530.663 Final Project Report

Jiaqi Yang

2021/05/11

Task 1

1.1 Serial Manipulator Planning in PRM and RRT Algorithm

In serial manipulator case, the Euclidean distance cannot be used to measure the distance between nodes Therefore, the change of joint angles is the key. With a given range of the changes, this range can be regarded as the unit distance delta_q in dot robot case. Then the PRM and RRT algorithm are applicable with this idea.

1.2 Experiment

1.2.1 Test Case PRM

Workspace: 50 x 50 rectangle [0 50 50 0; 0 0 50 50]
Obscale 1: [20 40 40 20; 34 34 45 45]
Obscale 2: [17 22 22 17 12 12; 12 14 22 28 22 14]
Obscale 3: [30 45 37; 14 14 25]
qI = [0.1 0 0 0 0.4]
qG = [1.2 0 -0.3 0.4 0]
number of links = 5
length of each link = [8 8 8 8 8];
proximal end = [0;0]
The maximum number of nodes is **3500**.
PRM will search **5** nearest neighbors.

1.2.2 Test Case RRT

Workspace: 40 x 40 rectangle [-20 20 20 -20;-20 -20 20 20]
Obscale 1: [-5 5 0; -12 -12 -4]
Obscale 2: [-10 -5 -5 -10; 5 5 10 10]
Obscale 3: [4 10 10 4; 6 6 12 12]
qI = [0 0 0 0]
qG = [2.1 -0.3 6.18 -0.8]
number of links = 4
length of each link = [4 4 4 4];
proximal end = [0 ;0]
The maximum number of nodes is **1500**.

1.3 Results

1.3.1 PRM Algorithm

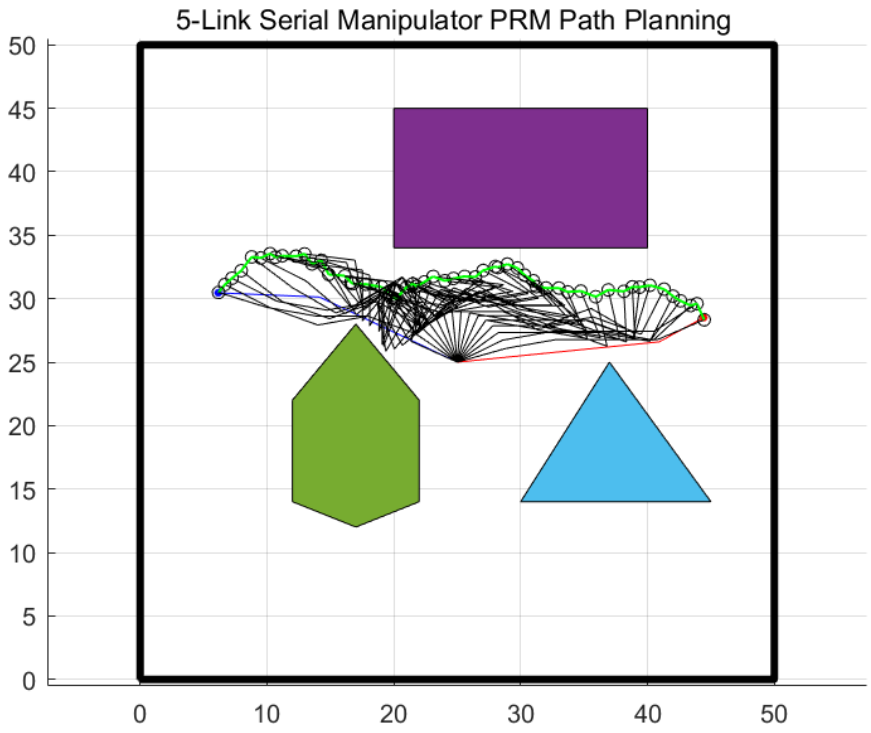


Figure 1.3.1 Plot of result for PRM algorithm

Red: qI Blue: qG Black: Manipulator Green Line: Path of the end effector

1.3.2 RRT Algorithm

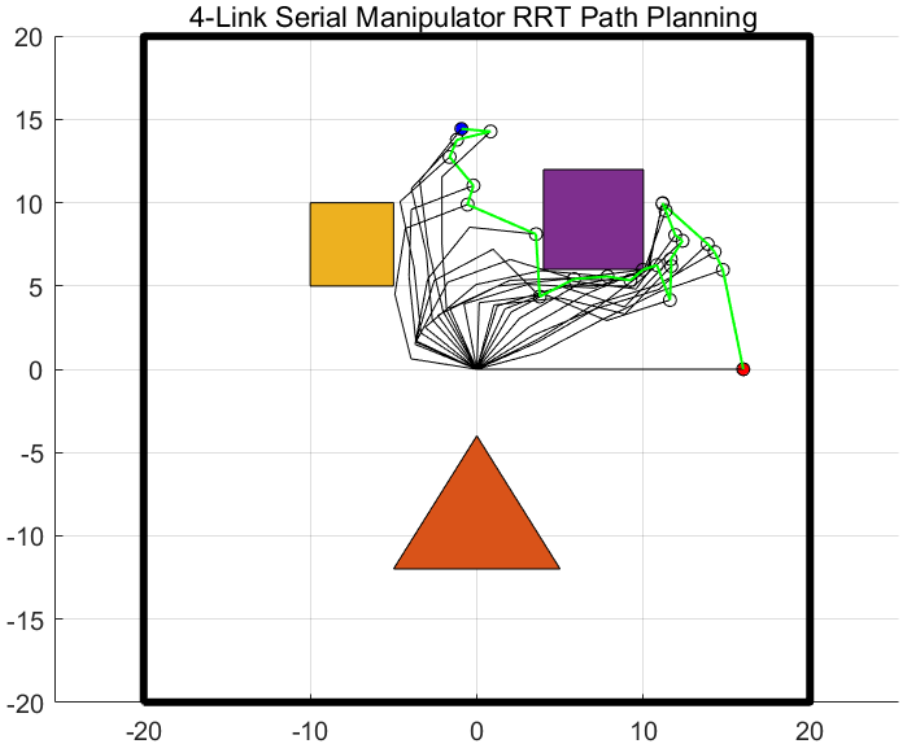


Figure 1.3.2 Plot of result for RRT algorithm

Red: qI Blue: qG Black: Manipulator Green Line: Path of the end effector

As **Figure 1.3.1** and **Figure 1.3.2** shown, the algorithms correctly move the end effector of manipulator to qG in

both test cases with a slightly different pose.

1.4 Discussion

Although both algorithms can find a correct path, 3500 and 1500 nodes are often not enough for complete the path planning for both algorithms, thus multiple attempts are necessary, which results in a considerably large amount of execution time and sometimes executes for more than ten minutes. Since the manipulator is moving in a weighted range of change of joint angles. Making the range too large with make the manipulator suddenly moves in a randomly inappropriate pose (changes of joint angles are too large). Inversely, making it small will require more nodes which significantly increases the execution time. Therefore, it needs additional experiments to figure out an acceptable range of changes of joint angles. **Figure 1.4.1**

Sometimes the algorithm fails when the path hits the obstacles. It happens because all the nodes are not placed uniformly due to the changes of angles are used instead of distance. Some nodes are placed on two sides of vertices of obstacles which requires multiple attempts to find a good path. **Figure 1.4.2**

Also, because the manipulator is move within a range of weighted changes of joint angles, the path planning terminates when the end effector hits the tolerance around the end effector of qG or the differences of each joints dropped to a certain "range" (use "or" instead of "and" because it may cause a huge amount of execution time). The final pose of manipulator and the ideal qG are slightly different. The only solution so far is to optimize the way to calculate weighted change of joint angles like adjusting the weights on each angle. **Figure 1.4.3**

```
Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!

Out of nodes, try again!
 No path found, try again!
```

Figure 1.4.1 Fail attempts

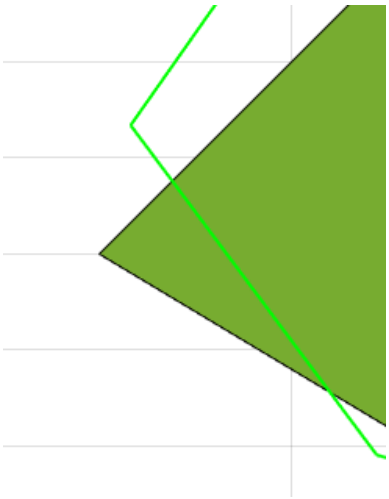


Figure 1.4.2 A failed path that hits obstacles

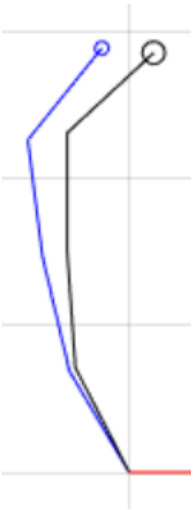
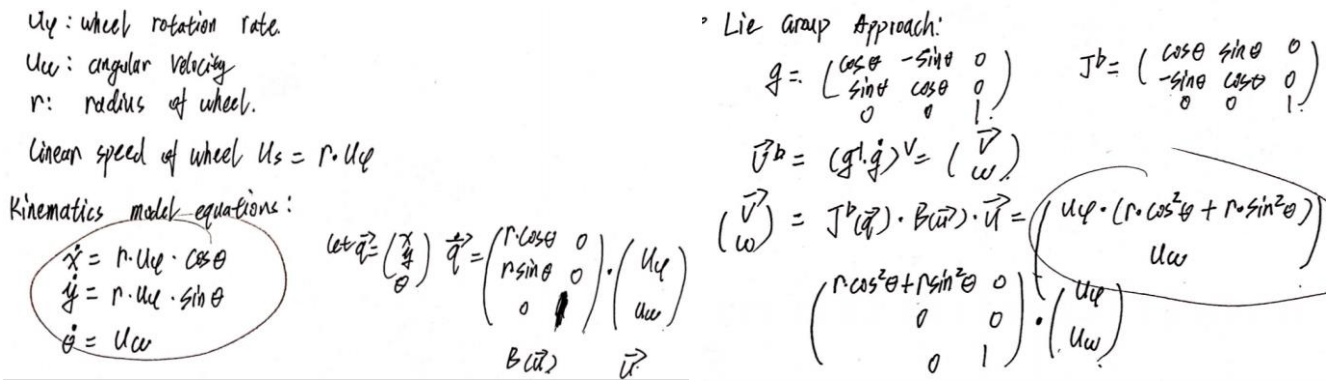


Figure 1.4.3 A slightly different manipulator (black) compared with qG (blue)

Task 2

2.1 Kinematic Model Equations of Needle



2.2 Method

Instead of giving three vertices to define the triangle. This algorithm uses the height and width of the base of the isosceles triangle. With given orientation of the body frame, all the vertices can be calculated by the three inputs.

Based on given parameters, generate all potential paths for the next step, ignore any path that the needle hits obstacles or runs out of the workspace. Then select the next node by RRT algorithm. Finally, move the needle to the next node and repeat the procedure again until the tip hits the stop region.

2.3 Experiment

The algorithm is tested in two test cases with the same qI = **[0 0 0]**^{T}. Each test case has three qG. The pedaling angular speed $\mathbf{u_\varphi}$ = 1. The angular speed of unicycle orientation is defined as $\mathbf{u_\omega} \in$ **{0 deg, −30 deg, 30 deg}.** The maximum number of path nodes for RRT is 500.

The needle model is a unicycle with the radius = **0.8**. Thus, the linear speed of the unicycle is $\mathbf{u_\varphi}$*r = **0.8**.

For the collision check, instead of defining the isosceles triangle by three vertices, this model uses the height, width of base of the triangle and the orientation angle of the base frame. In this experiment, set **height = 1.6**, **width = 1.0** then three vertices at the qI =**[0 0 0]**^{T} is **[1.6, 0]**(tip point), **[0, 0.5]** and **[0, -0.5]**.

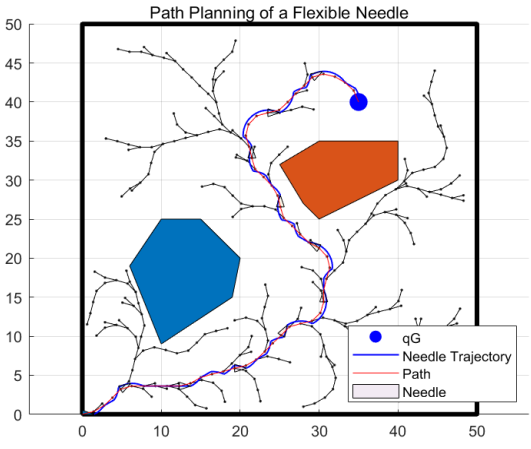


Figure 2.2.1 Path planning by RRT

2.4.1 Test Case 1

Workspace: 50 x 50 rectangle [0 50 50 0; 0 0 50 50]

Obscale 1: [10 19 20 15 10 6; 9 15 20 25 25 19]

Obscale 2: [30 40 40 30 25 28; 25 30 35 35 32 27]

qI = [0;0;0]

qG1 = [35;40] qG2 = [10;35] qG3 = [45;35]

2.4.2 Test Case 2

Workspace: 50 x 50 rectangle
Obscale 1: [10 15 20 15 10 5; 15 15 20 25 25 20]
Obscale 2: [30 40 40 30 25; 30 30 35 35 32]
Obscale 3: [25 30 40 45 40 30; 20 15 15 20 25 25]
qI = [0;0;0];
qG1 = [35;40] qG2 = [30;27] qG3 = [48;25]

2.4 Results

Note: Some of the triangles in the plot are hidden to keep the plot neat.
Red line is the path of nodes generated by RRT. Blue line is the needle tip trajectory.

2.4.1 Test Case 1

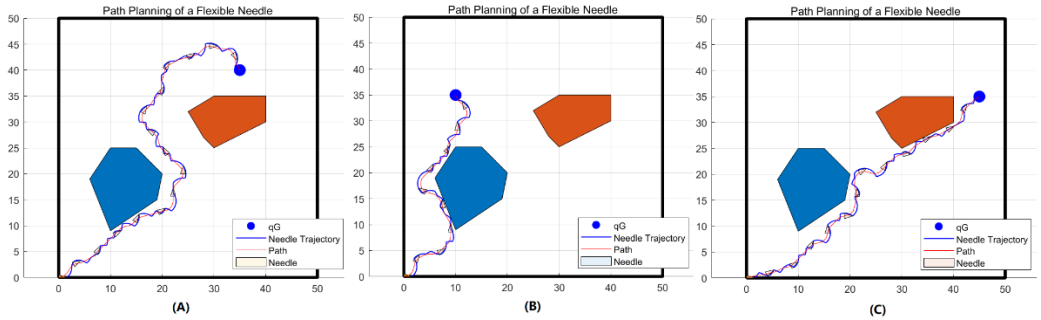


Figure 2.4.1 Plot of results for test case 1 (A): qG1 (B): qG2 (C): qG3

2.4.2 Test Case 2

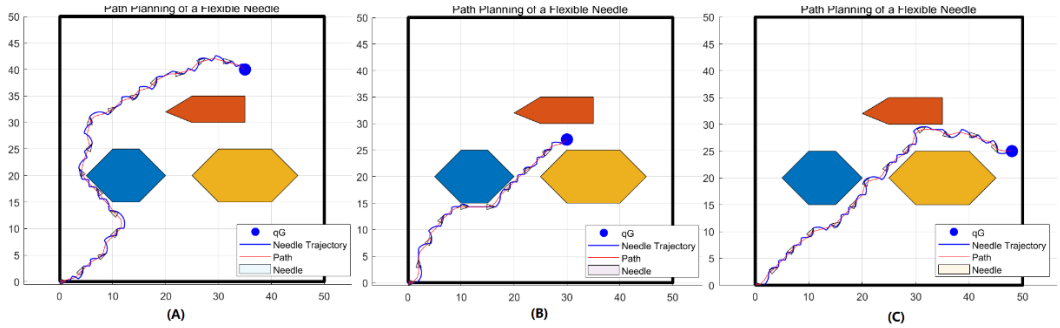


Figure 2.4.2 Plot of results for test case 2 (A): qG1 (B): qG2 (C): qG3

As **Figure 2.4.1** and **Figure 2.4.2** shown, the algorithm correctly drives the needle to qG in both test cases.

2.5 Discussion

For the given parameters in previous test cases, the needle correctly goes to the target point. However, in some cases, the needle shows a zig-zag trajectory even the path is a straight line. An unsmooth trajectory for needle insertion does not only increase the operation time but also cause unexpected damage to patient in real-life practice.

To make the trajectory as smooth as possible, one possible solution is to adjust the linear speed and reduce the angular speed of the needle. For the test case 2, if increase $\mathbf{u}_{\varphi}$ to 2 and reduce $\mathbf{u}_{\omega}$ to 20 deg, with the same qG, the needle shows aN obviously smoother trajectory.

However, merely adjusting linear and angular speed may also cause difficulties in the path planning which has to run the algorithm for

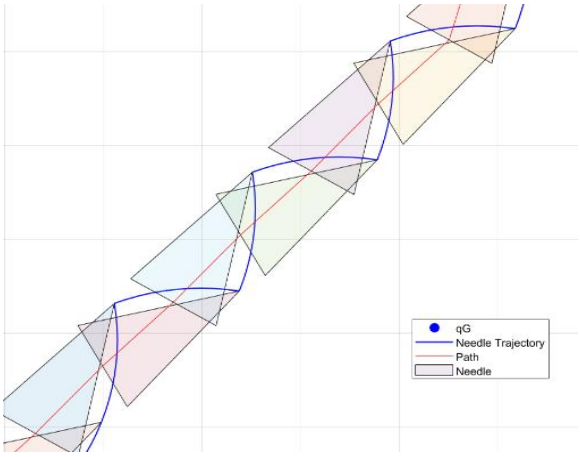


Figure 2.5.1 zig-zag needle trajectory

multiple times until an ideal path is found. Therefore, it needs to find a balance between the smoothness of trajectory and algorithm execution time.

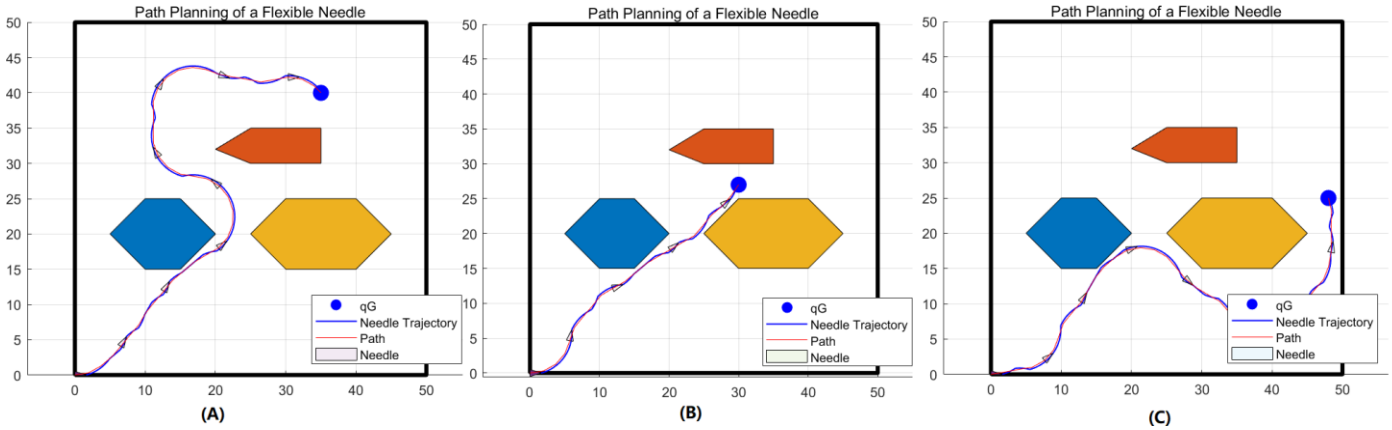


Figure 2.5.2 Smooth needle trajectories with $u_{\varphi} = 2$ and $u_{\omega} = 20$ deg