A Comparative Analysis Between Probabilistic Roadmaps and Rapidly Exploring Random Trees for Puma 560

Prasanth Murali, Himtanaya Bhadada, Robert Platt

Northeastern University

ABSTRACT

This work explores motion planning algorithms for the puma 560 robot in a high dimensional configuration space. We illustrate the two popular sampling algorithms (Probabilistic Roadmaps (PRM*) and Rapidly Exploring Random Trees (RRT*)) and present a comparative analysis for convergence of the algorithms in finding a collision free path from start to end position in the workspace.

INTRODUCTION

Sampling-based planners create possible paths by randomly adding points to a tree until some solution is found or time expires instead of evaluating all possible solutions. They are hence probabilistic complete since a path will be found if times goes to infinity.

RRTs can be thought of as creating a tree from a robot's starting point until one of its branches reaches the goal. On the other hand, probabilistic road-maps create a tree by randomly sampling points in the configuration space, keeping only those points that are collision free, connecting them with nearby points using paths that depend on the kinematics of a robot, and then using graph shortest search algorithms to find shortest paths on the resulting tree.

Such *probabilistic roadmaps* have to be created only once (assuming the environment is not changing) and can then be used for multiple queries. This is a particular advantage that the PRM presents, over the RRT algorithm. In contrast, RRT's are known as *single query* path-planning algorithms. Thus, for the same workspace, running a RRT for the robot can lead to different solutions each time while PRM is more consistent in this way.

But, it should be noticed that there is no gold standard or heuristic and even their parameter-sets are highly problem-specific.

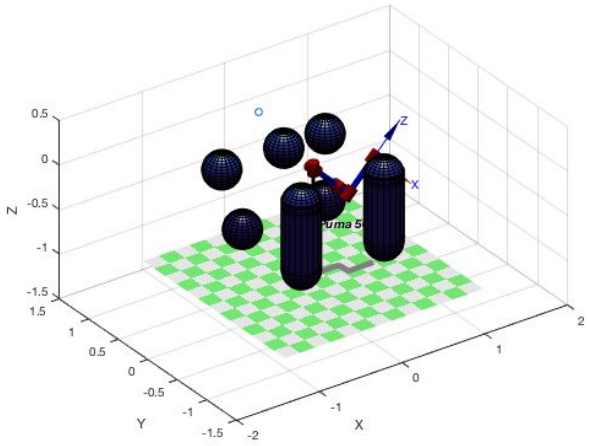

Figure 1: Workspace representing the Puma 560 robot and the obstacles

In this project, we are studying the application of PRM and RRT to the puma 560 robot, provided by Peter Corke's Robotics Toolbox. The puma 560 is modeled as a Serial Link object having 6 DOF. The script creates the workspace variable p560 which describes the kinematic and dynamic attributes of a Unimation Puma 560 manipulator using standard DH conventions.

The configuration space consists of five spherical, four hemispherical and two cylindrical obstacles as shown in Figure 1. The goal of the robot is to traverse from the current joint

configuration to [-0.3;0.5;0.5] (marked as blue circle).

We will now have an overview of the PRM and RRT algorithms implemented to achieve the above task.

PROBABILISTIC ROADMAP

Probabilistic Roadmaps are processed in two phases, the preprocessing phase to construct the roadmap; and the query phrase to connect the initial (qStart) and final (qGoal) configurations to this map and find a path between them.

The preprocessing stage involves building the probabilistic roadmap,which is an undirected graph G = (N, E). The nodes in N correspond to configurations of the robot in the collision free C-space. The edges of the graph in E denote the feasible straight-line paths. In the query phase, given a starting configuration (qStart) and a goal configuration (qGoal), the algorithm will search the roadmap for a path joining these two points together.

```
G.init() ; i = 0
While i < N
    α ← SAMPLE
    if α ∈ C_free
        G.add Vertex( α ); i = i+1
        for each q ∈ ngd(α, G)
            if (connect( α , G))
                G.addEdge( α , q)
```

Figure 2: PRM pseudocode

The above figure details the pseudocode behind the PRM algorithm. It starts with an empty graph: G(Vertex, Edge). Assuming we are generating N random points, a sample configuration is generated for every point. We now run the collision detector to find out whether the point is in collision. Post this, we need to find out all the edges that this point is able to connect to. This set of neighbourhood is made of nodes within a certain distance of R from this point. If there is no obstacle in the configuration space within the straight line, we add the edge to the graph G. Otherwise, we do not retain the edge. This process is repeated for all the points.

RAPIDLY EXPLORING RANDOM TREES

We use a baseline Rapidly Exploring Random Tree (RRT) searching for a path from the initial to the goal configuration to compare the performance of PRM against it.

For each step, the algorithm determines a target configuration and expands the tree towards it. During expanding, the algorithm only needs to verify whether each step is collision free but does not need to avoid obstacles. A pseudocode of the RRT algorithm is described in Figure 3.

During the search process, the target can either be a random configuration or the goal configuration itself, depends on the probability value defined by the user. This bias is introduced into the algorithm, since the algorithm might not converge otherwise. Introducing the goal configuration as a sampling point helps the algorithm sample and move towards the goal in shorter time.

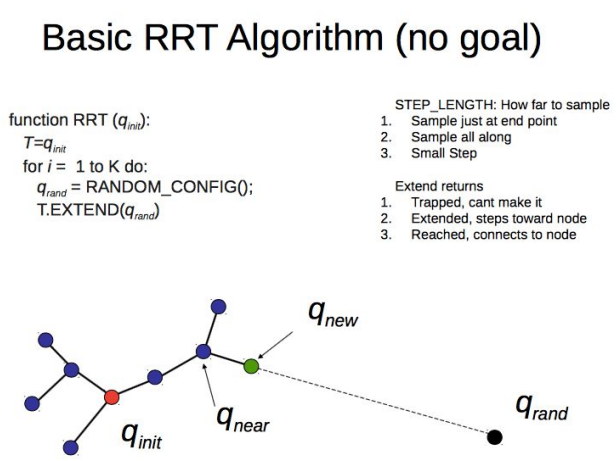

Figure 3: RRT algorithm

From the above descriptions, it is evident that the collision checks comprise of the most important parts of both the algorithms. We will have an overview of the collision check algorithms used in this project.

COLLISION CHECK

From figure 1, it is clear that our workspace consists of five spherical, four hemispherical and two cylindrical obstacles. The collision check involves calculating the cartesian coordinates for each joint of the robot using Forward Kinematics for the current joint configuration and checking for the presence of these points inside or on the boundary of the obstacles. The serial links each of the joints are checked for collision by dividing the line joining the joints into various points and checking those for collision. The manipulator is also checked for self collision at all points.

For the sphere with center at (x0,y0,z0) and radius (r) , a point (x,y,z) is on/inside the sphere if the following equation is satisfied :

$$(x - x0)^2 + (y - y0)^2 + (z - z0)^2 <= r^2$$

The hemisphere is checked for collision in a similar way. The hemisphere is essentially the intersection of a half space and a sphere. Thus, the point is checked for presence in both the cases and the intersection is taken as as the result.

For a cylinder with center at (x0,y0,z0) and height ‘h’ (parallel to z axis), the process to check if a random point (x,y,z) is on/inside the cylinder is as follows:

1. Check if the point is within the cylinder on the x-y plane, checking for presence of the point inside a circle denoted by center at (x0,y0) and radius (r) using the following equation:

$$(x - x0)^2 + (y - y0)^2 <= r^2$$

If the inequality is false, the point is outside, otherwise continue.

2. Check if the z coordinate of the point is with the z coordinates of the top and bottom surfaces of the cylinder.

$$ztop = z0 + h/2$$

$$zbottom = z0 - h/2$$

Thus, for each random joint configuration of the robot, the coordinates of each of the joints calculated using FK is checked for presence inside the obstacles using the above equations. The point is said to be collision free only if it is outside all the obstacles (thereby satisfying all of the equations).

Now, while calculating the edges of the graphs, we must ensure that the line is collision free. This is done by dividing an edge joining two nodes with a step size of ten and checking for collision for each of those ten points. The path is said to be collision free only if all the points are collision free.

After setting up the environment and implementing the algorithm, we conducted several experiments to compare the performance

of PRM and RRT and understand the application of both the algorithms better.

RESULTS AND DISCUSSION

Experiment 1:

In this experiment, we compared the path lengths generated from running both PRM and RRT for the puma 560 robot in the workspace (Figure 1). The square of connection distance for PRM was set at 10 units and the maximum number of nodes it could sample was set at 100. The experimental setup was the same for RRT as well. Each of the algorithms was run 100 times and the path length for each of those iterations were noted.

The average path length from PRM was **6 nodes (SD = 0.71)** (including qStart and qGoal). The average path length for RRT was **3.8(~4nodes)** again, but the standard deviation was equal to **2.3**. This also suggests a higher variance between the path lengths in case of RRT as compared to a more consistent path length across various runs in case of PRM. We also observed that the path traced by the robot from PRM in each of those iterations were more or less similar. Figure 4 represents the average path returned across all iterations of PRM.

Analysis: In PRM algorithm, all points in the configuration space are initially sampled into a graph and the path connecting the start and end configurations is queried separately post the generation of this graph. It is safe to assume that every configuration space has only a unique set of collision free points. Thus, upon running PRM for similar amounts of time, more or less similar set of collision free points and paths are going to be generated for a particular workspace. This means, connecting the initial and final configurations to this graph would only yield similar shortest paths across various turns and explains why the trajectory of robot when following the path generated by PRM was similar across most of its runs.

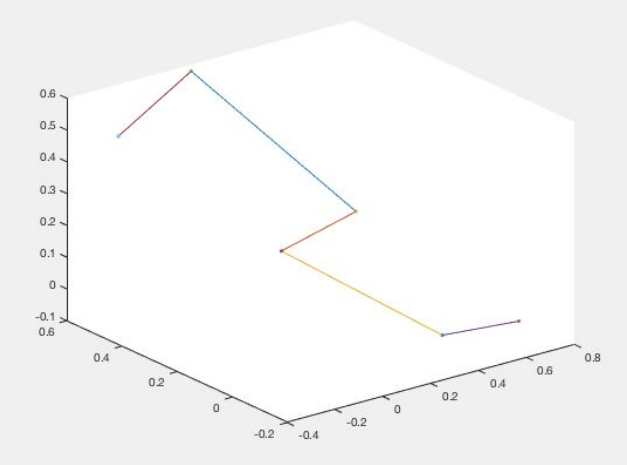

Figure 4: Denotes the average path returned by PRM over 100 iterations

In case of RRT, the tree starts at the initial position, repeatedly sampling random collision free paths that connect to the previously sampled point, until the goal is found. Thus, each tree is not determined by coverage of the configuration space as in PRM, but in how proximal the randomly chosen point is to the goal configuration in that particular iteration. This varies for each of the runs and hence the paths returned by the algorithm are not at all similar.

Experiment 2:

In the following experiment, we compared the running times of PRM and RRT for the above problem. The square of connection distance for PRM was set at 10 units and the maximum number of nodes it could sample was set at 50. Each of the algorithms was once again run 100 times for the experiment.

The following figure shows the variance of running time for PRM and RRT.

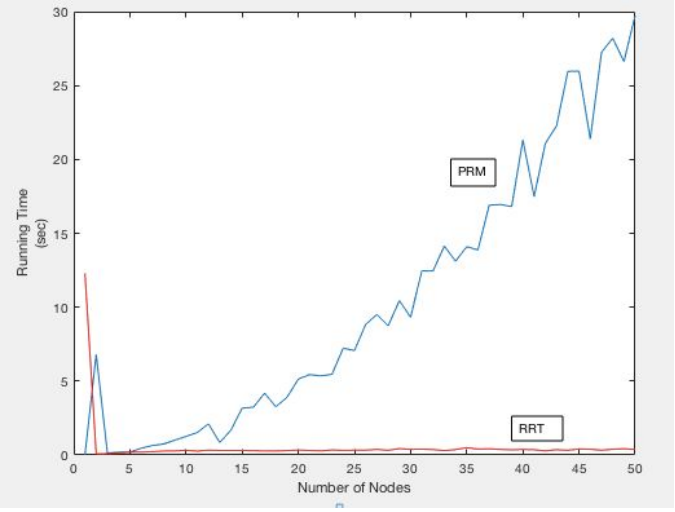

Figure 5: Plot showing Running time, varying the maximum number of nodes for PRM and RRT

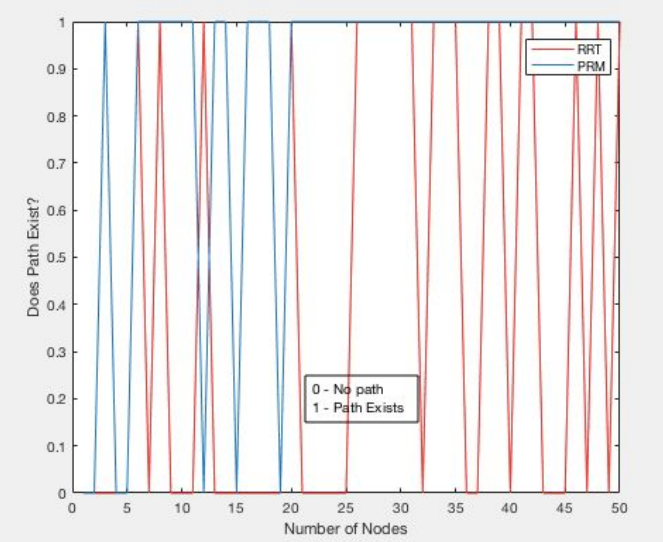

Figure 6: Plot shows the existence of the path (1 or 0) as a function of the maximum number of nodes for PRM and RRT

From the plot, it is clear that, for PRM, the running time increases exponentially with increasing number of nodes. The RRT seems to return significantly lesser ($p<0.01$) running times as seen from the plot. A summary of the results is presented in Table 1. From our analysis of the functions, it was evident that the computation time is significantly caused by the isCollision() methods.

	Mean Running Time	SD
PRM	10.31	9.06
RRT	0.55	1.69

($p<0.001$)

Table 1: Table shows that PRM significantly takes longer time to converge than RRT

As a part of the above experiment, we also collected data to check if the algorithm is able to find a path between the starting and goal configurations of the robot as a function of number of nodes.

Figure 6 shows the variance of existence of path with the number of nodes for both PRM and RRT. It is clear that the PRM converges after approximately 20 nodes and always returns a path, whereas the RRT fluctuates continuously and we cannot say with certainty if a path exists.

The reason for this could again owe to the randomness factor we discussed in the previous experiment. Once a certain number of nodes are sampled, the PRM is always able to find the shortest path joining the start and goal configurations of the robot within those sampled nodes. Thus, unless the additional nodes that come by virtue of more iterations actually contribute in getting a shortest path, the path is going to be similar across iterations. This cannot be true of RRT, since at each node, new nodes are randomly sampled and connected to the existing path.

Thus, concluding the experiment, two important tradeoffs are realized here:

1. powerful planners are slow, but can find paths between relatively isolated nodes

2. fast planner is less accurate, requires more nodes to be generated, and needs to called more often

Experiment 3:

We have established by virtue of the above experiments that PRM works better for the current workspace of the robot. Thus, the question arises as to the methods adopted in finding the closest neighbours and the distance adopted for the same.

The distance of neighbor R is the number that tells us the scope of the ability to connect to configurations by straight lines. There is a tradeoff when choosing R. If R is small, neighboring points will be dense which is easier to connect but takes a lot of time before filling up the whole search space. If R is large, the nodes are more sparse. In this case, we will be able to connect far away points with straight lines, but the algorithm will be easier to fail. The idea is that if we start building the graph and every point is very close to each other, eventually it will be very easy to chain all the points together to go from any point to any goal.

Thus arises the question of how do we arrive at the distance (R) ? To analyze this, we varied the PRM algorithm to obtain the path and running time, across square connection distance of 1 to 5 changing the number of nodes from 1 to 50.

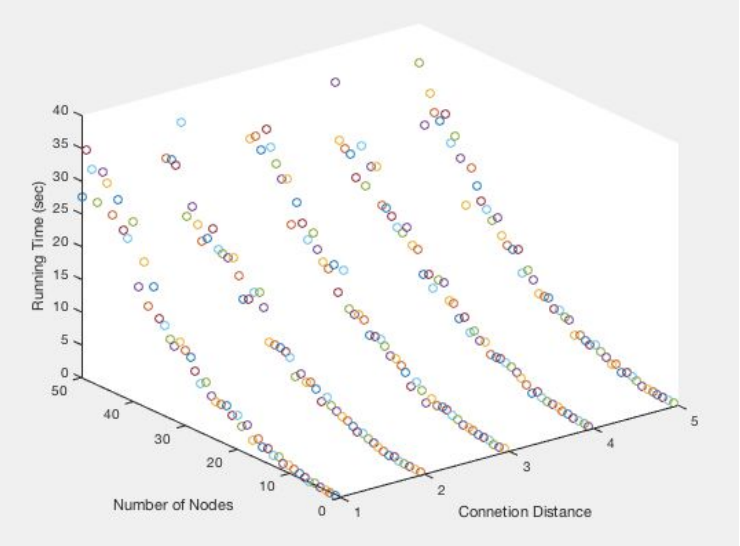

Figure 7: Plot of running time as a function of number of nodes (1:50) and connection distance (1:5)

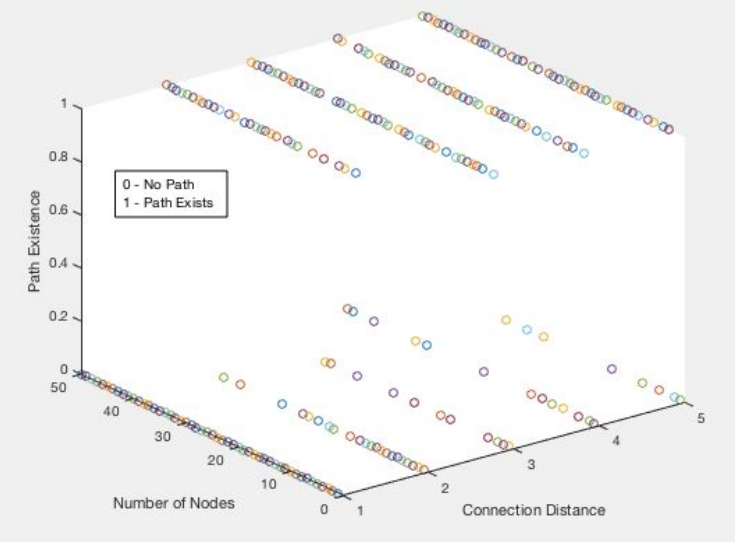

Figure 8: Plot of running time as a function of number of nodes (1:50) and connection distance (1:5)

For each connection distance, Figure 7 shows that the running time follows an exponential pattern across the number of nodes as expected. We also generate the plot for path existence as a function of number of nodes and connection

distance. Figure 7 and Figure 8 together will help us find the optimal parameters for this algorithm. It is clear from figure 8 that, from around 20 nodes, the algorithm converges to find a path irrespective of the connection distance and since the running time increases exponentially, as seen in Figure 7, it is beneficial to choose the lowest number of nodes that yield the accurate answer. Thus, for our workspace, PRM returns accurate paths consistently from **20 nodes** onwards. It does **not depend on the connection distance**. Thus, when searching for nearest neighbours, since the number of nodes is not too high, we adopt a nearest neighbours approach where we consider all nodes within a particular distance of a node to be its neighbours(not kNN).

CONCLUSION

Our work presents a comprehensive analysis between the applications of PRM and RRT to the puma560 robot in our workspace. We have looked to answer some of the typical questions in motion planning like path length, convergence time, choosing the right number of nodes and neighbour distances.

From our analysis, it is evident that PRM returns a viable path between qStart and qGoal in almost all scenarios, but spends a considerable amount of time on pre-processing. Neighbour selection is the most important tradeoff we need to make. Since each collision check is expensive, we should try to avoid these as much as possible. But, If we try too few connections, we will fail to connect the graph. Connected components in the graph play a crucial role here. We like to connect such components into larger components whenever we can. Also, we need to create new components in unexplored parts of the configuration space.

We realize that, in case of RRT,

1. The algorithm is sensitive to the metric for evaluating the configuration.
2. The algorithm bases itself on random sampling, which may yield relatively poor performance for a simple problem.

Thus, we can conclude that for the given problem, with some lenient constraints on the computational time, PRM consistently outperforms RRT in finding a path for the robot from qStart to qGoal.

FUTURE WORK

Taking this work forward, we plan to extend the proposed analysis by considering other path planning algorithms and analyze their performance in various scenarios. Incorporating multiple sampling techniques, we would also like to explore the effect of PRM and RRT in more narrow paths and study different modes of failure. Finally, we would also like to review the application of motion planning algorithms in a more complex environment with dynamic obstacles.

ACKNOWLEDGEMENTS

We would like to extend our sincere gratitude to Prof. Robert Platt and our TA Yang Xia from the CS5335 class for their constructive feedback and healthy discussions on related topics throughout the semester.

LINK TO CODEBASE

The following repository contains link to the project codebase as well as instructions for running the program.

https://github.com/prasanthmurali/PRM-vs-RRT-for-puma560

REFERENCES

1. http://www.ccs.neu.edu/home/rplatt/cs5335_fall2017/slides/prm.pdf
2. http://www.ccs.neu.edu/home/rplatt/cs5335_fall2017/slides/rrt.pdf
3. https://personalrobotics.ri.cmu.edu/files/courses/16662/notes/rrt/16662_Lecture12.pdf
4. http://www.cs.columbia.edu/~allen/F15/NOTES/Probabilisticpath.pdf