HW 3: Motion Planning, Sample Based

CS 4610/5335: Robotic Science and Systems (Spring 2023)

Report By: Francis Jacob Kalliath

M0)

The collision_check method, discretize the links into a group of points and compute the 3D coordinates of three points along the robot arm. It then determines whether any of these points fall within any of the spheres in the environment's collision radius and returns a boolean value indicating whether or not a collision took place.

If a resolution is not specified, the check_edge function first assigns a default value of 11 for the resolution. The robot is then generated in "n" different configurations, where "n" is equal to the resolution, along the straight path between q start and q end. The code then loops through every configuration of the robot along the edge after initializing the output variable "in collision" to false. The "check collision" function is used in each configuration to determine whether the robot has collided with any of the spherical obstacles. The function sets "in collision" to true and exits the loop if a collision is found. In order to show whether the edge is collision-free, the method finally returns "in collision".

The two main issues with the collision_check method is:
1)The first problem has to do with the function's resolution parameter, which controls how many points are used to look for collisions. In the absence of human input, this option defaults to 11, however in more complicated situations with smaller or irregularly shaped objects, this value might not be sufficient to reliably detect collisions. Therefore, in such cases, the system can fail to detect probable collisions.
2) The elbows and other projecting areas of the links are not checked for accidents. The only thing being checked are the links.

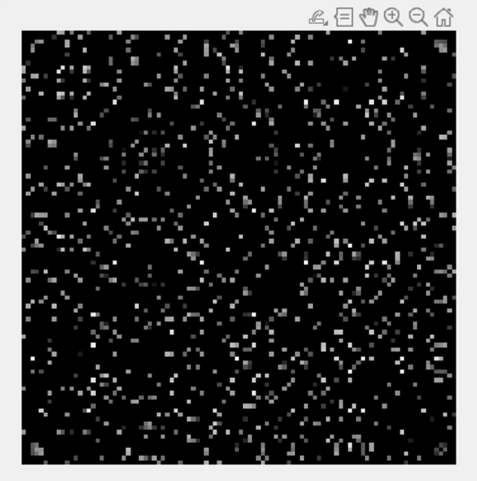
The following are a few potential issues with the check_edge function:
1)First, it's possible that the path planning's resolution isn't high enough to detect a collision. The number of locations along the straight-line path that are checked for collisions depends on the resolution. The function might not detect a collision if the resolution is set too low. The default resolution of 11 might not be enough for some applications.
2)We are seeking for collisions along straight lines in the configuration space. This might not be an accurate estimation.
3) Links' elbows and other protruding portions are not examined for accidents. The links are the only item being examined.

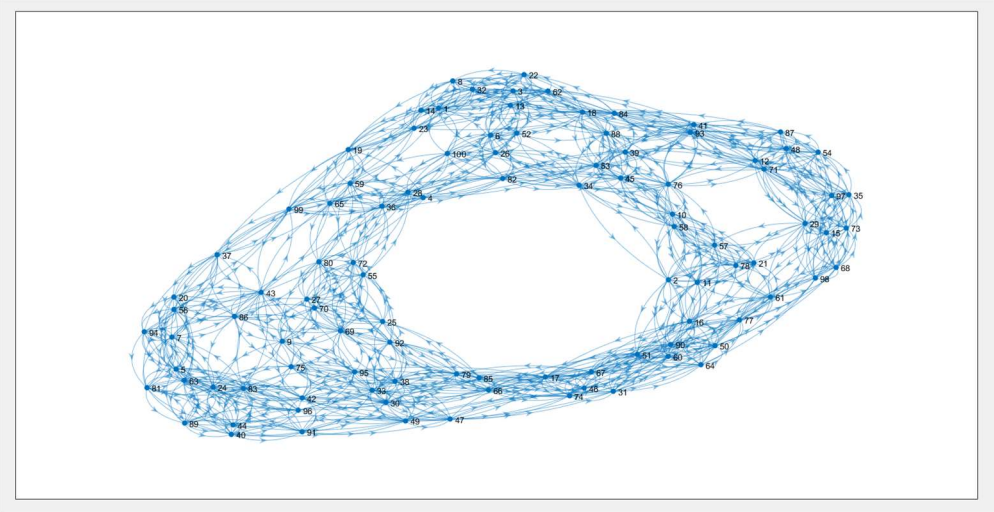
M1)

```
>> hw3_motion(1)
Number of samples: 100
Number within bounds: 100
Number in collision: 15
fx >>
```

Generated 100 samples out of which 15 had collisions.

M2)





The above image shows the obtained adjacency matrix and the plotted diagraph for the adjacency matrix.

M3)

```
>> hw3_motion(3,samples,adjacency);
   -0.2548   -0.7917         0   -1.0527
   -0.9363   -1.4951         0   -1.2261
   -0.7914   -1.6999         0   -1.3996
   -0.4551   -2.2606         0   -1.9321
   -0.2661   -2.5080         0   -2.2913
    0.2669   -2.6758         0   -2.8508

Path found with 6 intermediate waypoints:
         0   -0.7854         0   -0.7854
   -0.2548   -0.7917         0   -1.0527
   -0.9363   -1.4951         0   -1.2261
   -0.7914   -1.6999         0   -1.3996
   -0.4551   -2.2606         0   -1.9321
   -0.2661   -2.5080         0   -2.2913
    0.2669   -2.6758         0   -2.8508
         0   -3.0000         0   -3.0000
```

The above command window data shows that there are 10 intermediate waypoints.

M4)

```
Path found with 19 intermediate waypoints:
         0   -0.7854         0   -0.7854
    0.0723   -0.7201         0   -1.0156
   -0.0800   -0.8307         0   -1.1802
   -0.1437   -0.8662         0   -1.4193
   -0.3598   -0.9050         0   -1.5388
   -0.5900   -0.9375         0   -1.4467
   -0.6013   -1.1872         0   -1.4508
   -0.5647   -1.3717         0   -1.6155
   -0.6782   -1.2542         0   -1.8048
   -0.8212   -1.2183         0   -2.0067
   -0.7278   -1.4208         0   -2.1196
   -0.6345   -1.6234         0   -2.2325
   -0.6784   -1.8111         0   -2.0734
   -0.5758   -1.9909         0   -2.2135
   -0.4158   -2.1170         0   -2.3584
   -0.3268   -2.3060         0   -2.4957
   -0.2378   -2.4950         0   -2.6330
   -0.3413   -2.5801         0   -2.8441
   -0.1898   -2.7665         0   -2.9133
   -0.0383   -2.9529         0   -2.9825
         0   -3.0000         0   -3.0000
```

The above command window data shows that there are 18 intermediate waypoints.

M5)

```
Command Window
  Path found with 18 intermediate waypoints:
           0   -0.7854         0   -0.7854
      0.0833   -0.7918         0   -1.0210
     -0.0836   -0.7976         0   -1.2070
     -0.2723   -0.9099         0   -1.3265
     -0.5093   -0.9764         0   -1.2828
     -0.5883   -1.2003         0   -1.2044
     -0.6764   -1.4341         0   -1.2117
     -0.6027   -1.6699         0   -1.2501
     -0.6482   -1.7378         0   -1.4863
     -0.5701   -1.8899         0   -1.6687
     -0.5774   -1.9981         0   -1.8940
     -0.4872   -2.1546         0   -2.0668
     -0.3969   -2.3111         0   -2.2396
     -0.3067   -2.4677         0   -2.4124
     -0.2165   -2.6242         0   -2.5852
      0.0171   -2.7131         0   -2.5845
      0.0087   -2.8550         0   -2.7901
      0.1294   -3.0484         0   -2.8927
     -0.0556   -2.9792         0   -3.0460
           0   -3.0000         0   -3.0000

  Smoothed path found with 2 intermediate waypoints:
           0   -0.7854         0   -0.7854
     -0.5093   -0.9764         0   -1.2828
     -0.6482   -1.7378         0   -1.4863
           0   -3.0000         0   -3.0000
```

The above command window data shows that 18 intermediate waypoints and narrowed down to 2 intermediate waypoints.

Worked along with Arun Madhusudhan