

Code for Robot Path Planning using Probabilistic Roadmap (Octave)

Rahul Kala

Robotics and Artificial Intelligence Laboratory,
Indian Institute of Information Technology, Allahabad
Devghat, Jhalwa, Allahabad, INDIA

Email: rkala001@gmail.com

Ph: +91-8174967783

Web: <http://rkala.in/>

Version 1, Released: 7th June, 2014

© Rahul Kala, IIIT Allahabad, Creative Commons Attribution-ShareAlike 4.0 International License. The use of this code, its parts and all the materials in the text; creation of derivatives and their publication; and sharing the code publicly is permitted without permission.

Please cite the work in all materials as: R. Kala (2014) Code for Robot Path Planning using Probabilistic Roadmap (Octave), Indian Institute of Information Technology Allahabad, Available at: <http://rkala.in/codes.html>

1. Background

The code provided with this document uses Probabilistic Roadmap Algorithm for robot motion planning. Assume that you have a robot arena with an overhead camera as shown in Figure 1. The camera can be easily calibrated and the image coming from the camera can be used to create a robot map, as shown in the same figure. This is a simplistic implementation of the real life scenarios where multiple cameras are used to capture different parts of the entire workspace, and their outputs are fused to create an overall map used by the motion planning algorithms. This tutorial would assume that such a map already exists and is given as an input to the map.

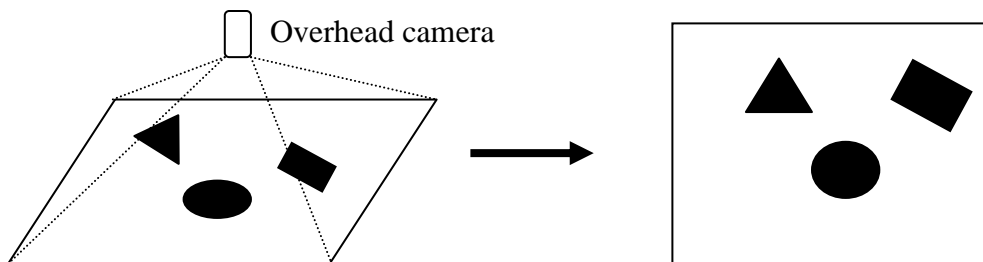


Figure 1: Overhead camera system and creation of robot map

The same camera can also be used to capture the location of the robot at the start of the planning and also as the robot moves. This solves the problem of localization. An interesting looking region of interest becomes the goal of the robot to be used in the motion planning algorithms. The robot is not shown in the map in Figure 1. This tutorial assumes that the source and goal of the robot is explicitly supplied. The aim of the current tutorial is only to plan a path for the robot; the code does not go forth with making the robot move on the desired path, for which a control algorithm is needed.

For simplicity, the robot is taken as a point-sized object. This enables a quick implementation and understanding, without delving into the libraries for collision detection and concepts of multi-dimensional configuration spaces.

2. Problem Solving using Probabilistic Roadmap

The readers should please familiarize themselves with the Probabilistic Roadmap (PRM) algorithm before reading this tutorial. The algorithm has two stages: an offline roadmap (graph) building stage and an online planning/query

stage. The aim of the offline roadmap (graph) building stage is to randomly draw a small graph across the workspace. All vertices and edges of the graph should be collision-free so that a robot may use the same graph for its motion planning. The PRM selects a number of random points (states) in the workspace as the vertices. In order to qualify being a vertex, a randomly selected point (state) must not be inside some obstacle. Let there be k number of states which is an algorithm parameter. Higher are the number of vertices or k , better would be the results with a loss of computational time. The algorithm then attempts to connect all pairs of randomly selected vertices. If any two vertices can be connected by a straight line, the straight line is added as an edge. The concept is shown in Figure 2.

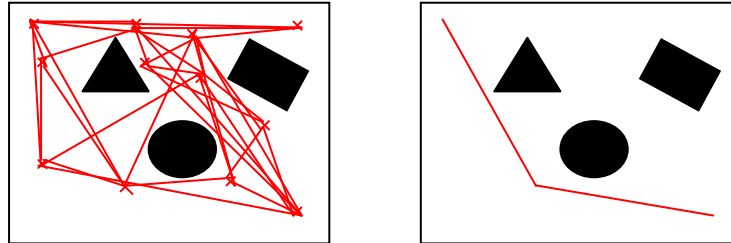


Figure 2: Probabilistic Roadmap (a) Roadmap (b) Path computed

The online planning/query stage aims to use the roadmap (graph) developed earlier for planning the path of a robot. Since a graph is already known, any graph search algorithm can be used. The code uses A* algorithm for the same. You may refer to the tutorial of robot path planning using A* algorithm. The weights of the edges is taken as the Euclidian distance between the connecting points, and the heuristic function (denoting the nearness of the point to the goal) is taken as the Euclidian distance to the goal. Both the functions are given as separate files from where they can be changed.

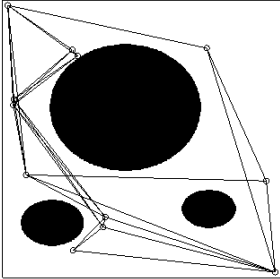
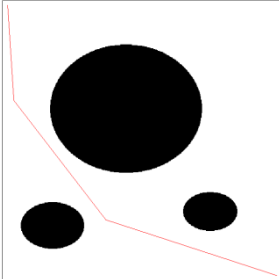
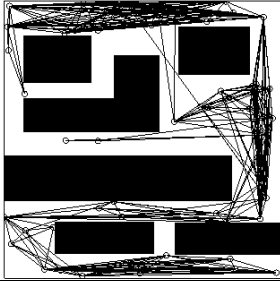
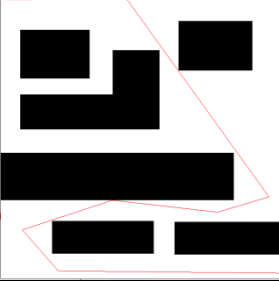
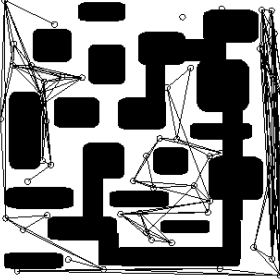
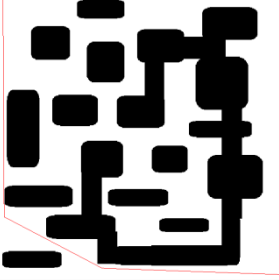
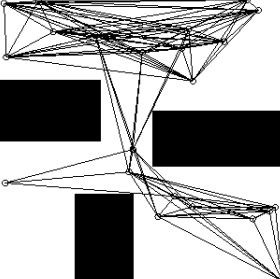
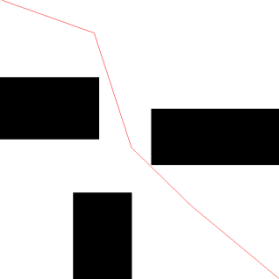
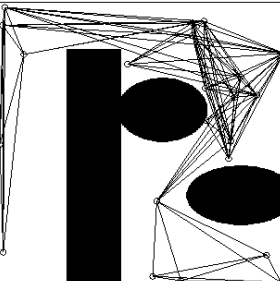
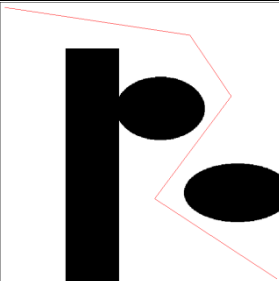
3. How to Execute and Parameters

In order to execute the code you need to execute the file 'astart.m'. First make a new bitmap file and draw any map on it, over which you need to execute the algorithm. Paint or any other simple drawing tool can be used. Make sure you save the file as a BMP. Place the file in the project folder. You may prefer to open any of the supplied maps and re-draw them. Change the name of the map file in the code to point out to the map that you created. Supply the source and the goal positions. You can use paint or any other drawing utility which displays the pixel positions of the points, to locate the source and goal on your drawn map.

Change the number of vertices in the roadmap (k) parameter. If your algorithm is taking too long to get a result, you will have to reduce the number of points. Similarly if you get results instantly, you may wish to increase the factor to hope to get a better path.

Execute the algorithm. You will need to take screenshots of the display, the tool does not do that for you. The first display shows the roadmap vertices. Click on the image to get the complete roadmap. **This will take a few seconds to minutes.** Click again to get the path. The path length and execution time are given at the console. Make sure you turn the display off by changing the display variable in the parameters before quoting the execution performance. This will not display any additional graphics and hence benefit from the additional time spent in the display.

3. Sample Results

S. No.	k	Roadmap	Path	Path Length	Execution Time (sec)*
1.	10			758	8
2.	50			1571	83
3.	50			880	54
4.	10			718	23
5.	10			1078	26

* Not for the results shown in the figure. The execution time was recorded on a separate execution with the display set off.

All results on Intel i7, 3.4 GHz 3.4 GHz with 12 GB RAM.

For all results:

source=[10 10]

goal=[490 490]

resolution of original map: 500×500

4. Disclaimer

Please feel free to ask questions and extended explanations by contacting the author at the address above. Please also report any errors, corrections or improvements.

These codes do not necessarily map to any paper by the author or its part. The codes are usually only for reading and preliminary understanding of the topics. Neither do these represent any state-of-the-art research nor any sophisticated research. Neither the author, nor the publisher or any other affiliated bodies guarantee the correctness or validity of the codes.