COMP3702/COMP7702 Artificial Intelligence (Semester 2, 2020) Assignment 2: Continuous motion planning in CANADARM

Name: Anshuman Mander

Student ID: 45123209

Student email: a.mander@uqconnect.edu.au

Note: Please edit the name, student ID number and student email to reflect your identity and **do not modify the design or the layout in the assignment template**, including changing the paging.

Question 1

Configuration Space (C-space):

Configuration space is the 2D workspace bounded by [0,1] X [0,1], which is populated by rectangular obstacles. The configuration space defines the valid robot's configuration within workspace. A robot configuration has three main components – arm length, arm angle and joint coordinates. The arm's length in bounded by min and max length specified in the map file. In general, 0 <= arm length (l) <= 1. Similarly, all angles lie between $-11 \times \pi/12$ <= arm angle (θ) <= $11 \times \pi/12$, except for grappled end angle which can vary from -π to π. Lastly, all joint coordinates (x, y) must lie within the confined work space (to ensure the arm lies within the workspace), i.e. 0 <= x <= 1 and 0 <= y <= 1.

Forbidden Region: Wherever in the workspace robot arm either collides with an obstacle or there's a self-collision, is defined as the forbidden region.

Free Space: All workspace excluding any forbidden region (C-space – Forbidden region).

- Search Methods – Probabilistic Roadmap (PRM)

  PRM is current being used to search in continuous space with state graph construction interleaved with graph search. Here, sampling strategy is used to generate valid samples which are then connected using connection strategies. After generation of fixed number of samples, the graph is scanned to find valid paths and tested to see if path's line segment is valid or not. This process of sample generation and path testing continues till a valid solution to goal is found.

Question 2

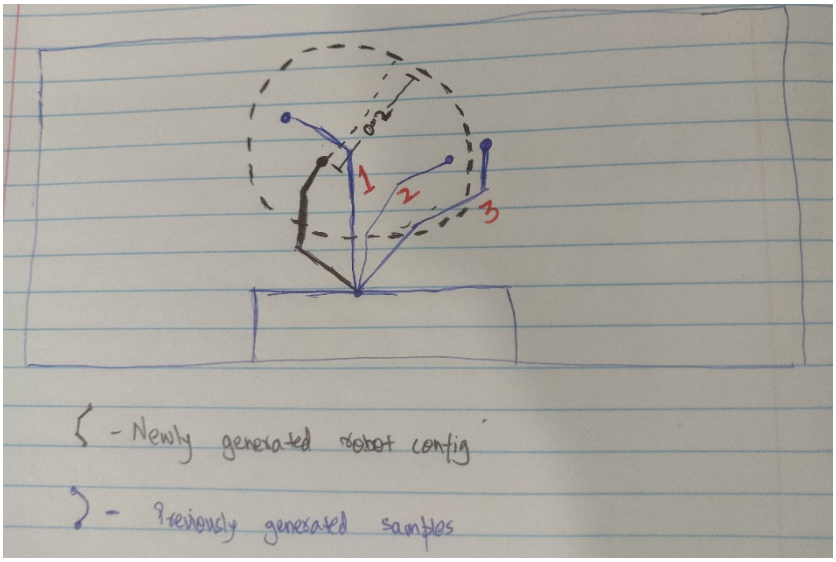
Sampling strategy:

Uniform random sampling strategy is used to generate random robot configurations. This strategy depends upon uniform distribution that original PRM uses. Here, random angles and random arm lengths (within the specified bounds) are generated and a new robot configuration is created using them. After creation, individual collision checking takes place. If collision is found, the process starts again otherwise the robot config is returned.

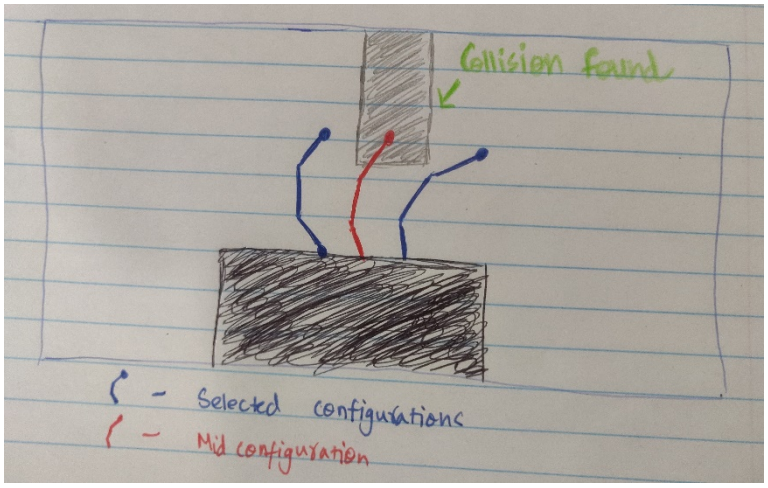
Connection Strategy:

There are two connection strategies being employed in the solution.

1. The first one is the distance checking where, whenever a random sample is generated, it's distance from previously generated robot configurations is checked. The end effector of newly generated robot config is used as the starting point and all other robot configs whose end effector is within 0.2 units distance of starting point is selected (0.2 units is selected as it gave the best results after trial and error with various distances).



   In the image above, black dotted circle represents the area under which all the robot configurations will be selected for connection. Robot configs 1 and 2 are selected but 3 isn't (even though bit of its arm lies within the circle) because the distance measured is calculated from end effector of new config (black config) to end effectors of previous configs (blue configs). All selected configs (in this case, 1 & 2) are then selected for second round of testing.
2. In the second round, a lazy collision check takes place between newly generated config and previously selected configs (those that passed first test). In lazy check, the selected two configuration's mid configuration is created and tested for collisions. If a collision free mid config is found, the two configs are connected together. In the image below, there's collision in the mid configuration so they aren't connected together.



Lastly, when interpolating path for final solution, if there's collision found between two configs, their connection is manually removed. This removal of connection ensures that when searching the graph for path next time, invalid paths aren't tested as valid solutions again.

Configuration Validity:

Configuration validity takes place with the help of testing methods in tester.py support code. Here, the configuration is tested for obstacle collision, self-collision, environment bounds and angle constraints. Since sampling strategy generates samples within given angle and arm length constraints, testing those aren't useful in individual collision check. Angle constraint is only used to ensure the no joint angle is less than $15°$ when interpolating between two robot configurations. Finally, primitive step constraints are maintained while interpolating (i.e. only primitive steps are taken).

Checking if a line segment in C-space is valid or not:

Interpolate_path(config_1, config_2) function is used to check if a line segment is valid or not. Given two robot configs, this function finds the number of steps required to interpolate and uses that as the base to take a primitive step. After each primitive step, the new robot configuration is checked for configuration validity. If there's any collision, None is returned and the path is marked as invalid (connection removed as mentioned before) otherwise, an array of robot configs are returned representing the valid line segment in C-space.

Question 3

The program's only able to solve the testcases with single grapple points. This is due to no bridge configuration being present, any problem spec with more than one grapple points isn't solvable. Within the single grapple point problems, the program is able to accommodate different arm lengths, some obstacles, narrow passageways and passageways with corners.

As an illustration of being able to handle different arm lengths, the solver can solve testcase 4g1_m1 which is similar to 3g1_m1 but with much longer arm lengths. Here, the random generation of arm angles and arm lengths does the trick and helps solve the testcase. No particular gimmick is being used except for sampling of different spec arm lengths. This sampling allows the bigger arm to subside in the narrow passageway (with time when samples are generated) and pass through. Though it is not the most efficient method, it still does the trick. Usage of heuristic sampling would have been more appropriate to suite testcases with bigger arm lengths.

Similarly, the handling of corners is also done through the random arm length generation, that is able to accommodate any corner stitch ups. As an example, the program's able to solve both 3g1_m1 and 3g1_m2. In both testcases, the arm lengths are almost the same with the only difference being that in 3g1_m2, the arm is wrapped around corner.

Lastly, since 4g1_m1 is being solved but 4g1_m2 doesn't solve (in time), it is indicative of the program having struggle with solving problem spaces with multiple obstacles. Here, single segment generation would have been useful to accommodate more obstacles.

Overall, basic strategies are used for handling testcases therefore, only basic testcases pass. In simple cases, mostly all problem specs. The program fails whenever one of the aforementioned accommodations becomes too complex.