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

Name: Chenkuan Ge

Student ID: s4596347

Student email: chenkuan.ge@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 (Complete your full answer to Question 1 on the remainder page 1)

Ans: The Configuration space of the Problems is the set of all possible Canadarm configurations. The C-space contains the coordinate x and y of the start points, angles and lengths of each segment and the two grappled state to test the canadarm used which grappled points.

For example, if a canadarm has n segments, which means the configuration space should have 2n + 4 dimensions. There has n angles and lengths, two start points coordinate and two state of grappled point.

The method I used was Probabilistic Roadmap (PRM) and the whole method has 4 key components.

Sampling strategy, interpolation, Collision check and Connection Strategy.

Firstly, sampling a random configuration of robot arm, the angles should be in the permitted range(-165, 165) and the length of each segment should be in the correct range between minimum length and maximum length of segment. And then check the validation of the random samples if they have any collisions with obstacles and itself. If everything is all good, add this sample into the empty graph. Looping to do these steps until get enough number of samples.

Secondly, connect each node with their nearest neighbours. If there has no collision of this connection, add the config to new neighbours.

Thirdly, using the BFS algorithm to find the correct path within the existed neighbours.

Last, get the primitive steps by the configurations we get from the BFS. if we get two configurations and we can calculate the lengths and angles between them, and then divided by the limited step 0.001 unit. We could get the next configuration direction and the primitive step based on this direction. The little steps on x and y separately and then get the configuration within 0.001 unit.

Question 2 (Complete your full answer to Question 2 on pages 2 and 3, and keep page 3 blank if you do not need it)

Ans:

Sampling Strategy:

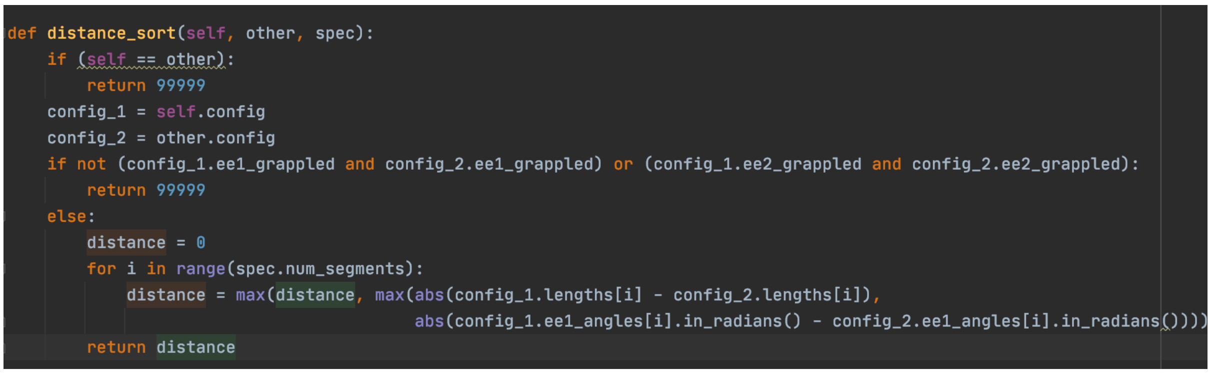
I used uniform random sampling strategies, and generate new samples by choosing angles, grappled points and lengths randomly. The Angles should be in the range from -165 to 165, and the lengths should be in the given range of minimum and maximum lengths. And then use the function make_robot_config_from_ee1 to make a new configuration which start from ee1 grapple point. At the same time, I tried to create a new configuration by method make_robot_config_from_ee2 to make an opposite configuration of first one to deal with the situation to switch grapple point.

At the same time, I used one more method to sample the configuration which could connect two grapple points, and this would solve the problem that can not connect other grapple points. I tried to random the first two segments and then calculate the last segment. We could use Gaussian distribution here and the key point is to limit the existed conditions and then use loop to get a correct configuration in a small but high possible region to select configuration.

Furthermore, it should be possible to add some limitation on the uniform random sampling strategies. And then the samples would be controlled by sampled at the region which has more possibility to gain the goal configuration. This way could decrease the time and effort.

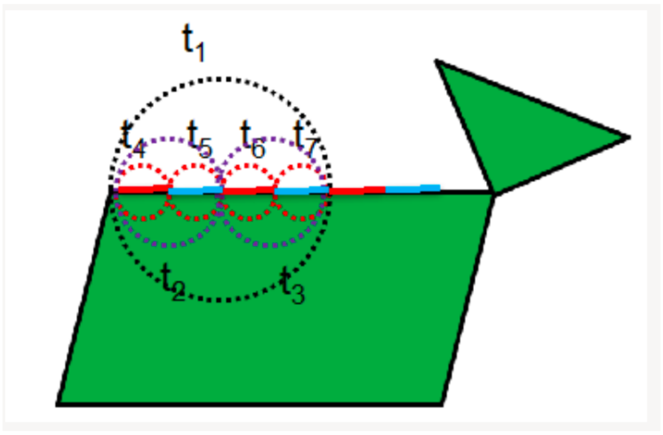
Connection Strategy:

Get each sample from the suitable samples set which have samples from sampling strategies. If the sample has no collision, connect the sample to existing vertices in the graph using valid edges. The connection strategy compares the two valid nodes and get the middle points between two configurations. For this pair of configurations get their middle points at least n times, like to get their sub-set in their ways to add their neighbours. And the list of two points, was sorted by the distances which sort from small to biggest. And the distance was calculated by both angles and lengths between two configurations.



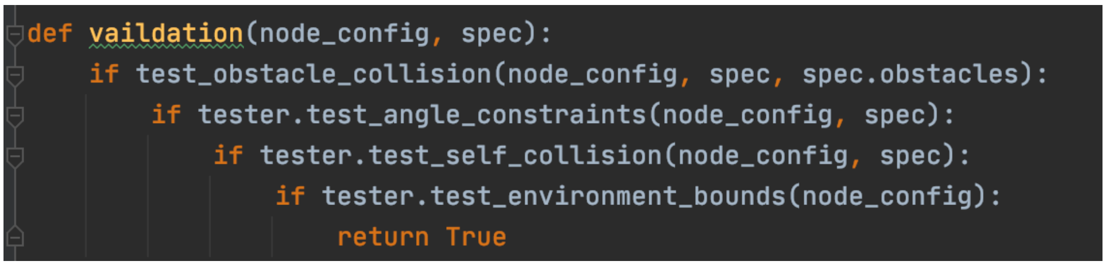
```
def distance_sort(self, other, spec):
    if (self == other):
        return 99999
    config_1 = self.config
    config_2 = other.config
    if not (config_1.ee1_grappled and config_2.ee1_grappled) or (config_1.ee2_grappled and config_2.ee2_grappled):
        return 99999
    else:
        distance = 0
        for i in range(spec.num_segments):
            distance = max(distance, max(abs(config_1.lengths[i] - config_2.lengths[i]),
                                         abs(config_1.ee1_angles[i].in_radians() - config_2.ee1_angles[i].in_radians())))
        return distance
```

To connect the samples, there could be divided by many points between two configurations, and we could check if it was available, no collision with obstacles and itself. If everything is all good, use add_connection function to add two configuration as neighbours with each other. For example if I tried to divide 2 times, and I could get 5 configurations like the figure below.



Checking if a configuration is valid or not:

For checking the configuration, I used the method which provided by tester.



```
def vaildation(node_config, spec):
    if test_obstacle_collision(node_config, spec, spec.obstacles):
        if tester.test_angle_constraints(node_config, spec):
            if tester.test_self_collision(node_config, spec):
                if tester.test_environment_bounds(node_config):
                    return True
```

These four functions could test if the configuration is in collision with obstacle. (test_obstacle_collision). Checking the angle is in valid range that between -165 and 165 or not. (test_angle_constraints). And check if the segments have collision with themselves. (test_self_collision) and test the segments if out of environmental range or not. (test_environment_bounds).

If all situations are all good, it will return true and else it will return False.

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

When it comes to interpolation, if the middle points append in the obstacles region (the non free region) we should remove this neighbour and try other ways. Discretise the line segment into small segments. Take the mid-point to represent the small segment and if the mid-point is in the collision region return false to represent, this connection is not valid.

By testing, I tried to test the state of configuration if it in collision within many steps, because each step we only move 0.001 unit that it is small. So we could test ten steps(0.01) to check if it in collision. In this method, it would decrease the run time.

Question 3 (Complete your full answer to Question 3 on page 4)

Scenarios would be easy to solve:

1. **C-space has less dimensions.**

   If there has only one segment or one grapple point. It would be easy to get the sample in such less time.

2. **More open free region.**

   Having more free regions which contain the correct paths.

3. **Less forbidden regions**

   Getting less forbidden regions would affect the effective of sampling.

Scenarios would be difficult to solve:

1. **Have more dimensions**

   If the configuration has more needed segments and more grapple points. For example, there has 10 segments and 10 grapple points. It would be much harder to getting suitable path in the enough time.

2. **Have less free regions and more forbidden regions**

   If free regions is less and forbidden regions are more, it would be hard to sample.

3. **Have more narrow space.**

   When it comes to a narrow cave for segments to pass, it would be hard to sample an available configuration in the narrow space. We have to find the configuration which can pass such tight space so that the running time would be increased. It is hard to get the suitable sample in the narrow space and that we could get a right neighbour and pass to the goal in this region.

   For example, as shown in testcases the example of 3g2_m1.txt and 3g2_m2.txt

   The second space has a narrow space for robot to move through, it would be much easier for me to pass the first case, but the second one is much harder.

Overall, to be ideal, if we can get more samples like 1 million at the first, it would be easy to find goal. On the contract, this situation will cost me too many times. Similar with that, if I could get as more neighbours as I can, the problem would be easy to solve, because there will exist more available path, but it will cost much time by breath first search. More samples and easier to solve problem, however, the cost like time would be increased.

To optimize the search, we could optimize the method of sampling strategy by Gaussian distribution. It would be easy to get sample at the good region, closer to the goal.