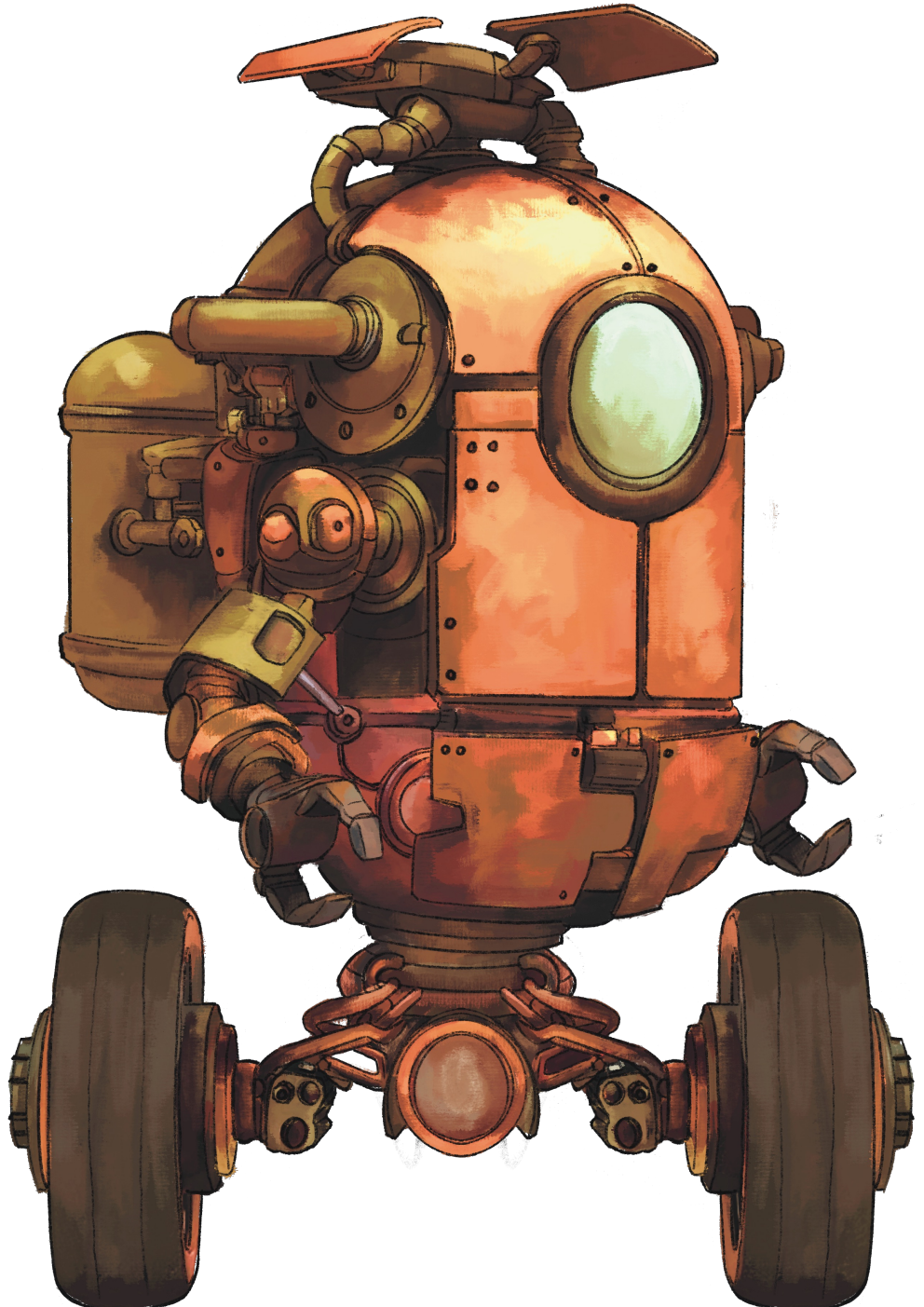
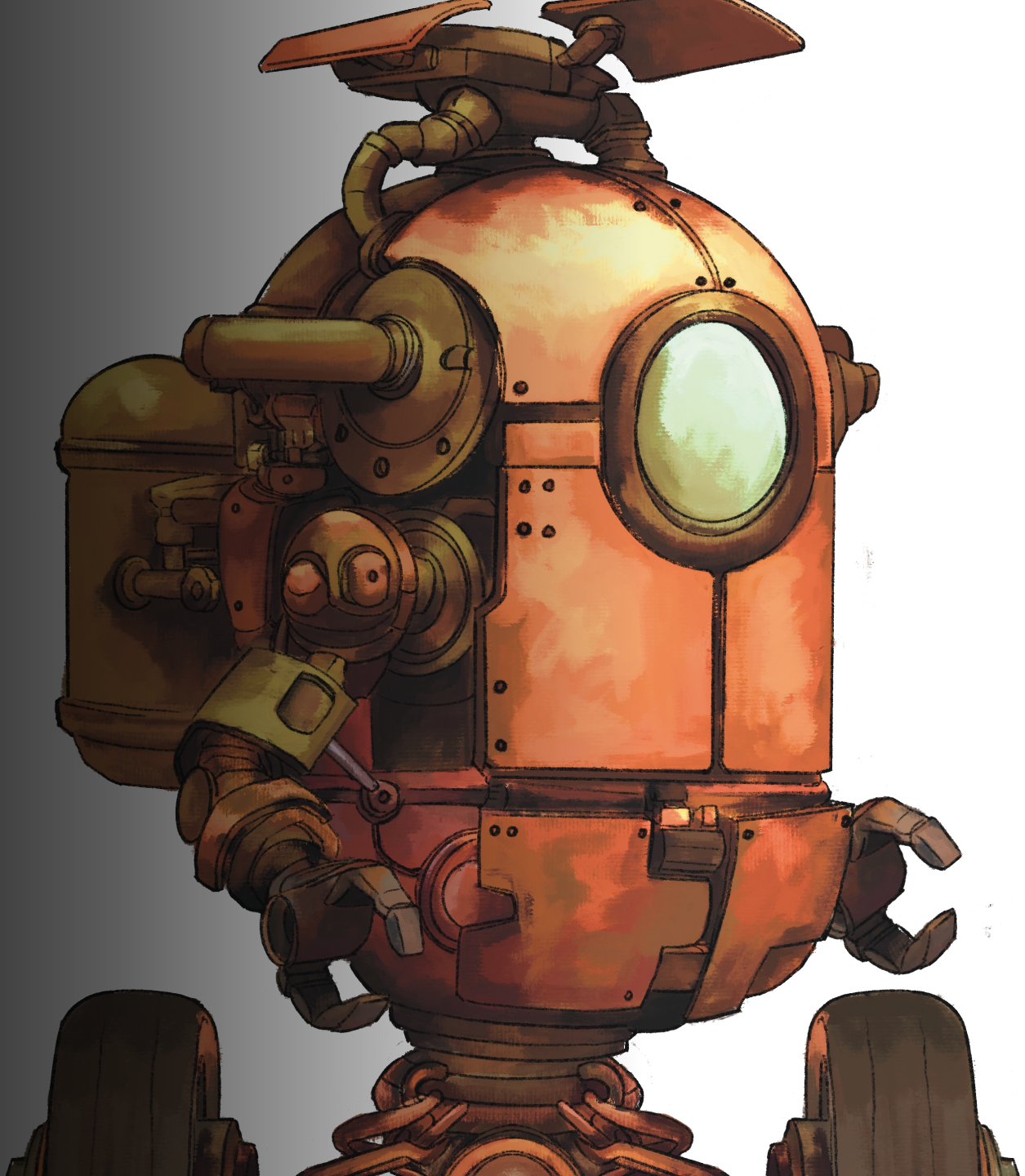
CS 3630, Fall 2026

Lecture 20:
Differential Drive Robots:
Motion Planning



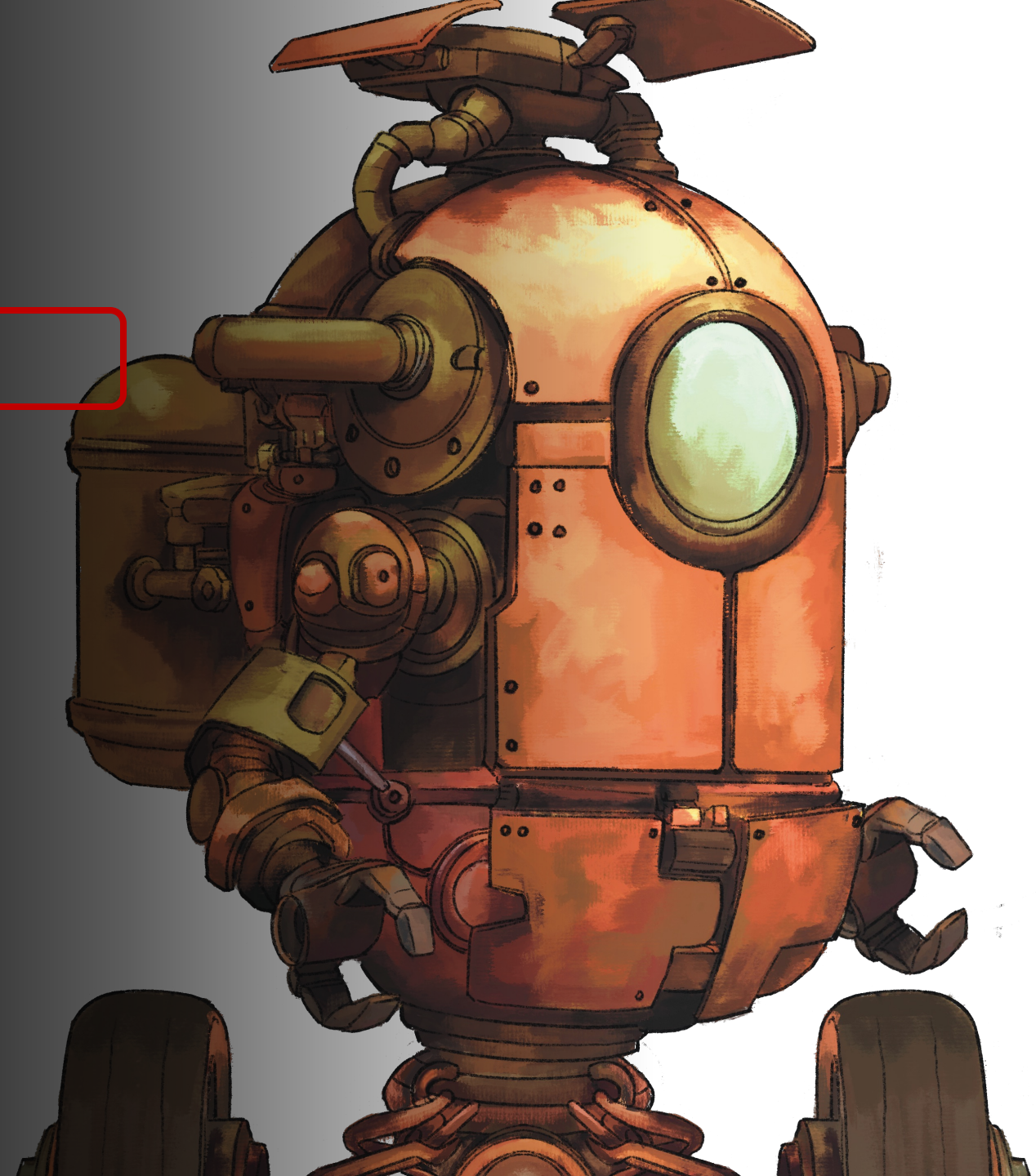
Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants



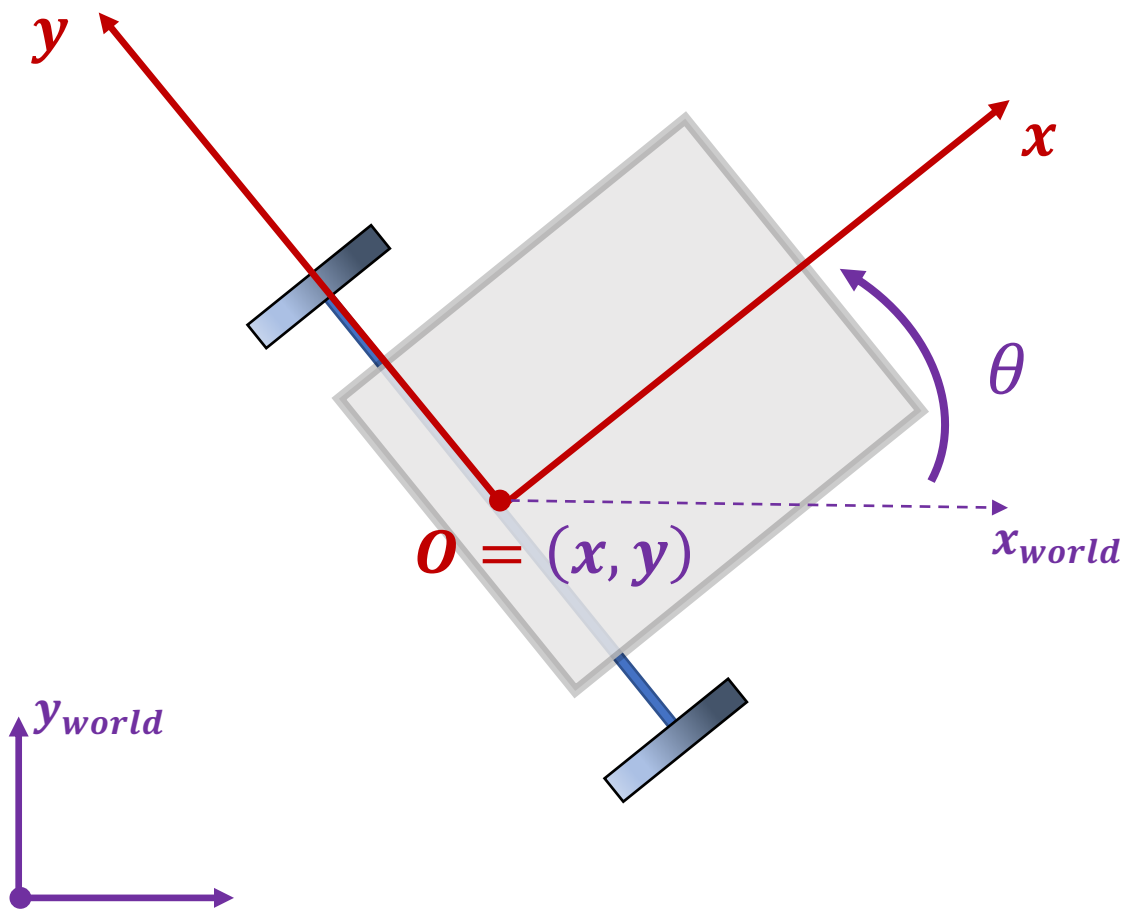
Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants



Configuration Space

- A ***configuration*** is a complete specification of the position of every point in a robot system.
- The ***configuration space*** is the set of all configurations.
- We use q to denote a point in a configuration space $\mathcal{Q}$.



Because our DDR can rotate in the plane, it is necessary to know both the position and the orientation of the body-attached frame to specify a configuration:

$$\mathcal{Q} = \mathbb{R}^2 \times [0, 2\pi)$$

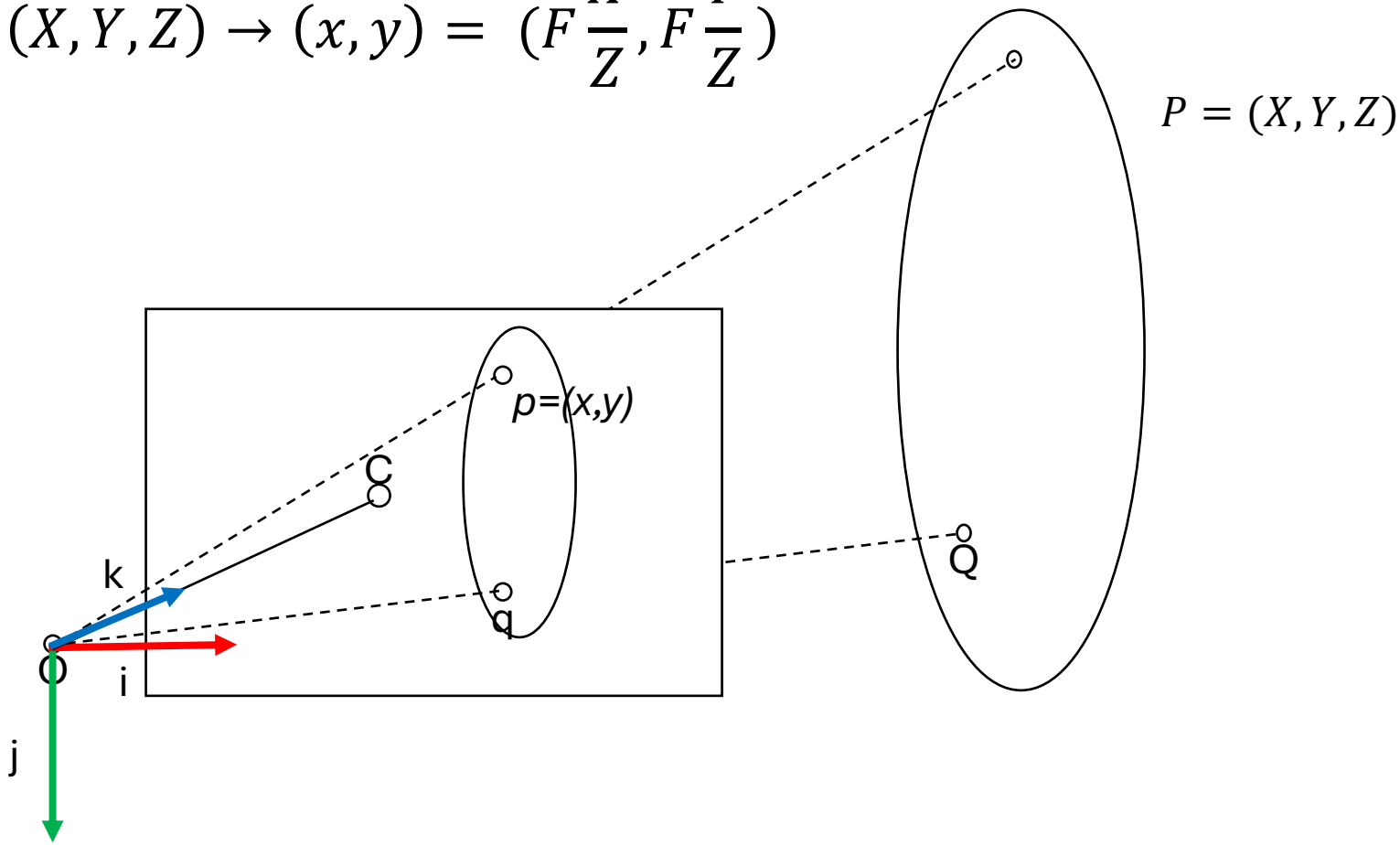
$$q = (x, y, \theta) \in \mathcal{Q}$$

If we know the configuration, $q = (x, y, \theta)$, we can compute the location of any point on the robot.

Pinhole Cameras

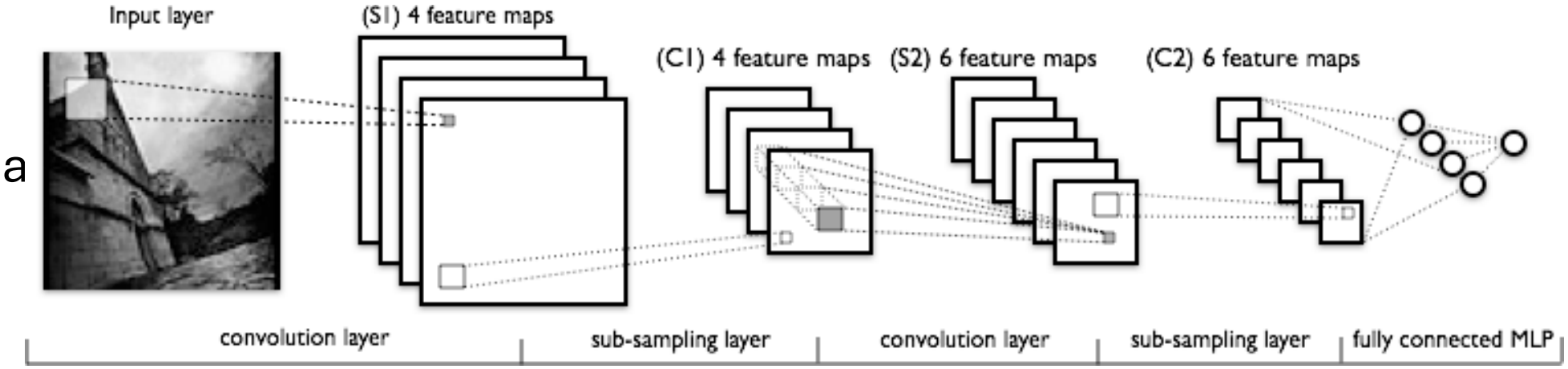
- Fundamental equation:

$$(X, Y, Z) \rightarrow (x, y) = \left(F\frac{X}{Z}, F\frac{Y}{Z}\right)$$



Convolutional Neural Networks

a



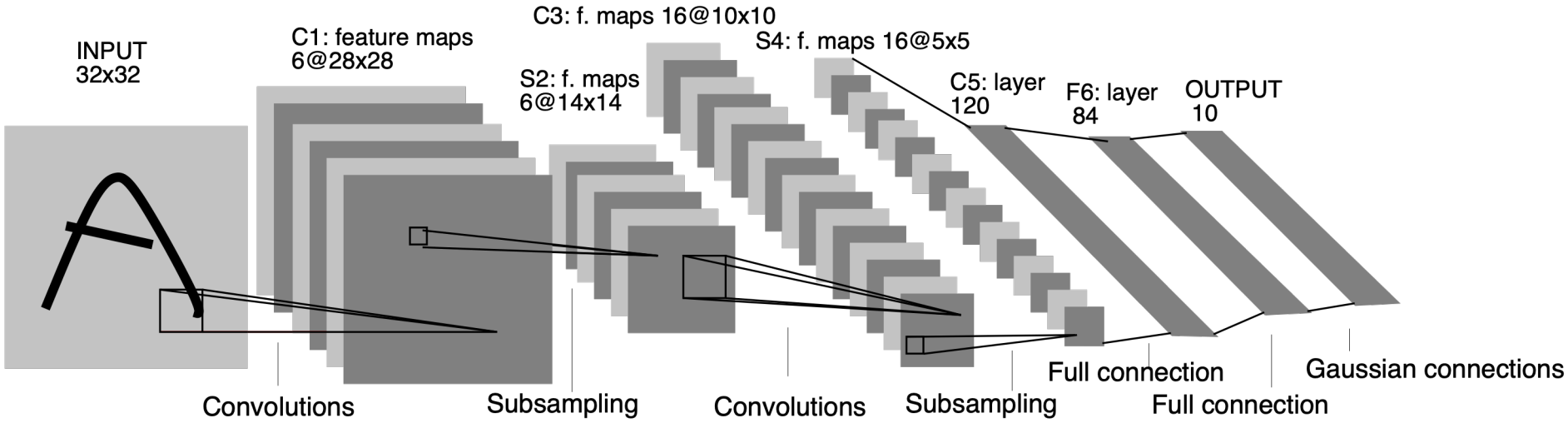
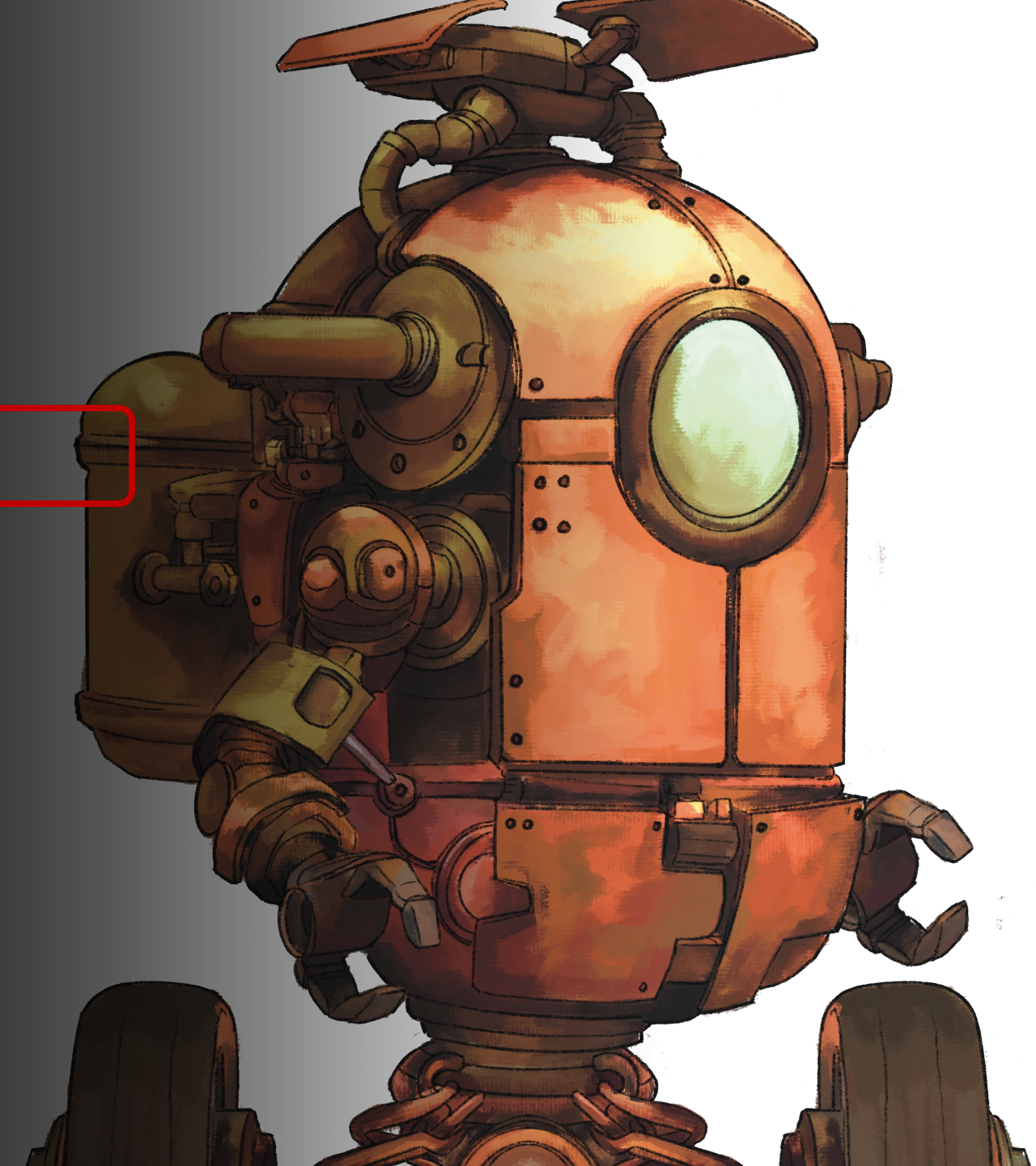


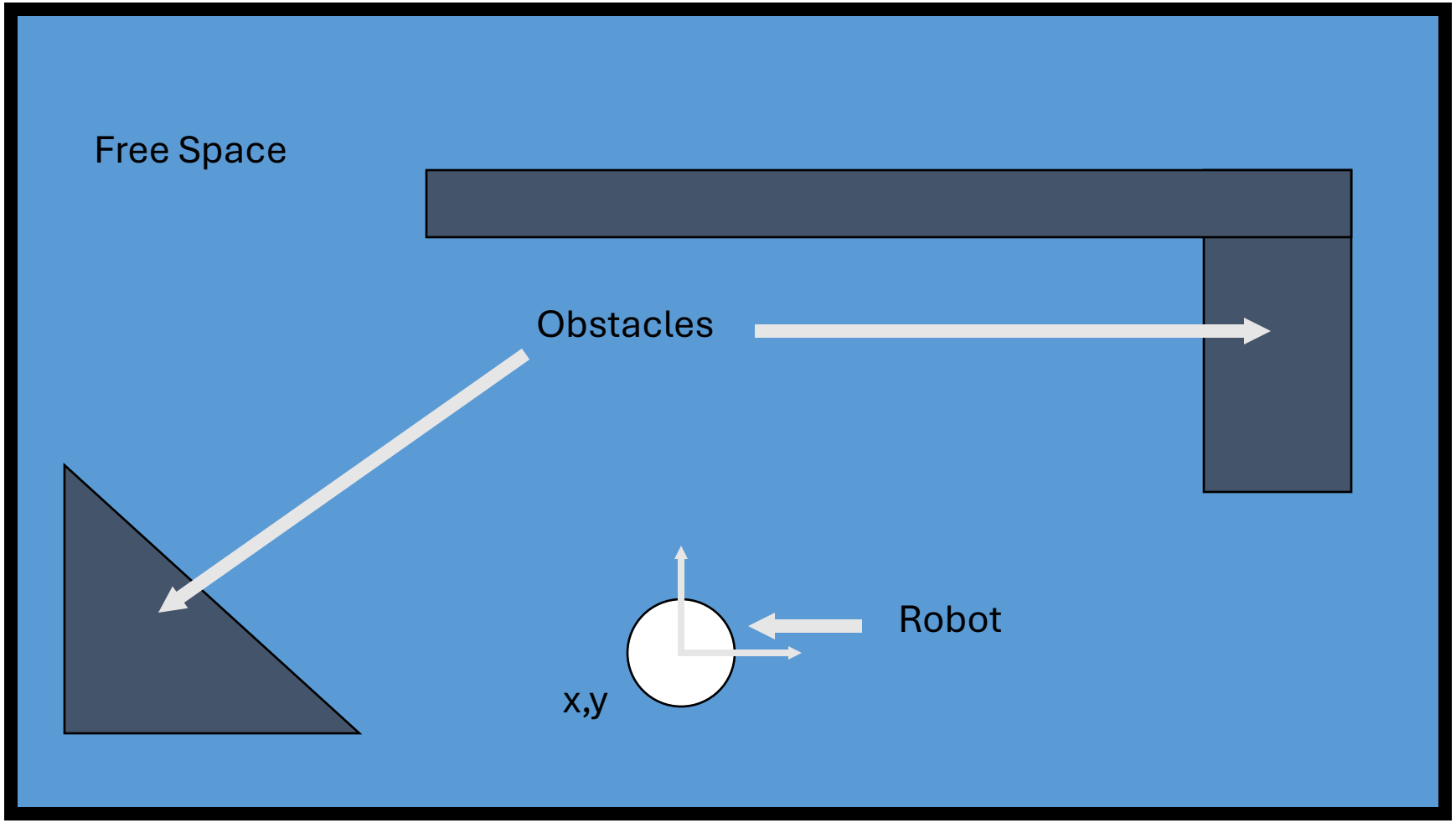
Image Credit: Yann LeCun, Kevin Murphy

Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants

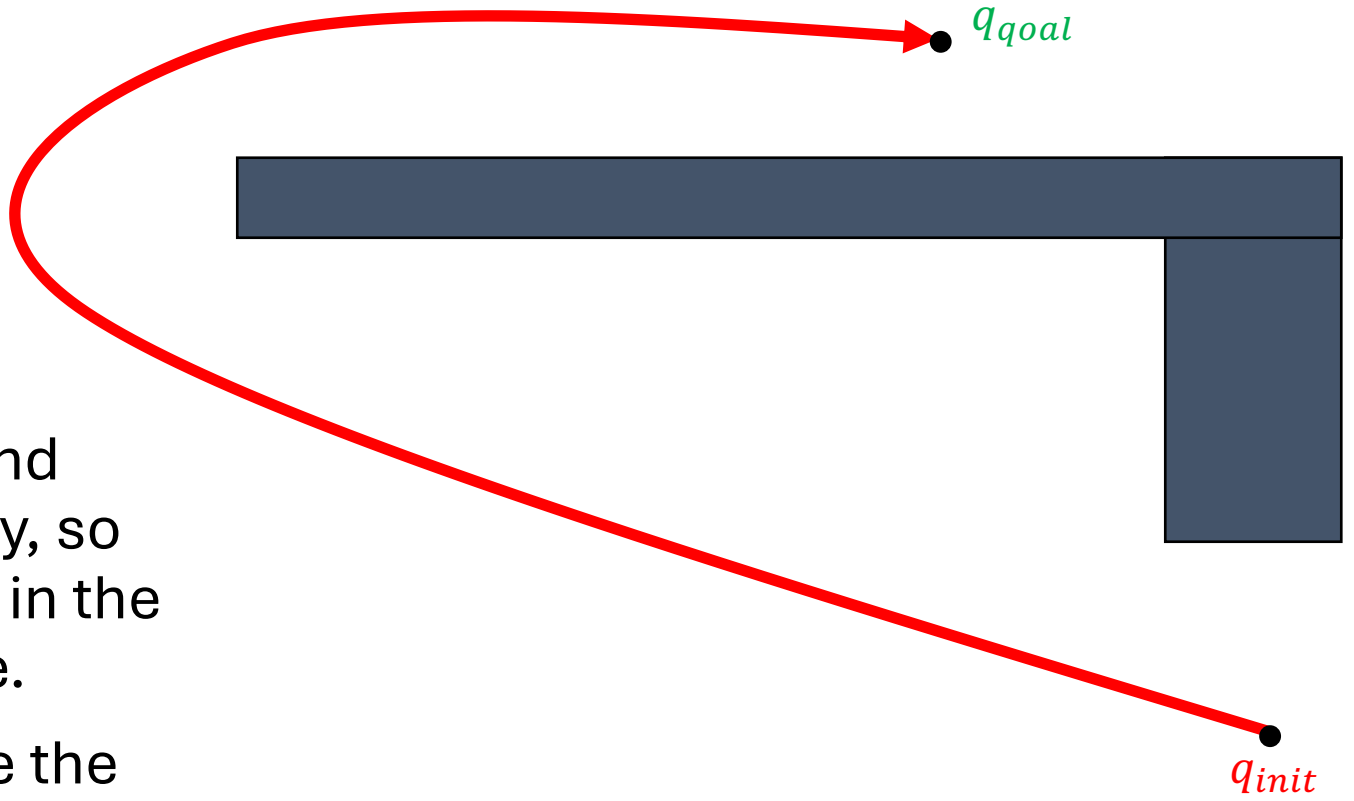


Example of a World (and Robot)



Path Planning

- Find a collision-free path from the
  starting configuration q_{init}
  to a
  goal configuration q_{goal}.
- Collision checking between the robot and obstacles can be computationally heavy, so we deal with this by mapping obstacles in the world to the robot's configuration space.
- In the configuration space, we now have the problem of finding a path for a single point (which represents the configuration of the robot).
- In general, computing this mapping can be difficult computationally, but it's not so bad for robots that translate in the plane.



Obstacles in C-Space

- Let q denote a point in a configuration space $\mathcal{Q}$
- The path planning problem is to find a mapping γ: $[0,1] \rightarrow \mathcal{Q}$ s.t. no configuration along the path intersects an obstacle.
- Denote the i-th workspace obstacle by $\mathcal{O}_i$, *and by* $R(q)$ *the volume occupied by the robot at configuration* q.
- A *configuration space obstacle* $\mathcal{QO}_i$ is the set of configurations q at which the robot intersects $\mathcal{O}_i$

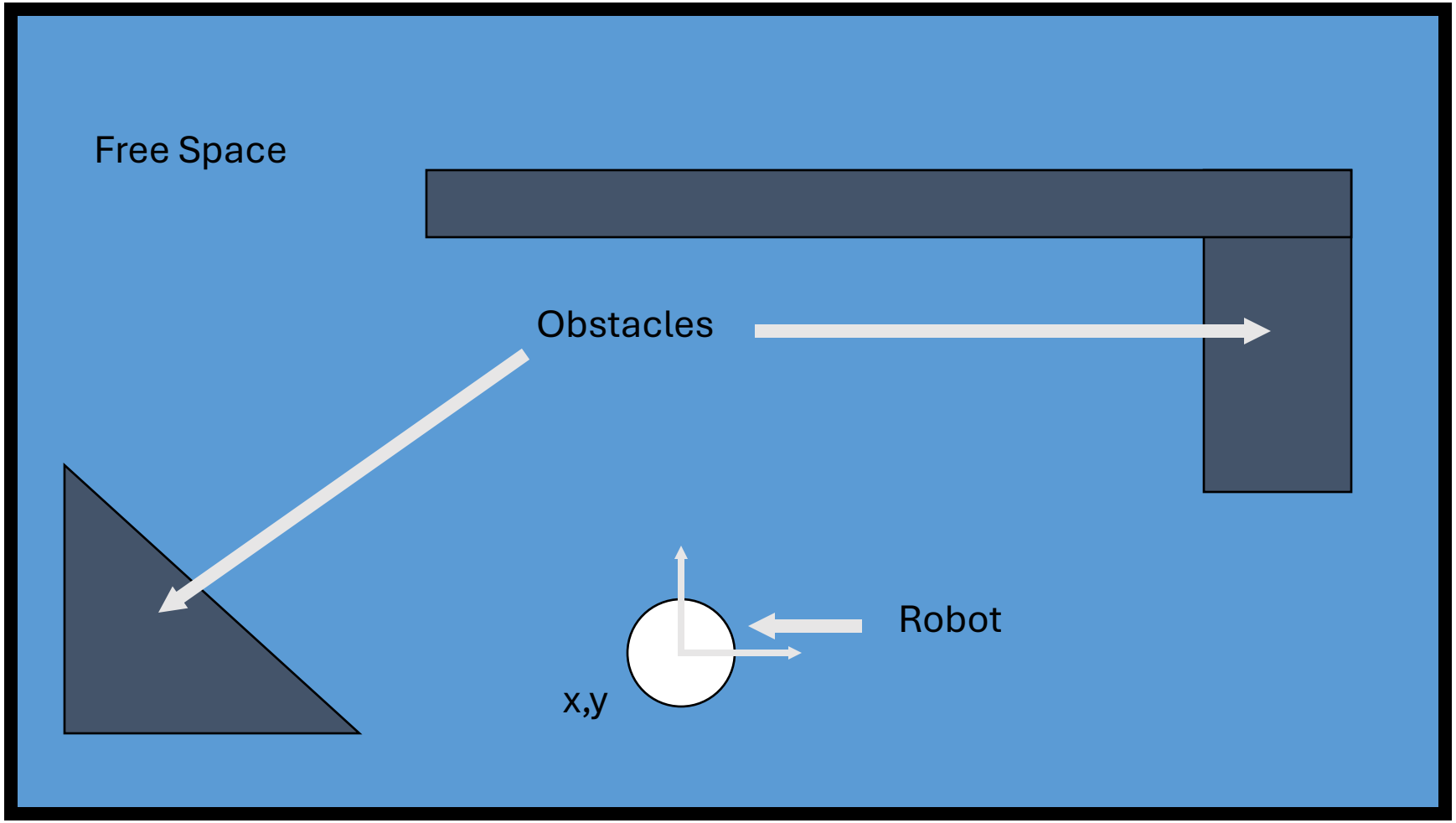
$$\mathcal{QO}_i = \{\, q \in \mathcal{Q} \mid R(q) \cap \mathcal{O}_i \neq \emptyset \}$$

- The *free configuration space* (or just *free space*) $\mathcal{Q}_{free}$ *is*

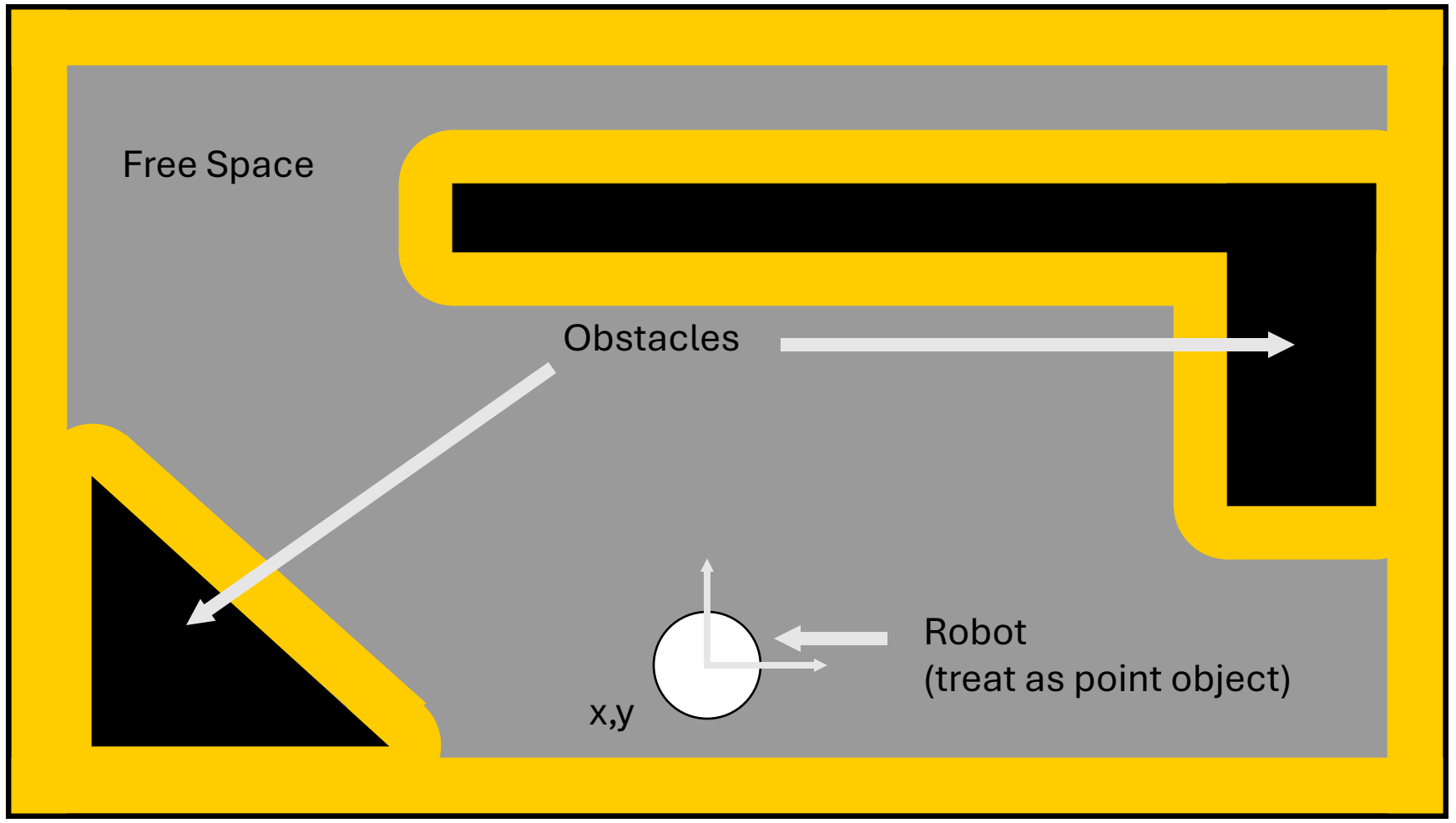
$$\mathcal{Q}_{free} = \mathcal{Q} - \cup_i \mathcal{QO}_i$$

- A *free path* is a mapping γ: $[0,1] \rightarrow \mathcal{Q}_{free}$.

Example of a World (and Robot)



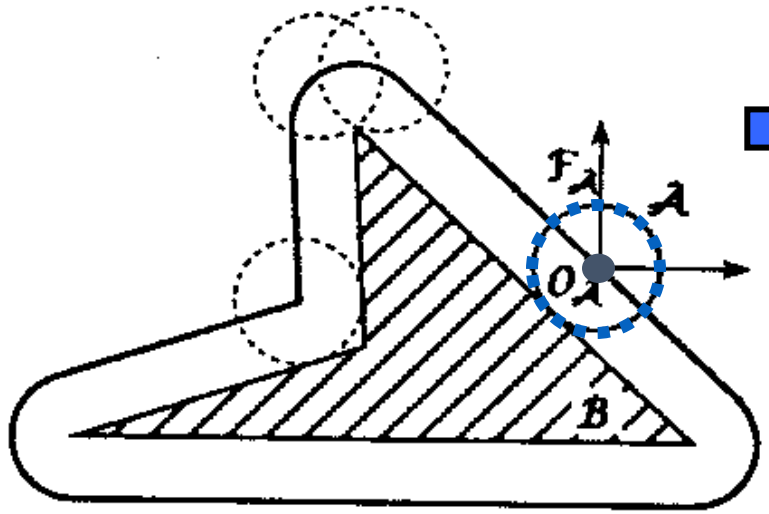
Configuration Space: Accommodate Robot Size



Disc in 2-D workspace

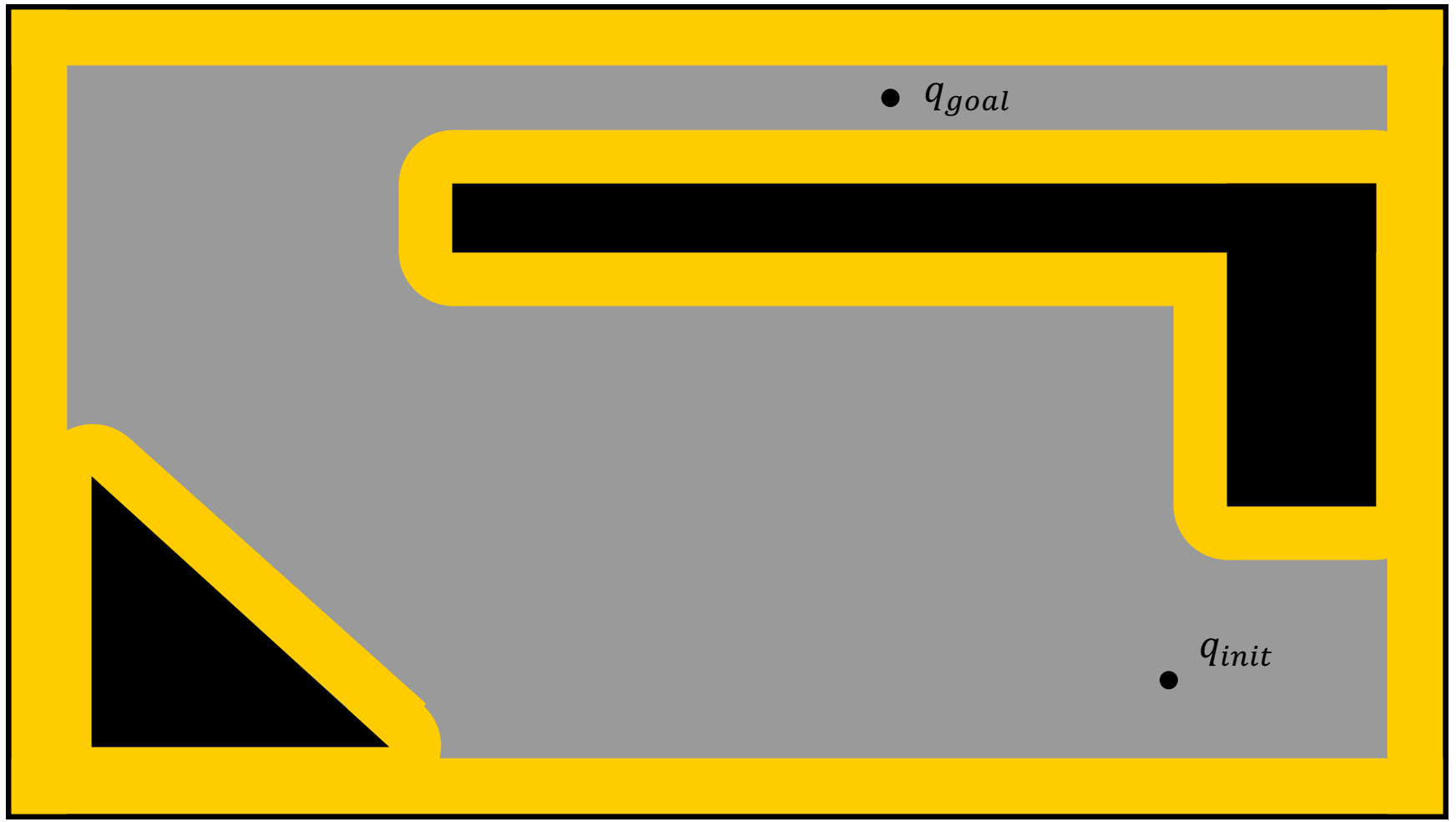
workspace

configuration space



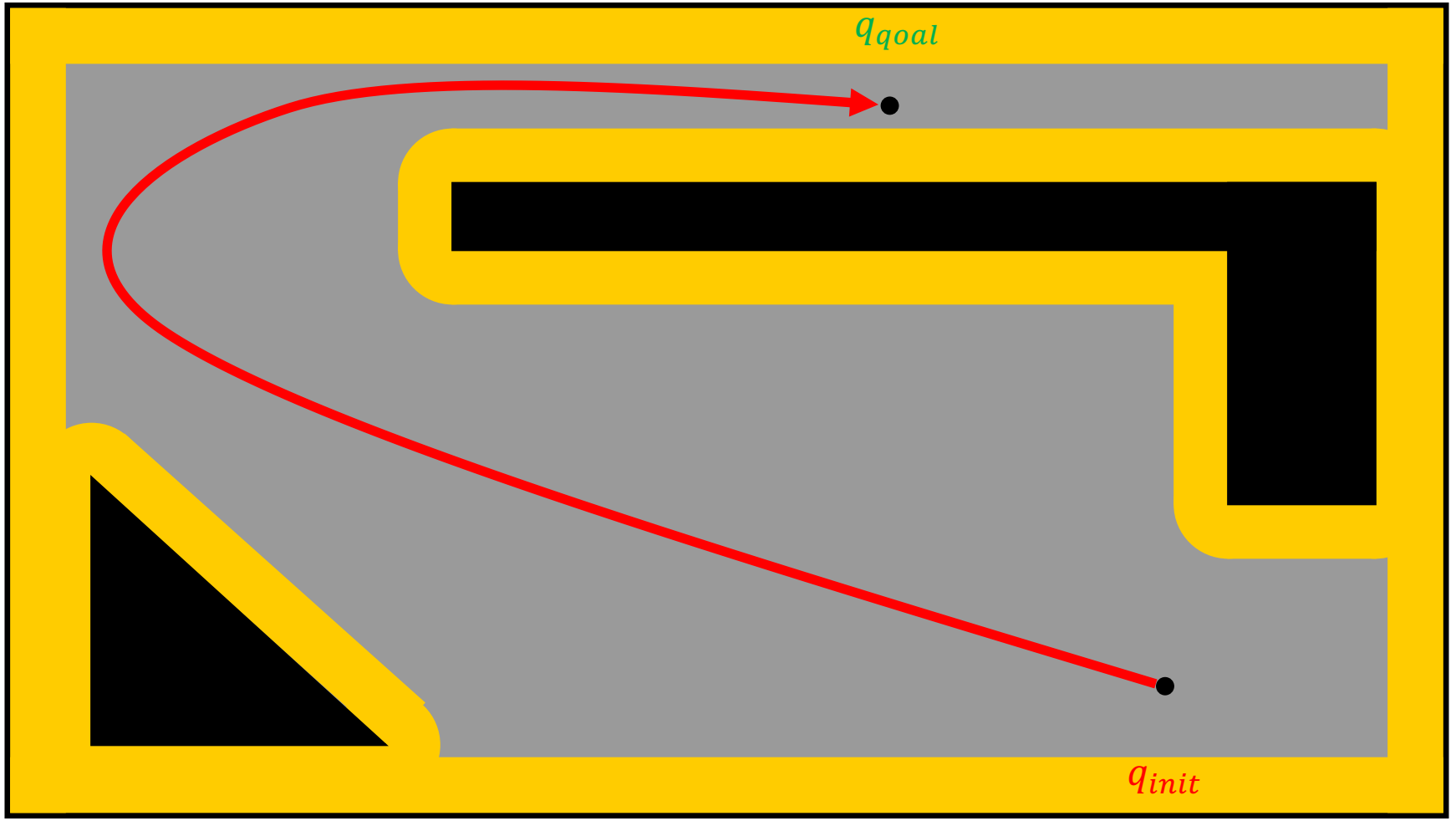
Planning a Collision-Free Path

Find a collision-free path from q_{init} to q_{goal}



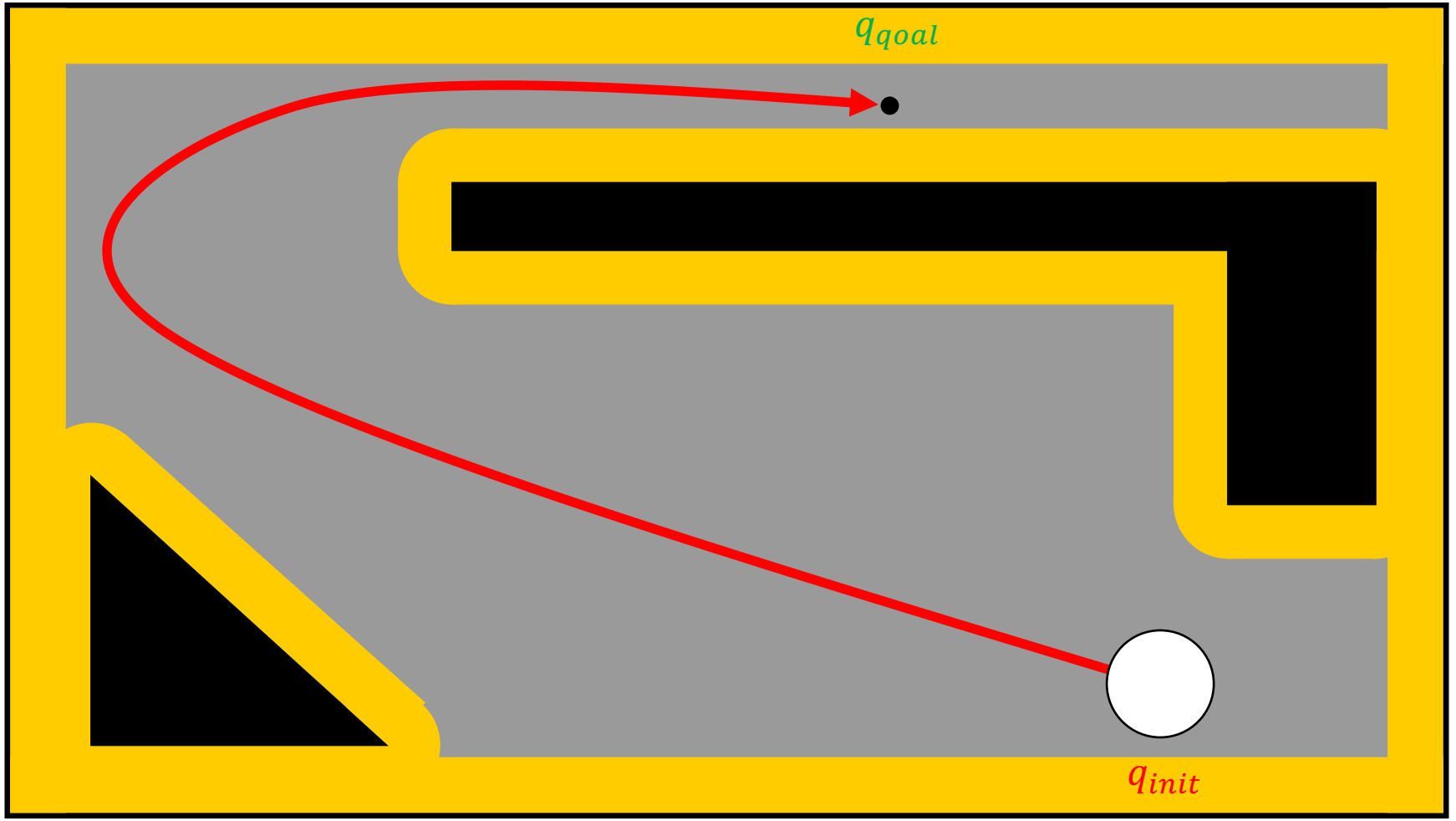
Planning a Collision-Free Path

Find a collision-free path from q_{init} to q_{goal}



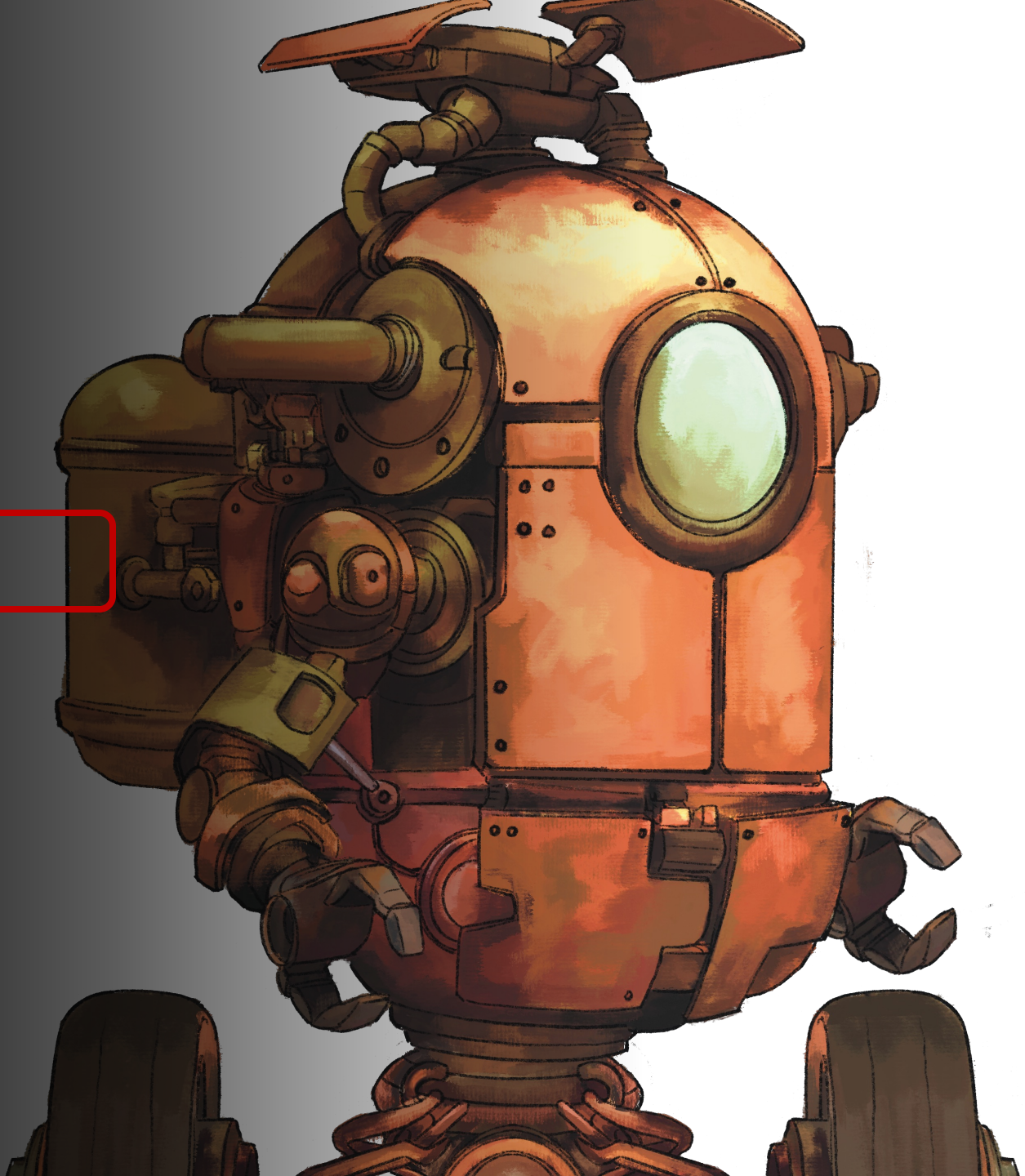
Planning a Collision-Free Path

Find a collision-free path from q_{init} to q_{goal}



Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants



Completeness

- Complete algorithm → Slow
  - A **complete** algorithm finds a path if one exists and reports no otherwise in finite time.
  - Heuristic algorithm → Unreliable (e.g., potential fields)

- **Probabilistic completeness**
  - Intuition: If there is a solution path, the algorithm will find it with high probability.

The Rise of Monte Carlo Techniques

- KEY IDEA:
  Rather than exhaustively explore ALL possibilities, randomly explore a smaller subset of possibilities while keeping track of progress

- Facilities “probing” deeper in a search tree much earlier than any exhaustive algorithm can

- What’s the catch?
  Typically, we must sacrifice both *completeness* and *optimality*
  Classic tradeoff between solution quality and runtime performance

Sampling Based Planning:

Search for collision-free path only by sampling points.

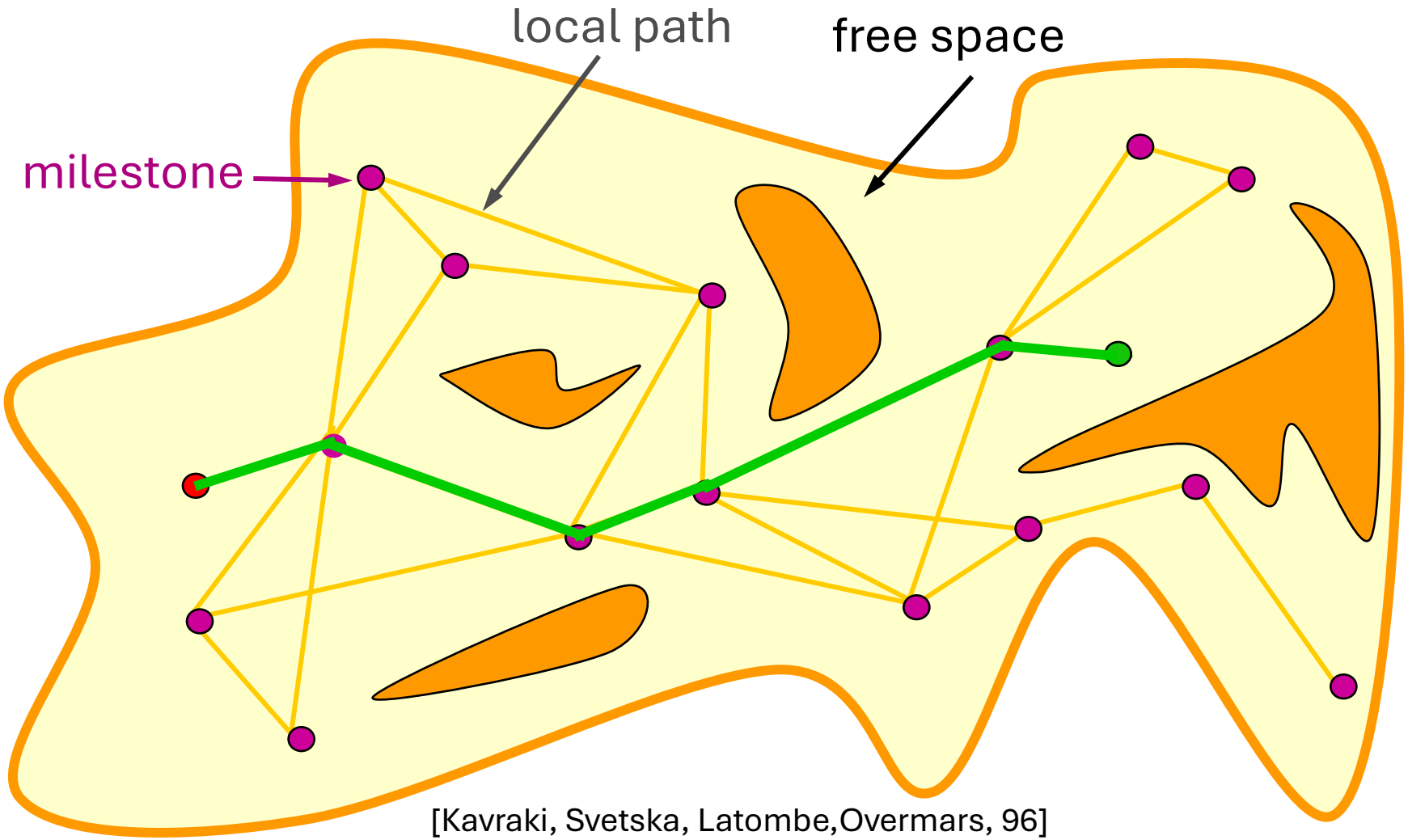
Probabilistic Road Map (PRM)

- Probabilistic Roadmap methods proceed in two phases:

1. Preprocessing Phase – to construct the roadmap *G*
2. Query Phase – to search given q_{init} and q_{goal}

The roadmap is an undirected graph *G* = (*N*, *E*). The nodes in N are a set of configurations of the robot chosen over C-free. The edges in E correspond to feasible (straight-line) paths.

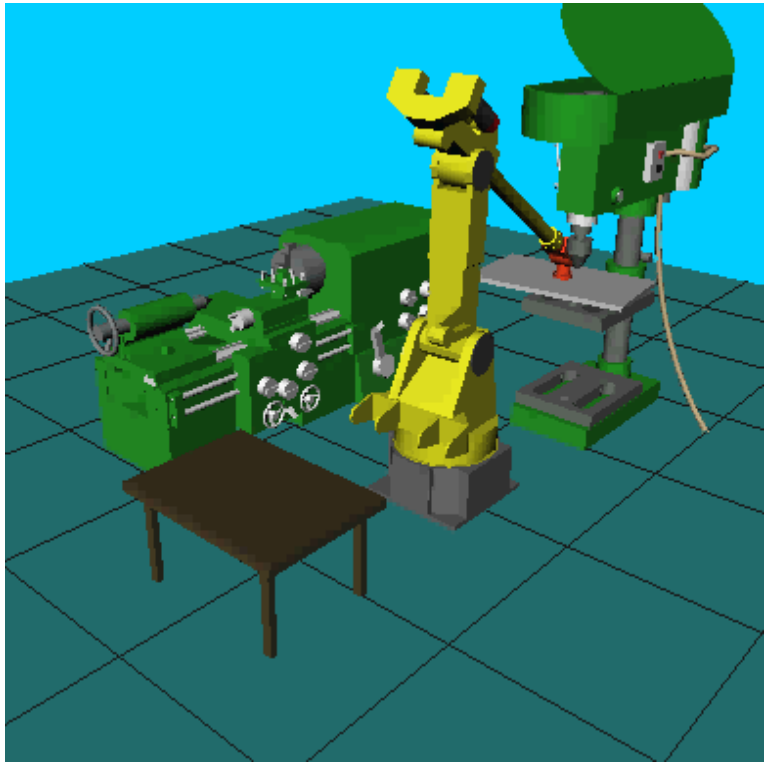
Probabilistic Roadmap (PRM): multiple queries



[Kavraki, Svetska, Latombe,Overmars, 96]

Assumptions

- Static obstacles
- Many queries to be processed in the same environment
- Examples
  - Navigation in static virtual environments
  - Robot manipulator arm in a workcell
- Advantages:
  - Amortize the cost of planning over many problems
  - Probabilistically complete



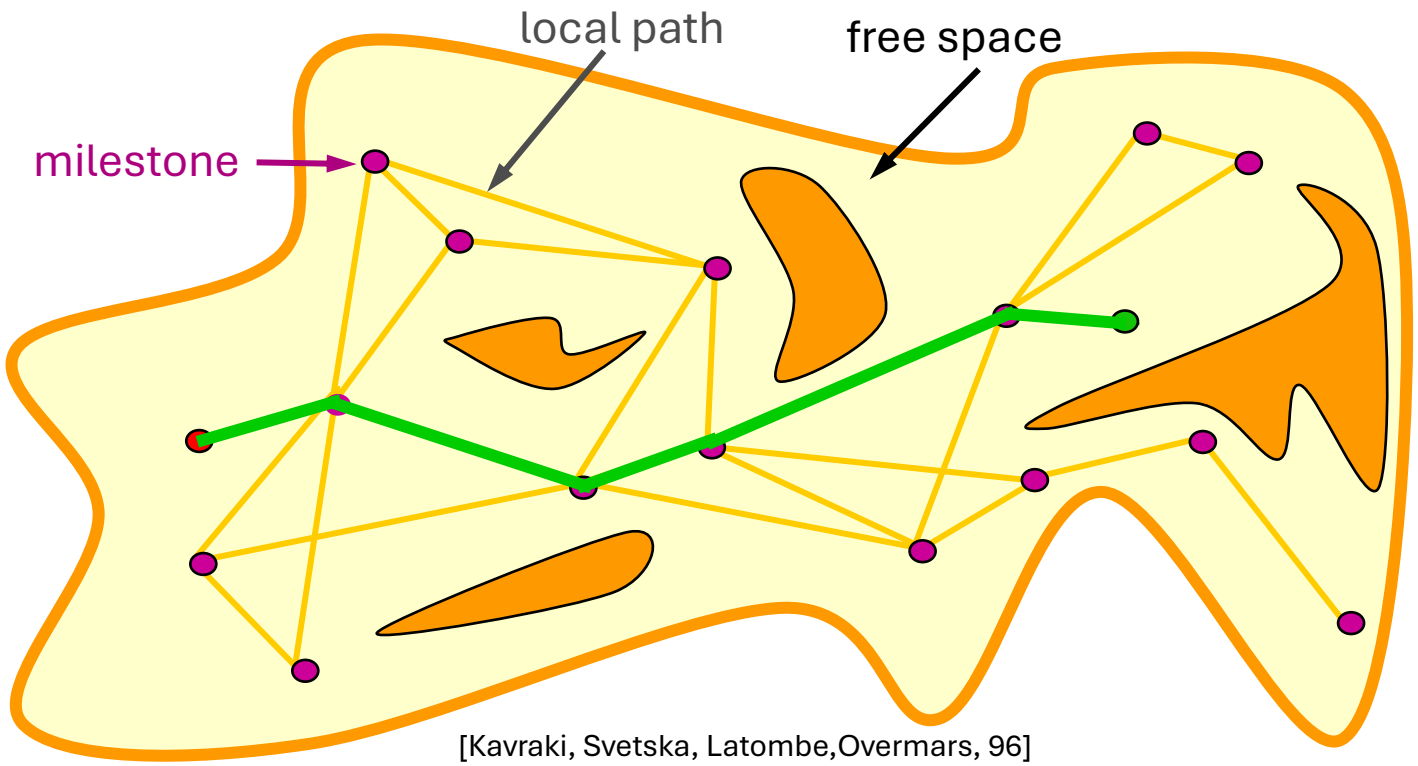
Uniform sampling

Input: geometry of the moving object & obstacles
Output: roadmap G = (V, E)

1: V $\leftarrow \emptyset$ and E $\leftarrow \emptyset$.
2: **repeat**
3: q $\leftarrow$ a configuration sampled uniformly at random from C.
4: **if CLEAR**(q)**then**
5: Add q to V.
6: $N_q \leftarrow$ a set of nodes in V that are close to q.
7: **for** each q' $\in N_q$, in order of increasing d(q,q')
8: **if LINK**(q',q)**then**
9: Add an edge between q and q' to E.

Some terminology

- The graph G is called a **probabilistic roadmap**.
- The nodes in G are called **milestones**.



[Kavraki, Svetska, Latombe,Overmars, 96]

How do we determine a *random free* configuration?

- We want the nodes of V to be a ***uniform*** sampling of Q_{free}
  - Draw each of its coordinates from the interval of values of the corresponding degrees of freedom. (Use the uniform probability distribution over the interval)
  - Check for collision both with robot itself and with obstacles
  - If collision free, add to V, otherwise discard
  - What about rotations? Since DDRs live in the plane, we can merely sample uniformly in the interval $[0, 2\pi]$.

What's the local path planner: Link(q',q) ?

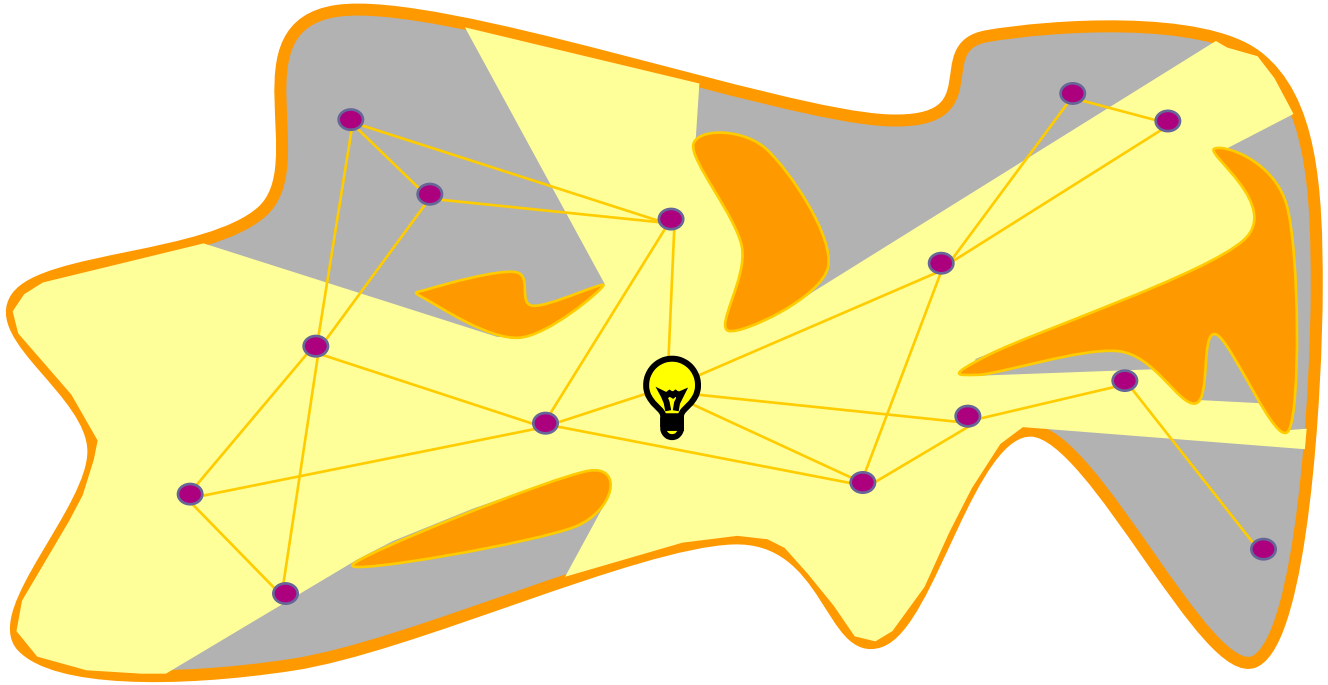
- There are plenty of possibilities
  - Nondeterministic (include a randomized "wandering" component)
    - We'll have to store local paths in roadmap
  - Powerful
    - Slower but maybe we'll need fewer nodes if we do some hard work during roadmap construction?
  - Fast and simple
    - Less powerful, Roadmap will need more nodes

Go with the fast local planner

- Need to make sure start and goal configurations can connect to graph, which requires a somewhat dense roadmap
- Can reuse local planner at query time to connect start and goal configurations
- Don’t need to memorize local paths

Why does it work? Intuition

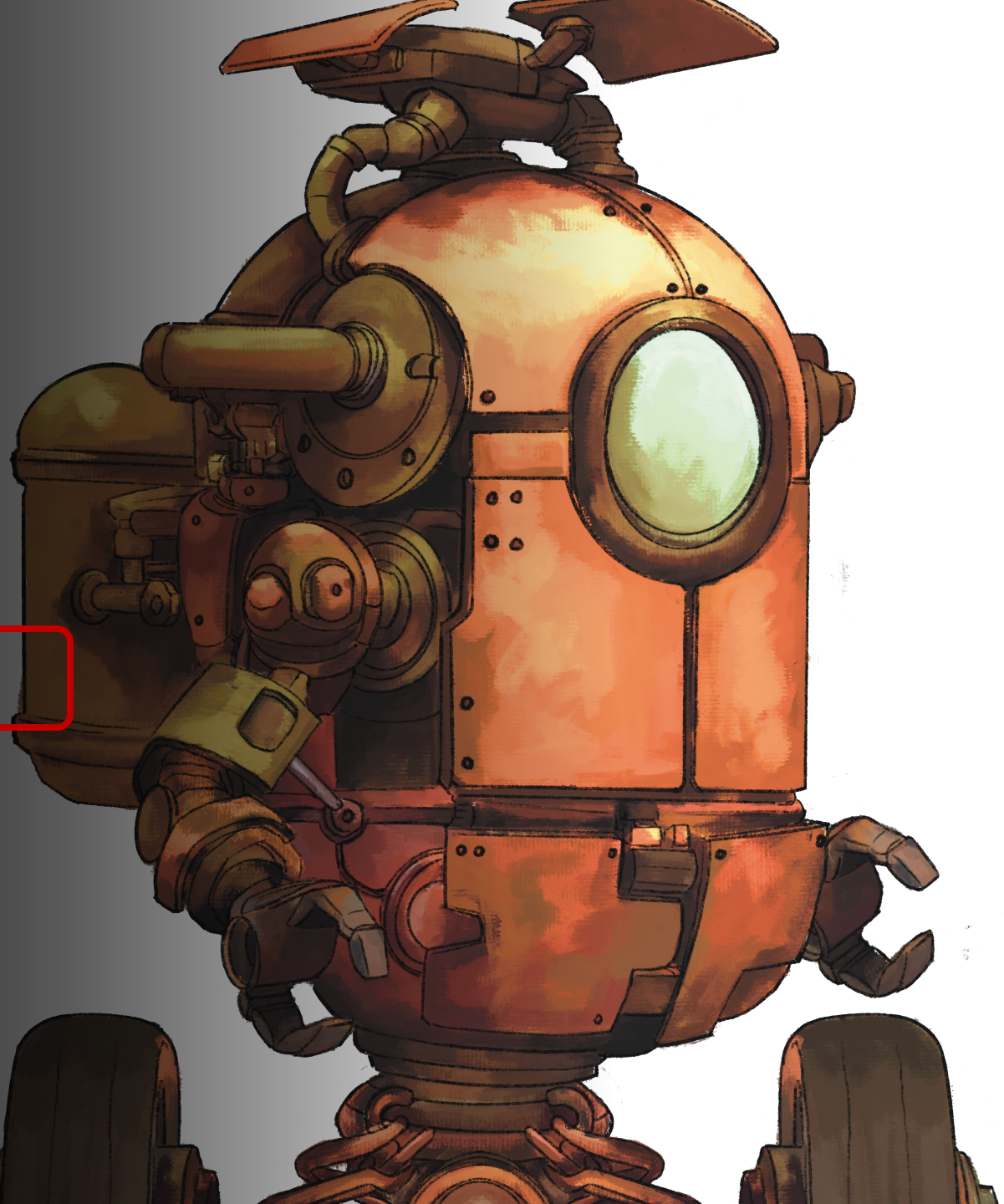
- A small number of milestones **almost** “cover” the **entire** configuration space.



- Rigorous definitions exist (of course!)

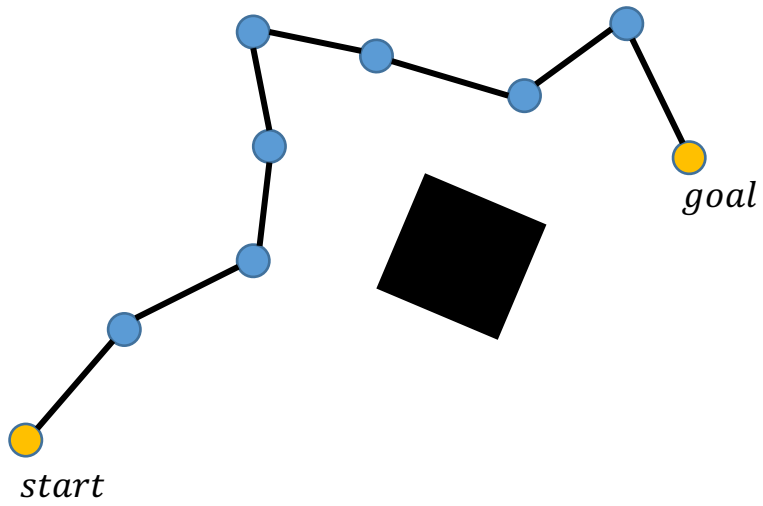
Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants



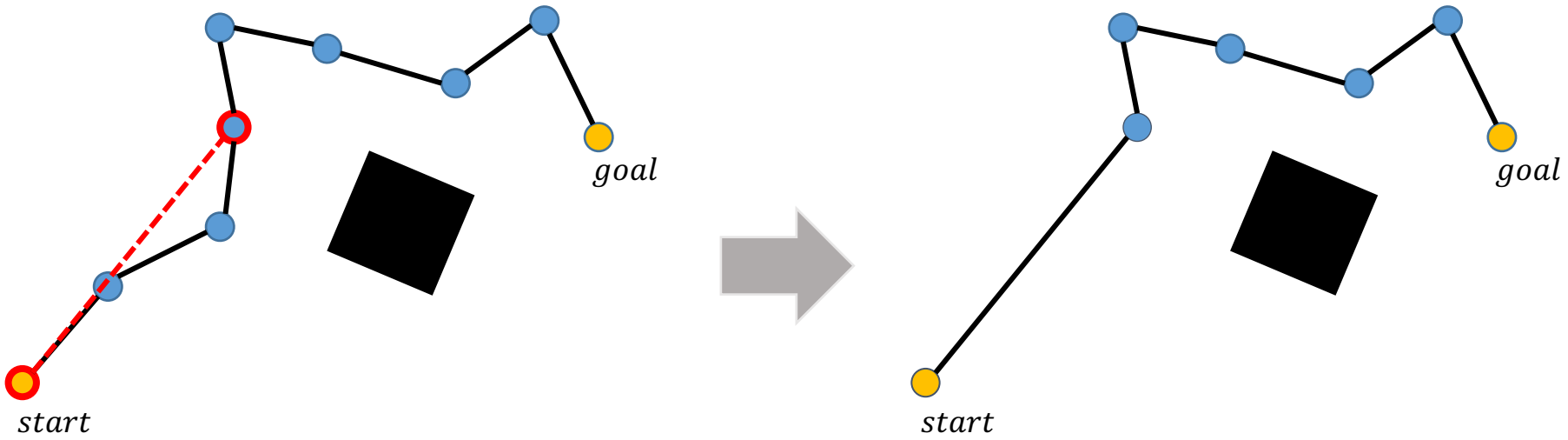
Optimizing the path

- Milestone-based paths are far from optimal and require additional refinement before they are usable
- A typical solution can look like this:



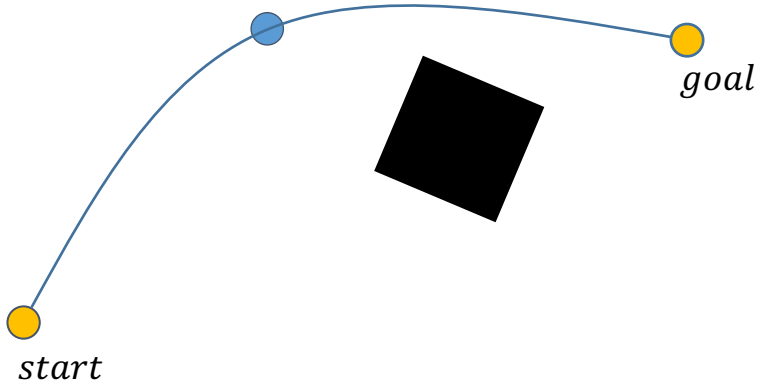
Optimizing the path

- A simple way to improve the path, is to repeatedly pick two nodes at random, and check whether they can be connected by a straight line without collision. If so, use the line to shorten the path.

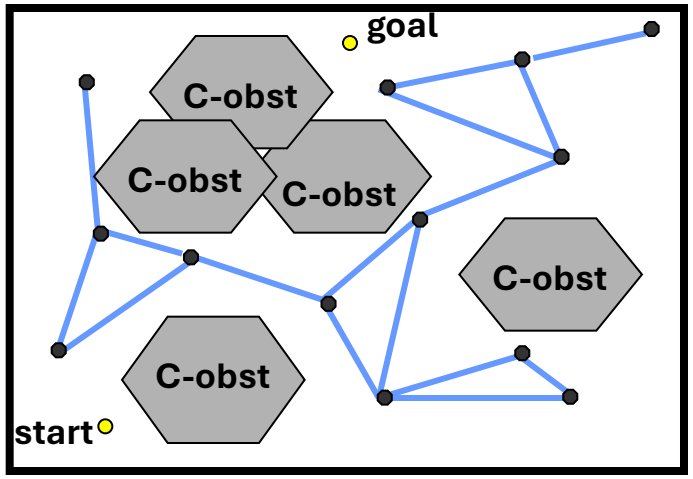


Smoothing the path

- Optionally, the shortened path can then be smoothed to allow for continuous robot motion



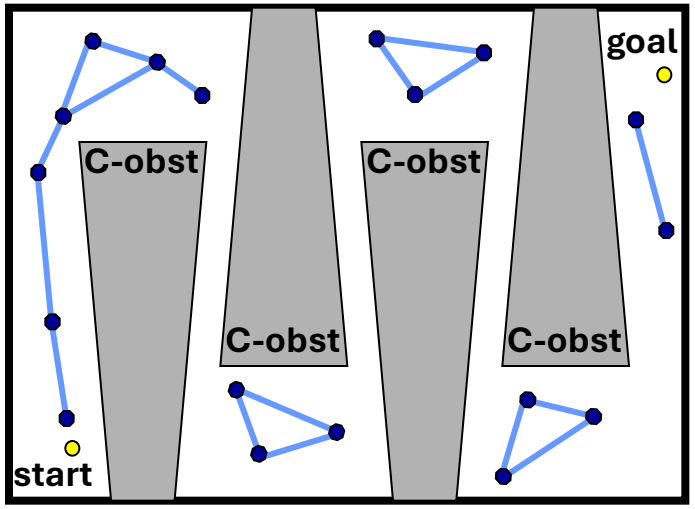
Good news, but bad news too



Sample-based: The Good News

1. *probabilistically complete*
2. Does not construct the C-space
3. applies easily to high-dimensional C-space
4. supports fast queries w/ enough preprocessing

Many success stories where PRMs solve previously unsolved problems

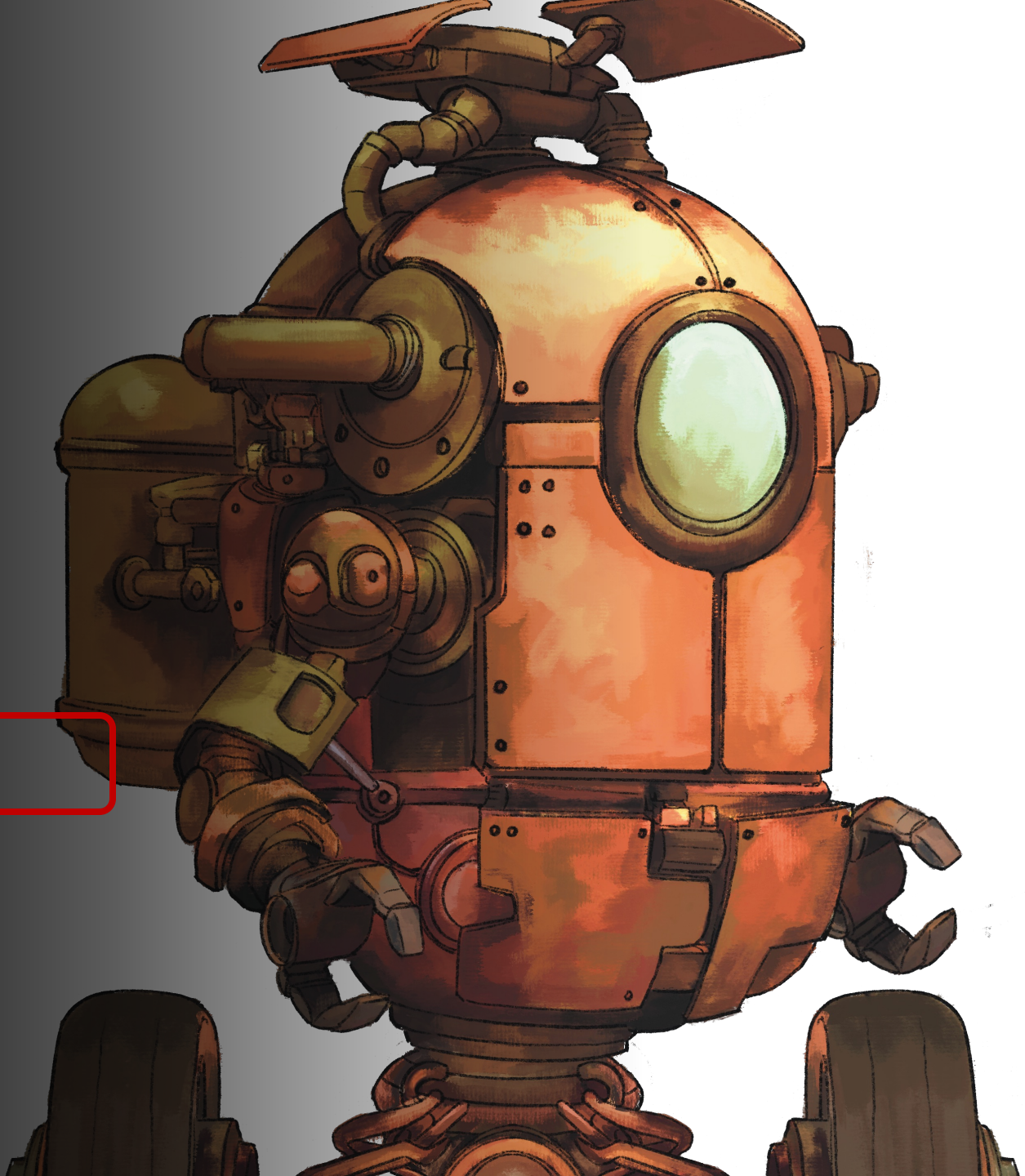


Sample-Based: The Bad News

1. Does not work as well for some problems:
– unlikely to sample nodes in *narrow passages*
– hard to sample/connect nodes on constraint surfaces
2. No optimality or completeness

Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants

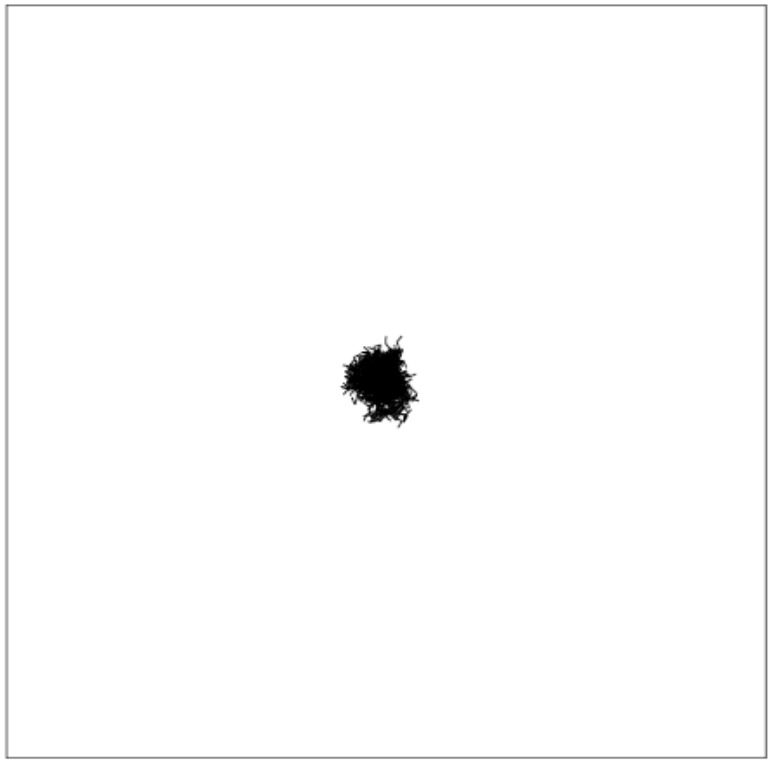


Rapidly-Exploring Random Tree (RRT)

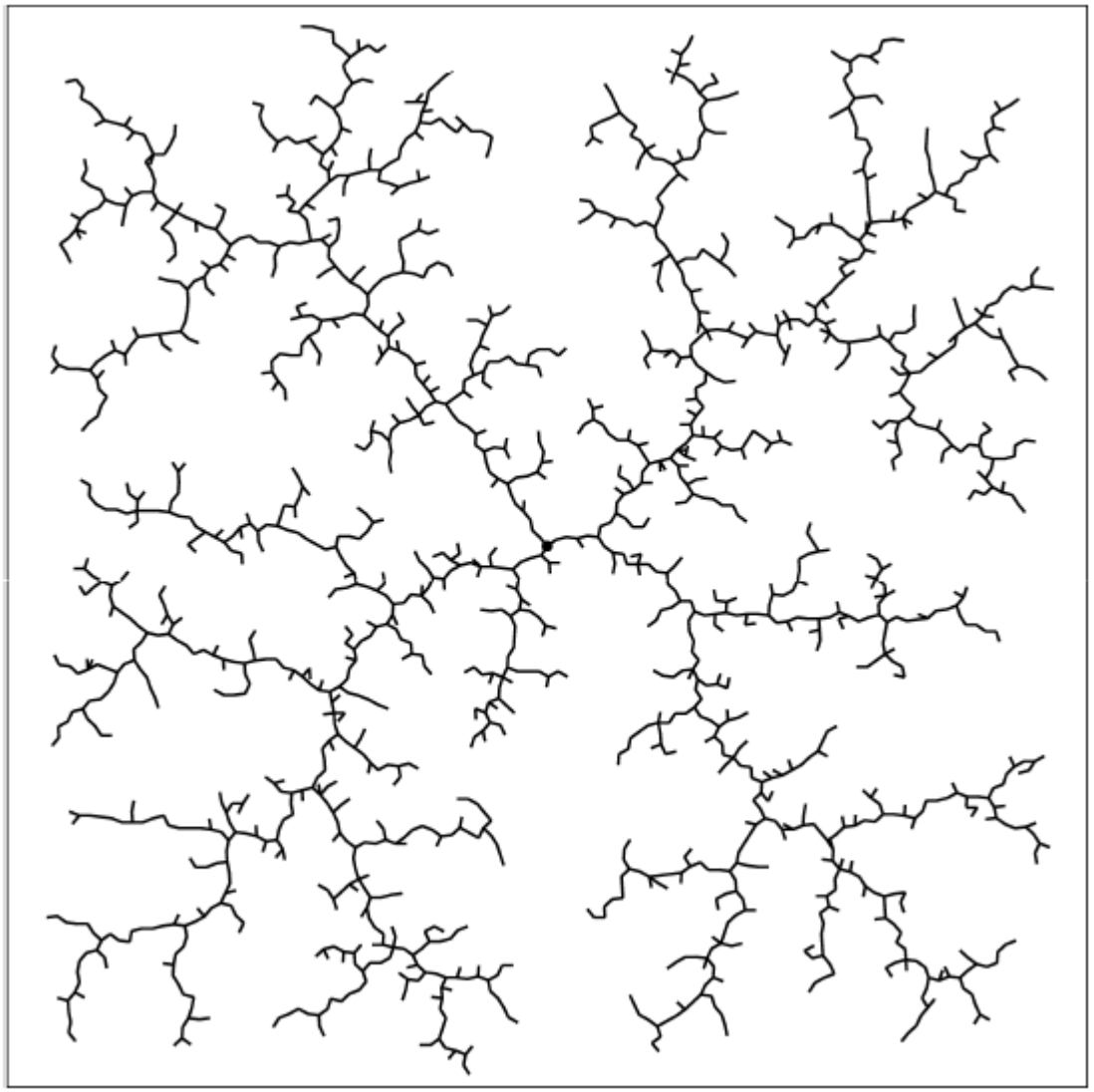
- Searches for a path from the initial configuration to the goal configuration by expanding a search tree
- For each step,
  - The algorithm samples a target configuration and expands the tree towards it.
  - The sample can either be a random configuration or the goal configuration itself, depends on the probability value defined by the user.

Naïve random tree

- Pick a vertex at random
- Move in a random direction to generate a new vertex
- Repeat…



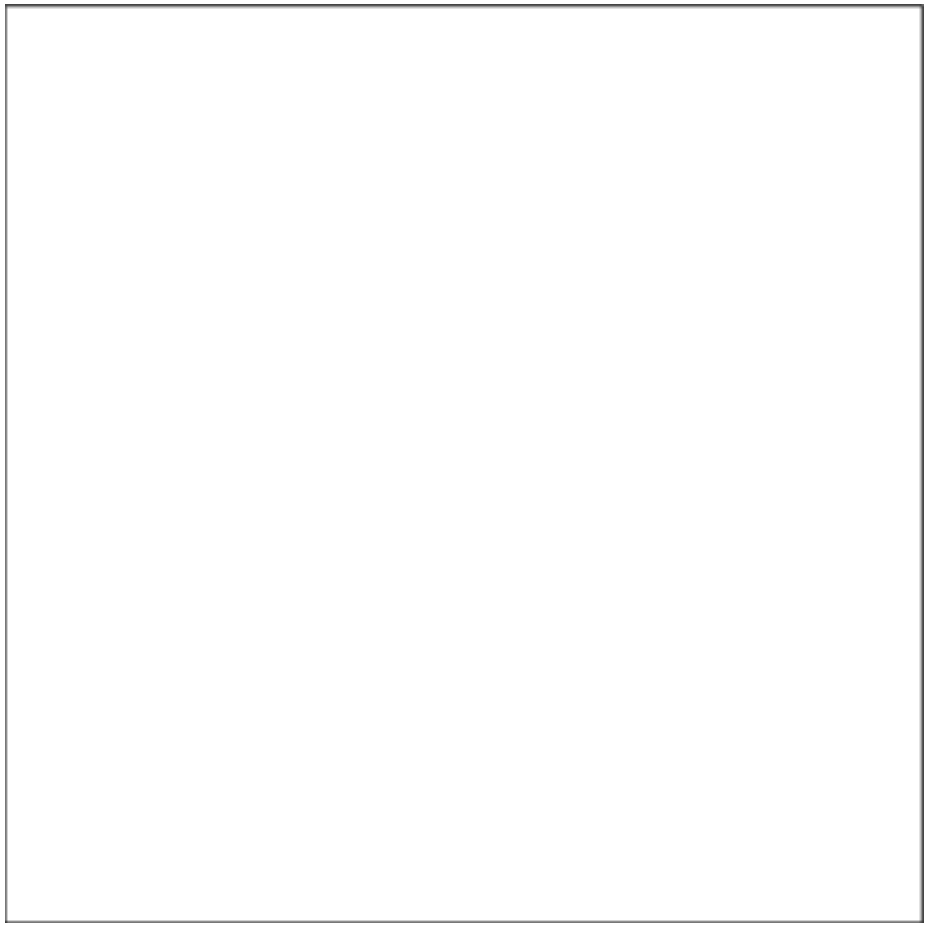
Rapidly-Exploring Random Tree



The Basic Idea: Iteratively expand the tree

- Denote by T_k the tree at iteration k
- Randomly choose a configuration q_{rand}
- Choose $q_{near} = \arg\min_{q \in T_k} d(q, q_{rand})$
  - q_{near} is the nearest existing node in the tree to q_{rand}
- Create a new node, q_{new} by taking a small step from q_{near} toward q_{rand}

RRT Animation



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

RRT

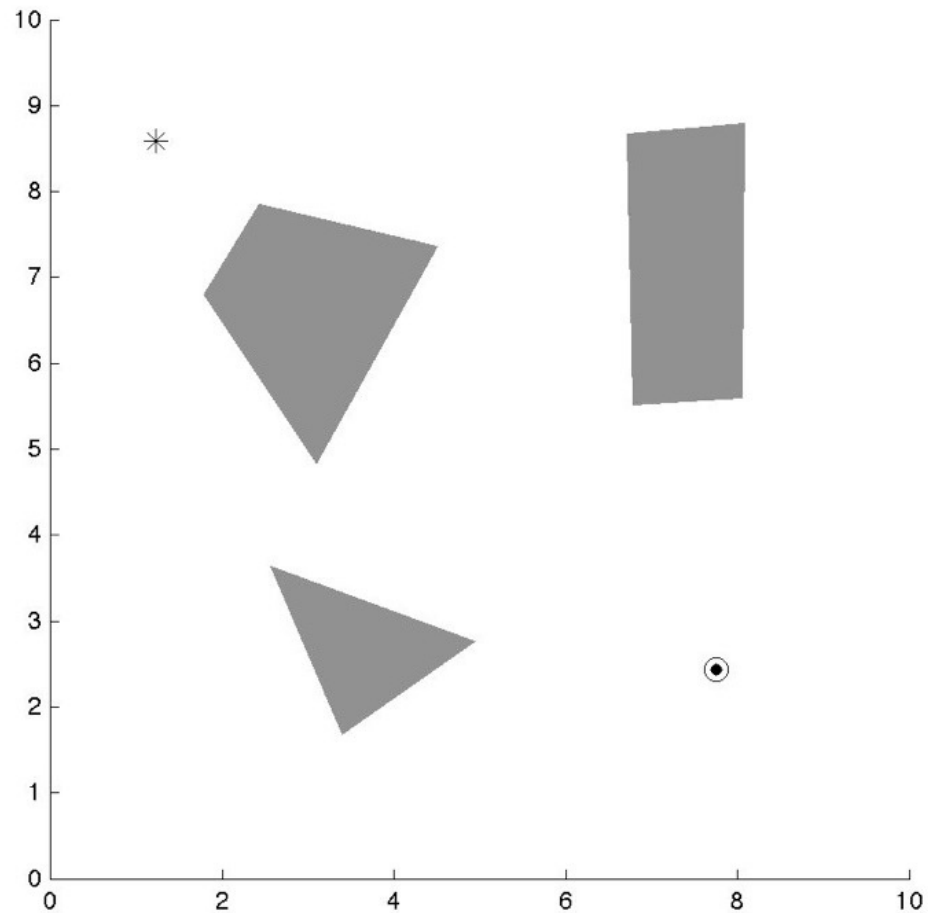
Requires the following functions:

- p = RandomSample()
  - ➢ Uniform random sampling of free configuration space
- v = Nearest(p)
  - ➢ Given point in Cspace, find vertex on tree that is closest to that point
- p’ = Steer(p, goal)
  - ➢ For a point p and a goal point, find p’ that is closer to the goal than p
- ObstacleFree(p)

➢ Check if a given Cspace point is in the free space

RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$
for $i = 1$ to N
  $G \leftarrow (V, E)$
  $x_{rand} \leftarrow RandomSample()$
  $x_{nearest} \leftarrow Nearest(G, x_{rand})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
  if $ObstacleFree(x_{nearest}, x_{new})$
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

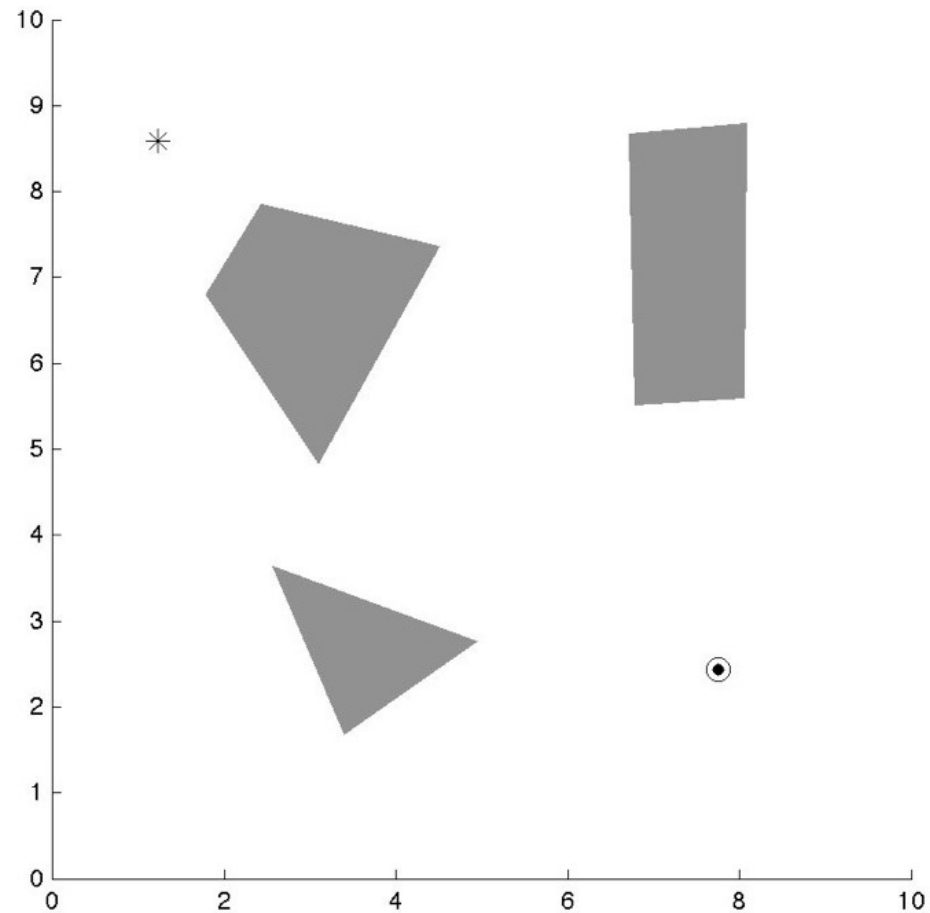
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

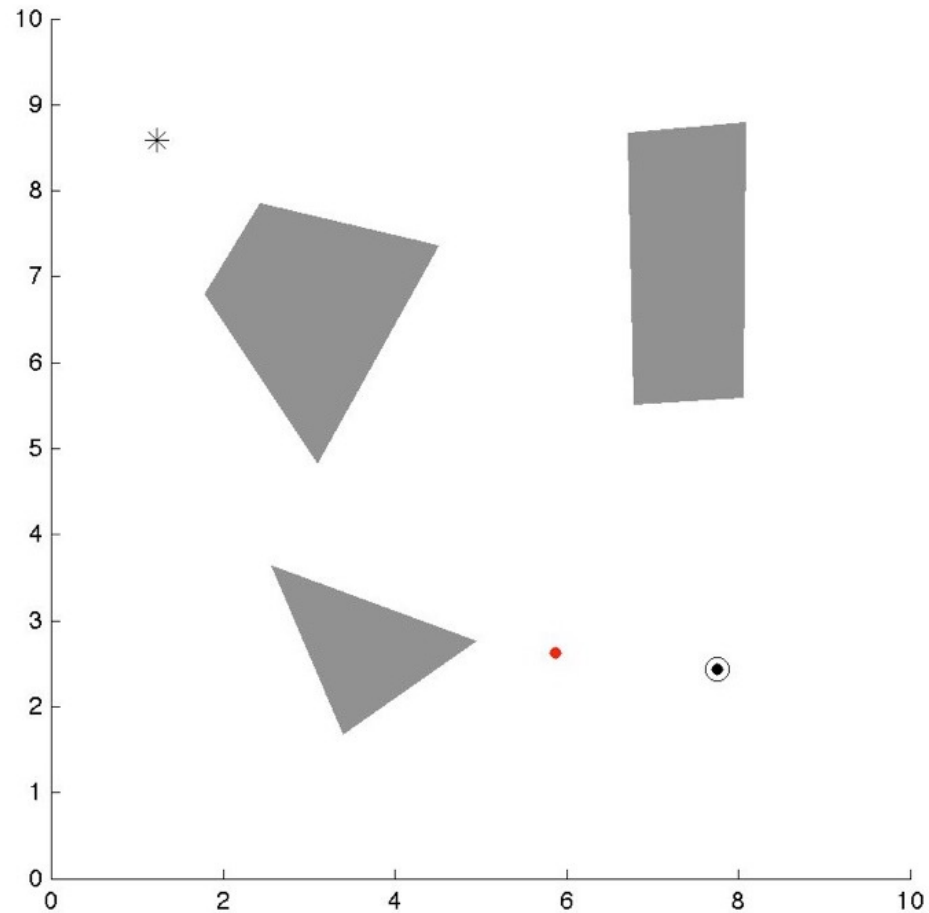
$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$
for $i = 1$ to N
  $G \leftarrow (V, E)$
  $x_{rand} \leftarrow RandomSample()$
  $x_{nearest} \leftarrow Nearest(G, x_{rand})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
  if $ObstacleFree(x_{nearest}, x_{new})$
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

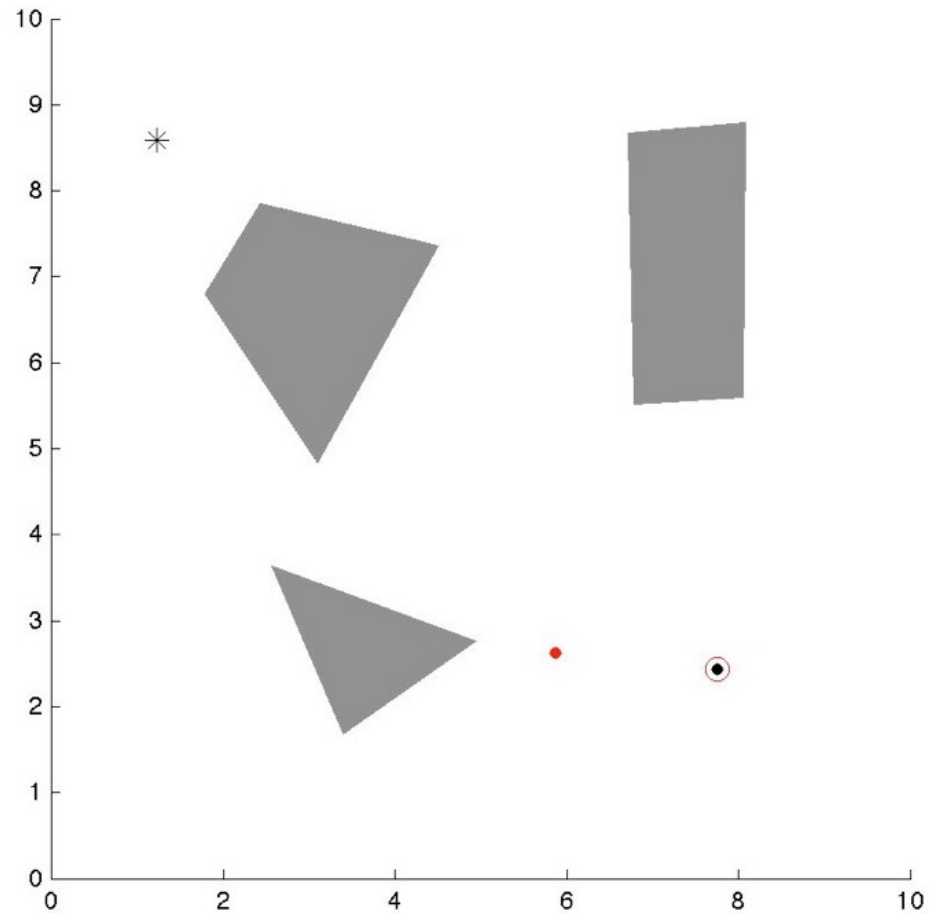
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

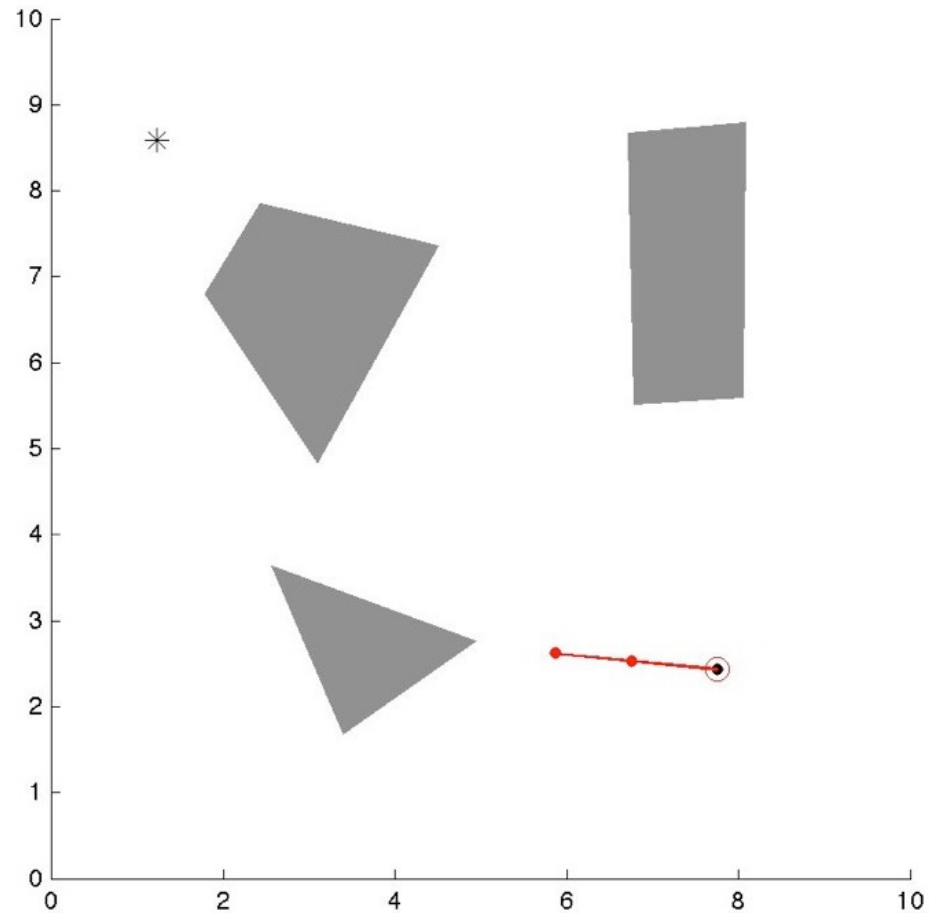
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

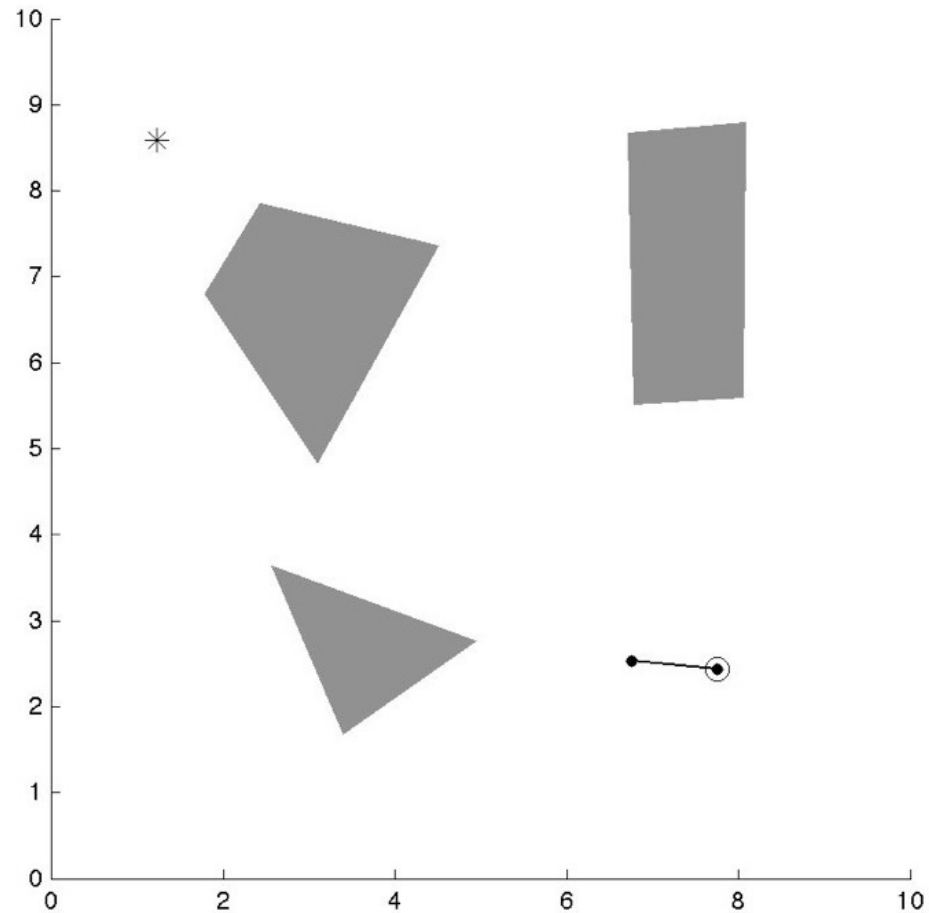
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

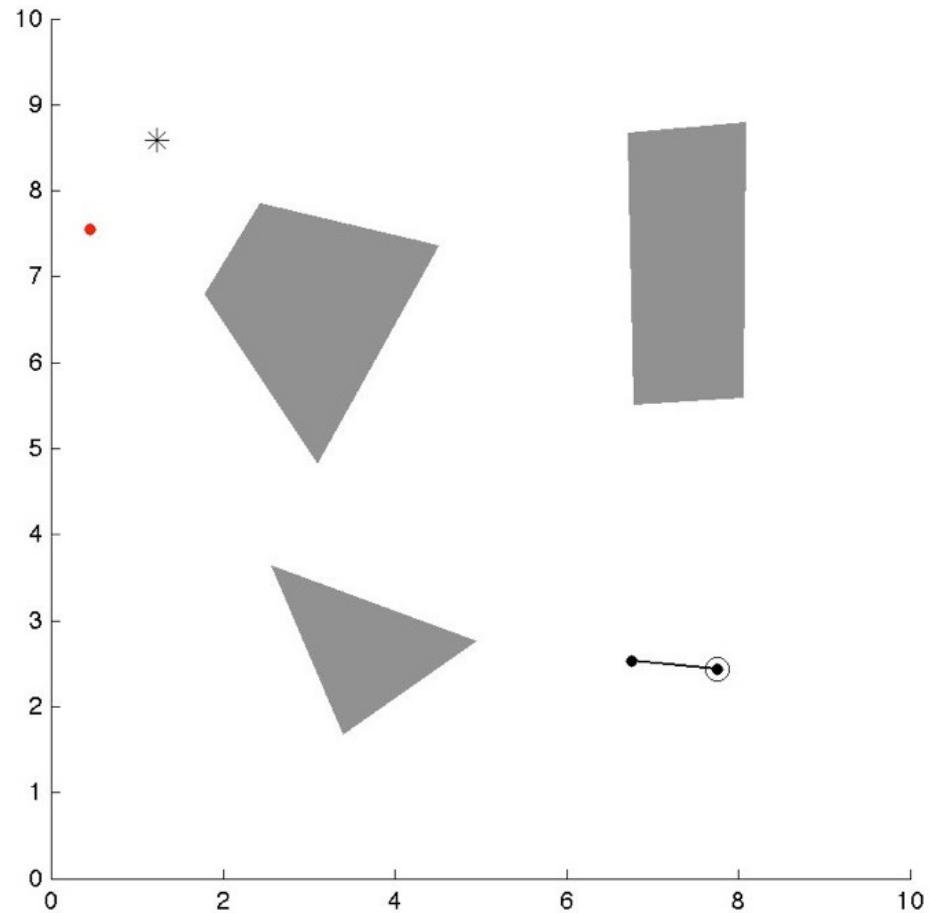
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

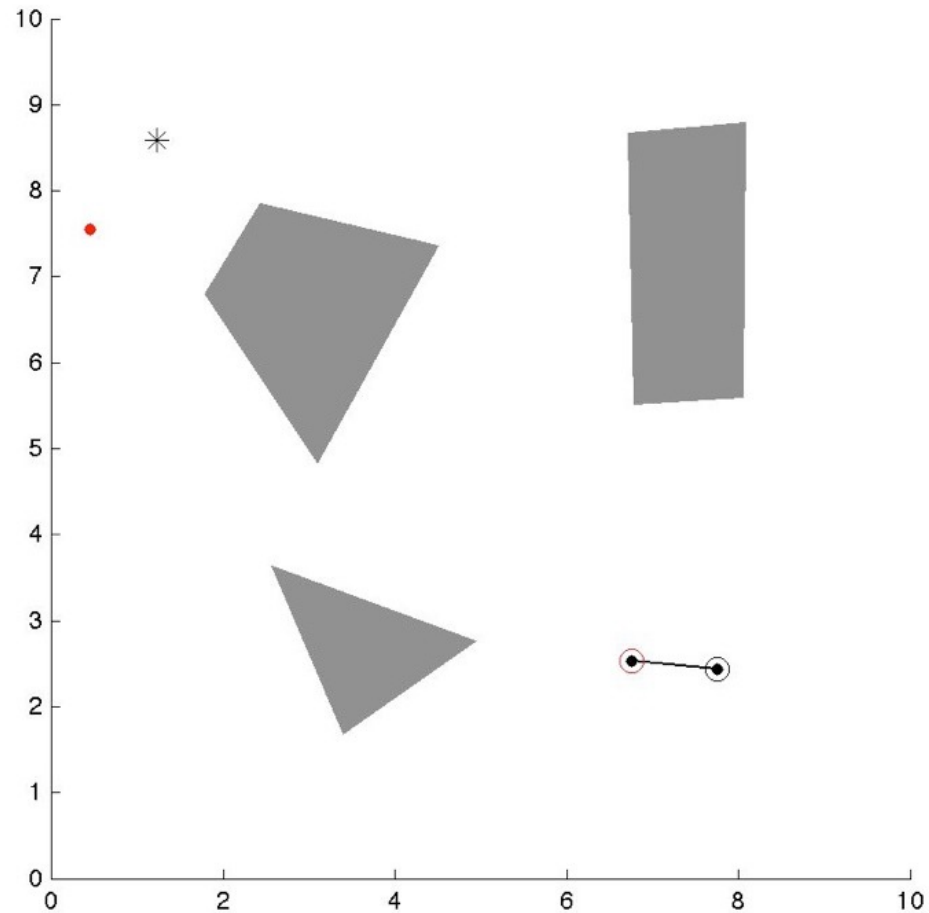
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

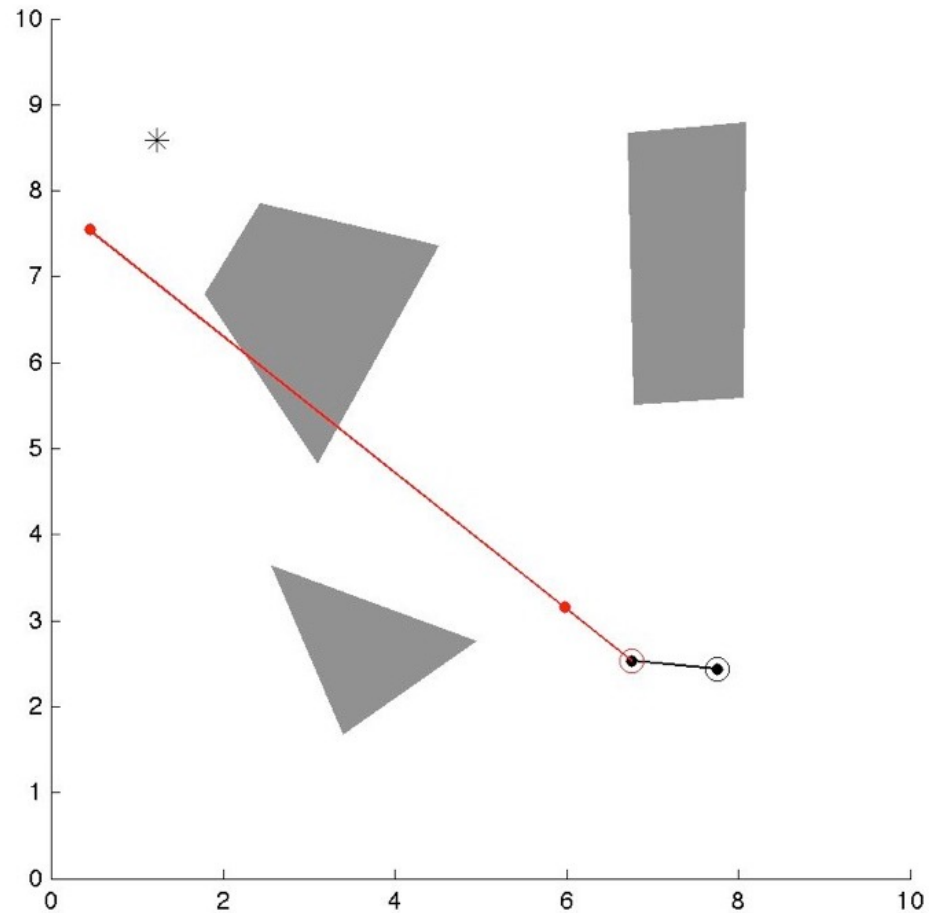
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

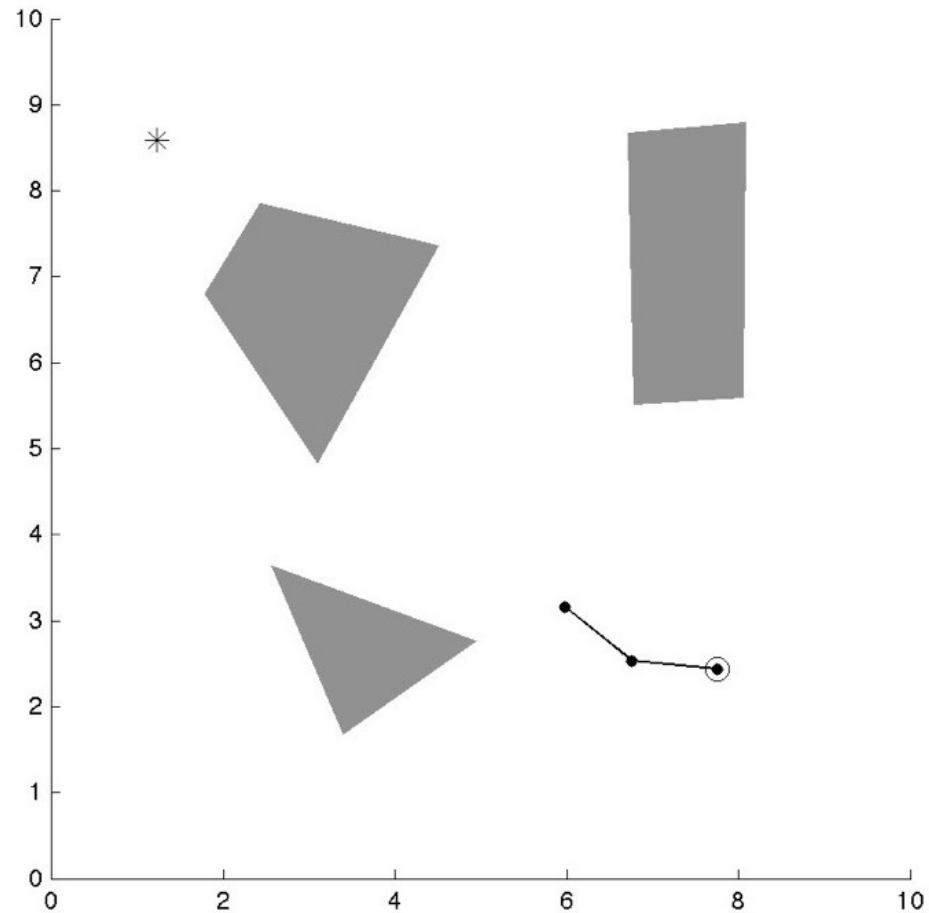
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

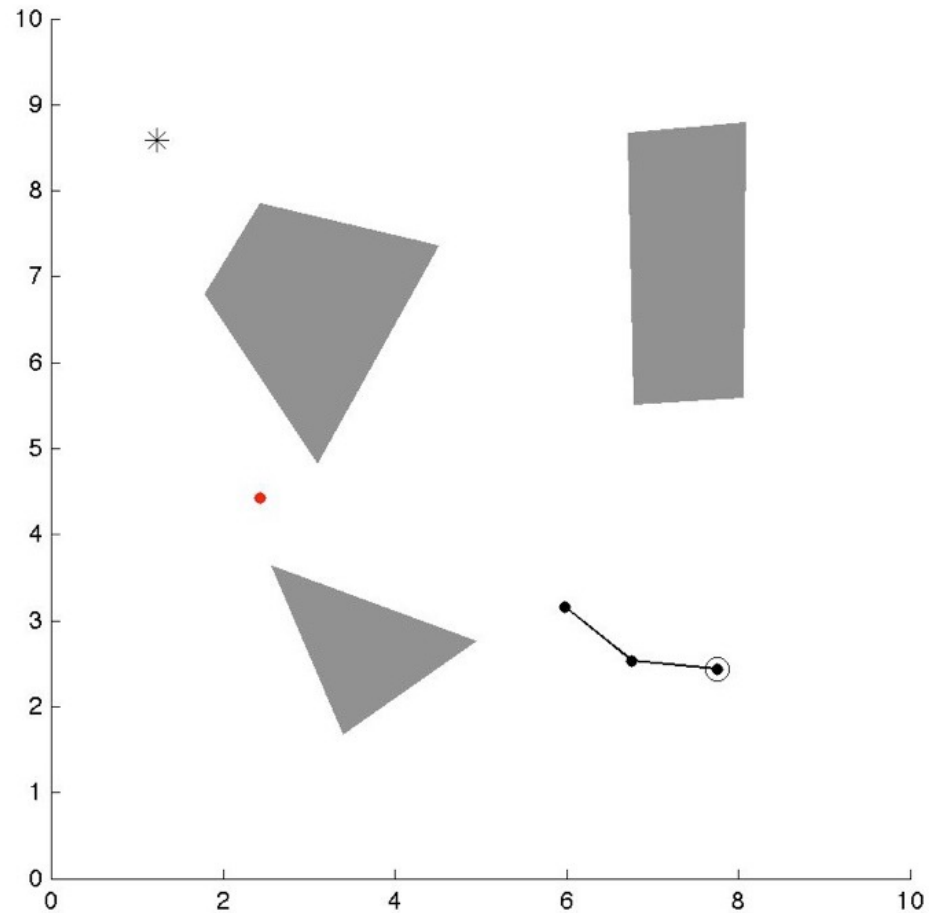
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

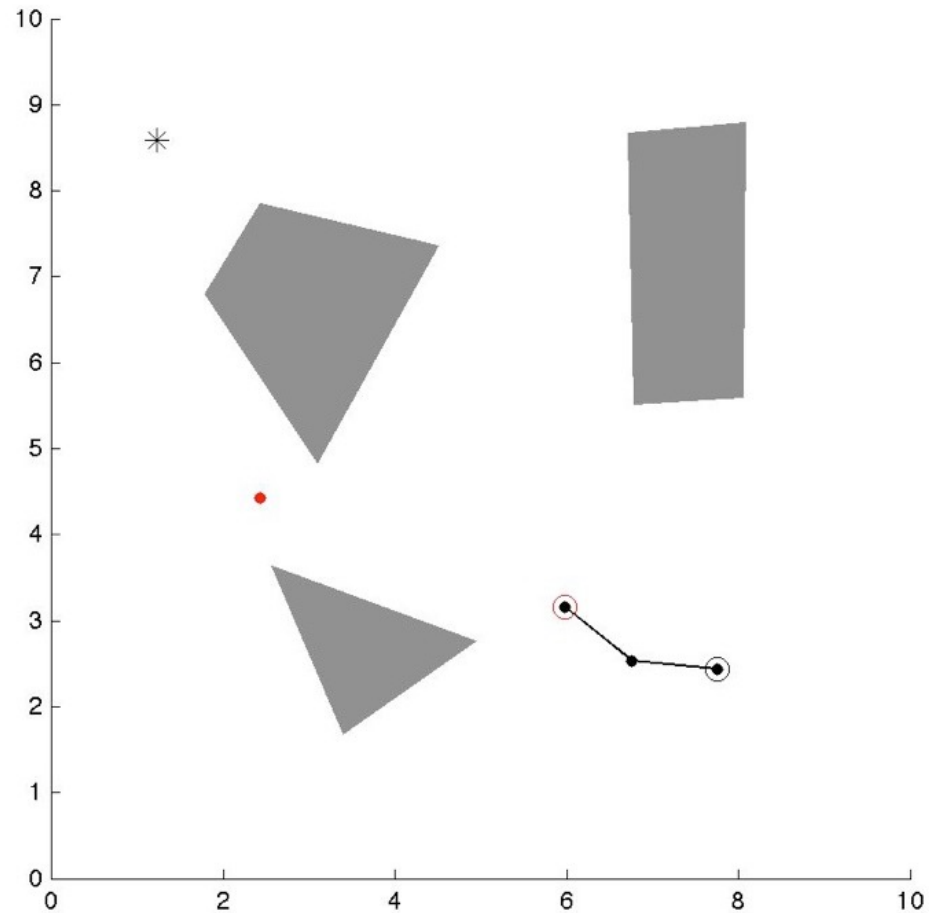
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

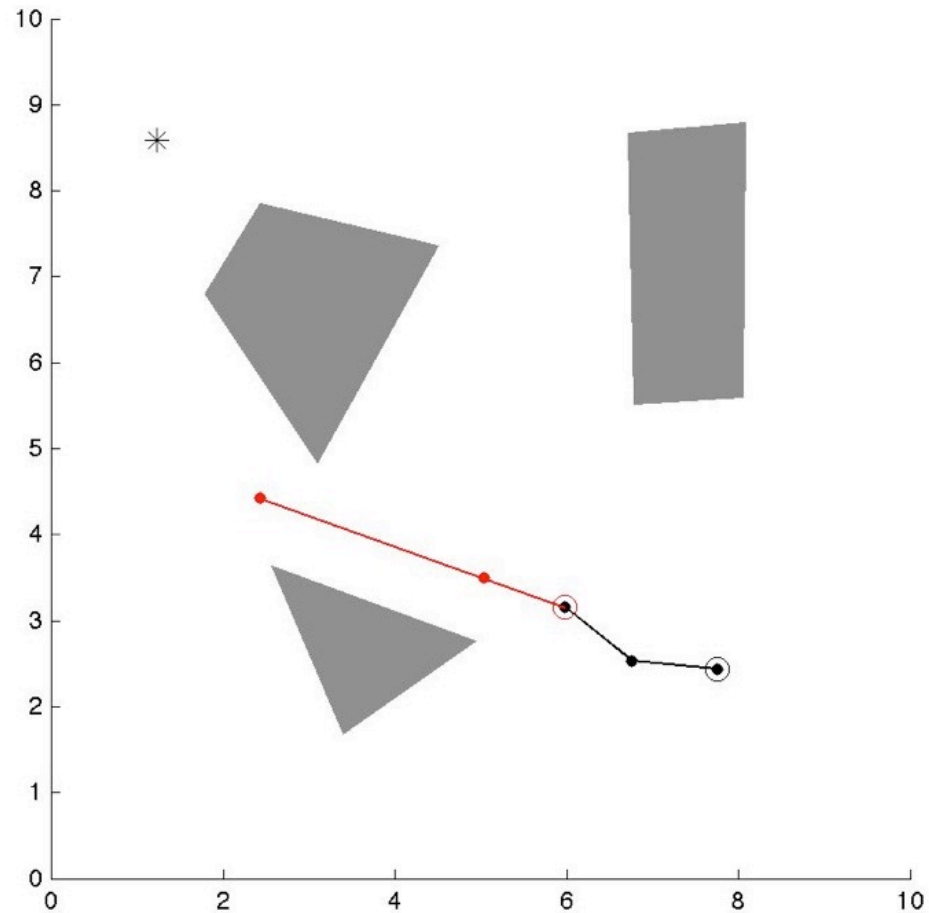
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

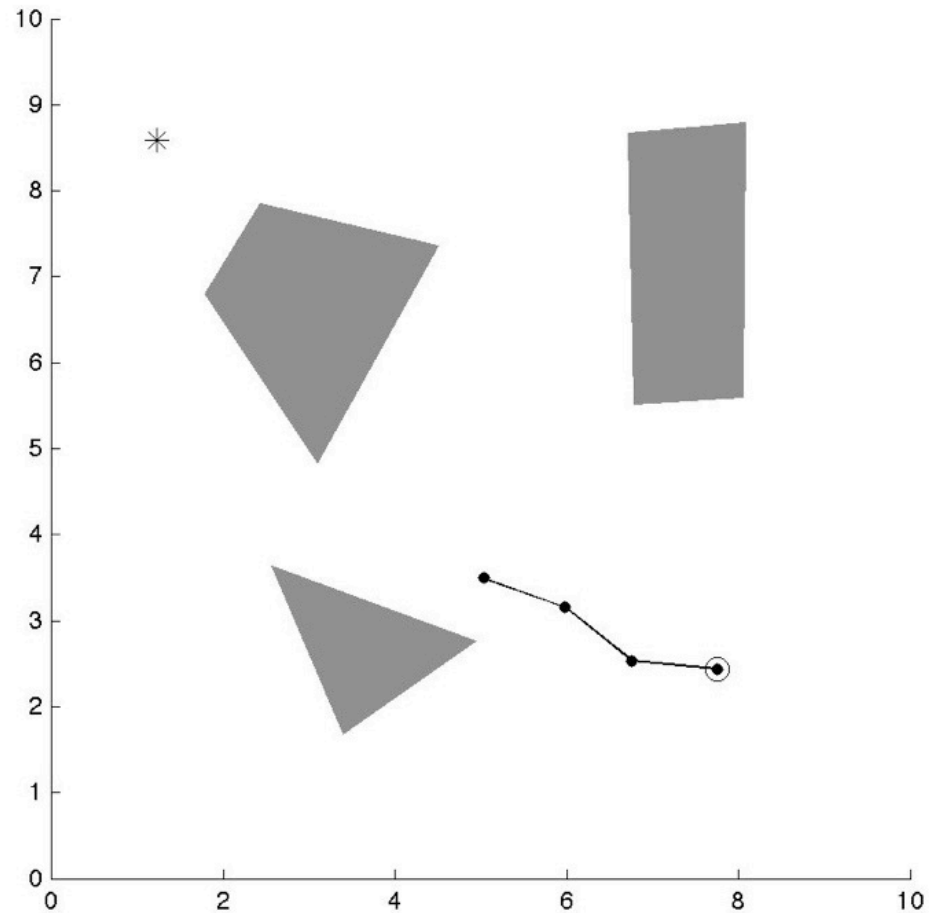
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

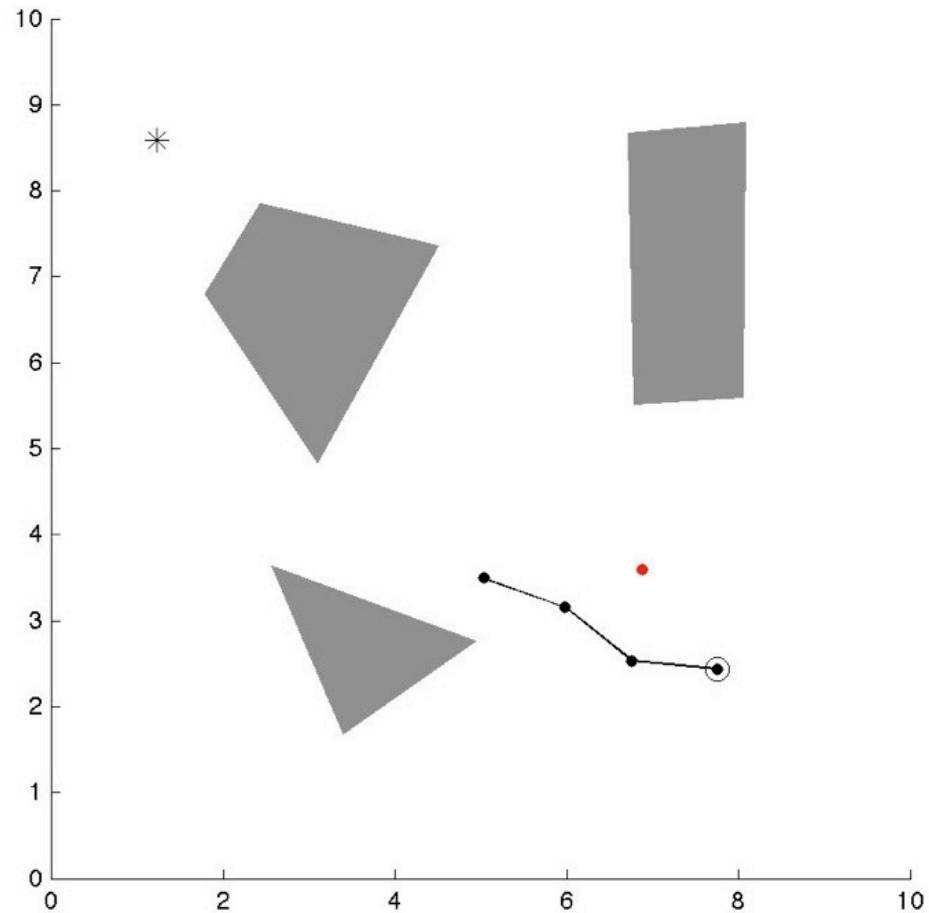
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

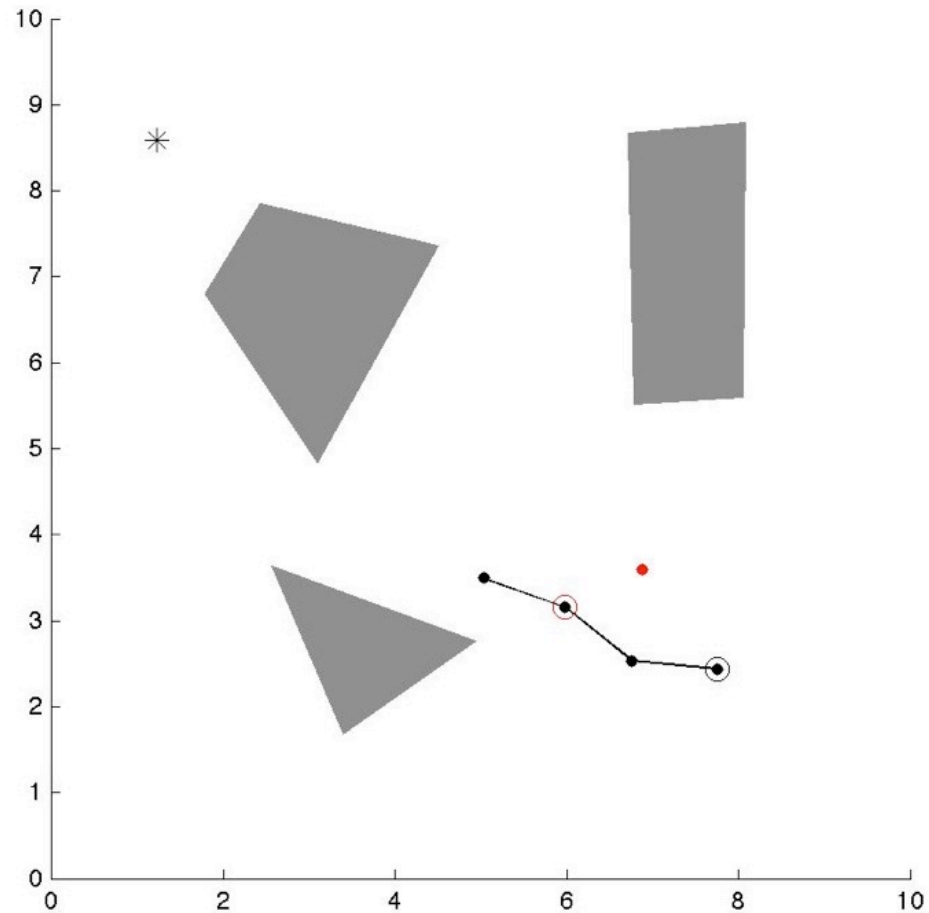
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

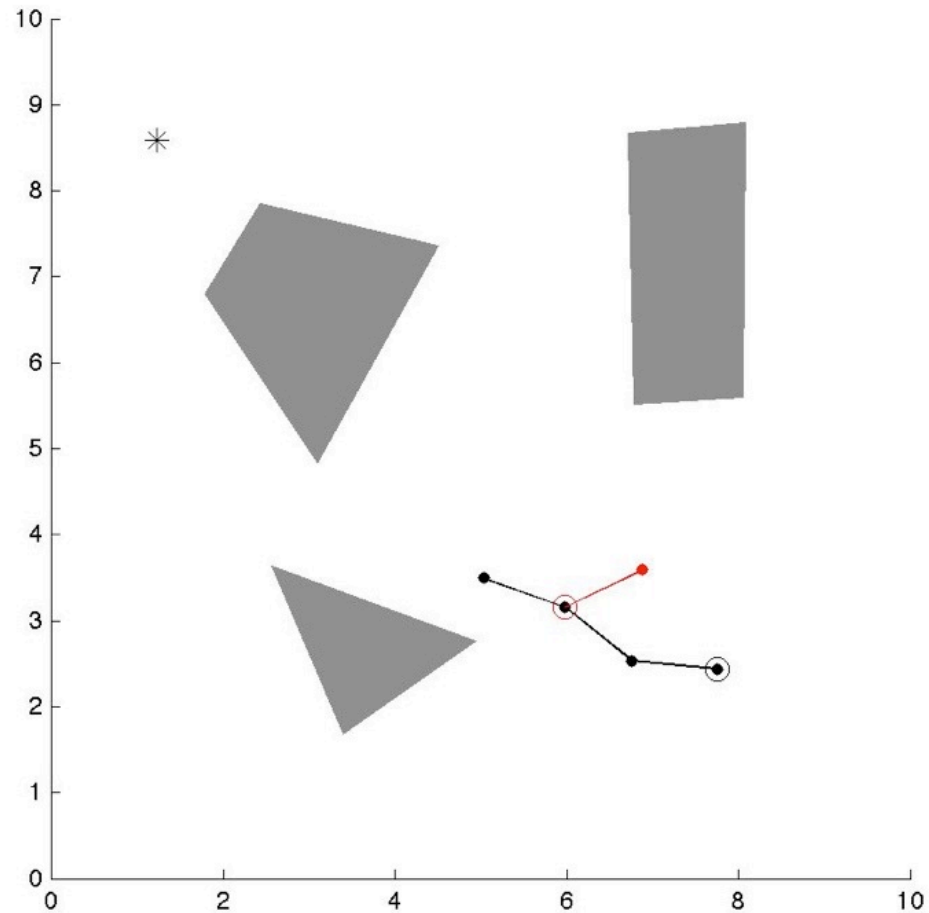
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

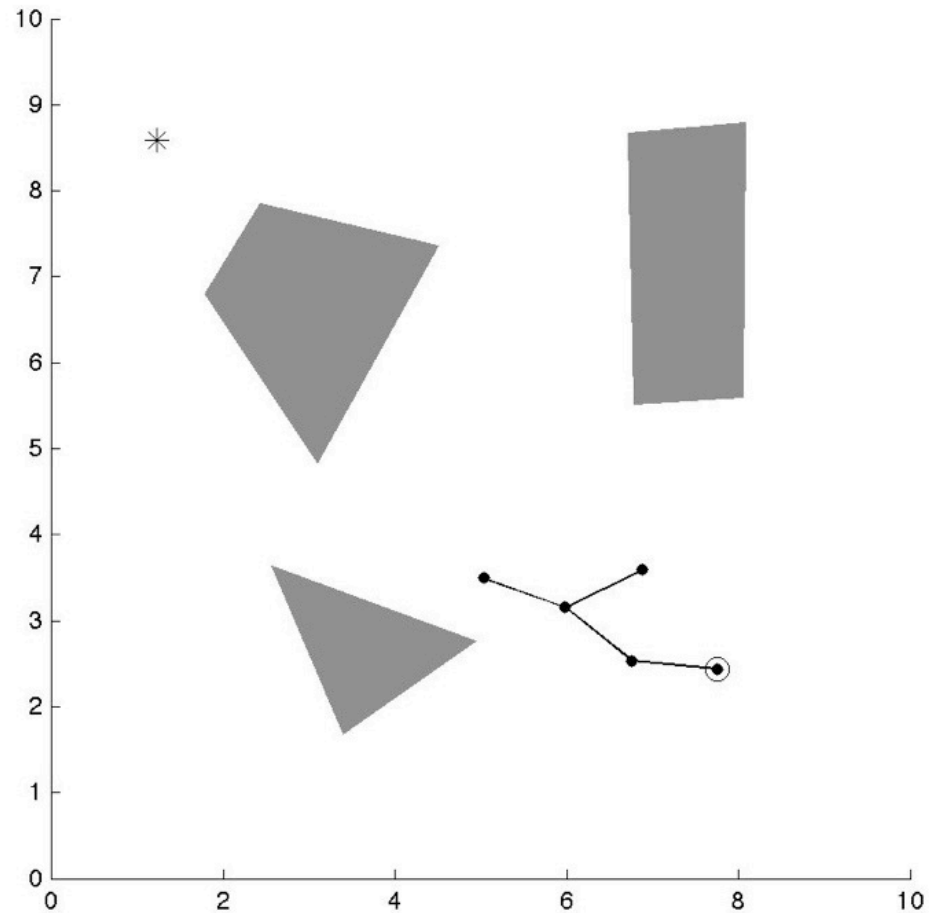
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

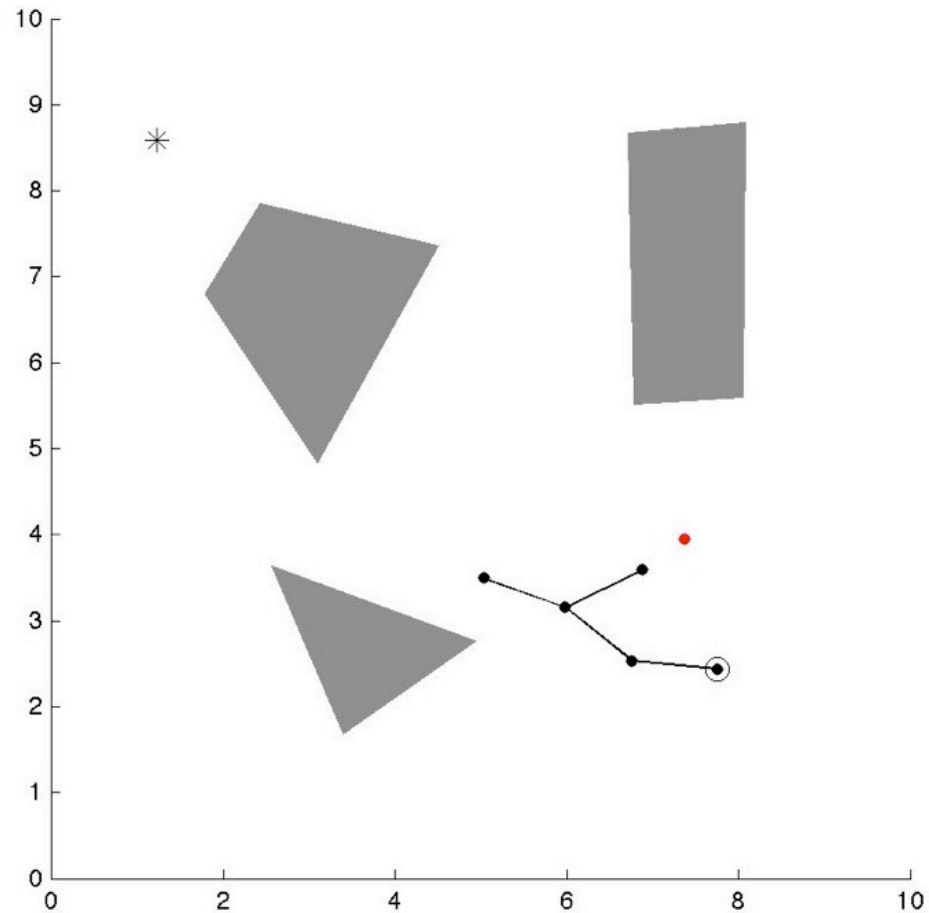
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

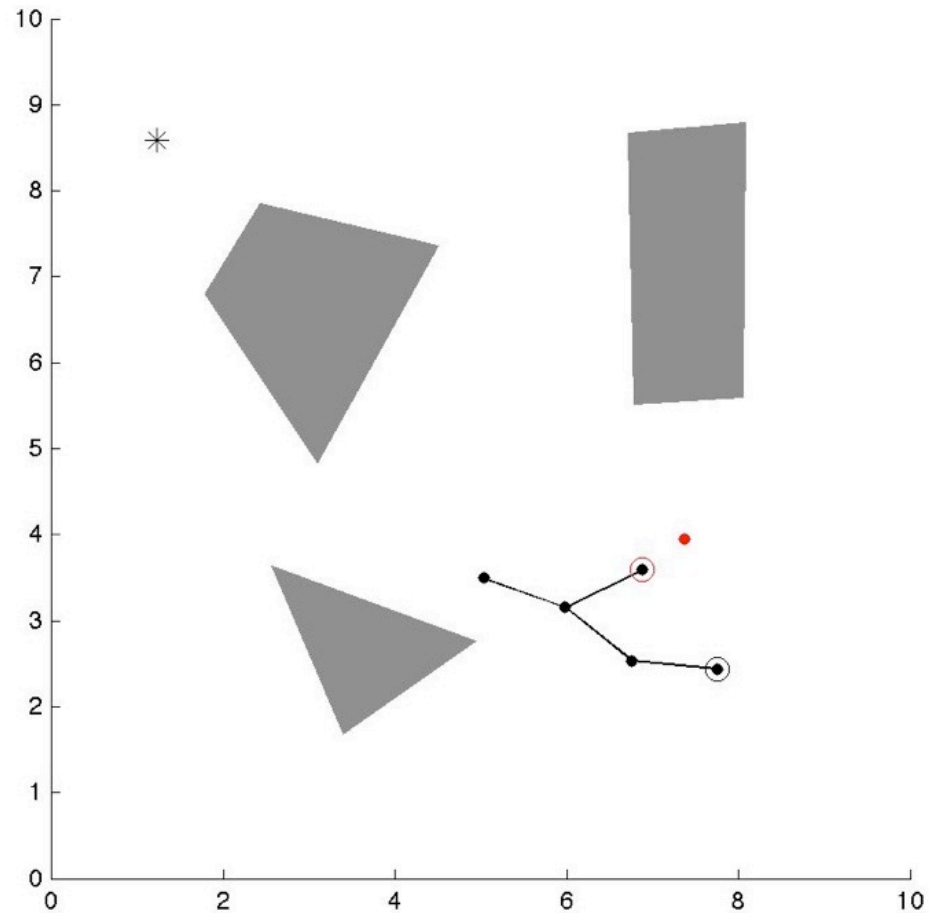
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

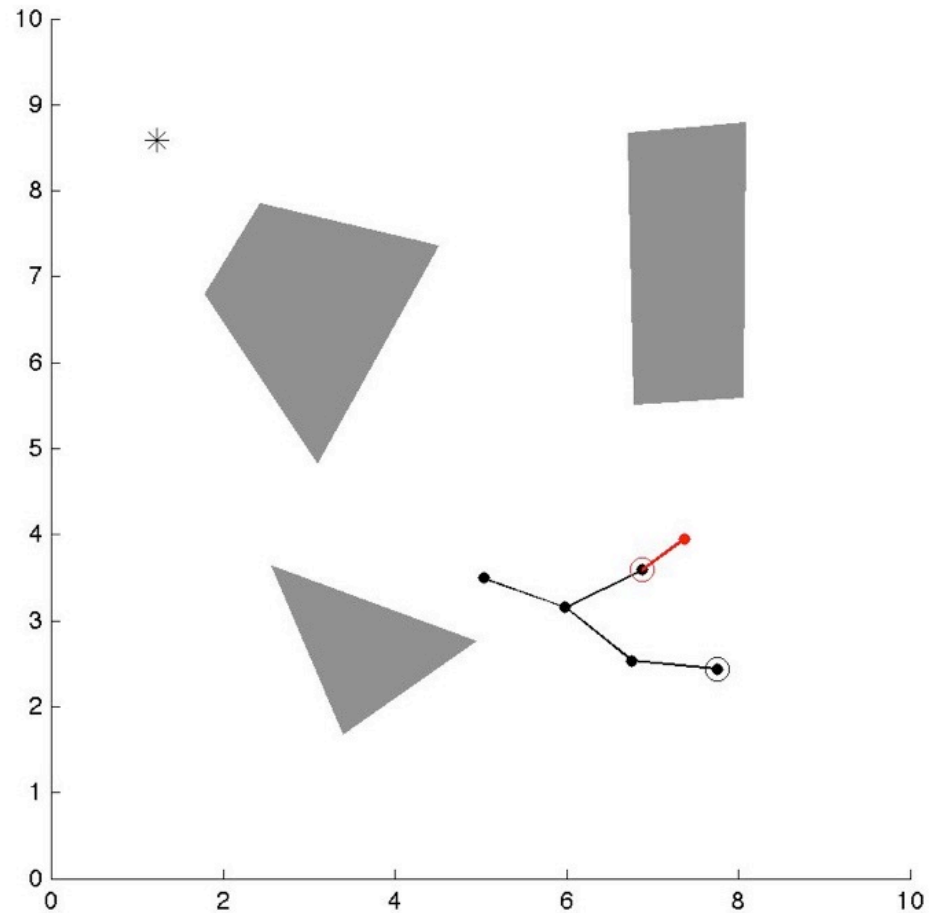
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

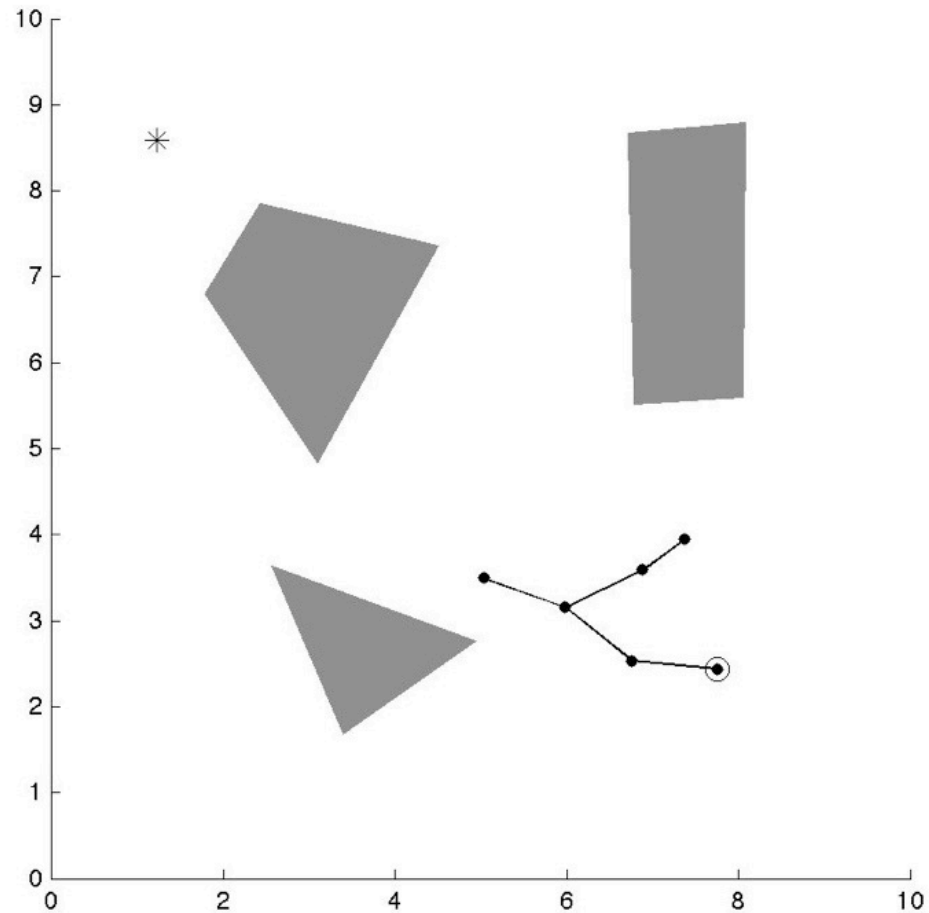
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

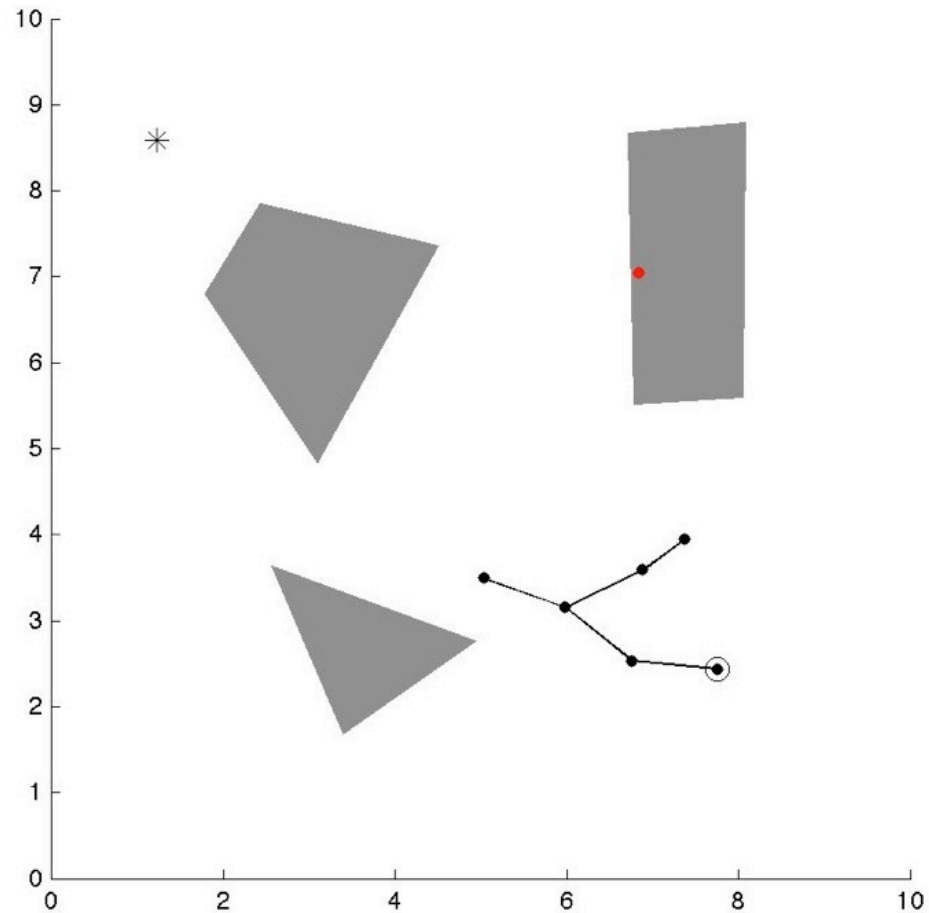
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

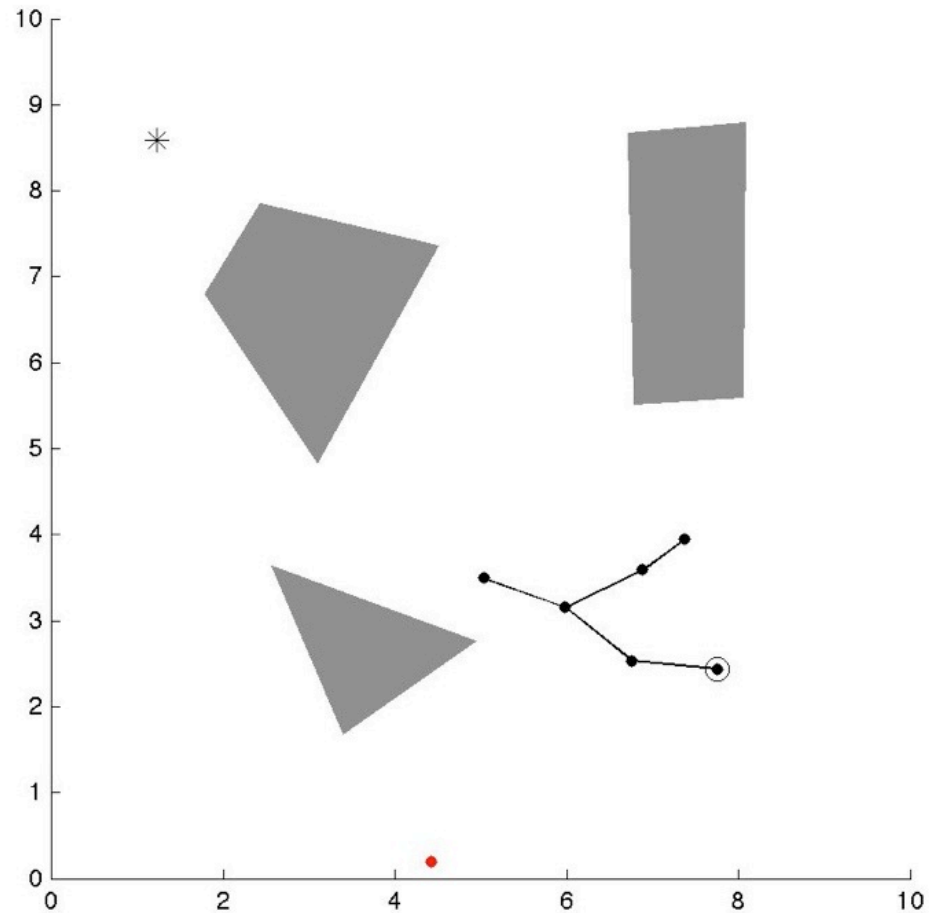
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

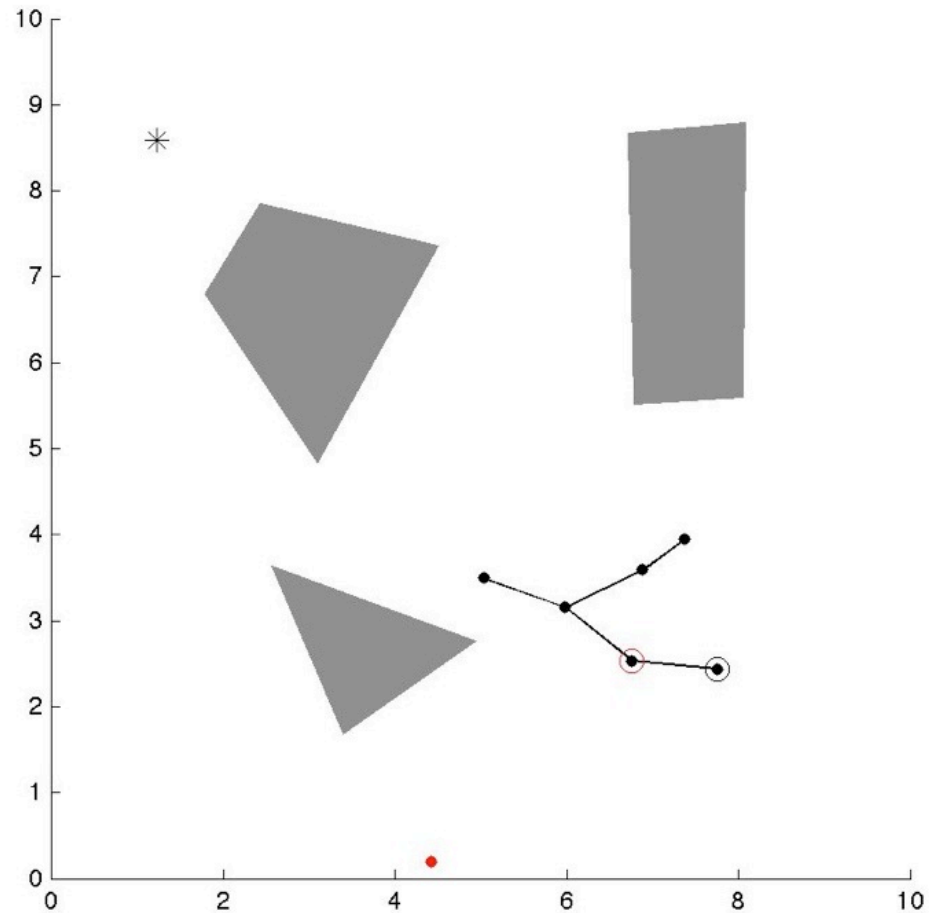
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

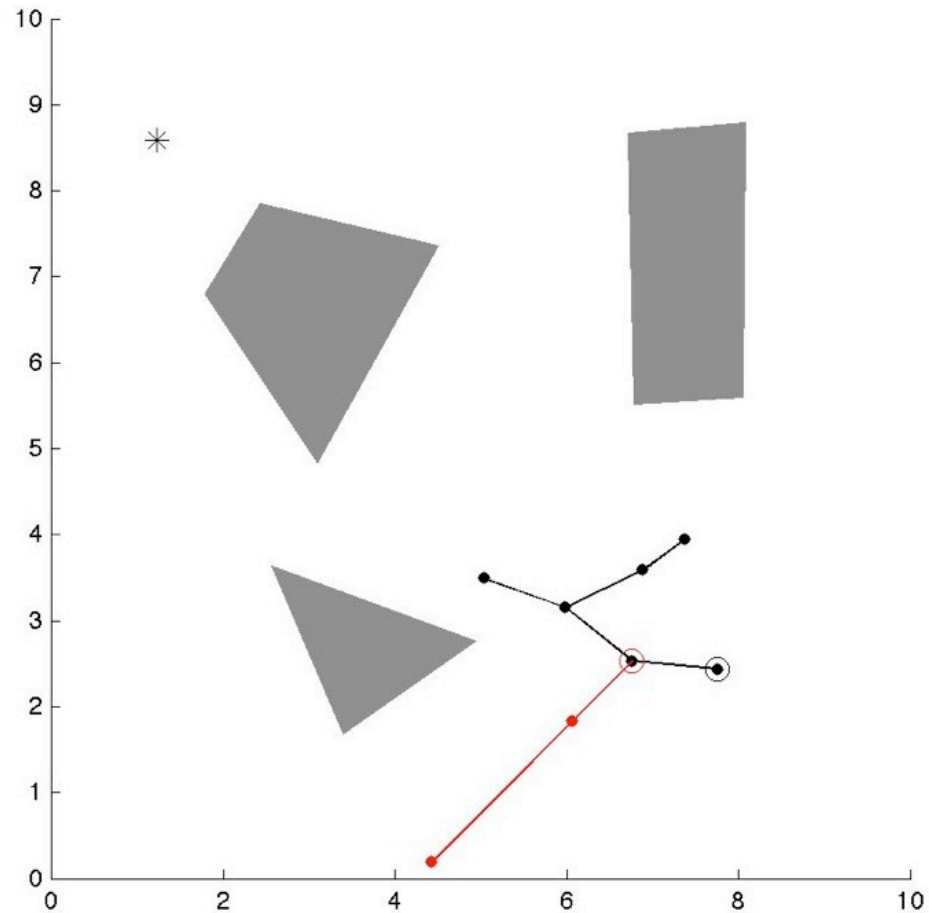
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

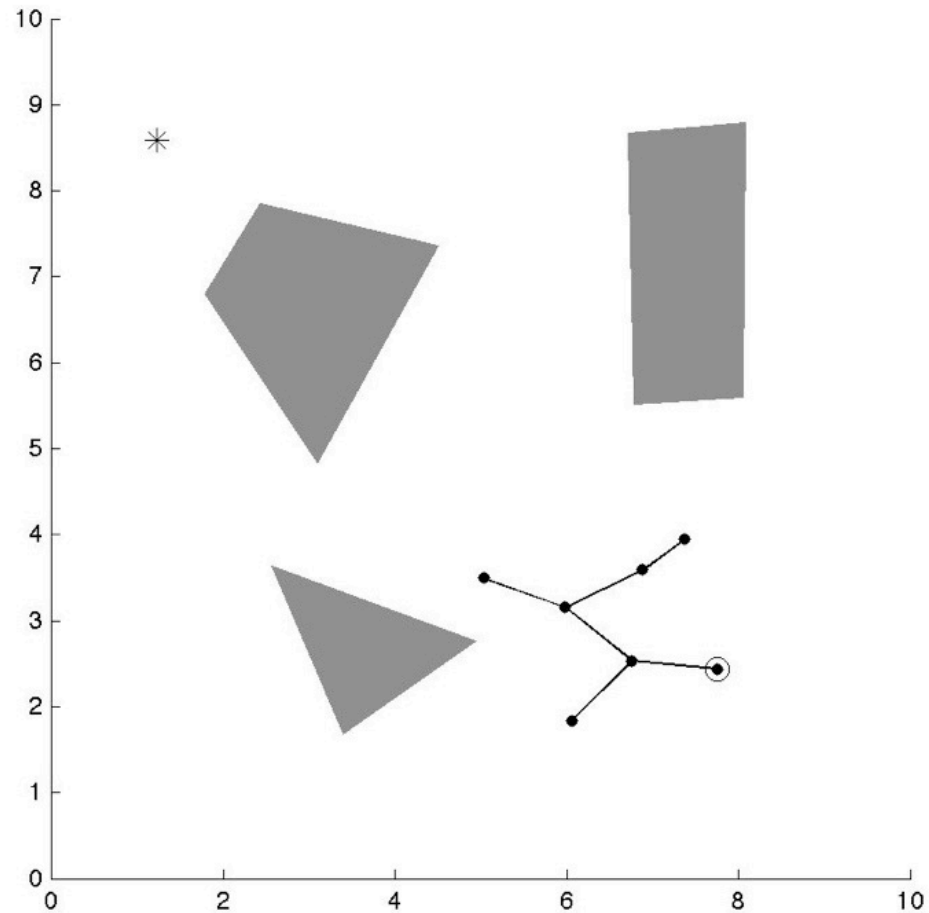
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

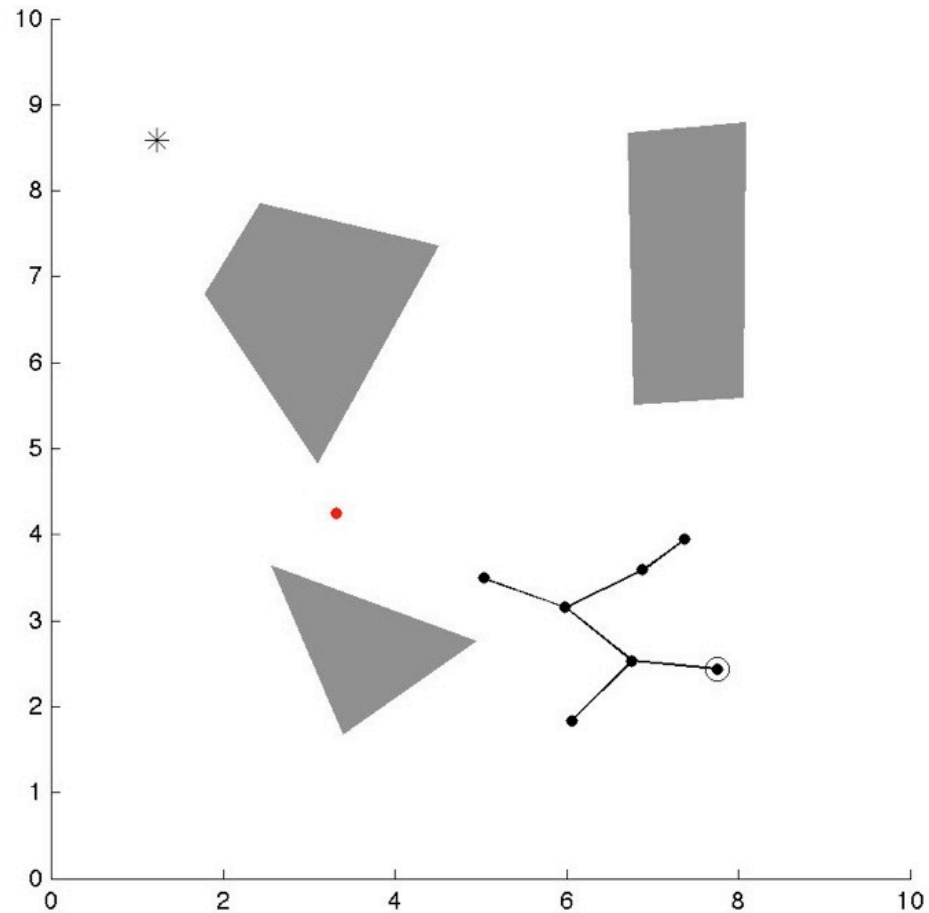
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

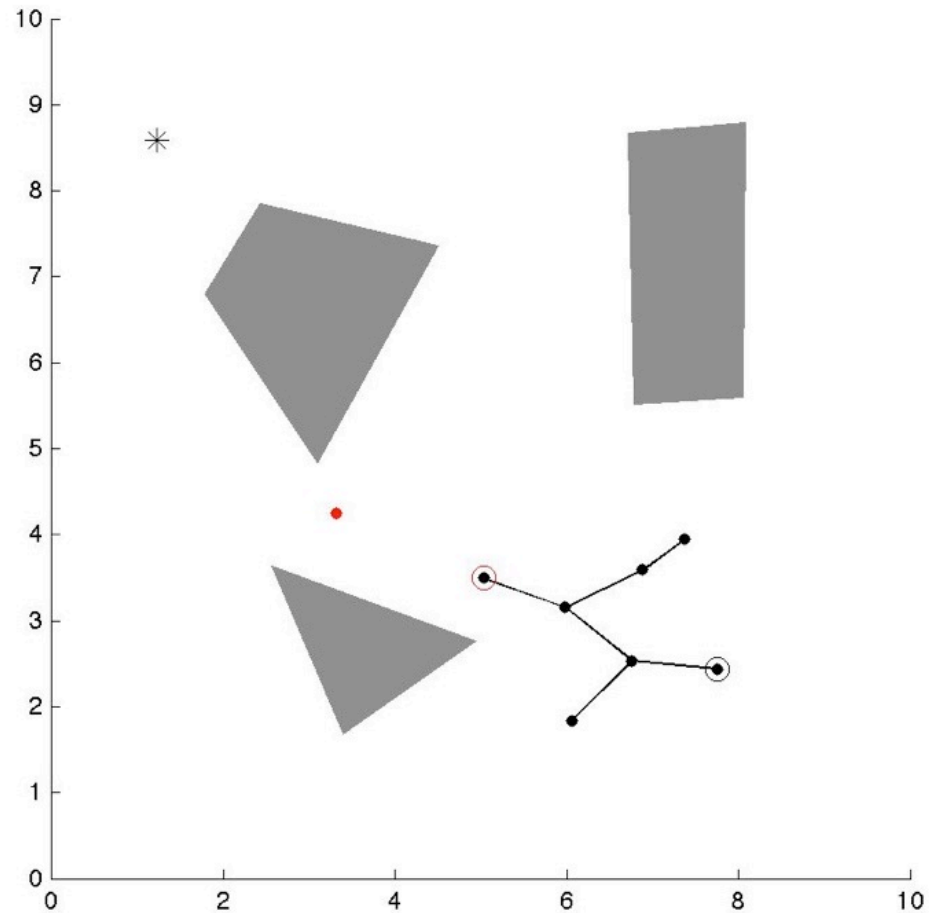
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

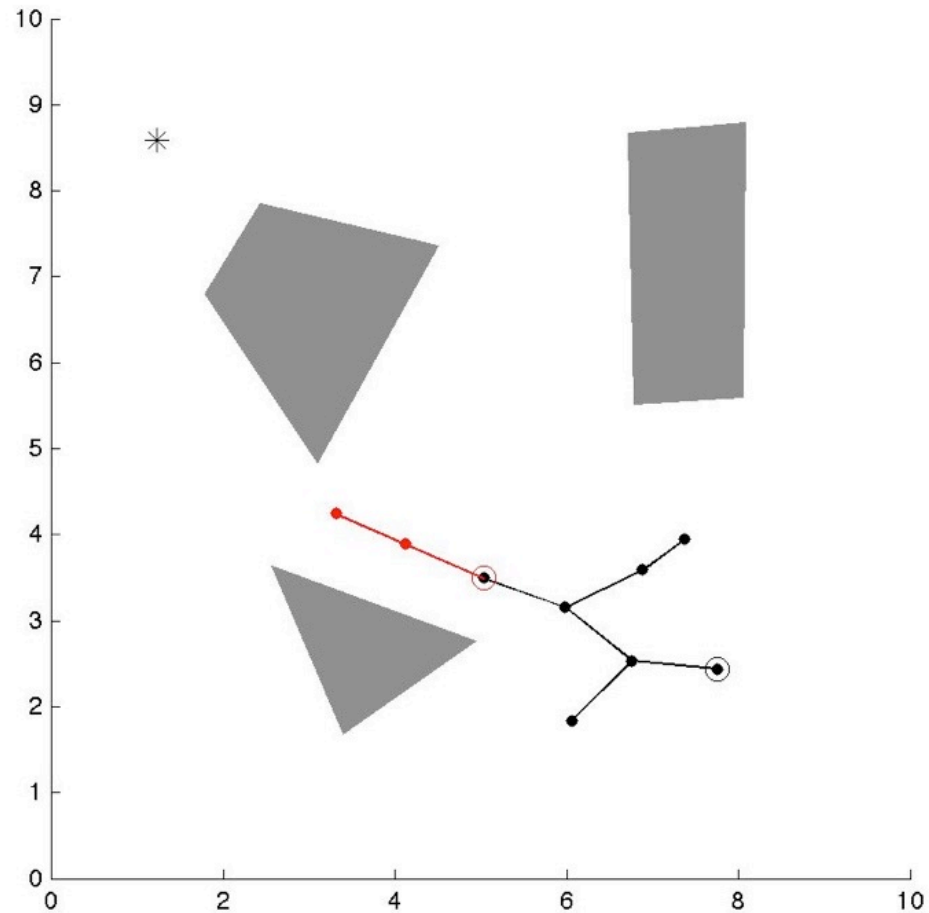
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

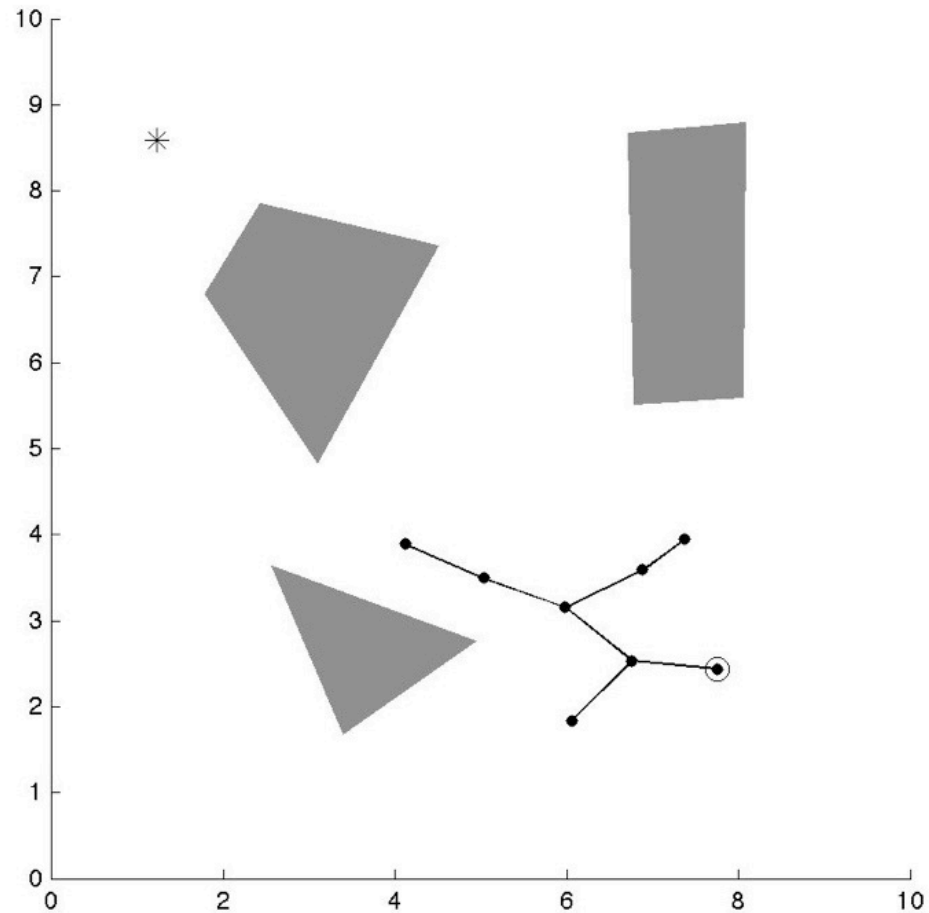
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

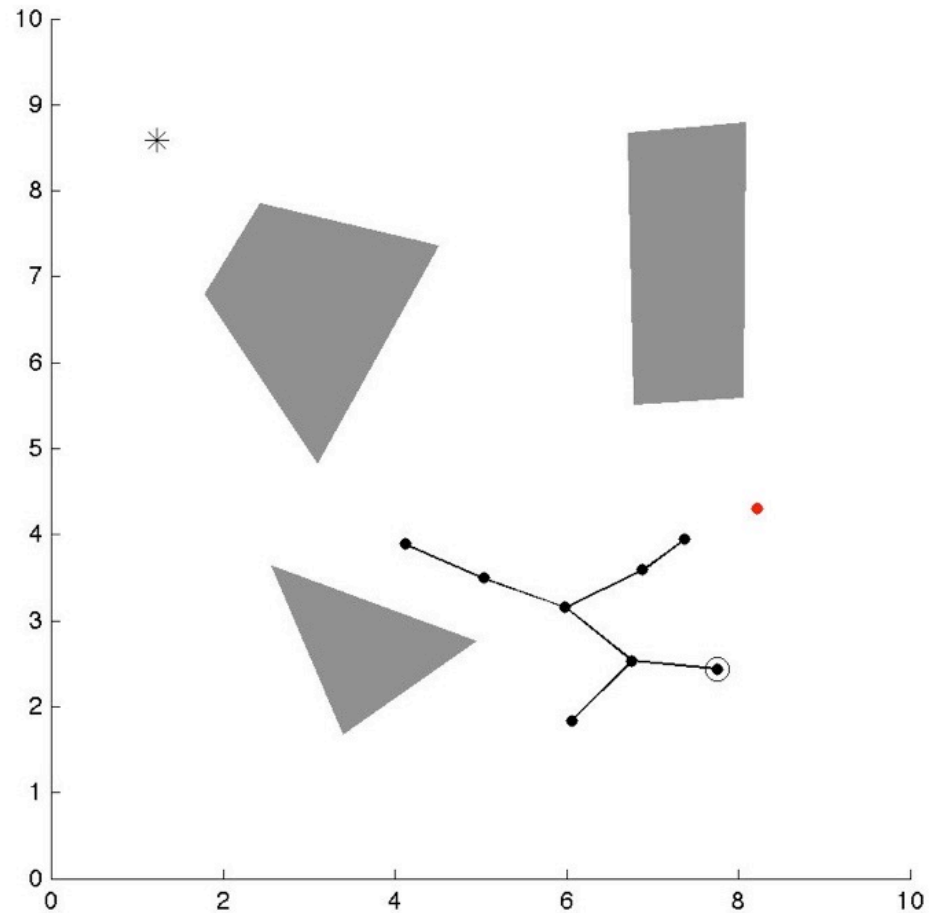
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

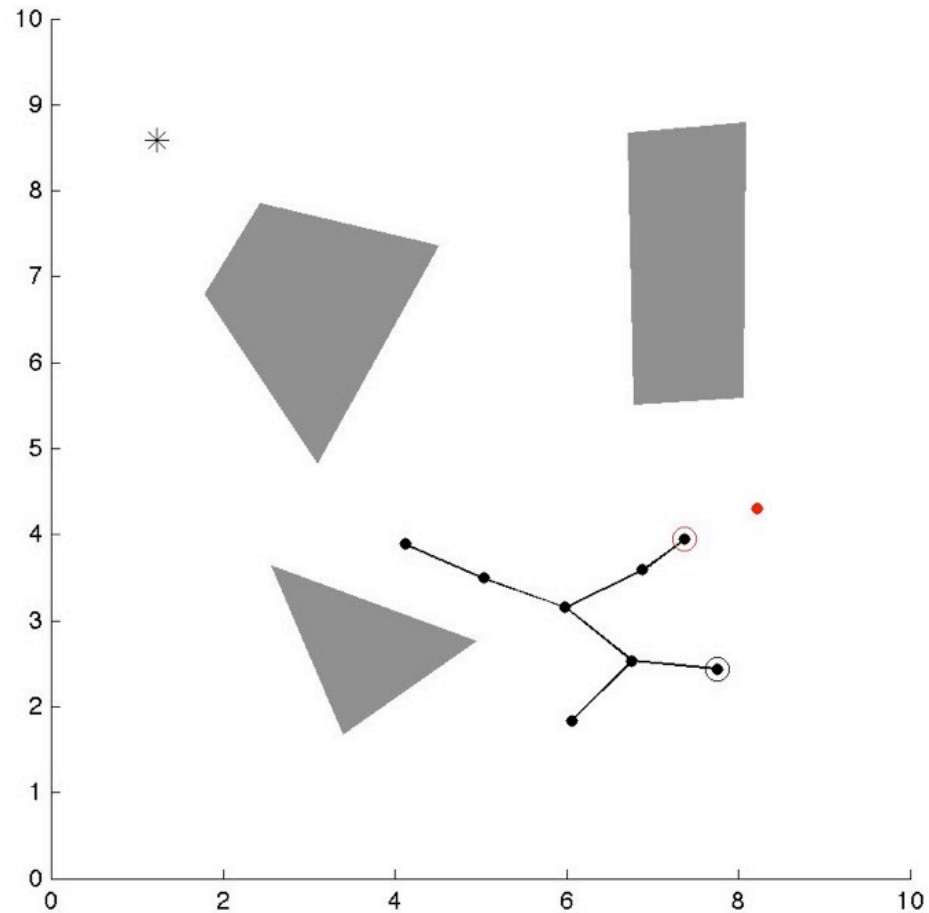
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

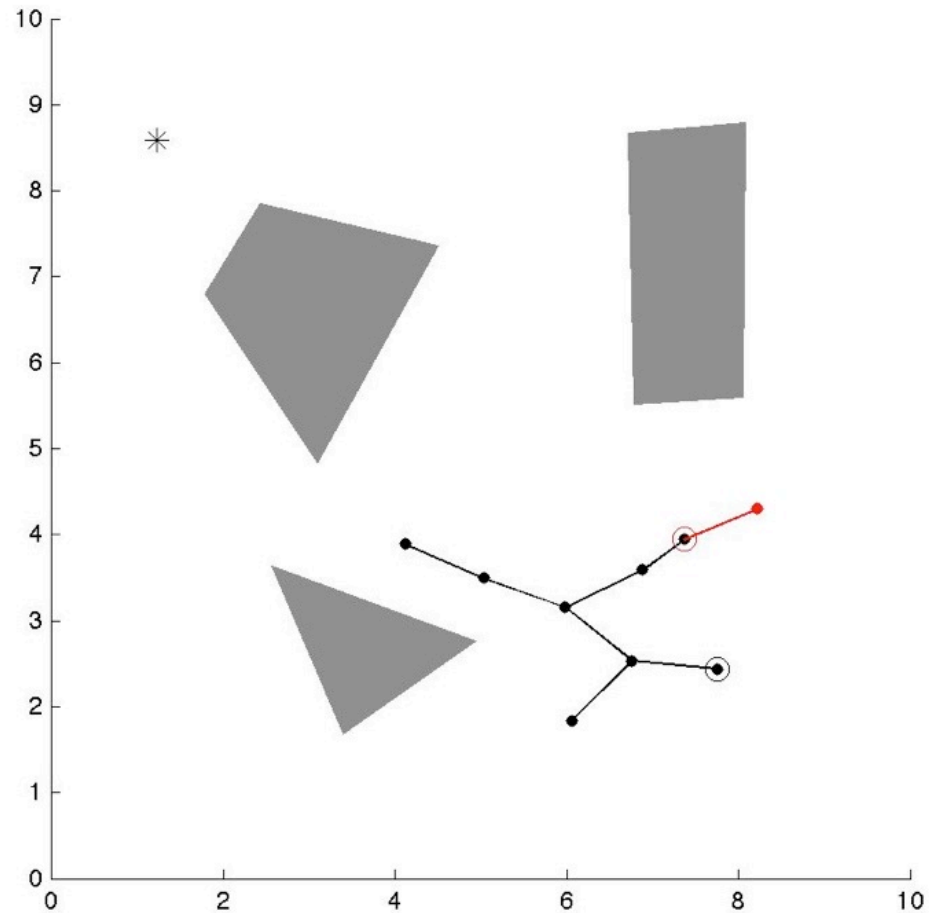
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$

for $i = 1$ to N

$G \leftarrow (V, E)$

$x_{rand} \leftarrow RandomSample()$

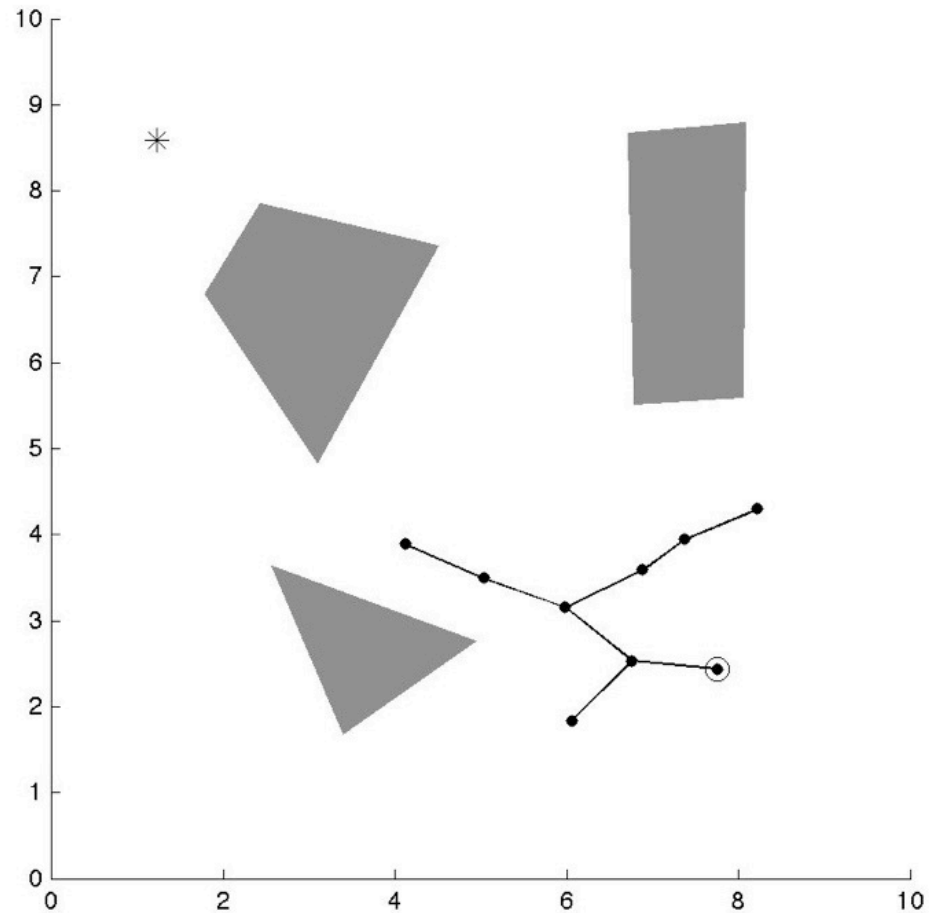
$x_{nearest} \leftarrow Nearest(G, x_{rand})$

$x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$

if $ObstacleFree(x_{nearest}, x_{new})$

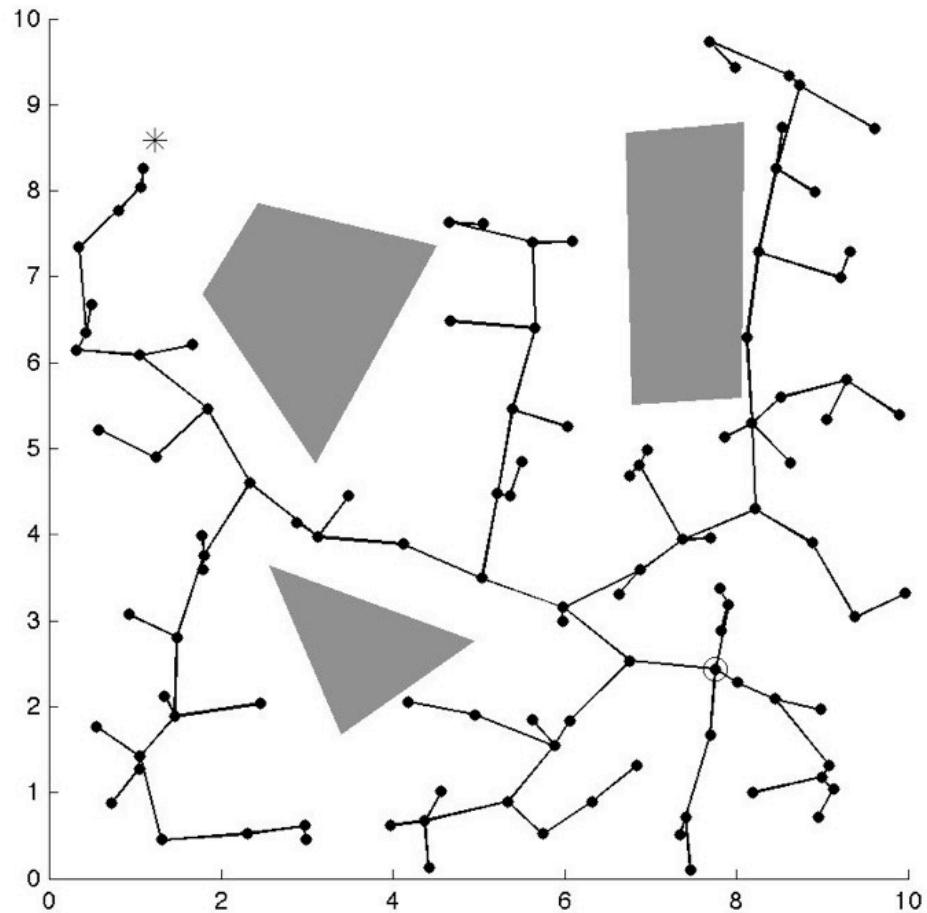
$V \leftarrow V \cup \{x_{new}\}$

$E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$



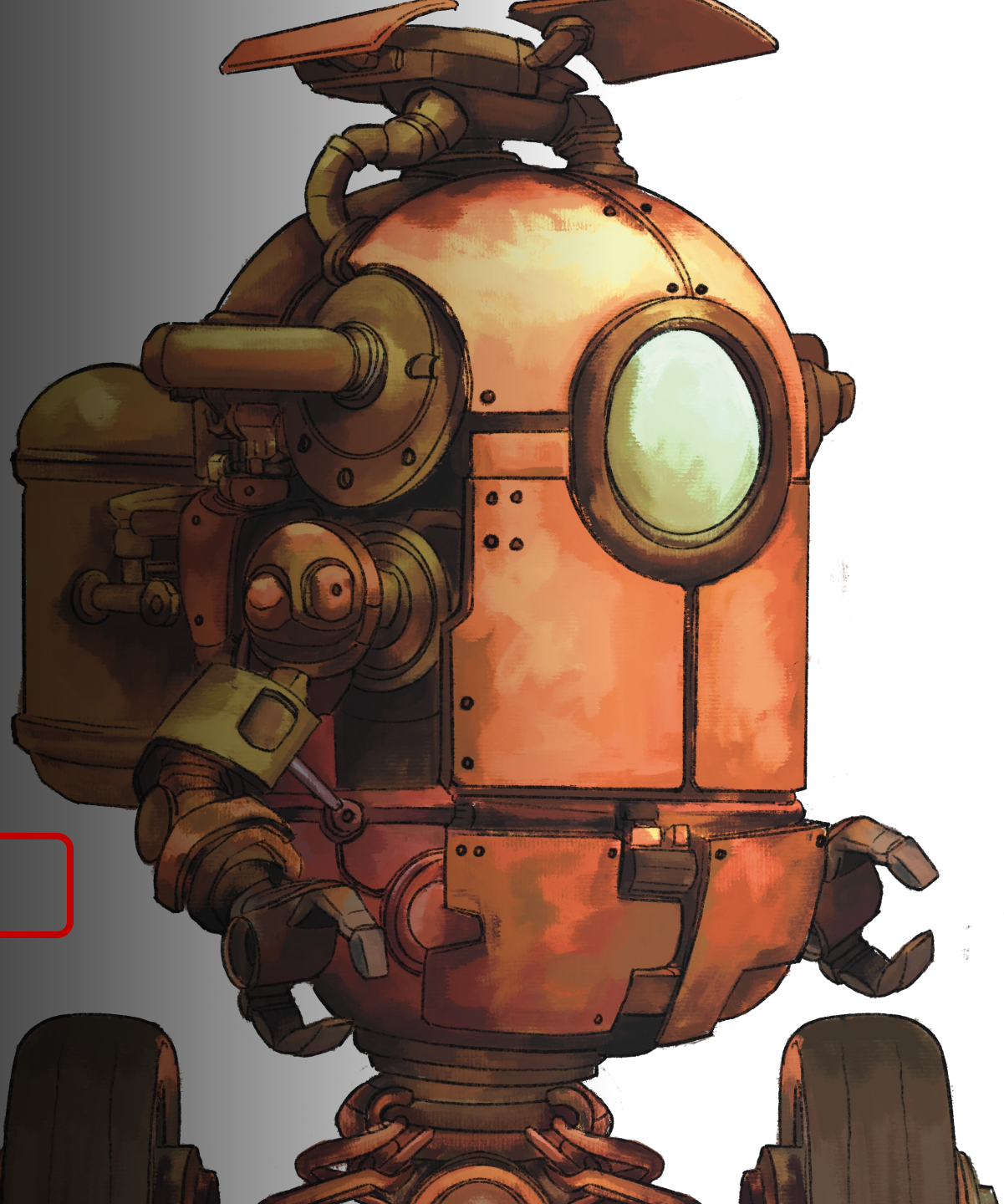
RRT

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$
for $i = 1$ to N
  $G \leftarrow (V, E)$
  $x_{rand} \leftarrow RandomSample()$
  $x_{nearest} \leftarrow Nearest(G, x_{rand})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
  if $ObstacleFree(x_{nearest}, x_{new})$
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$

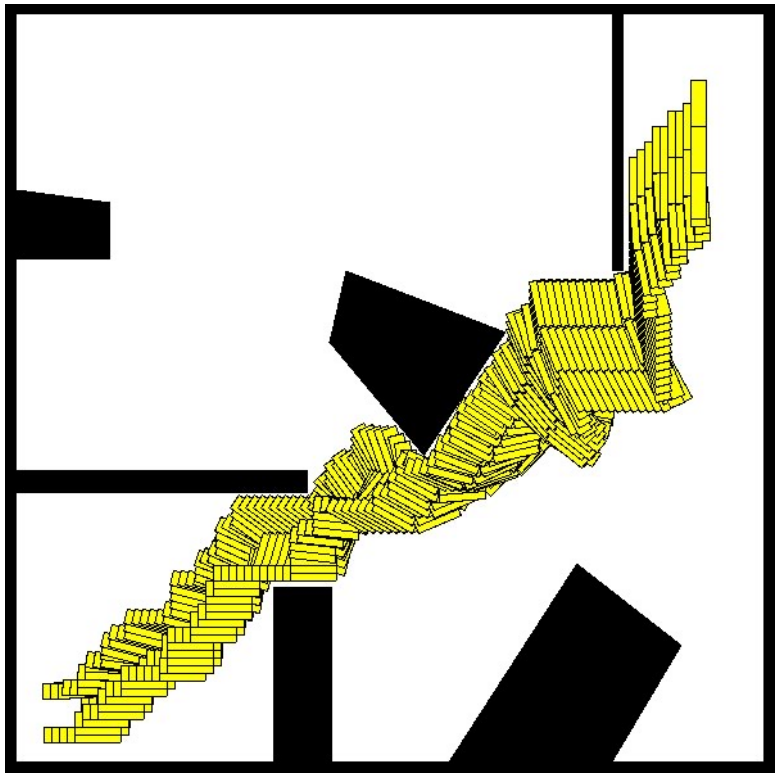


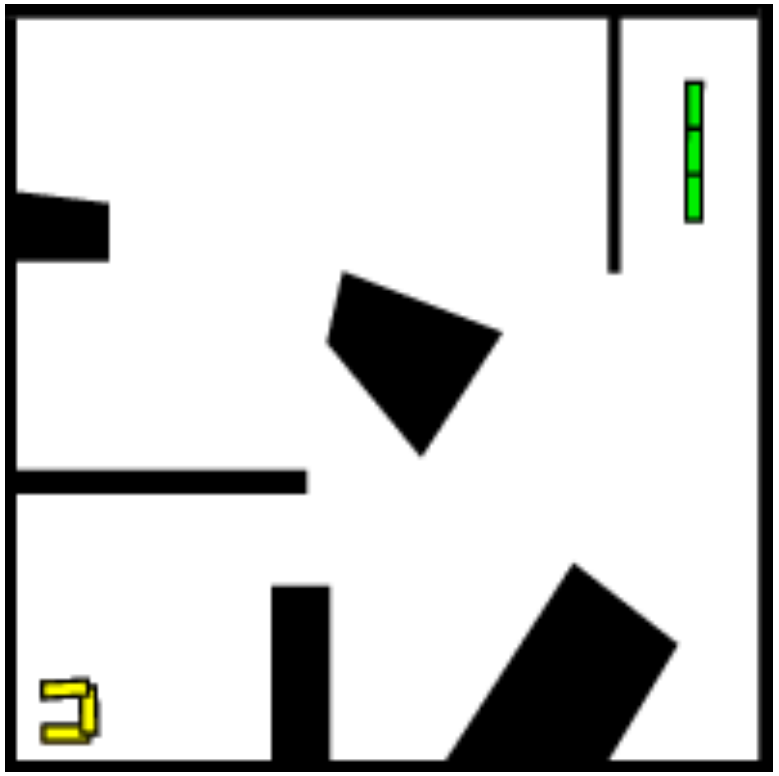
Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants

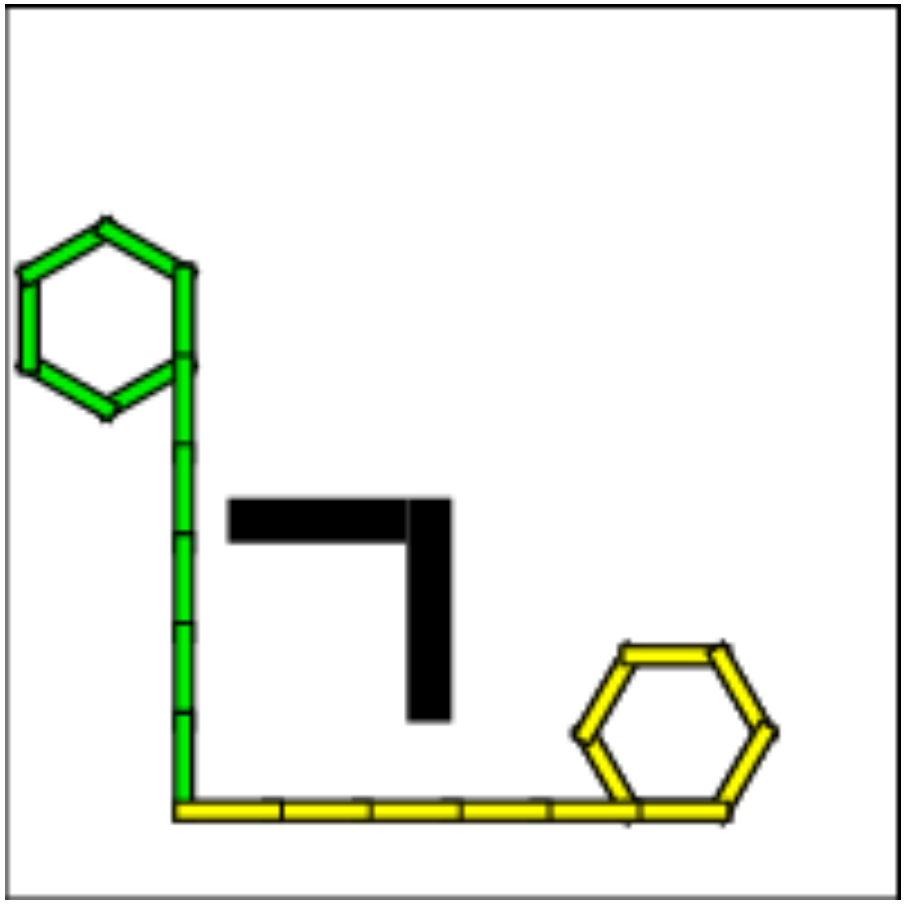


Articulated Robot

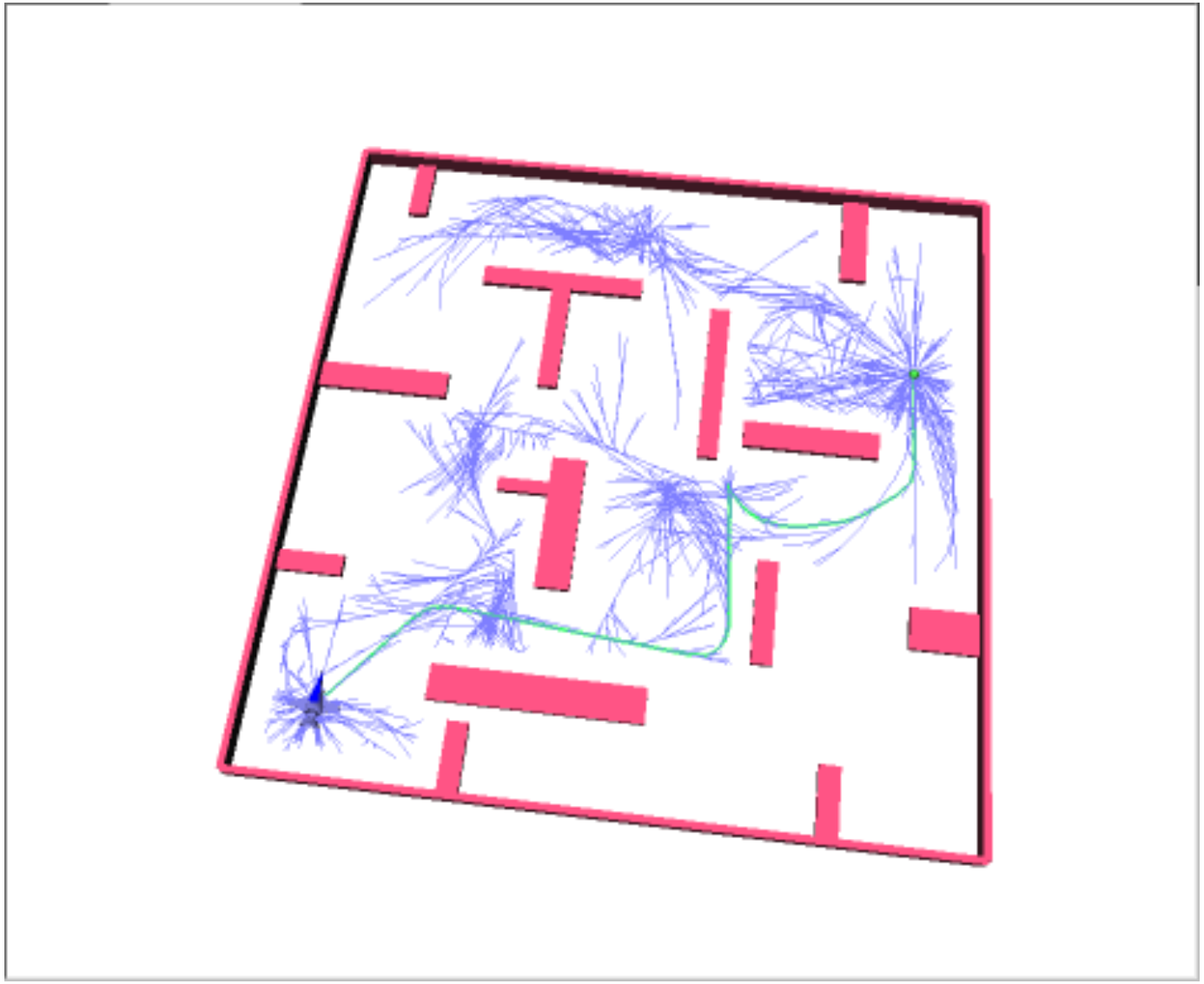




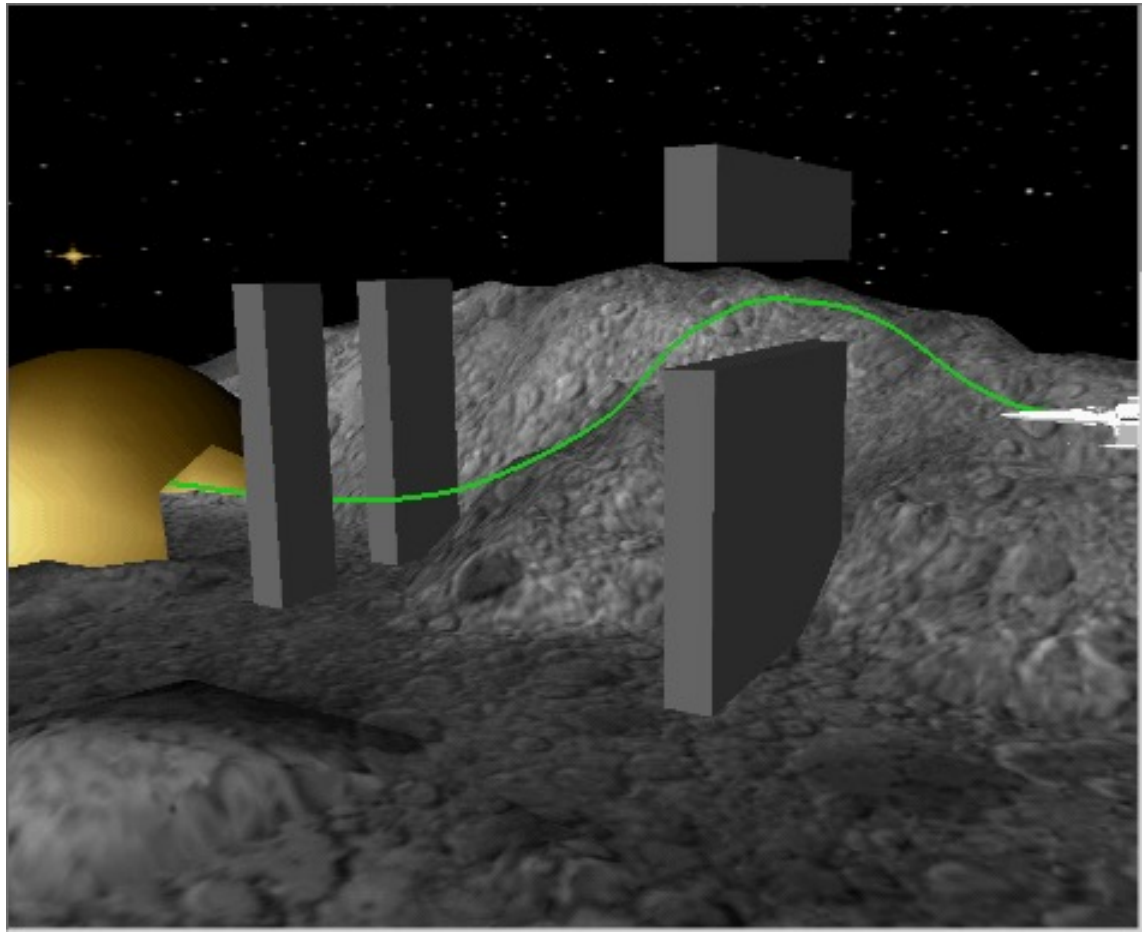
Highly Articulated Robot



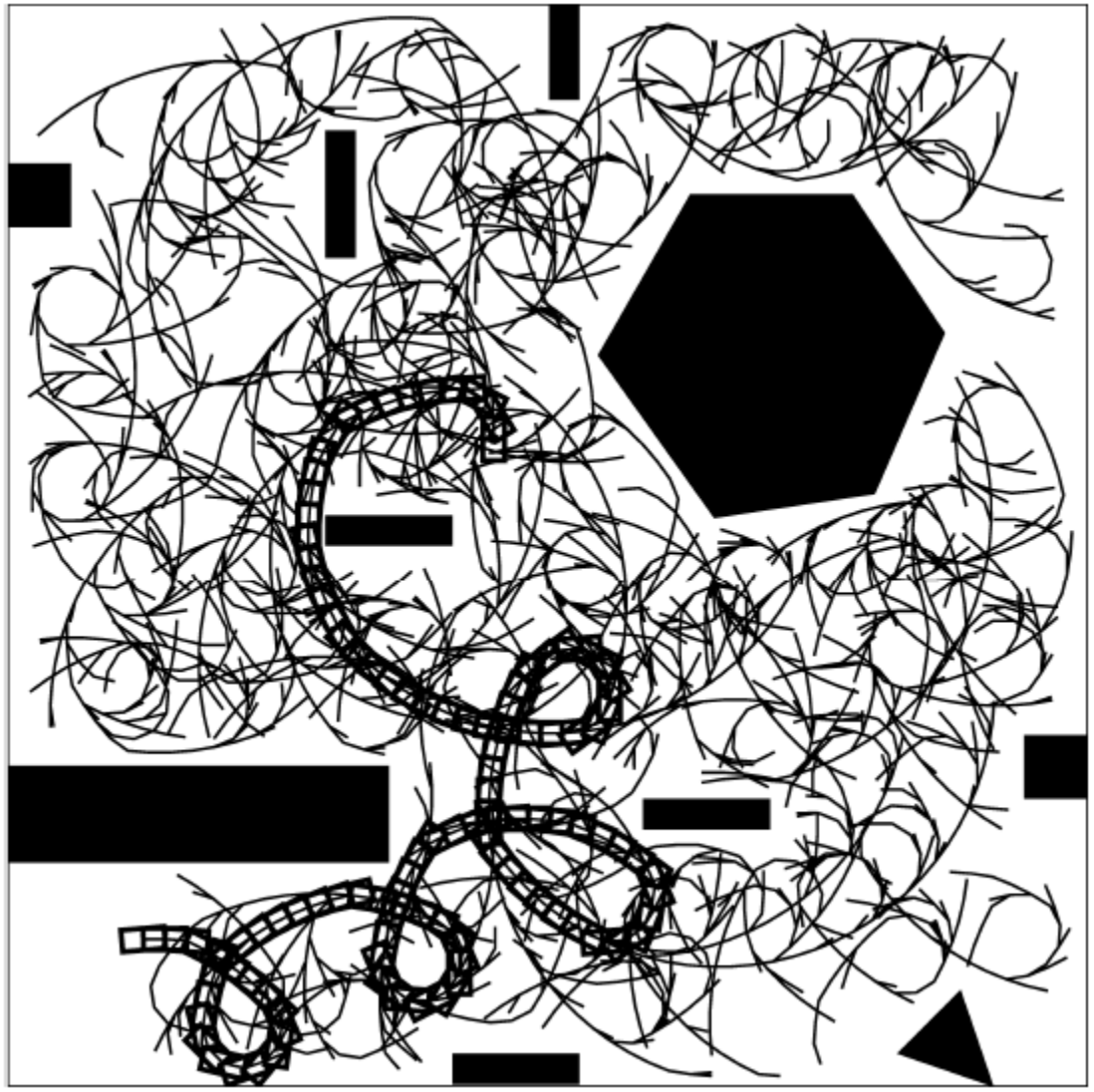
Hovercraft with 2 Thrusters



Out of This World Demo



Left-turn only forward car



Rapidly-Exploring Random Tree (RRT)

- Advantages of RRT: very fast, works well for dynamic environments
- Disadvantages: Not optimal
  - in fact, it has been proven by Karaman & Frazzoli that the probability of RRT converging to an optimal solution is 0

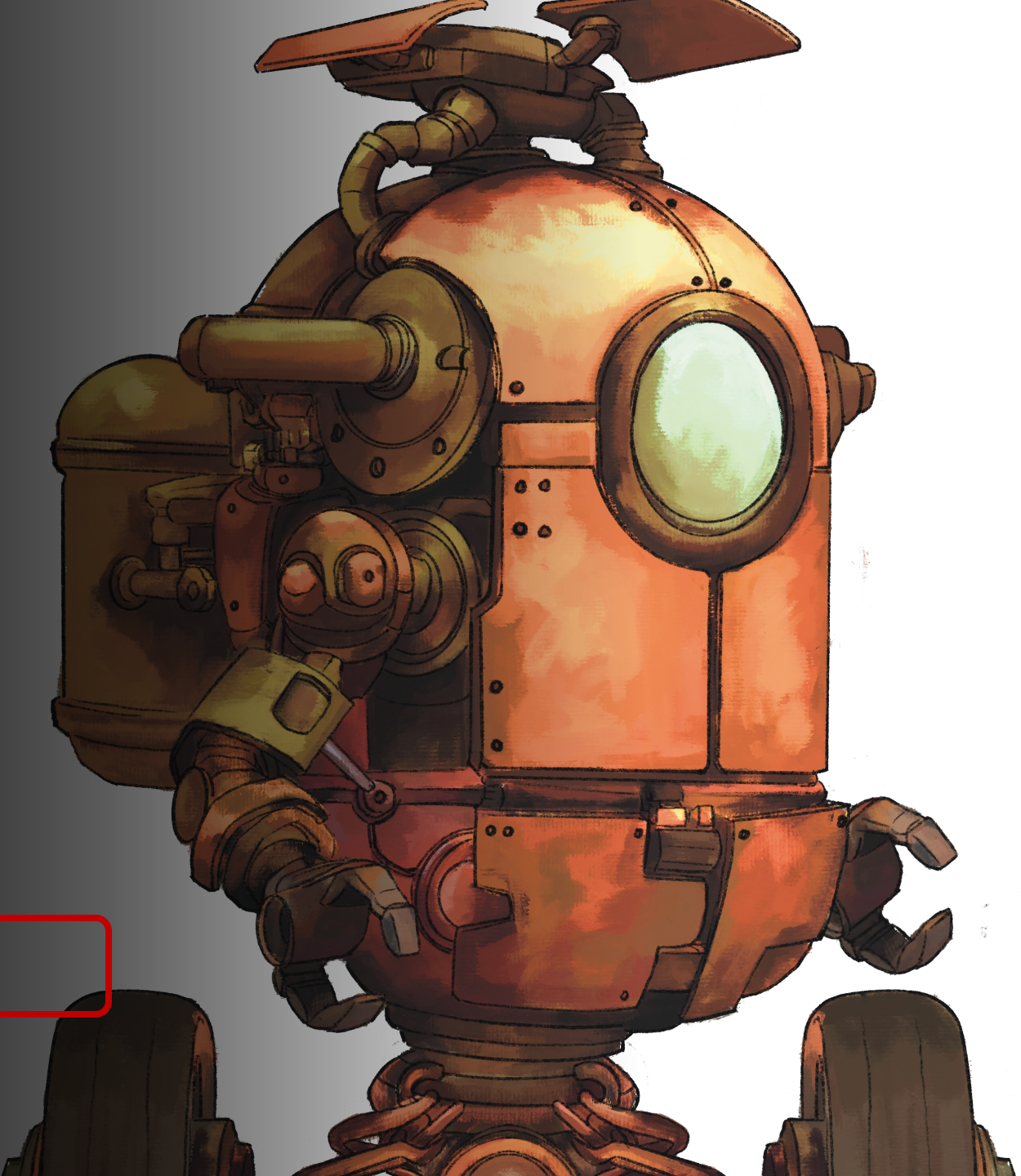
Summary

- Both RRT and PRM are examples of **sampling based algorithms** that are **probabilistically complete**

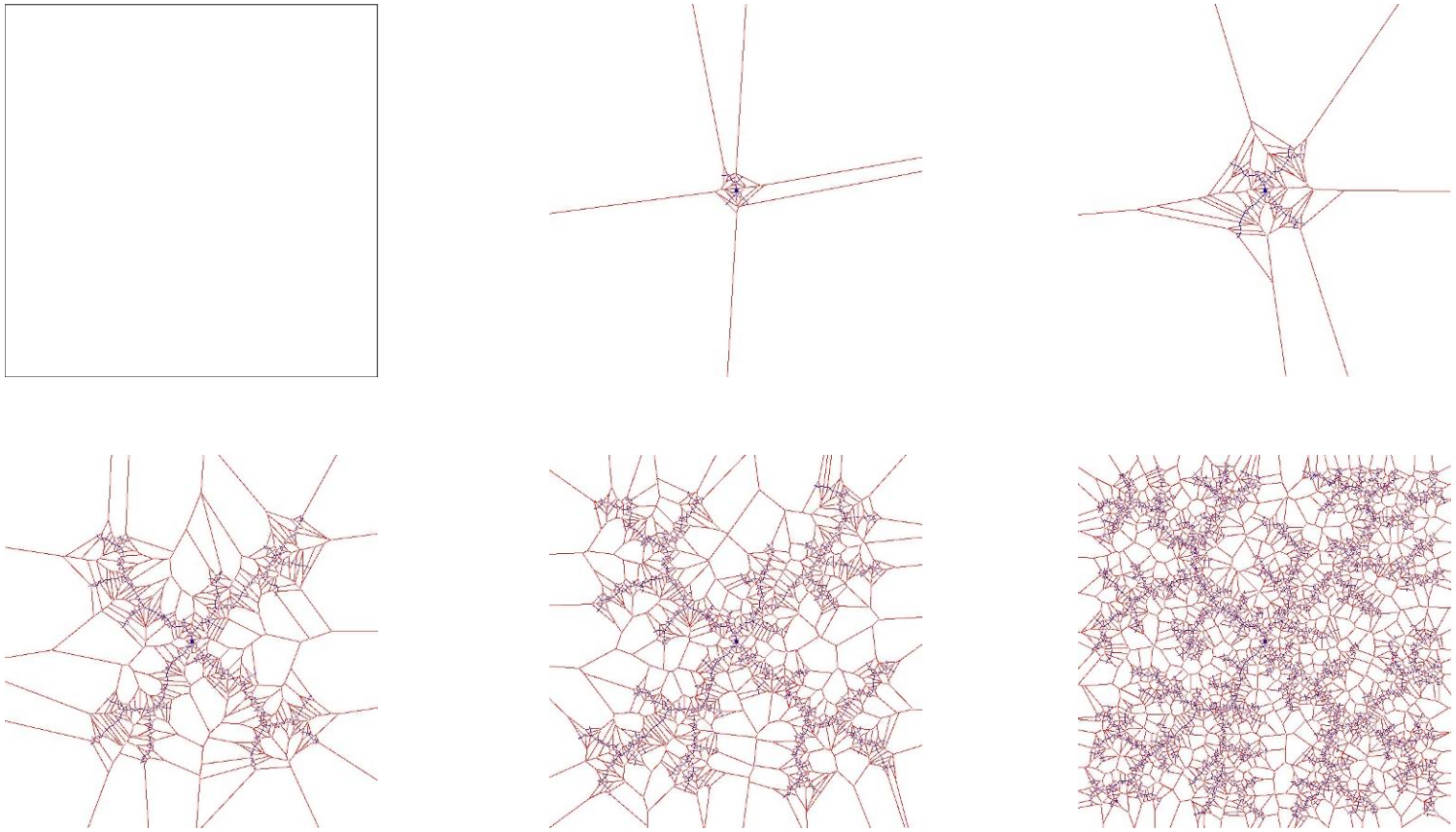
- Definition: A path planner is ***probabilistically complete*** if, given a solvable problem, the probability that the planner solves the problem goes to 1 as time goes to infinity.

Outline

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants

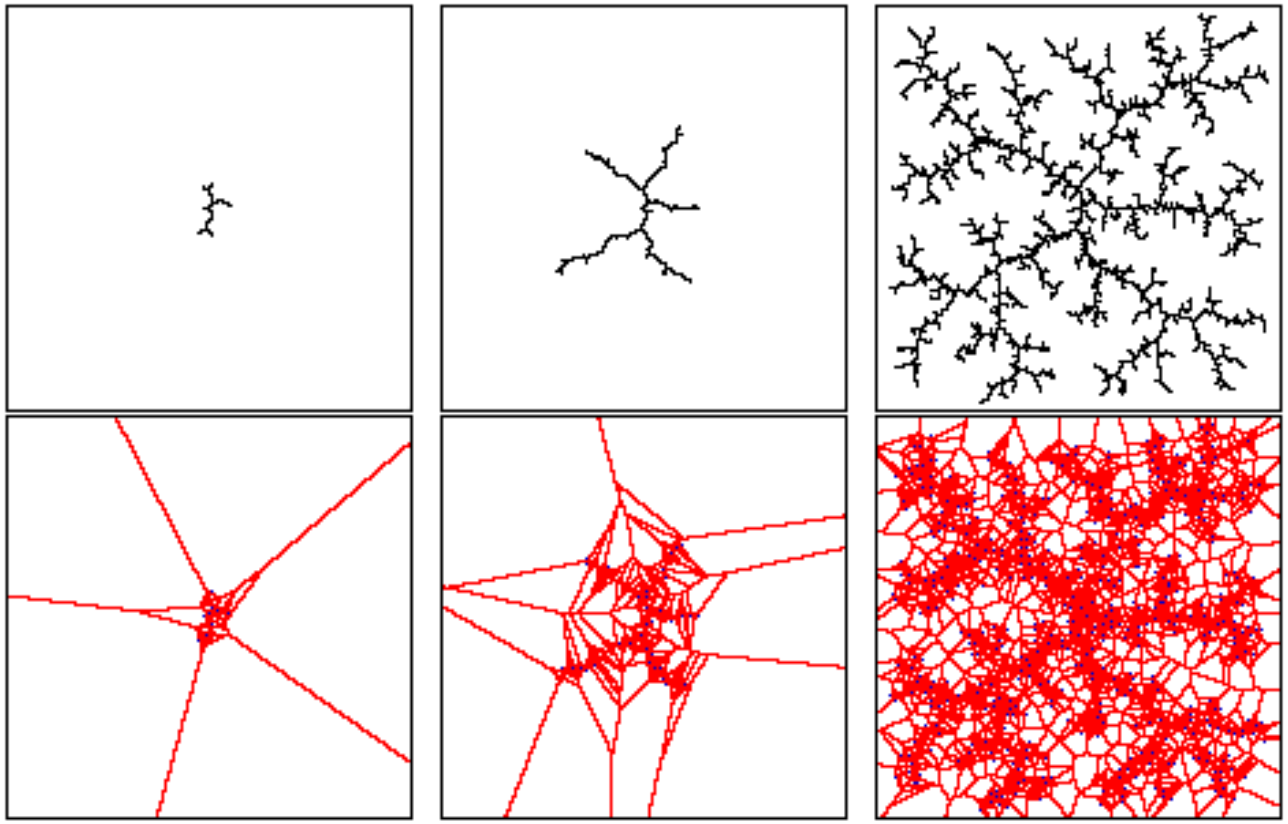


RRTs and
Bias toward large Voronoi regions



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

Why are RRT's rapidly exploring?



The probability of a node being selected for expansion (i.e. being a nearest neighbor to a new randomly picked point) is proportional to the area of its Voronoi region.

Biases

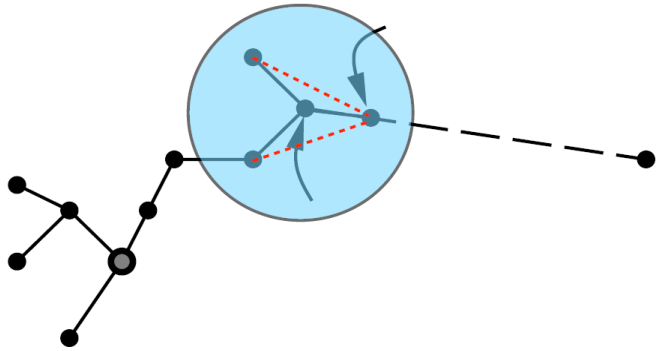
- Bias toward larger spaces
- Bias toward goal
  - When generating a random sample, with some probability pick the goal instead of a random node when expanding
  - This introduces another parameter
  - James' experience is that 5-10% is the right choice
  - If you do this 100%, then this is a RPP

RRT - Bias to Goal

$V \leftarrow \{x_{init}\}; \quad E \leftarrow \emptyset$
for $i = 1$ to N
  $G \leftarrow (V, E)$
  with probability p
    $x_{rand} \leftarrow RandomSample()$
  otherwise
    $x_{rand} \leftarrow x_{goal}$
  $x_{nearest} \leftarrow Nearest(G, x_{rand})$
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
  if $ObstacleFree(x_{nearest}, x_{new})$
    $V \leftarrow V \cup \{x_{new}\}$
    $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$

Variants of RRT

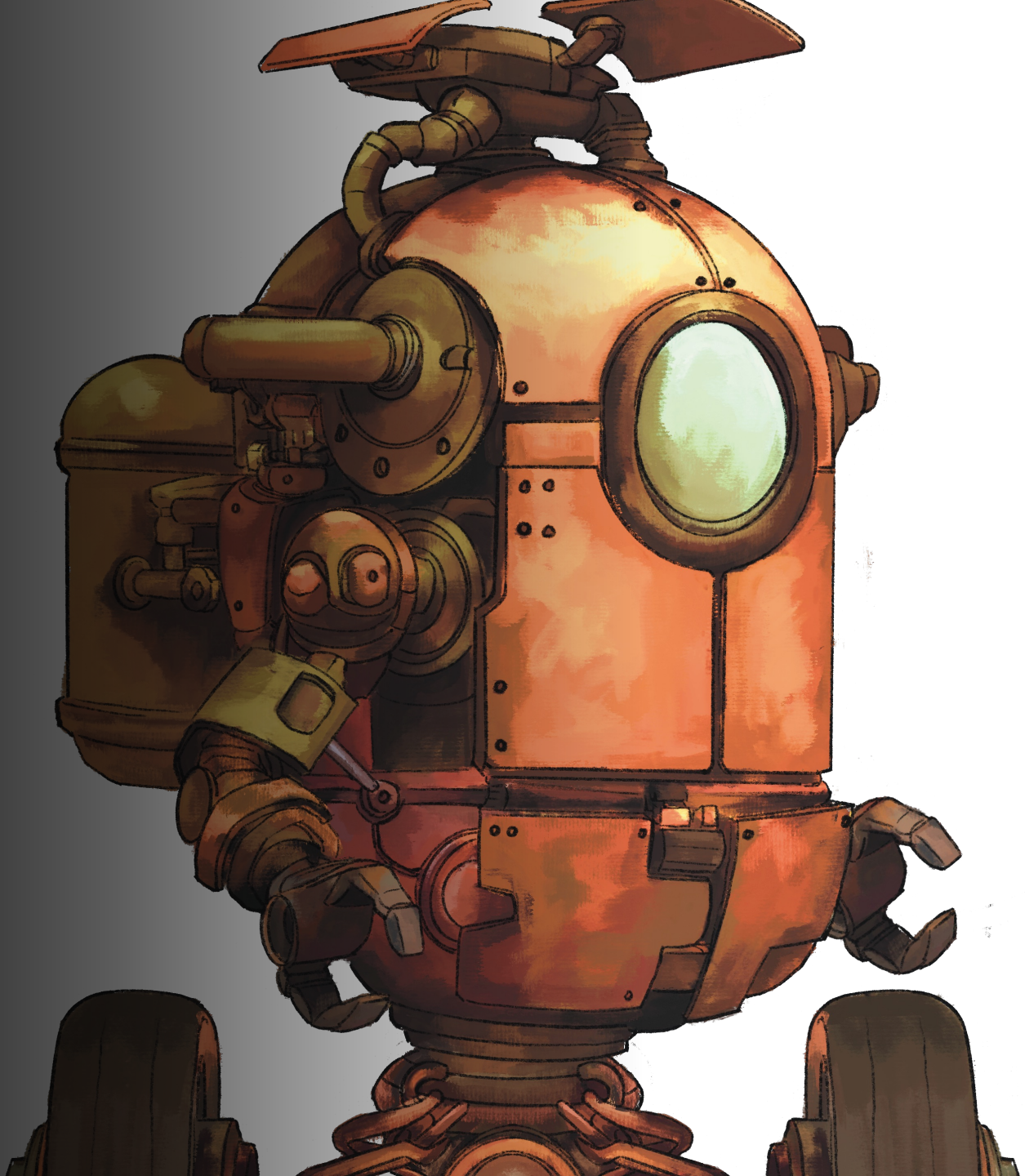
- There are (very) many…
- Rapidly-exploring Random Graph (RRG):
  - Connect all vertices within neighboring region, forming a graph



- RRT*:
  - a variant of RRG that essentially “rewires" the tree as better paths are discovered.

Summary

1. Recap
2. Path Planning
3. Probabilistic Roadmaps
4. Path Refinement
5. Rapidly-Exploring Random Trees
6. Planning Demonstrations
7. RRT Bias and Variants



Links to Further Reading

- Steve LaValle's online book:
  "Planning Algorithms" *(chapters 5 & 14)*
  http://planning.cs.uiuc.edu/

- The RRT page:
  http://msl.cs.uiuc.edu/rrt/

- Motion Planning Benchmarks
  Parasol Group, Texas A&M
  http://parasol.tamu.edu/groups/amatogroup/benchmarks/mp/