CPSC 8810
Motion Planning

Assigned: Feb 28, 2020
Due: 11:59pm, March 10, 2020

Programming Assignment 4 – Sampling-Based Navigation *(Grading: 0–10 points)*

Problem Description

the fourth assignment, you will implement the probabilistic roadmap method to answer motion planning queries for a 2D robot moving in an environment filled with static obstacles. In the basic requirements, you will assume that the robot is a translating disk, whereas in the advanced extension (extra credits), the robot is a 2D rigid body capable of both translation and rotation.

In the zipped file `prm.zip`, you will find the scenario that you can use to test your code. I have also provided `Python` code, that reads and visualizes scenarios along with other GUI functionality. You can also find helper functions in `utils.py`, and a small library for handling roadmaps in `graph.py`.

Basic Requirements

Your task is modify the `prmplanner.py` file and implement the two phases of PRM:

1. *Construction phase*: Modify the `build_roadmap` function to build a roadmap for the given problem that captures the connectivity of the free configuration space. To construct the roadmap, you have to make a number of choices such as the sampling strategy that you want to employ, how to select nearby vertices for a given sample, and the local planner that you can use to try to connect two vertices and add a new edge to the roadmap. Please see lectures 14 and 15 for more details.

2. *Query phase*: Modify the `find_path` function to plan a path for a given initial configuration to a given goal configuration. Your code should connect the two configurations to the roadmap and then search for a path using a search algorithm such as A*.

You should test your code with the default query, as well as with random ones. You are also welcome to create your own scenes and load them to the framework. Before coding, I suggest you spend some time to familiarize yourself with the `Roadmap` class in `graph.py`, as you will be needing it in both phases. As usual, please include a brief `Readme` file to report certain aspects of your code and implementation decisions that you made.

Extra credits (3pts)

Set the `disk_robot` flag to `False` to change the robot from a disk to a rigid body robot that can both translate and rotate on the 2d plane. While your query phase can be used as is, certain choices made in the construction phase need to be revisited. In the disk case, the Euclidean distance that the origin of the robot travels can be used to determine nearby neighbors. However, the distance function needs to be revised here as the robot can also rotate. One approach is to use a weighted combination of the translational distance and the rotational distance of the robot. A different solution is to use a metric that estimates the displacement of the robot in the workspace (see lecture 15). In terms of the local planner, as the robot is holonomic, you can still use a straight line to connect two configurations and check for collisions. Though, extra care should be taken when interpolating between two angles as you can either rotate clockwise or counter clockwise to reach a target orientation.

Submission

Submit the assignment using Canvas. You can work in pairs if you want to. If you do so, though, please let us know in advance. Along with your code, please upload a `Readme` file documenting the choices that you made in the two phases of PRM.

Help

If you get stuck, please do not hesitate to contact us for help, and stop by during the office hours. We also encourage you to post questions and initiate discussions on Canvas. Your colleagues are also there to help you.