ME/AER 676 Robot Modeling & Control
Spring 2023

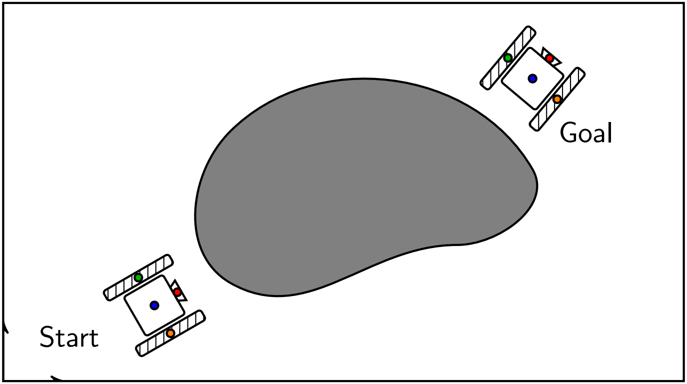
Sampling-Based Motion Planning

Hasan A. Poonawala

Department of Mechanical Engineering
University of Kentucky

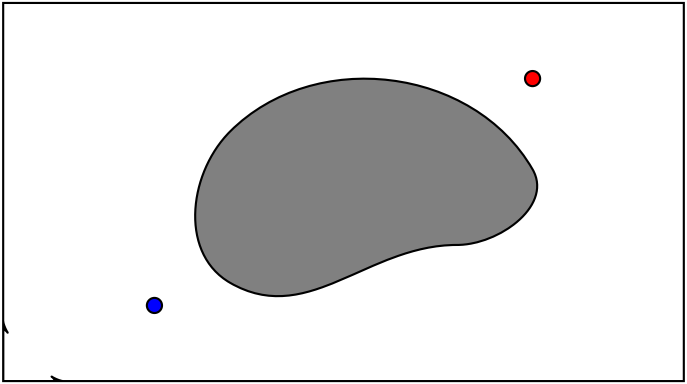
Email: hasan.poonawala@uky.edu
Web: https://www.engr.uky.edu/~hap

Motion Planning



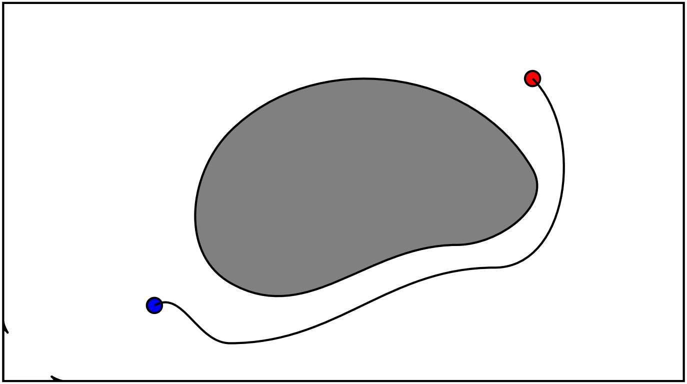
Motion Planning Problem

Motion Planning



Over-simplify the problem

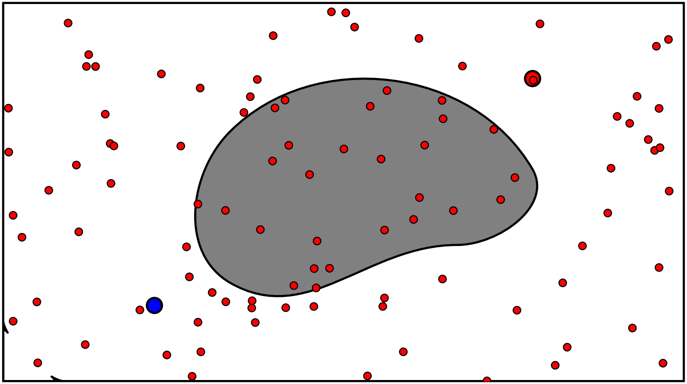
Motion Planning



A valid continuous path

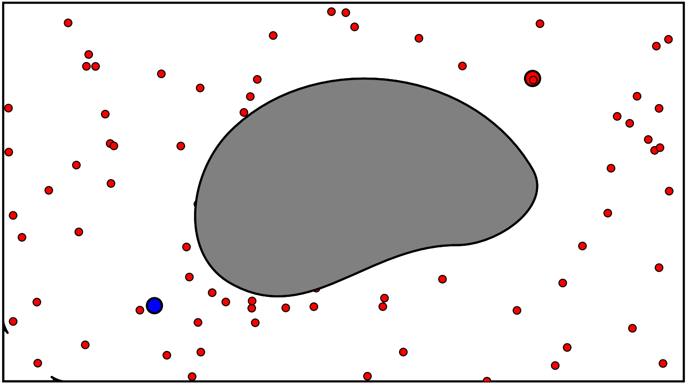
How would we obtain such a path using graph search?

Motion Planning



Randomly pick configurations to be nodes

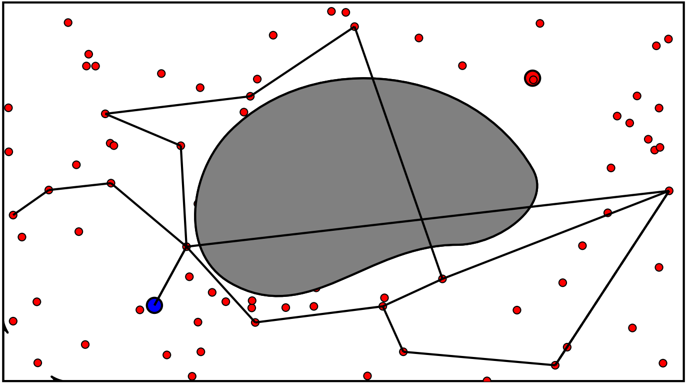
Motion Planning



Randomly pick configurations to be nodes

Discard nodes in obstacles

Motion Planning

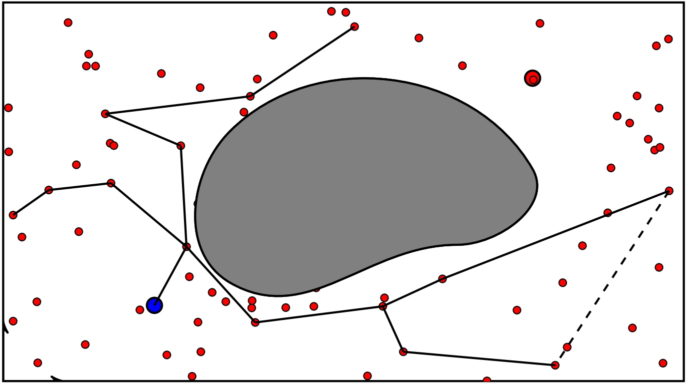


Randomly pick configurations to be nodes

Discard nodes in obstacles

Build graph by adding edges

Motion Planning



Randomly pick configurations to be nodes

Discard nodes in obstacles

Build graph by adding edges that can be physically realized

Motion Planning Continuous Space

Sampling-based motion planning (MP) algorithms define nodes/edges for continuous space and then develop a graph (PRM/RRG) or a tree (RRT).

- PRM: Probabilistic Road Map
- RRT: Rapidly-Expanding Random Tree
- RRG: Rapidly-Expanding Random Graph

Motion Planning Continuous Space

Sampling-based motion planning (MP) algorithms define nodes/edges for continuous space and then develop a graph (PRM/RRG) or a tree (RRT).

- PRM: Probabilistic Road Map
- RRT: Rapidly-Expanding Random Tree
- RRG: Rapidly-Expanding Random Graph

This conversion of MotionPlanning into a graph enables use of graph search algorithms

Motion Planning Continuous Space

Sampling-based motion planning (MP) algorithms define nodes/edges for continuous space and then develop a graph (PRM/RRG) or a tree (RRT).

- PRM: Probabilistic Road Map
- RRT: Rapidly-Expanding Random Tree
- RRG: Rapidly-Expanding Random Graph

This conversion of MotionPlanning into a graph enables use of graph search algorithms

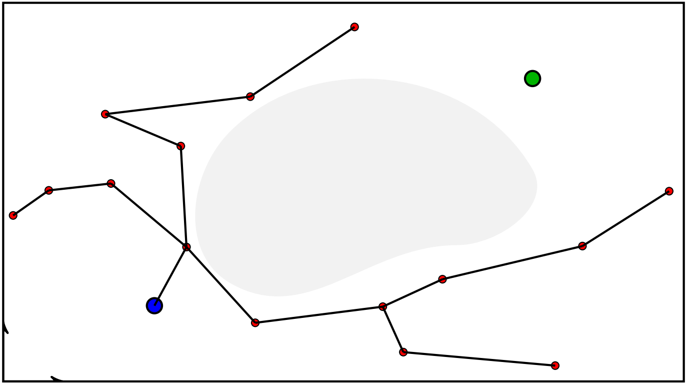
The large variety in sampling-based motion planning algorithms are variations of the two following steps.

1. Randomly sample configurations to create ‘nodes’
2. Use motion models/constraints to ‘connect’ samples

Motion Planning Continuous Space

Every algorithm has

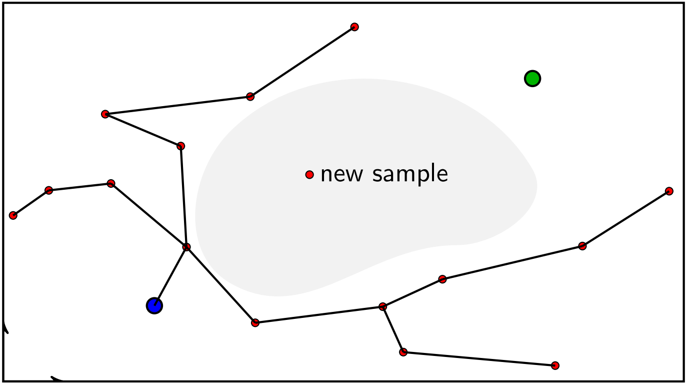
- Sampling mechanism + collision check for creating nodes
- Select existing nodes to try and connect new samples to
- Local planner to check if we can connect new sample with selected existing nodes (dynamics, obstacles along path, local planner, etc.)



Motion Planning Continuous Space

Every algorithm has

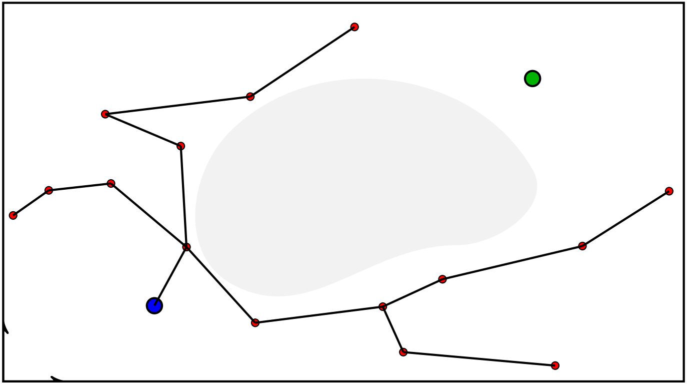
- Sampling mechanism + collision check for creating nodes
- Select existing nodes to try and connect new samples to
- Local planner to check if we can connect new sample with selected existing nodes (dynamics, obstacles along path, local planner, etc.)



Motion Planning Continuous Space

Every algorithm has

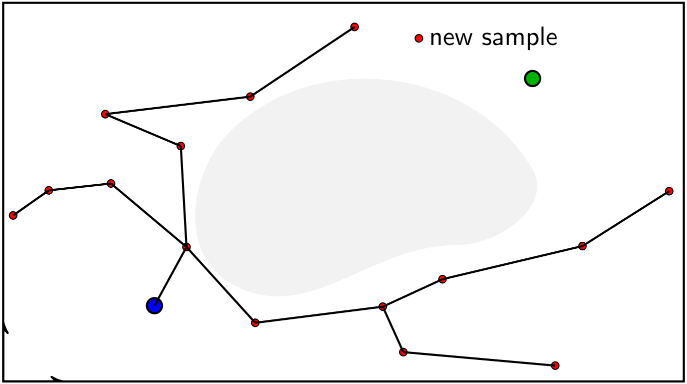
- Sampling mechanism + collision check for creating nodes
- Select existing nodes to try and connect new samples to
- Local planner to check if we can connect new sample with selected existing nodes (dynamics, obstacles along path, local planner, etc.)



Motion Planning Continuous Space

Every algorithm has

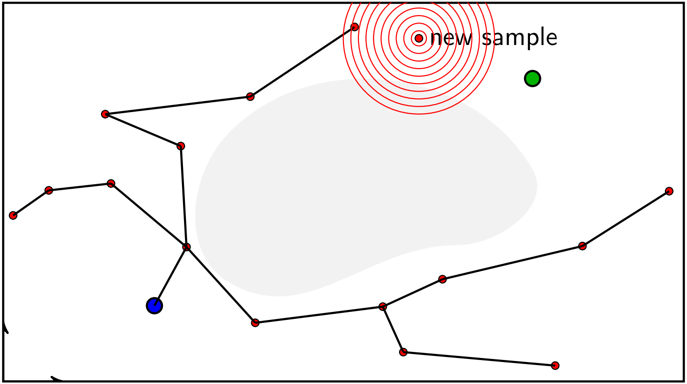
- Sampling mechanism + collision check for creating nodes
- Select existing nodes to try and connect new samples to
- Local planner to check if we can connect new sample with selected existing nodes (dynamics, obstacles along path, local planner, etc.)



Motion Planning Continuous Space

Every algorithm has

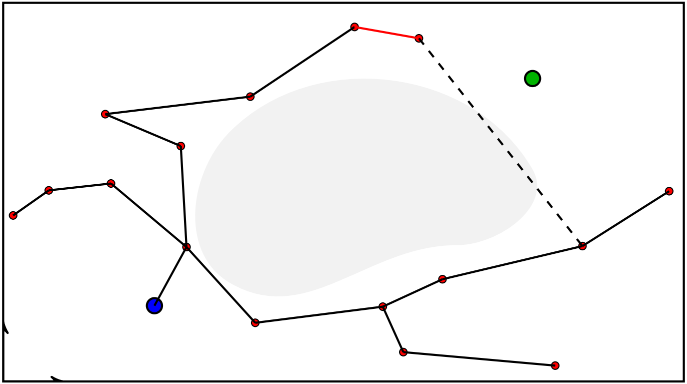
- Sampling mechanism + collision check for creating nodes
- Select existing nodes to try and connect new samples to
- Local planner to check if we can connect new sample with selected existing nodes (dynamics, obstacles along path, local planner, etc.)



Motion Planning Continuous Space

Every algorithm has

- Sampling mechanism + collision check for creating nodes
- Select existing nodes to try and connect new samples to
- Local planner to check if we can connect new sample with selected existing nodes (dynamics, obstacles along path, local planner, etc.)



Motion Planning Continuous Space

Evolution:

- Early methods generated new sample by randomly choosing a control and ‘taking a step’. Poor exploration.

Motion Planning Continuous Space

Evolution:

- Early methods generated new sample by randomly choosing a control and ‘taking a step’. Poor exploration.
- PRM said choose a configuration, use planner to figure out control between them. Often connections failed.

Motion Planning Continuous Space

Evolution:

- Early methods generated new sample by randomly choosing a control and 'taking a step'. Poor exploration.
- PRM said choose a configuration, use planner to figure out control between them. Often connections failed.
- RRT said create new sample, then create sample corresponding to step from existing node towards new sample. Rapid exploration unlocked.

Motion Planning Continuous Space

Evolution:

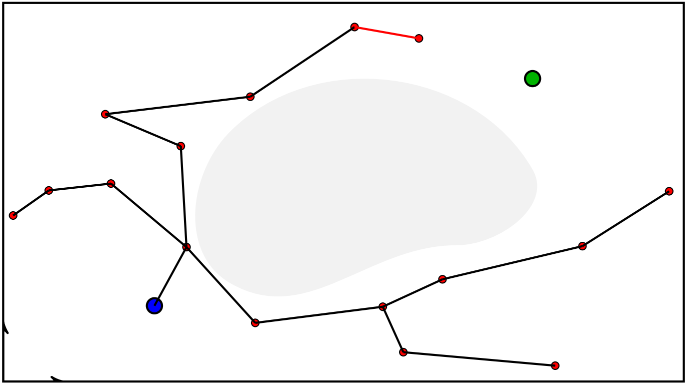
- Early methods generated new sample by randomly choosing a control and 'taking a step'. Poor exploration.
- PRM said choose a configuration, use planner to figure out control between them. Often connections failed.
- RRT said create new sample, then create sample corresponding to step from existing node towards new sample. Rapid exploration unlocked.
- RRT*: rewire connections so that all paths in **tree** are optimal

Motion Planning Continuous Space

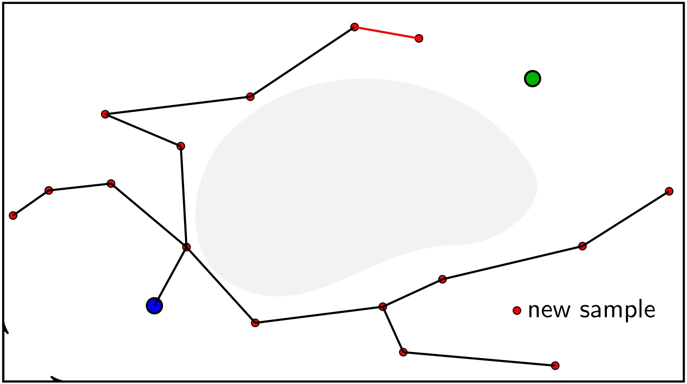
Evolution:

- Early methods generated new sample by randomly choosing a control and ‘taking a step’. Poor exploration.
- PRM said choose a configuration, use planner to figure out control between them. Often connections failed.
- RRT said create new sample, then create sample corresponding to step from existing node towards new sample. Rapid exploration unlocked.
- RRT*: rewire connections so that all paths in **tree** are optimal
- RRG: Connect to multiple neighbors, use shortest-path algos later.

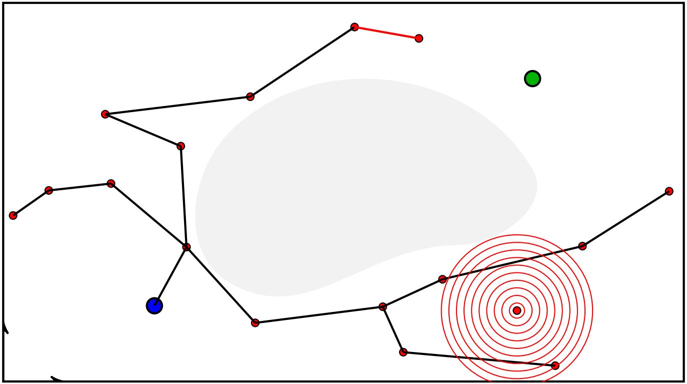
RRT



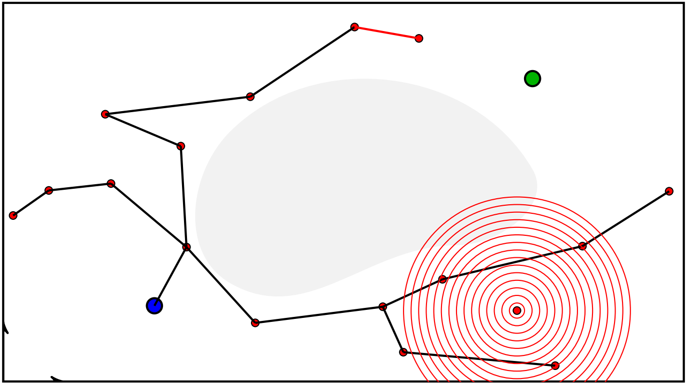
RRT



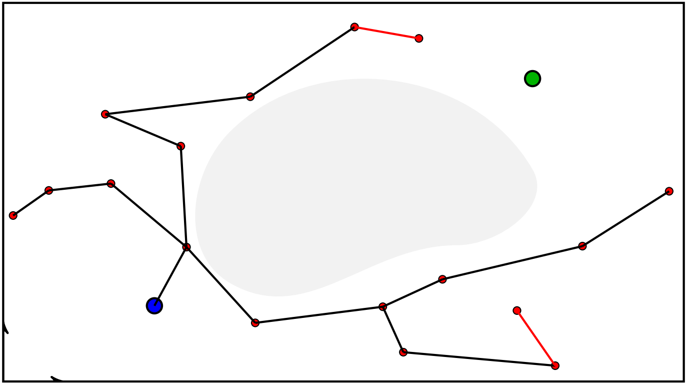
RRT



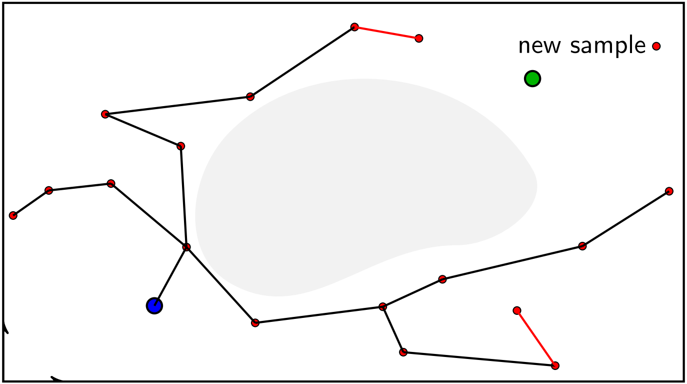
RRT



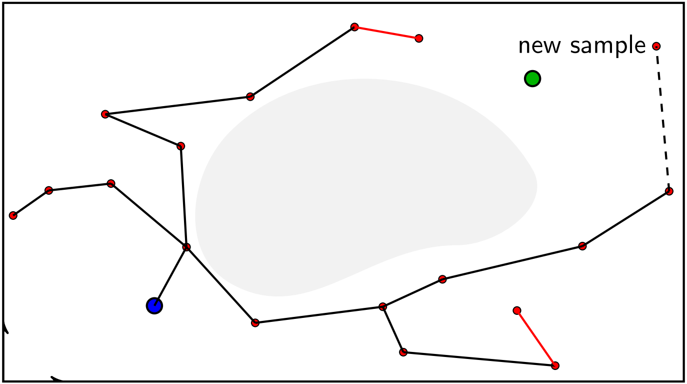
RRT



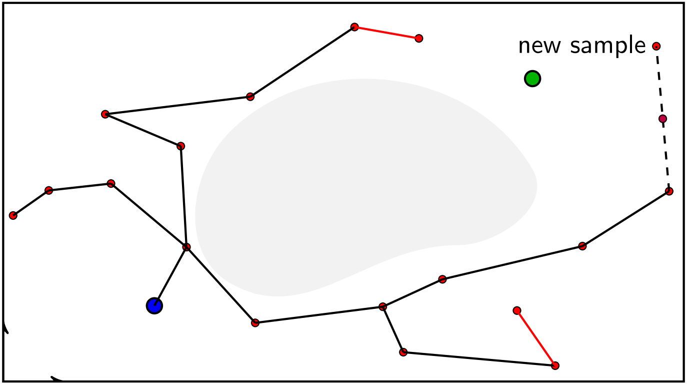
RRT



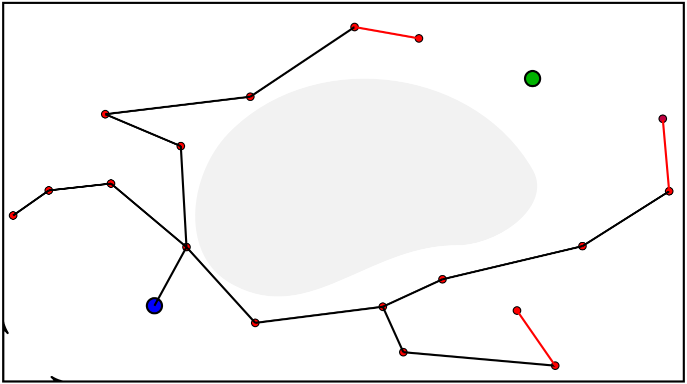
RRT



RRT



RRT



Implementing RRT

You need to implement the following functions that work with a tree data structure and the robot/environment model.

- *sample* (from state space)
- *nearest neighbor* (in state space distance)
- (*local*) *steer* (local planner)
- *collision check* (along steer solution)
- *cost* or *distance*
- *nearest vertex* (in tree distance)