Pathfinding with Obstacle Avoidance Using Fuzzy Logic

Jelle van Dijk
University of Amsterdam
Student ID: 10989048

Gawan Dekker
University of Amsterdam
Student ID: 11025654

Victor Gladys
University of Amsterdam
Student ID: 10523626

***Abstract*—The abstract goes here (to be done later).**

I. Introduction

Self driving cars are being developed at almost every major car and tech company in the world right now. Although they are still relatively early in development, they are considered by many to be the future of personal transportation. To try and gain a better understanding of autonomous moving vehicles, in this project we attempt to recreate a simple one using a fuzzy logic system (FLS).

The goal of this project is to create and design a FLS that can guide a robot through different environments without colliding with any obstacles. Because of limited time and recourses we have not implemented this system for an actual vehicle or robot, instead, we worked with a simulation. Both the FLS and the simulation are created by ourselves. We then applied our FLS to our simulated robot in place of an obstacle avoidance algorithm.

II. Literature Review

There is already a lot of literature about trying to make a robot traverse an environment using fuzzy logic. In this section, we will briefly discuss three papers that are relevant to our problem.

In “Artificial Neural Fuzzy Logic Algorithm for Robot Path Finding” [1], X. Bajrami, A. Dërmaku and N. Demaku create and compare two fuzzy logic implementations to guide a robot from point A to point B in a simulated environment with obstacles.

The first implementation uses a combination of FLSs with mamdani-type inferrence and handmade rulebases and membership functions to guide the robot. The input (distances to closest objects and goal, angle towards goal and preferred turn) is used to determine acceleration levels for the left and right wheels of the robot for different objectives (like obstacle avoidance) and to determine different weights for those objectives.

The second implementation used an FLS with a Sugeno-type inference system, where the membership functions were trained using a neuro-adaptive learning method.

After running tests in a simulated environment, the two implementations were compared.

The first implementation in the paper comes close to what we have implemented. We have used the the distance to obstacles in three different directions as inputs to give information about the area the robot is moving towards. And give two angles of rotation, one to the left and one to the right, as outputs. These are combined to create a single rotation angle.

In “A Novel Hybrid Fuzzy A* Robot Navigation System for Target Pursuit and Obstacle Avoidance” [2], the researcher created a hybrid system to control a robot. The first layer of the system consisted of an A* algorithm. The A* algorithm is a path finding algorithm, it is considered to be very fast. The A* path finding layer calculated the optimal route to from the position of the robot to the end point, this route consisted of way points. The second layer of the system is a fuzzy logic system. This system had as input information about the next way point as well as about the nearest obstacle. With these pieces of information a speed and turning speed for the robot were chosen.

This paper used some techniques that can be very useful for our project. Since they used a moving end point in their simulations, their test scenarios where harder. However, the same principles they used to control their robot will apply to the robot in the simulations used in our project.

One problem that the researchers in [2] encountered, was that in some situation the fuzzy logic system would try to avoid an obstacle and by doing so, would not be able to move to the next way point. This is something to keep in mind for our project, but could be fixed by tweaking the membership functions.

In “Adaptive two layer fuzzy control of a mobile robot system” [3], a genetic algorithm has been used to adapt fuzzy rules in a two layer fuzzy logic system, which is then used by two robots to navigate towards a target without colliding. The genetic learning has been applied to generate a new layer of fuzzy rules that can be integrated into an already existing rulebase.

The first of the two fuzzy layers is used to determine the angle at which to continue moving, whilst the second layer determines the speed of the robot. By encoding the rulebase into a bitstring, Mohammadian *et al.* were able to modify the rulebases with a genetic algorithm using cross-over and single-point mutations over a number of generations. To do this they used a modified cross-over procedure that ensured that these bitstrings were cut only at points that defined boundaries between rules.

The paper concludes in noting that, altough most of their tests resulted in a positive outcome, some had trouble at the corners of the tested driving areas. This suggests that genetic algorithms find a maximum in optimizing fuzzy systems that is not necessarily easily applicable to a change in, or extension of, the initial learning environment.

III. Proposed Approach

A. Design

Our Fuzzy Logic System uses three inputs and gives two outputs. The inputs we use are the distance to the nearest object in three different directions: front, front-left and front-right. The output is given in the form of two angles a left angle and a right angle, the robot will move in the direction of the combination of these two angles.

Our input variables have a mix of triangular and trapezoidal membership functions ( ). For all three inputs the range (in pixels) and membership functions are equal. The placement of the functions is chosen such that only a very small distance (about the size of the robot) is seen as a small distance, anything above about 5x the robots size is seen as a large distance and in between is a medium amount of distance. The functions have a reasonable amount of overlap since the random obstacle movement in our environment is cause for a decent amount of overlap.

We use a full rulebase of 27 rules for our system ( ). In general, the rules are chosen such that the robot makes sharper turns if the distance to the objects in front of it is smaller and will always prioritize a left turn over a right turn.

B. Implementation

¡¡¡¡¡¡¡ HEAD We used python to build both the Fuzzy Logic System implementation and the simulated environment. For the simulations we primarily used the pygame package.

1) Fuzzy Logic System implementation: We implemented a variety of possible membership functions for our FLS implementation. Most of which use standard functions to return membership. For the triangular and trapezoidal membership functions, however, we used a sequence of if-else statements to calculate membership, rather than taking the minimum value from two lines. This way it was easier to make sure we do not accidentally devide by zero.

For our input and output variables, we simply record a name, range and list of membership functions. When calculating the membership, we return a dictionary with all memberships of a datapoint for the membership functions of the variable.

For our rules, we implemented AND and OR rules. AND rules can be calculated using the minimum operation or the algebraic product. OR rules can be calculated using the max operation or the algebraic sum.

2) Simulated Environment: The environment our robot moves around in is a box with an amount of obstacles. The obstacles are rectangular in shape and move in a random direction. Every certain amount of timesteps, the direction in which the obstacles move changes in a random new direction. The robot moves around in this environment at a fixed speed and has to avoid colliding with the obstacles. The change in angle of the robot at every tick is determined by the FLS.

The distance from the robot to an object are determined by raycasting. For each timestep, the robot casts two rays slightly around the desired direction. With this ray, the distance to the nearest object (that includes the walls of the simulation) is determined. The distance in that direction is then the smallest value returned from those two rays. This is done for angles around 0 degrees, 35 degrees and -35 degrees and the distances are passed to the FLS. The FLS then determines a left and right turning degree and the middle of those two angles is taken as the change in directions that the robot follows.

C. Link to repository

https://gitlab-fnwi.uva.nl/10989048/fuzzy_logic_final_project

IV. Experiments and Results

We will conduct our experiments by running two kinds of series of simulations. Firstly we will measure the average amount of frames until a collision occurs, secondly the average amount of collisions over a number of frames. We will do these measurements for various MFs. (Triangular, trapezoidal, gaussian, bellshaped, sigmoidal)

We have chosen to use frames as measurement of time since this allows us to speed up our simulations. The reason why we use two kinds of series of simulations is that we can get different kinds of information from them: From the first we only get to know what the average time (in frames) is between getting into a collision from a random starting position, but the second one tells us about the average amount of collisions in a given time period (in frames), meaning that we get to know more about the avergae time between collisions.

V. Discussion

The discussion goes here (to be done later).

References

[1] X. Bajrami, A. Dërmaku, and N. Demaku, "Artificial neural fuzzy logic algorithm for robot path finding," *IFAC-PapersOnLine*, vol. 48, no. 24, pp. 123–127, 2015.

[2] A. P. Gerdelan and N. H. Reyes, "A novel hybrid fuzzy a* robot navigation system for target pursuit and obstacle avoidance," in *Proceedings of the First Korean-New Zealand Joint Workshop on Advance of Computational Intelligence Methods and Applications*, vol. 1, 2006, pp. 75–79.

[3] M. Mohammadian and R. J. Stonier, "Adaptive two layer fuzzy control of a mobile robot system," in *Evolutionary Computation, 1995., IEEE International Conference on*, vol. 1. IEEE, 1995, p. 204.