

Code for Robot Path Planning using Fuzzy Logic

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 publically is permitted without permission.

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

1. Background

The code provided with this document uses Fuzzy Logic 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.

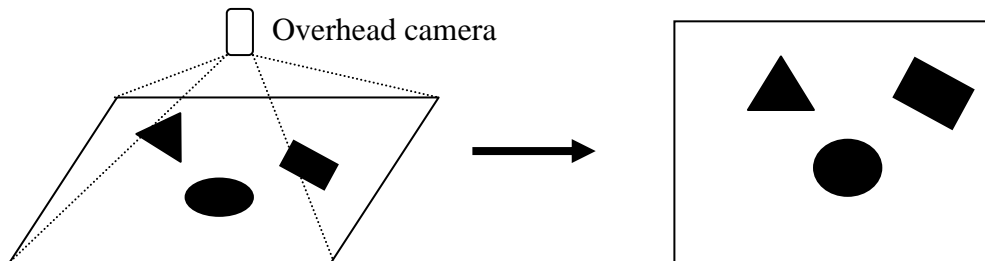


Figure 1: Overhead camera system and creation of robot map

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.

2. Problem Solving using Fuzzy Logic

The readers should please familiarize themselves with Fuzzy Logic before reading this tutorial. Fuzzy based navigation is a reactive planning technique, where the immediate position and distances from obstacles is considered to compute the immediate move, without much bothering about the future. In such a manner immediate actions lead to motion of the robot, ultimately leading to the goal. In order to solve the problem using fuzzy logic, we first need to select a few inputs which best represent the situation that the robot is currently placed in. The decision of motion is made purely on the basis of these inputs and not the actual scenario. For this problem 6 inputs are selected. These are distance from the obstacle in front, distance from the obstacle at the front left diagonal, distance from the

obstacle at the front right diagonal, angle between the heading direction of robot and the goal, distance of the robot from the goal and preferred turn. The different inputs are summarized in Figure 2.

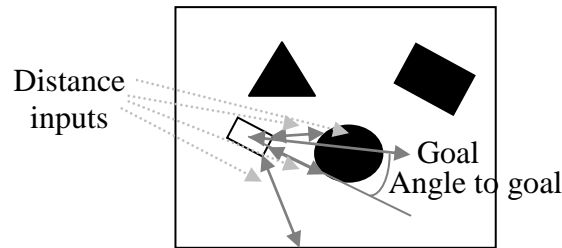


Figure 2: Different inputs for the fuzzy planning

The last input, preferred turn indicates whether it would be beneficial to turn clockwise or anti-clockwise, all other inputs ignored. A simple rule is used to set the parameter. If the front obstacle is far away, turn is so as to more face the goal. If the front obstacle is close and a new front obstacle is encountered, turn using the side of the goal is preferred. If the front obstacle is close and the same obstacle as encountered in the previous step is found, the same turn as made previously is repeated.

The fuzzy system produces a single output, which is the steering to make or the immediate angular speed. The fuzzy rules are written such that the robot avoids the obstacles and aligns itself towards the goal. The fuzzy system is a result of a lot of manual tuning of the rules and membership functions, over a wide variety of scenarios.

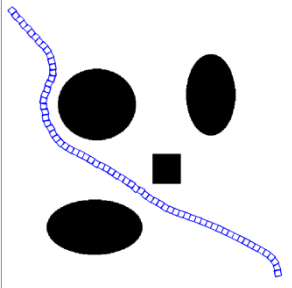
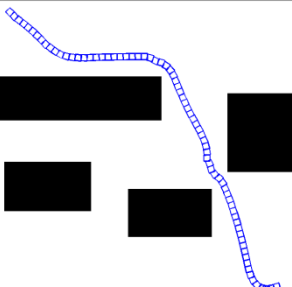
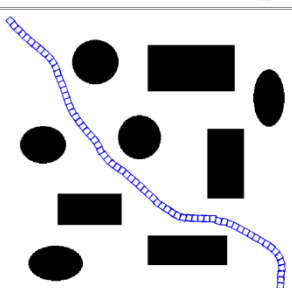
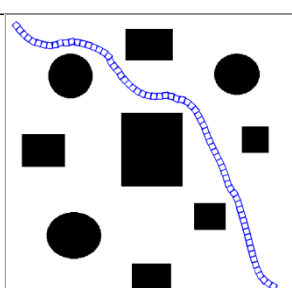
3. How to Execute and Parameters

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 and the initial heading direction (in radians). 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. Robot static and kinematic parameters may be changed to suit the requirements. However this will change the ideal fuzzy inference system. Changes to the rules and the membership functions can be carried at the fuzzy editor.

Execute the algorithm. You will need to take screenshots of the display, the tool does not do that for you. The simulation will stop when the robot reaches its goal, collides or is forced to stop as per the safety precautions. The path length and execution time are given at the console.

3. Sample Results

S. No.	Path	Path Length	Execution Time (sec)
1.		904	1.008

2.		899	1.013
3.		901	0.997
4.		898	0.900
5.		900	0.947

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

For all results:

source=[20 20]

goal=[480 480]

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.