Probabilistic Roadmap Path Planning on the PUMA 560 robot

Pratik Devikar
Northeastern University
devikar.p@husky.neu.edu

1 Problem Statement

Motion Planning (also known as Navigation problem) is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement[1]. In this paper we describe the Probabilistic Roadmap Planner (PRM). This planner has been applied with success to multiple planning problems involving robots with 3 to 16 dof moving in static environments.. In this paper, we describe the application of the PRM algorithm on the PUMA 560 6-link robot to find the best path from start configuration to the goal configuration and report the results.

2 Approach

PRM algorithm consists of 2 phases: *preprocessing* phase and the *query* phase. During the preprocessing phase a probabilistic roadmap is constructed by repeatedly generating random free configurations of the robot and connecting these configurations using some simple, but very fast motion planner. Then a query asks for a path between two free configurations of the robot. To answer a query PRM first attempts to find a path from the start and goal configurations to two nodes of the roadmap. Following is the pseudocode for the PRM algorithm[4]:

- ❖ STEP 1: Learning the map
  - Initially empty Graph G
  - A configuration q is randomly chosen
  - If q→Q_free then added to G (collision detectionneeded here)
  - Repeat until N vertices chosen
  - For each q, select k closest neighbors
  - Local plannerΔconnects q to neighbor q'
  - If connect successful (i.e. collision free local path), add edge (q, q')
- ❖ STEP 2: Finding a path
  - Given q_init and q_goal, need to connect each to the roadmap
  - Find k nearest neighbors of q_init and q_goal in roadmap, plan local path Δ
  - Problem: Roadmap Graph may have disconnected components
  - Need to find connections from q_init, q_goal to same component
  - Once on roadmap, use Dijkstra algorithm

The path finding algorithm used in this paper is Dijksta's algorithm. It is an algorithm for finding shortest paths between nodes in a graph. Let the node at which we are starting be called the **initial node**. Let the **distance of node *Y*** be the distance from the **initial node** to *Y*. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the *unvisited set*.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.[13]
3. For the current node, consider all of its unvisited neighbours and calculate their *tentative* distances through the current node. Compare the newly calculated *tentative* distance to the current assigned value and assign the smaller one. For example, if the current node *A* is marked with a distance of 6, and the edge connecting it with a neighbour *B* has length 2, then the distance to *B* through *A* will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
4. When we are done considering all of the unvisited neighbours of the current node, mark the current node as visited and remove it from the *unvisited set*. A visited node will never be checked again.

5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.[3]

3 Comparisons

1. Sample count = 5

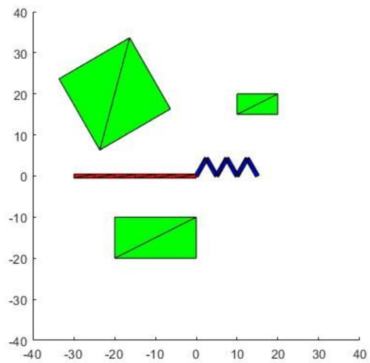
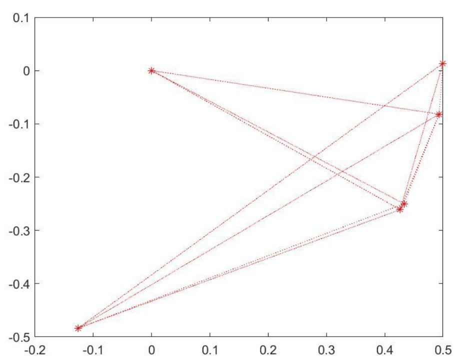


Fig 1: The robot arm wont move from the initial position if the path isnt found



2. Sample count = 10

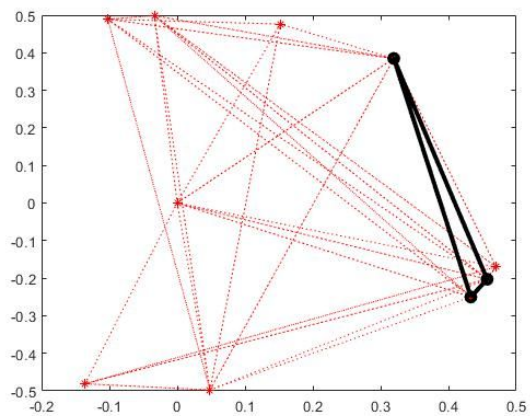
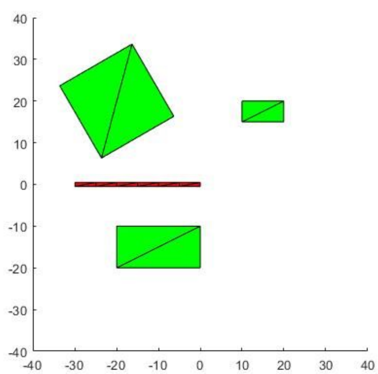
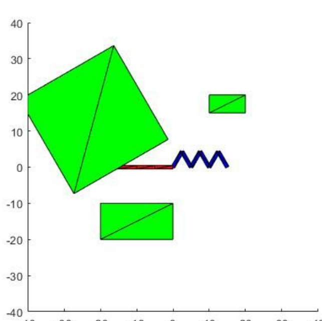


Fig 2: There are more edges in the graph and the final robot position is same as the goal position



4 Results from additional experiments

I. The program wont work if the robot arm is placed in the obstacle



5 Conclusion

We have described PRM, a two phase method for solving robot motion planning problem in static workspace. In the preprocessing phase, PRM constructs a probabilistic roadmap as a collection of configurations randomly selected across the free C-space .In the query phase, it uses this roadmap to quickly process path planning queries, each specified by a pair of configurations.

We have also tested the PRM against various parameters and the results are as expected. A challenging goal would be to extend the method to dynamic scenes. Another challenge in the future might be if the obstacles move during the path planning of the robot.

6 References

[1] https://en.wikipedia.org/wiki/Motion_planning

[2]https://www.cs.rice.edu/CS/Robotics/papers/kavraki1998prm-for-robot-path-plan.pdf

[3] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[4]http://www.cs.columbia.edu/~allen/F15/NOTES/Probabilisticpath.pdf