Code for Robot Path Planning using Genetic Algorithms (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

1. Background

The code provided with this document uses Genetic 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.

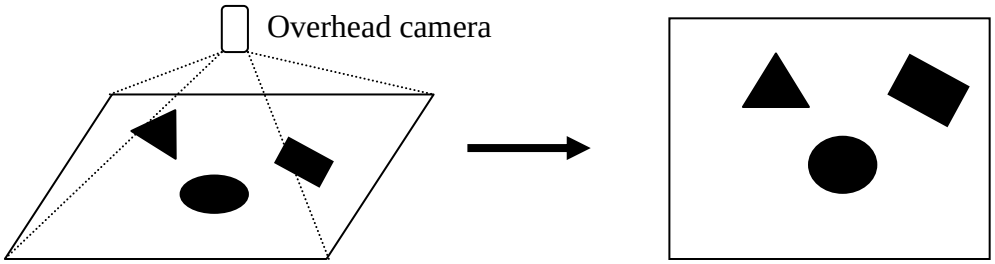


Figure 1: Overhead camera system and creation of robot map

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 Genetic Algorithms

The readers should please familiarize themselves with the Genetic Algorithms (GA) before reading this tutorial. For this tutorial GA would be used as a standard optimization algorithm. As an optimization algorithm, the GA does not require any information about the derivatives for functioning. Stress is given in this tutorial on how to model the

problem as an optimization problem, rather than on the functioning of the GA which is the same as for any optimization problem.

To model the problem as an optimization problem, we need an objective function and specification of variables of that objective function (along with their bounds). Let a path be characterized by a fixed number of points in the robotic map. In order to make some path from this set of points, we start from the source and connect it to the first point by a straight line. The first point is connected to the second point by a straight line, and so on. At the end the last point is connected to the goal. The objective function is the length of this path. A heavy penalty is added if any part of the path lies inside the obstacle, while the penalty is proportional to the length of the path inside the obstacle. The locations of each of these fixed number of points (both X and Y axis positions) are the optimization variables. The variable bounds are such that the point lies inside the map (lower bound 1 and upper bound as the length/width of the map for the X/Y axis). All points (both X and Y axis values) put one by one make the genetic individual used for optimization. The concept is shown in Figure 2.

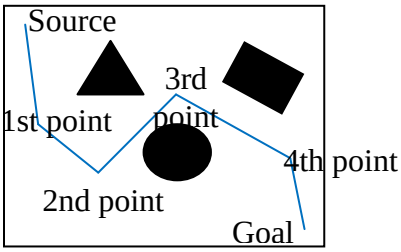


Figure 2: Genetic Algorithm Individual Representation

Each point in the path marks a point of turn. The total number of points is an algorithm parameter and should be equal to the maximum number of turns a robot is expected to make in the robot map. Setting this number too high would result in very large computational requirements. If the algorithm is not allowed a large computational time, random results may be the output. Setting a large value in simple scenarios will result in useless turns and hence a high path length. Too small value of the parameter may not give enough flexibility to the algorithm to model the optimal path, thus resulting in collision-prone paths.

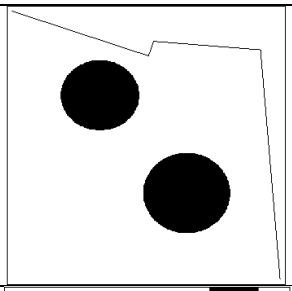
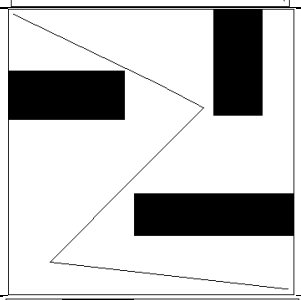
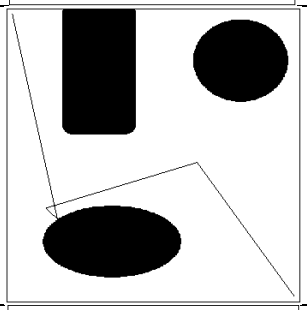
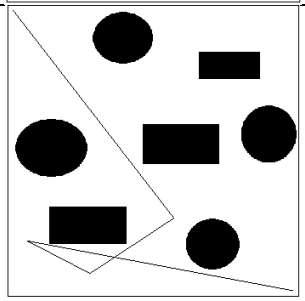
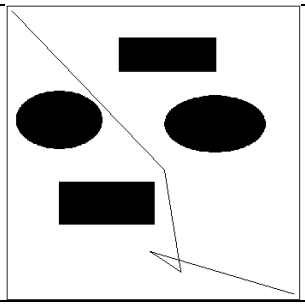
3. How to Execute and Parameters

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 number of points parameter based on the scenario of operation. The other important parameters are the population size and the number of iterations of the GA. Both are standard GA parameters. You have the option of using spline smoothening to smoothen the path of the robot useful for non-holonomic robots.

Execute the algorithm. You will need to take screenshots of the display, the tool does not do that for you. **It can take a few minutes to get the results, depending upon the settings used.** When the optimization stops, the path is displayed. The path length and execution time are given at the console.

3. Sample Results

S. No.	Number of Points	Path	Path Length	Execution Time (sec)
1.	3		882	67
2.	3		1161	72
3.	3		927	69
4.	3		1203	78
5.	3		862	67

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
Population Size: 30
Number of generations: 10

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.