Sampling-Based Motion Planning

Reading: Modern Robotics 10.2.3, 10.4 – 10.5

This Lecture

- What are some common data structures for motion planning?
- Why not just discretize the space into a grid?
- How do sampling-based motion planners work?

Graph

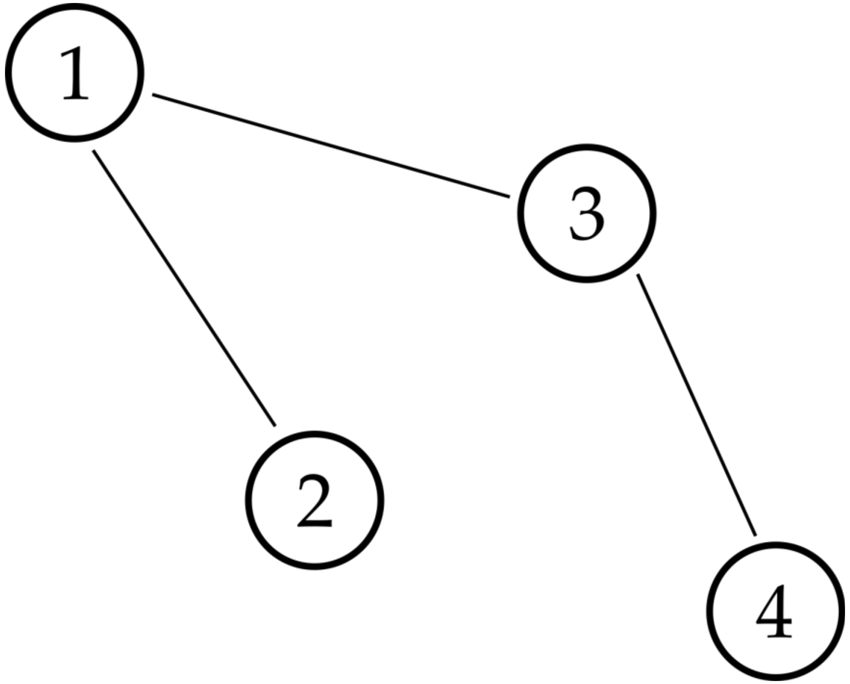
A **graph** is a collection of nodes and edges.

Nodes contain information

Example: node represents joint position

Each **edge** connects two nodes

Example: edge indicates we can move from one node (joint position) to another node (joint position) without collisions

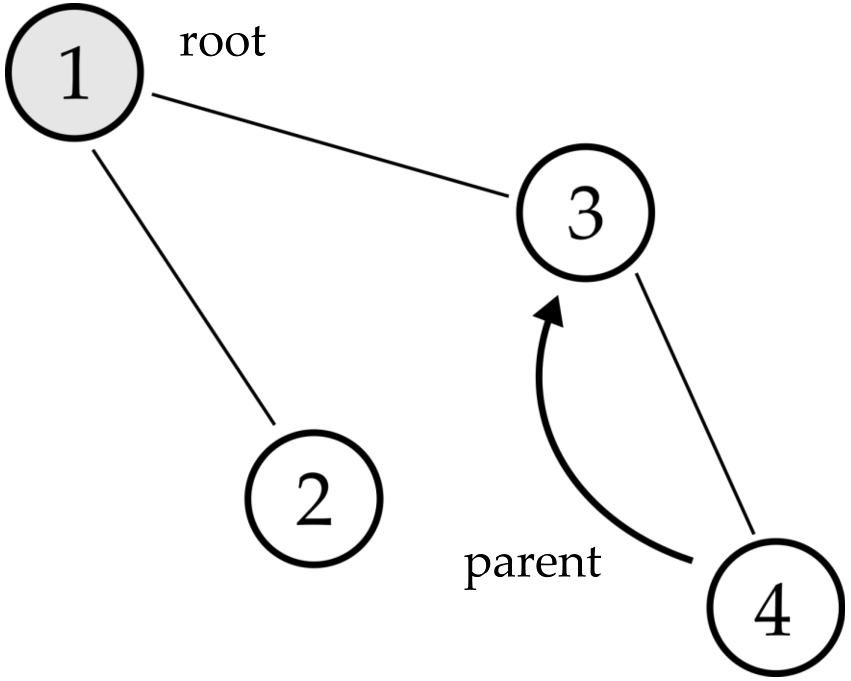


Graph

A **graph** is a collection of nodes and edges.

A **tree** is a type of graph for motion planning.

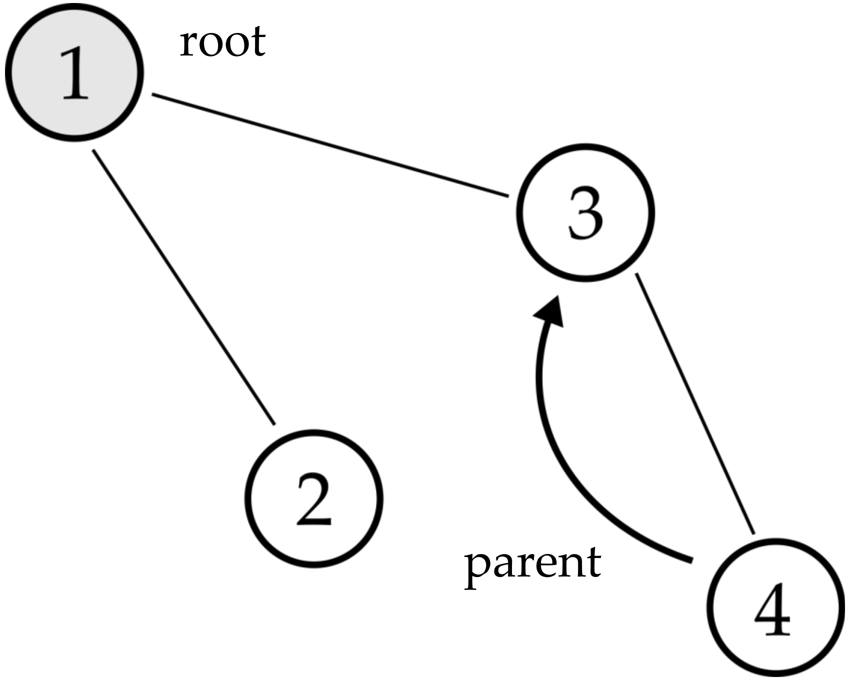
- There are no cycles (closed loops)
- The root node has no parents
- All other nodes have one parent



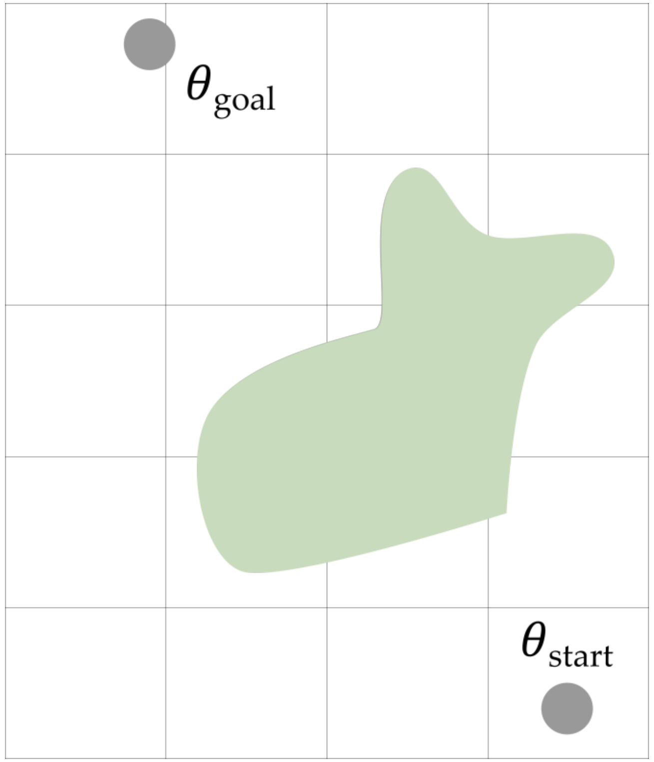
Graph

Command Window

```
>> node.theta = [0; 1];
>> node.parent = 3;
>> node

node =

  struct with fields:

     theta: [2×1 double]
    parent: 3
```



Grid Methods

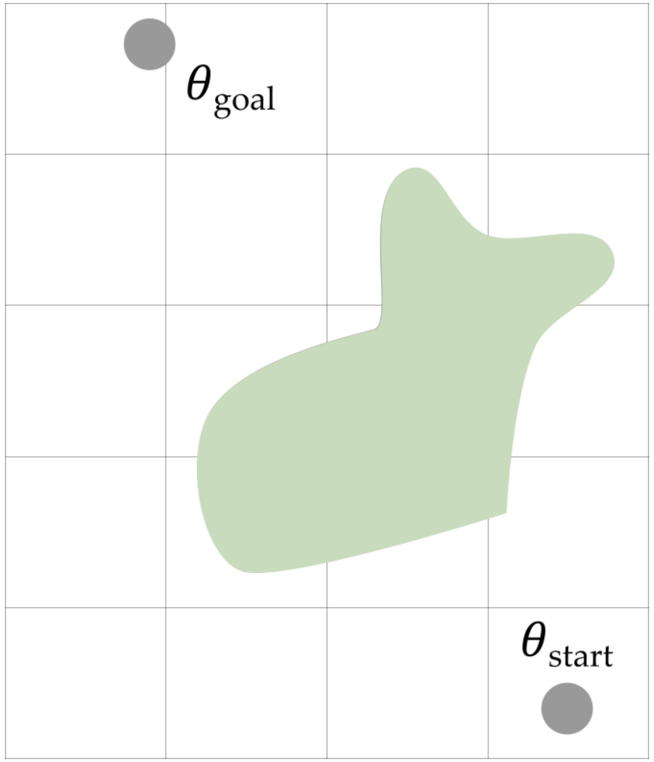


Naive solution

- Discretize the environment into a grid
- Assign a node at every grid cell
- Search the grid to find shortest path (*example: A* algorithm*)

What are some issues with this approach?

Grid Methods



Naive solution

- The number of grid cells increases exponentially with the number of dimensions
- Reducing resolution may miss free paths

Say we have a 7-DoF robot arm, and we discretize each joint with resolution $k = 100$. *We will have to search* $k^7 = 100$ *trillion nodes!*

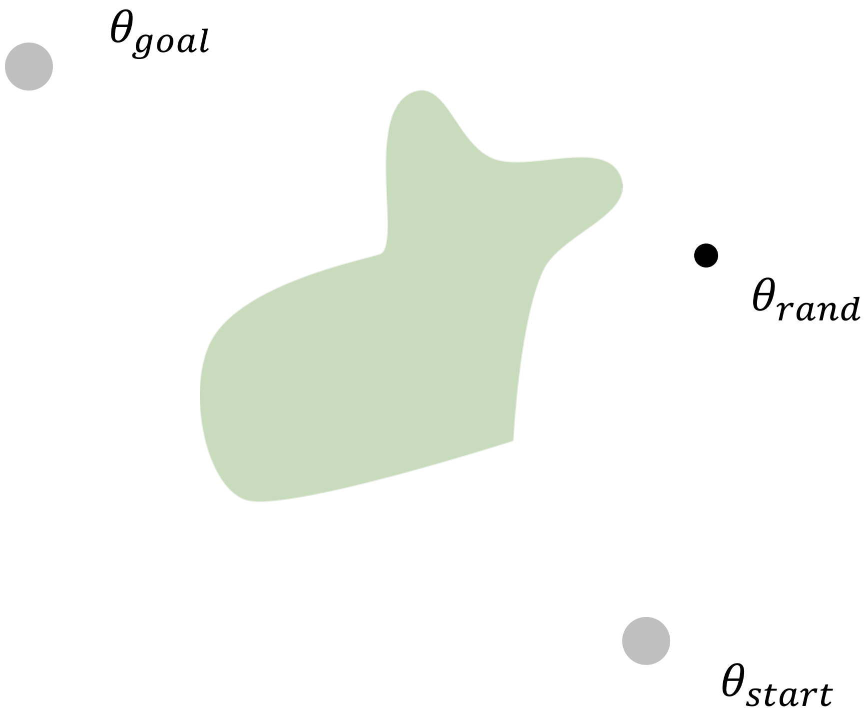
How can we efficiently find motion plans for **high-dimensional** robot arms?

RRT Algorithm

Initialize graph G with root θ_{start}

while length$(G) < N$

$\theta_{rand} \leftarrow$ sample random joint position



RRT Algorithm

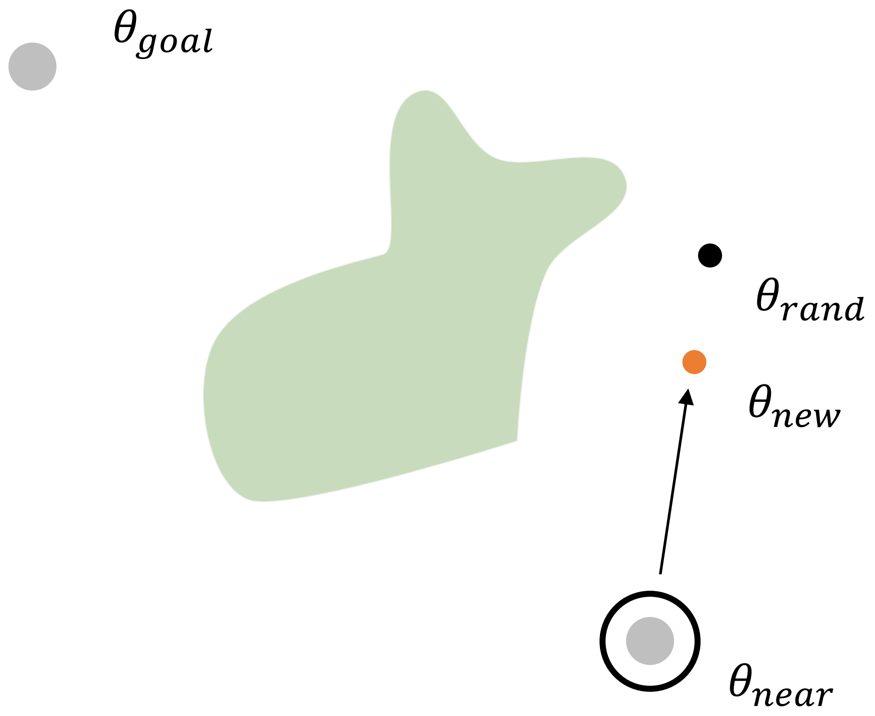
Initialize graph G with root θ_{start}

while length(G) $< N$

  $\theta_{rand} \leftarrow$ sample random joint position

  $\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

  $\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}



RRT Algorithm

Initialize graph G with root θ_{start}

while length(G) $< N$

$\theta_{rand} \leftarrow$ sample random joint position

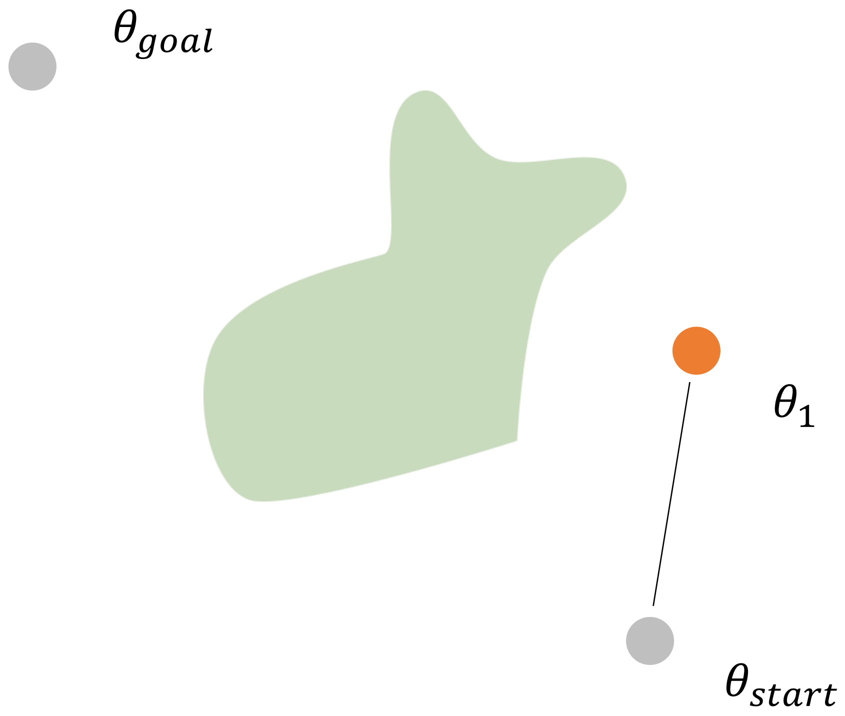
$\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

$\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}

If θ_{new} is collision free

Parent $\theta_{new} \leftarrow \theta_{near}$

Add θ_{new} to G



RRT Algorithm

Initialize graph G with root θ_{start}

while length(G) $< N$

$\theta_{rand} \leftarrow$ sample random joint position

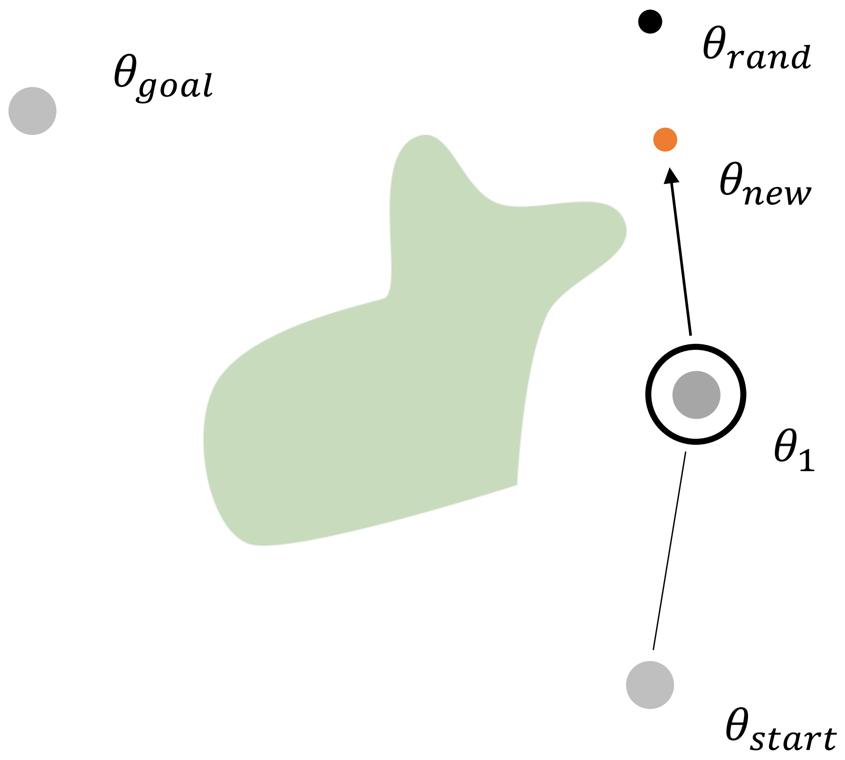
$\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

$\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}

If θ_{new} is collision free

Parent $\theta_{new} \leftarrow \theta_{near}$

Add θ_{new} to G



RRT Algorithm

Initialize graph G with root θ_{start}

while length$(G) < N$

$\theta_{rand} \leftarrow$ sample random joint position

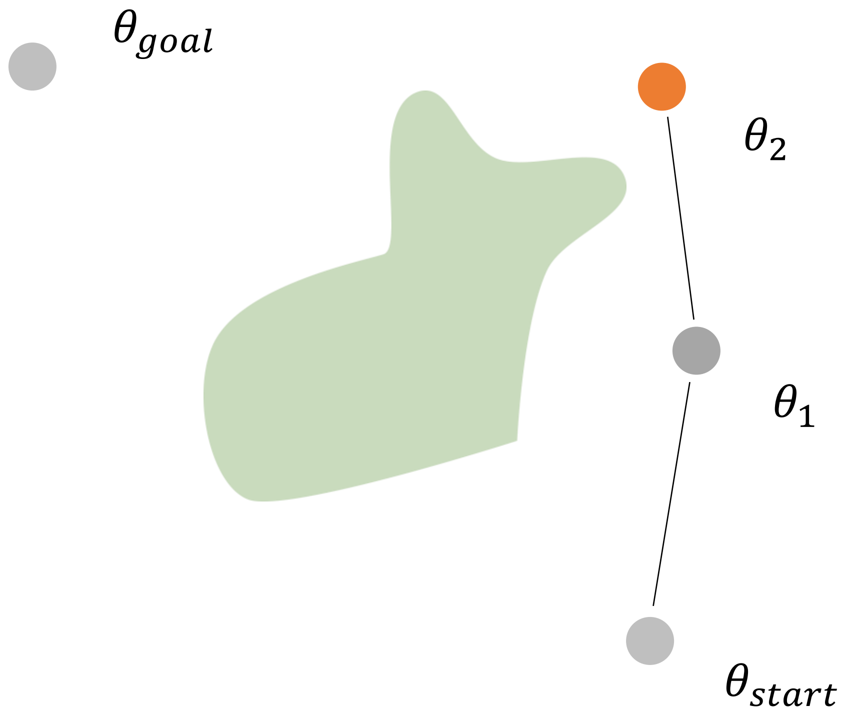
$\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

$\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}

If θ_{new} is collision free

Parent $\theta_{new} \leftarrow \theta_{near}$

Add θ_{new} to G



RRT Algorithm

Initialize graph G with root θ_{start}

while length$(G) < N$

$\theta_{rand} \leftarrow$ sample random joint position

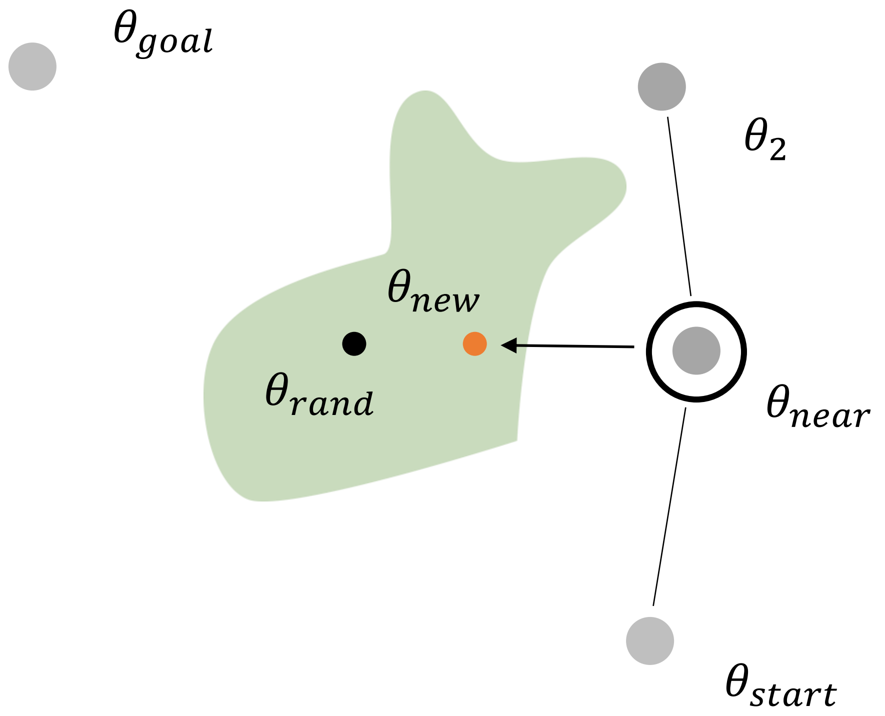
$\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

$\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}

If θ_{new} is collision free

Parent $\theta_{new} \leftarrow \theta_{near}$

Add θ_{new} to G



RRT Algorithm

Initialize graph G with root θ_{start}

while length(G) $< N$

$\theta_{rand} \leftarrow$ sample random joint position

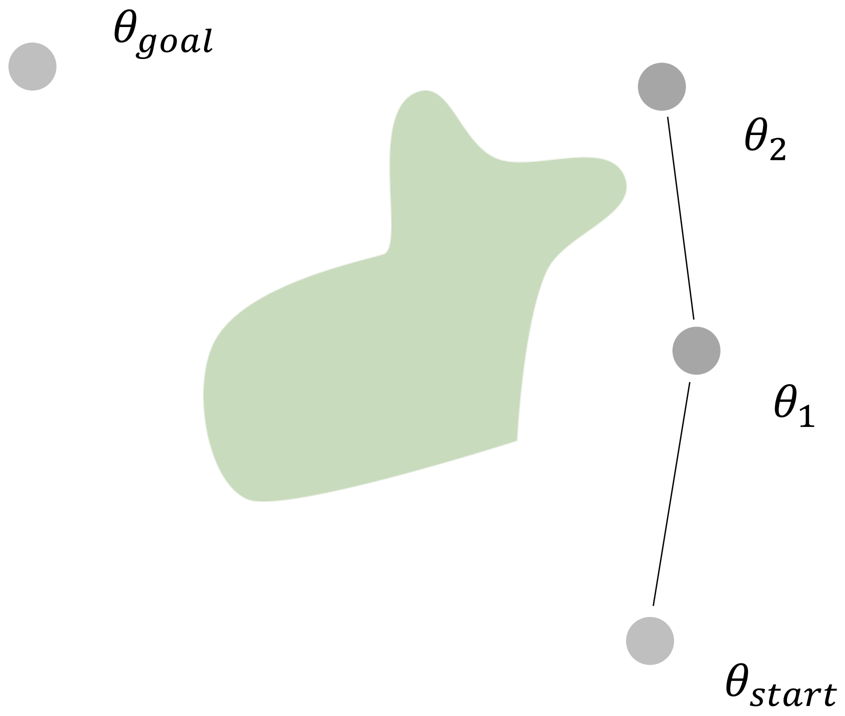
$\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

$\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}

If θ_{new} is collision free

Parent $\theta_{new} \leftarrow \theta_{near}$

Add θ_{new} to G



RRT Algorithm

Initialize graph G with root θ_{start}

while length(G) $< N$

$\theta_{rand} \leftarrow$ sample random joint position

$\theta_{near} \leftarrow$ node in G nearest to θ_{rand}

$\theta_{new} \leftarrow$ take step from θ_{near} towards θ_{rand}

If θ_{new} is collision free

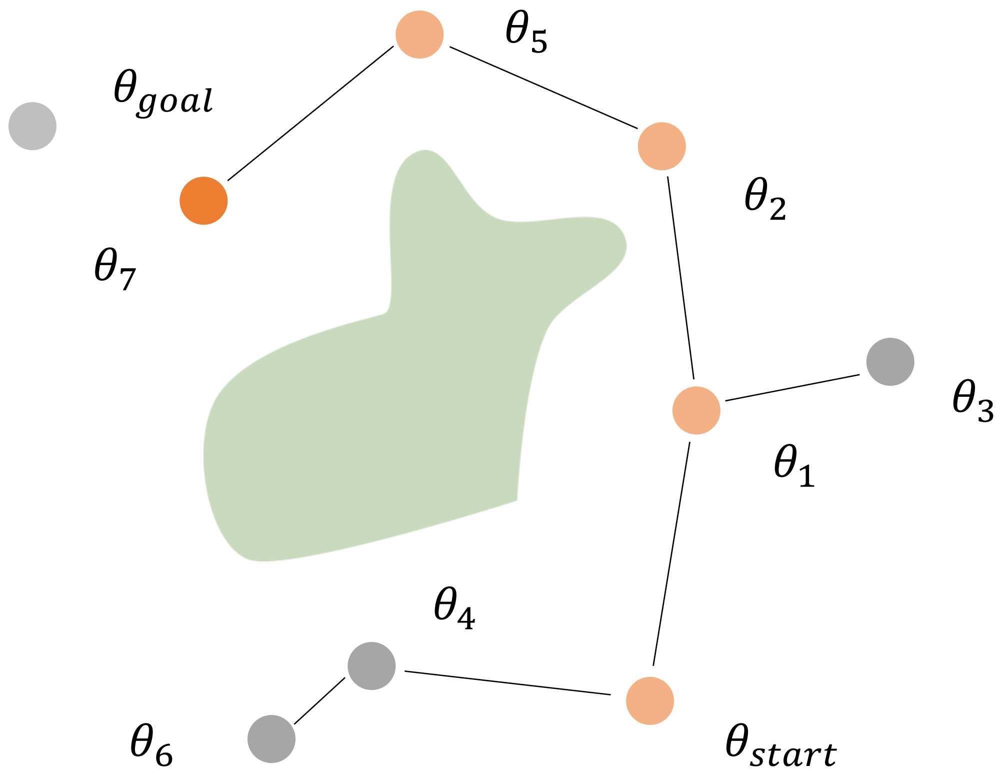
Parent $\theta_{new} \leftarrow \theta_{near}$

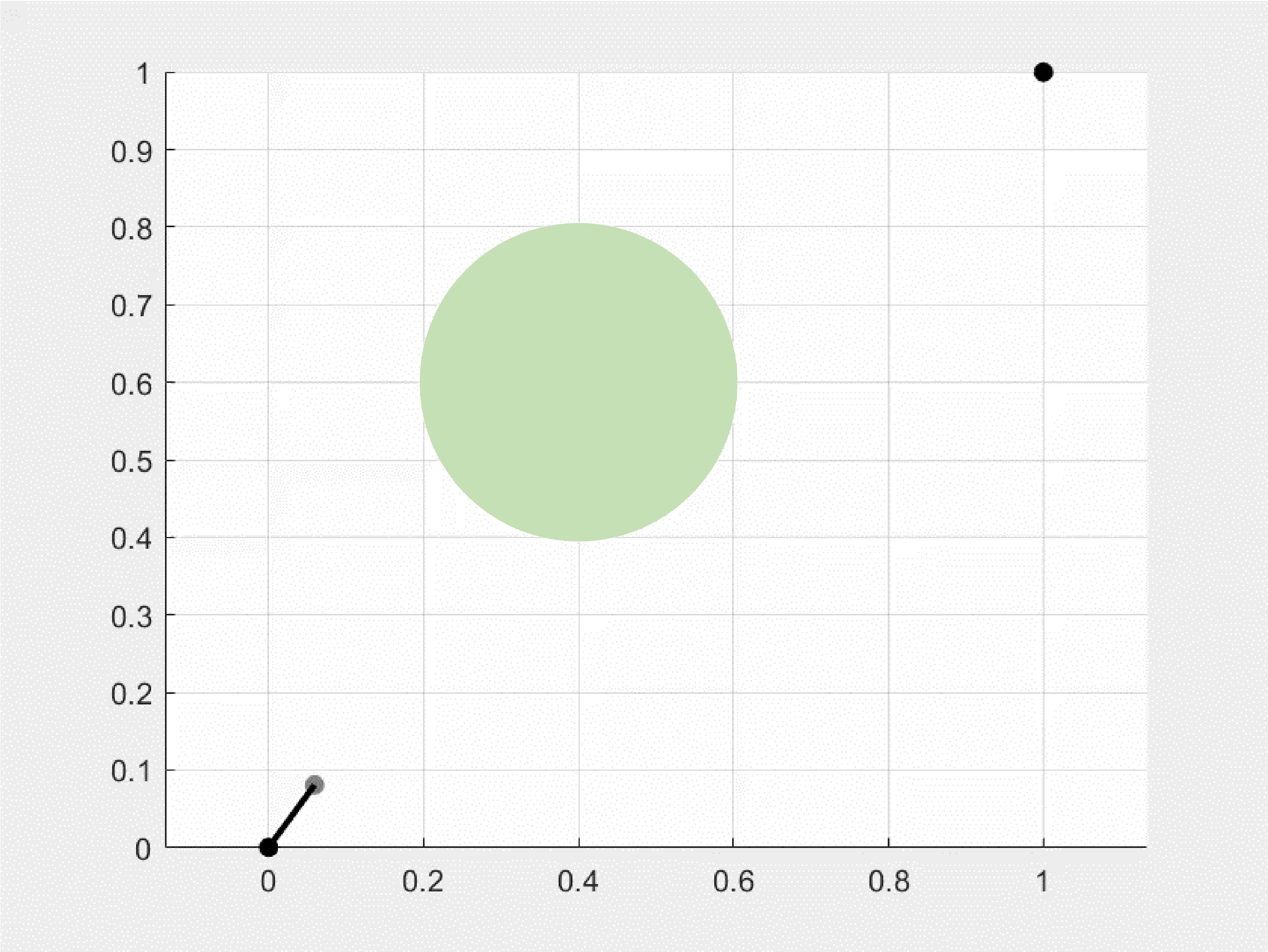
Add θ_{new} to G

If $\|\theta_{new} - \theta_{goal}\| < \varepsilon$

Return Success

Return Failure




1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8
1

What are some **pros and cons** of sampling methods?

Sampling Methods

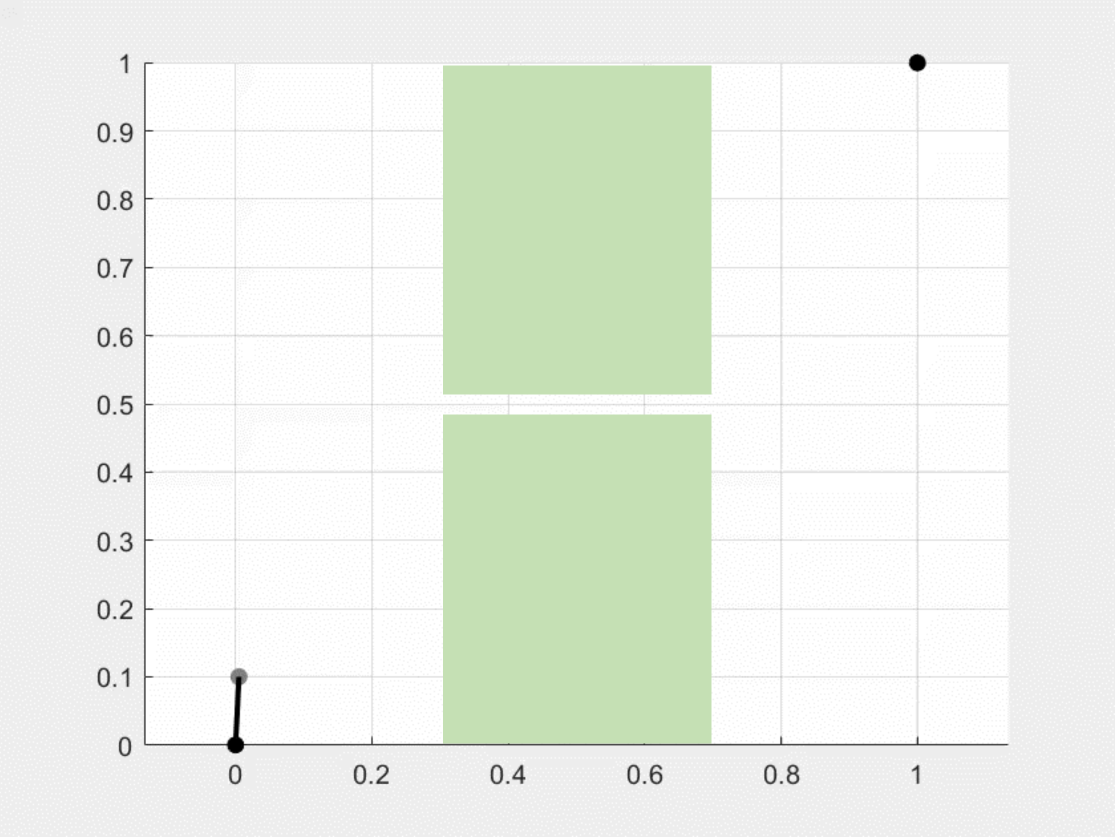
Advantages:

- ***Probabilistically* complete**.
  If a solution exists, the planner will find it as the number of samples increases
- Works for high-dimensional spaces
- As far as the environment goes, we only need a collision checker

Sampling Methods

Disadvantages:

- The resulting trajectory is not ***smooth***
- Struggles with environments that have narrow passages



This Lecture

- What are some common data structures for motion planning?
- Why not just discretize the space into a grid?
- How do sampling-based motion planners work?

Next Lecture

- Summarizing the semester!
- What are some open questions in robotics?