COL864: Special Topics in AI

Semester II, 2020-21

Planning Motions

Rohan Paul

Today's lecture

- Last Class
    - Physical Agent Representation
- This Class
    - Sample-based Motion Planning

How to describe a planning problem?

- What is planning?
  - Determining the sequence of actions that will attain a goal state.
- Planning Model
  - Planning problems require a model of the world.
  - Sometimes the term “model” means the parameters of T where the states and actions are fixed.

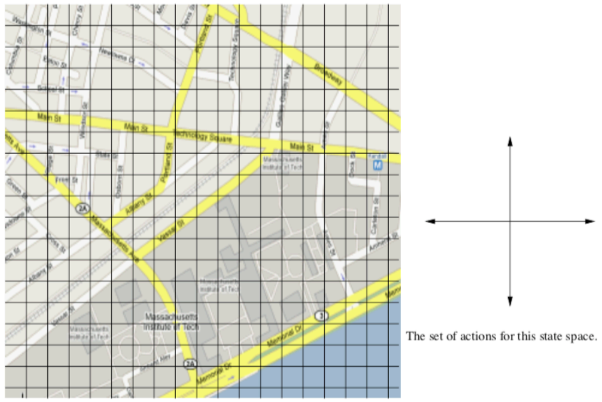
An example of a planning model.

- Set of states $s \in S$ that the world might be in.
- Set of actions $a \in A$ that we might take.
- Transition function $T : S \times A \mapsto S$

How the current state of the world will change as a result of taking an action.

Search applied to a planning model

- Note
  - There are only search algorithms, not planning algorithms.
- Various search algorithms
  - no cost function,
  - have a cost function but no heuristic,
  - cost function and heuristics etc.



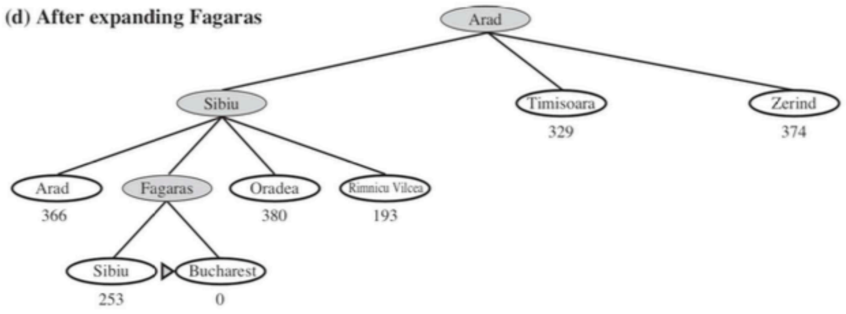


Figure: The A* Tree from Russell & Norvig.

Planning Notions

- Planning problem
  - Tuple: start state s_0, goal state s^g and a planning model M.
- Satisficing (feasibility)
  - Any plan that gets from the start state to the goal state.
- Optimality
  - Comparing multiple plans from the start to the goal.
  - Notion of costs C or rewards R (associated with taking an action a in a state s and reaching state s')
  - Optimal plans minimize the total cost or maximize the total reward.
- Completeness
  - If an optimal plan exists (connecting the start to the goal state) then the search algorithm will find it.

$$\{s_0, s^g, M\}$$

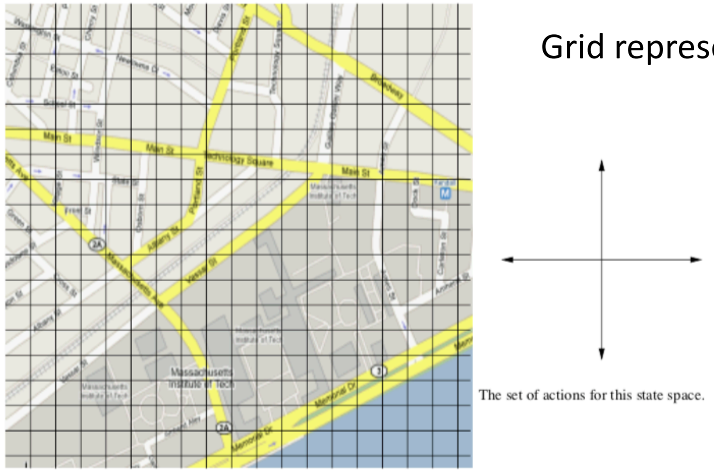
If a plan τ is specified as a sequence of states and actions $\{s_0^\tau, a_0^\tau, \ldots, a_T^\tau, s_{T+1}^\tau\}$, then the best plan might be

$$\tau^* = \operatorname*{argmax}_\tau \sum_t R(s_t^\tau, a_t^\tau, s_{t+1}^\tau)$$

$$\text{such that } s_0^\tau = s_0 \text{ and } s_{T+1}^\tau = s^g$$

Planning Model

- Constructing an appropriate model is often the most important part of planning.
- Example
  - Different discretizations lead to different models with different state and action spaces, even though the underlying problem hasn't changed.



Grid representation

Topological representation

Plan Execution

- Once we have a plan, it is to be executed.
- There could still be failures in plan execution due to uncertainty.
  - Aleatoric uncertainty
    - The uncertainty in action outcome because the real world is stochastic.
  - Epistemic uncertainty
    - The uncertainty because there is something that we do not know (the topology changes, new things added in the map).

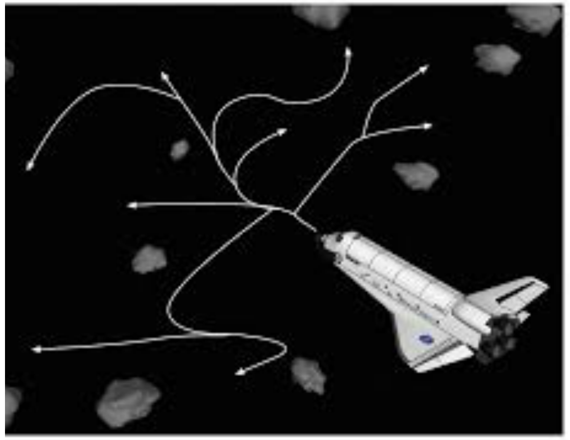
Maps can change over time.

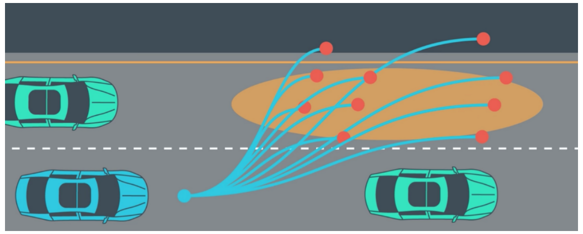
Plan Execution

- Re-planning
  - A form of execution monitoring.
- Action monitoring
  - Is the *current action optimal/feasible* from the current state?
- Plan monitoring
  - Is the *current plan* still *optimal or feasible*?
- Goal monitoring
  - Is the *current goal* still *feasible/desirable*?

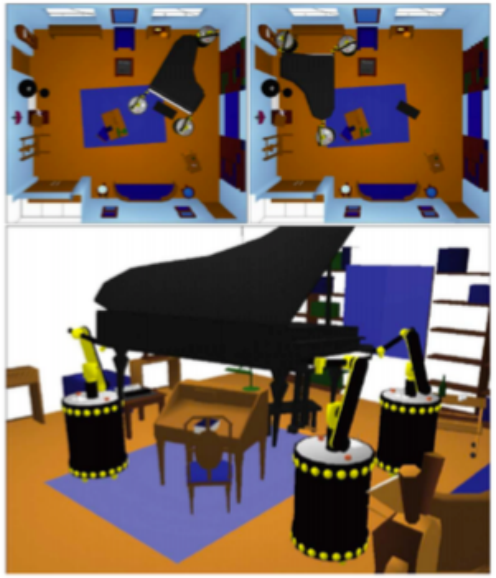
Motion Planning

- **Motion Planning**
  - Given a start and and a goal state
  - Determine a sequence of actions (control inputs) that leads from the start to the goal state.
  - A certain kind of planning
- **Challenge**
  - Need to avoid obstacles.
  - The agent may not be able to move to any coordinate at will.
  - Example: car, helicopter, airplane, but also robot manipulator hitting joint limits

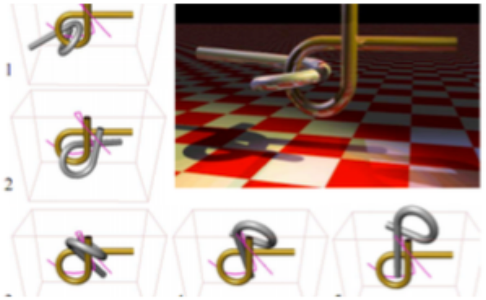




Motion planning for vehicles



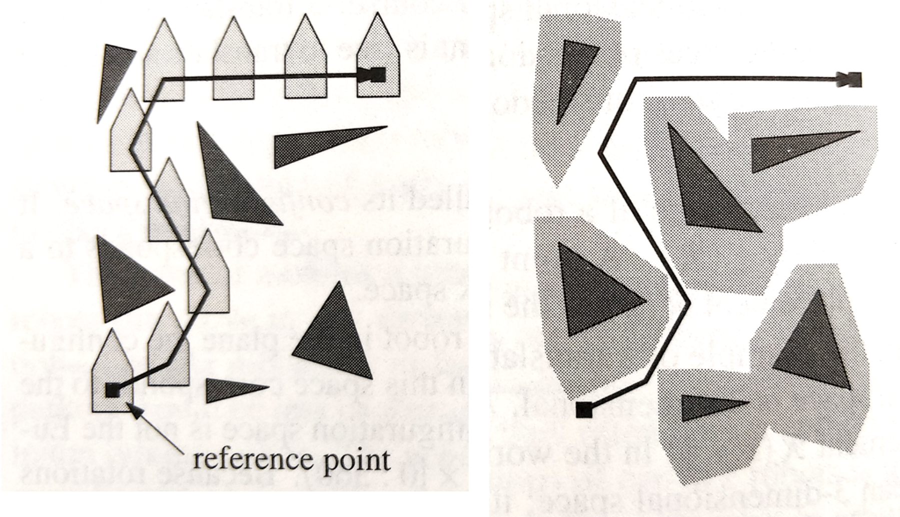
Re-arrangement planning

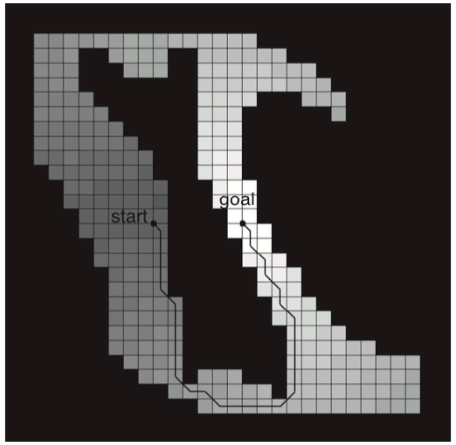


Benchmark problem – pulling bars apart (J. Kuffner)

Planning Motions in Continuous Spaces

- **Task space**
  - Actual cartesian space in which the agent moves.
- **Configuration space**
  - Space of possible configurations of the agent.
- **Continuous spaces**
  - Space in which we are looking for a plan is **continuous**.
  - We do not have access to a **grid** or a graph a-priori.
  - Naïve discretization leads to a prohibitively large set of states when the configuration space is high-dimensional.





Sample-based Motion Planning

- **Monte Carlo methods**
  - Tackling continuous spaces (or when the space is too large), cannot exhaustively explore all possibilities.
  - Randomly explore a smaller subset of possibilities while keeping track of progress.

- **Tradeoff**
  - Between solution quality and runtime performance.
  - More sampling leads to a better solution.

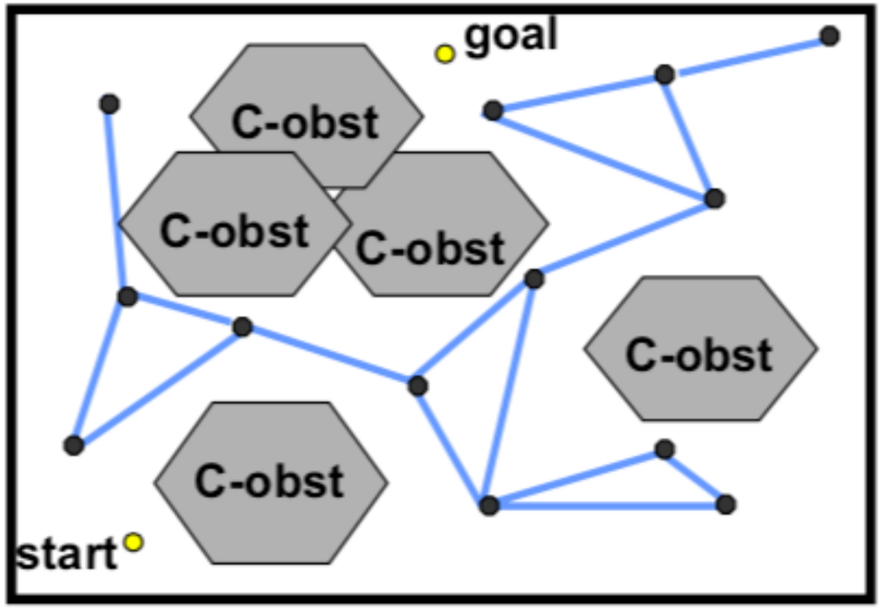
- **Sample-based Planning**
  - Search for a path (collision-free) only by sampling paths.
  - Probabilistic Roadmaps (PRMs)
  - Rapidly exploring random trees (RRTs)

RRT sampling based motion planner.

Probabilistic Roadmaps

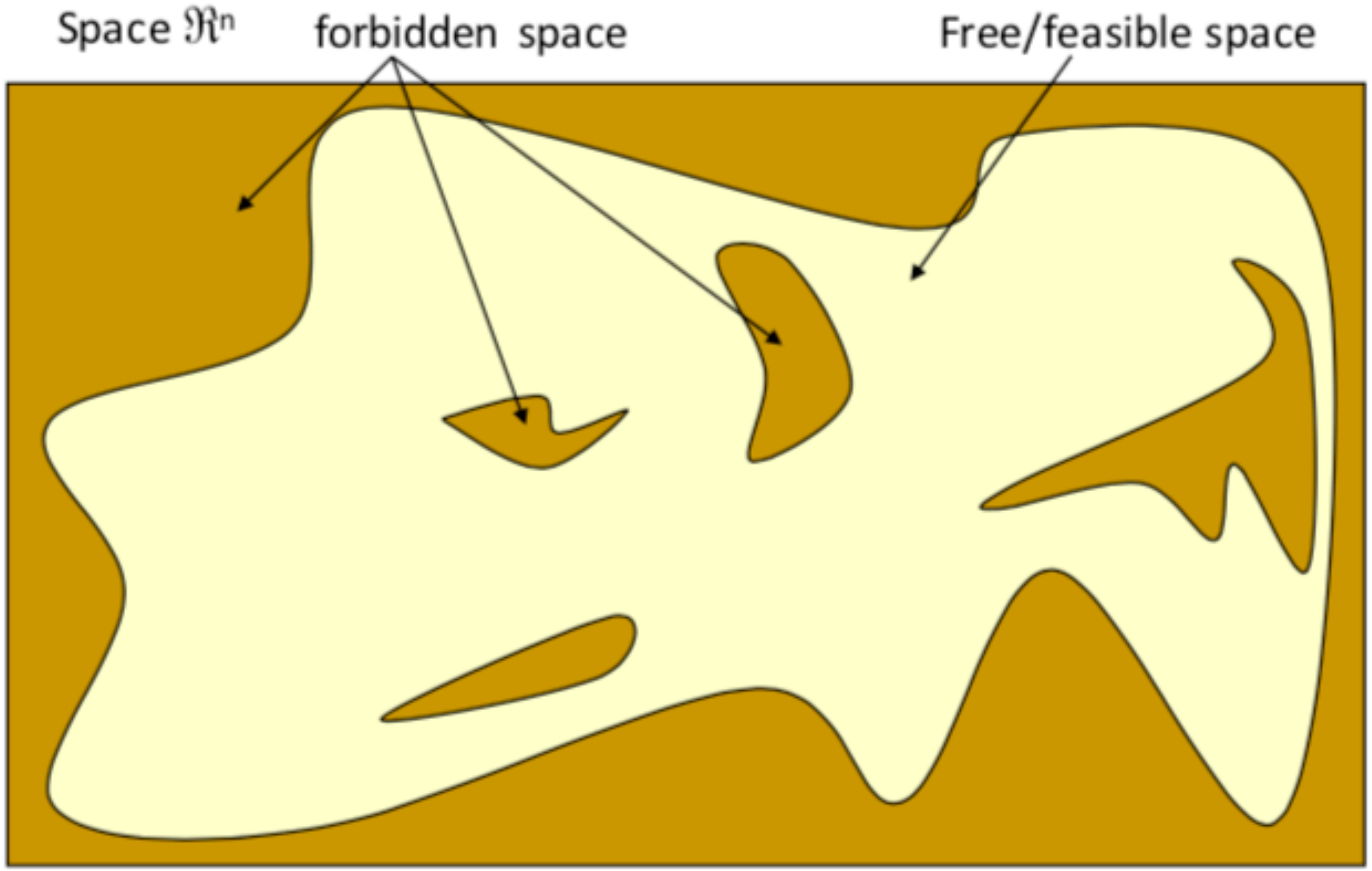
- **Central Idea**
  - Sample and find collision-free configurations
  - Connect the configurations (create a graph)
  - Search the graph for solution

- **Phases**
  - Learning phase
  - Query phase. Inherently, a multi-query planner.



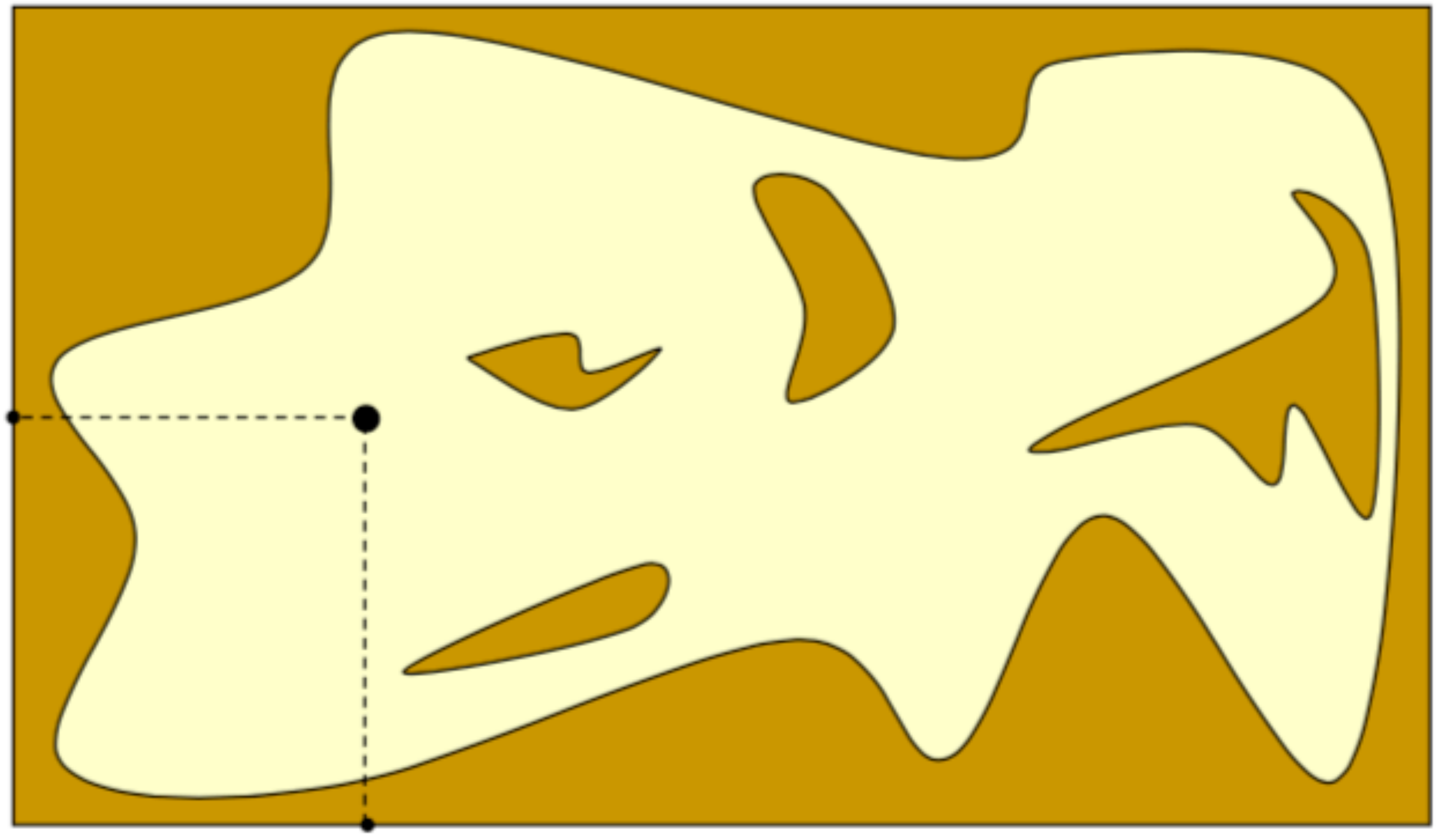
Probabilistic Roadmap (PRM)

- Configuration space
  - Forbidden and free space



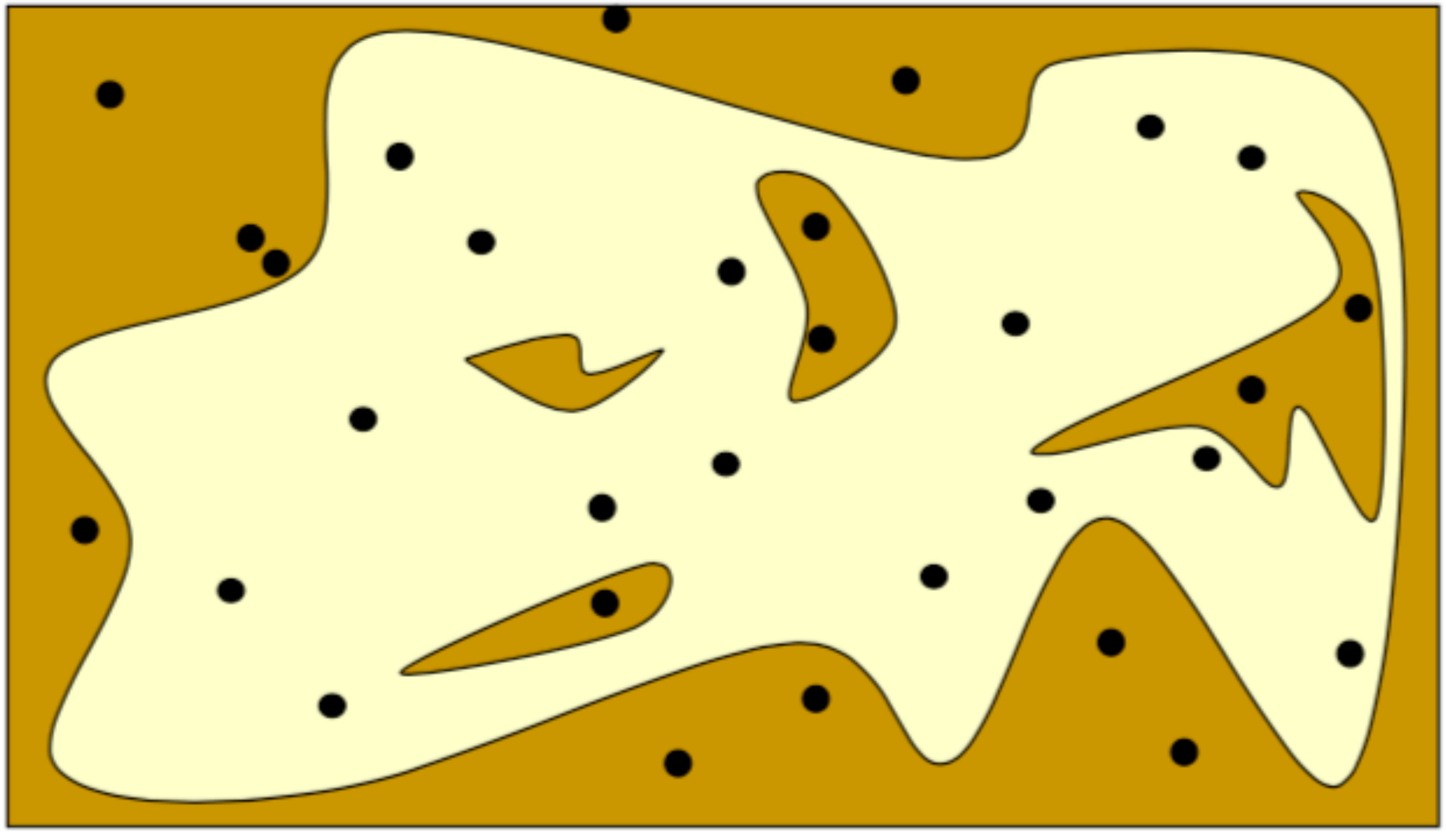
Probabilistic Roadmap (PRM)

- Configurations are sampled by picking coordinates at random.



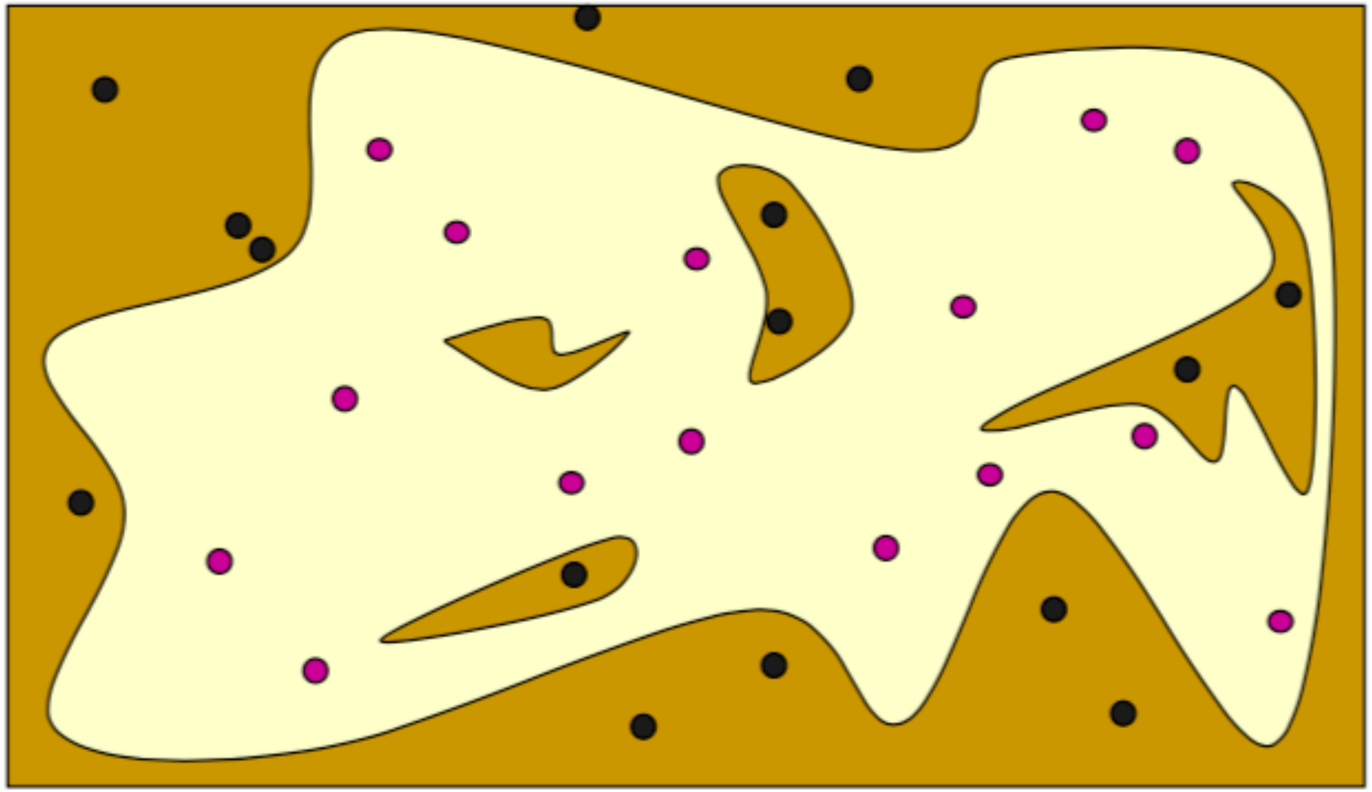
Probabilistic Roadmap (PRM)

- Configurations are sampled by picking coordinates at random.



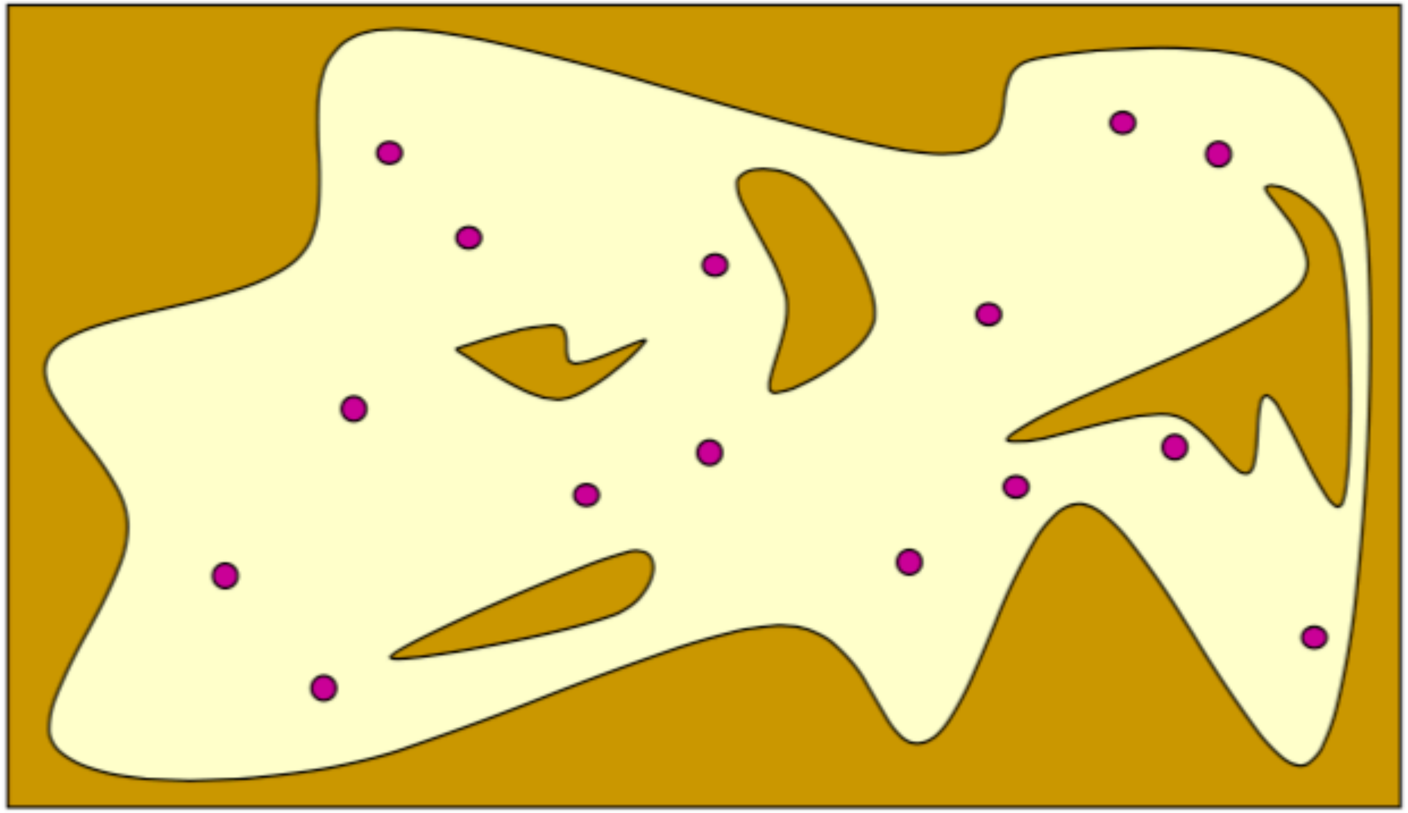
Probabilistic Roadmap (PRM)

- Sampled configurations are tested for collisions.



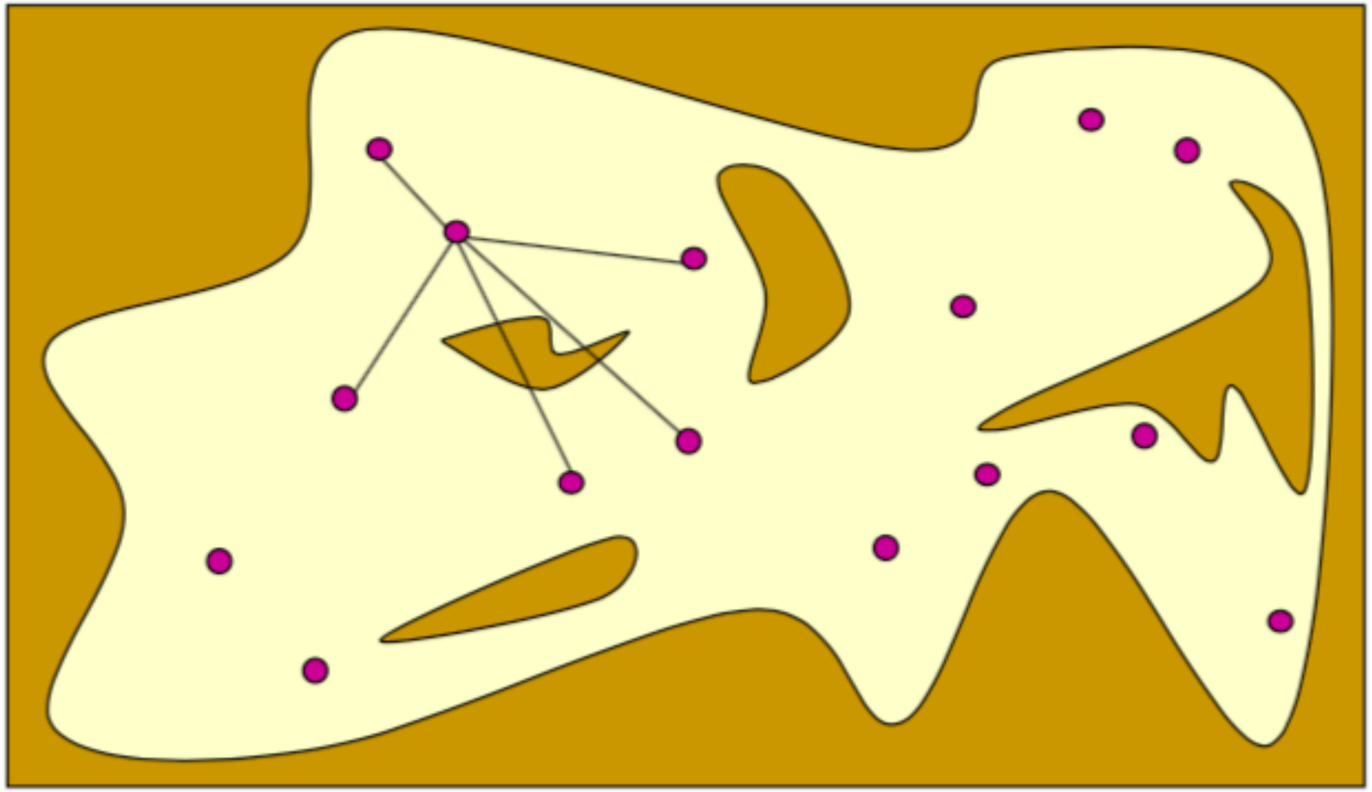
Probabilistic Roadmap (PRM)

- Feasible configurations (collision free) are retained as “milestones”.



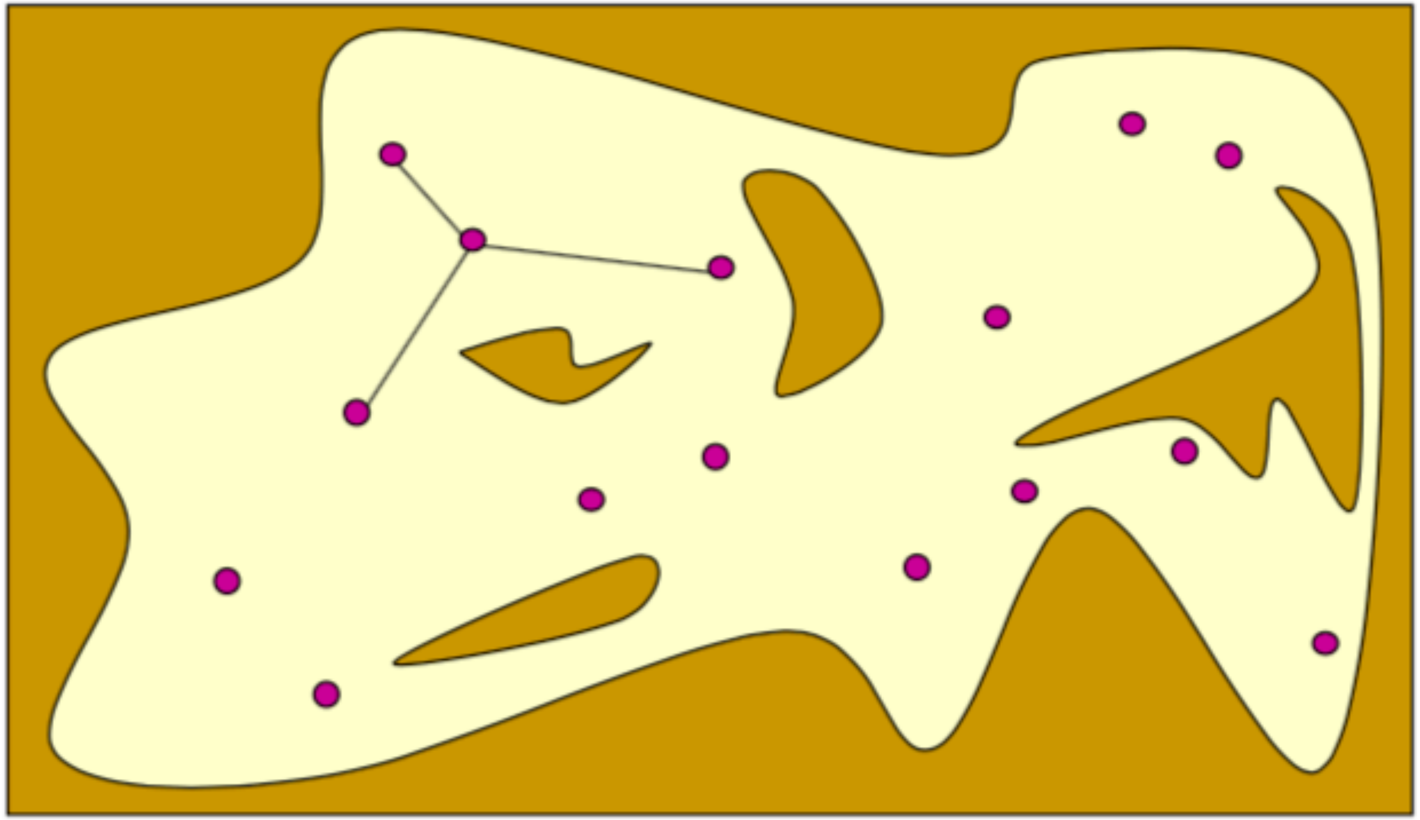
Probabilistic Roadmap (PRM)

- Each milestone is linked by straight paths to its nearest neighbours.



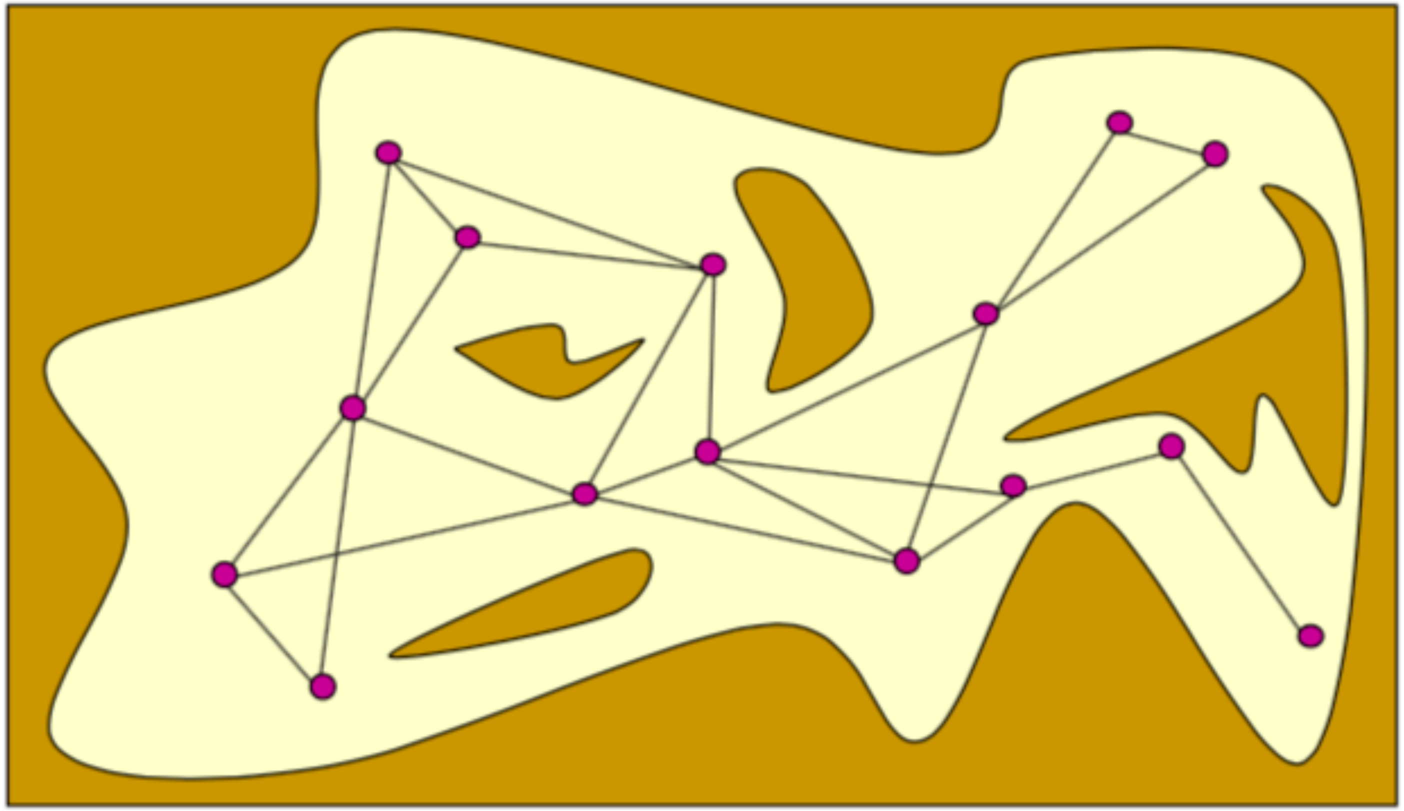
Probabilistic Roadmap (PRM)

- Each milestone is linked by straight paths to its nearest neighbours.
- Retain only feasible connections.



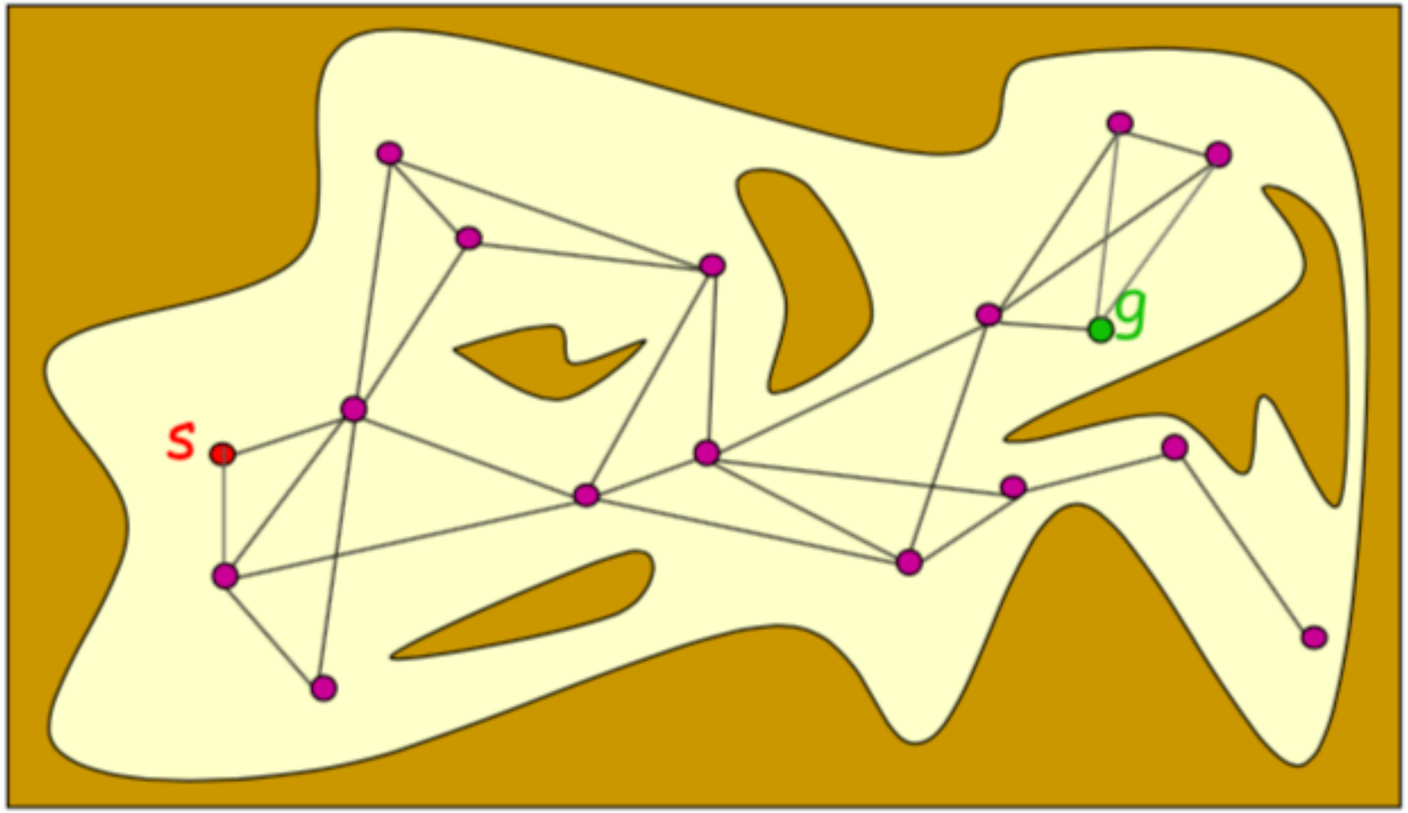
Probabilistic Roadmap (PRM)

- The collision-free links are retained as local paths. This is the output of the learning phase.
- The graph structure is the roadmap. The sampling is probabilistic, hence called PRM.



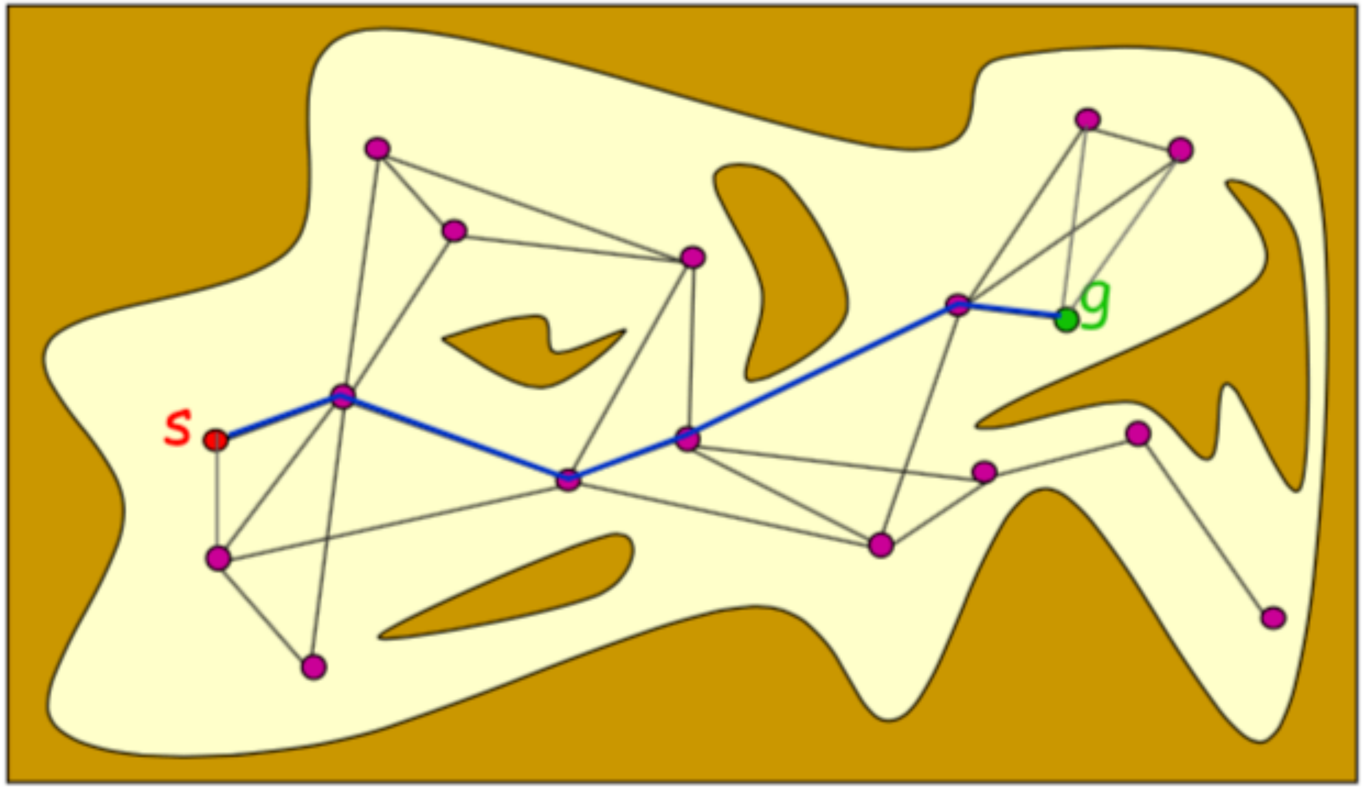
Probabilistic Roadmap: Query Phase

- Query phase.
- The start and the goal configurations are included as milestones.



Probabilistic Roadmap: Query Phase

- The graph is searched for a path from s to g.

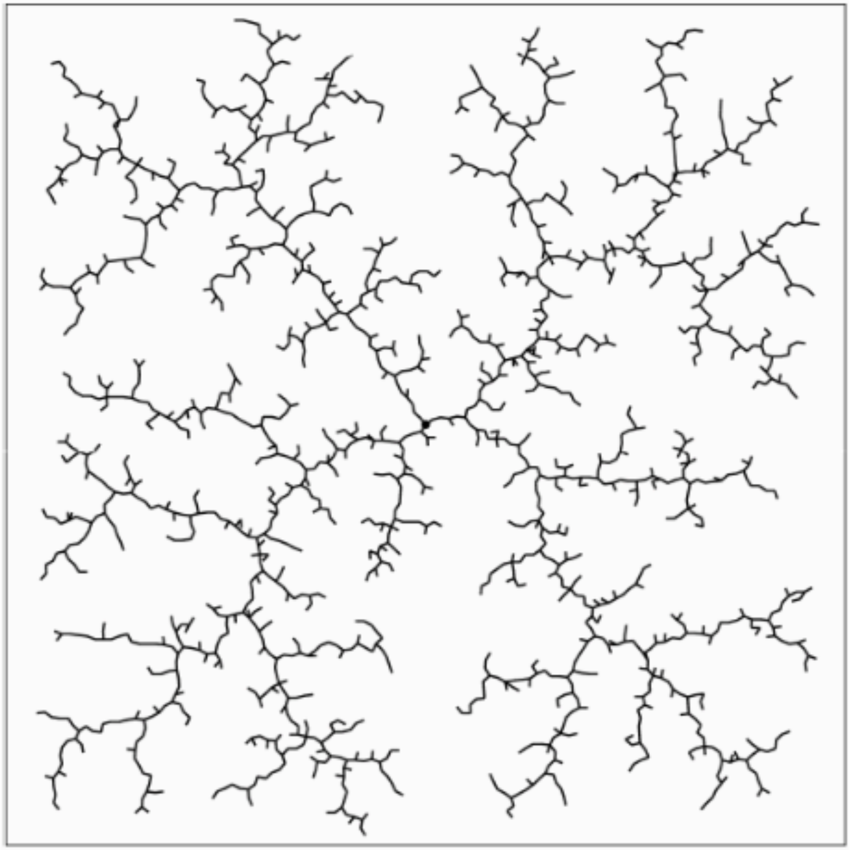


Probabilistic Roadmap

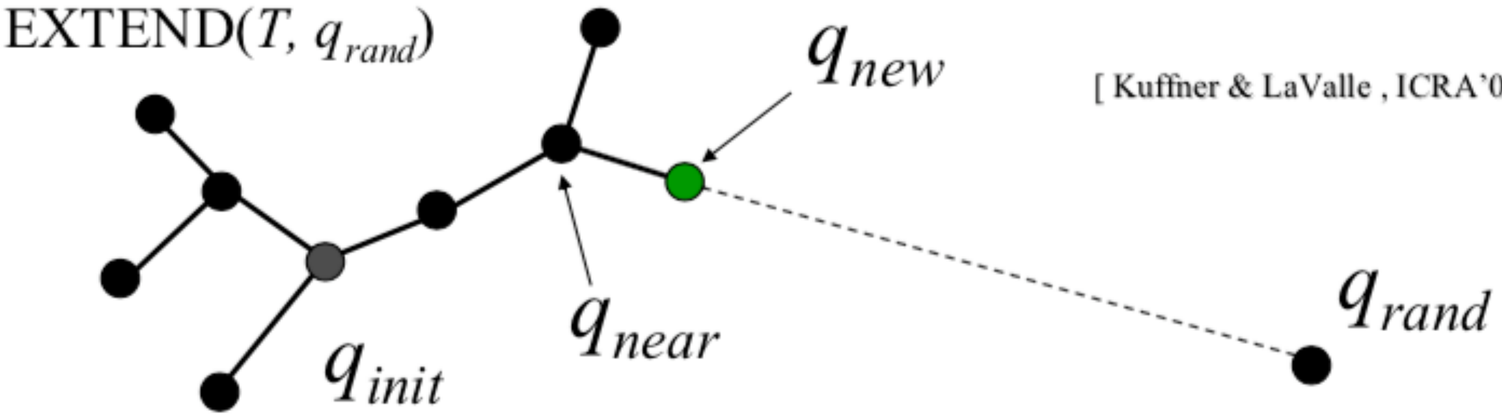
- **Key Steps**
  - Initialize set of points with X_S and X_G
  - Randomly sample points in the configuration space
  - Connect nearby points if they can be reached from each other
  - Find a path from X_S to X_G in the graph.
- **Probabilistically completeness**
  - If the algorithm is run infinitely often then by probability one it will contain a solution path if one exists.

Rapidly Exploring Random Tree (RRT)

- **Central Idea**
  - Build up a tree in the search space through generating “next states”.
  - Select a random point and expand the nearest vertex in the tree towards the sampled point.

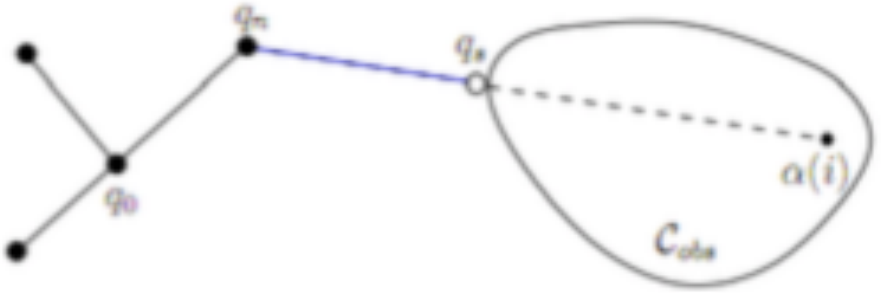


RRT Extension



[Kuffner & LaValle , ICRA'00]

In the presence of an obstacle.



Rapidly Exploring Random Tree (RRT)

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});

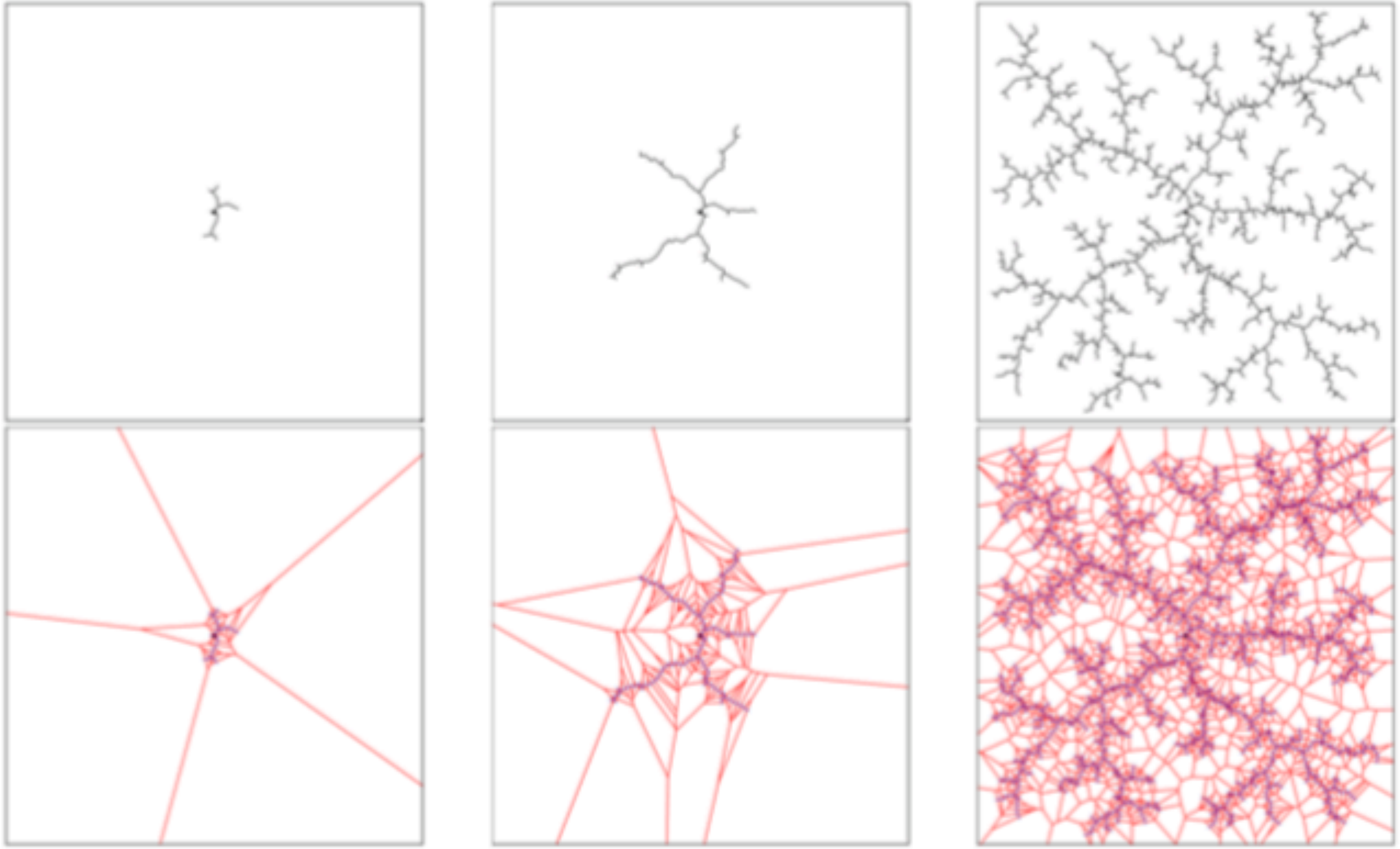
  $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);

  $\mathcal{T}$.add_vertex(x_{new});

  $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);

Return $\mathcal{T}$

Rapidly Exploring Random Tree (RRT)



http://msl.cs.uiuc.edu/rrt/gallery.html

Biases

- Biases towards larger spaces.
- Creating a bias towards the goal.
  - When generating a random sample, with some probability pick the goal instead of a random node when expanding.

RANDOM_STATE(): often uniformly at random over space with probability 99%, and the goal state with probability 1%, this ensures it attempts to connect to goal semi-regularly

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 $\mathcal{T}$

Growing an RRT

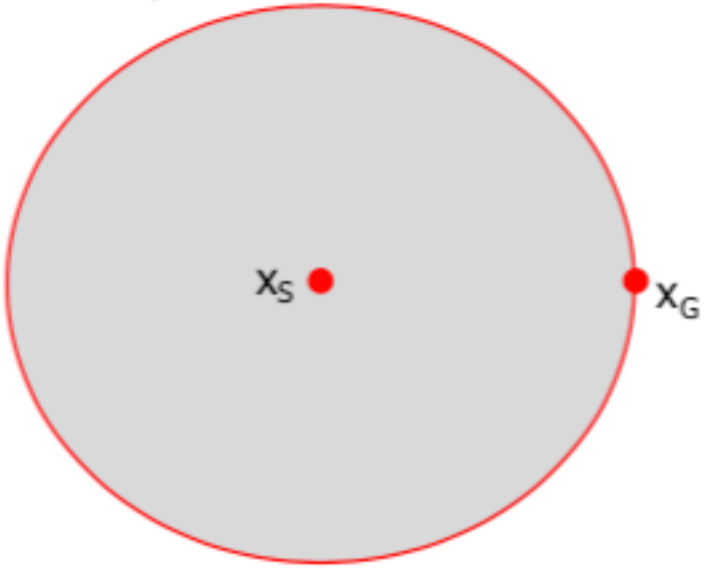
A visualization of an RRT graph after 45 and 390 iterations

An animation of an RRT starting from iteration 0 to 10000

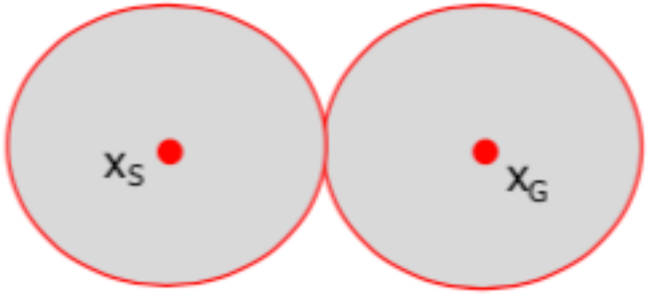
https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree

Bi-directional RRT

- Volume swept out by unidirectional RRT:

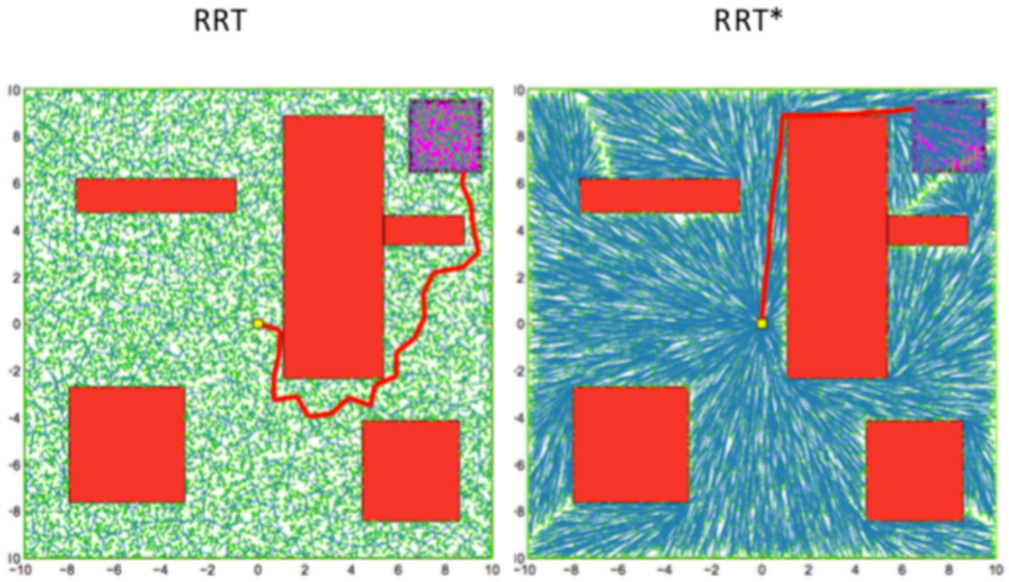


- Volume swept out by bi-directional RRT:



RRT*

- Asymptotically optimal
  - In the limit, will find the optimal path.
  - Performs re-wiring of the tree.
- Karaman and Frazolli
  - https://dspace.mit.edu/handle/1721.1/63170
  - https://www.youtube.com/watch?v=6Pngam882hM
  - https://www.youtube.com/watch?v=2WOBMswcCA8



RRT Applications

- Robotics Applications
  - mobile robotics
  - manipulation
  - humanoids
- Other Applications
  - biology (drug design)
  - manufacturing and virtual prototyping (assembly analysis)
  - verification and validation
  - computer animation and real-time graphics
  - aerospace
- RRT extensions
  - discrete planning (STRIPS and Rubik's cube)
  - real-time RRTs
  - anytime RRTs
  - dynamic domain RRTs
  - deterministic RRTs
  - parallel RRTs
  - hybrid RRTs

Other approaches: Potential Fields

At any point $\mathbf{x}$ we can write the total potential $\mathbf{U}_\Sigma$ as a sum of the potential induced $\mathbf{U}_o$ by k obstacles and the potential induced by the goal $\mathbf{U}_g$:

$$\mathbf{U}_\Sigma(\mathbf{x}) = \sum_{i=1:k} \mathbf{U}_{o,i}(\mathbf{x}) + \mathbf{U}_g(\mathbf{x}) \tag{2.7}$$

Now we know that the force $\mathbf{F}(\mathbf{x})$ exerted on a particle in a potential field $\mathbf{U}_\Sigma(\mathbf{x})$ can be written as :

$$\mathbf{F}(\mathbf{x}) = -\nabla\mathbf{U}_\Sigma(\mathbf{x}) \tag{2.8}$$

$$= -\sum_{i=1:k} \nabla\mathbf{U}_{o,i}(\mathbf{x}) - \nabla\mathbf{U}_g(\mathbf{x}) \tag{2.9}$$

Other approaches: Potential Fields

$\rho(\mathbf{x})$ Shortest distance between the obstacle and vehicle at x.

ρ_0 limit on the region of space affected by the potential field

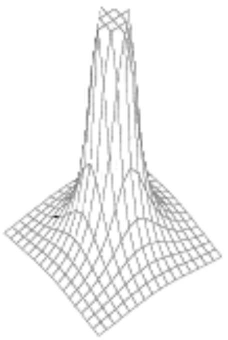
$$\mathbf{U}_{o,i}(\mathbf{x}) = \eta \begin{cases} \frac{1}{2}\left(\frac{1}{\rho(\mathbf{x})} - \frac{1}{\rho_0}\right)^2 & \forall \quad \rho(\mathbf{x}) \leq \rho_0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{U}_g(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_g)^2$$

$$\mathbf{F}_{o,i} = \begin{cases} \eta\left(\frac{1}{\rho(\mathbf{x})} - \frac{1}{\rho_0}\right)\frac{1}{\rho(\mathbf{x})^2}\frac{\partial\rho(\mathbf{x})}{\partial\mathbf{x}} & \forall\rho(\mathbf{x}) \leq \rho_0 \\ 0 & \text{otherwise} \end{cases}$$

Two typical potential functions - inverse quadratic for obstacle and quadratic for the goal.

point obstacle potential



goal potential

Limitation: Local Minima

Only acts locally. There is no global path planning.

The vehicle simply reacts to local obstacles, always moving in a direction of decreasing potential.

Example: the vehicle will descend into a local minima in front of the two features and will stop. Any other motion will increase its potential.