CS5335
Final Report
Jian Cui

Implement the PRM on Puma 560 by using Corke's Matlab robotics toolbox

1. The problem I try to solve.

In the real world, our robot will meet some obstacle while it is moving. My program try to find the way for the robot and avoid the collision.

The condition we got for the problem:

– a point-robot (robot is a point in space)

– a start and goal configuration

The result:

– path from start to goal that does not result in a collision

2. The algorithms I am using

I use the Probabilistic roadmap algorithm to solve this question. The algorithm includes two parts. The first part is construction phase. During this phase, we need create a roadmap that shows what point can be reach in this environment. For this part, I choose to sample some point in the environment. At first, I want to make sure I only create the number of sample I needed. So, my first sampling algorithm is as below:

1. Create a sample array S.

2. Random choose a point p (p is within the robot limitation)
3. Check if p is a point collision, if not add it to S, else go back to step 2
4. Add p to S. Check if S is a connected graph, if not delete p and go back to step 2.
5. Check if qStart and qEnd can be connected to S. if not go back to step 2, else output S,

In this way, every time we add a new point to the sample S, S can become a new connected graph. As long as qStart and qEnd can be connected to the graph, the sample is enough and there is a path between qStart and qEnd. In this way, I only consider about the number about samples. I want the sample to be as less as possible. However, in this way, I need to do lots of checking. For step 4, if S has 2 points, we need check twice, if S has 3 points we need at least check 3 times and etc. So we need do n! times checking, if we has n samples. For example, 10! = 3628800, if we have 10 sample, we need check at least 3628800 times. We also need check same times for step 5.

Therefore, I change the sampling algorithm. I choose to sample a lot of point. And I assume those points can become a connected and connected with qStart and qEnd as long as I sample enough amount of points. In this way, I only need to check n times for checking point collision.

After sampling the points, I will use A* algorithm to calculate the path. I set a cost f(n) = g(n) + h(n) for every point. g(n) means the cost for qStart to current point and h(n) means the predicted cost from current point to qEnd. And the cost of the path is sum of the cost of point which is in the path. First, I set h(n) = 0. In this case, the A* algorithm is as same as dijkstra algorithm. The program will find the nearest point one by one, until the nearest point is the end. Then, I set h(n) = norm (qEnd – qCurrent), in this case, the program will measure which point near both the start and the end, add go to that point.

3. The experiments and the result

I use Corke's Matlab robotics toolbox to create a experiments. The robot I used is Puma 560. The rob is start at its alpha position ([0, 0.4318, 0.0203, 0, 0, 0]) which shows in figure.1 and end at [2.8368, -2.3679, 1.1261, 3.1416, 1.7453, 0.3048] which shows in figure.2 .

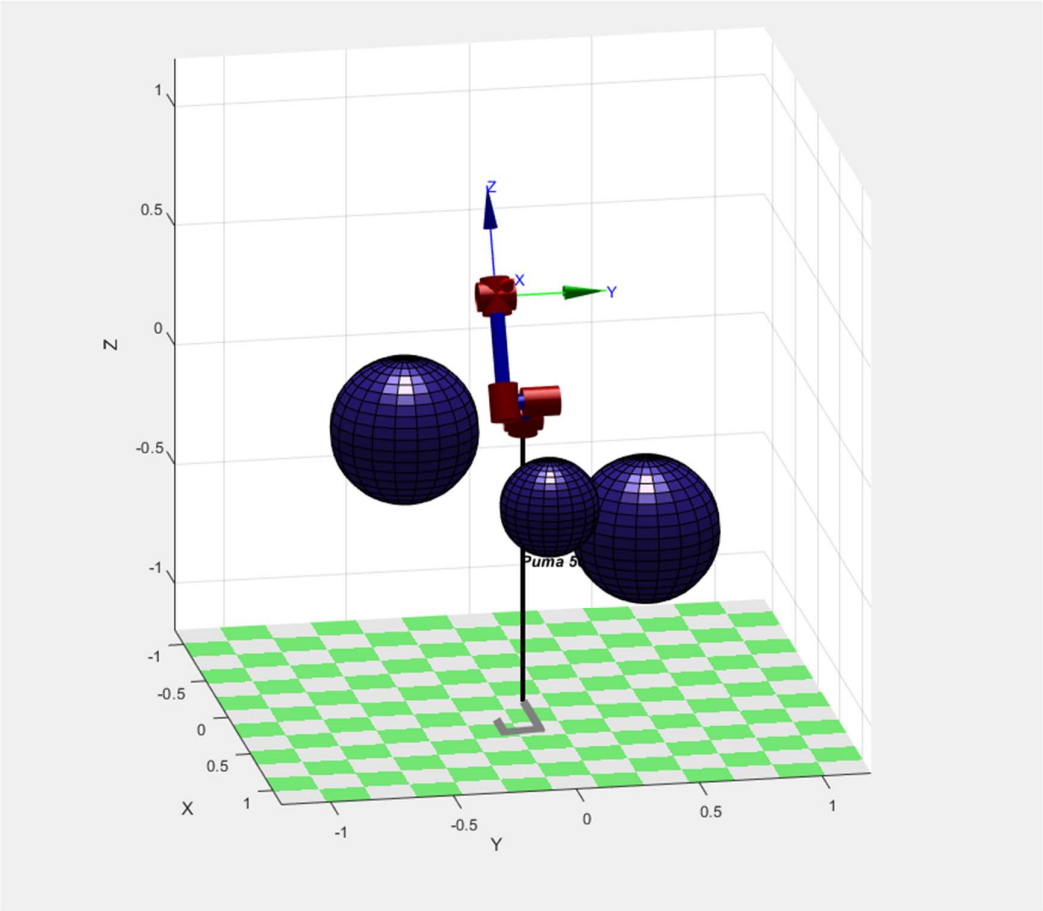

figure.1

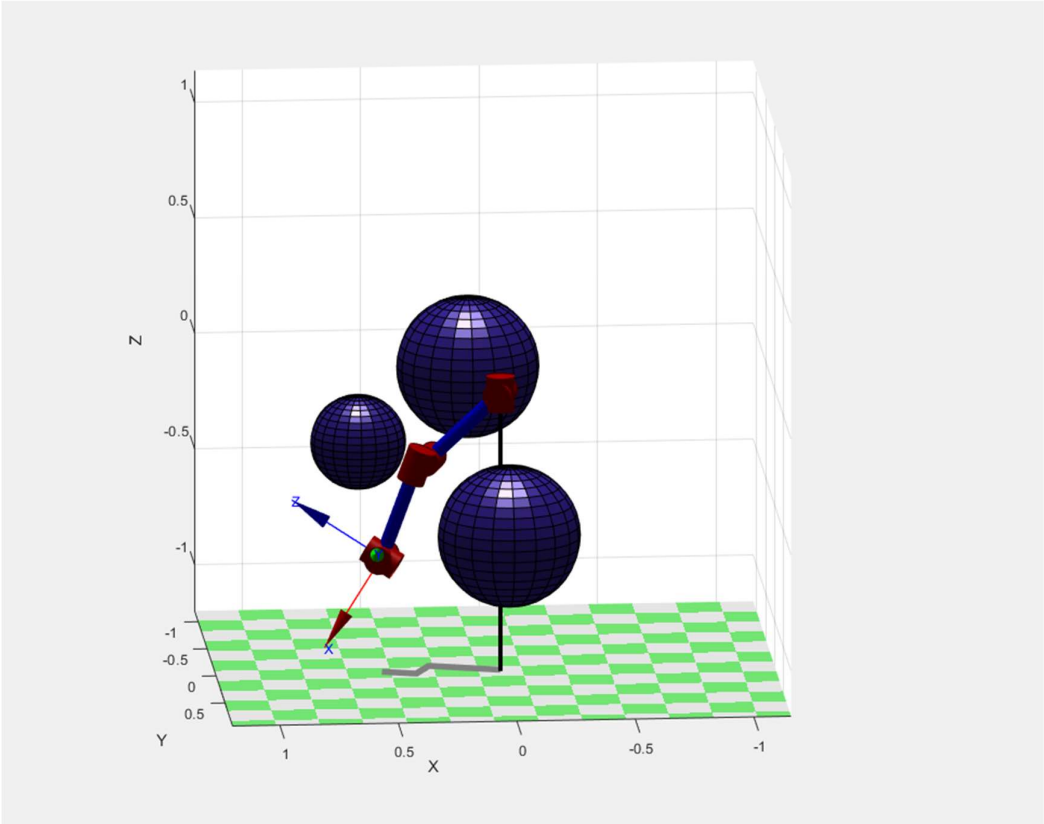

figure.2

The data of three obstacles is shows as below

	Sphere Center	Sphere Radius
Obstacles 1	[0.6;0.0; -0.2]	0.2
Obstacles 2	[0.1; -0.5;0]	0.3
Obstacles 2	[0.0;0.5; -0.5]	0.3

At first, I set number of sampling points equal to 300 and I let my program print the number of the points is check. The output is shows below.

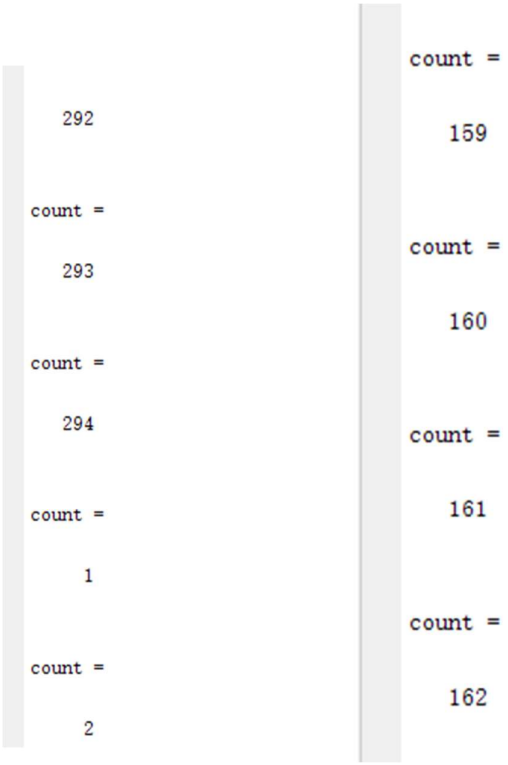

figure.3

When I set h(n) to 0, the program searched 294 points to get the best path. It is almost all the sample. However, when I set h(n) to norm (qEnd – qStart), the program only searched 162 points to get the best path.

300 points is still a large data. I changed the sample size to 50 and test 10 times. The result is shows below:

Number of visited points	
h(n) = 0	h(n) = norm(qEnd – qStart)
49	42
46	32
49	42
49	46
49	41

49	36
49	38
46	36
49	44
47	35

4. Conclude

From the result, we can find when we use A* algorithm, if we set (h(n) = 0), the program will visit most of sample to find the best path. However, if we use h(n) = norm (qEnd – qStart) to teach the program. The program will avoid visiting some useless point.

Therefore, choose an appropriate is h(n) important.