
Assignment III – CSC376 – Fall 2019

Fundamentals of Robot Design
Department of Mathematical and Computational Sciences
University of Toronto Mississauga

Abstract

This assignment covers Module 7 - Motion Planning. It consists of two parts. In the first part, a serial manipulator has to move from one position to the next target position while avoiding collision with the environment. In the second part, write a concise report which summarizes your approach and results. By the submission of this assignment you should:

- be able to implement a widespread probabilistic complete motion planner.
- understand the chosen motion planner.
- use the API of the V-REP simulation environment.



Remember to write your full names and student numbers prominently on your submission. To avoid suspicions of plagiarism: at the beginning of your submission, clearly state any resources (people, print, electronic) outside of your group, the course notes, and the course staff, that you consulted.

For Assignment II and Assignment III you are allowed to work in groups of 2 students or by yourself. Please note that the group cannot be changed between those two assignments. For assignment handling, group forming, and submissions use MarkUs:

<https://mcsmark.utm.utoronto.ca/csc376f19>

Please upload your submission as a single .pdf file alongside with an updated git-repository containing your code on MarkUs. By uploading, you confirm that this is your original work without using any references other than the listed ones.

Answer each question completely, always justifying your claims and reasoning and explaining your calculation/derivation. Furthermore, use meaningful and explanatory comments to go along your submitted code. Your solution will be graded not only on correctness, but also on clarity. Answers that are technically correct but hard to understand will not receive full marks. Mark values for each question are contained in the [square brackets].

Dean W 	Roman H 	11/29/2019
_____ Name 1	_____ Name 2 (if applicable)	_____ Date

Due Date: Nov 29, 2019 (11:59pm)

CSC376: Assignment 3: Due Nov 29, 2019 by 11:59pm

Roman H
Dean W

Nov 29, 2019

Part I: Probabilistic Complete Motion Planning

The probabilistic roadmap planner is implemented inside *assignmentIII.py* and *Node.py*.

Part II: Short report

Running the Algorithm

The PRM planner can be invoked by passing the the desired start and goal position numbers as positional arguments in their respective order. The positions are referenced by their assigned position numbers visible in the table below. The following command describes the positional arguments:

```
python assignmentIII.py <start> <goal>
```

Example for path planning from position 1 to position 3:

```
python assignmentIII.py 1 3
```

Requirements

Besides the V-REP dependencies and python 2.7, our implementation depends on the following python packages which can be installed as follows:

```
pip install scipy  
pip install numpy  
pip install dijkstar
```

Found V-REP Bug/Issue

We have noticed that our simulation runs faster when we are interacting with the scene using the mouse. We mentioned this to the TAs and they suggested that the V-REP process could be running faster because it is inheriting a higher priority from the mouse. To recreate this, click and hold anywhere in the scene while the simulation is running on a Linux system.

Motion Presentation (Video/GIF)

Video #	Start Config	Goal Config	Filename
1	Position_1	Position_2	"pos1_to_pos2.*"
2	Position_1	Position_3	"pos1_to_pos3.*"
3	Position_1	Position_4	"pos1_to_pos4.*"
4	Position_1	Position_5	"pos1_to_pos5.*"
5	Position_2	Position_3	"pos2_to_pos3.*"

Our unique motions are presented as both videos and as GIFs. The table above outlines the pairs of start and goal configurations, as well as their screencapture filenames. All media files are located in the following directories:

a3/video/*.avi
a3/video/gif/*.gif

NOTE: all of the unique motions are also presented in the README.md as looping GIFs when viewed in a markdown compatible viewer.

Approach

We chose to implement the Probabilistic Roadmap planner because it is well suited for static obstacles such as the ones we are dealing with. PRM has the advantage of being a multiple query planner over the single query capabilities of Rapidly-exploring Random Tree planners (RRTs). This means that if we wanted to provide not only the shortest found path, but also alternative found paths, then we could do so with fairly minor modifications to our implementation.

Discussion

Our approach begins by creating the subroutines outlined in Part I. We combine these functions together when we are ready to plan our robot's motion. First, our *planMotion(start_config, goal_config)* function randomly samples values for each joint (respecting their joint limits), and checks if this configuration results in a collision or not. If it is collision-free, then we add this configuration to our roadmap graph R , as well as this node's data to a KD-tree.

Once we have checked a specified number of samples of configurations, we find the K nearest neighbours for each node in our graph R . We then check if we can connect each node to each of its neighbouring nodes, and if we can, we check if this path is also collision-free. This results in our roadmap graph R having nodes for collision-free configurations, and edges for collision-free paths between those configurations.

Finally, our *planMotion* function performs Dijkstra's shortest path algorithm (a form of the A^* algorithm without a heuristics function to bias towards the goal) to find a path between our starting node and goal node. If a path in the graph is found, we return each node in this path, and then we pass each of the node's configuration data to the V-REP scene so that we move the robot along the collision-free path from our starting position to the end position. If a path is not found, then we repeat the process outlined above until one is found.

Improvements

Since we are randomly sampling configurations to populate our graph, it is possible that we don't sample a set of configurations that return a path from the start position to the end position. This could be improved by sampling points with a general bias towards the goal position, since many random configurations that are sampled are not used in our path. We could save computation time as well as having a higher probability of finding a path by biasing our samples towards the goal. This was a difficulty for us because it would take many attempts of running our algorithm to find a solution with our motion planner. This was the reason that we re-iterate the entire PRM algorithm with new samples if we do not find a path on the current attempt. If we did find a path, it is clear (seen in the videos) that the path is not optimal and ultimately it is wasted computation time.

Another improvement that could be made is improving our collision checker to improve run-time. For example, we would like a way to mathematically check for collisions so that we do not need to actually move the robot in the V-REP scene and check if it collides with scene data. This would greatly reduce the overall time our algorithm takes to complete, since it takes minutes to sample an amount of configurations that only sometimes gives us a solution (since it is random sampling and not biased towards the goal, which A^* with heuristics provides).