Sampling-Based Motion Planning

CMPT 882

Mar. 6

Outline

- Configuration space
- Probabilistic road maps (PRM)
  - PRM*
- Rapidly-exploring random trees (RRT)
  - RRT*
- Robust real-time planning (FaSTrack)

Configuration Space (C-Space)

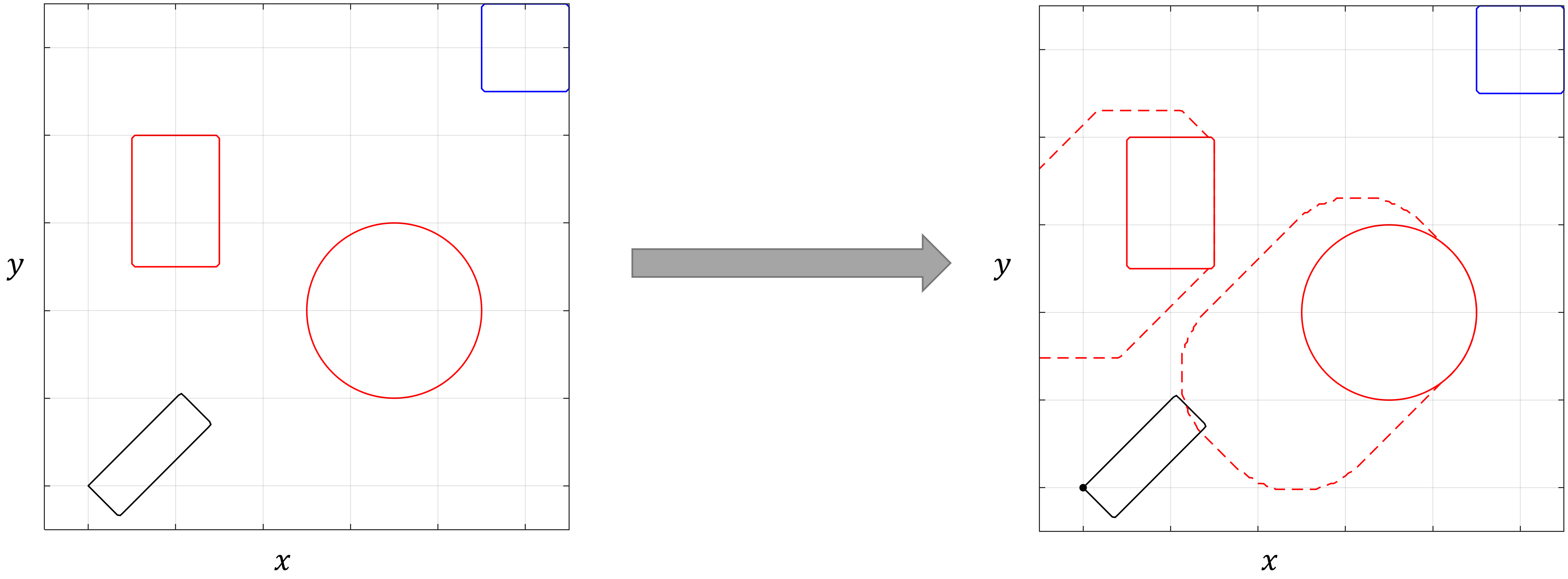
- Similar to state space, but considers reachability
  - Usually state space does not consider the set of states that a system can reach
  - Configuration space is the subset of the state space reachable by the system

  - Example: mechanical joints

- Rigid bodies in 2D:
  - 2D position and one rotation angle
- Rigid bodies in 3D:
  - 3D position and three rotation angles
- Connected rigid bodies
  - Concatenate positions and angles (but not every position and angle is reachable)

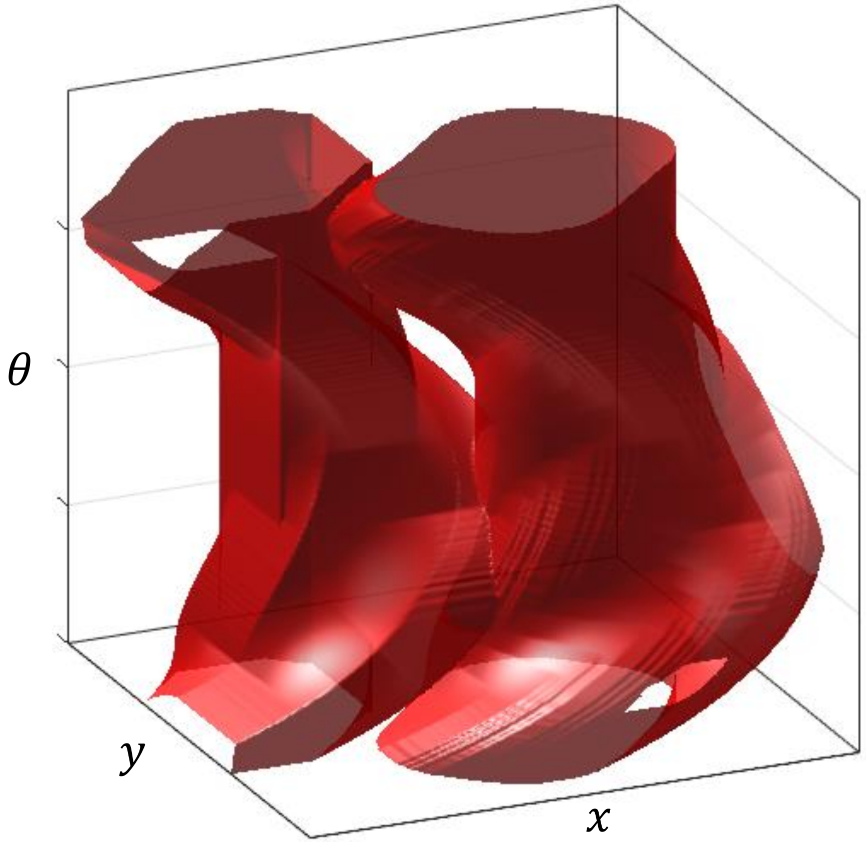
Planning in C-Space

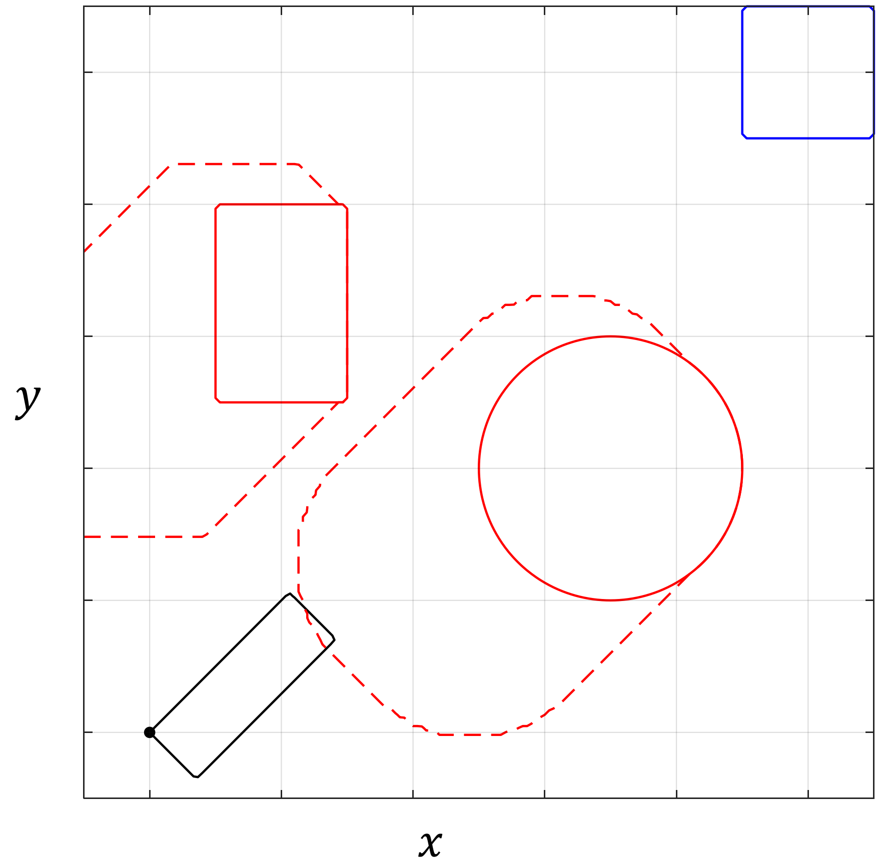
- Reduce objects to a point, and augment obstacles
  - Example: Sliding rectangular block to goal



Planning in C-Space

- Reduce objects to a point, and augment obstacles
  - Example: Sliding **and rotating** rectangular block to goal





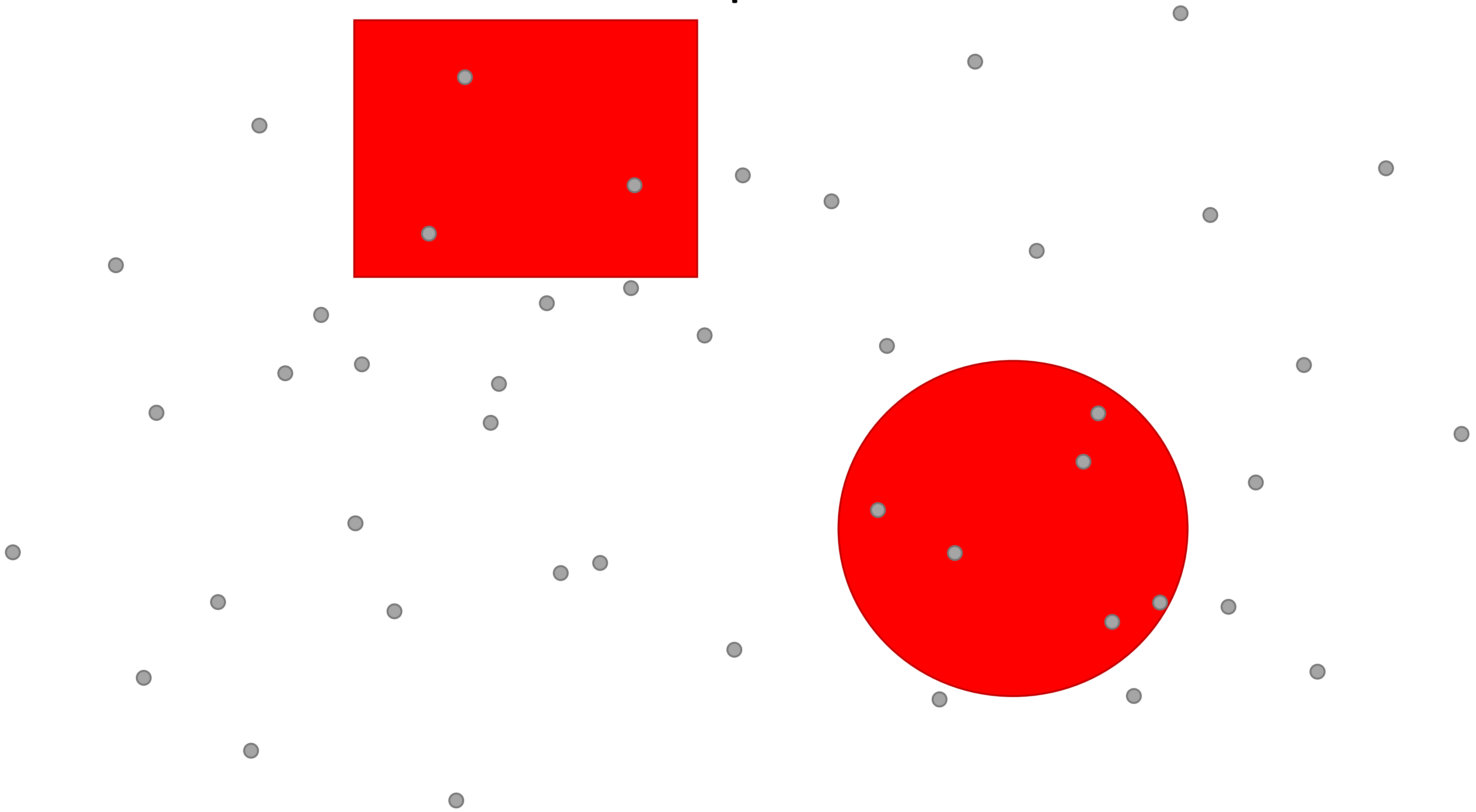
Probabilistic Road Map

- Draw N samples
  - Keep points outside of obstacles
- Choose a disk radius

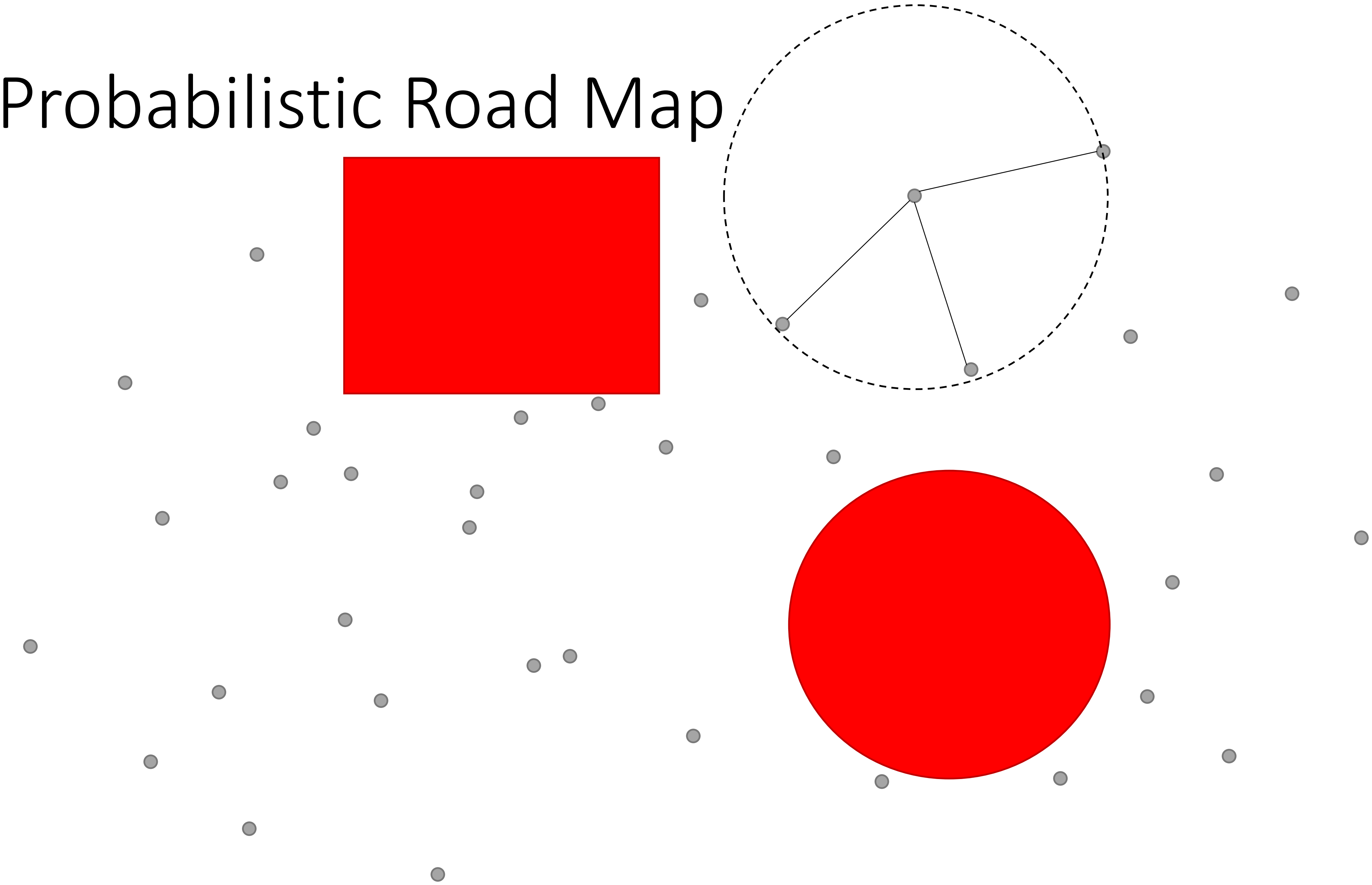
- For each kept point, draw edge between it and all other points within the disk
  - Keep edges that are collision free (expensive)

- Use graph search algorithm (e.g. A*, Dijkstras) on the resulting graph to find a path

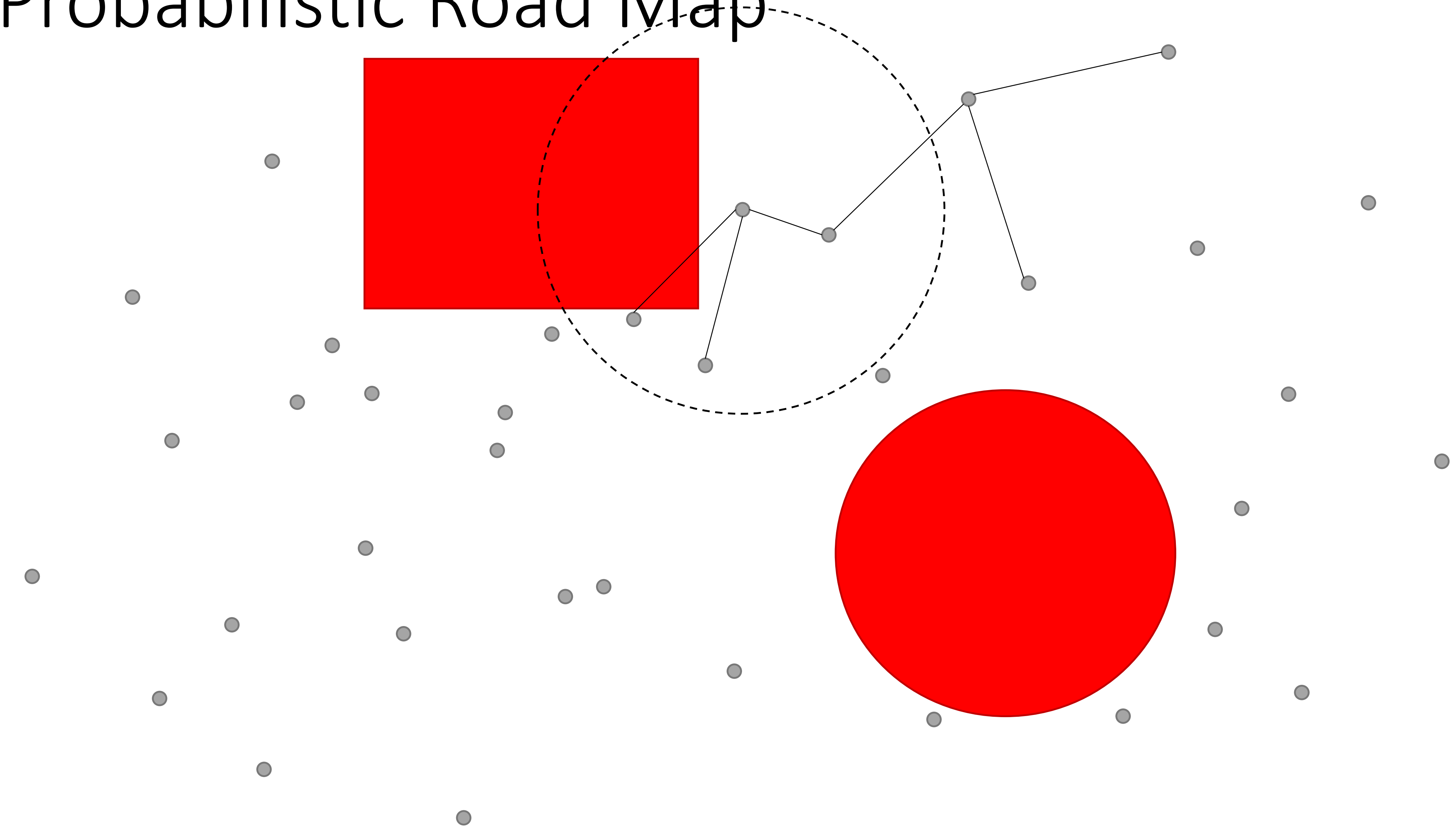
Probabilistic Road Map



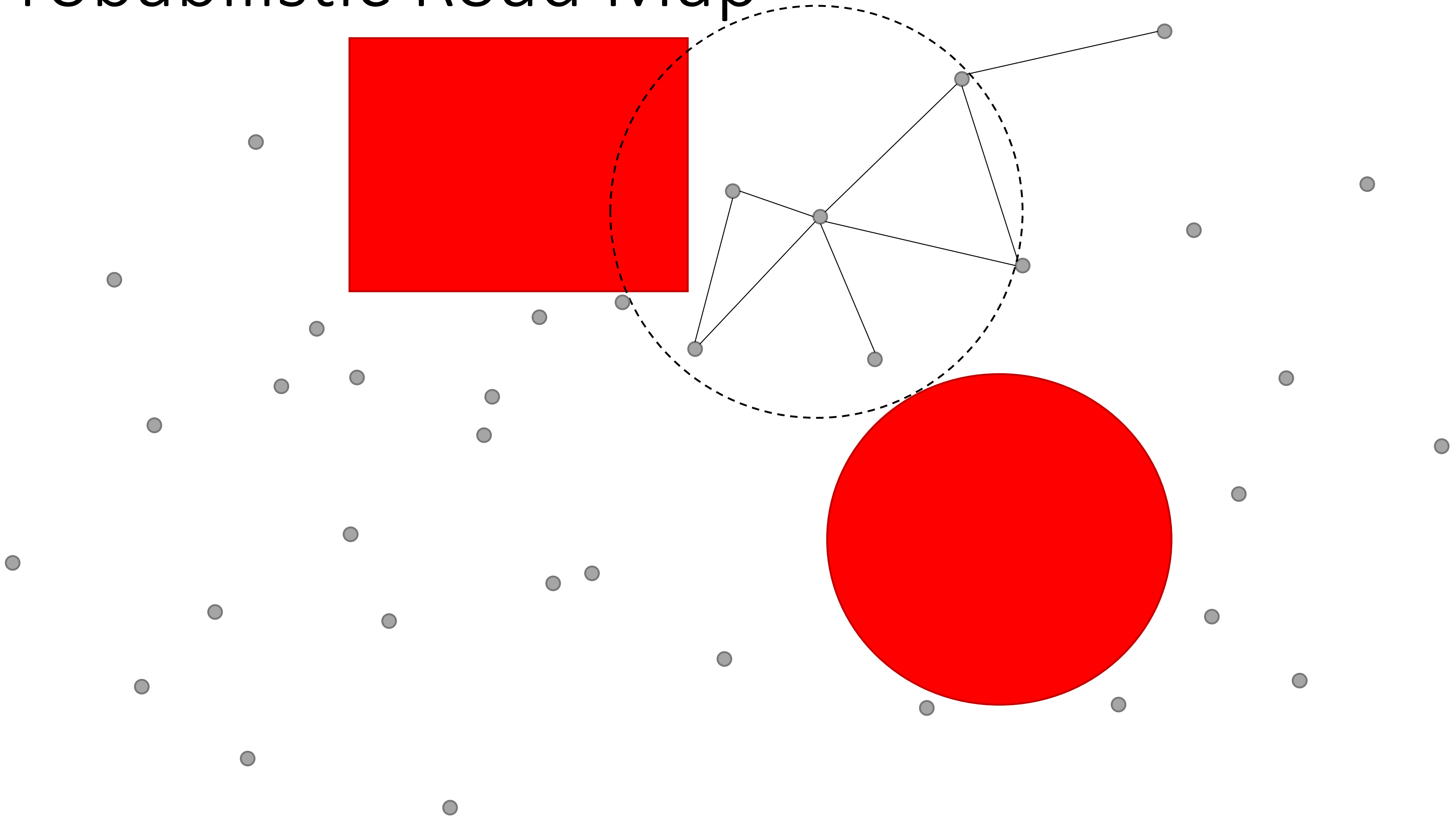
Probabilistic Road Map



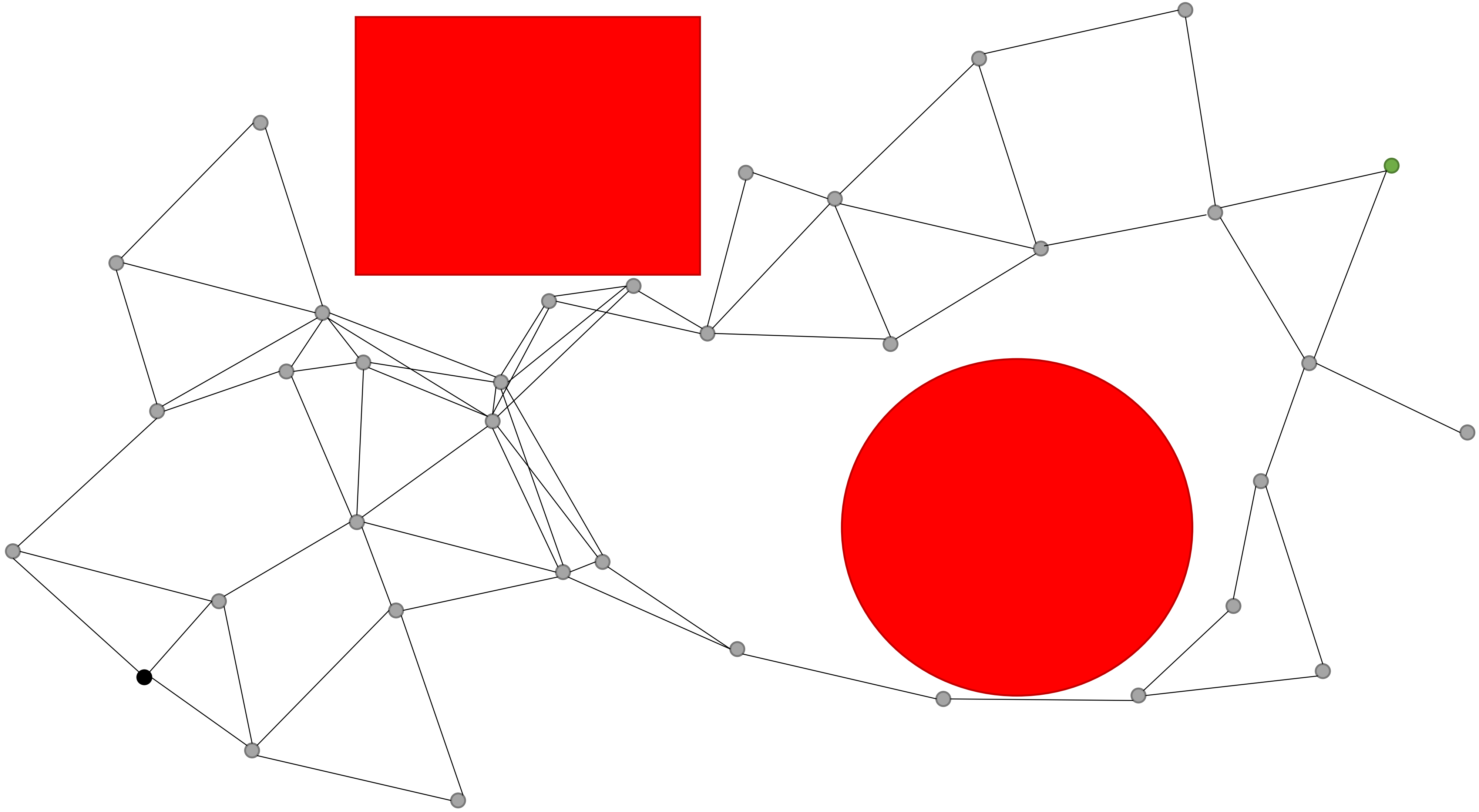
Probabilistic Road Map



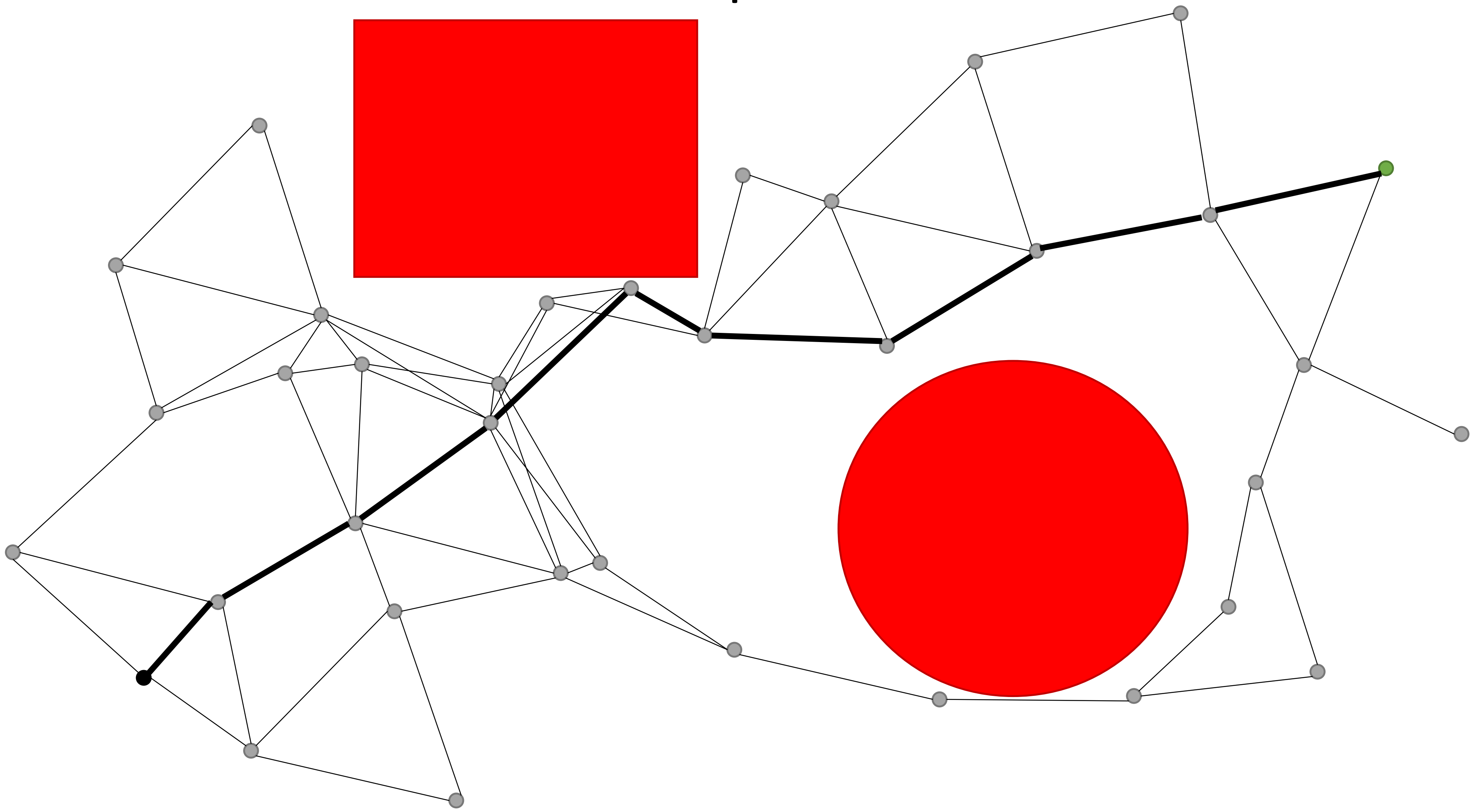
Probabilistic Road Map



Probabilistic Road Map



Probabilistic Road Map



Probabilistic Road Map

- Draw N samples
  - Keep points outside of obstacles
- Choose a disk radius

- For each kept point, draw edge between it and all other points within the disk
  - Keep edges that are collision free (expensive)

- Use graph search algorithm (e.g. A*, Dijkstras) on the resulting graph to find a path

Tuning Parameters

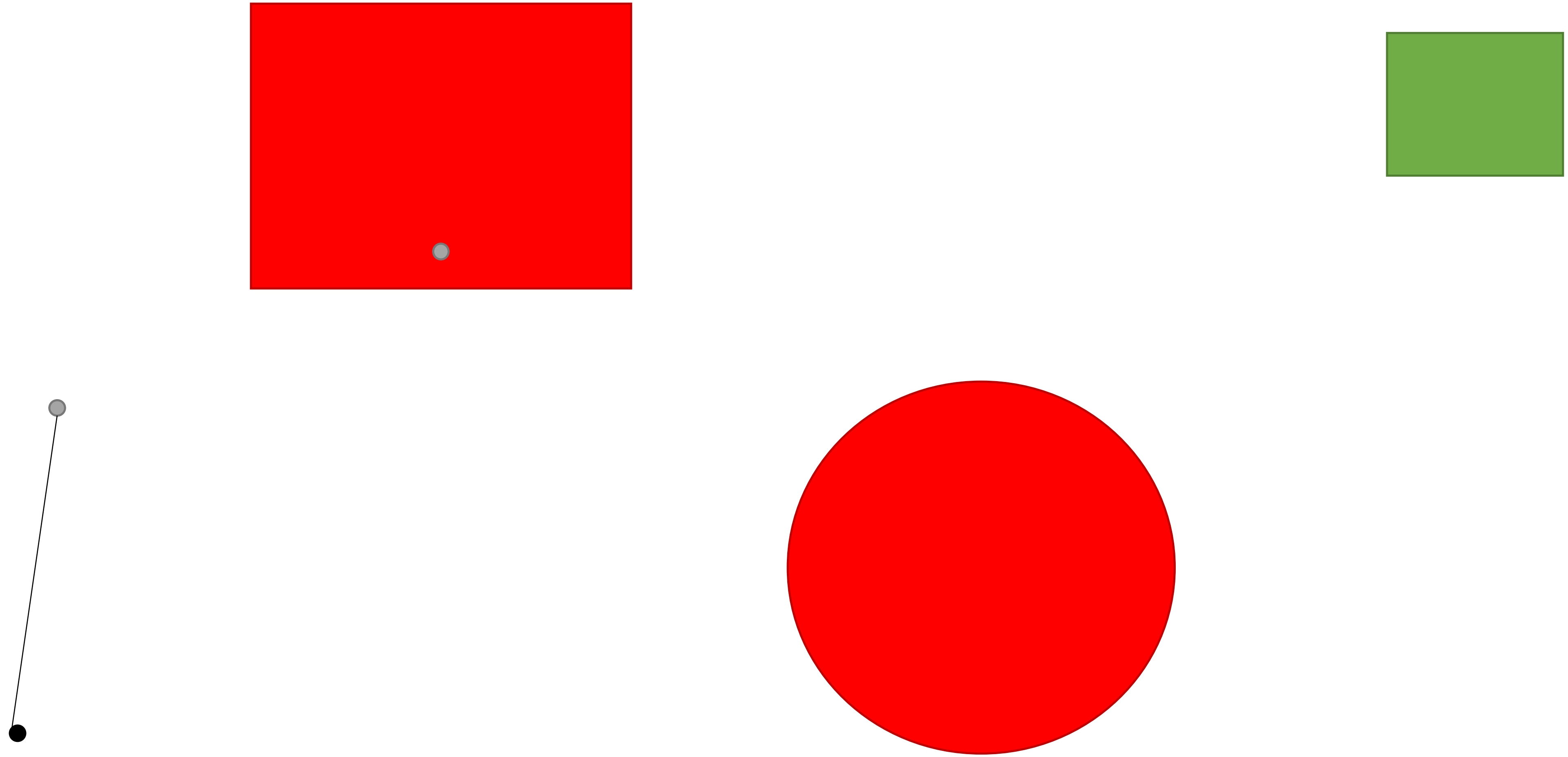
- Sampling distribution
- Deterministic samples are possible
- Collision checker – determines type of obstacles that can be considered

Rapidly-Exploring Random Tree

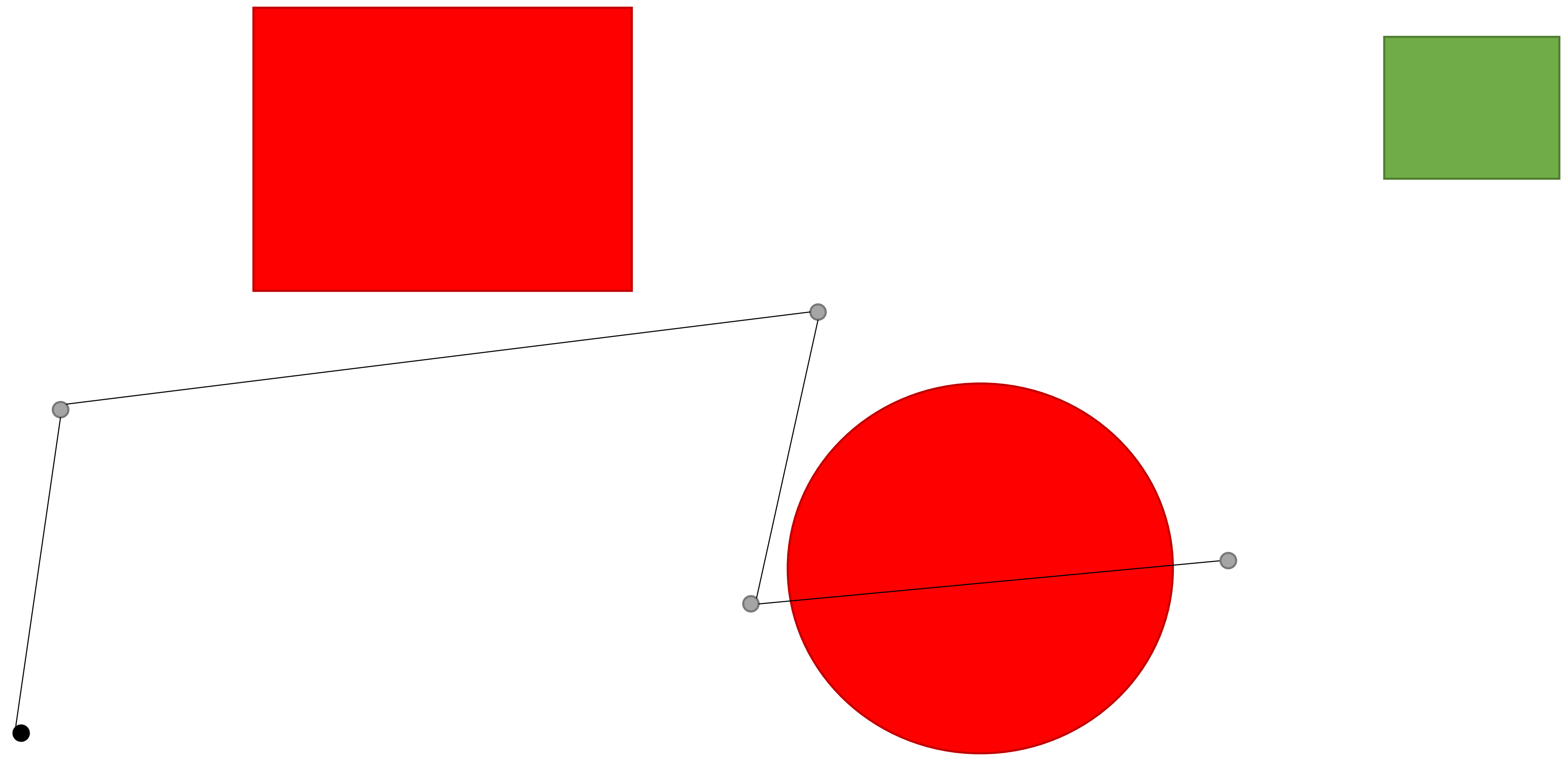
- Draw a sample
  - Connect to nearest neighbour

- Continue until path is found

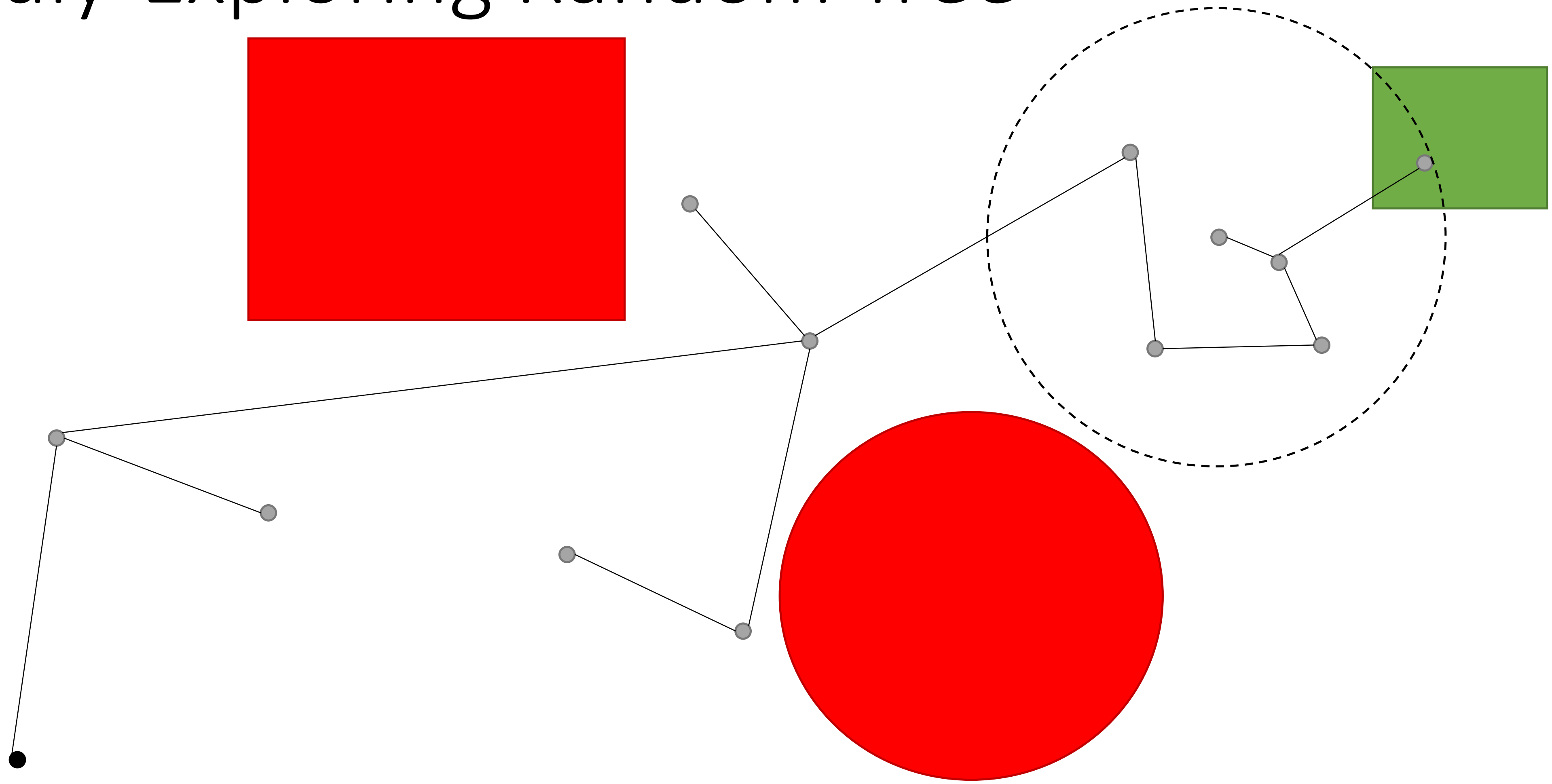
Rapidly-Exploring Random Tree



Rapidly-Exploring Random Tree



Rapidly-Exploring Random Tree



Rapidly-Exploring Random Tree

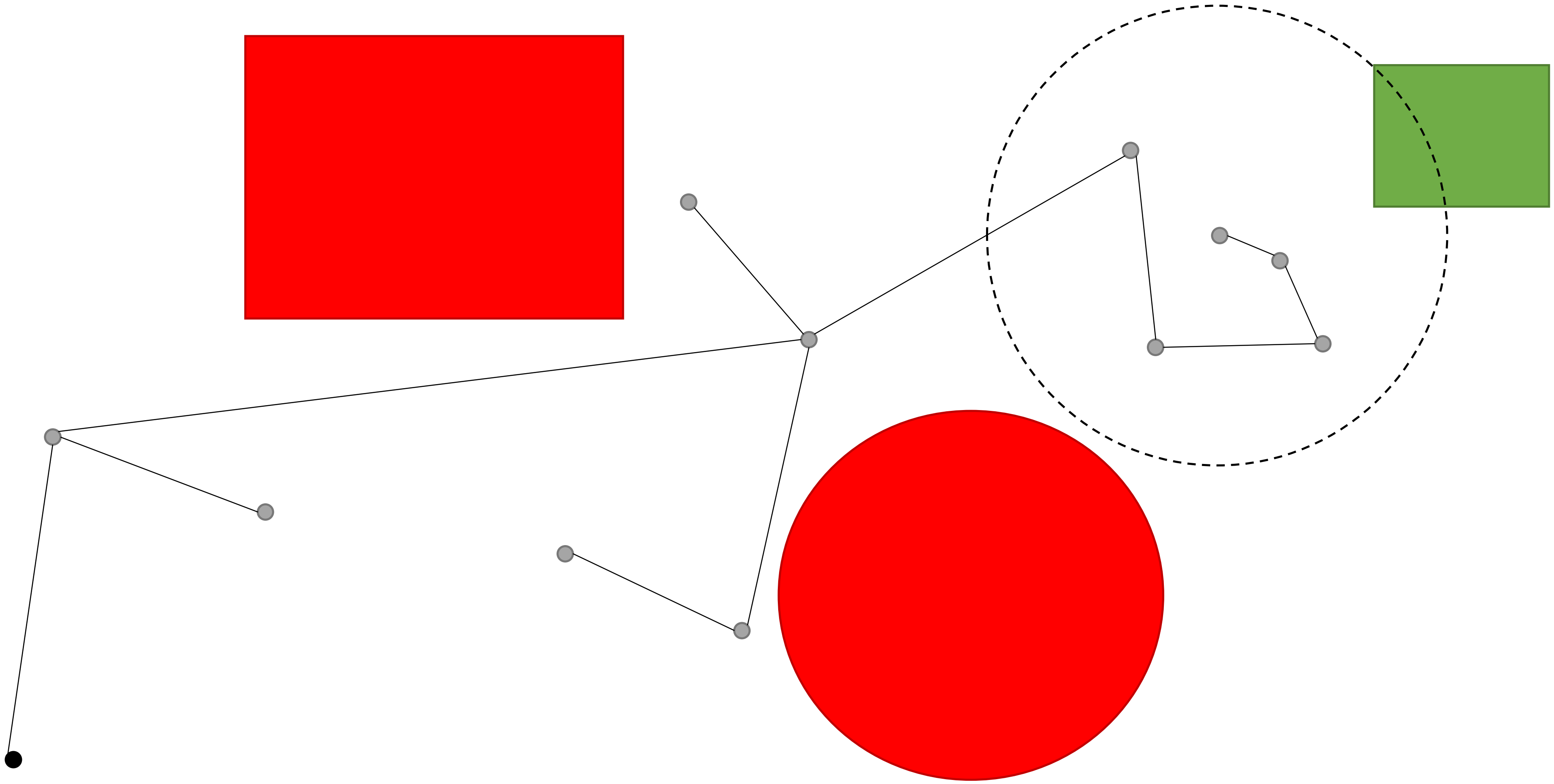
- Draw a sample
  - Connect to nearest neighbour

- Continue until path is found

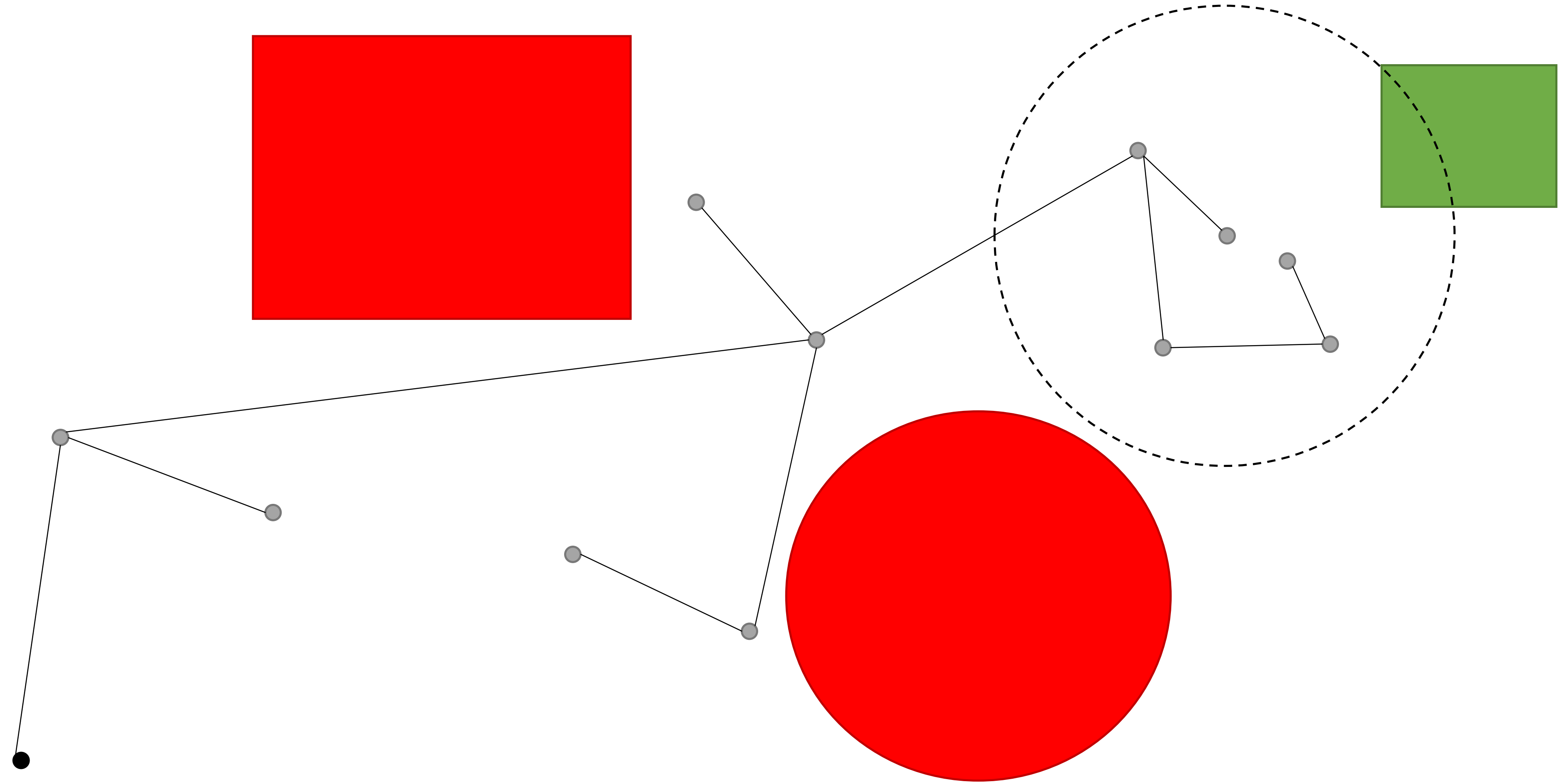
RRT*

- Draw a sample
  - Connect to nearest neighbour
  - Rewire paths in a cheaper way
    - Look within some radius
    - Can we get to the sample in a cheaper way?
    - Is there a cheaper way to get to the samples within the radius?
- Continue until path is found

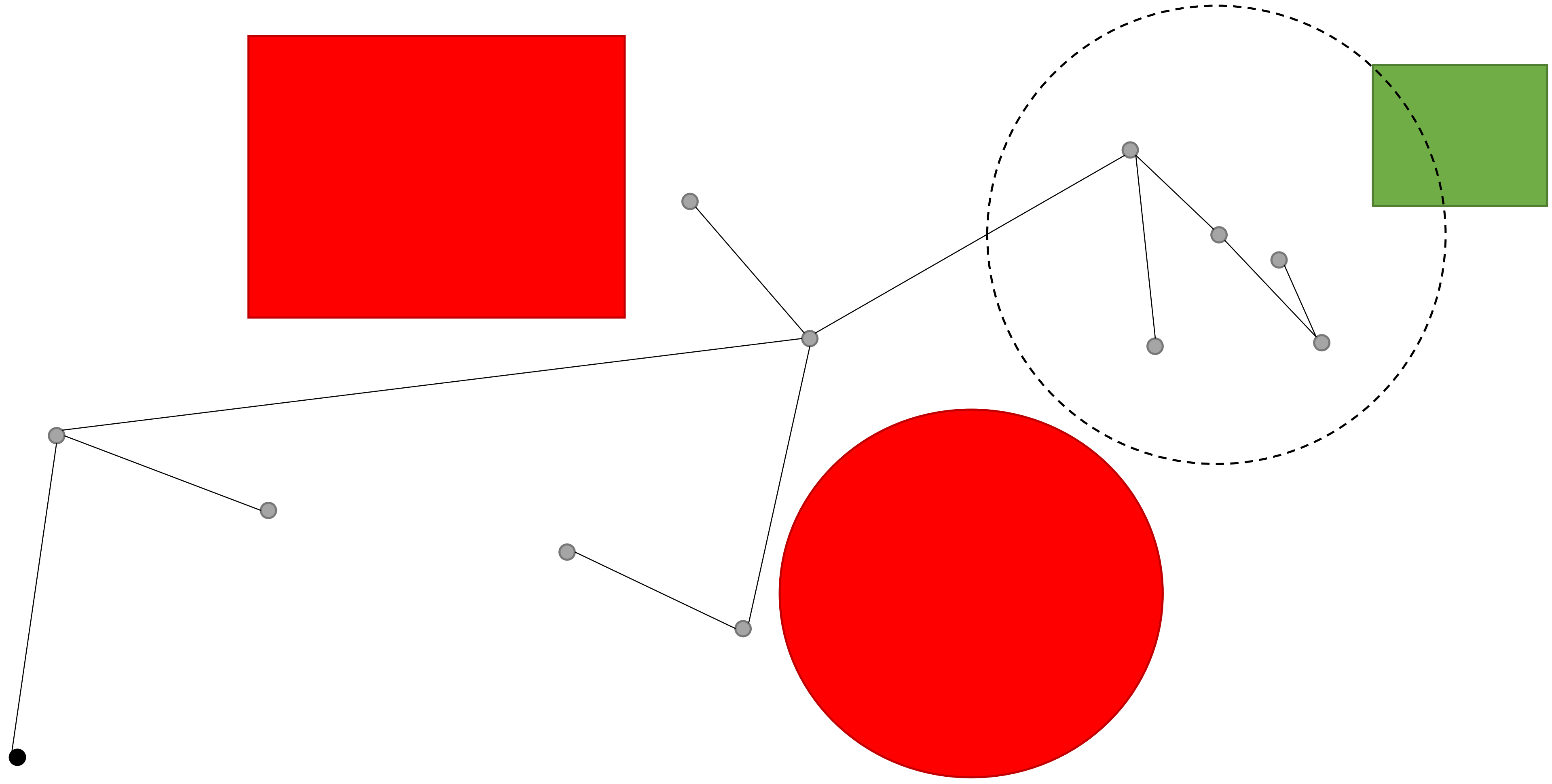
RRT*



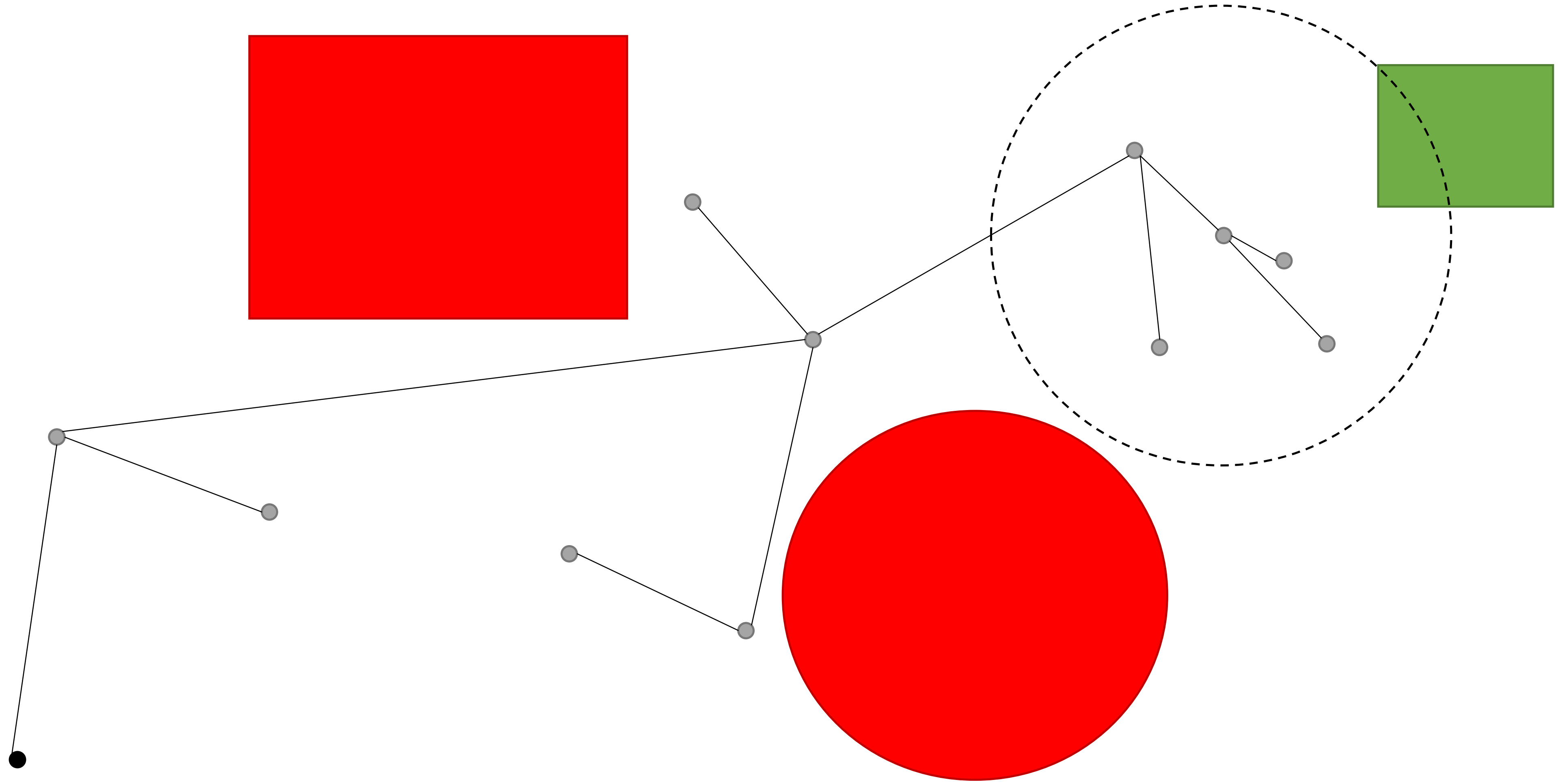
RRT*



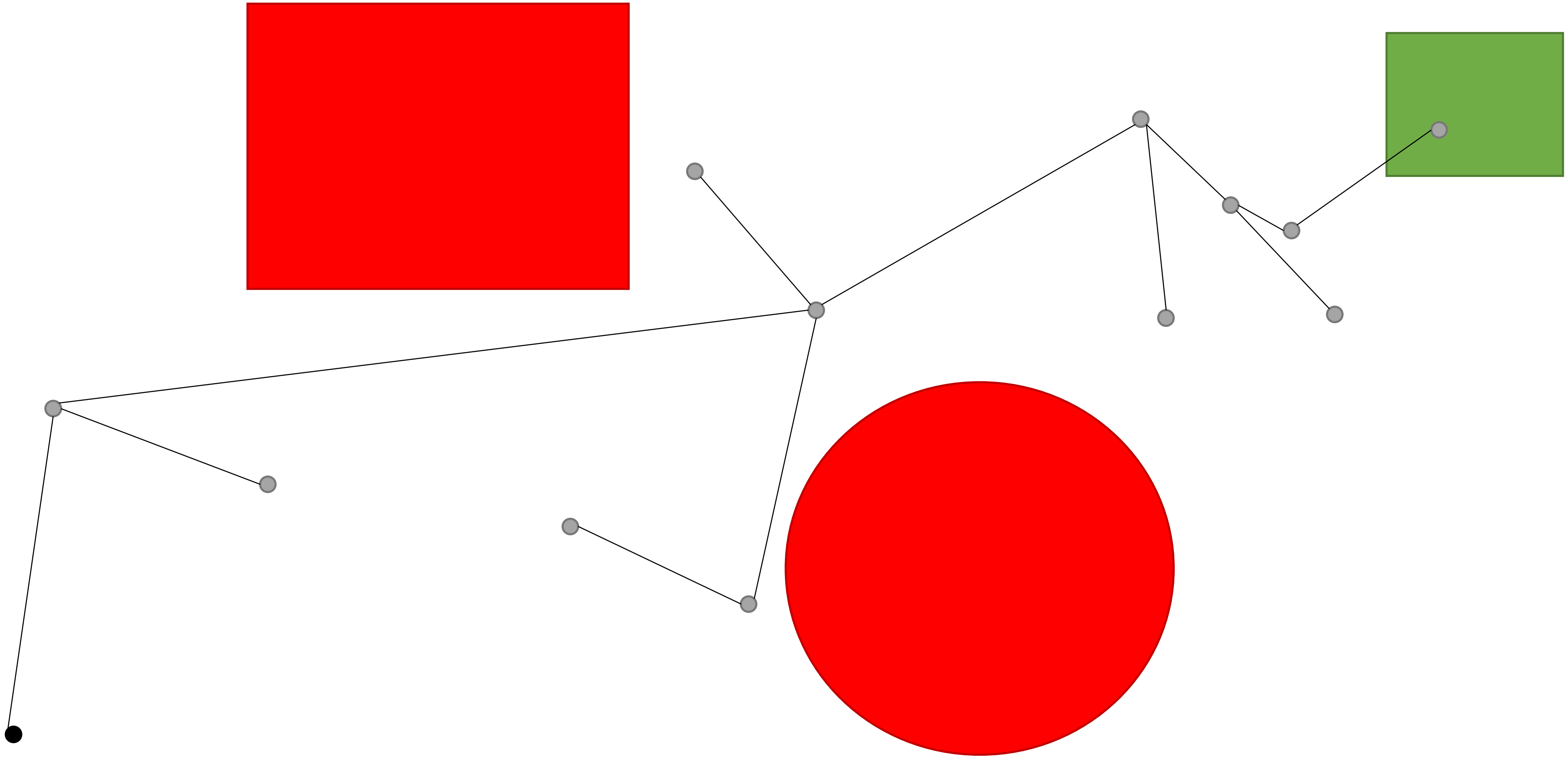
RRT*



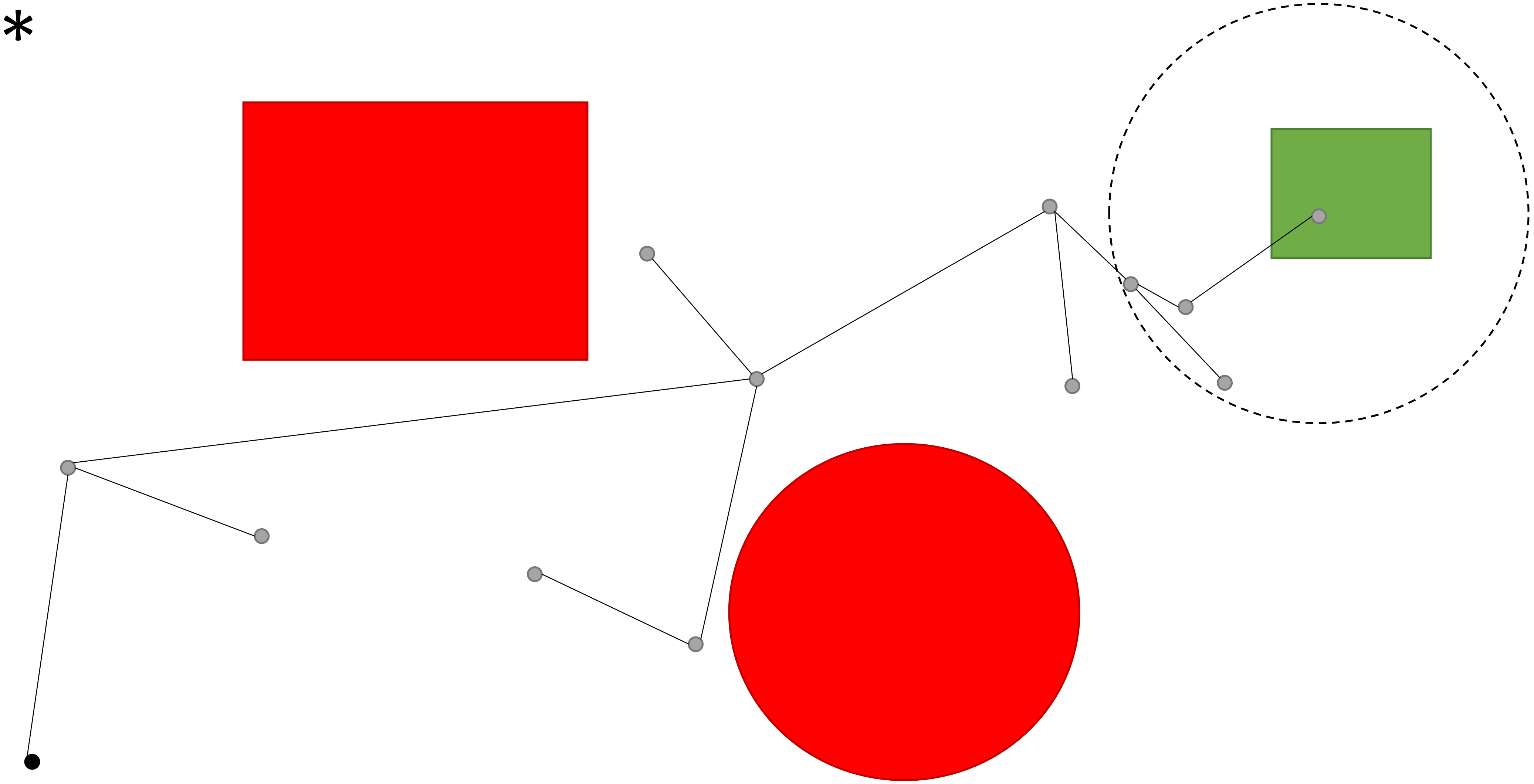
RRT*



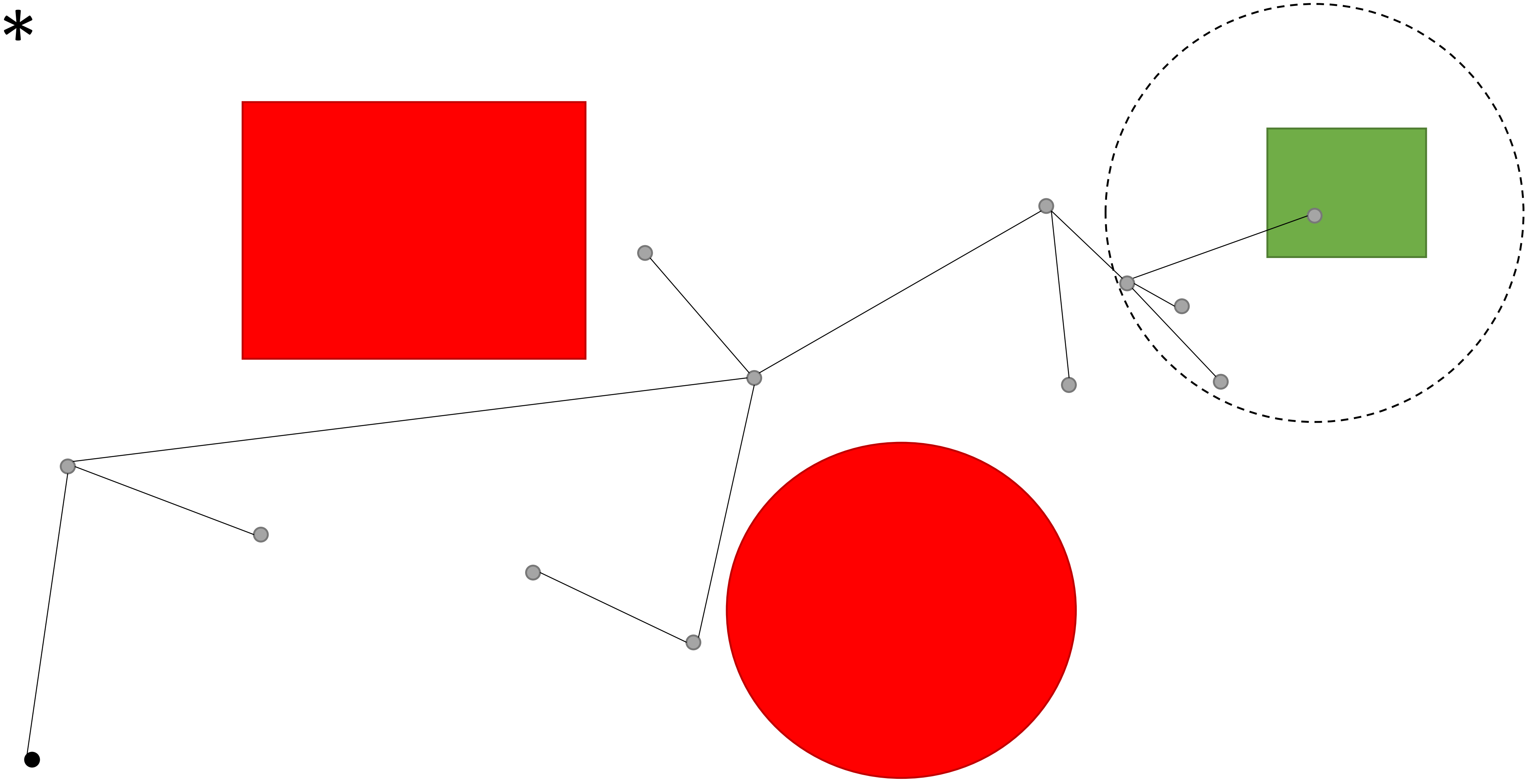
RRT*



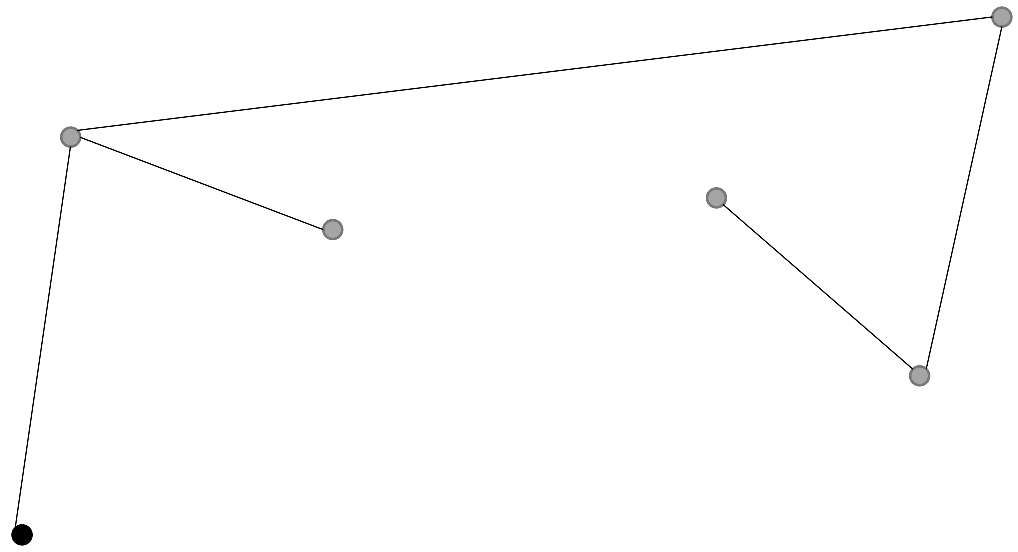
RRT*



RRT*

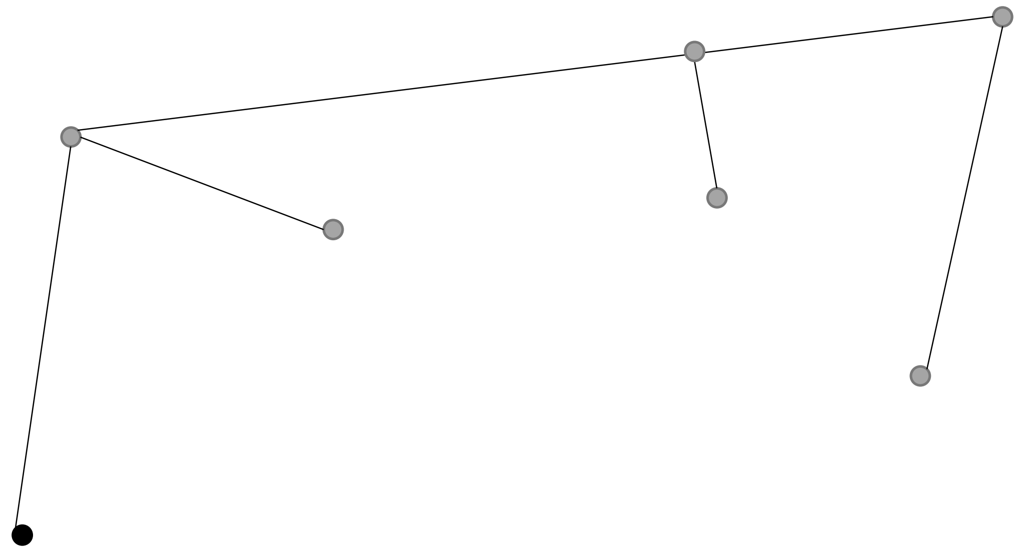


Tuning Parameters



- Sampling distribution
  - Deterministic samples are possible
- Nearest sample vs. nearest point covered by the tree

Tuning Parameters



- Sampling distribution
  - Deterministic samples are possible
- Nearest sample vs. nearest point covered by the tree
- Collision checker – determines type of obstacles that can be considered
- Number of trees
  - Having more than one tree may decrease time needed to find a feasible path

Comments

- PRM
  - Could be slow
  - May need many samples

- RRT
  - Incrementally draw samples → has potentially fewer samples
  - Quality of solution may be very poor → RRT*

- Inherently, complexity is still exponential
  - Hope: obtain feasible (potentially bad) solution quickly

A Difficult Case

- Narrow gaps

Analysis

- Potential computational bottlenecks
  - Collision checking
  - Nearest-neighbour finding

- Optimality
  - Asymptotic, as number of samples goes to infinity
  - Convergence rate: suboptimality bound is $O\left(n^{-\frac{1}{d}}\right)$
    - n – number of samples
    - d – number of dimensions

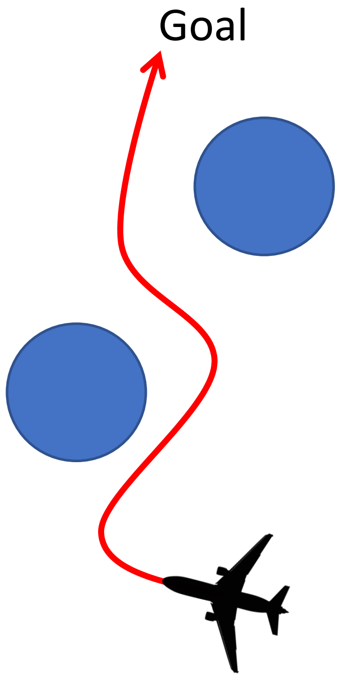
System Dynamics

- RRT and PRM, as presented, is a geometric planner
  - Cannot account for system dynamics
- Incorporating system dynamics
  - Make sure edges between nodes are dynamically feasible
  - Difficult in general, but has been done for special cases
  - Backward and forward reachability concepts are useful
  - Some references
    - LaValle, Kuffner. “Randomized Kinodynamic Planning,” 2001.
    - Webb, Van Den Berg. “Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics,” 2013.
    - Schmerling, Janson, Pavone. “Optimal Sampling-Based Motion Planning under Differential Constraints:the Drift Case with Linear Affine Dynamics,” 2015.

Robustness

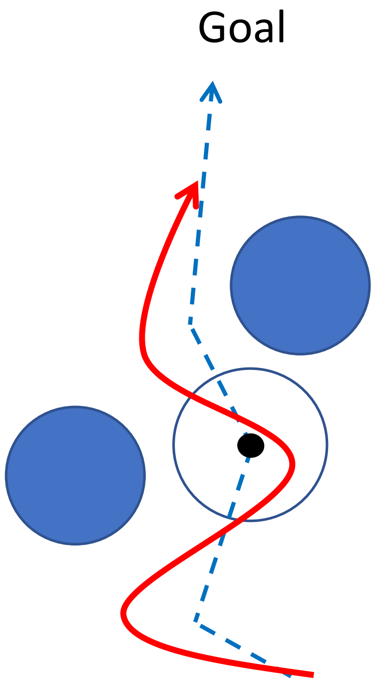
Slide courtesy of Sylvia Herbert

Slow and Accurate Planning:



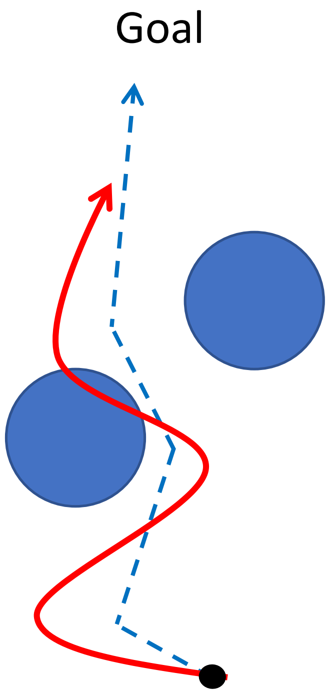
- Optimal control
- Guarantees on safety and goals
- Handles external disturbances (e.g. wind)
- Slow to compute

FaSTrack:



- Precompute a tracking error bound based on relative state between the true system and the planned path
- Make it modular & easy to incorporate in all sorts of real-time path/trajectory planners

Fast (but less accurate) Planning:

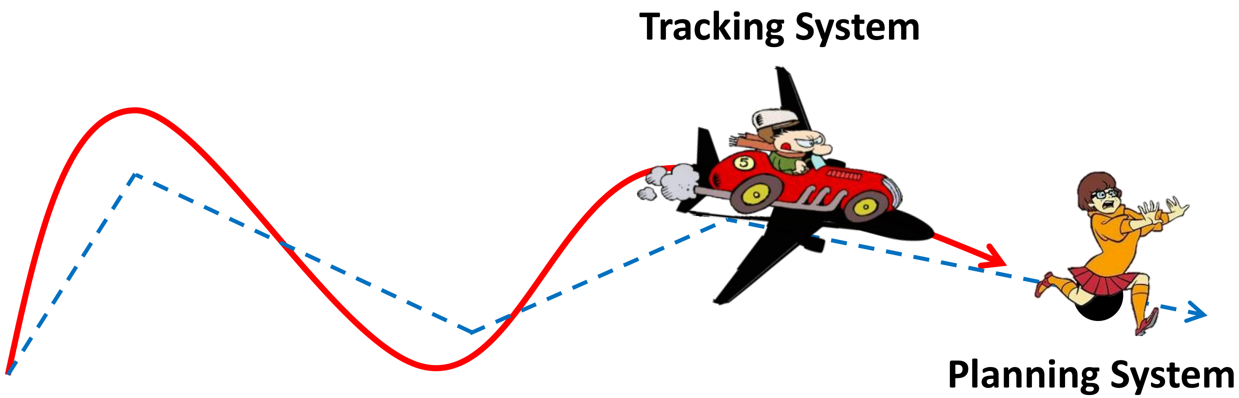


- Very fast with simple dynamics
- May not capture all system behavior
- Not necessarily robust to disturbances

Herbert, Chen, Han, Bansal, Fisac, Tomlin. “FaSTrack: a modular framework for fast and guaranteed safe motion planning,” 2017.

Slide courtesy of Sylvia Herbert

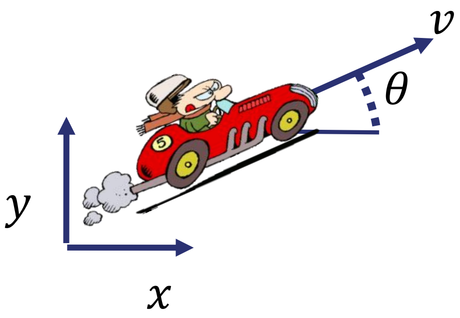
Precomputed Tracking Bound



Rufus Isaacs

Homicidal Chauffeur Problem (1951)

Tracking System Model



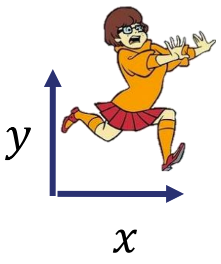
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{matrix} v\cos\theta \\ v\sin\theta \\ \omega \end{matrix}$$

$$u_s = \omega$$

Planning System Model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{matrix} u_{px} \\ u_{py} \end{matrix}$$

$$u_p = [u_{px}, u_{py}]$$



Precomputed Tracking Bound

- Tracking system (car) pursues planning system (runner)
- Planning system tries to evade tracking system
- What will be the maximum relative distance over time?

Note:

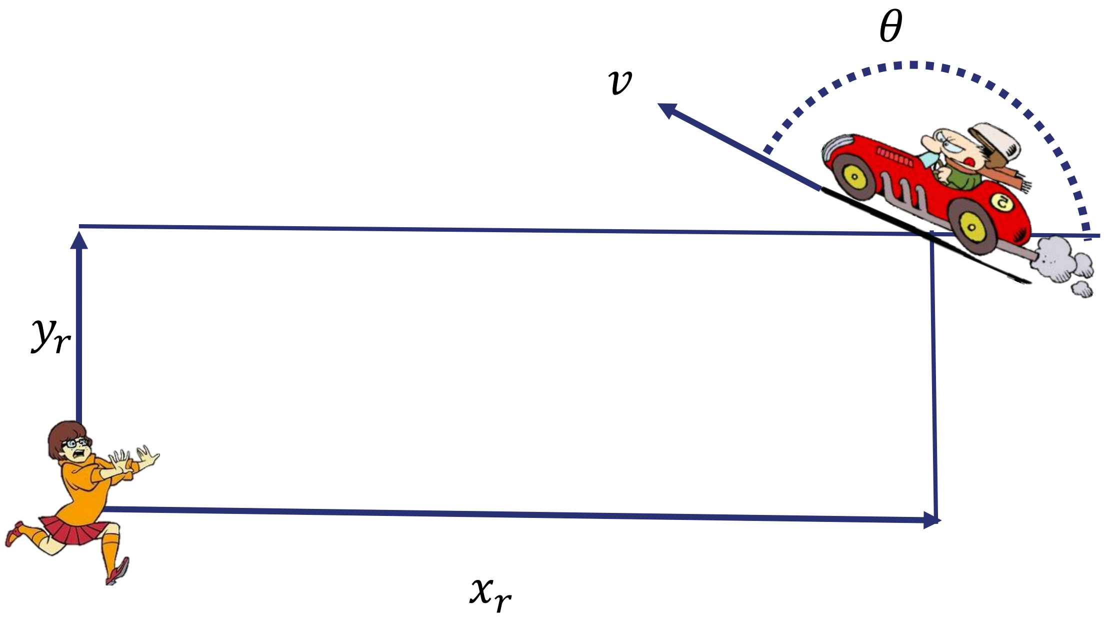
Maximum relative distance over time

≡ Worst possible tracking error over time

≡ Tracking error bound

Precomputed Tracking Bound

Goal: Map initial relative state to worst possible tracking error over time



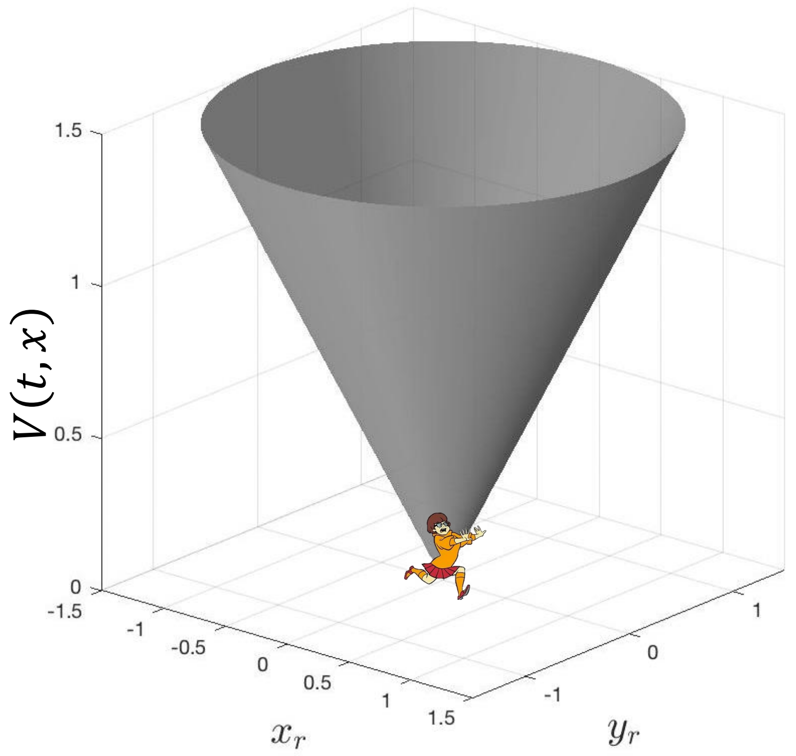
Relative System

$$\dot{r} = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta} \end{bmatrix} = \begin{matrix} v\cos\theta - u_{px} \\ v\sin\theta - u_{py} \\ \omega \end{matrix}$$

Slide courtesy of Sylvia Herbert

Precomputed Tracking Bound

Goal: Map initial relative state to worst possible tracking error over time



Planning system tries to maximize error

Tracking system tries to minimize error

Keep track of maximum cost over time

$$V(t, x(t)) = \max_{\Gamma[u](\cdot)} \min_{u(\cdot)} \max_{s \in [t,0]} l(x(s))$$

- Take $t \to -\infty$ for infinite time horizon case

Example: 10D Tracking 3D Single Integrator using RRT

10D near-hover quadrotor model

$$\begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{\theta}_x \\ \dot{\omega}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{\theta}_y \\ \dot{\omega}_y \\ \dot{z} \\ \dot{v}_z \end{bmatrix} = \begin{bmatrix} v_x \\ g \tan\theta_x \\ -d_1\theta_x + \omega_x \\ -d_0\theta_x + n_0 u_x \\ v_y \\ g \tan\theta_y \\ -d_1\theta_y + \omega_y \\ -d_0\theta_y + n_0 u_y \\ v_z \\ k_T u_z - g \end{bmatrix}$$

3D single integrator

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$