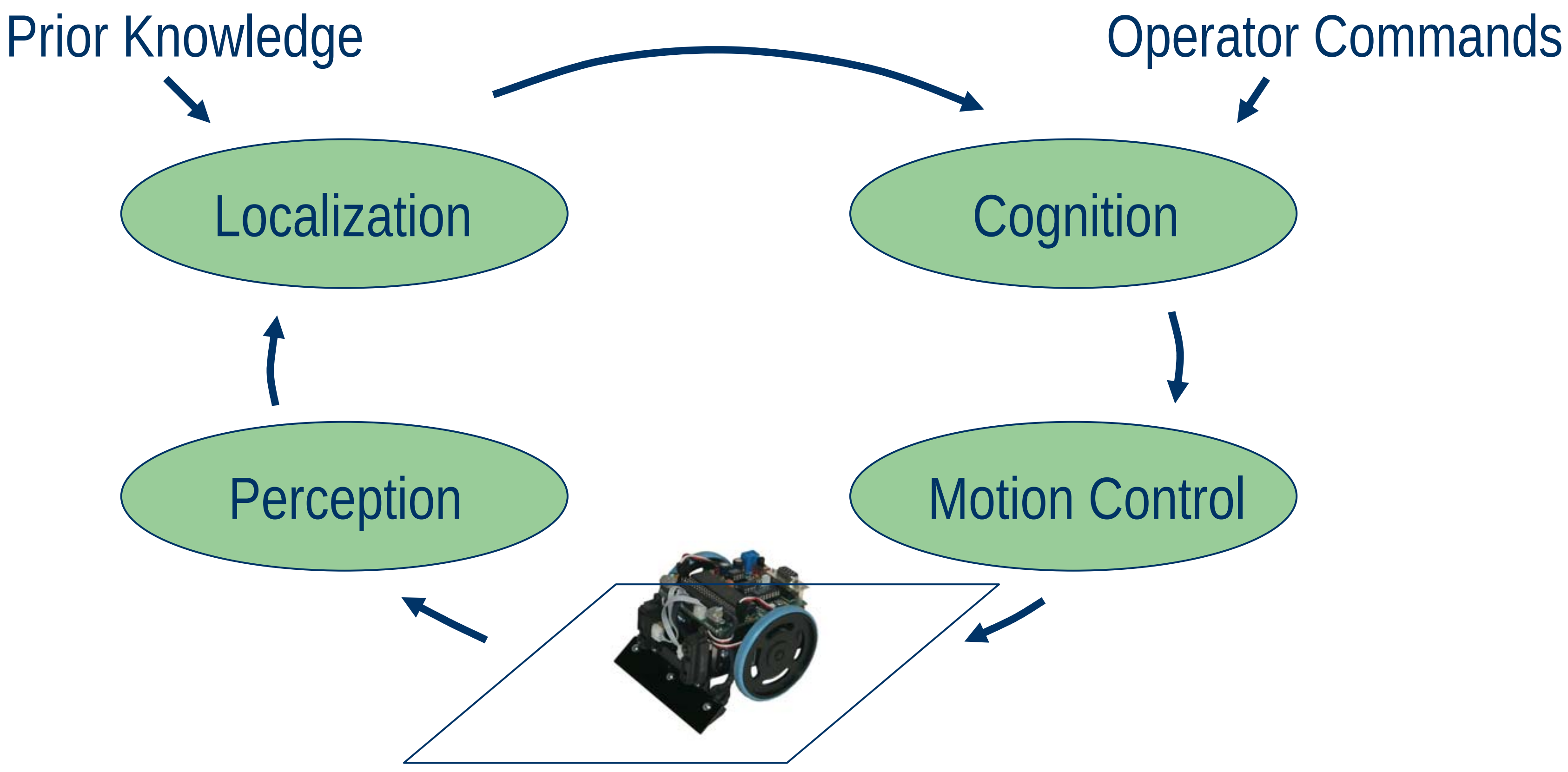
ME 597/747- Lecture 9
Autonomous Mobile Robots

Instructor: Chris Clark

Term: Fall 2004

Figures courtesy of Siegwart & Nourbakhsh

Navigation Control Loop



Motion Planning: General Approach

- Motion planning is usually done with three steps:

Defining the configuration space

↓

Discretization of the configuration space

↓

Searching the configuration space

Motion Planning: Discretizations

1. Roadmap
   - Represent the connectivity of the free space by a network of 1-D curves
2. Cell decomposition
   - Decompose the free space into simple cells and represent the connectivity of the free space by the adjacency graph of these cells
3. Potential field
   - Define a function over the free space that has a global minimum at the goal configuration and follow its steepest descent

Cognition II: Outline

1. Discretizations
    1. Single-QueryProbabilistic Road Maps
    2. Cell Decompositions
    3. Potential Fields
2. Search Algorithms
    1. BFS, DFS, A*

Motion Planning: Probabilistic Road Maps

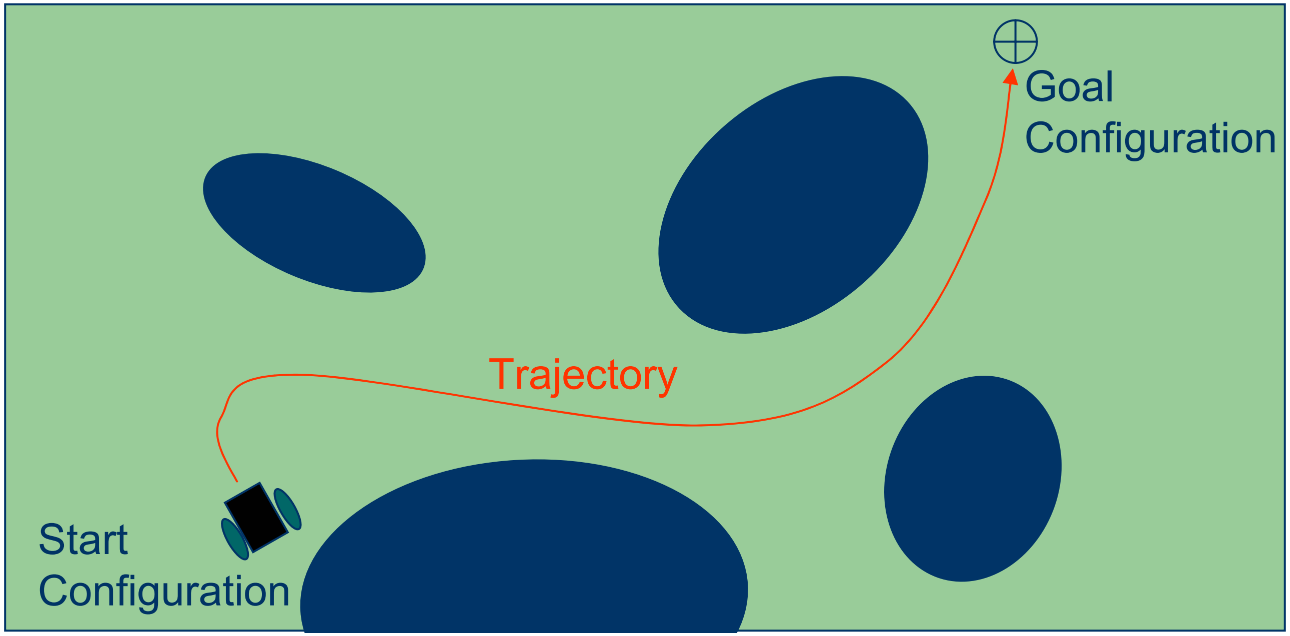
- Single-Query PRMs (a.k.a. Rapidly Exploring Random Trees - RRTs)
  - Try to only sample a subspace of F that is relevant to the problem.
  - Probabilistically complete assuming *C* is *expansive* [Hsu et. al. 2000].
  - Very fast for many applications (allow for on-the-fly planning).

Motion Planning: Probabilistic Road Maps

- Two approaches:
  1. Single Directional:
     - Grow a milestone tree from start configuration until the tree reaches the goal configuration
  2. Bi-Directional:
     - Grow two trees, one from the start configuration and one from the goal configuration, until the two trees meet.
     - Can't take time into configuration space

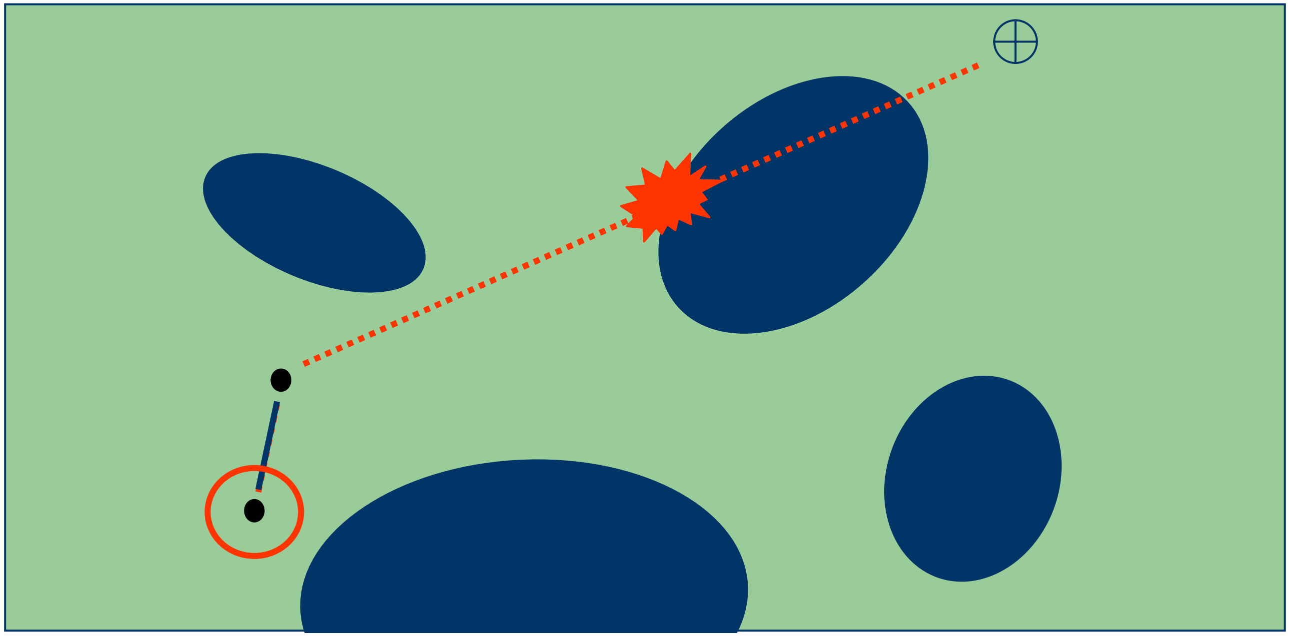
Motion Planning

- Example:



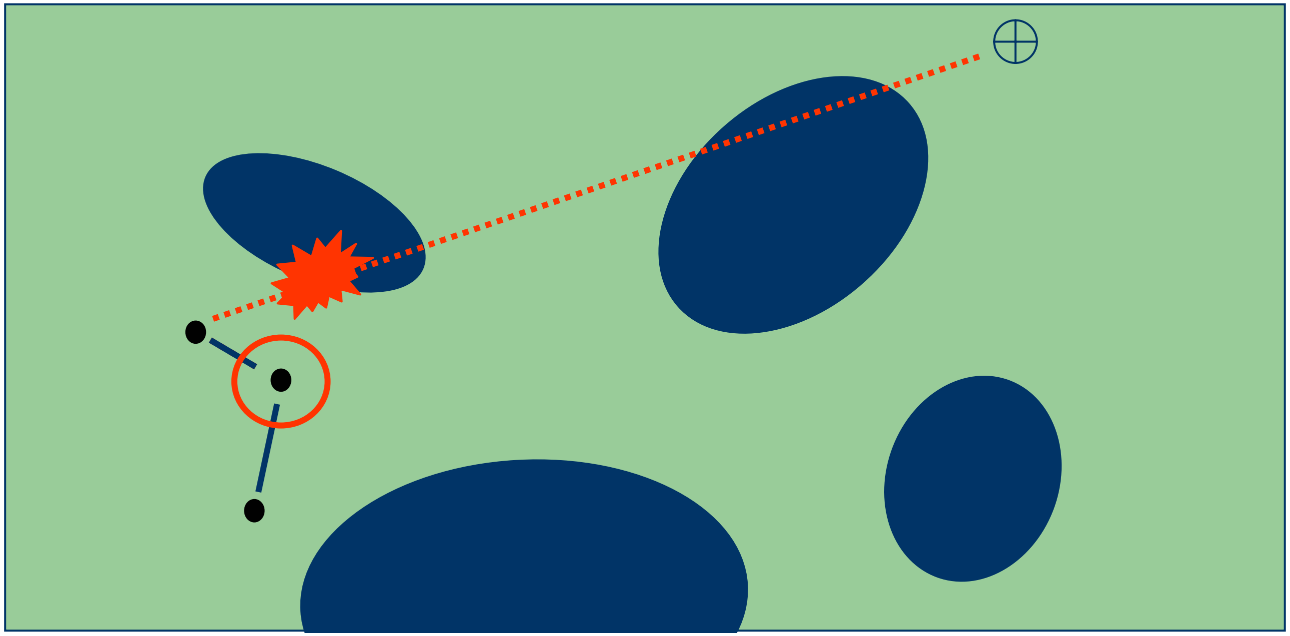
Motion Planning

- Example: Iteration 1



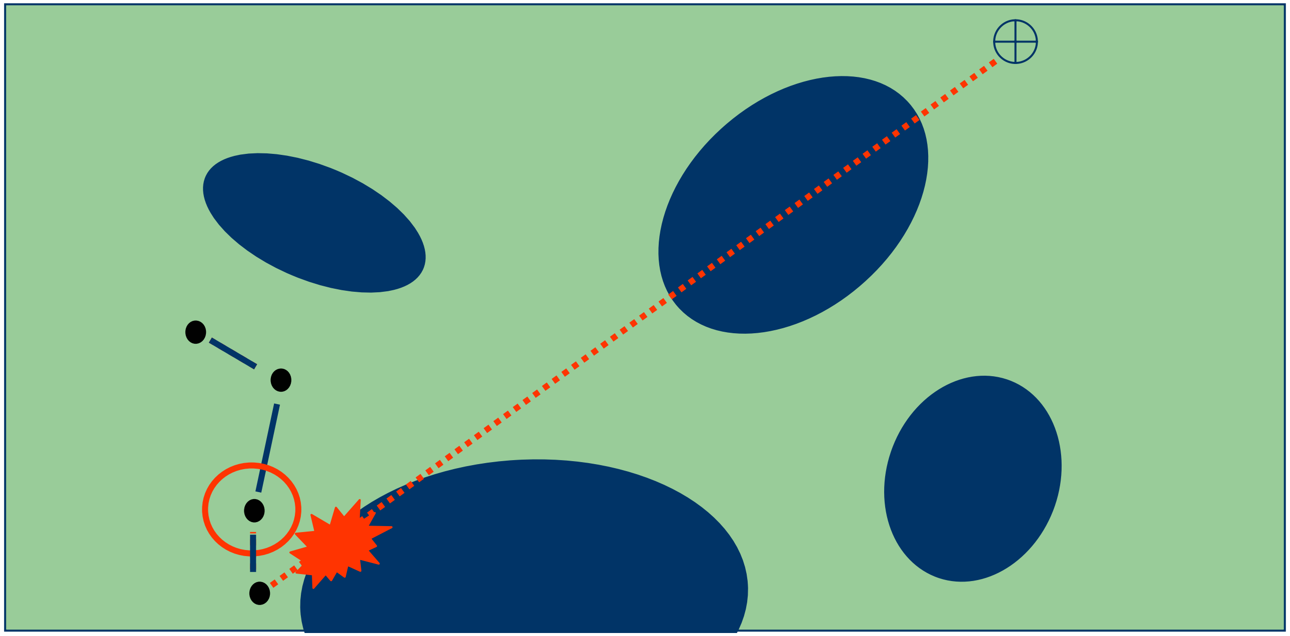
Motion Planning

- Example: Iteration 2



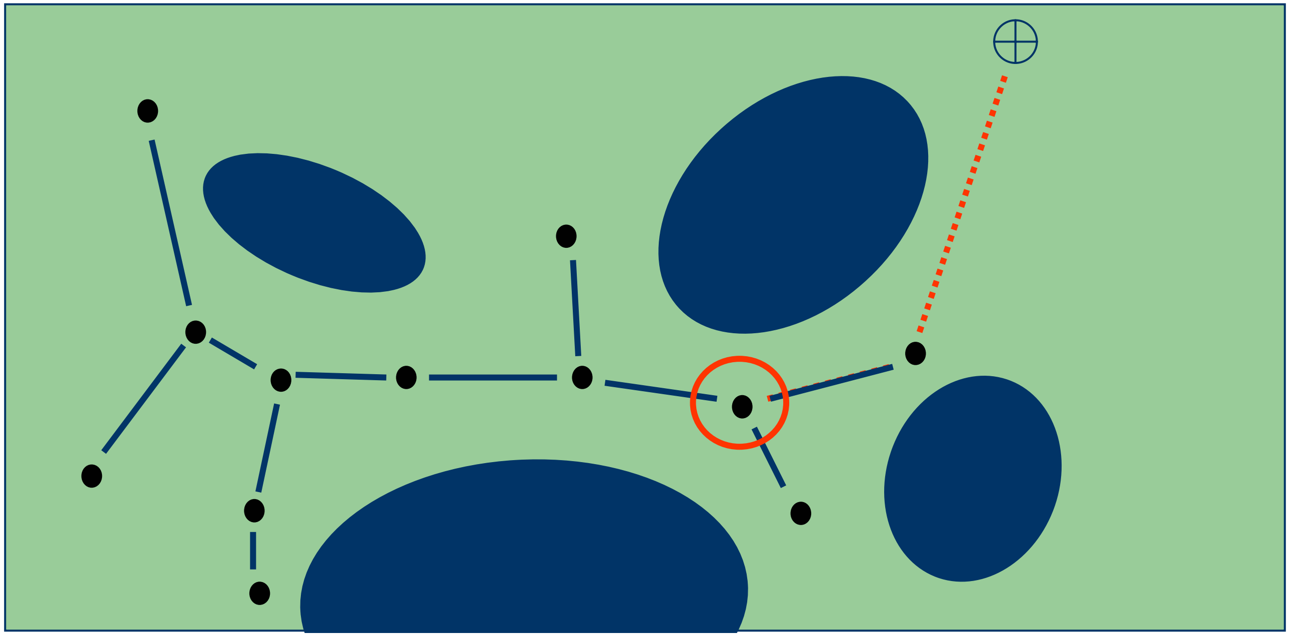
Motion Planning

- Example: Iteration 3



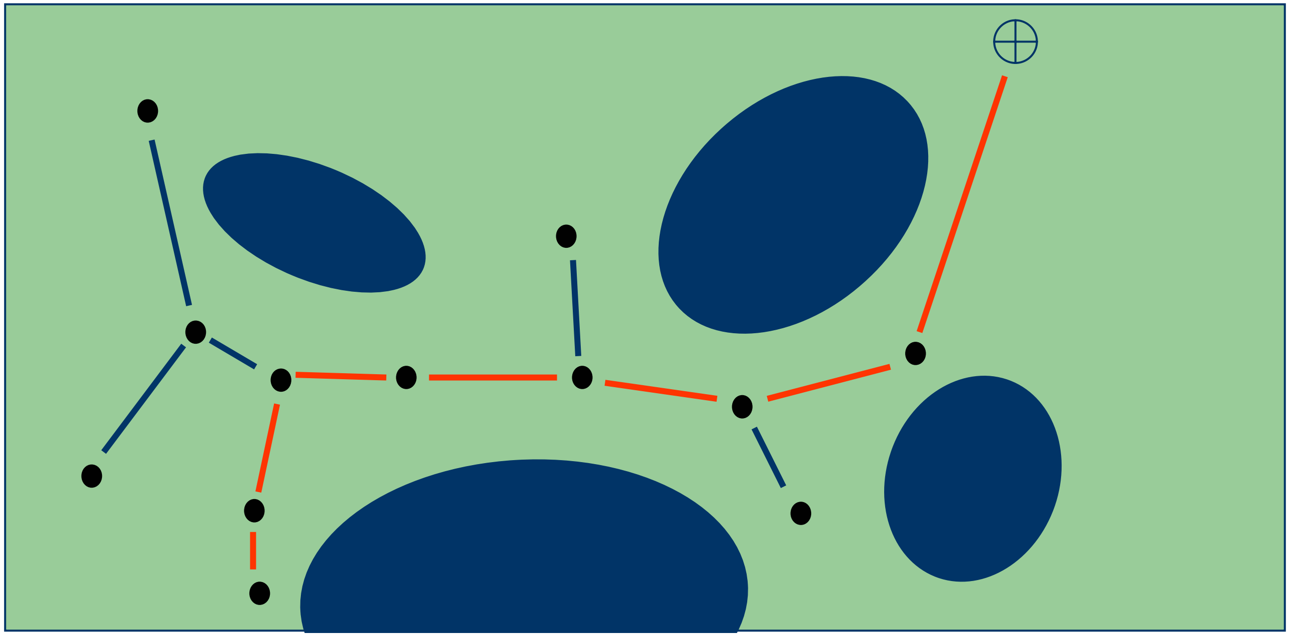
Motion Planning

- Example: Iteration 11



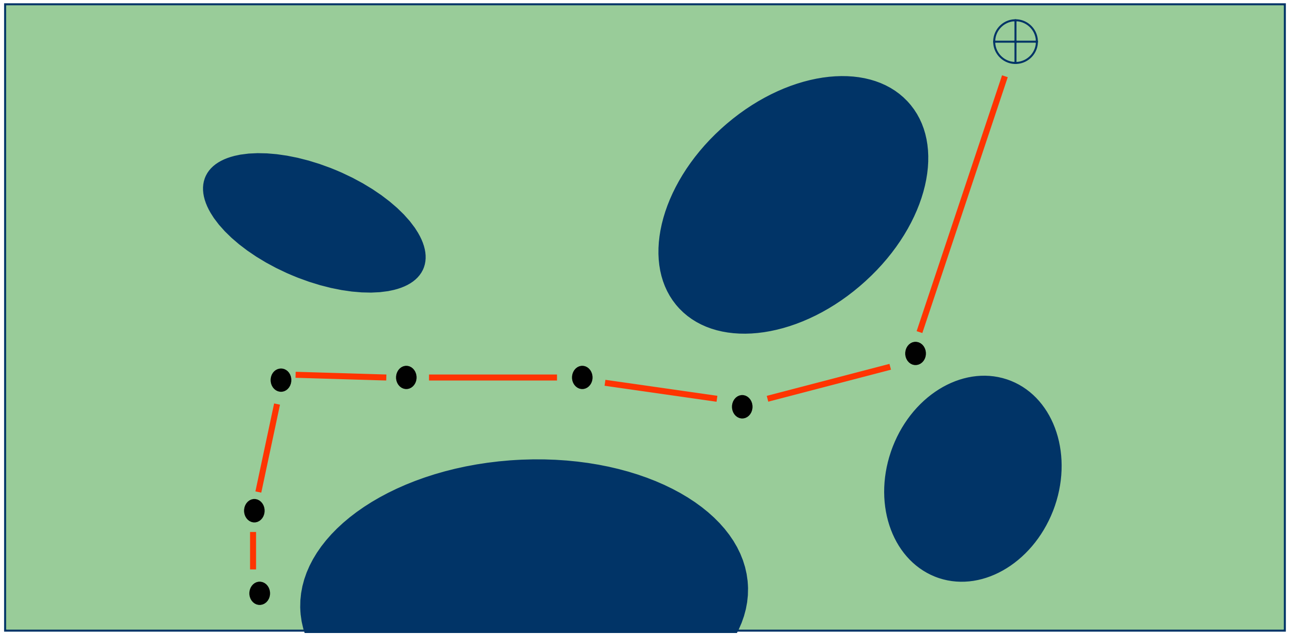
Motion Planning

- Example: Construct Path



Motion Planning

- Example: Construct Path



Probabilistic Road Maps: Learning Phase

- Nomenclature

$R=(N,E)$	RoadMap
N	Set of Nodes
E	Set of edges
c	Configuration
e	edge

Motion Planning: Probabilistic Road Maps

- Algorithm
  1. Add start configuration c_{start} to $R(N,E)$
  2. Loop
  3. Randomly Select New Node c to expand
  4. Randomly Generate new Node c' from c
  5. If edge e from c to c' is collision-free
  6. Add (c, e) to R
  7. If c' belongs to endgame region, return path
  8. Return if stopping criteria is met

Motion Planning: Probabilistic Road Maps

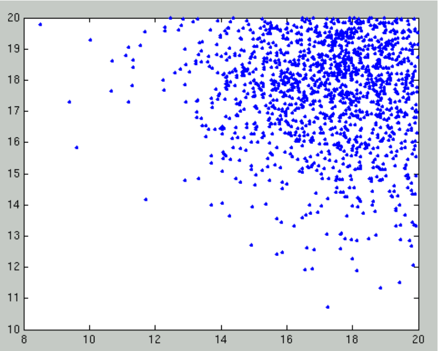
- Sampling strategies
  - Node Selection (step 3)
  - Node Generation (step 4)
  - Endgame Region (step 7)

Motion Planning:
PRM Node Selection

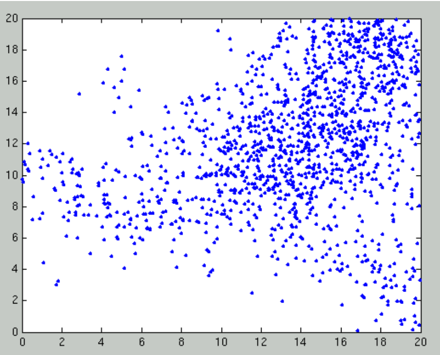
- One could pick the next node for expansion by picking from all nodes in the roadmap with equal probability.
  - This is easy to implement, but leads to poor expansion ⟶ *Clustering*
  - Method is to weight the random selection of nodes to expand, this can greatly affect the roadmap coverage of the configuration space.
  - Want to pick nodes with probability proportional to the inverse of node density.

Motion Planning: PRM Node Selection

- Example:
  - Presented is a 2DOF configuration space where the initial node in the roadmap is located in the upper right corner.
  - After X iterations, the roadmap produced from an unweighted expansion has limited coverage.



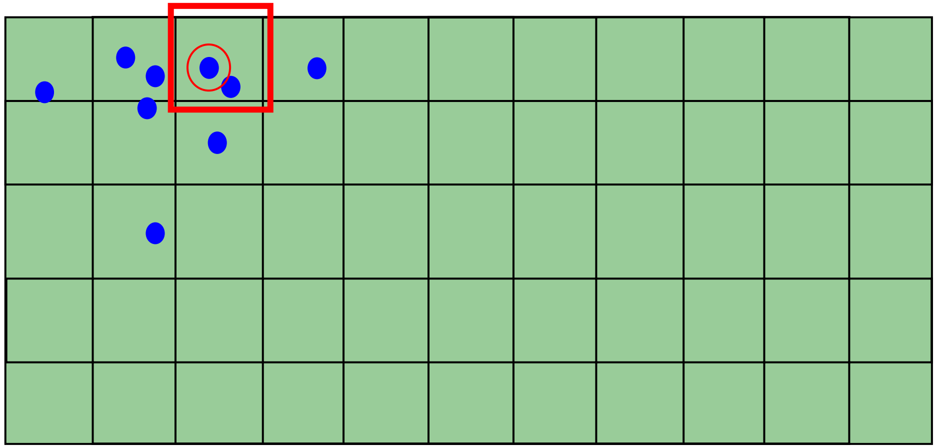
Unweighted



Weighted

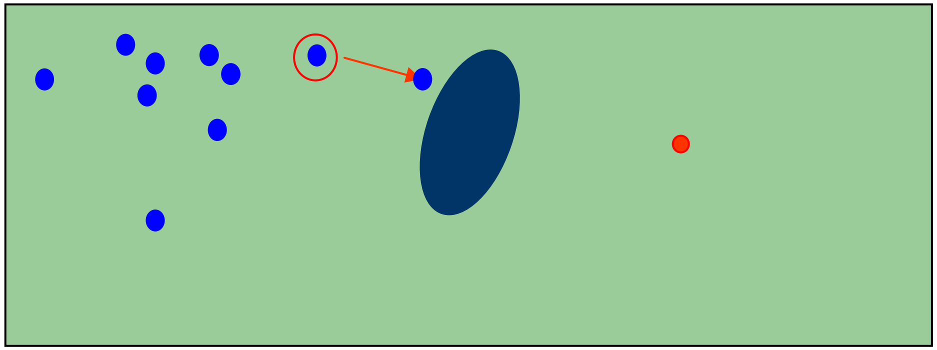
Motion Planning:
PRM Node Selection Technique 1

- The workspace was divided up into cells to form a grid [Kindel 2000].
  - Algorithm:
    1. Randomly pick an occupied cell from the grid.
    2. Randomly pick a milestone in that cell.



Motion Planning: PRM Node Selection Technique 2

- Commonly used in Rapidly exploring Random Trees (RRTs) [Lavalle]
  - Algorithm:
    1. Randomly pick configuration c_{rand} from C.
    2. Find node c from R that is closest to node c_{rand}
    3. Expand from c in the direction of c_{rand}



Motion Planning: Probabilistic Road Maps

- Sampling strategies
  - Node Selection (step 3)
  - Node Generation (step 4)
  - Endgame Region (step 7)

Motion Planning: PRM Milestone Generation

- Use random control inputs to propagate robot from previous node *c* to new configuration *c'*
  - Algorithm:
    1. Randomly select controls *u* and *Δt*
    2. Use known dynamics/kinematics equation *f* of robot to generate new configuration
       $$c' = f\,(c, u, \Delta t)$$
    3. If path from *c* to *c'* is collision-free, then add *c'* to *R*

Motion Planning: PRM Milestone Generation

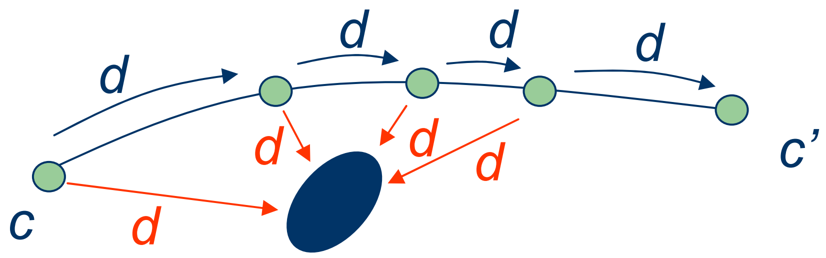
- Example: Differential drive robot
  1. Randomly select controls v_{left}, v_{right} and Δt
  2. Propagate:
     1. Get Δs_{left} and Δs_{right} from $\Delta s = v\Delta t$
     2. Calculate new state *c'* with:

$$c' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

     3. Use iterative search to check for collisions on path.

Motion Planning: PRM Milestone Generation

- Example: Differential drive robot (cont')
  - Iterative Collision checking is simple but not always efficient:
  - Algorithm:
    1. Calculate distance *d* to nearest obstacle
    2. Propagate forward distance *d* along path from *c* to *c'*
    3. If *d* is too small, return *collision*
    4. If *c* reaches or surpasses *c'*, return *collision-free*

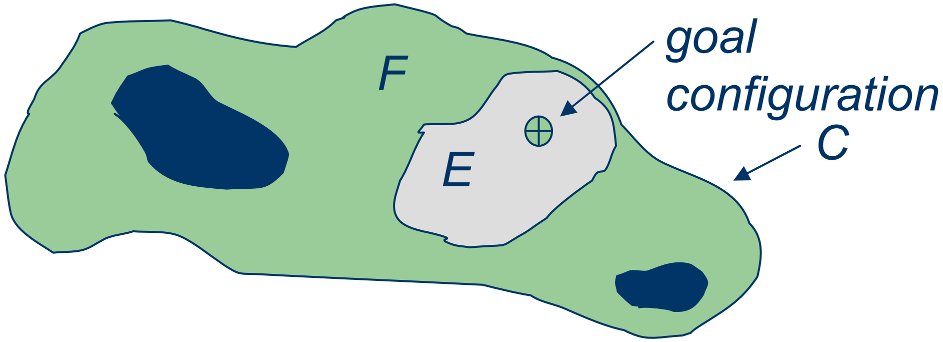


Motion Planning: Probabilistic Road Maps

- Sampling strategies
  - Node Selection (step 3)
  - Node Generation (step 4)
  - Endgame Region (step 7)

Motion Planning: PRM Endgame Region

- We define the endgame region *E*, to be the set of configurations that have a *simple* connection to the goal configuration.
- For each planning problem, we can define a unique method of making *simple* connections.
- This method will inherently define *E*.



Motion Planning: PRM Endgame Region

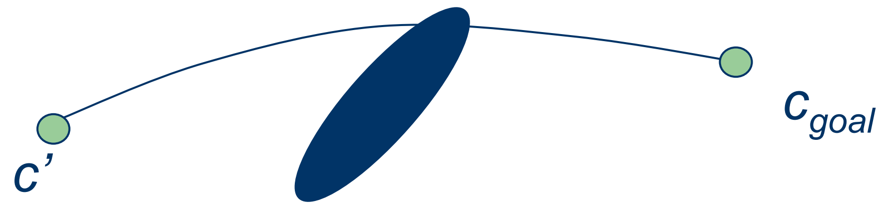
- Given the complexity of most configuration spaces, it is very difficult to model *E*.
- In practice, we develop a simple admissibility test to calculate if a configuration *c'* belongs to the *E*
- At every iteration of the algorithm, this test is used to determine if newly generated configurations are connected to the goal configuration.

Motion Planning: PRM Endgame Region

- In defining *E*, we need two things for good performance:
  1. The region *E* should be large: this increases the chance that a newly generated milestone will belong to *E* and provide us a solution.
  2. The admissibility test to be as fast as possible. This test is conducted at every iteration of the algorithm and will greatly affect the algorithm running time.

Motion Planning: PRM Endgame Region

- Several endgame definitions exist:
  1. The set of all configurations within some radius *r* of the goal configuration
  2. The set of all configurations that have “simple”, collision-free connection with the goal configuration.
     - Example: Use circular arc for differential drive robots.

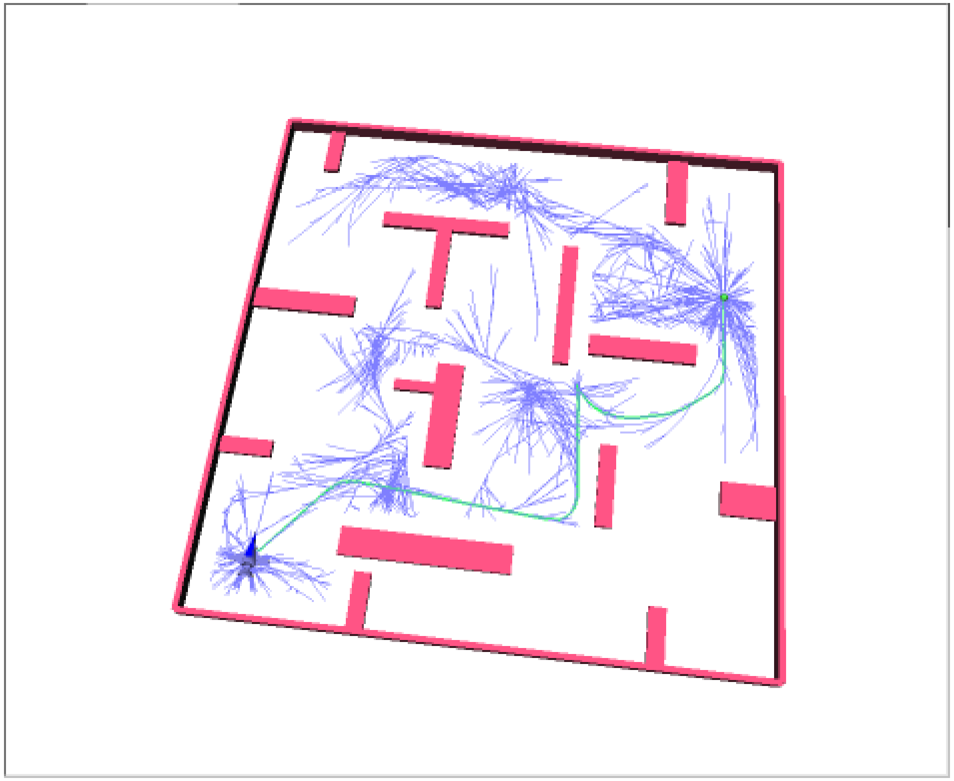


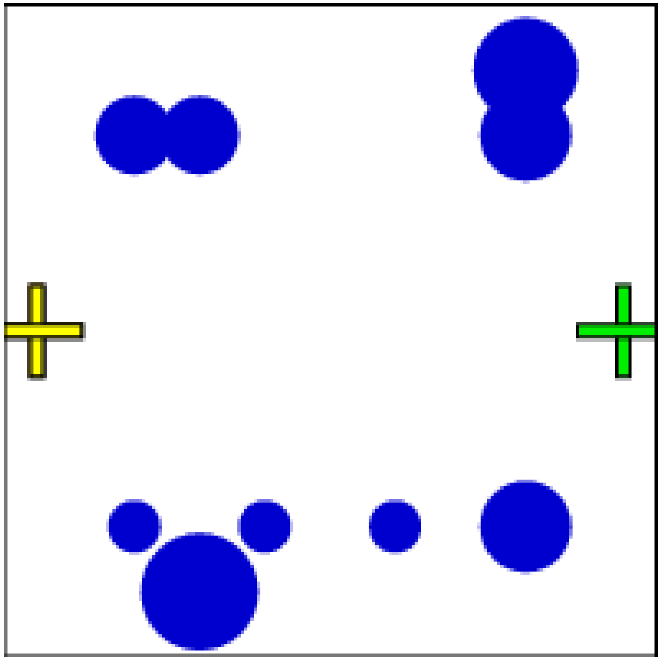
Motion Planning: Probabilistic Road Maps

- Video example [Lavalle]

Motion Planning: Probabilistic Road Maps

- Video example [Lavalle]





Cognition II: Outline

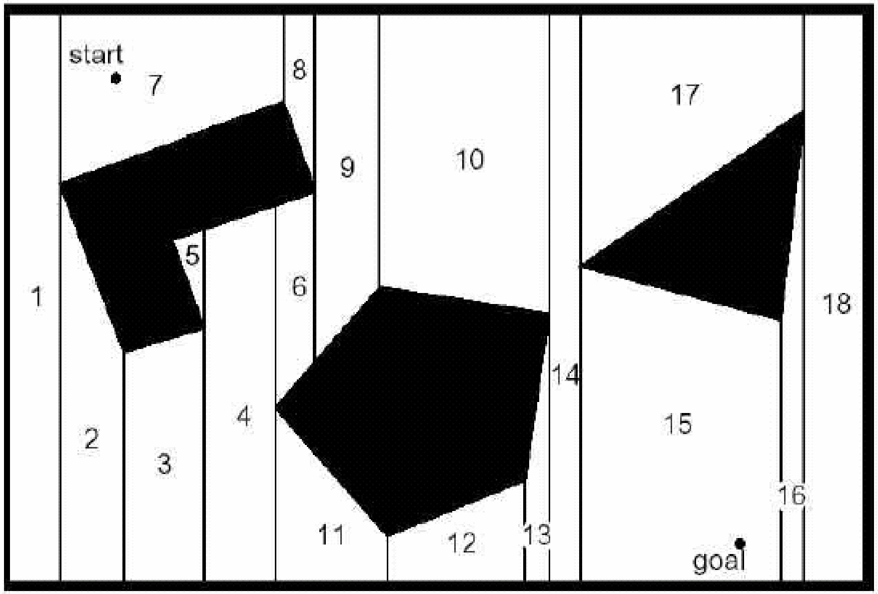
1. Discretizations
   1. Single-QueryProbabilistic Road Maps
   2. Cell Decompositions
   3. Potential Fields
2. Search Algorithms
   1. BFS, DFS, A*

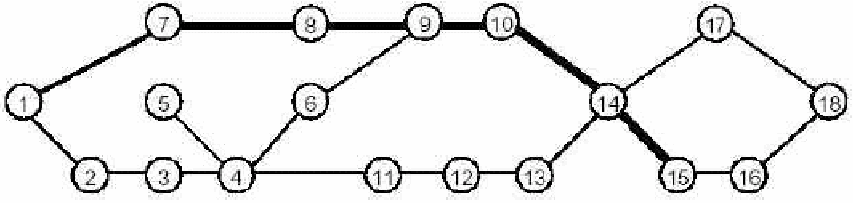
Motion Planning: Cell Decomposition

- Divide space into simple, connected regions (i.e. cells)
- Determine which open cells are adjacent and construct a connectivity graph
- Find cells in which the initial and goal configuration (state) lie and search for a path in the connectivity graph to join them.
- From the sequence of cells found with an appropriate search algorithm, compute a path within each cell.
  - Example: passing through the midpoints of cell boundaries or by sequence of wall following movements.

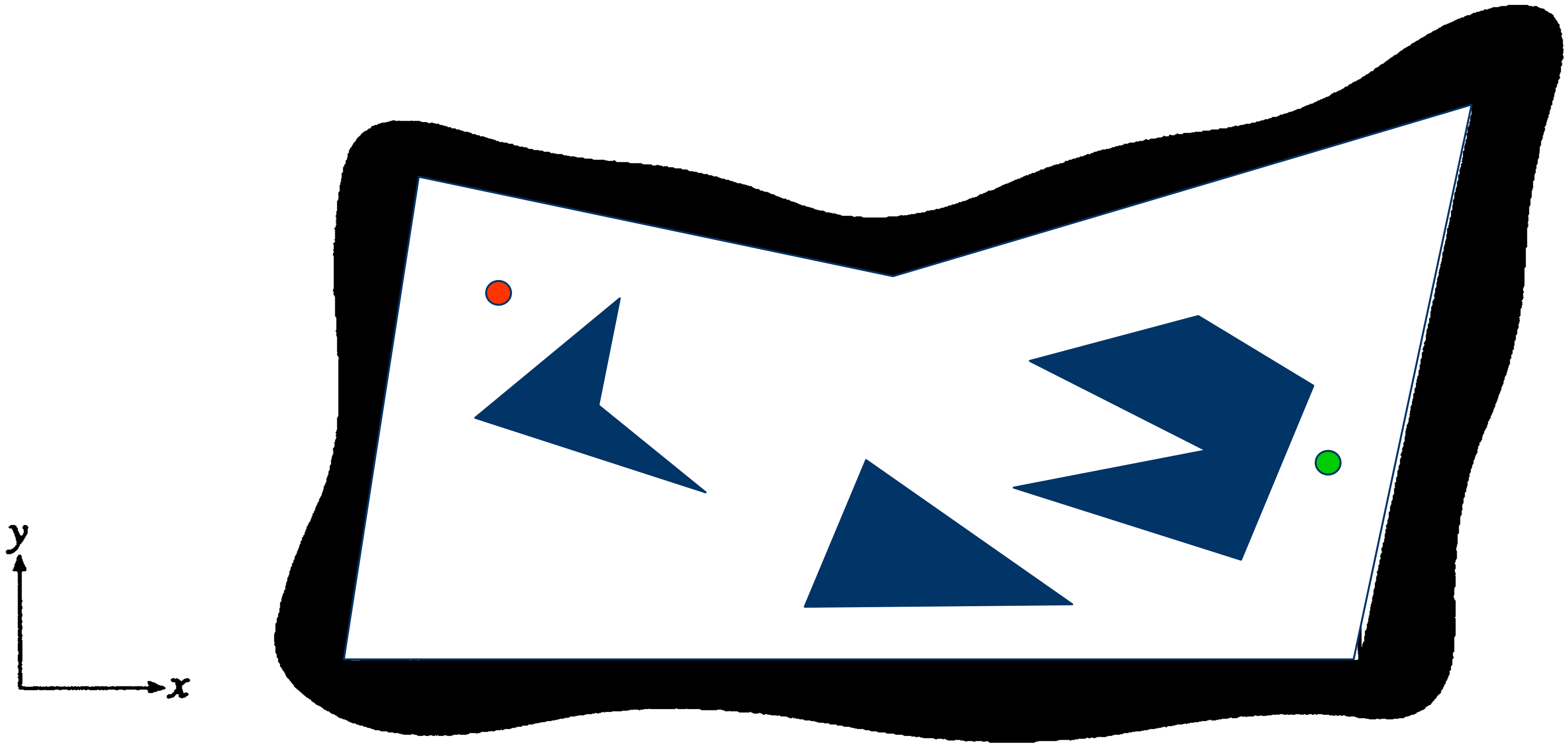
Motion Planning: Cell Decomposition

- Exact Cell Decomposition – Trapezoidal Alg.



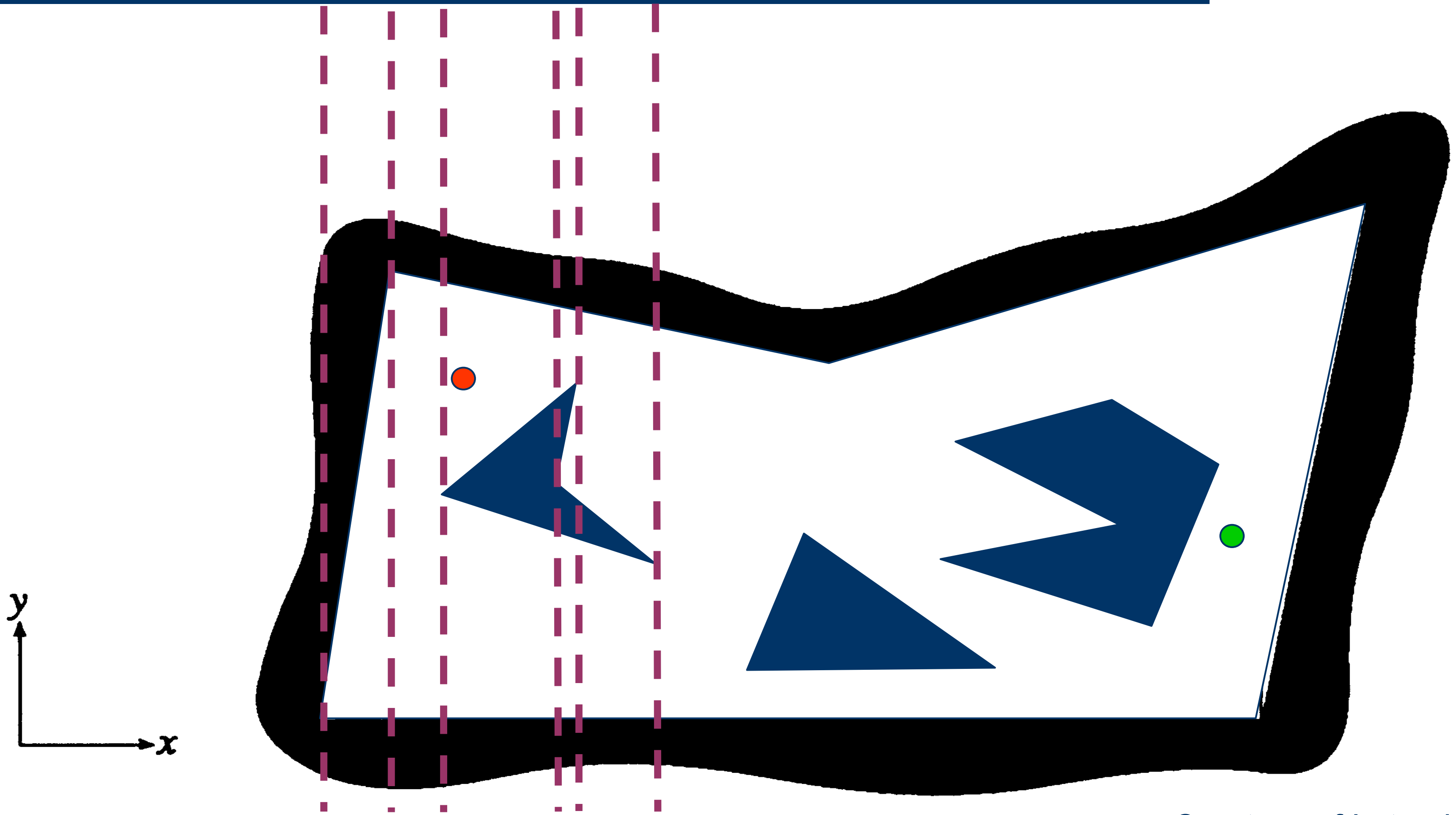


Motion Planning: Cell Decomposition



Courtesy of Latombe

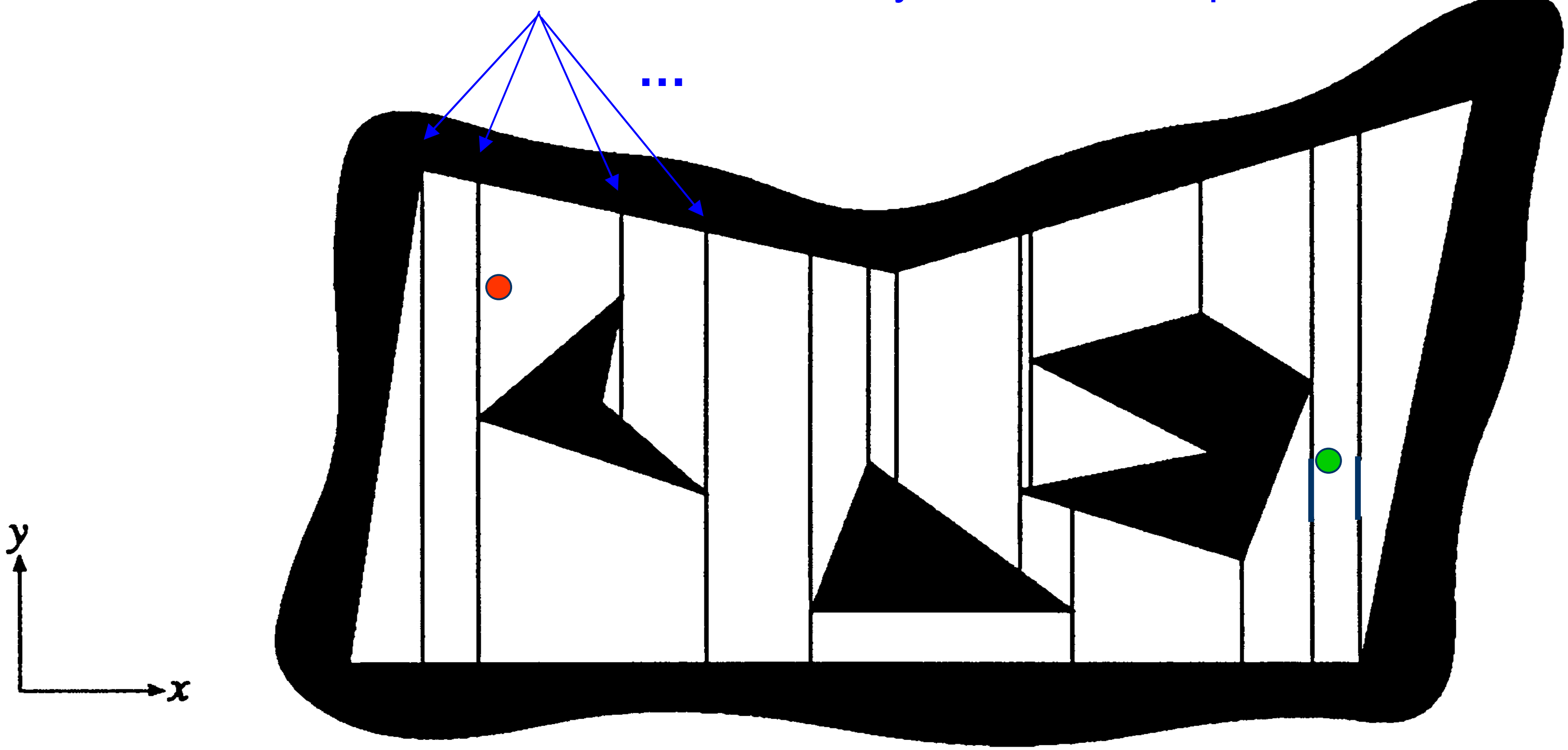
Motion Planning: Cell Decomposition



Courtesy of Latombe

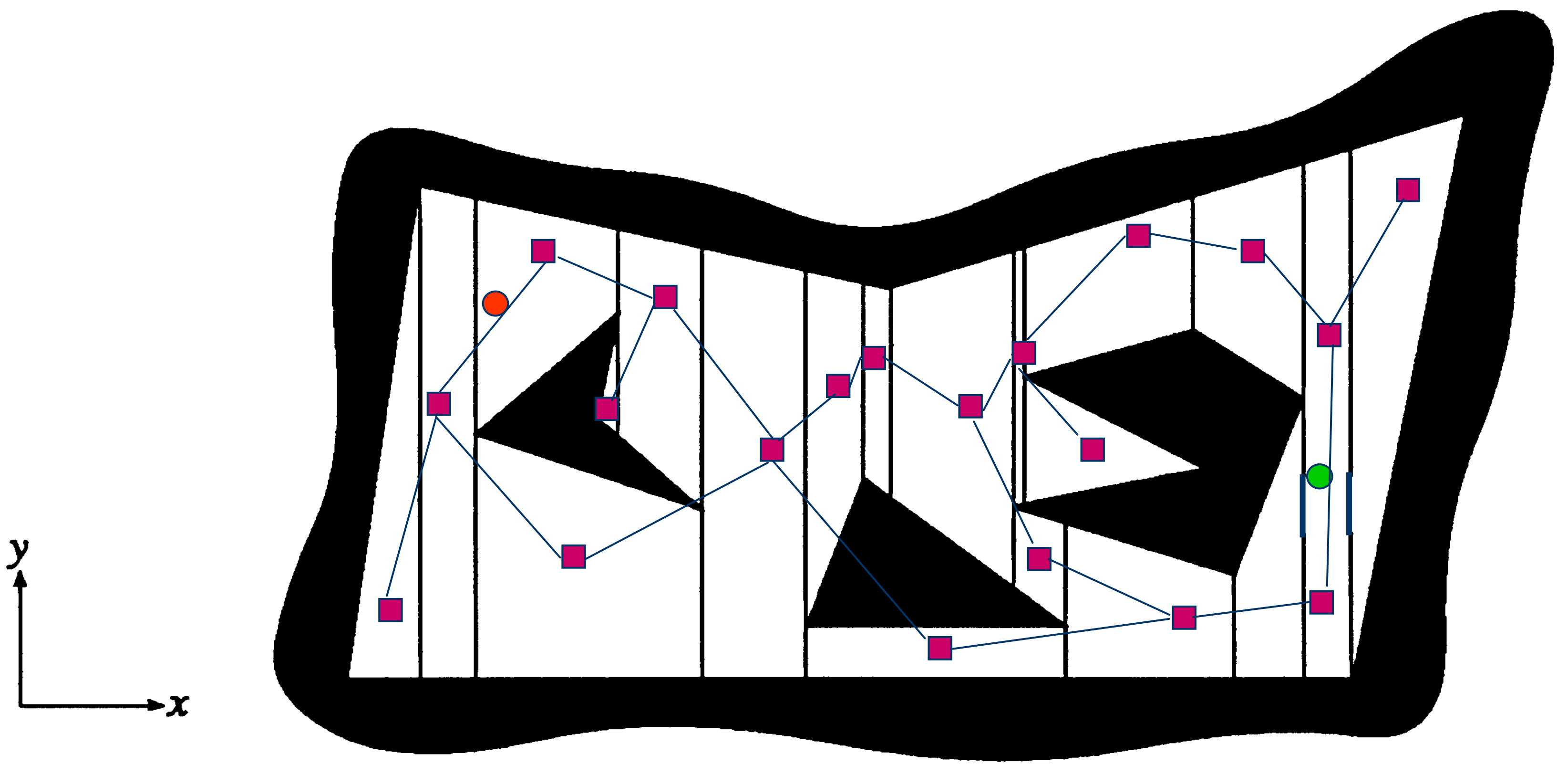
Motion Planning: Cell Decomposition

critical events → criticality-based decomposition



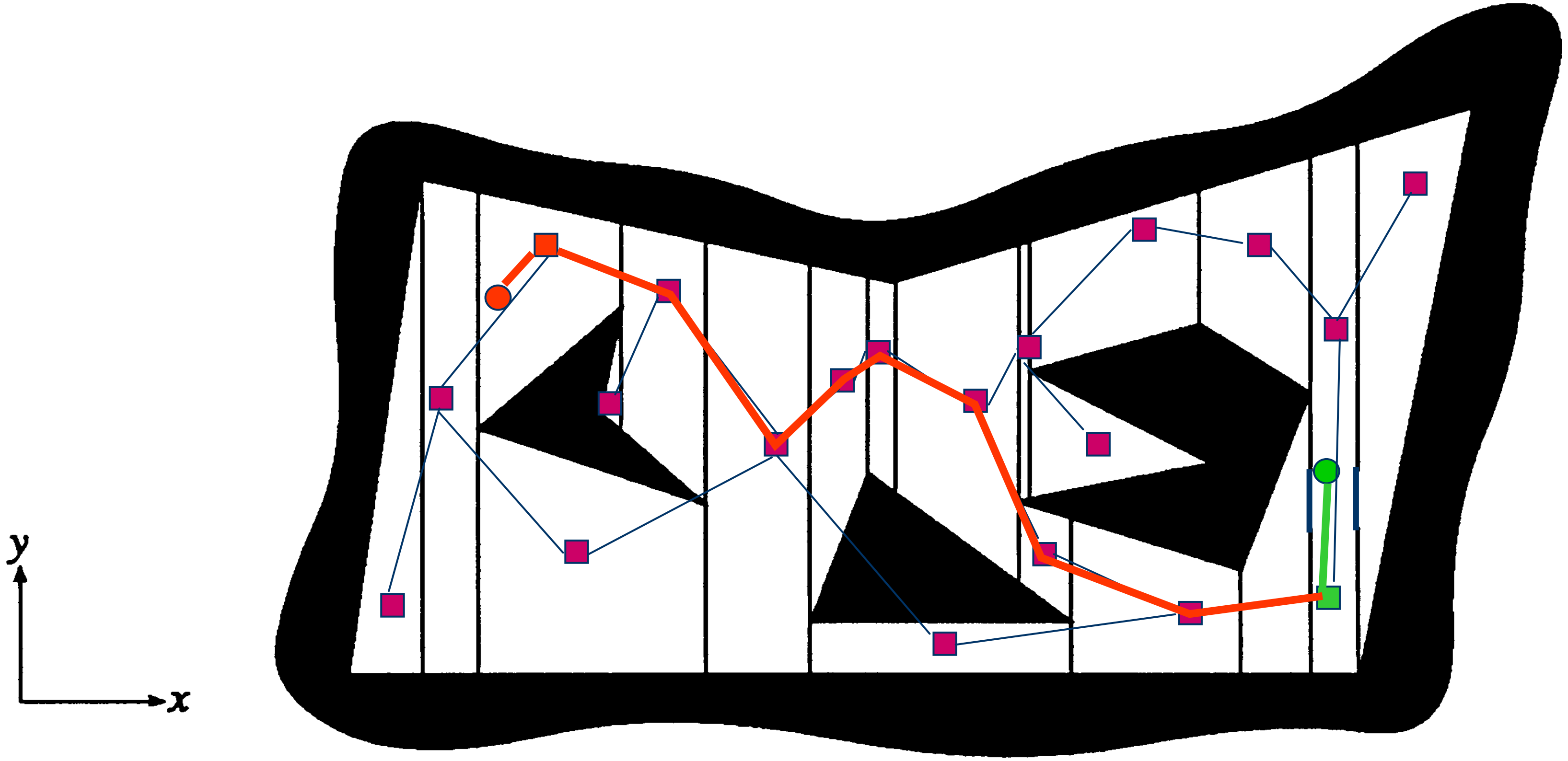
Courtesy of Latombe

Motion Planning: Cell Decomposition



Courtesy of Latombe

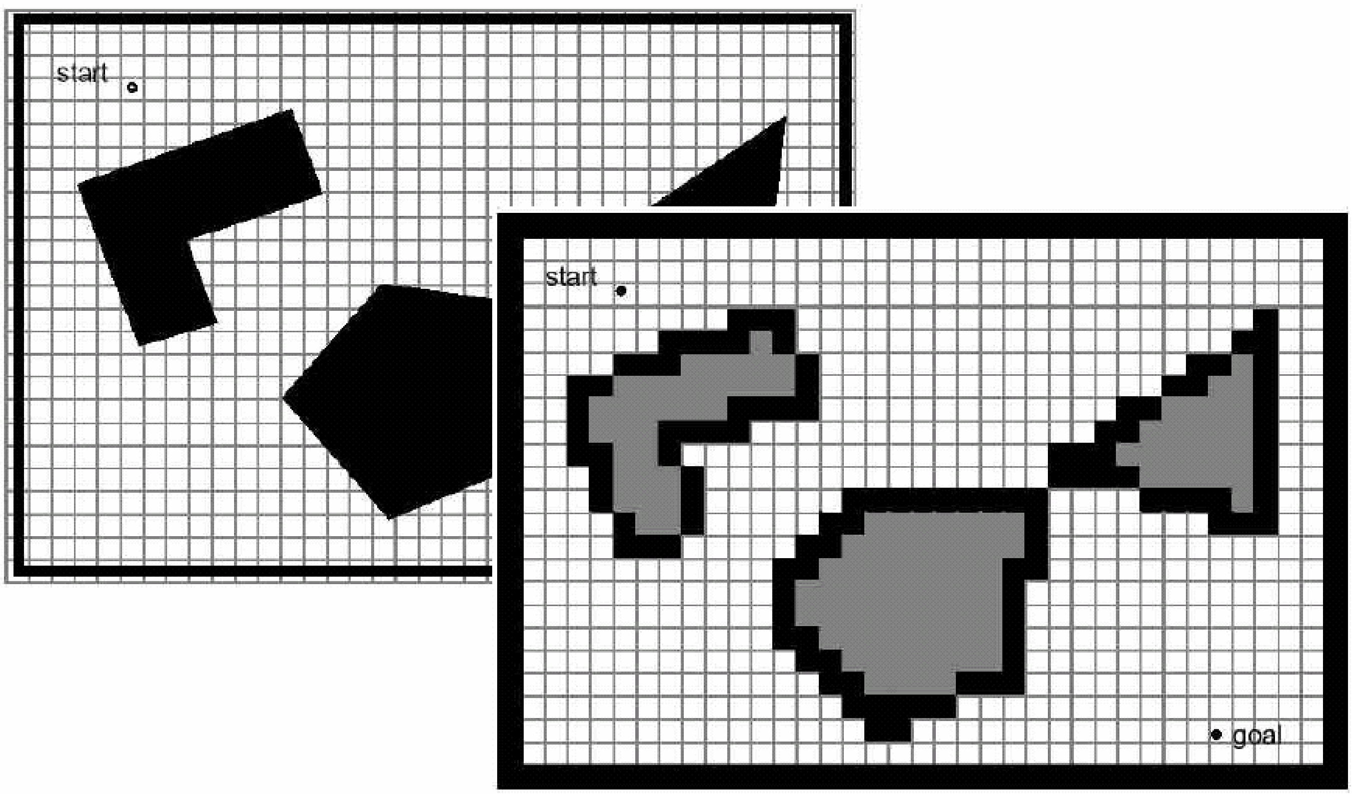
Motion Planning: Cell Decomposition



Courtesy of Latombe

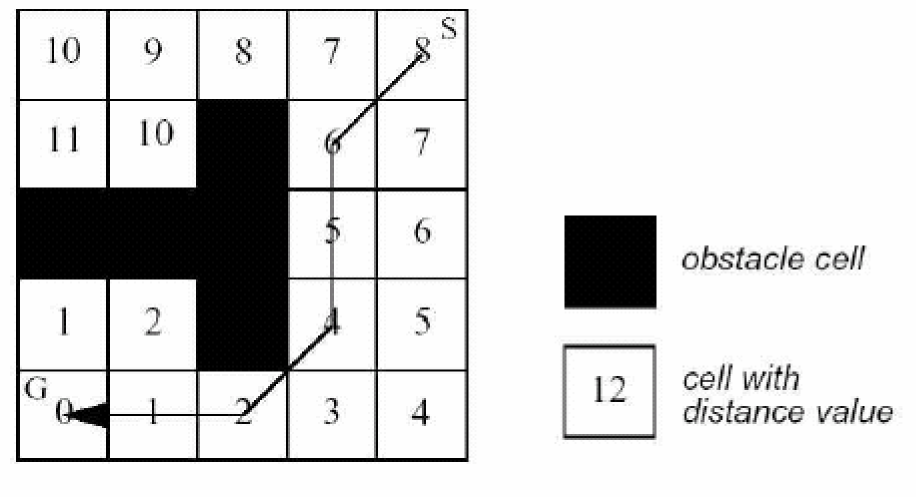
Motion Planning: Cell Decomposition

- Approximate Exact Cell Decomposition



Motion Planning: Cell Decomposition

- Approximate Exact Cell Decomposition
  - Example:

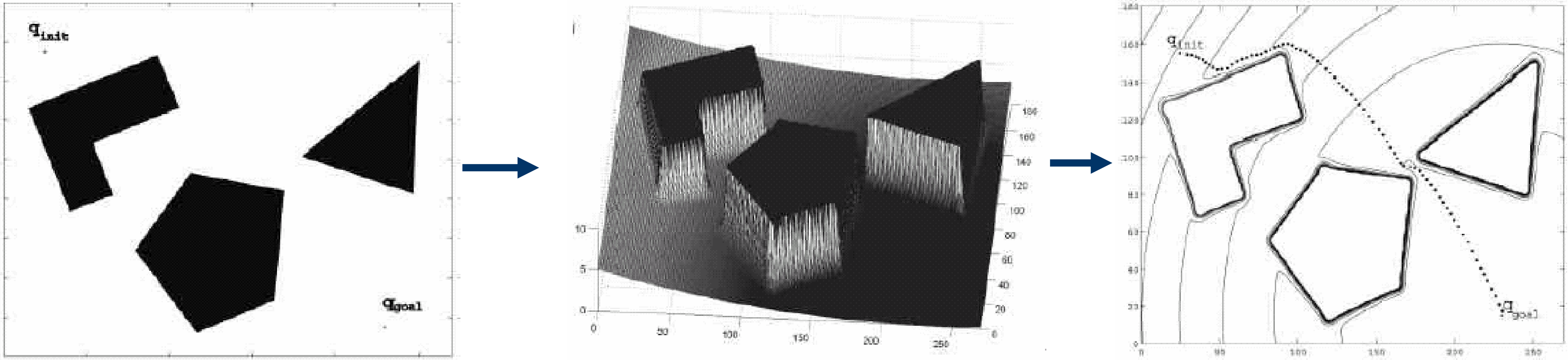


Cognition II: Outline

1. Discretizations
    1. Single-QueryProbabilistic Road Maps
    2. Cell Decompositions
    3. Potential Fields
2. Search Algorithms
    1. BFS, DFS, A*

Motion Planning: Potential Fields

- Robot is treated as a *point under the influence* of an artificial potential field.
  - Generated robot movement is similar to a ball rolling down the hill
  - Goal generates attractive force
  - Obstacles generate repulsive forces



Motion Planning: Potential Field Generation

- Generation of potential field function *U(q)*
  - attracting (goal) and repulsing (obstacle) fields
  - summing up the fields
  - functions must be differentiable
- Generate artificial force field *F(q)*

$$F(q) = -\nabla U(q)$$

$$= -\nabla U_{att}(q) - \nabla U_{rep}(q)$$

$$= \begin{bmatrix} \delta U / \delta x \\ \delta U / \delta y \end{bmatrix}$$

Motion Planning: Potential Field Generation

- Set robot speed (v_x, v_y) proportional to the force $F(q)$ generated by the field
  - the force field drives the robot to the goal
  - if robot is assumed to be a point mass

Motion Planning: Attractive Potential Fields

- Parabolic function representing the Euclidean distance $\rho_{goal}(q) = || q - q_{goal} ||$ to the goal.

$$U_{att}(q) = \frac{1}{2} k_{att} \rho^2_{goal}(q)$$

- Attracting force converges linearly towards 0 (goal)

$$\begin{aligned} F_{att}(q) &= -\nabla U_{att}(q) \\ &= -k_{att}(q - q_{goal}) \end{aligned}$$

Motion Planning: Repulsive Potential Fields

- Should generate a barrier around all the obstacle
  - strong if close to the obstacle
  - not influence if far from the obstacle

$$U_{rep}(q) = \begin{cases} \dfrac{1}{2} k_{rep} \left[\dfrac{1}{\rho(q)} - \dfrac{1}{\rho_0} \right] & \text{if} \quad \rho(q) \le \rho_0 \\ 0 & \text{if} \quad \rho(q) > \rho_0 \end{cases}$$

  - Where $\rho(q)$ is the minimum distance to the object

Motion Planning: Repulsive Potential Fields

- Field is positive or zero and tends to infinity as q gets closer to the object

$$F_{rep}(q) = -\nabla U_{rep}(q)$$

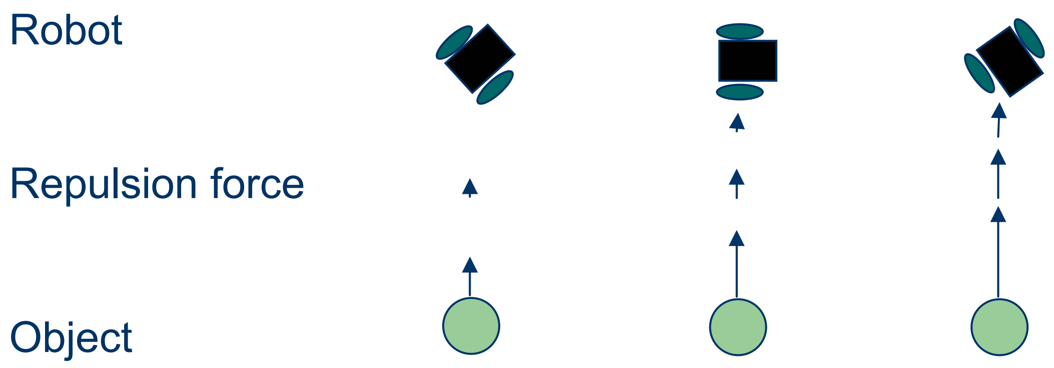
$$= \begin{cases} k_{rep}\left(\dfrac{1}{\rho(q)} - \dfrac{1}{\rho_0}\right)\dfrac{q - q_{obj}}{\rho^3(q)} & \text{if} \quad \rho(q) \le \rho_0 \\ 0 & \text{if} \quad \rho(q) > \rho_0 \end{cases}$$

Motion Planning: Potential Fields

- Get local minimum
- If objects are not *convex* there exists situations where several minimal distances exist and can result in oscillations
- Not complete

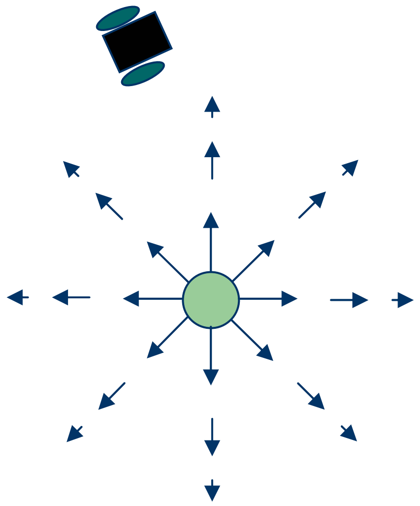
Motion Planning: Potential Fields

- Extended Potential Fields
  - Many modifications to potential fields have been done in order to improve completeness, optimality.
  - Example: Rotation potentials
    - Can increase potential depending on orientation of robot

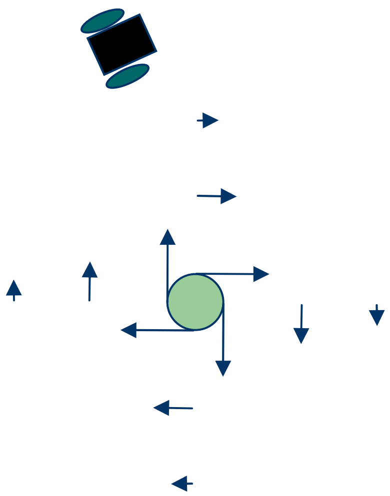


Motion Planning: Potential Fields

- Extended Potential Fields
  - Also, can use rotational fields in one direction



Linear source



Rotational source

Cognition II: Outline

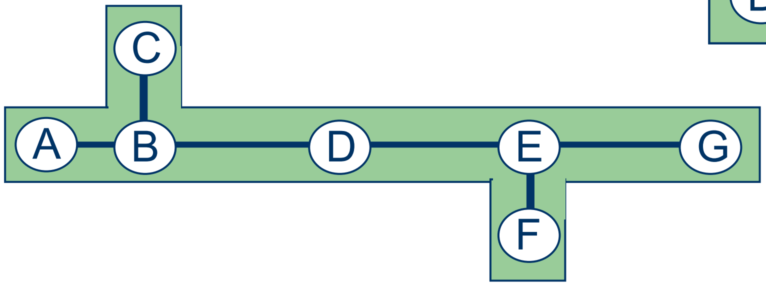
1. Discretizations
   1. Single-QueryProbabilistic Road Maps
   2. Cell Decompositions
   3. Potential Fields
2. Search Algorithms
   1. BFS
   2. DFS
   3. A*

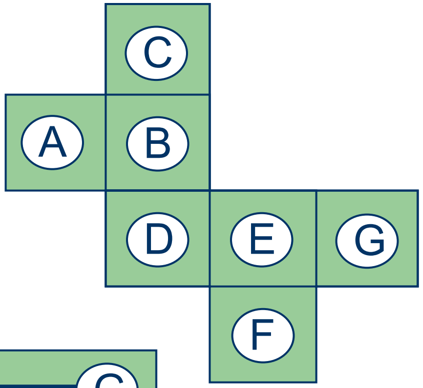
Motion Planning: Search Algorithms

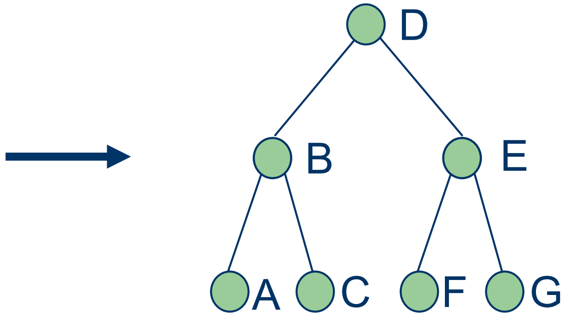
- Given a graph *R(N,E),* (topological map or grid), how do we find a connected path from any two nodes in the *R?*
  1. Breadth First Search
  2. Depth First Search
  3. A*

Motion Planning: Tree Search

- Once the space is discretized, we can perform a tree search
  - Note: we know the connections, not the whole tree!
  - Example: How do we get from D to G?

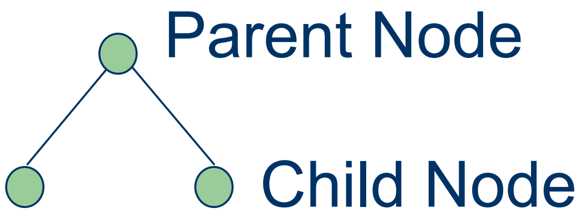




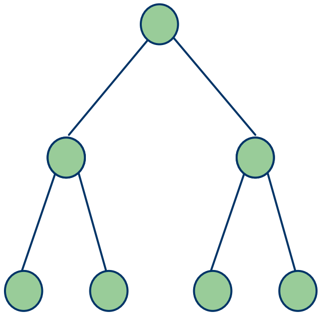


Motion Planning: Breadth First Search

- Tree nomenclature:



- Algorithms differ in the order in which they search the branches (edges) of the tree

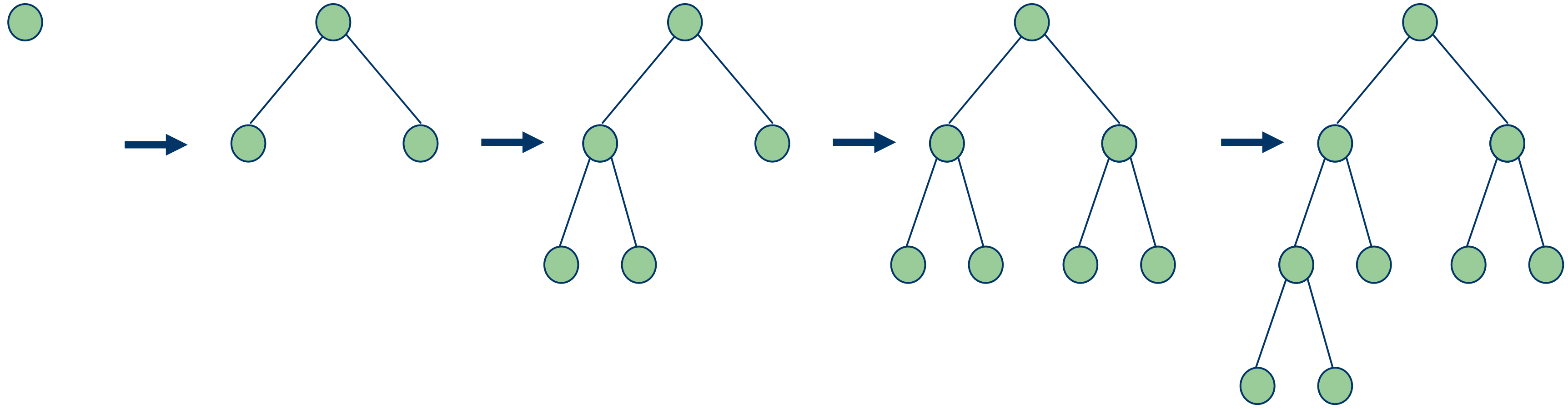


Cognition II: Outline

1. Discretizations
   1. Single-QueryProbabilistic Road Maps
   2. Cell Decompositions
   3. Potential Fields
2. Search Algorithms
   1. BFS
   2. DFS
   3. A*

Motion Planning: Breadth First Search

- Search a tree, one level at a time.



Motion Planning: Breadth First Search

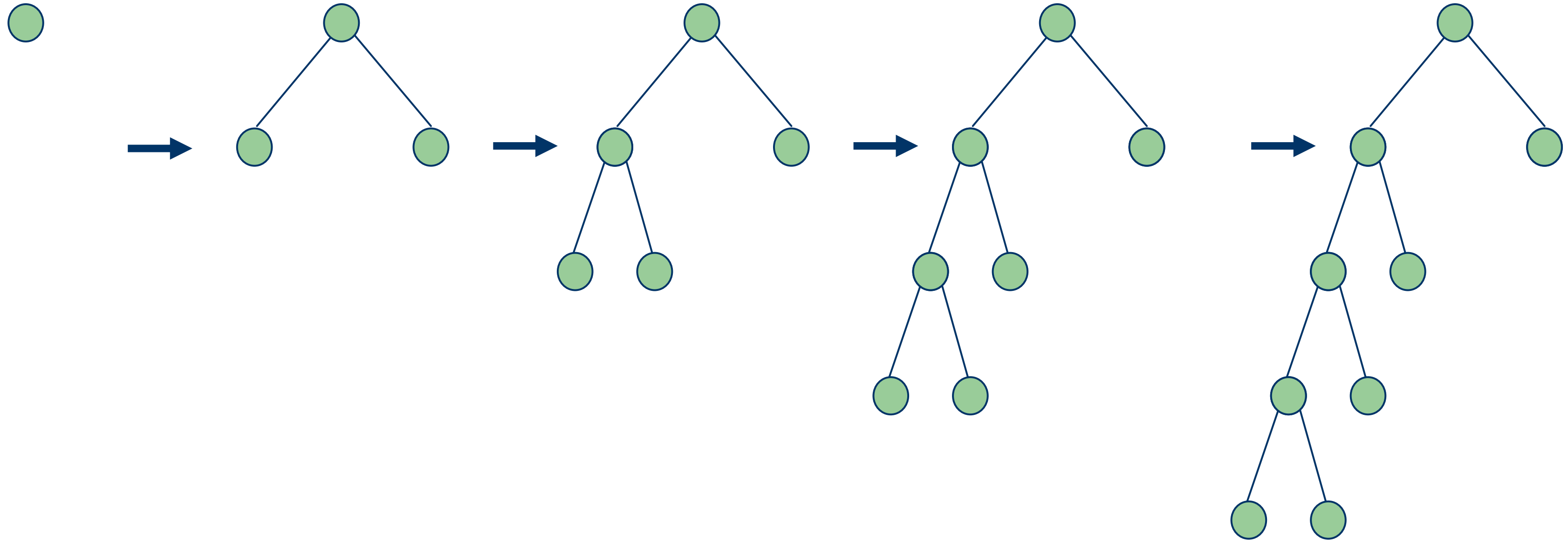
- Complete
- Optimal if cost is increasing with path depth.
- Time complexity $O(b^d)$, where b is the branching factor and d is the depth
- Space (memory) complexity $O(b^d)$

Cognition II: Outline

1. Discretizations
   1. Single-QueryProbabilistic Road Maps
   2. Cell Decompositions
   3. Potential Fields
2. Search Algorithms
   1. BFS
   2. DFS
   3. A*

Motion Planning: Depth First Search

- Search a tree, always expand to deepest level until final depth is reached.



Motion Planning: Depth First Search

- NOT Complete if infinite depth
- NOT Optimal
- Time complexity $O(b^m)$, where b is the branching factor and m is the depth
- Space (memory) complexity $O(bm)$
- Good if there are many solutions

Cognition II: Outline

1. Discretizations
   1. Single-QueryProbabilistic Road Maps
   2. Cell Decompositions
   3. Potential Fields
2. Search Algorithms
   1. BFS
   2. DFS
   3. A*

Motion Planning: A* Search

- There are a set of algorithms called “Best-First Search”
- They try to search the children of the “best” node to expand.
- A* has become incredibly popular because it attempts to make the best node the one that will find the optimal solution and do so in less time.

Motion Planning: A* Search

- We evaluate a node *n* for expansion based on the function:

$$f(n) = g(n) + h(n)$$

- Where

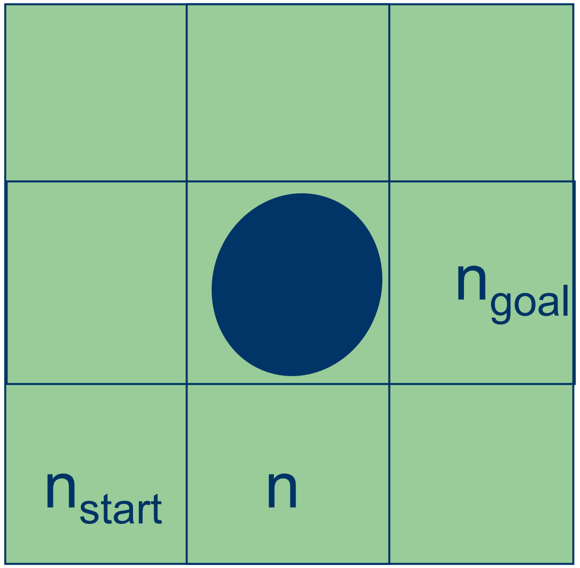
  $g(n)$ = path cost from the start node to *n*

  $h(n)$ = estimated cost of the cheapest path from node n to the goal

Motion Planning: A* Search

- Example: Cost for one particular node

$$f(n) = g(n) + h(n)$$



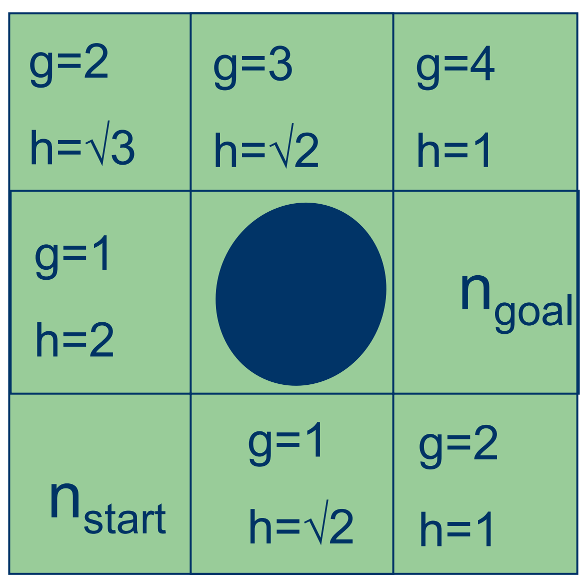
$g(n) = 1$

$h(n) = \sqrt{2}$

Motion Planning:
A* Search

- Example: Cost for each node

$$f(n) = g(n) + h(n)$$



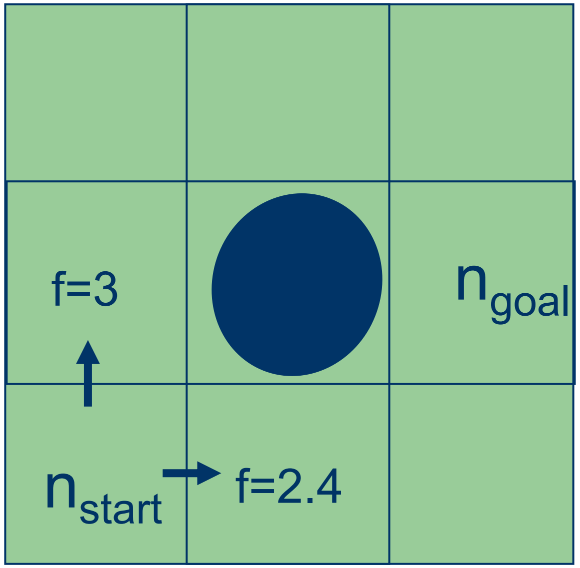
Motion Planning: A* Search

- The strategy is to expand the node with the cheapest path (lowest *f*).
- This is proven to be complete and optimal, if *h(n)* is an *admissible* heuristic.
- Here, an admissible heuristic is one that never *overestimates* the cost to the goal
  - Example: the Euclidean distance.

Motion Planning: A* Search

- Search example: Iteration 1

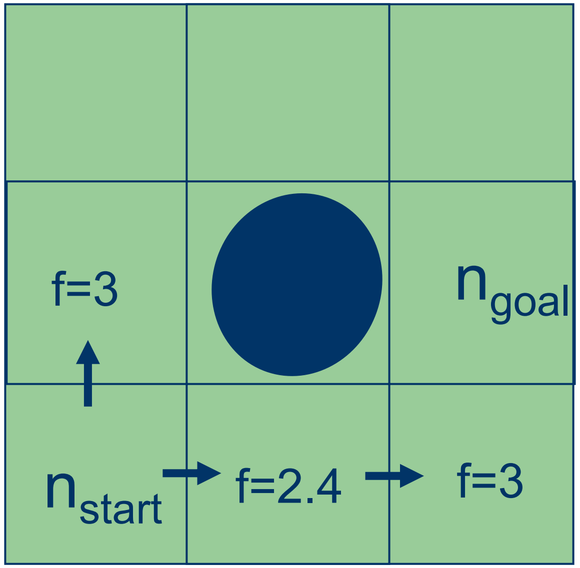
$$\textit{Fringe set} = \{f_1 = 2.4, f_2 = 3\}$$



Motion Planning:
A* Search

- Search example: Iteration 2

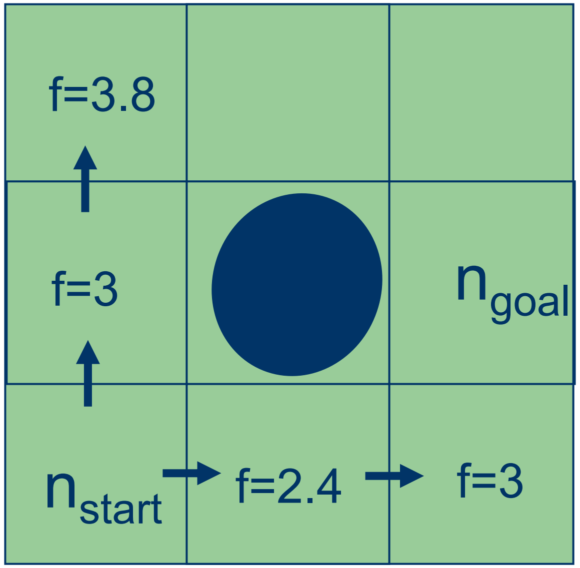
$$\textit{Fringe set} = \{f_2 = 3, f_3 = 3\}$$



Motion Planning:
A* Search

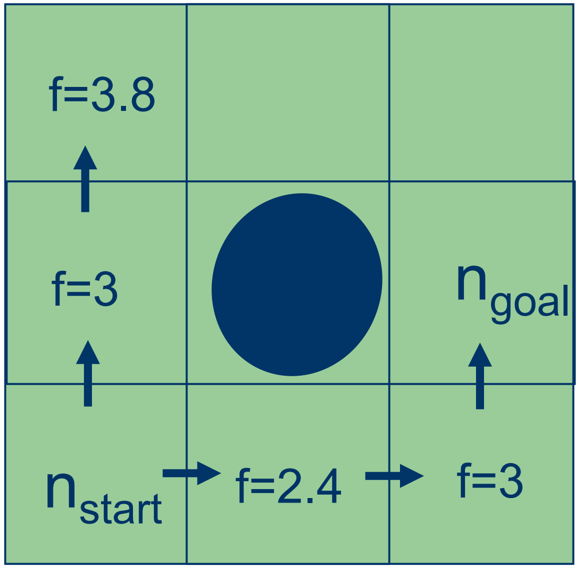
- Search example: Iteration 3

Fringe set $= \{f_3 = 3, f_4 = 3.8\}$



Motion Planning: A* Search

- Search example: Iteration 4



Motion Planning: Final Note

- A robot is often implemented with two planners:
  - Global Planner: A planner that plans an optimal plan with respect to some course discretization of a map.
  - Local Planner: A reactive planner for obstacle avoidance and kinematic consideratations.