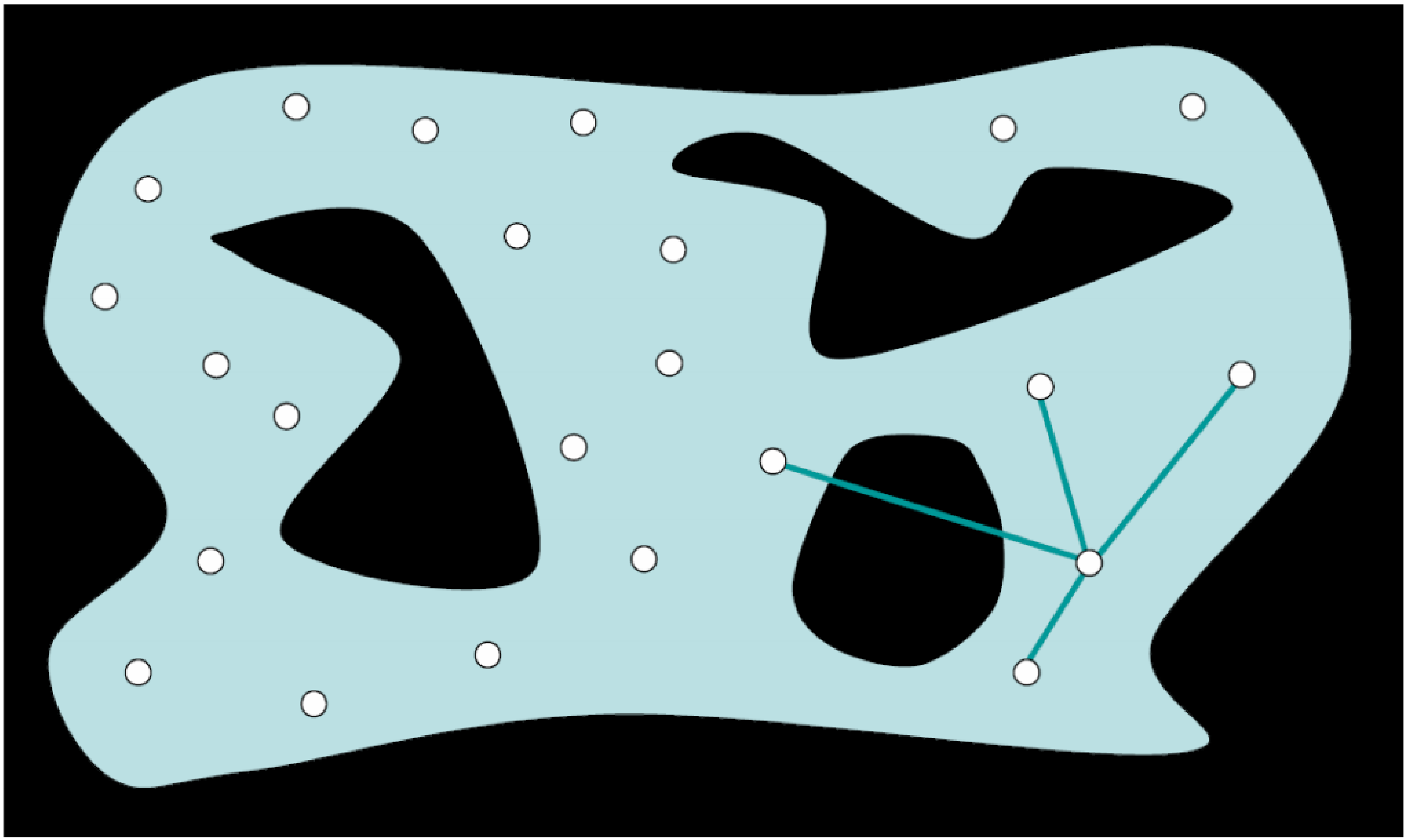


Probabilistic Road Map (PRM)



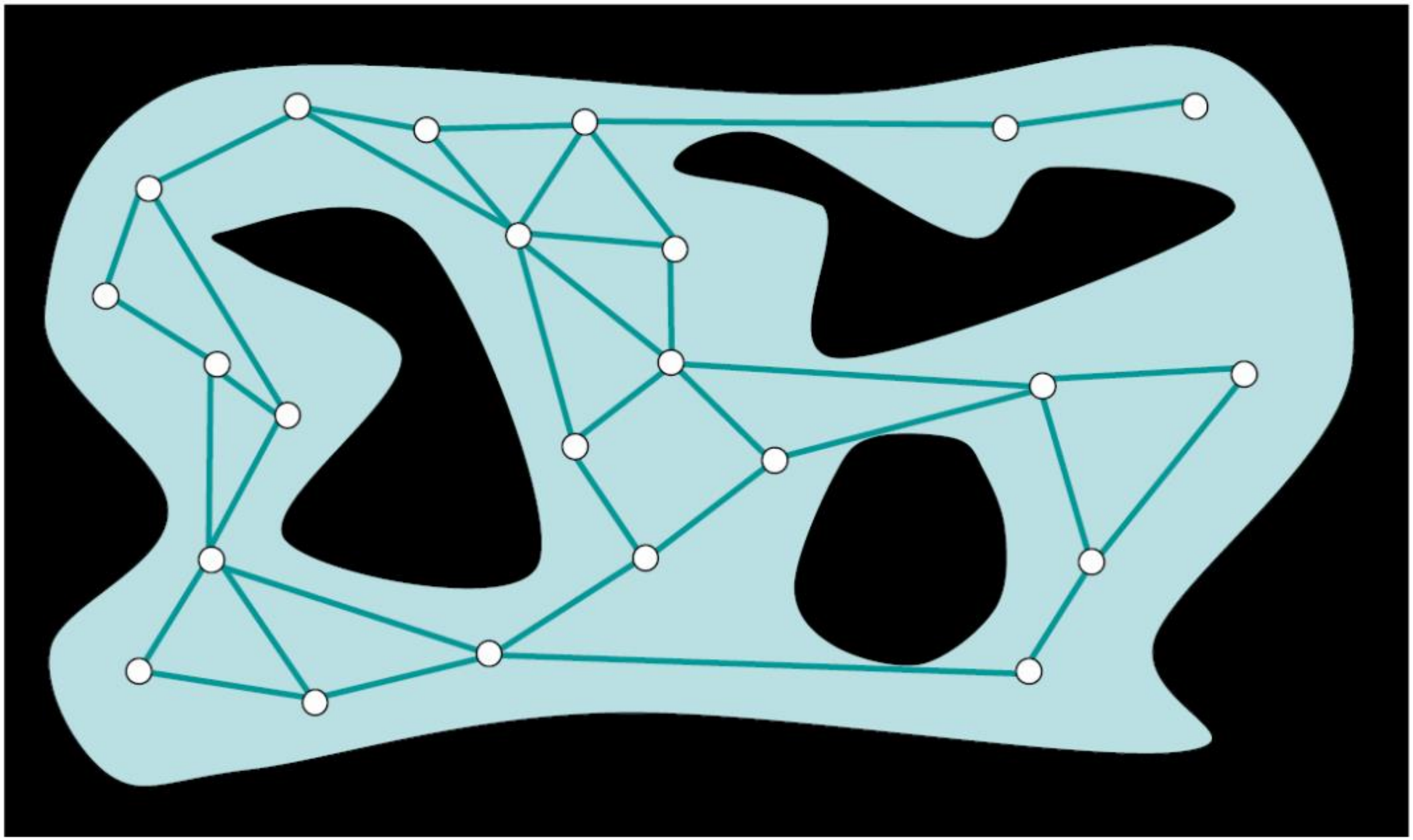
Sample random locations!

Probabilistic Road Map (PRM)



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

Probabilistic Road Map (PRM)

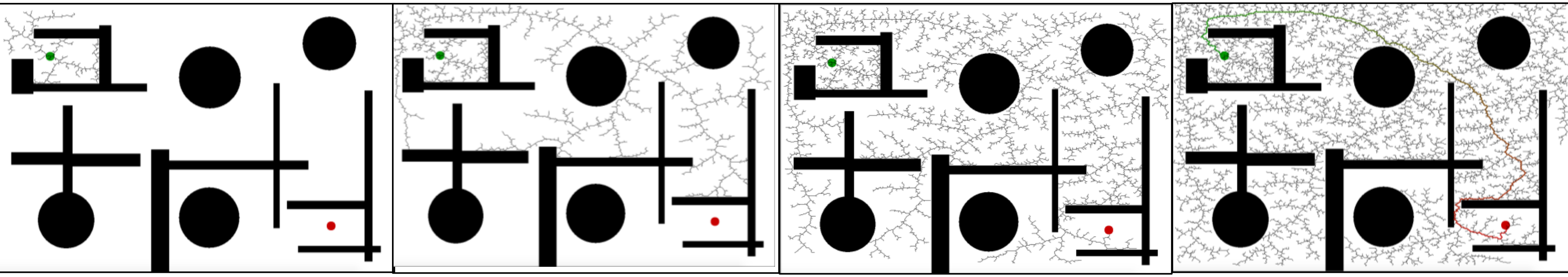


Remove edges crossing forbidden areas

How to sample points?

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

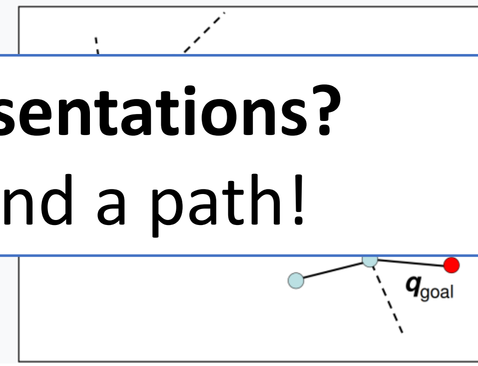
Rapidly-exploring Random Trees (RRT)



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

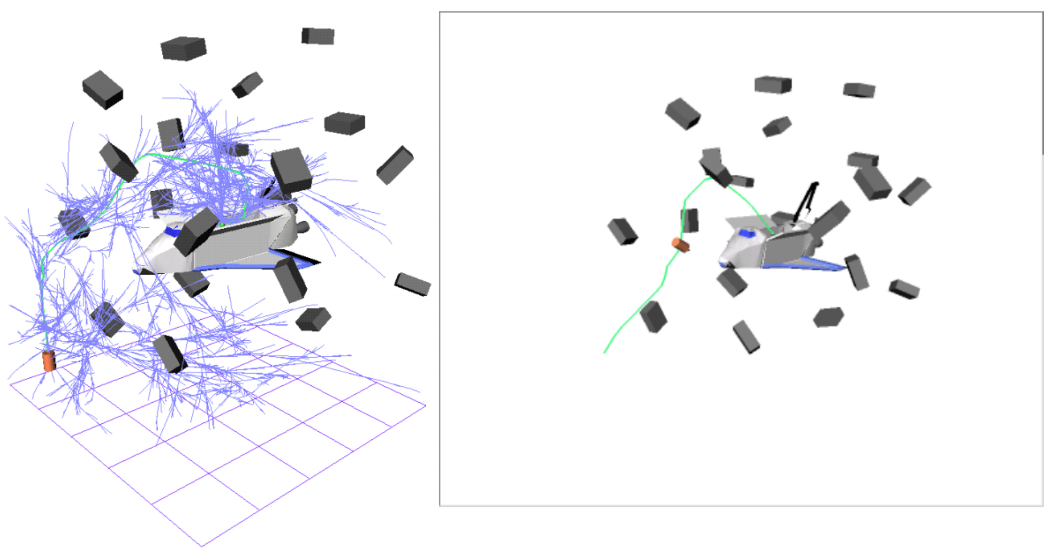
Handling Non-holonomic agents:

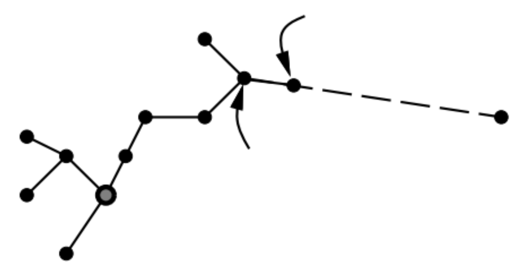
- Need an approximation of dynamics (a simulator)
- x_{new} is one ‘step’ forward in time (one action ahead of x_{near})

GENERATE_RRT($x_{init}, K, \Delta t$)

$\mathcal{T}$.init(x_{init});
for $k = 1$ **to** K **do**
    $x_{rand} \leftarrow$ RANDOM_STATE();
    $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
    $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
    $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
    $\mathcal{T}$.add_vertex(x_{new});
    $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
Return $\mathcal{T}$

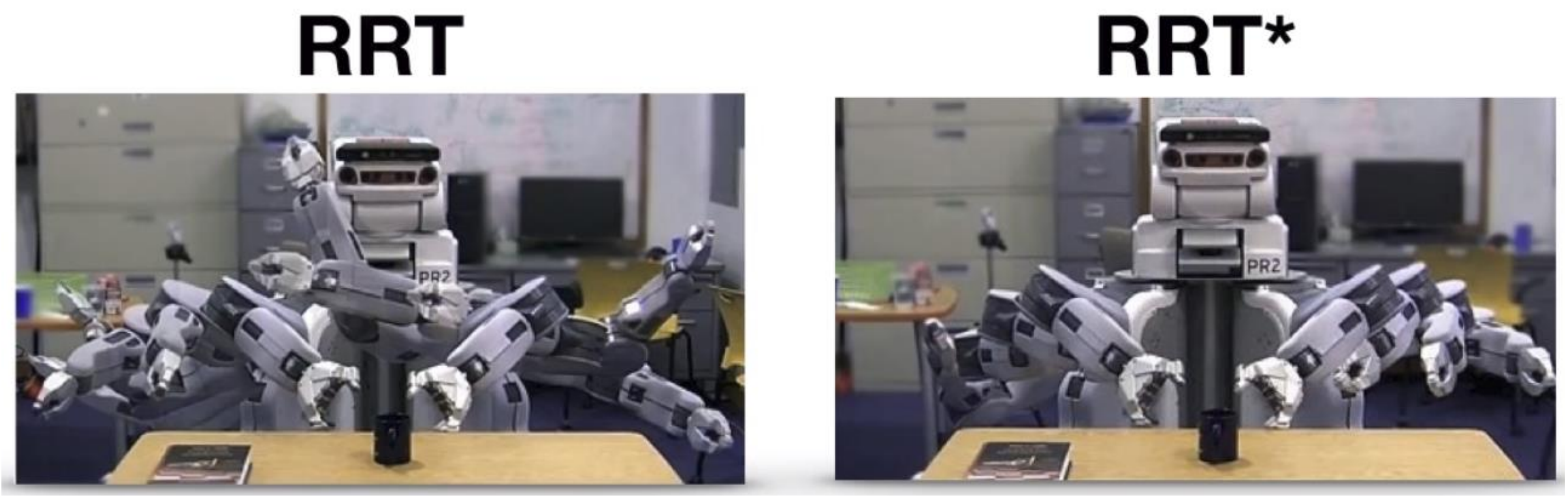
The result is a tree, $\mathcal{T}$, rooted at x_{init}.





RRT: Limitations

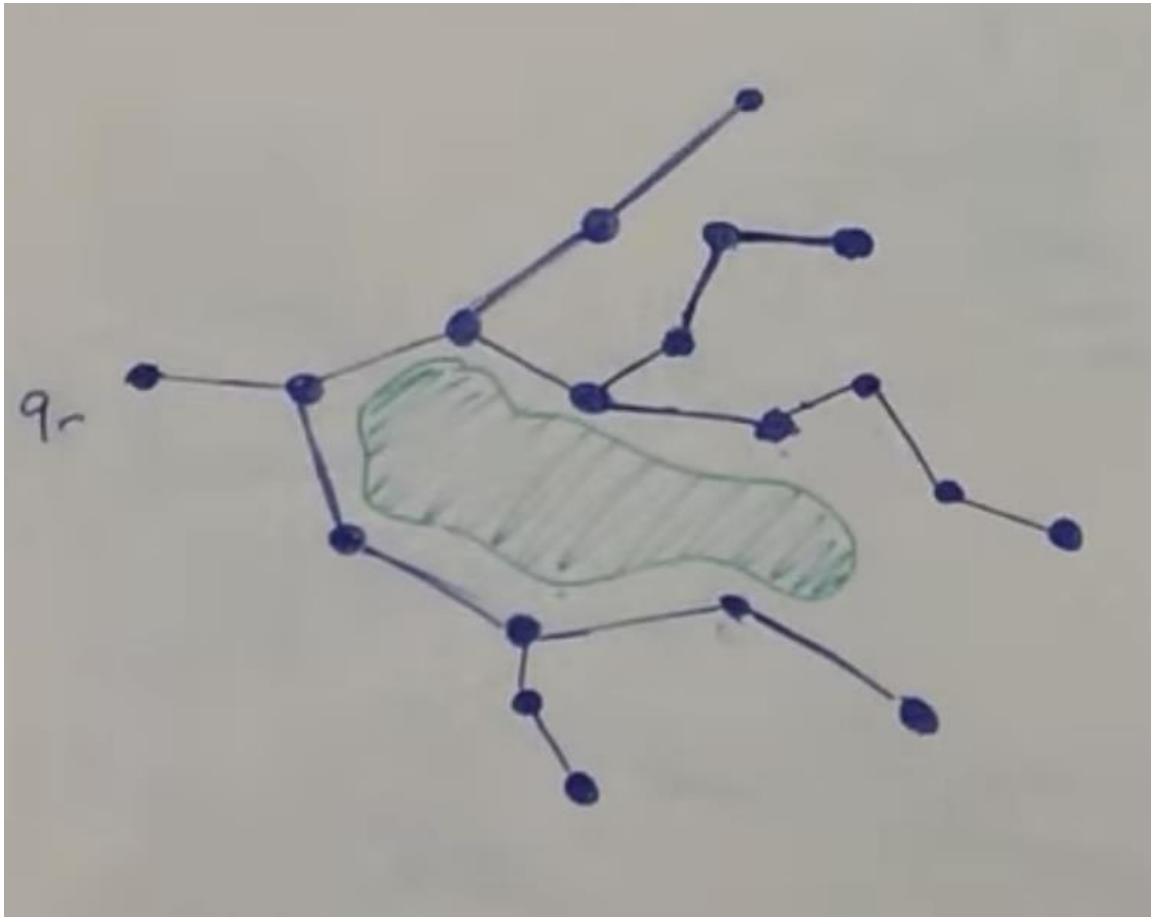
- RRT fails to converge to optimal solutions
  - Early solutions end up constraining the search
- RRT* guarantees asymptotic optimality (convergence to optimal solutions)
- RRT and RRT* require the same (asymptotic) computational resources



RRT*: Changes to RRT

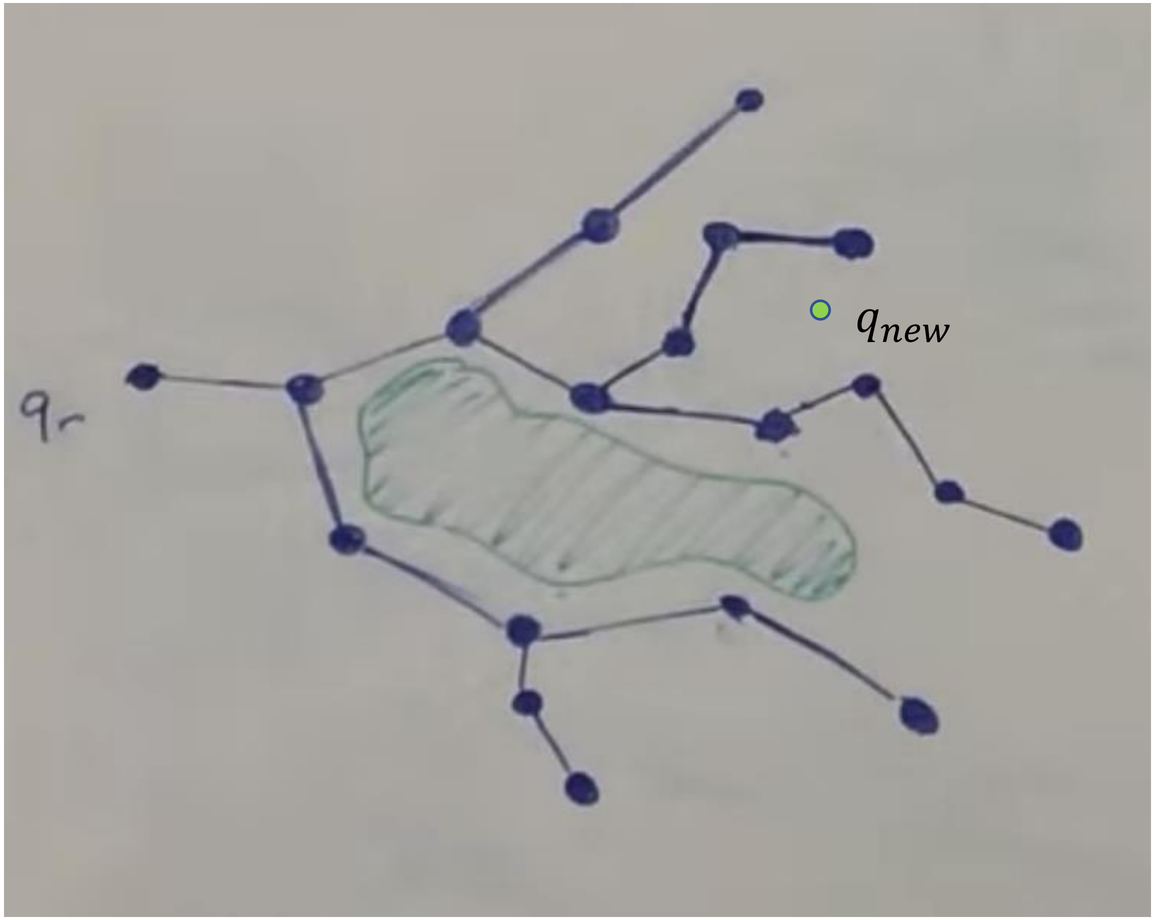
- Evaluate neighborhood around q_{new} instead of just picking $q_{nearest}$
  - Connect vertex from neighborhood that creates shortest path to q_{new} from q_{start}
- Re-wire network to optimize cost to get to q_new's neighbors
  - Compare costs:
    1) Shortest path from q_{start} to $q_{neighbor}$ via q_{new}
    2) Existing path from q_{start} to $q_{neighbor}$
  - If shortest path involves q_{new}, remove final edge of path #2 and add new edge between q_{new} and $q_{neighbor}$

RRT*



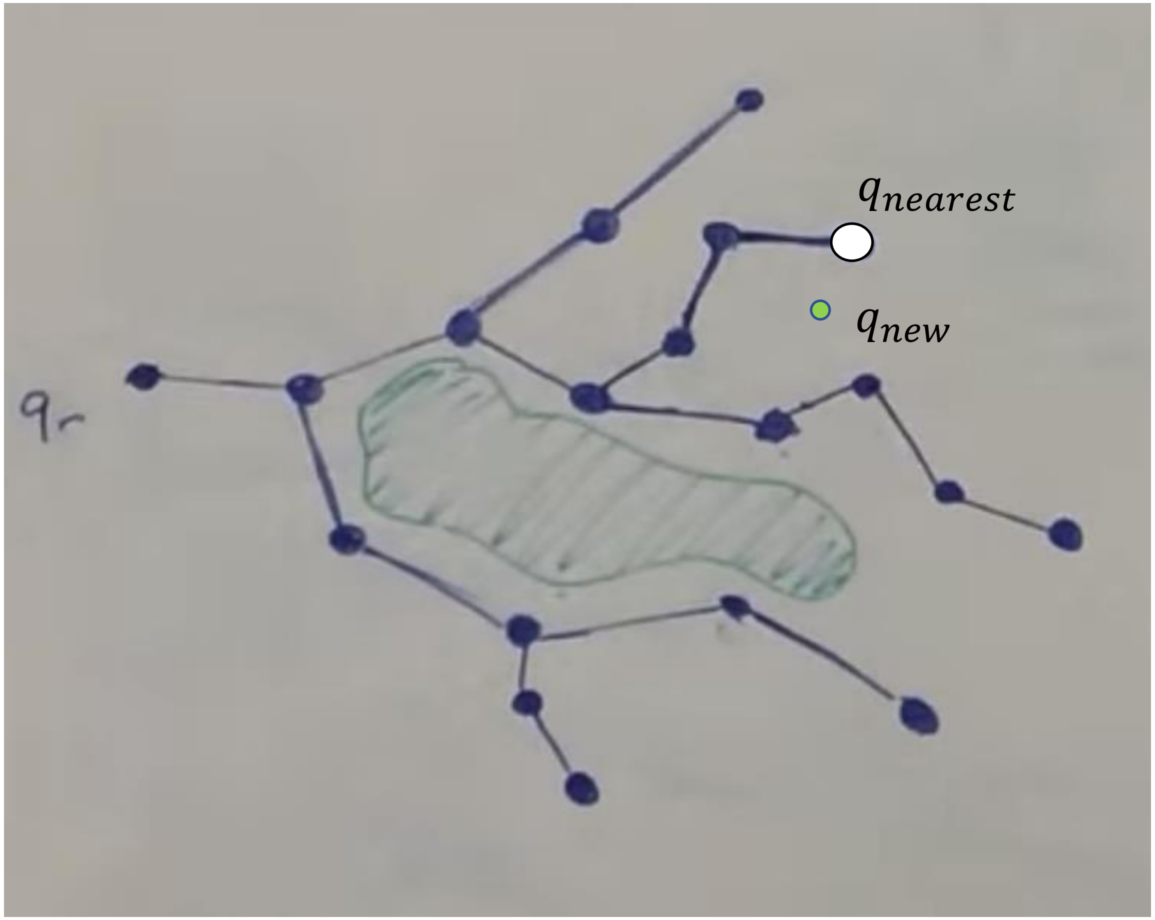
https://www.youtube.com/watch?v=JM7kmWE8Gtc

RRT*: Sample



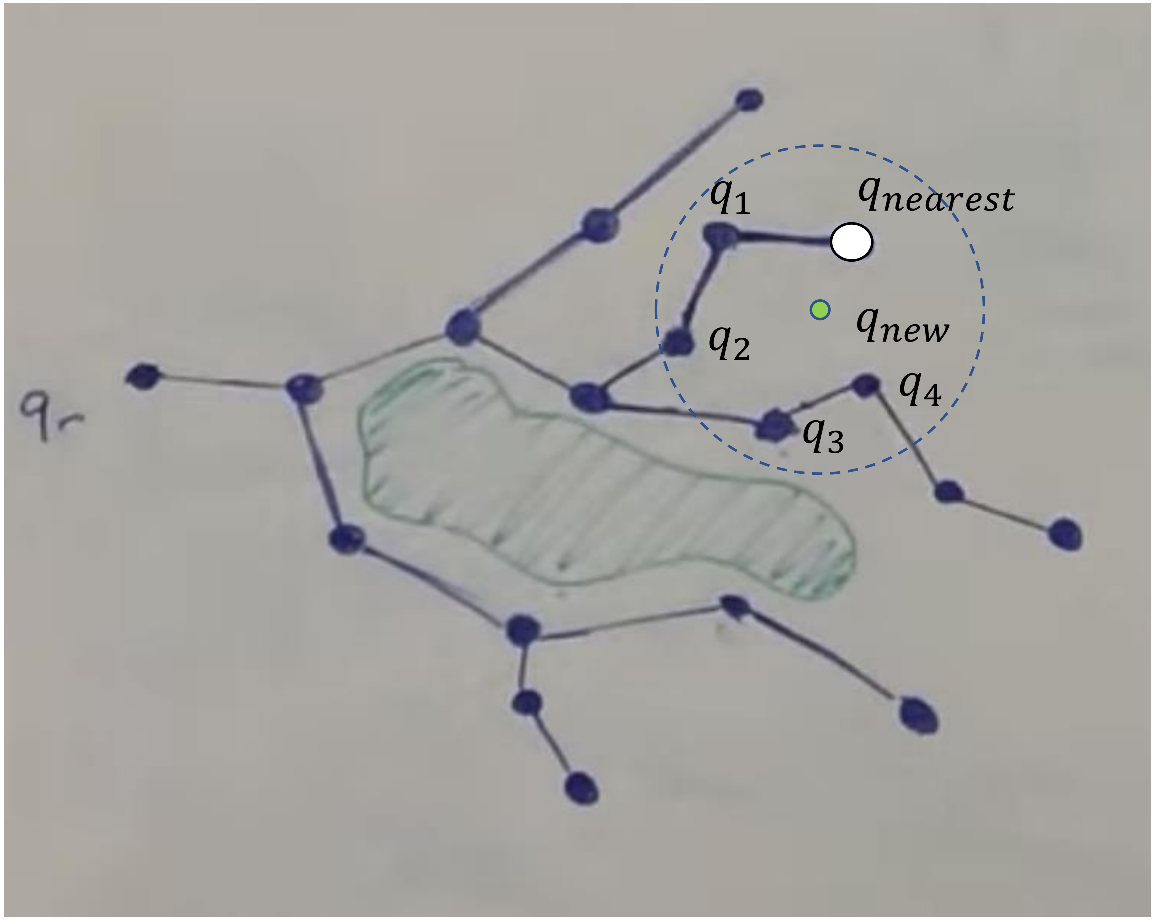
https://www.youtube.com/watch?v=JM7kmWE8Gtc

RRT*



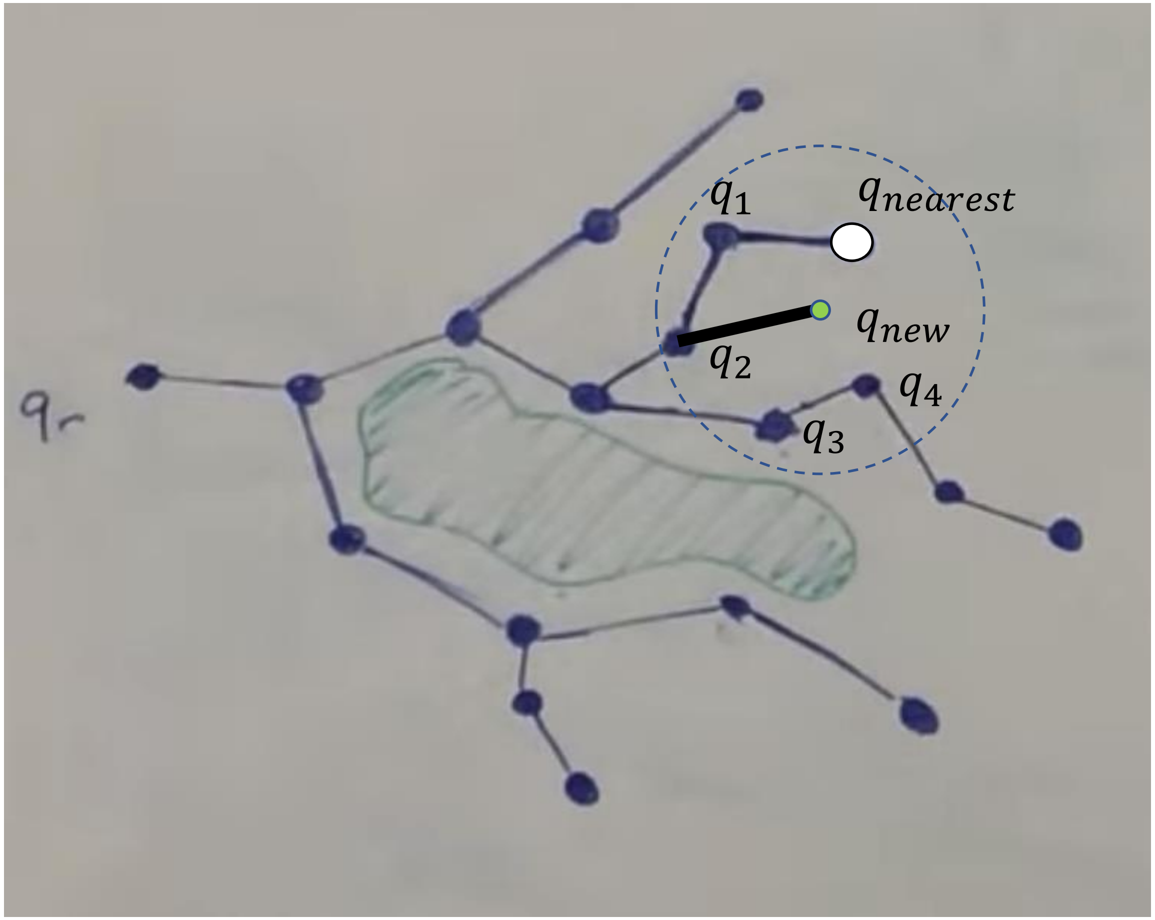
https://www.youtube.com/watch?v=JM7kmWE8Gtc

RRT*: Find neighborhood



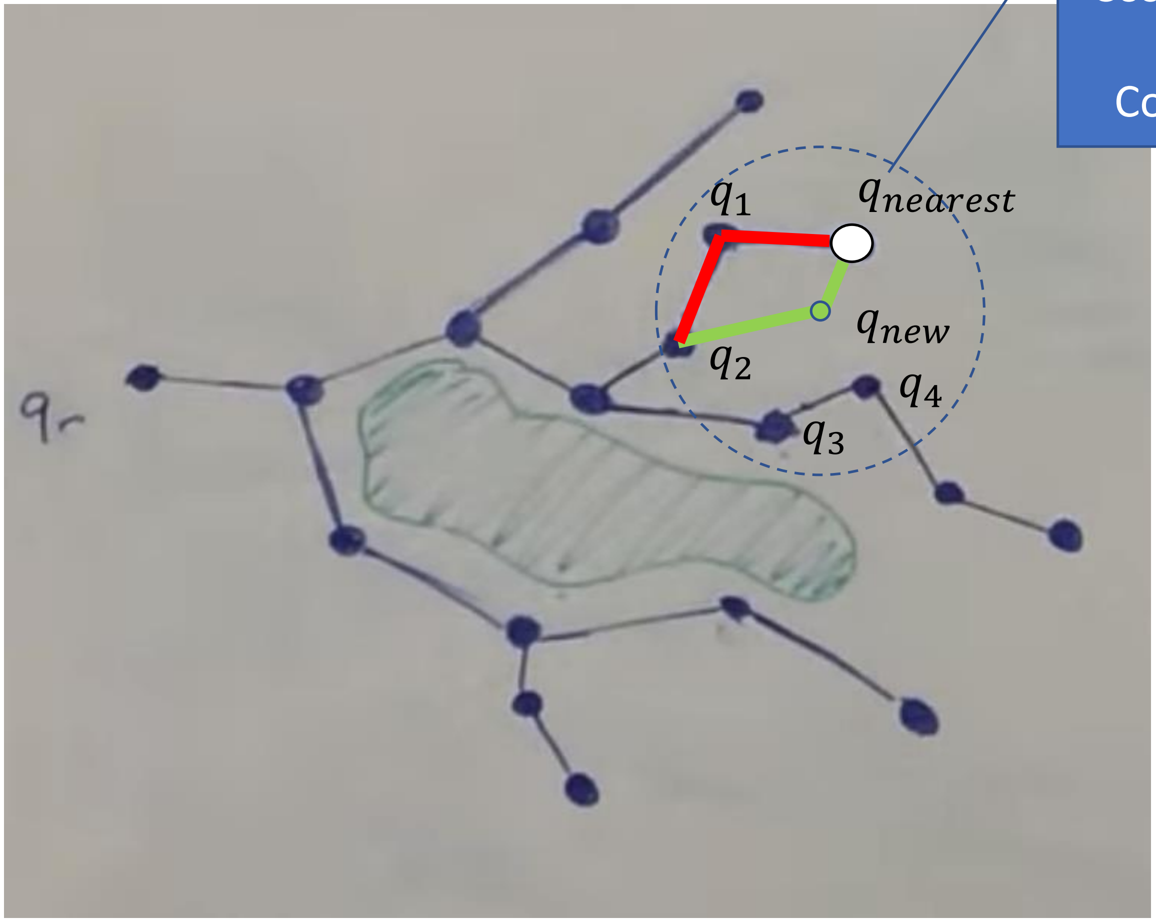
https://www.youtube.com/watch?v=JM7kmWE8Gtc

RRT*: Connect cheapest path



https://www.youtube.com/watch?v=JM7kmWE8Gtc

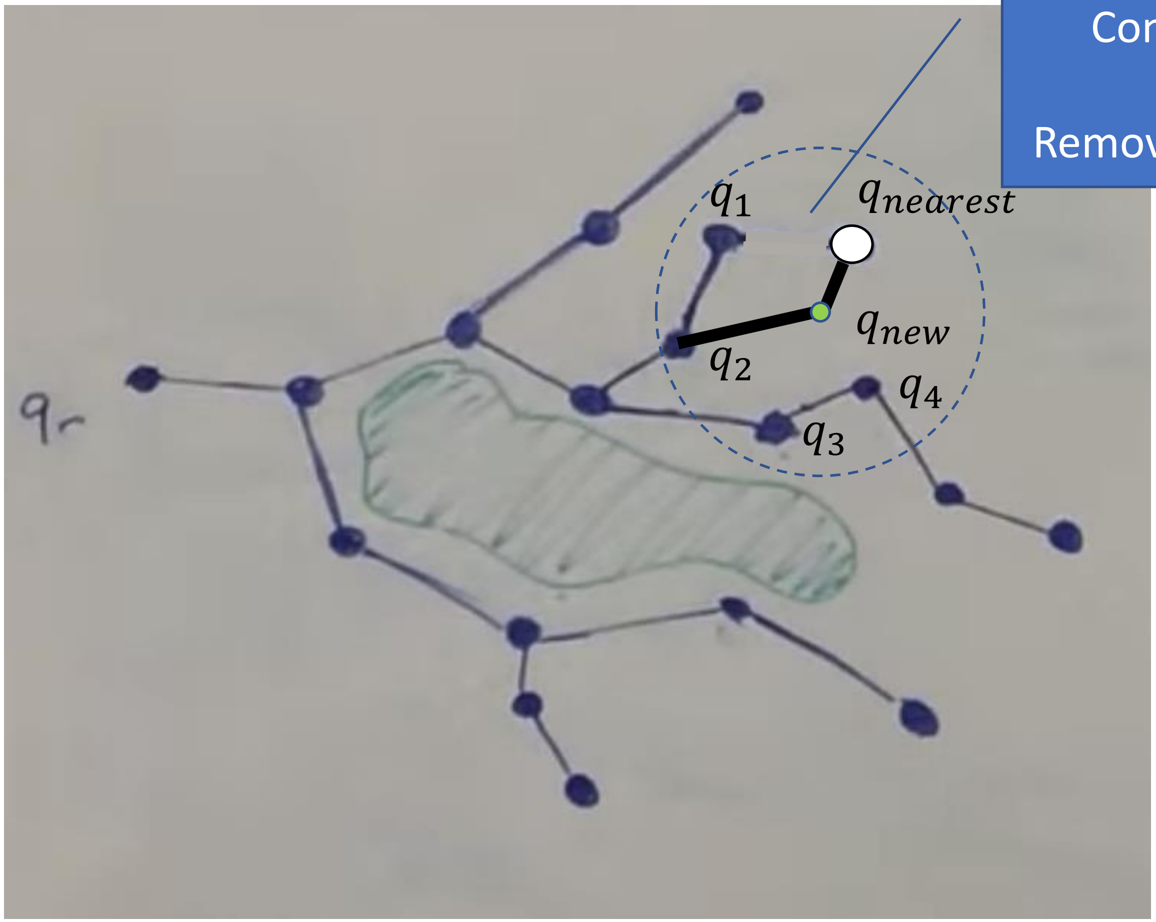
RRT*: Re-wire Graph



Cost of green route
<
Cost of red route

https://www.youtube.com/watch?v=JM7kmWE8Gtc

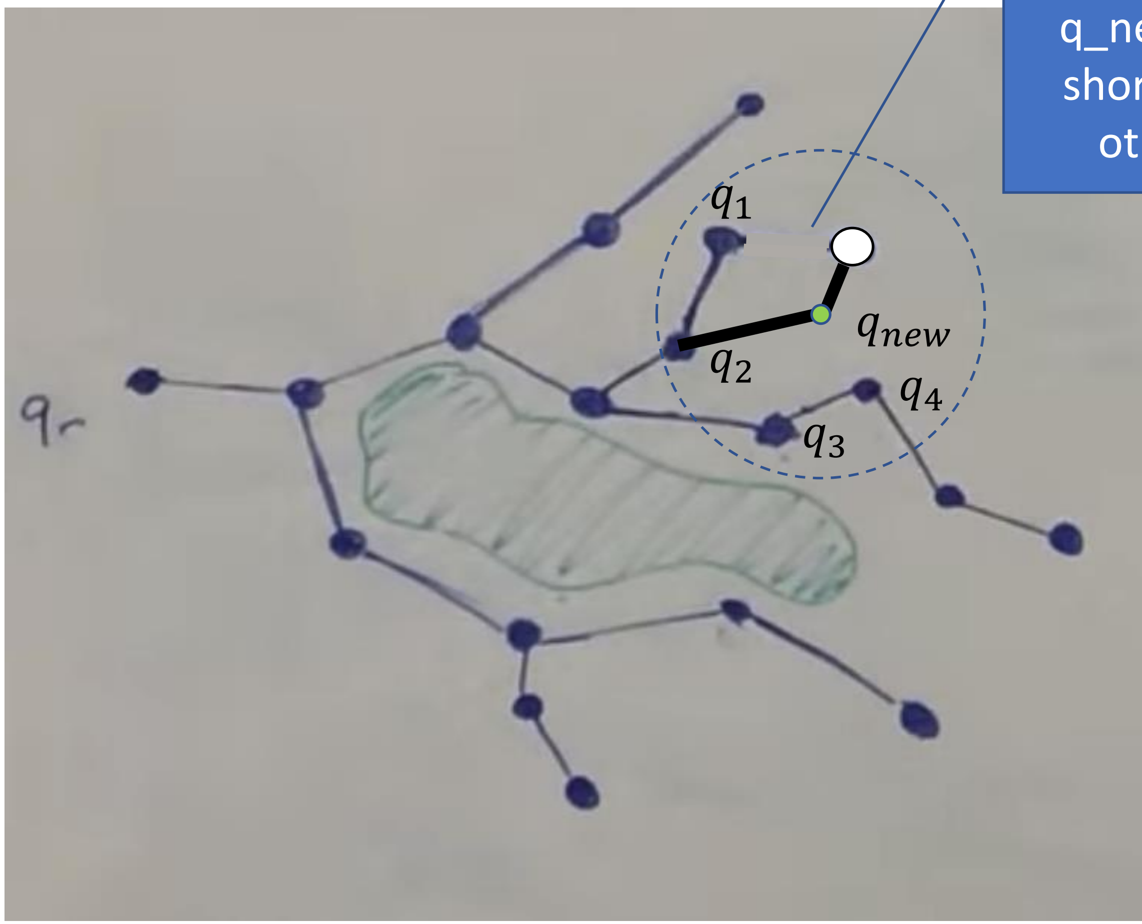
RRT*: Re-wire Graph



Connect q_{new} to $q_{nearest}$
Remove q_1 to $q_{nearest}$

https://www.youtube.com/watch?v=JM7kmWE8Gtc

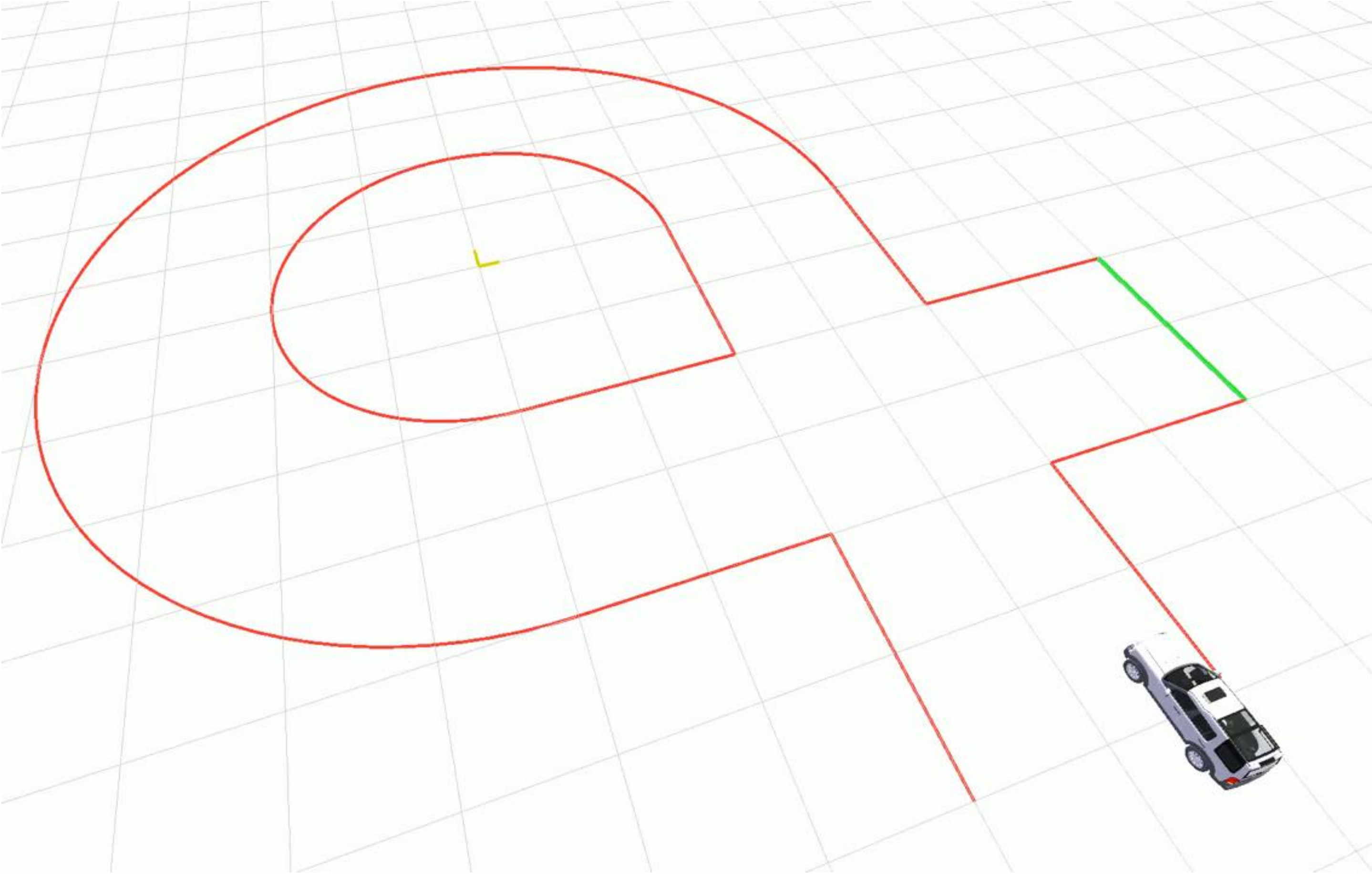
RRT*: Re-wire Graph



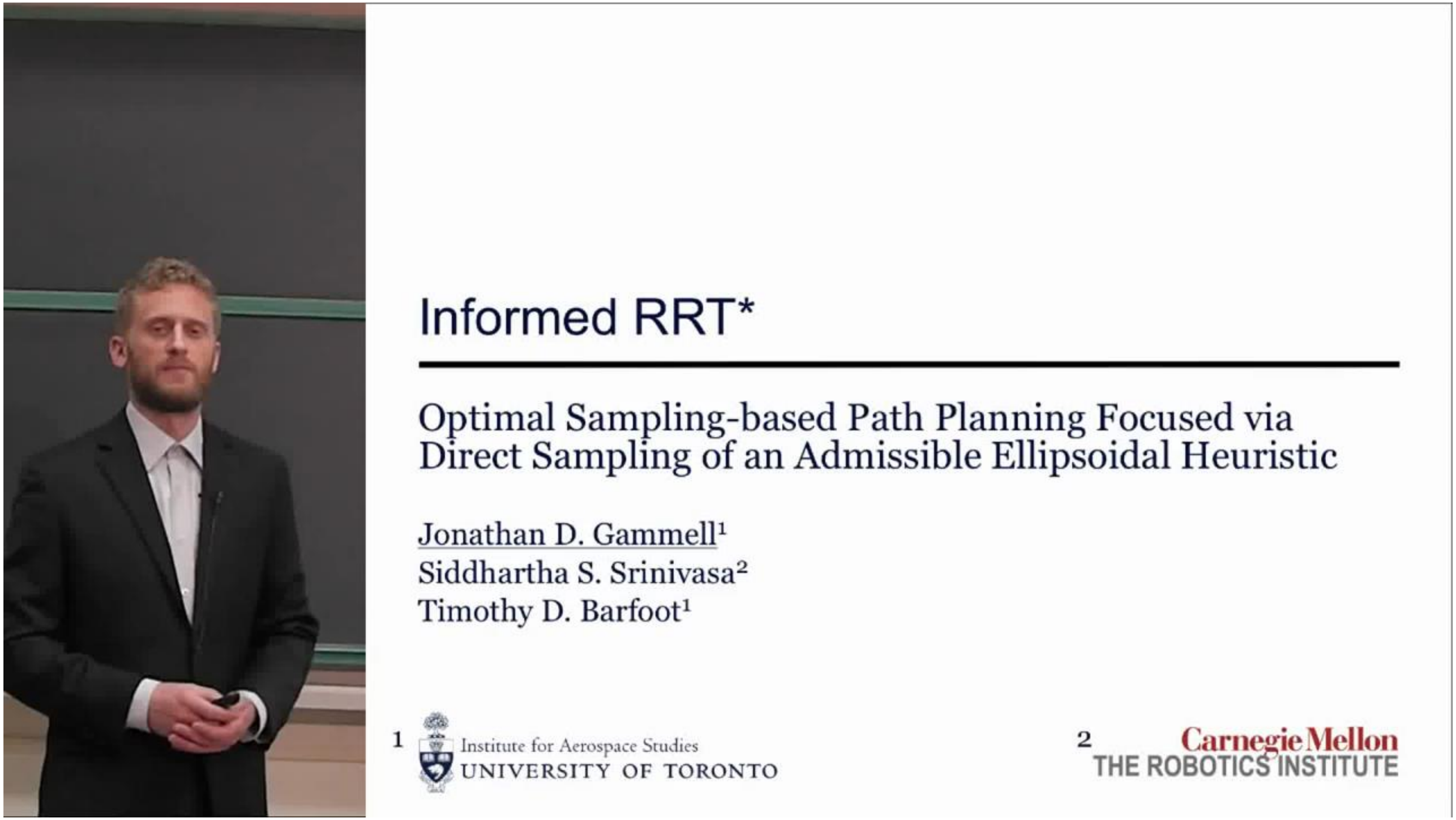
Compare path through q_new with existing shortest path for all other neighbors

https://www.youtube.com/watch?v=JM7kmWE8Gtc

RRT*

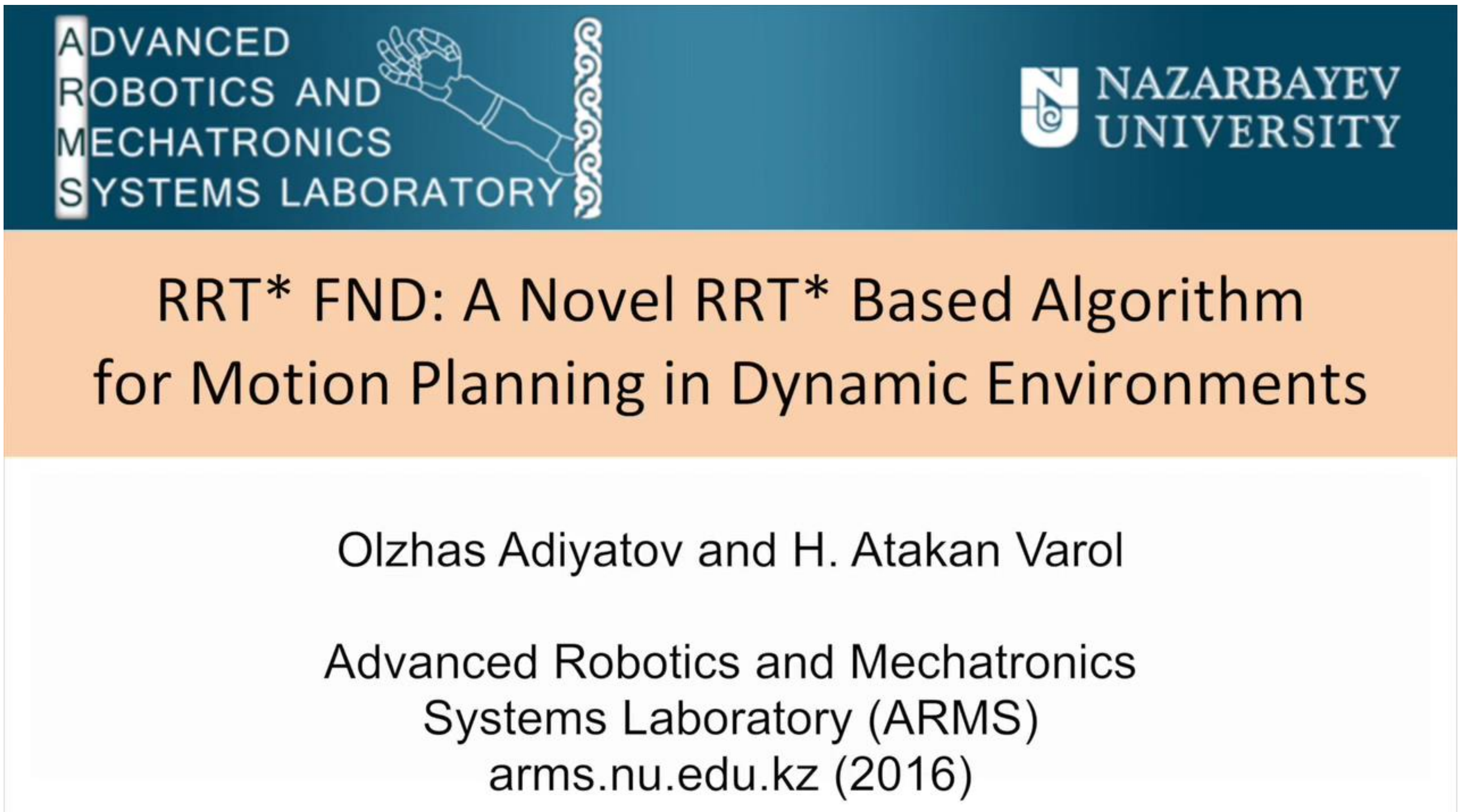


State-of-the-art Work: RRT*

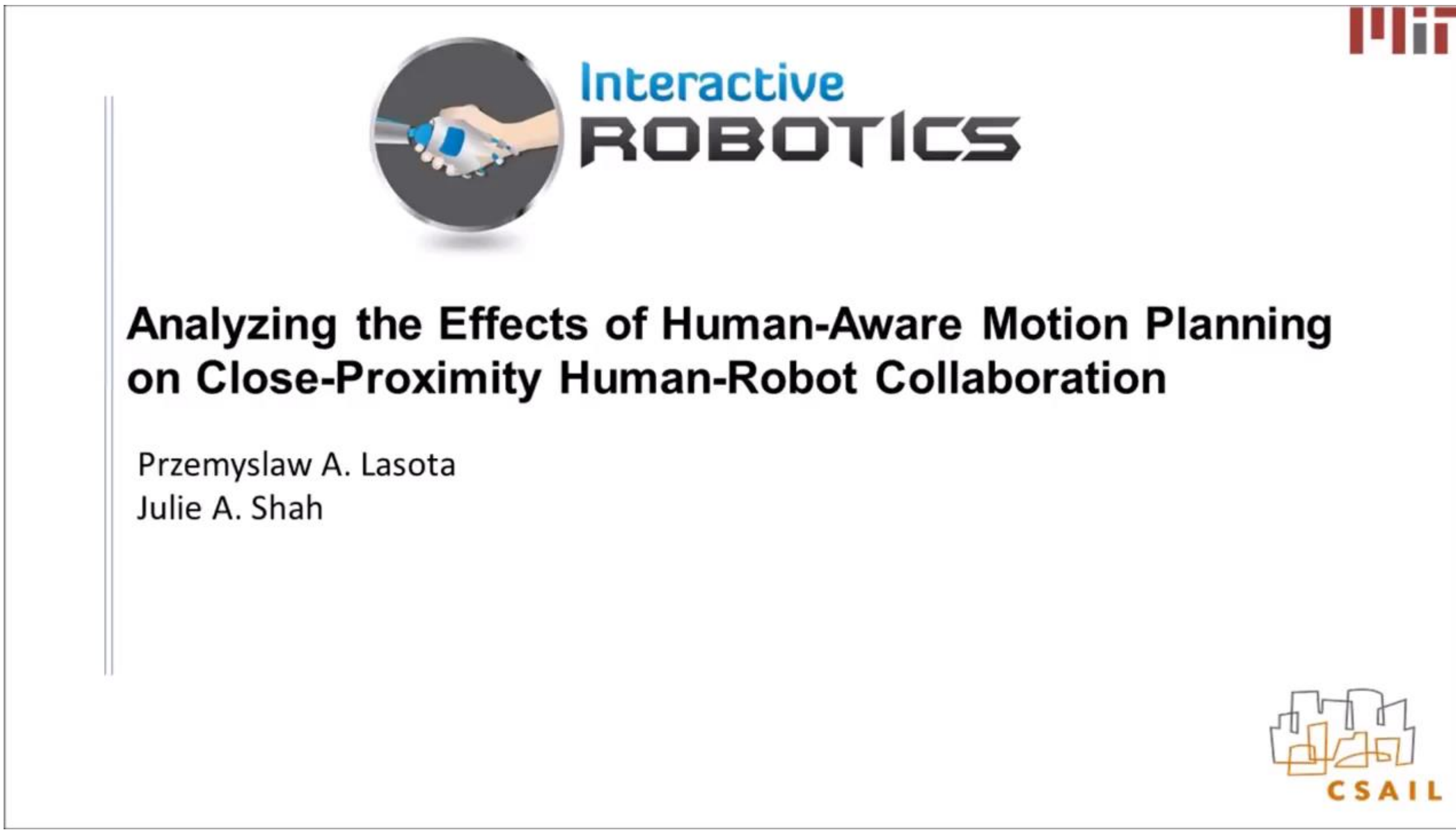


A-HRI: Open Problems

Dynamic Motion Planning



Human-Aware Motion Planning

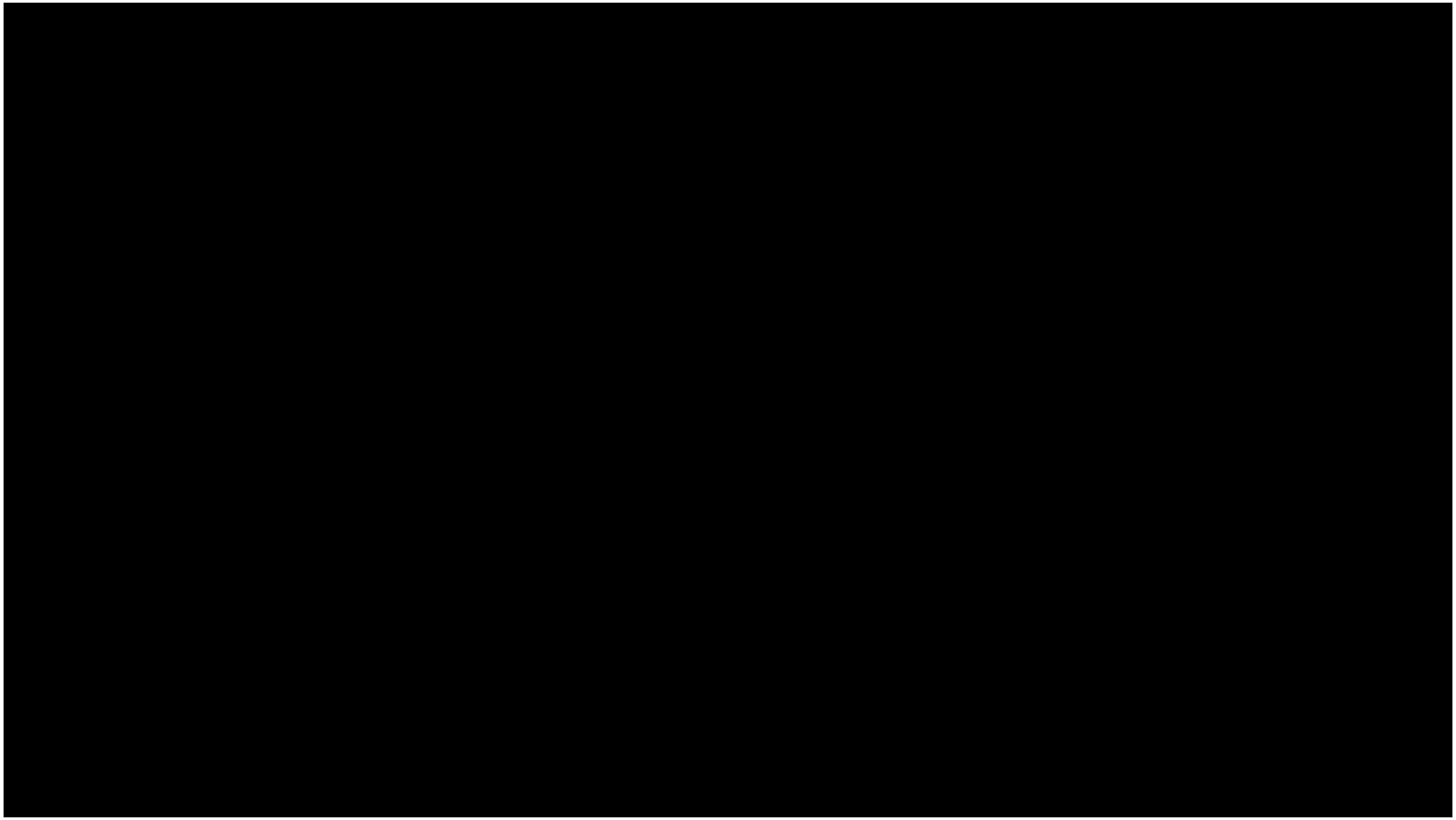


Assistive Teleoperation

Assistive Teleoperation

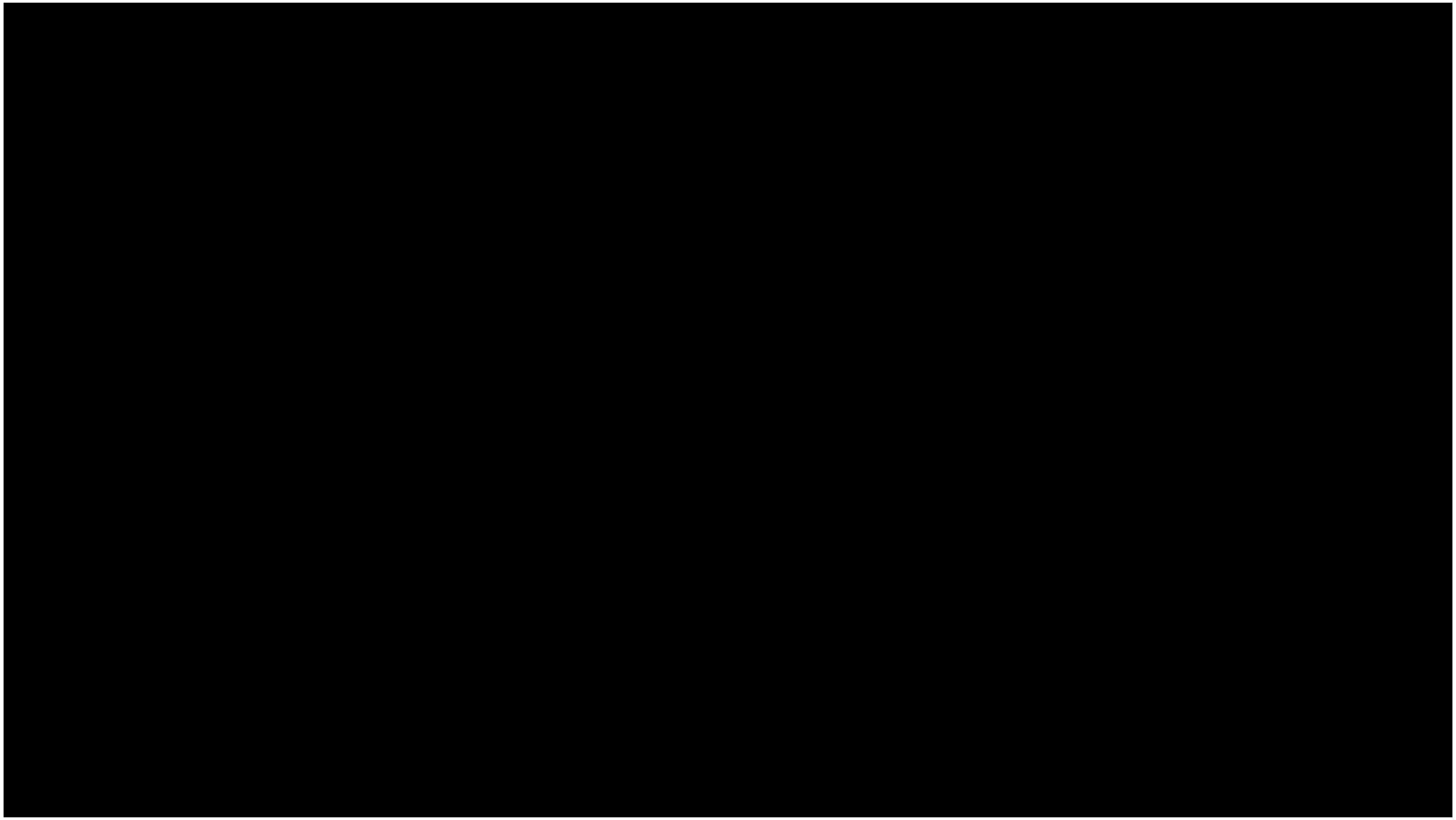
Anca Dragan
Siddhartha Srinivasa
Personal Robotics Lab, Carnegie Mellon

Assistive Teleoperation



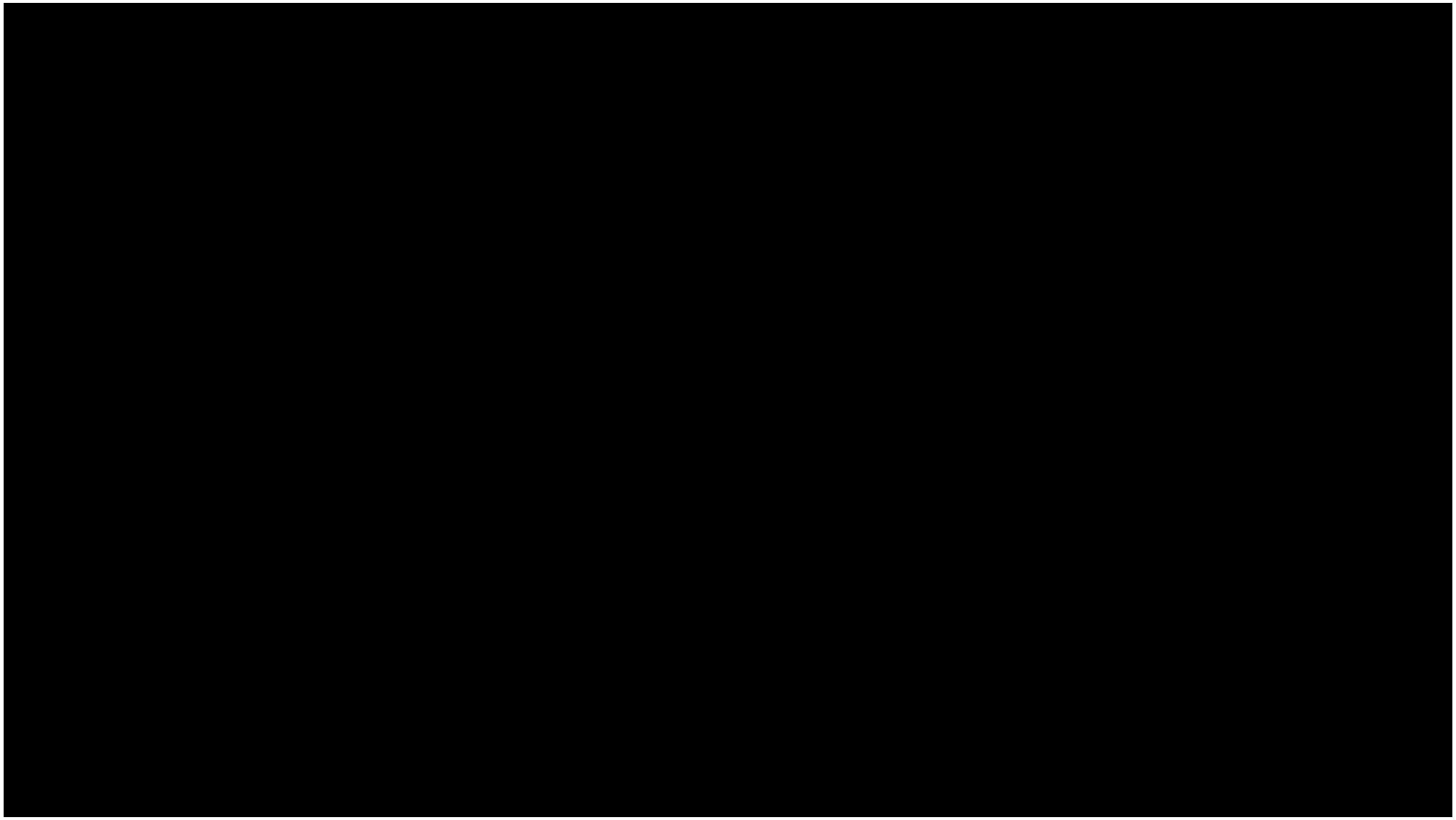
Kinesthetic Teaching

Kinesthetic Teaching

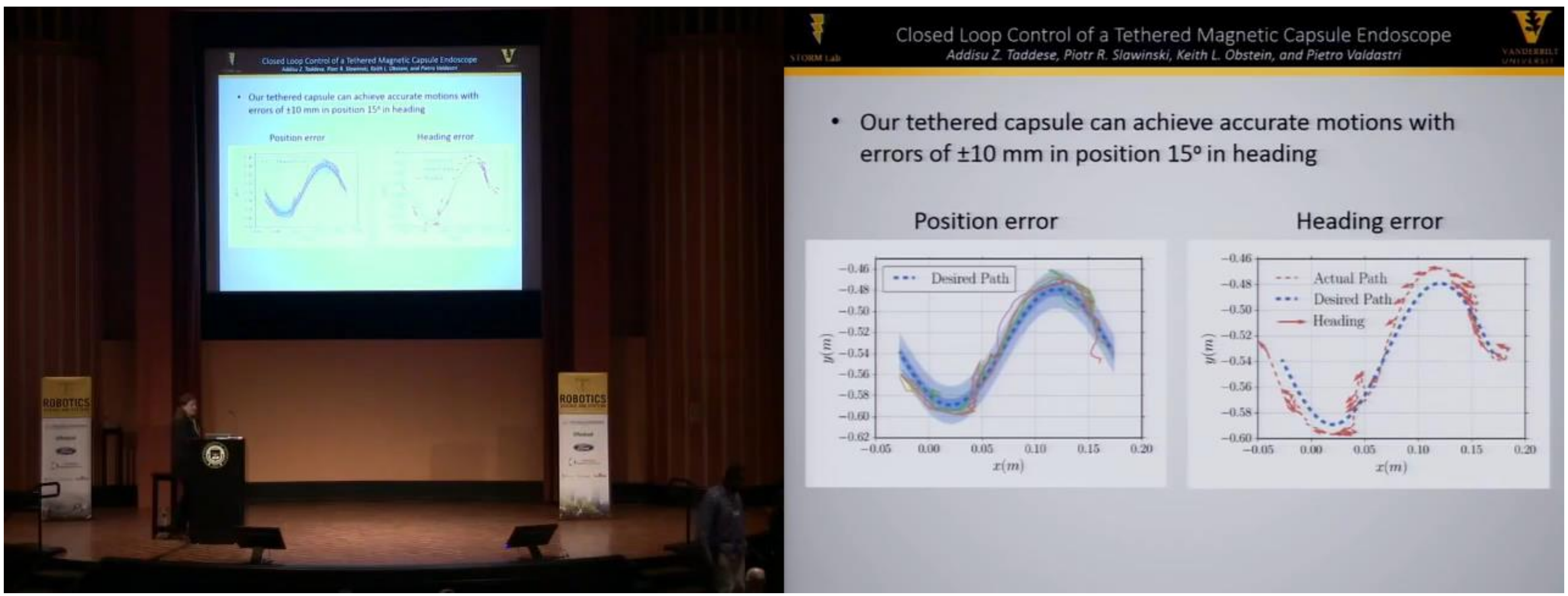


Active Learning

Preference-aware Human-Robot Collaboration



Collaborative Manipulation



Decision Support

[Robot Decision Support]

Next Time: Trajectory Optimization