Path Planning for Robotics

Nicholas Roy

March 2nd, 2011

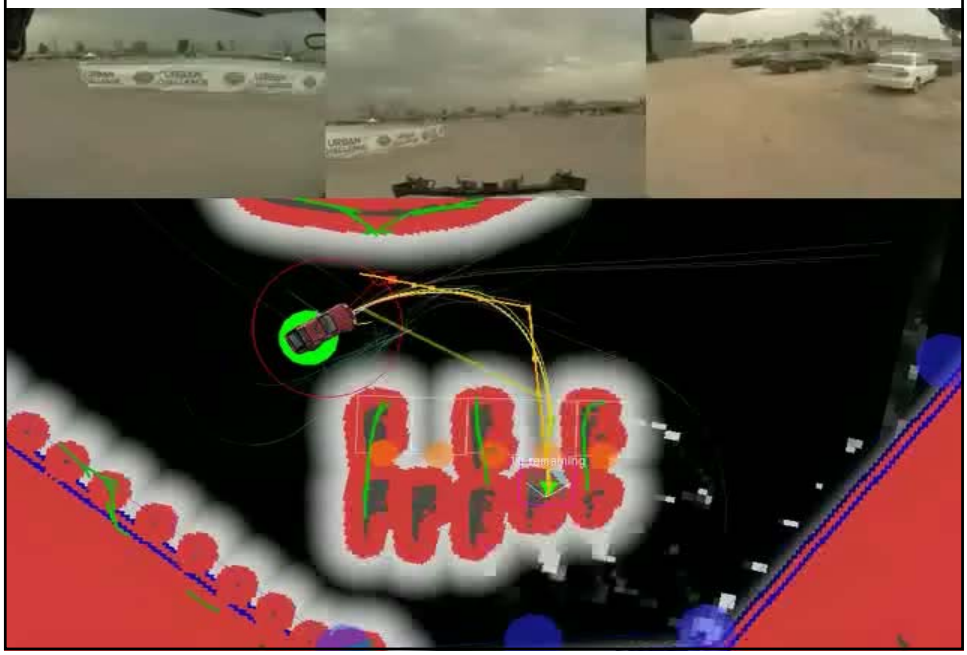
With additional material courtesy of S. Teller

Movie courtesy of S. Thrun

Why is Path Planning Hard?

- Time and resource constraints
- Uncertainty about effects of actions
- Uncertainty about environment (e.g., initial state)
- Uncertain sensing / perception
- Other agents / external events can affect goal achievement

Movie courtesy of E. Frazzoli, J. How, J. Leonard, S. Teller



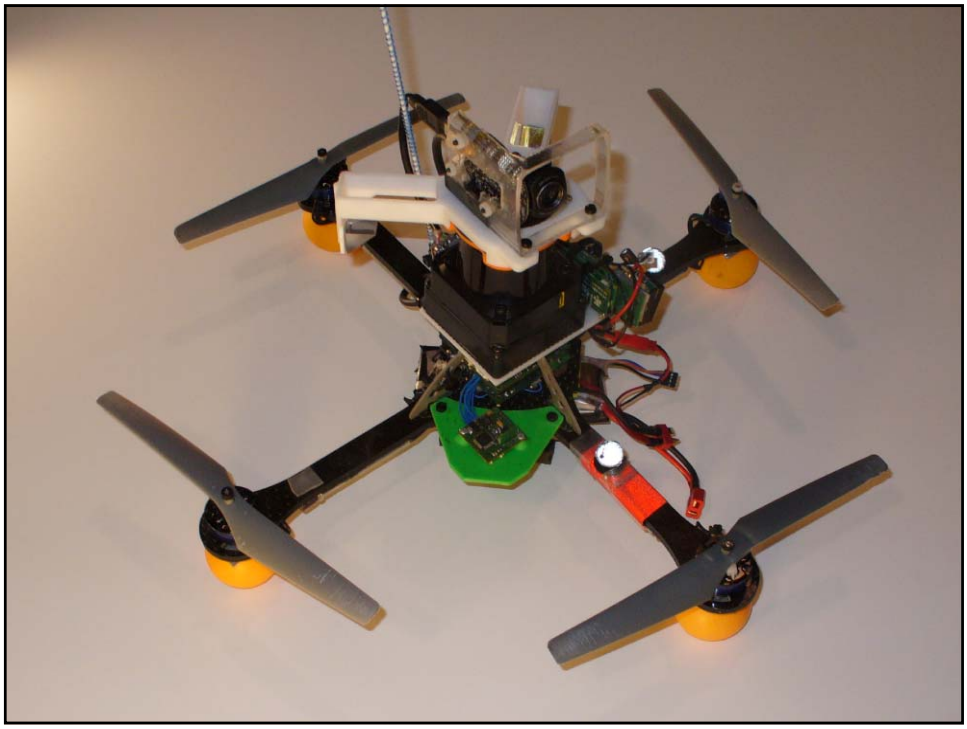
Why is Path Planning Hard?

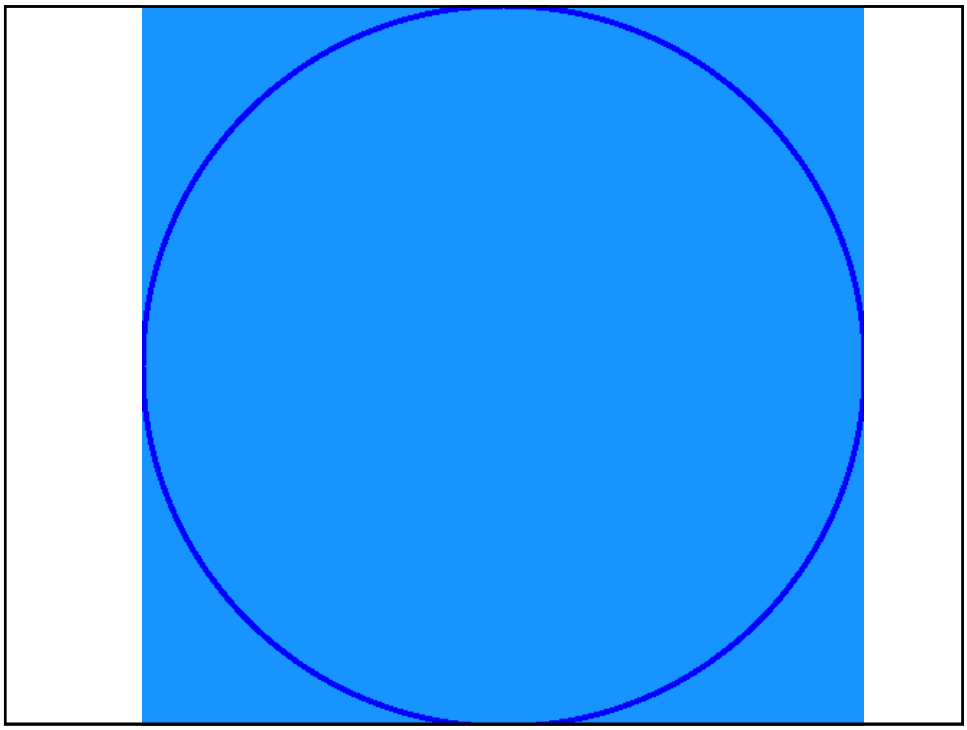
- Time and resource constraints
- Uncertainty about effects of actions
- Uncertainty about environment (e.g., initial state)
- Uncertain sensing / perception
- Other agents / external events can affect goal achievement

Movie courtesy of R. Tedrake

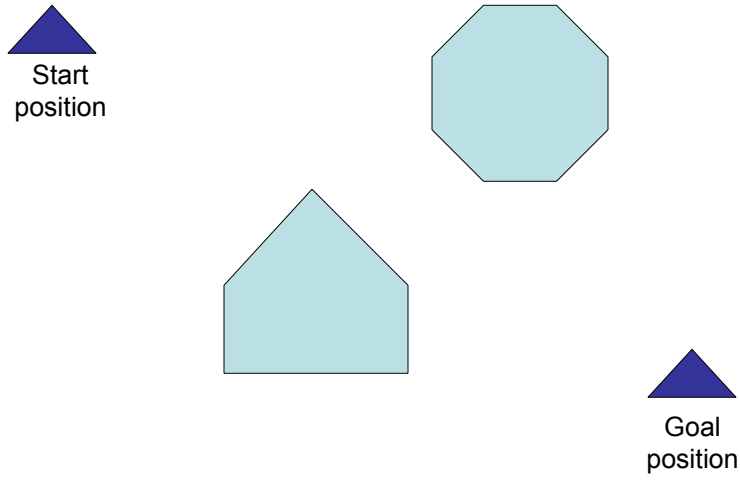
Why is Path Planning Hard?

- Time and resource constraints
- Uncertainty about effects of actions
- Uncertainty about environment (e.g., initial state)
- Uncertain sensing / perception
- Other agents / external events can affect goal achievement





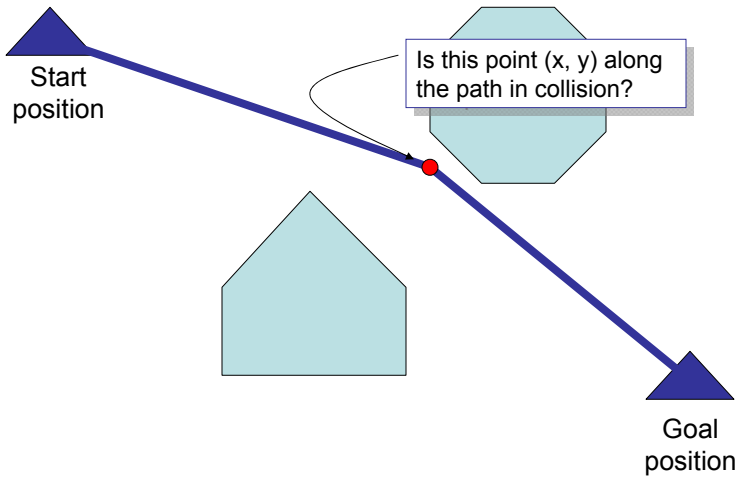
Motion Planning



Complete Motion Planning

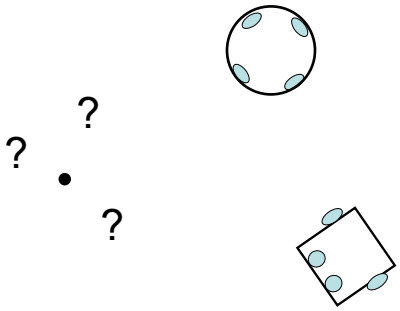
- Formal statement of motion planning problem:
  - Compute a collision-free path for a rigid or articulated moving object among static (or dynamic) obstacles
- Ideally we desire a “complete” motion planner:
  - If a solution exists, the planner is guaranteed to return it
  - Otherwise, planner indicates that no solution exists
- CMP is known to be computationally difficult
  - In general it requires exponential running time in the number of DOFs (articulation, # of obstacles etc.)
  - … Even with access to perfect, global information!

Motion Planning



A *Point* Robot?

- Can't fit much robot into a zero-area point …
  - Configuration space: the set of configurations of the robot
  - Size of C-Space depends on degrees of freedom of robot



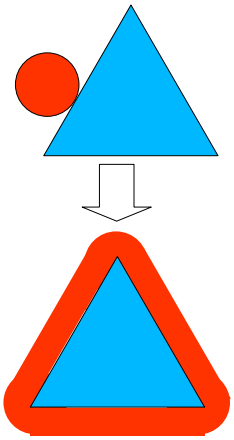
Path Planning Representations

- Configuration space
- Topological vs. Metric?
- Discrete vs. continuous spaces?

- Graph-based representations:
  - Visibility graphs
  - Voronoi diagrams
  - Probabilistic roadmaps
  - Rapidly-exploring randomized trees
  - Distinctiveness surfaces
- Metric representations:
  - Potential fields
  - Numerical potential fields
  - Value functions

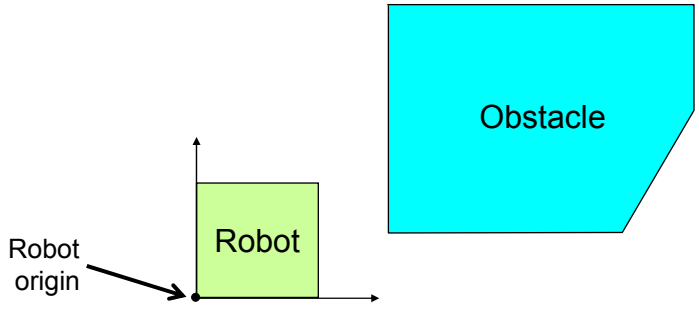
Configuration Space

- The set of configurations of the robot
- Size of C-Space depends on degrees of freedom of robot

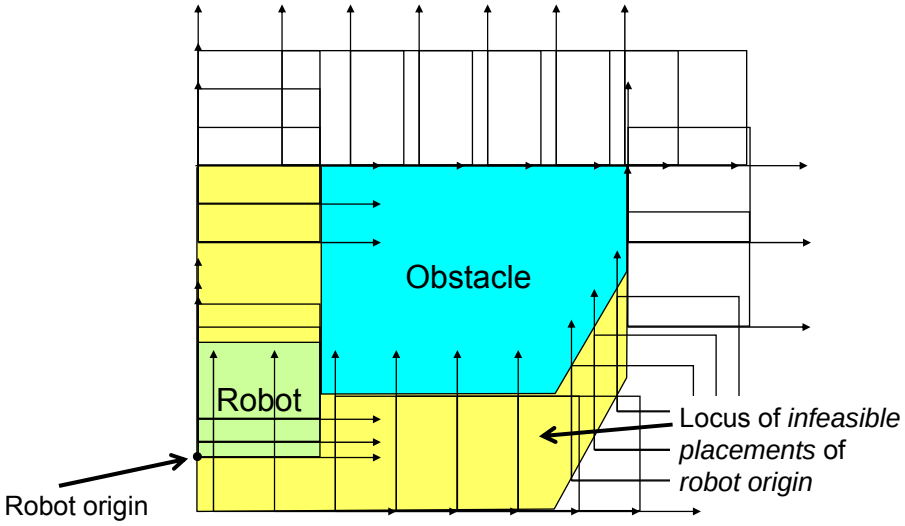


Intuition

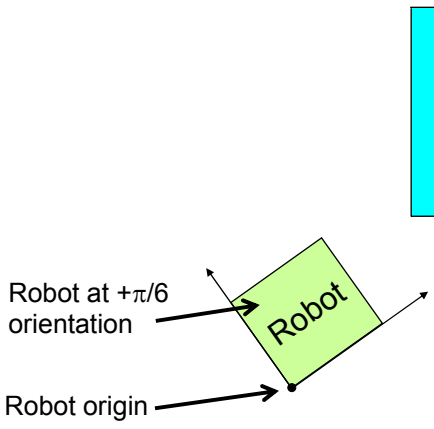
- Suppose robot can move only by translating in 2D
- How can it move in the presence of an *obstacle*?
  - How to describe *infeasible placements* of robot origin?



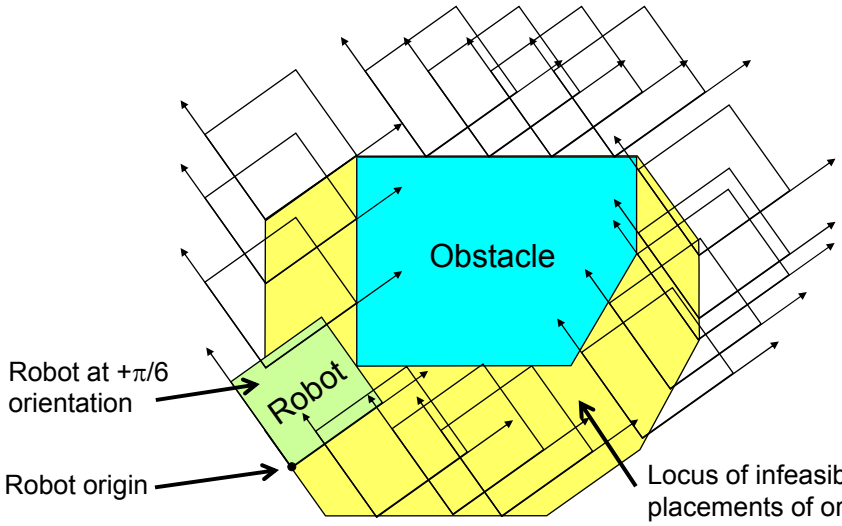
Infeasibility Under Translation



What if Robot can also Rotate?

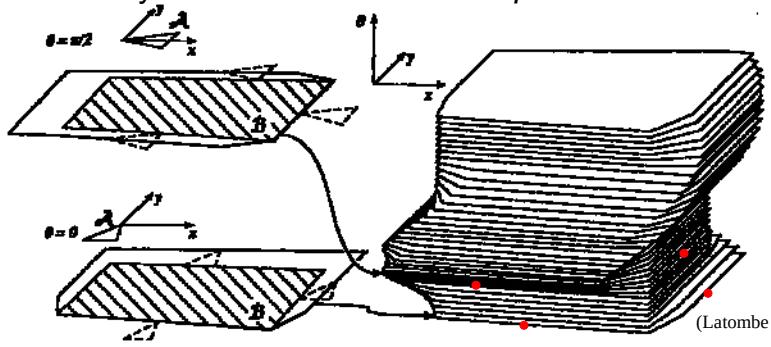


Infeasibility under 3-DOF Motion



Configuration Space

For a robot with *k* total motion DOFs, C-space is a coordinate system with *one dimension per DOF*



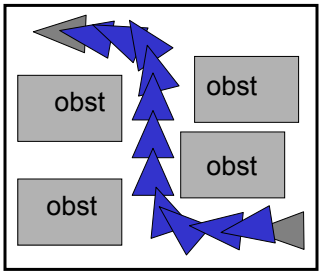
(Latombe 1991)

In C-space, a robot pose is simply

… and a workspace obstacle is a

Motion Planning Transformation

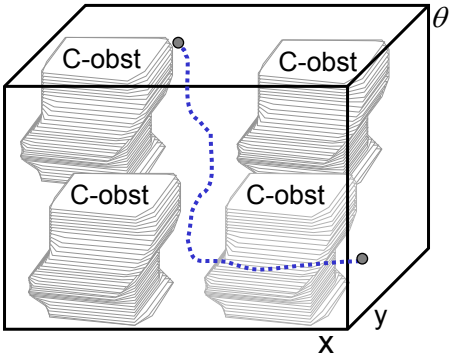
Workspace
(*x*, *y*)



Robot

Path is swept volume

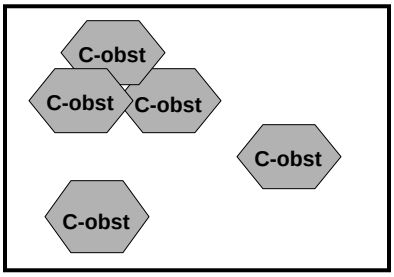
C-space
(*x*, *y*, θ)



Robot

Path is space curve

C-space Examples

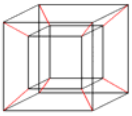


- Define space with one dimension per robot motion (or pose) DOF
- Map robot to a point in this space
- C-space = all robot configurations
- C-obstacle = locus of infeasible configurations due to obstacle

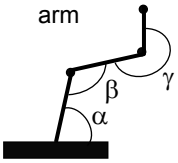
Some example configuration spaces:

Translation + rotation in 2D

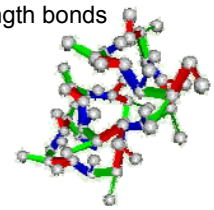
Translation + rotation in 3D



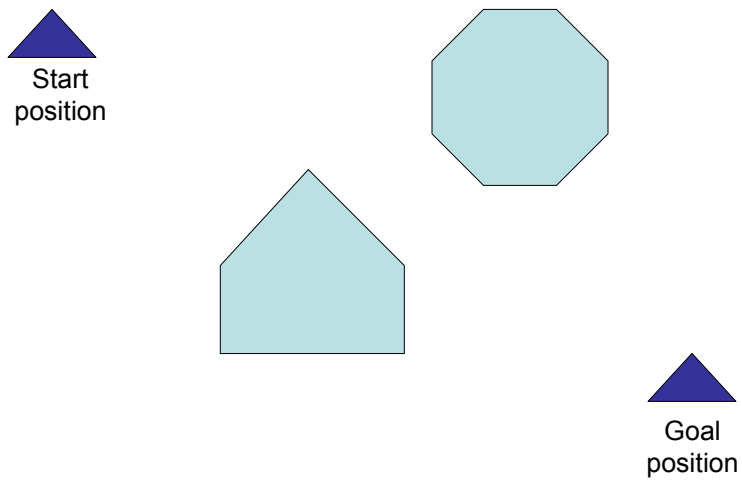
3-link arm



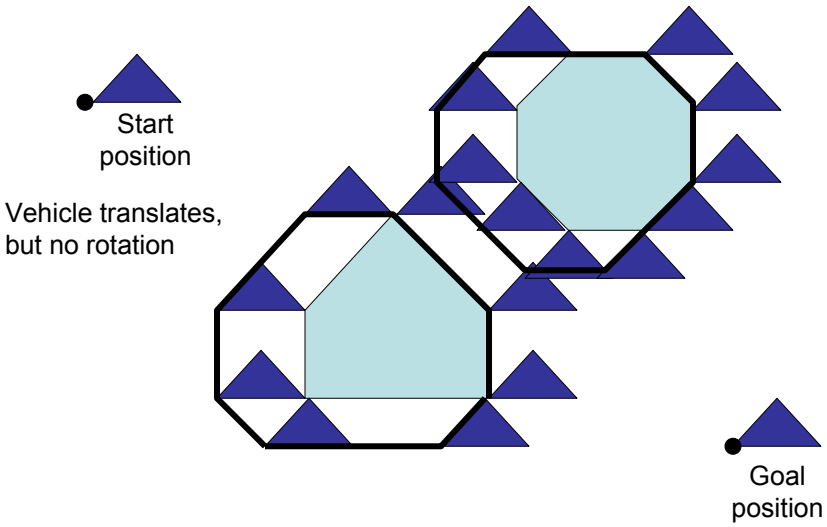
Molecule with *n* fixed-length bonds



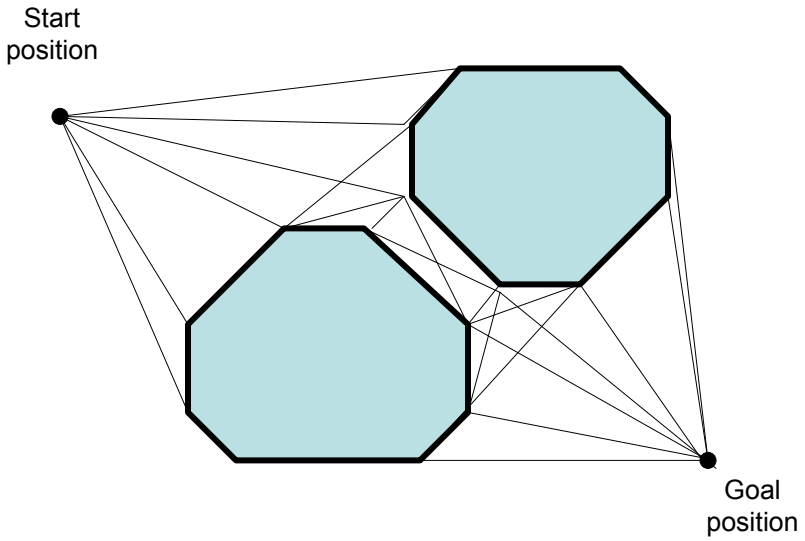
Visibility Graph Algorithm



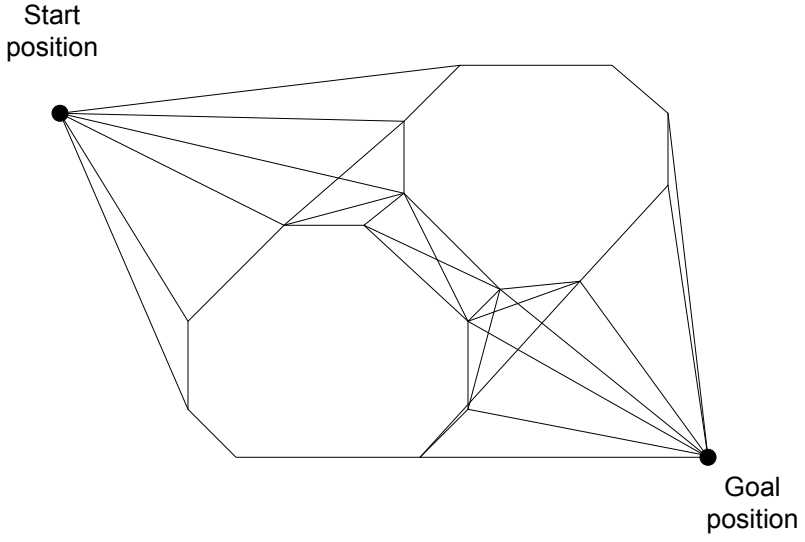
1. Create Configuration Space



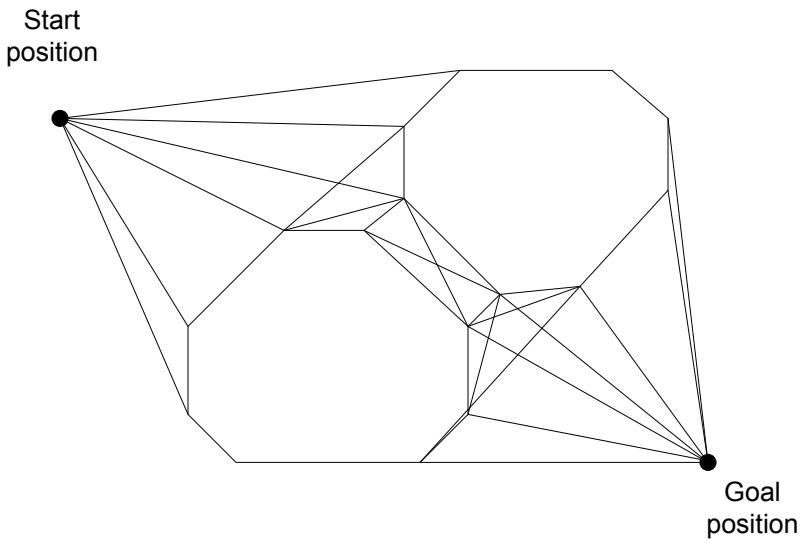
2. Map From Continuous Problem to Graph Search: Create Visibility Graph



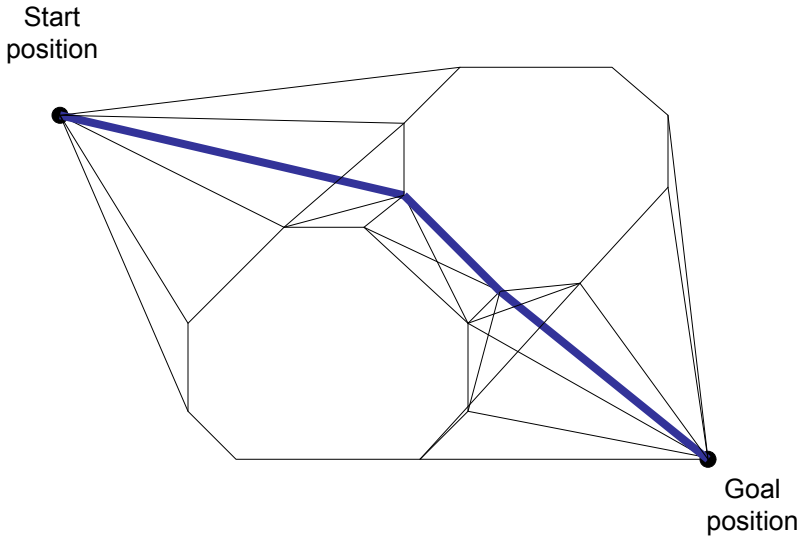
2. Map From Continuous Problem to Graph Search: Create Visibility Graph



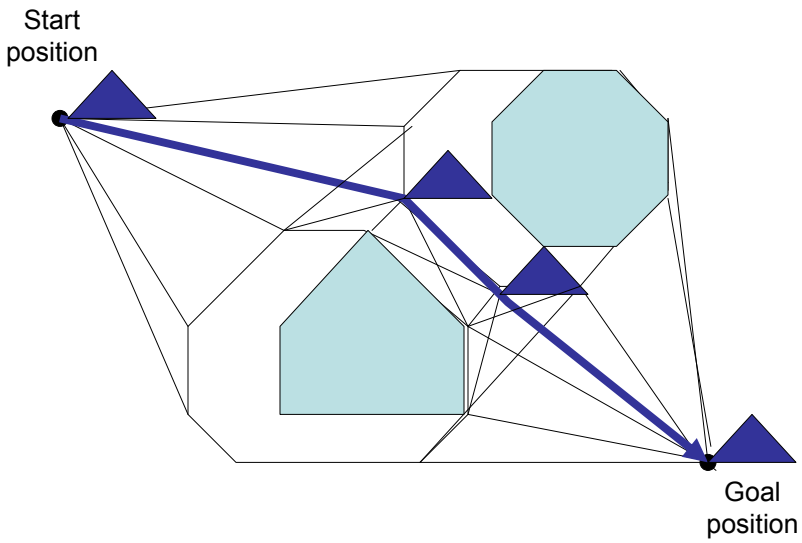
3. Find Shortest Path



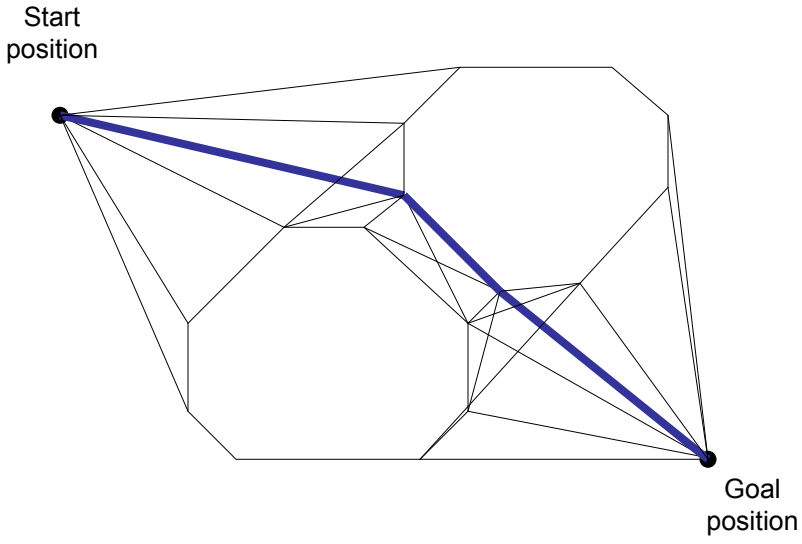
3. Find Shortest Path



Resulting Solution



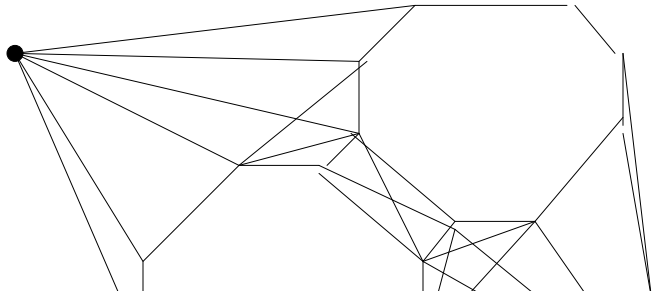
A Visibility Graph is a Kind of Roadmap



A Visibility Graph is a Kind of Roadmap

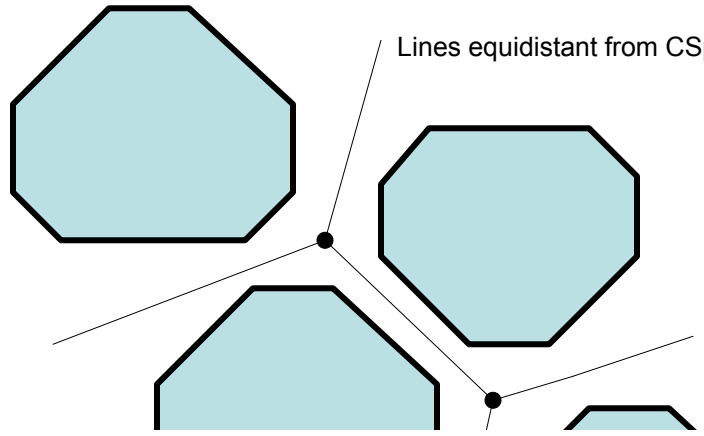
Start position

What are some other types of roadmaps?



- What do we use as edge weights?
- Memory usage?
- Running time?
- Can we optimize by omitting reflex vertices?
- What major assumption have we made about the robot?

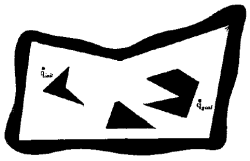
Voronoi Diagrams



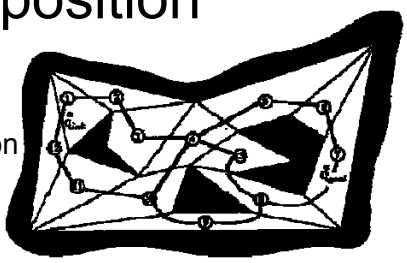
- The Voronoi Graph is the dual of the Delaunay Triangulation.
- A fast way to construct Voronoi graphs in a polygonal world:

1. Construct a polyhedral world by turning each polygon co-ordinate (x, y) into a polyhedral co-ordinate $(x, y, (x^2 + y^2)^{1/2})$
2. Compute the convex hull of this world to get Delaunay triangulation (dual of Voronoi)
3. Project hull to 2-space, and recover dual

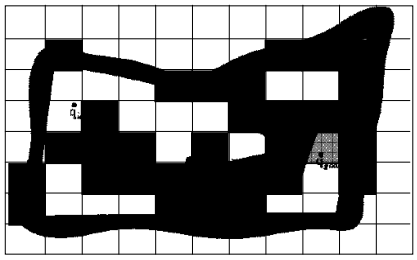
Cell-decomposition



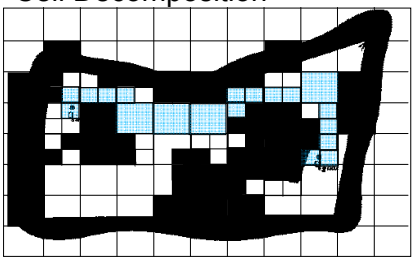
Exact Cell Decomposition



Approximate Cell Decomposition

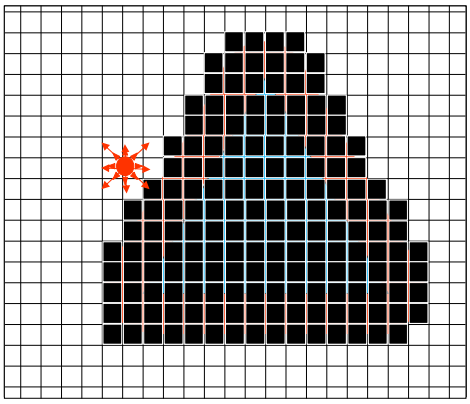


Variable-resolution Approximate Cell Decomposition

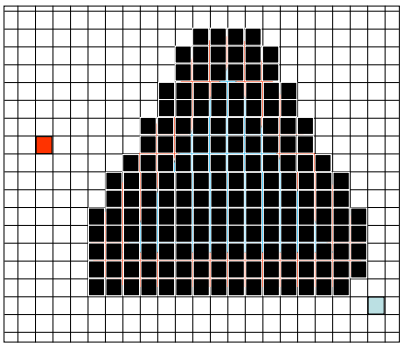


Example of a Discrete State Space

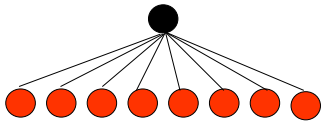
- Cartesian space
- Configuration space
- Actions take robot from one state to another
- Objective is to find a path from the start state to the goal state

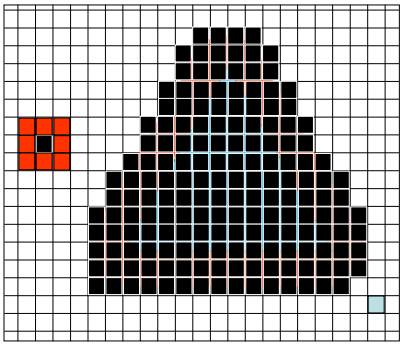


Planning as Tree Search

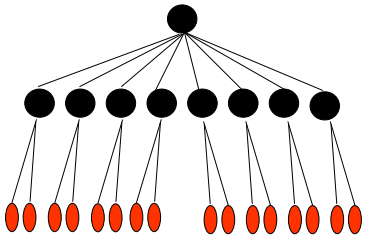


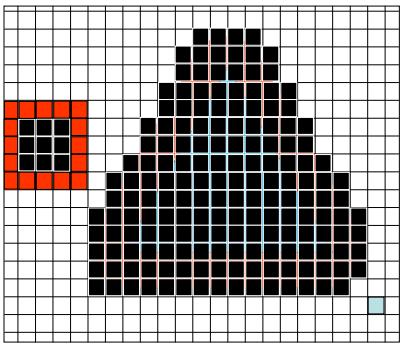
Planning as Tree Search



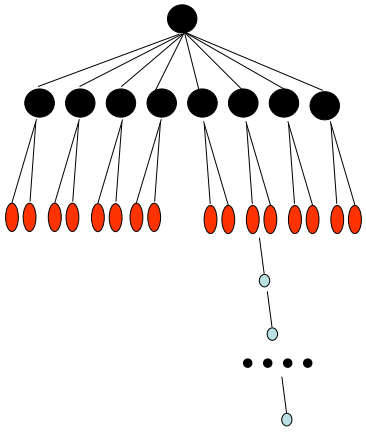


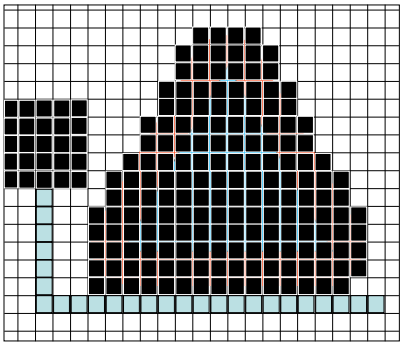
Planning as Tree Search





Planning as Tree Search

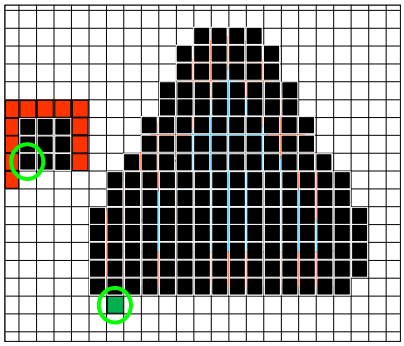




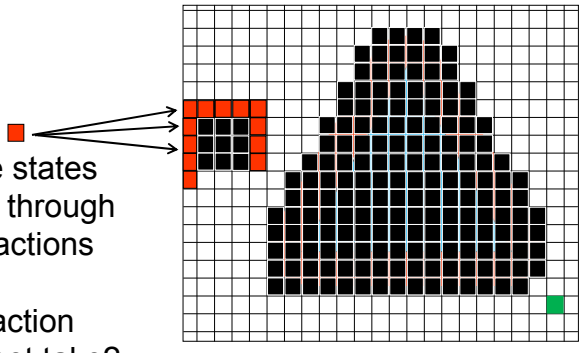
… How can such searching be made *effective* and *efficient*?

Move Generation

- Which state-action pair to consider next?
- Shallowest next
  - Aka: Breadth-first search
  - Guarantees shortest path
  - But: storage-intensive
- Deepest next
  - Aka: Depth-first search
  - Can use minimal storage
  - But: no optimality guarantee



Informed Search – A*



Candidate states reachable through available actions

… which action should robot take?

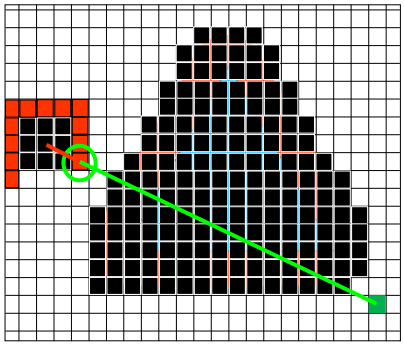
Informed Search – A*

- Use domain knowledge to bias the search
- Favor actions that might get closer to the goal
- Each state gets assigned an approximate cost

$f(x) = c(x) + h(x)$

Cost incurred to here from the start state

Estimated cost from here to the goal, aka the "heuristic" cost

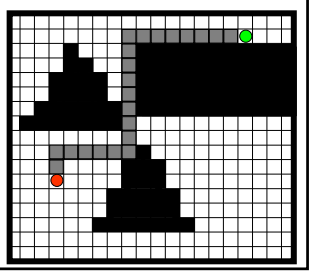


- For example:
  - c(x) = 3, h(x) = ||x-goal|| = sqrt(8^2+18^2) = 19.7, so f(x)=22.7

Once the search is done, and we have found the goal

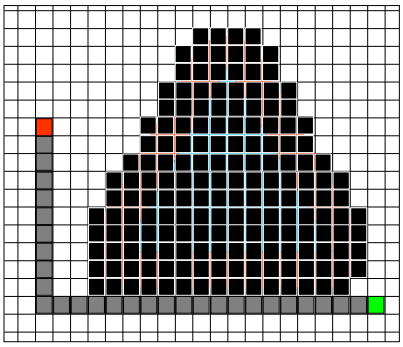
- We have a tree that contains a path from the start to the goal
- Follow the parent pointers in the tree and trace back from the goal to the root, keeping track of which states you pass through
- This set of states constitutes your plan

- To execute the plan, use your PD controller to face the first state in the plan, and then drive to it
- Once at the state, face and drive to the next state



A problem with plans

- We have a plan that gets us from the start to the goal
- What happens if we take an action that causes us to leave the plan?

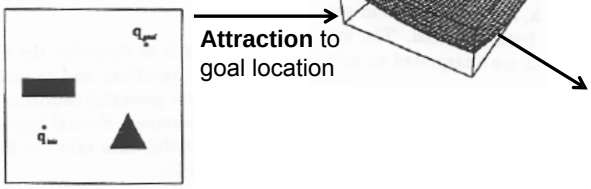


1) It's a problem with planners!
   We should use behaviours!
2) We can replan
3) We can keep a cached conditional plan
4) We can keep a policy

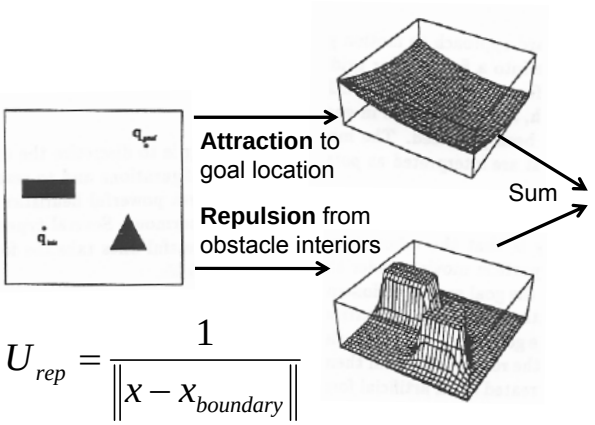
Potential Field Method

- Real-time collision avoidance method [Khatib 1986]
- Construct scalar potential field throughout freespace

$$U_{att} = \frac{1}{2}\left\| x - x_{goal} \right\|^2$$

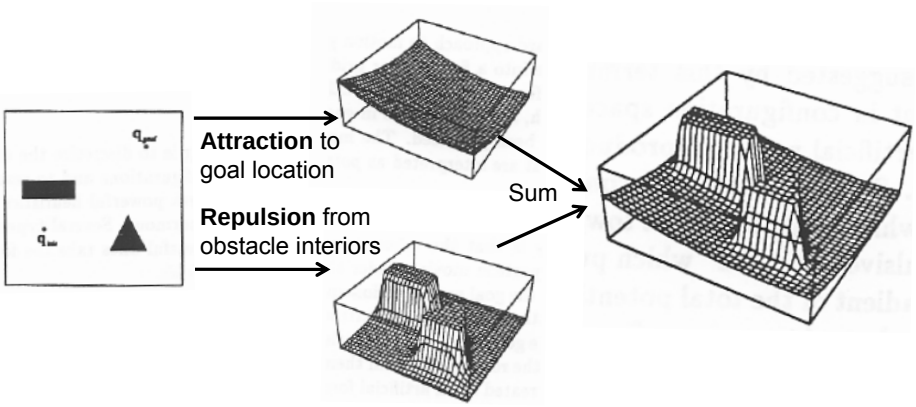


Potential Field Method



$$U_{rep} = \frac{1}{\left\| x - x_{boundary} \right\|}$$

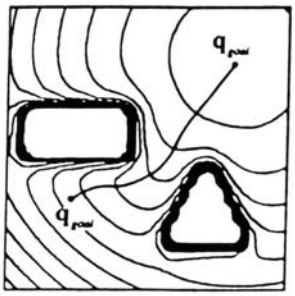
Potential Field Method

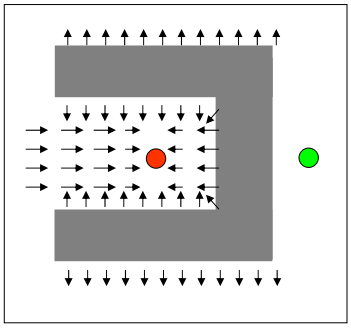


- Robot moves along negative gradient of potential field

Ideal Potential Field

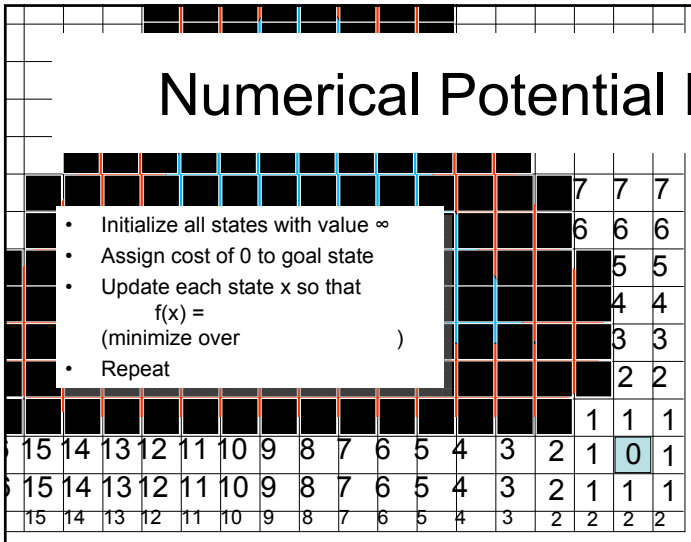
- We want to construct the potential field so that it:
  - Is nearly infinite close to obstacles
  - Has a global minimum at the goal (so no local minima)
  - Is smooth everywhere
  - Does this algebraic method achieve this?





If only life were so easy…

Numerical Potential Field

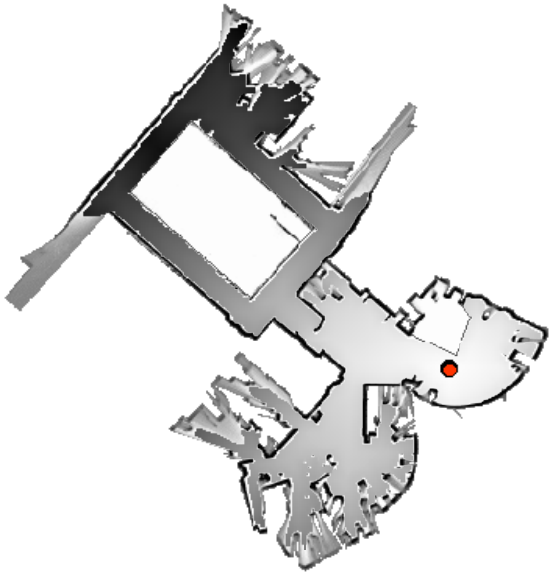


- Initialize all states with value ∞
- Assign cost of 0 to goal state
- Update each state x so that
  f(x) =
  (minimize over)
- Repeat

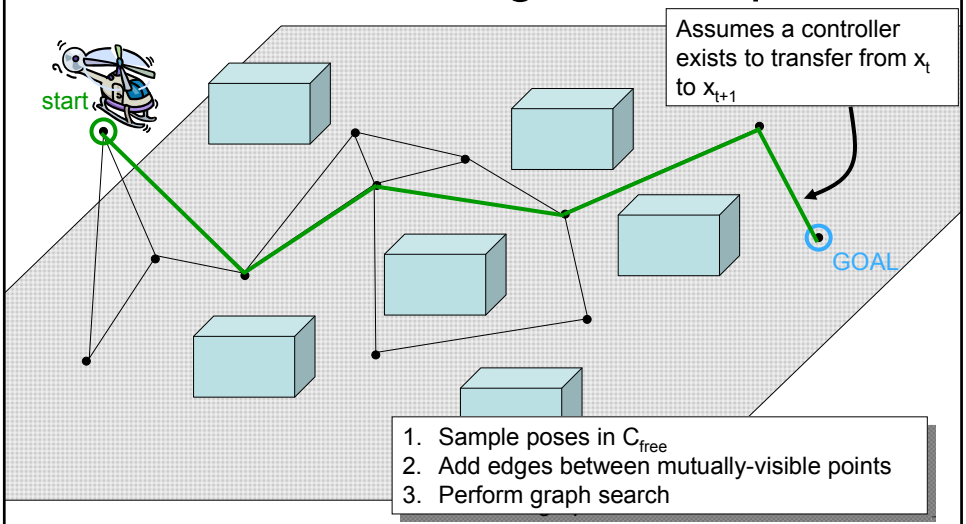
Numbers shown are for an obstacle-induced cost of ∞, and a goal-induced cost of 1 unit per grid cell (can also make it costly to approach obstacles)

Post-plan: for each state

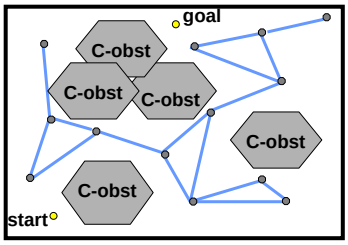
Example Output Value Function



Motion Planning in High Dimensional Configuration Spaces



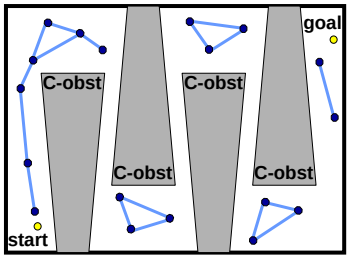
PRMs: Pros and Cons



Advantages

1. *Probabilistically complete*
2. Easily applied to high-dimensional C-spaces
3. Support fast queries (w/ enough preprocessing)

Many success stories in which PRMs were applied to previously intractable problems



Disadvantages

PRMs don't work well for some problems:
– Unlikely to sample nodes in *narrow passages*
– Hard to connect nodes along constraint surfaces

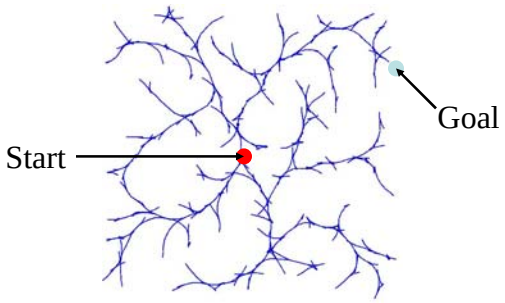
Rapidly Exploring Random Trees

- Robots have
  - inertia and dynamiccs
  - limited control authority or limited actuation
  - uncertainty
- Rapid replanning is required

- RRTs: Rapidly Exploring Random Trees
  - Incremental construction of the roadmap
  - Plans integrate kinodynamic constraints

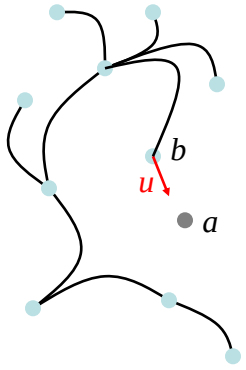
RRT

- Build a rapidly-exploring random tree in state space (X), starting at s_{start}
- Stop when tree gets sufficiently close to s_{goal}



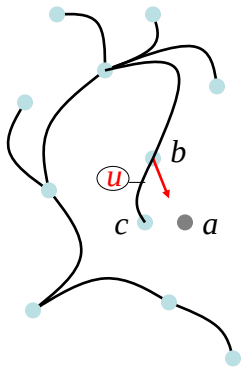
Building an RRT

- To extend an RRT:
  - Pick a random point a in X
  - Find b, the node of the tree closest to a
  - Find control inputs u to steer the robot from b to a



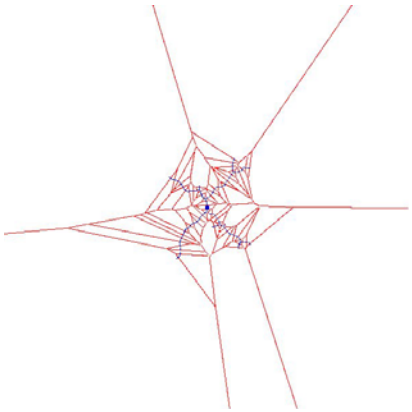
Building an RRT

- To extend an RRT (cont.)
  - Apply control inputs u for time δ, so robot reaches c
  - If no collisions occur in getting from a to c, add c to RRT and record u with new edge

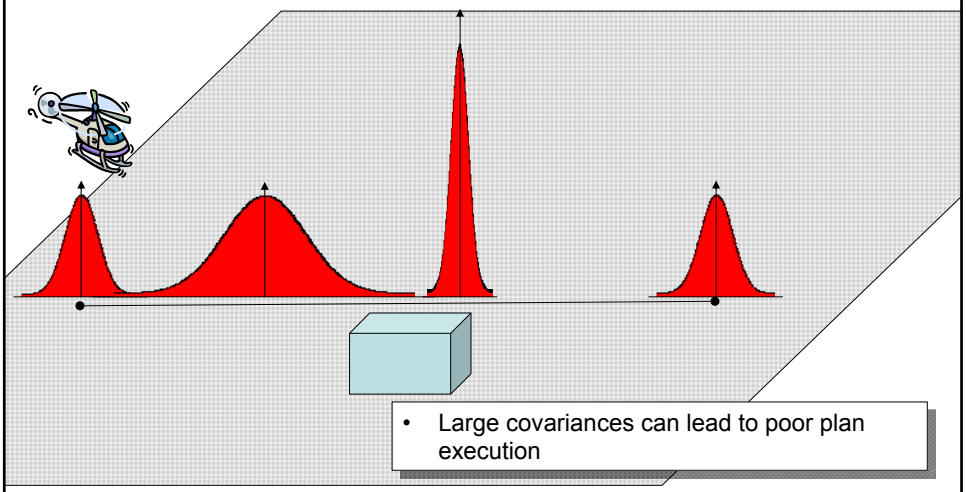


Principle Advantage

- RRT quickly explores the state space:
  - Nodes most likely to be expanded are those with largest Voronoi regions

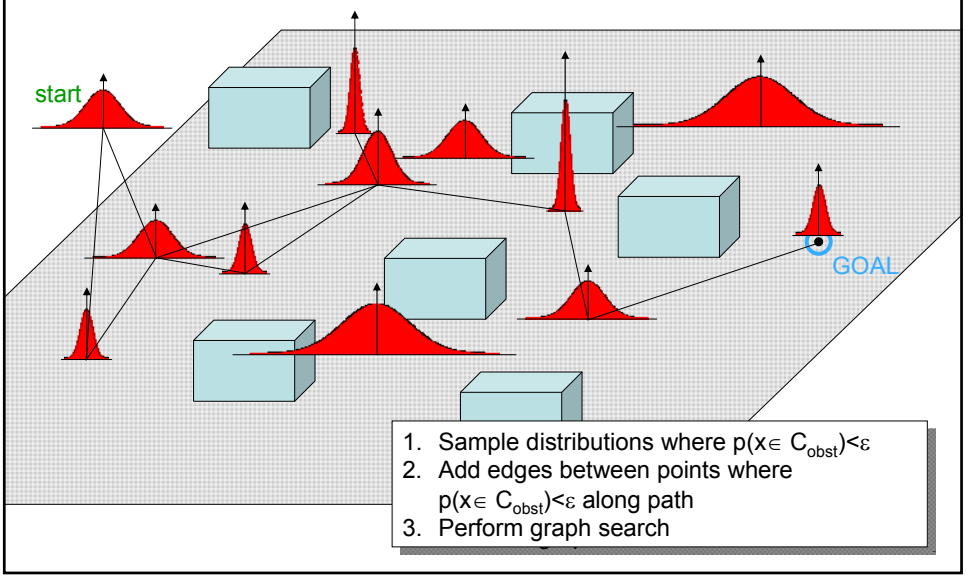


State vs. Information Space



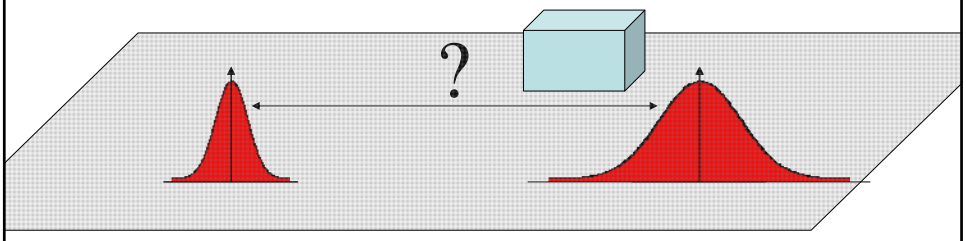
- Large covariances can lead to poor plan execution

Motion Planning in Information Space

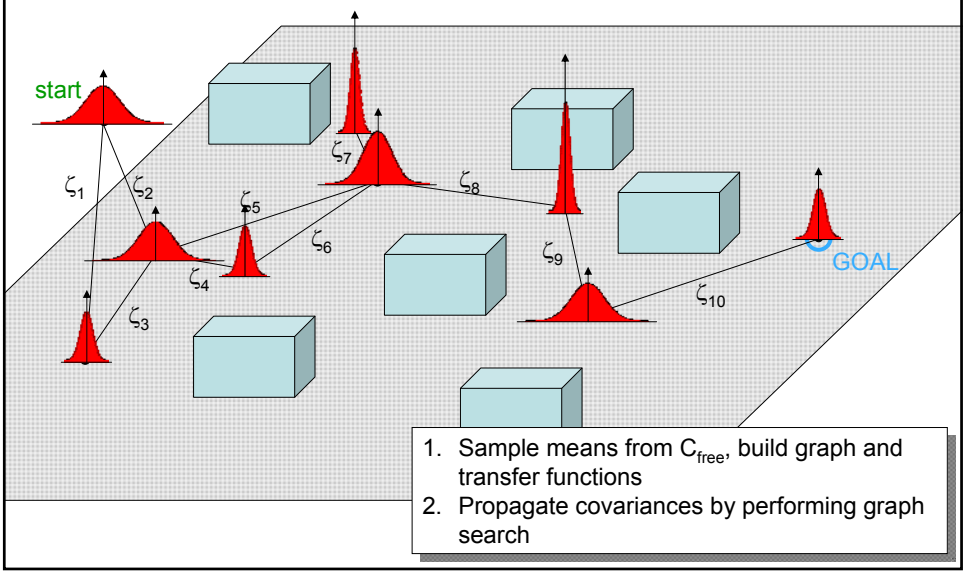


1. Sample distributions where $p(x \in C_{obst}) < \varepsilon$
2. Add edges between points where $p(x \in C_{obst}) < \varepsilon$ along path
3. Perform graph search

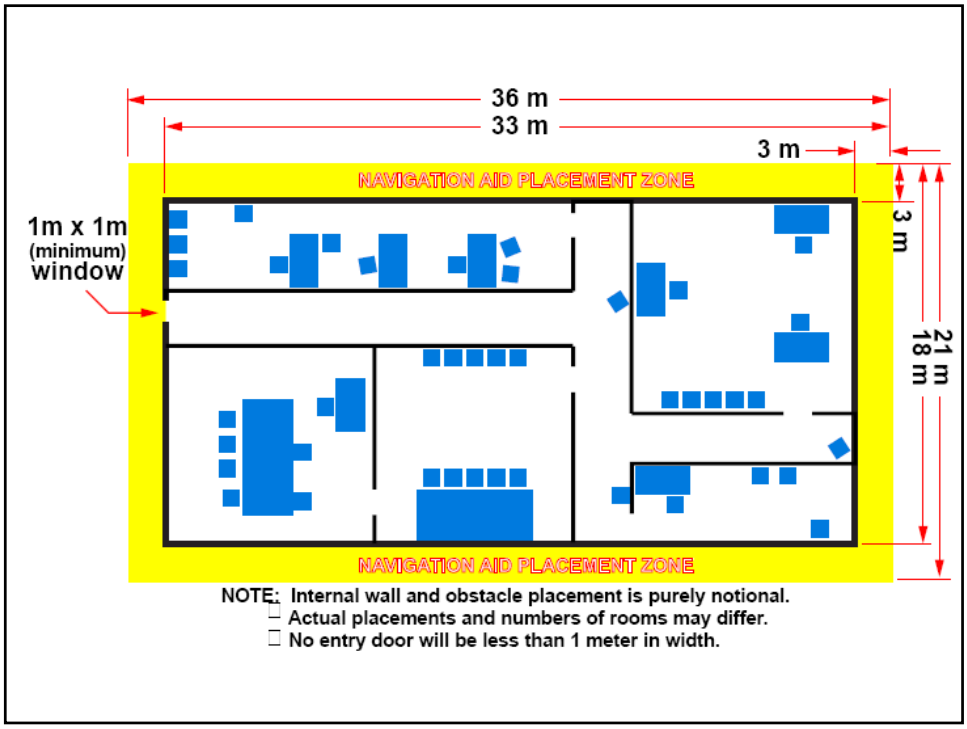
Problem: Edge Construction



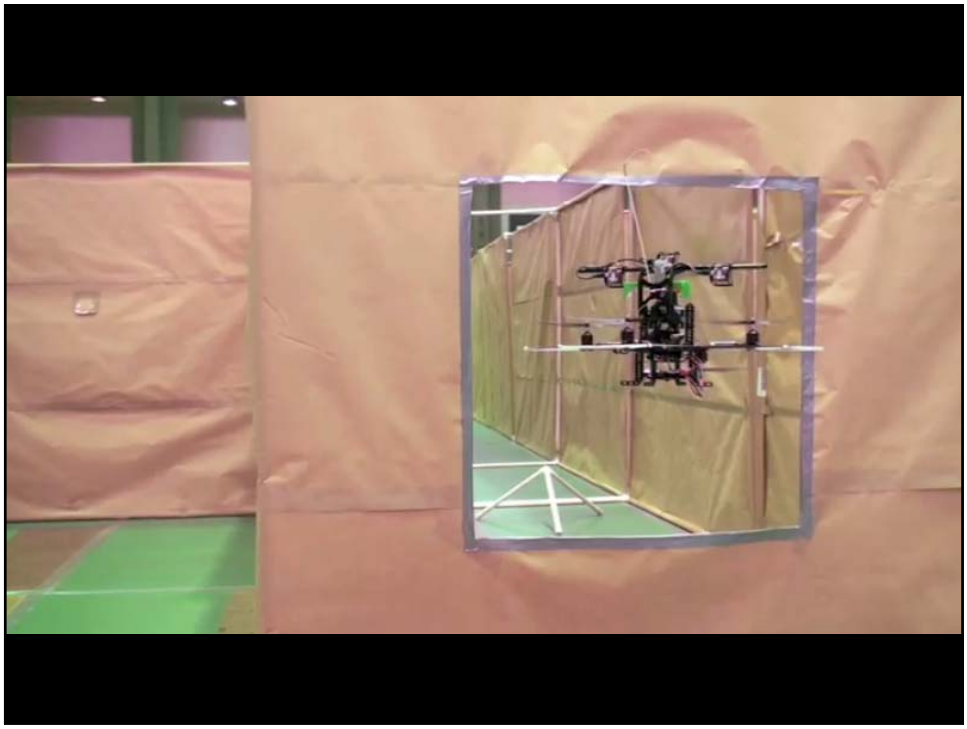
The Belief Roadmap Algorithm

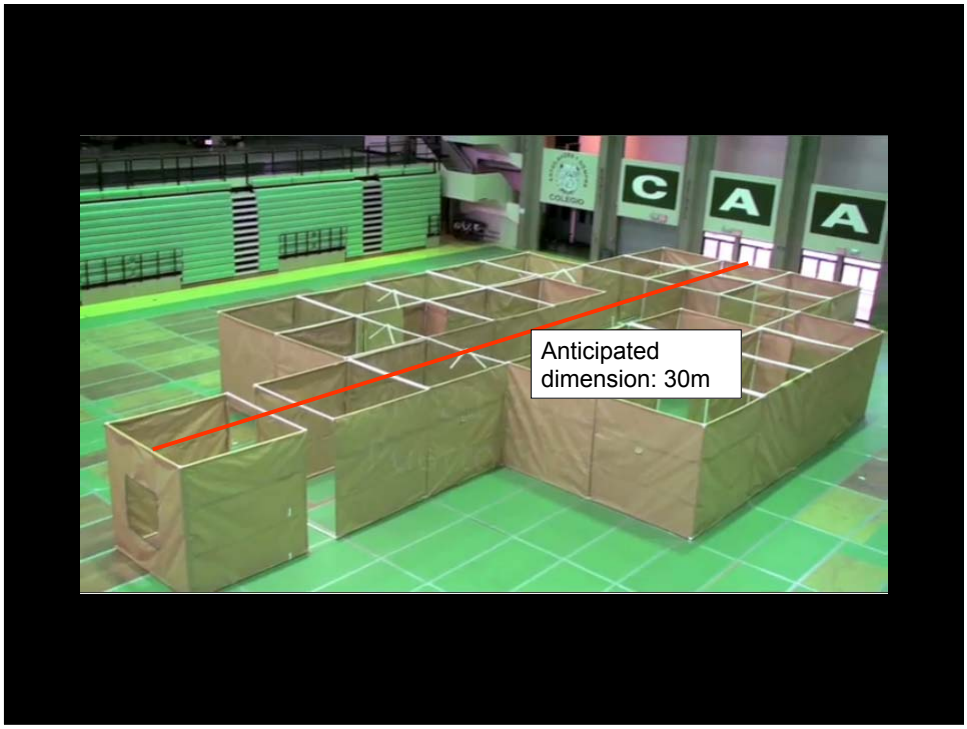


1. Sample means from C_{free}, build graph and transfer functions
2. Propagate covariances by performing graph search

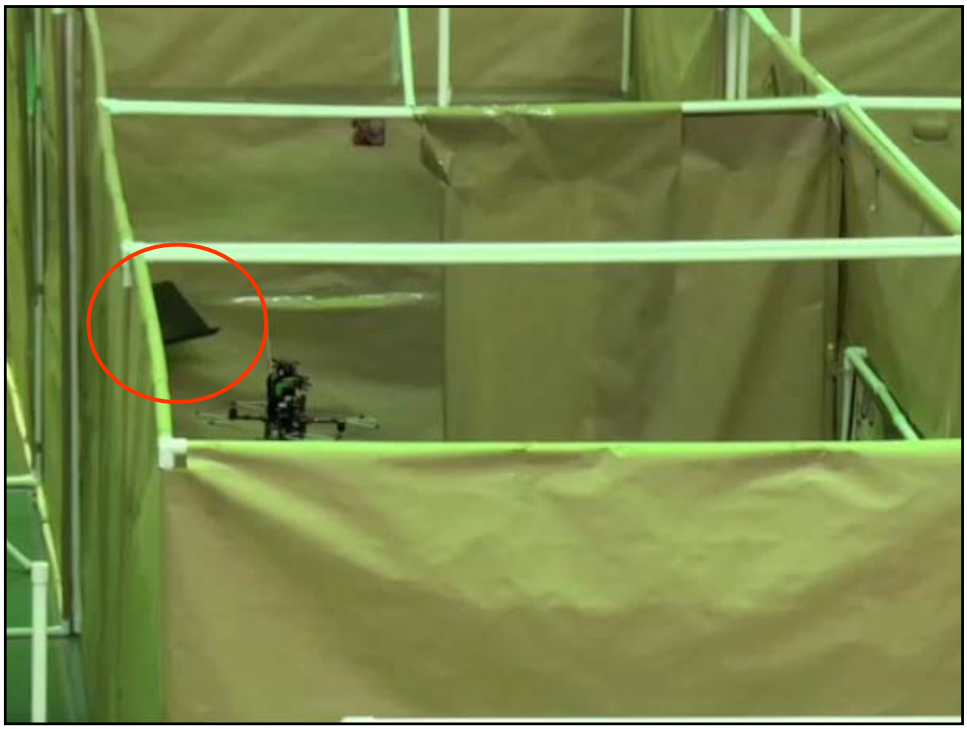

36 m
33 m
3 m
NAVIGATION AID PLACEMENT ZONE
1m x 1m
(minimum)
window
3 m
21 m
18 m
NAVIGATION AID PLACEMENT ZONE
NOTE: Internal wall and obstacle placement is purely notional.
– Actual placements and numbers of rooms may differ.
– No entry door will be less than 1 meter in width.

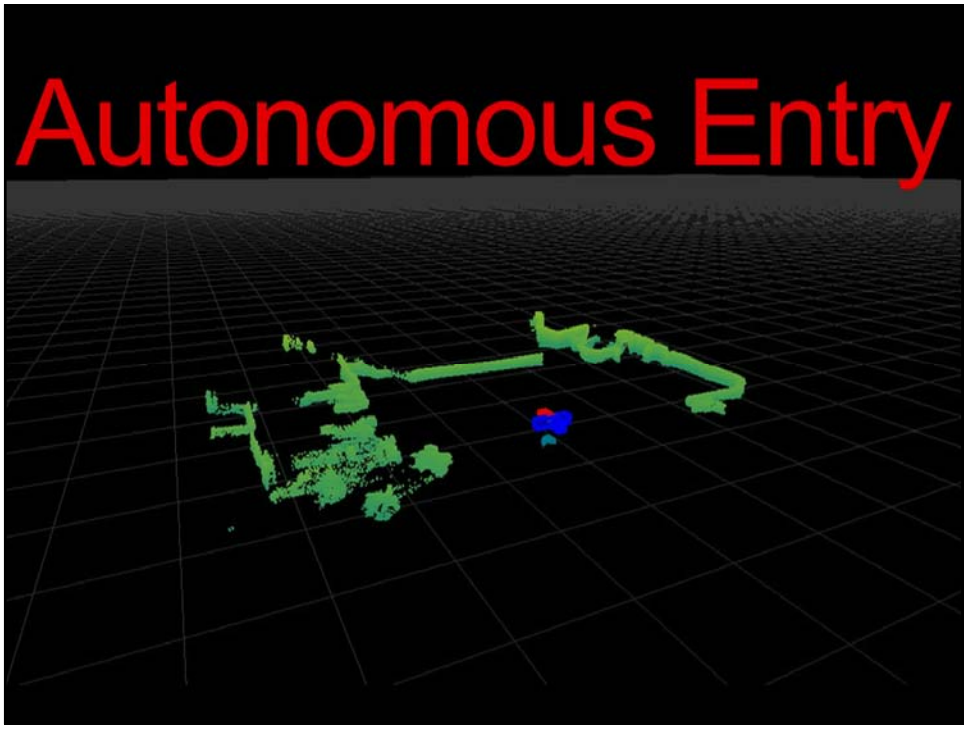

C
A
A




C
A
A
Anticipated
dimension: 30m


U
P
R
R
U
M




Autonomous Entry

Recap: Design Decisions

- How is your map described? This will have an impact on the state space for your planner
  - Is it a list of polygons?
  - Is it a grid map?
- What are you trying to optimize?
  - The fastest path (time)?
  - The shortest path (wear and tear)?
  - The lowest-energy path (battery usage)?
- What kind of search should you use?
  - Can you formulate a reasonably good heuristic?
  - If so, then maybe A* is a good idea
- Physical intuition can yield useful algorithms
  - Potential field method