

Code for Robot Path Planning using Rapidly-exploring Random Trees

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 Rapidly-exploring Random Trees, Indian Institute of Information Technology Allahabad, Available at: <http://rkala.in/codes.html>

1. Background

The code provided with this document uses Rapidly-exploring Random Trees 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.

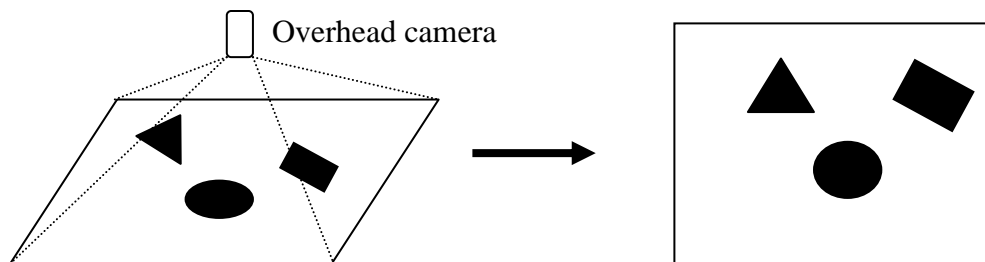


Figure 1: Overhead camera system and creation of robot map

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 Rapidly-exploring Random Trees

The readers should please familiarize themselves with the Rapidly-exploring Random Trees (RRT) algorithm before reading this tutorial. The RRT algorithm grows and maintains a tree where each node of the tree is a point (state) in

the workspace. The area explored by the algorithm is the area occupied by the tree. Initially the algorithm starts with a tree which has source as the **only node**. At each iteration the tree is expanded by selecting a random state and expanding the tree towards that state. Expansion is done by **extending the closest node** in the tree **towards the selected random state** by a small step. The algorithm runs till some expansion takes the tree near enough to the goal. **The size of the step** is an algorithm parameter. Small values result in slow expansion, but finer paths or paths which can take fine turns. The tree expansion may be made biased towards the direction of the goal by selecting goal as the random state with some probability. The concept is shown in Figure 2.

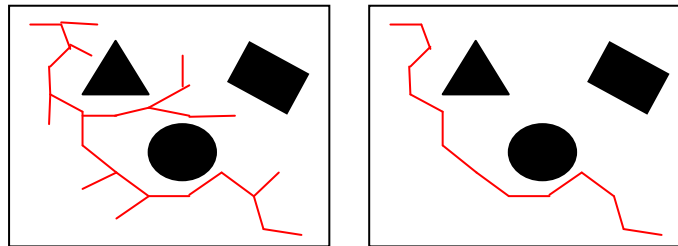


Figure 2: RRT Tree (a) Roadmap (b) Path computed

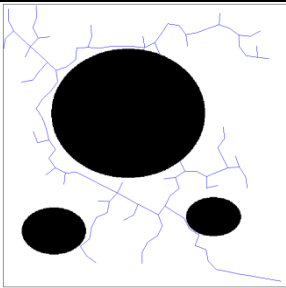
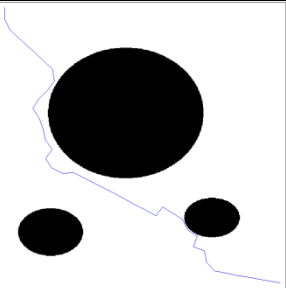
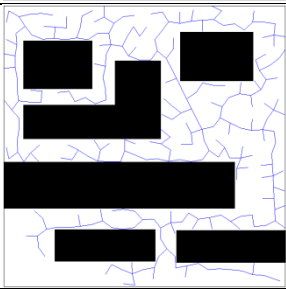
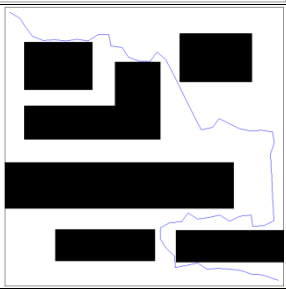
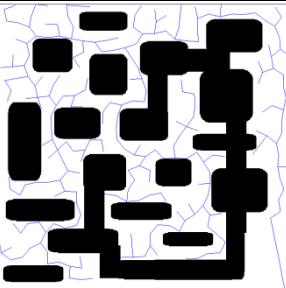
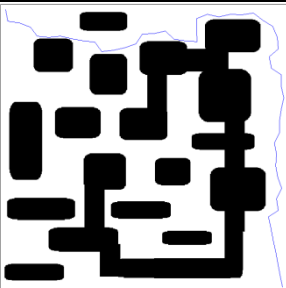
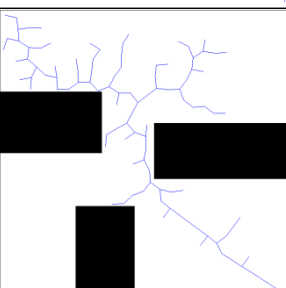
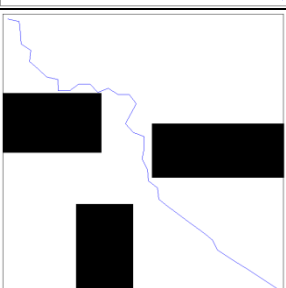
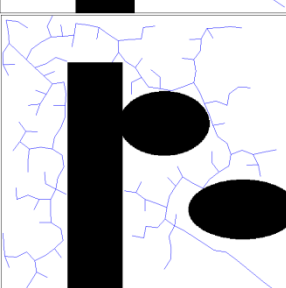
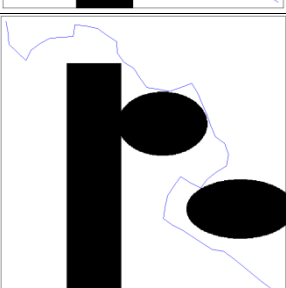
3. How to Execute and Parameters

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.

You can change the step size parameter based on the results. If your algorithm is taking too long to get a result, you will have to increase the step size. Similarly if you get results instantly, you may wish to increase the factor to hope to get a finer path. Note that decreasing the step size will not always result in better paths. The algorithm treats all nodes with a distance less than *disTh* as the same node. This factor is always in proportion to the step size. The algorithm stops if more than *maxFailedAttempts* number of iterations fail to generate a feasible expansion. The algorithm would then display a 'no path found' error.

Execute the algorithm. You will need to take screenshots of the display, the tool does not do that for you. When the simulation stops, the RRT tree is displayed. Click on the image to get the robot 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.	Step Size	Tree	Path	Path Length	Execution Time (sec)*
1.	20			846	0.892
2.	20			1301	0.501
3.	20			1055	0.928
4.	20			814	0.315
5.	20			1057	0.474

* 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.