Lab 4: Planning

Pre Lab Due Date: 11/05/2021 @ 11:59pm
Code Due Date: 11/12/2021 @ 11:59pm
Report Due Date: 11/15/2021 @ 11:59pm

This assignment consists of both a written Prelab (due Nov 2 at 11:59pm) and the main portion of the lab (code and report). You will submit all parts of the assignment through Gradescope. Late submissions will be accepted per the standard late submission policy, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza *before the submission deadline* to request an extension if you need one due to a special situation such as illness. The Prelab is worth 5 points and must be completed and submitted individually, and the report and code are each worth 20 points. You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. **If you make use of outside sources, you are expected to cite them in your report.** When you get stuck, post a question on Piazza or go to office hours!

Note: the main lab (like all past and future labs) is a partner assignment! You will complete and submit all Prelabs individually, but submitting the main lab individually is not allowed.

1 Prelab

The goal of the prelab is to connect ideas you are learning in class with the actual tasks you will work on in the projects. These assignments are "pen and paper" assignments in that you should not need to write any code, but if you would like to use digital tools to help explain your answers or improve your own understanding of the problems you are welcome to do so.

1. Discusses how you plan to implement RRT. Some questions you will need to answer are:
   (a) Provide a high level overview of your algorithm
   (b) Will you plan in the configuration space, or the workspace? If you are planning in the configuration space will you plan in the full $\mathbb{T}^7$ (the 7-torus), a subset considering the joint limits, or some lower dimensional subspace? Justify your decision.
   (c) How do you plan to check for collisions between the robot and the environment?
   (d) How will you handle the volume of the robot's links when checking for collisions? Describe what approximations you might be able to make.
   (e) Would you prefer that your solution be conservative (sometimes saying there is a collision even when there isn't) or liberal (sometimes saying a collision will not occur even when it does)? Why?
   (f) How do you plan to handle self-collisions between the links of the robot?

   This lab involves more design decisions and fewer strict correct answers than the previous labs - not everyone's solutions will look the same. The intent is that you seriously consider each of these concerns and use your engineering judgment to justify your choices.

Submit the Prelab on Gradescope. **You must complete and submit your prelab assignment individually**, and your writeup should be clear and concise.

2 Simulation Setup

To update your own private fork of the `meam520_labs` repo for Lab3 please do the following:

```
$ cd ~/meam520_ws/src/meam520_labs/
$ git pull upstream master
```

This is going to look at the public TA repository and grab the code stubs and launch files that we've added since the last lab. Occasionally when running this command, you might encounter a merge conflict. We don't expect that to occur, but if it does, don't hesitate to reach out to the TAs for help.

3 RRT

In this lab you will implement a Rapidly-Exploring Random Tree (RRT) planner for the Panda arm. Although stub code is provided, the amount of scaffolding is significantly less than in prior labs, so be prepared to make your own decisions on how to structure your code in a modular and testable manner.

3.1 Tasks

Implement your planner in the file `rrt.py`. The function takes in a map struct (see below) as well as two configurations q_{start} and q_{goal}, which are the starting and ending configurations of the robot, respectively. It should return an $N \times 7$ array containing the sequence of joint variable values along your trajectory that you send to the Panda robot.

To keep your code organized,

- We **suggest** you separately fill out and call a function `isRobotCollided` to detect collisions for a robot configuration q.
- We have provided a function `detectCollision` (see below) to help you with this check. You may change the function `detectCollision` if it does not serve your purposes in its basic form. Include a copy of the `detectCollision` file in your submission.

3.2 Provided code

In order to help you we have provided the following classes:

1. `loadmap`: loads in a map from a text file and stores it as a map struct. Usage: `map = loadmap(filename)`, where `filename` is a string indicating the location of the text file.

   The map struct contains two entries: obstacles and boundaries. Both of these are stored as `[xmin,ymin,zmin,xmax,ymax,zmax]` arrays.

2. `detectCollision`: Detects collision between a line segment and a rectangular prism. Usage: `iscollided = detectCollision(linePt1, linePt2, box)`, where `linePt1` and `linePt2` are $N \times 3$ arrays, with each row $[x, y, z]$ being the start/end locations of the line segment, and `box` is an array of the form `[xmin,ymin,zmin,xmax,ymax,zmax]`. Returns a Boolean `iscollided`, which is true if the line segment intersects with the box and false otherwise. Keep in mind that this assumes a **line of zero thickness**.

   - `detectCollision` is vectorized to accept any number of line segments, but only one box per call. If you call it with multiple lines, `iscollided` will be row-aligned with the line segments.

In addition to the above code we have provided you with a maps folder containing maps.

4 A*

For this lab we will be providing a complete implementation of the A* algorithm. In order to avoid biasing your RRT design decisions we will be releasing the A* code after the RRT code is due, but before the report is due. When we release the A* code we will provide documentation on how to use it. Design a few tests (environments, start and end positions) to check whether your planner works. Explain why you have chosen those tests. Run each test several times to evaluate the performance of your planner and compare it against the A* planner we have provided.

Note: try making these comparisons using simpler maps. The more challenging maps may be hard for A* to solve in a reasonable amount of time (which is part of why we use randomized approaches like RRT).

5 Testing

We recommend that you test your solution separately from the simulator first.

Once you are happy with the performance of your planner, it's time to test in Gazebo. Launch the simulator with

```
roslaunch meam520_labs lab4.launch map:=map1
```

Note: there should be no space anywhere in `map:=map1`. You can replace `map:=map1` with the name of any of the maps in the `meam520_labs/maps` folder, which is also where you can create new maps. Now run the test script via

```
cd ~/meam520_ws/src/meam520_labs/labs/lab4
python rrt_demo.py 1
```

where you can replace 1 with any number to run that numbered test. To add more tests, modify the starts, goals, and maps list with the relevant start, goal, and map.

This script will first move your arm to the start position, then plan with your RRT algorithm to the goal, before executing the path. For map 1 and map 2 we have provided an example start and end pose. For the remaining provided maps, you are expected to come up with your own interesting tests. You are also expected to come up with your own tests for map 1 and map 2.

6 Report

You will submit a written report describing your work in this lab. **The report must be no more than 8 pages.** You are free to use whatever typesetting workflow you are comfortable with, but we highly recommend using LaTeX. Handwritten submissions will not be accepted. If you need to sketch some diagrams by hand, this is okay. Your report should address all of the tasks listed in this section.

6.1 Methods

Summarize your planning strategy. If you are making simplifying assumptions, justify them. (Note: you should have already described your initial plan in your Prelab. The purpose of this section of the report is to describe the planner you ended up actually implementing, and in particular to make note of any changes from your initial plan. The description should stand alone, since the TA reading your report may not be the same one who read your Prelab. If you decide that your Prelab description was sufficient, you can simply reproduce its content here.)

6.2 Evaluation

Design a few tests (environments, start and end positions) to check whether your planner work. Explain why you have chosen those tests. Run each test several times to evaluate the performance of your planner and compare it against the A* planner we have provided. Consider questions such as:

- How often does your path planner succeed?
- How long does it take for your planner to find a path (running time)?
- When your planner finds a path, is the path the same over multiple runs?
- How do the answers to these questions change for different environments or variations in your implementation?

6.3 Analysis

Discuss any conclusions on your planner based on the evaluations.

- What kinds of environments/situations did your planner work well for? What kinds of environments/situations was it bad at?
- What issues did you run into or what differences were there between your expected and experimental performance?
- What changes would you make to your implementation if you had more time with the lab?
- What differences do you notice between the grid based planner (A*) and the random search (RRT)?

7 Submission via Gradescope

- **Group Submission:** Each group should only submit once to Gradescope per Gradescope assignment. While going through the submission please ensure that you **add your group members to the submission**. Failure to do so could impact the grades of all parties involved.
- Please submit `RRT.py`, and any other files needed to run your code to the Lab 4 Code on Gradescope (including those from previous labs which you rely on). Once submitted, we will run automated tests to grade your code. You may submit as many times as you want, but any attempts to try to figure out what test cases we are running will result in your code getting a 0.
- Please submit your written report as a PDF to the Lab3 Report assignment on Gradescope.

IMPORTANT, please submit the correct thing to each assignment. Don't submit the code to the report assignment. Don't submit the report to the code assignment.