INTELLIGENT SYSTEMS AND ROBOTICS ASSIGNMENT REPORT

A REPORT

submitted as part of the assignment

in

CE801 – Intelligent Systems and Robotics

by

ISHWAR VENUGOPAL

(1906084)

School of Computer Science and Electronic Engineering
University of Essex

January 2020

ABSTRACT

Researchers have long been trying to imitate the human reasoning and decision making process. This has led to the development of many behaviour-based control algorithms. Fuzzy logic is one such approach which takes into account the uncertainties in the human thoughts that lead to decisions, which has proved to work very efficiently in real world scenarios. The work presented here consists of two tasks. The first tasks deals with implementing a PID controller on a mobile robot to achieve a right edge following behaviour. The second task deals with creating fuzzy logic systems for a right edge following behaviour and an obstacle avoidance behaviour separately, which later was combined using the techniques of subsumption and context blending. Fuzzy logic systems have increasingly been applied to real life systems and is affecting our day-to-day lives in many ways ranging from automated vacuum cleaners to self-driving cars.

CONTENTS

1. INTRODUCTION

The broad area comprising the design, manufacture, control and programming of robots is called robotics. Recent years have seen immense increase in the use of robots to undertake tasks which would otherwise be practically not feasible to humans due to extreme environmental conditions or minute-scale sophistication. Studies on programming and designing robots have mainly been inspired from studying the behavioural systems of human beings and animals in nature [5]. The term robotics was first coined by the science fiction writer Isaac Asimov, who also laid down the three fundamental rules for robotics [21]. The rules formulated by Asimov were widely interpreted and followed [21, 4] . The three laws as quoted from the "Handbook of Robotics, 56th Edition, 2058 A.D." are as follows [21]:

"1) A robot may not injure a human being or, through inaction, allow a human being to come in harm.

2) A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.

3) A robot must protect its own existence as long as such protection does not conflict with the First or Second Law."

The first humanoid robot was revealed in the NY World's Fair in 1921, and was called the Elektro [2]. And later on in 1956, the world's first robot company called the Unimation was established. In 1961, PUMA (Programmable Universal Machine for Assembly) became the first Industrial robot that came into existence [29]. At this stage, robots were just used to perform mechanical tasks, it was not anywhere close to the logic and reasoning powers of actual human beings.

When robots are capable of performing useful function which are guided by the desired outcome and also based on the knowledge it acquires, it is said to be an 'intelligent'

system. Such systems are capable of adapting their functions in accordance to the changes in the environment. In 1970, SRI International created the first mobile robot which was capable of reasoning and responding to changes in its environment through artificial intelligence [15, 26] . Thereafter, the world has witnessed immense development in the field of robotics in applications ranging from computer controlled walking machines and space satellite technologies.

1.1 Robot Control

Robots as we now see have sensors for input, control systems for decision-making and end effectors for output. The earliest robot control architectures revolved around the sense-think-act paradigm, which as the name suggests, is an alogrithm that makes the robot sense its environment using various sensors and makes it come up with a plan to be executed to achieve a specified outcome. This technique was developed by SRI in 1967 [15, 26].

Further improvements to the techniques of robot control was focused on intelligent behaviours i.e. robots being able to adapt to robust environments. Efforts were made to create something that made the robot perform simple but effective actions at each step. This gave rise to the domain of behaviour based robotics [5].

1.2 Fuzzy Logic Systems

According to [19], it was observed that “”more often than not, the classes of objects encountered in the real physical world do not have precisely defined criteria of membership”. This gave rise to the notion of fuzzy sets. A new perspective towards approaching problems with notions of approximate mental representations began with this idea. Fuzzy sets takes into consideration the day-to-day usage of natural language terms to describe a measure of quantities, using functions called the membership functions [11]. This notion of uncertainty forms an import part of the control systems in intelligent robots [8, 28].

The concept of fuzzy logic was designed to allow the computer to understand the differences in data similar to human reasoning. The wide range of applications of fuzzy logic systems is due to the fact that it is a machine learning technique which is flexible

and easy to implement. Being able to imitate human thought gives it an upper-hand in solving problems which deals with uncertain or approximate reasoning [3]. The control systems based on fuzzy logic is based on a rule base which categorizes the fuzzified inputs to certain values. The defuzzification process then gives the actual output values required for the system.

1.3 Applications of robots

Robots have wide ranges of applications, of which we will be discussing a few notable ones. In the field of Assistive robots, studies have now been conducted in designing robots that can interact both physically and socially with the human being it is attached to [27]. This has been applied to areas like care of the aged people, assisiting people with physical recovery due to any injury or disease and has now also opened doors towards assisting individuals with social disabilities [27].

In environments which are harsh and uninhabitable for human beings, like outer space and extra-terrestrial conditions, robots have come in very handy. For advantages ranging from reducing space, weight and operation cost to having the ability to self-repair and adapt to the unanticipated changes in the environment, such robots are being widely used for space explorations [30].

Aerial robots are now being widely used for search and rescue operations. It can be used for various assistance to humans in tasks like locating vulnerable victims, transporting and efficiently supplying rescue items in harsh environments [6]. Robots are more efficient in transporting heavy materials over harsh conditions as compared to humans and are being increasingly deployed for the same [24].

One of the major use of robots is in the field of medicine. Sophisticated surgeries, highly efficient drug delivery and controlled medical processes are now being increasingly done with the help of advanced nano-sized robots [7, 20]. Micro and nano sized robots have been observed to be efficiently able to deliver drugs to the affected sites and thereby reducing the side effects that would have been otherwise caused by highly toxic drugs [20].

Another major area where robots have been deployed to perform large amounts of work is agriculture. For increasing productivity, reducing the human labour and farming

machinery to produce large yield, large scale farmers around the world have started to use the effectiveness of the products designed by the robotics industry [17, 18].

Robots can thus be seen in almost all areas of life at these times, and the use of robots is only set to increase in the coming years. Even with all these advantages of using robots instead of human labour, there lies an inherent threat of human unemployment and complete dependency of humans on machines. These are issues worth looking at in the years to come.

2. ROBOT CONTROL TECHNIQUES

2.1 PID controller

Proportional Integral Derivative (PID) controllers are used to create a closed loop feedback mechanism to control various systems, robots in this particular case. This mechanism help to maintain a process at the target level or value that we want it to work at. In the context of this assignment, the task at hand is to implement a right edge following behaviour on a robot using a PID controller. We take the error between the desired distance and the current distance to be the input of the controller. The output produced is the proportional angular speed of the wheels of the robot.

For the proper functioning of a PID controller, we need to calculate the following errors in the following sequence:

$$e \ = desired\, distance \ - \ current\, distance \tag{2.1}$$

$$e_i = e_i + e \tag{2.2}$$

$$e_d = e - e_{previous} \tag{2.3}$$

$$e_{previous} = e \tag{2.4}$$

e, e_i and e_d are called the PID errors and the corresponding coefficients: k_p,k_i and k_d are called the PID parameters. The PID output is then calculated as:

$$Output = k_p * e + k_i * e_i + k_d * e_d \tag{2.5}$$

This PID output can then be converted to the left and right motor speeds of the robot using the following equations:

$$leftVel = baseVel - \frac{\omega d}{2} \tag{2.6}$$

$$rightVel = baseVel + \frac{\omega d}{2} \tag{2.7}$$

Here baseVel is a constant that is set beforehand, d is the constant distance between the two wheels and ω is the desired angular speed. The PID output and the desired angular speed can be connected using the relation -d$\frac{}{2}\omega$.

2.1.1 Algorithm

The basic algorithm for the implementation of a PID controller is presented below. In each time step, do the following:

- Get the sonar readings from the robot sensors
- Calculate the current distance of the robot from the wall. This can be done by combining readings from multiple sensors using proper trigonometric relations.
- Ignore the readings that are beyond the maximum range of the sensors (Maximum range of the sonar sensors used is 5000)
- Calculate the PID errors, namely the current error, integral error and the difference between the current and the previous error
- Calculate the PID output
- Calculate the left and right motor speeds of the robot using the appropriate equations
- Set the calculated speeds to the robot wheels

In this case, we have used the sensors at 90^o and 50^o for calculating the current distance of the robot from the wall [Figure.2.1]. We have used the horizontal component of the sensor at 50^o and taken its average with the other sensor. This helps to get a better reading than directly combining both the sensor readings straightaway.

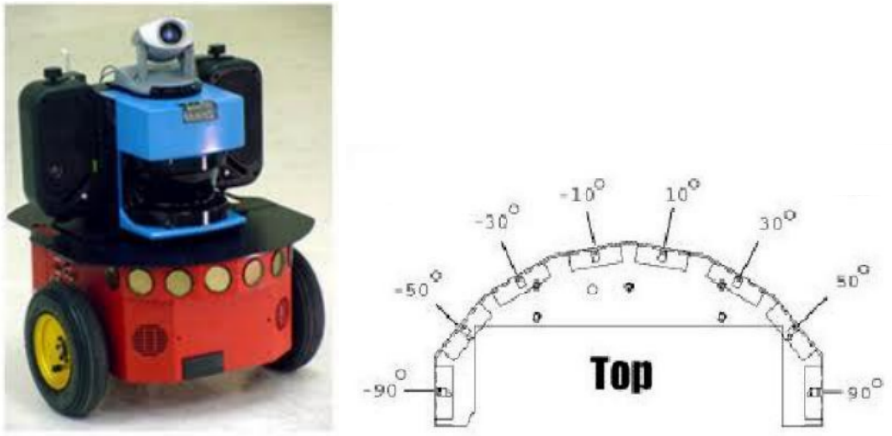

Figure 2.1: (a) Mobile robot - Pioneer which has been used for the experiments (b) The orientation of the sonar sensors in front of the robot

2.1.2 Applications of PID controller

PID controllers find a variety of use in the industrial sector. One of the major used of the PID control algorithm is to control the temperature on large furnaces [12, 14]. The errors that arise due to the changes in the levels of gases like carbon monoxide or any other gas of relevance can be used as the PID input for the controller to work. In the areas of experimental chemistry research and in chemical industries, maintaining a constant pH is a highle challenging task. This can also be solved by using a PID control system [13]. While furnaces correspond to an open system, the applications of a PID controller can also be extended to closed systems in batch temperature control. The noise in the system can be monitored to provide a suitable PID input [23].

2.2 Fuzzy Logic Control

Fuzzy Logic control is an example of behaviour-based control of the robot. Behaviours can refer to any of the actions that fall into the category of directional behaviour/exploration, goal-oriented behaviour, aversive/protective behaviours or path following behaviours. In this assignment we are trying to address the problem for Right Edge following (path following behaviour) and obstacle avoidance (protective behaviours) for a robot as in Figure.2.1 using Fuzzy logic control.

Some basic concepts that build a fuzzy logic system are fuzzy sets, membership func-

tions, linguistic variables and fuzzy rule base. Fuzzy set refers to a set without a clearly defined boundary. It is usually represented as triangles, trapezoids etc. Membership functions are curves that define how each point in the input space becomes mapped to a certain degree of membership attaining values between 0 and 1. Linguistic variables are those variables which use words as values rather than numbers, which identifies closely to how human beings think in the real world. Rules consisting of if-then statements which are later used to formulate conditional rules in the program are called the Fuzzy rules. A set of such rules form the rule base for the system.

2.2.1 Algorithm

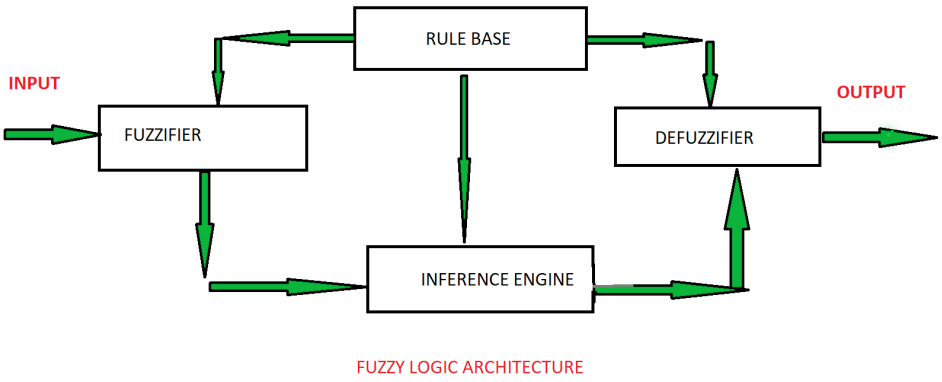

Figure 2.2: Flowchart describing the processes involved in a fuzzy logic control system [1]

The first step after obtaining crisp inputs from the robot is fuzzification. This refers to the conversion of the crisp numbers to fuzzy sets. The membership values corresponding to each fuzzy set is retained for each crisp input value. The inference system defines the degree to which the current inputs match with the rules laid down in the fuzzy rule base. It decides which rules are to be fired according the fuzzy sets the crisp inputs belong to. This leads us to the final step which is the defuzzification. It this step, the fuzzy sets obtained from the inference engine is again converted back to a crisp value so that it can be used in the real world scenario. In our case, the crisp outputs correspond to the left motor speed and the right motor speed.

2.2.2 Right Edge Following

For the right edge following behaviour, we had two inputs. The first input was a minimum of the sensor reading at 10^o, 30^o and 50^o [2.1]. The second input was the sonar sensor at 90^o . The rule base is as shown in Figure.2.3. The outputs corresponds to the left motor speed and the right motor speed of the robot. The membership functions and the corresponding fuzzy sets used for the inputs and the outputs have been illustrated in Figure.2.4

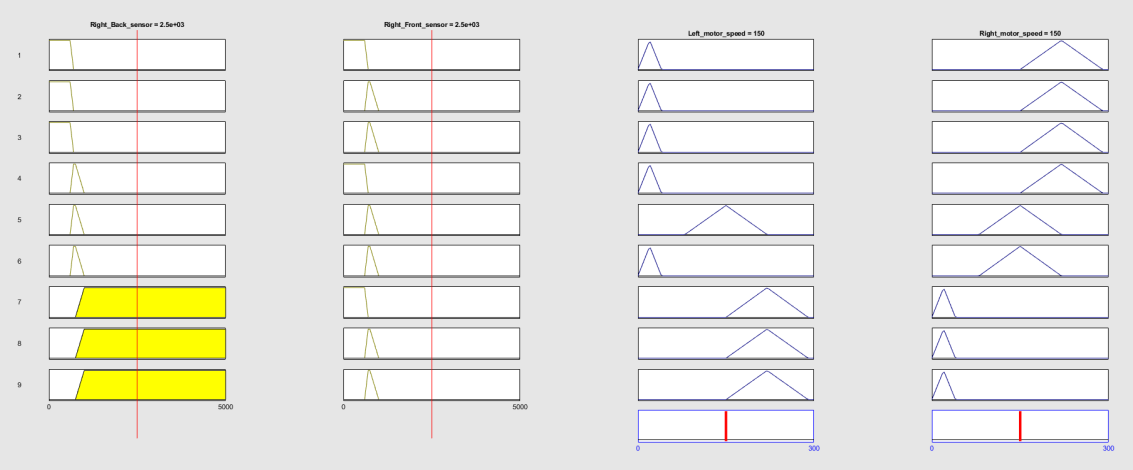

Figure 2.3: The rule base used for the right edge following behaviour

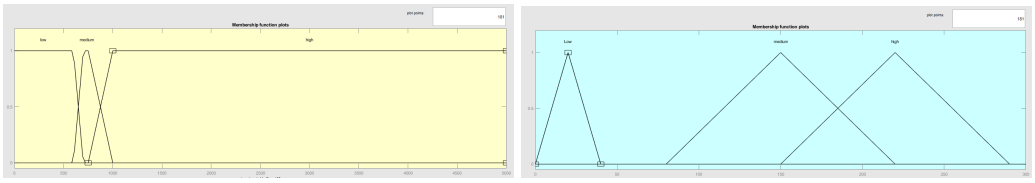

Figure 2.4: (a) The input membership functions used for the right edge following behaviour (b) The output membership functions used for the left and right motor speeds of the robot

The surface diagrams for two particular instances as generated from the fuzzy toolkit in MATLAB has also been illustrated in Figure.2.5. The rules basically makes the robot move towards the right if its too far away from the wall, and to go straight if it maintains a 'medium' distance from the wall.

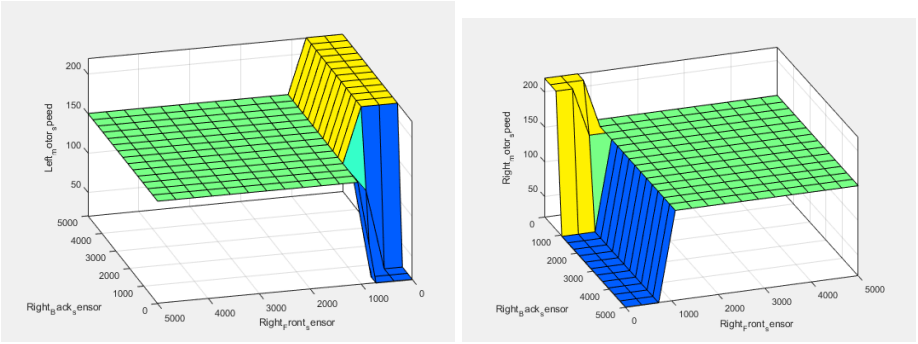

Figure 2.5: (a) Surface diagram plotted for right back sensor, right front sensor and the left motor speed (b) Surface diagram plotted for right back sensor, right front sensor and the right motor speed

2.2.3 Obstacle Avoidance

In obstacle avoidance behaviour, we have three inputs. The front sensor, the left sensor (oriented towards the direction of motion) and the right sensor (aligned to the direction of motion). The outputs remain the same, namely the left motor speed and the right motor speed of the robot. The input and output membership functions and the corresponding fuzzy sets are as shown in Figure.2.7. The rule base and a sample of a surface diagram as generated in MATLAB is illustrated in Figure.2.6 and Figure.2.8.

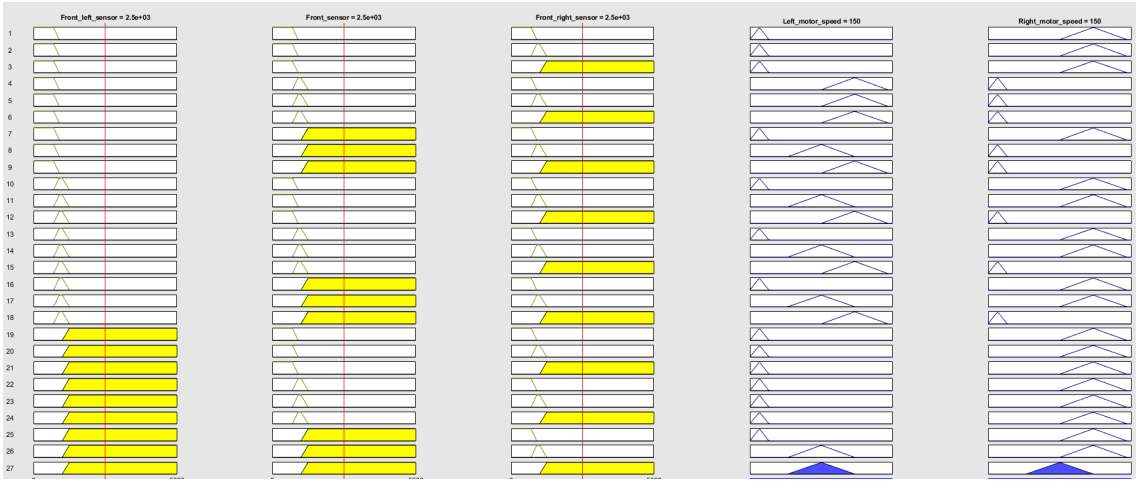

Figure 2.6: The rule base used for the obstacle avoidance behaviour

The rule base in this case is trying to create a protective behaviour for the robot. Hence, if it comes too close to an object to the right then it moves left. Similarly, when the robot comes too close to an object on its left then it turns right. If it encounters a very

close object directly at the front, it can turn either left or right; both of which have shown basically the same kind of results.

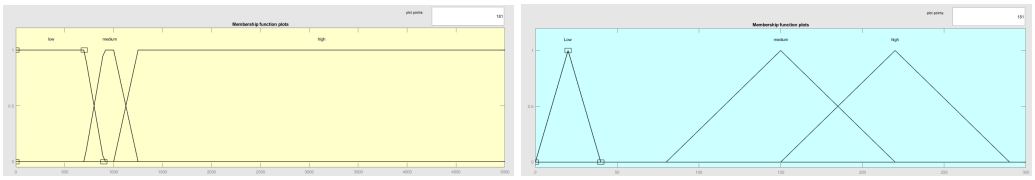

Figure 2.7: (a) The input membership functions used for the obstacle avoidance behaviour (b) The output membership functions used for the left and right motor speeds of the robot

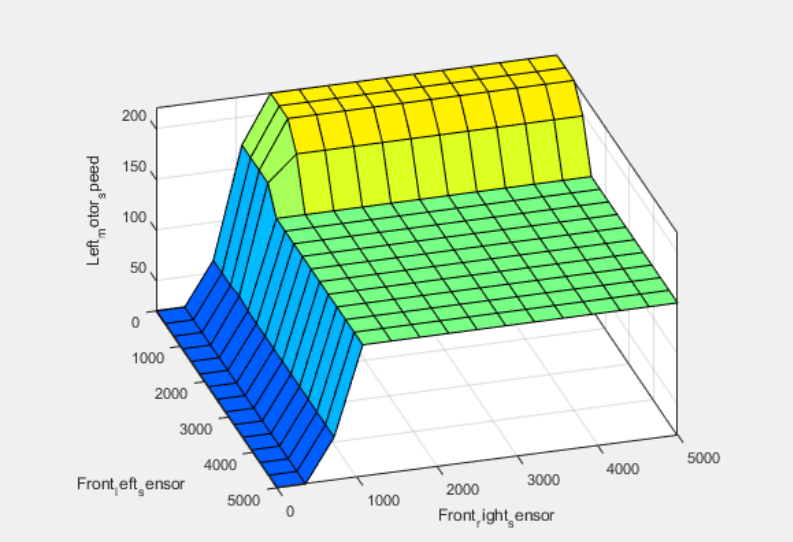

Figure 2.8: A sample surface diagram plotted for front left sensor, front right sensor and the left motor speed

2.2.4 Applications of Fuzzy Logic

Fuzzy logic has been applied to various systems helping in our day-to-day lives. Ranging from vacuum cleaners at home to face detection systems [16]. Some of the worth mentioning non-trivial applications include developing systems which can automatically detect fires that may come up in the batteries of electrical vehicles [9]. Fuzzy logic systems have also been used to simplify the process of data analysis to some extent and making much more effective results out the data [10]. There have also been a large amount of interesting researches using fuzzy logic, some of which are the studies on how the human work efficiency is affected by the growing noise pollution in the road traffic [22]! The

main aim of fuzzy logic systems are to be able to have the reasoning power similar to humans. There have been studies which have researched on the effectiveness of using fuzzy logic to improve the decision making process [25]. The applications and use of fuzzy logic systems are increasing day by day.

2.3 Subsumption Architecture

The Right Edge following behaviour and the Obstacle avoidance behaviour can be combined in various ways. In this subsection and the next, we will be discussing two such techniques.

Subsumption architecture is basically a set of if-else conditions which determine what kind of behaviour is to be activated based on the sensor reading to the front of the robot. As per intuition, if the robot is very close to some object then only the obstacle avoidance behaviour must be activated. If the robot is very far from any obstacle, only right edge following behaviour is preferred. The boundary between when these two behaviours are switched is very crisp in subsumption.

In our experiments, the distance which was used to decide which behaviour had to activated was the minimum of the readings of the sensors at -30^o, -10^o, 10^o and 50^o [2.1]. And the threshold distance was kept to be 600mm.

2.4 Context Blending

In context blending, the separation between when to switch between the right edge following behaviour and the obstacle avoidance behaviour is not as crisp as in subsumption. We use membership functions and a corresponding fuzzy set [Figure.2.9] to decide when the robot is near or considerably far from an obstacle, unlike using a crisp threshold distance for carrying out the same.

The minimum of the readings of the sensors at -30^o, -10^o, 10^o and 50^o [2.1] is calculated. Then the corresponding membership values with respect to the fuzzy set described in Figure.2.9 is calculated. The final output is calculated by taking a weighted average of the output obtained from the right edge following behaviour as well as the obstacle avoidance behaviour.

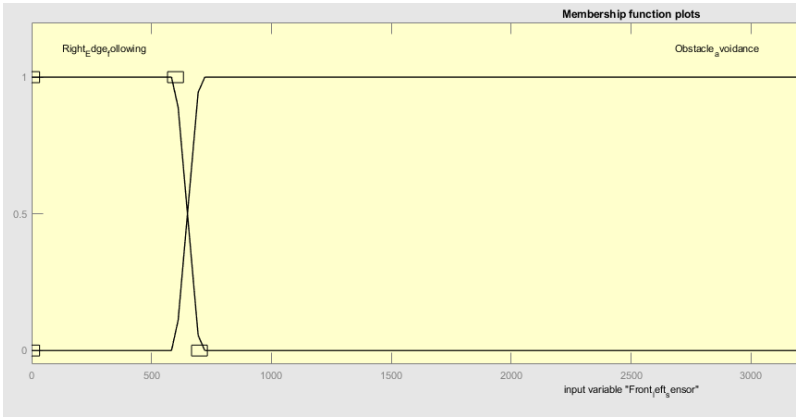

Figure 2.9: The membership functions used to determine the degree of each behaviour in the context blending architecture

We have used the centroid de-fuzzification method for the de-fuzzification step in all of the experiments described in the above sections. In the centroid defuzzification method, the crisp output is calculated as per the following equation:

$$crisp\ output = \frac{\sum f * c}{\sum f} \tag{2.8}$$

where f corresponds to the membership values and c corresponds to the centroids of the membership functions for the output variables. In our case, the centroids are: 20 for low, 150 for medium and 220 for high.

3. RESULTS AND DISCUSSIONS

3.1 PID Controller

For a PID controller, it was first observed that the robot was taking very sharp turns and moving at very high speeds. This was corrected by calculating the integral error only after certain specified time steps (in this case, 5), rather than adding them in every single time step. This gave rise to a much better performance of the robot.

The desired distance from the wall at the right edge was kept as 500mm. The robot performs very well when the walls are wrapped with bubble-wraps. This make the sensor readings to bounce back ideally as it is meant to be. For very shiny surface, the sensor beams are deflected elsewhere and ultimately the obstacle is not detected.

The final values of the parameters were as follows: $k_p = 0.6$, $k_i = 0.005$ and $k_d = 0.02$. Many different combinations of values were tried on the robot before arriving on these final values for the parameters.

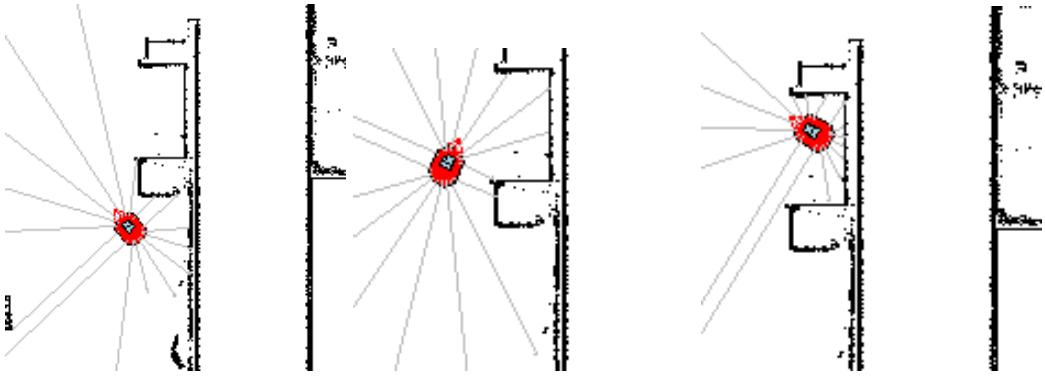

Figure 3.1: Screenshots from a trial run on the simulator for a right-edge following behaviour (from left to right) using a PID controller

3.2 Fuzzy Logic Control Systems

The right edge following and the obstacle avoidance behaviours were first tested separately. The main challenge while combining these two behaviours was that, care had to be taken that the rules from the independent rule bases of each behaviour did not cancel each other out when combined. Initially in some cases while testing context blending architecture, it was observed that the robot seemed to stop when it encountered something very close to it at the front. This was because the right-edge following behaviour was asking the robot to turn left whereas at the same time the obstacle avoidance behaviour was asking the robot to turn right at the very same instant. Effectively, the robot did not turn to any side and stopped moving. This was solved by altering the rule base of the obstacle avoidance behaviour and make it turn left when it encounters any obstacle very close to it at the front.

The main observation after implementing both subsumption and context blending architecture was that context blending provided much more smoother turns near corners as compared to the sharp turns around obstacles for subsumption. This is evident from the fact that one of them uses crisp threshold conditions to decide which behaviour to follow and the other one uses a fuzzy set to determine the same.

For the step where the program decides which rules to fire, I had tried using the AND operator and also the minimum operator. Even though the values of the rules changed, it hardly affected the final performance of the robot.

The program that was created to demonstrate the fuzzy logic control systems [refer Appendix] was made in such a way that some lines could be commented each time it runs depending on what behaviour we want the robot to have. The same program can be used to illustrate the right edge following behaviour and the obstacle avoidance behaviour separately; and also to demonstrate the subsumption and context blending architecture separately.

The final values for the membership functions and the centroid values for the output variables were chosen after trial runs in the lab with different combination of values.

4. CONCLUSIONS

A PID controller was implemented on a mobile robot to follow a right edge following behaviour. With proper tuning of the parameters, the robot had a relatively good performance in the laboratory.

For the fuzzy logic control systems, different sets of rule bases and membership functions were formed for right edge following behaviour and obstacle avoidance behaviour. Both these behaviours were later combined in two methodologies, namely subsumption and context blending. Even as both the techniques worked without much faults, context blending seemed to give a much more smoother working of the robot in the laboratory.

4.1 Future Directions

With regard to the work done in this assignment, the rule base can be further modified to obtain a smoother transition in the surface diagrams [Figure.2.5, Figure.2.8]. Smoother transitions corresponds to smoother turns when it comes to robots avoiding obstacles and turning around the corners. Laser sensors can also be included along with sonar sensors to get a better performance on the robot.

BIBLIOGRAPHY

[1] Fuzzy logic — introduction. https://www.geeksforgeeks.org/fuzzy-logic-introduction/.

[2] Md. Akhtaruzzaman and A. A. Shafie. Evolution of humanoid robot and contribution of various countries in advancing the research and development of the platform. *ICCAS 2010*, 1 2011.

[3] Pedro Albertos and Antonio Sala. Fuzzy logic controllers: Advantages and drawbacks. *Congresso de la Asociacion Chitena de Control Aoutmatico*, 3, 9 2998.

[4] S.L Anderson. Asimov's "three laws of robotics" and machine metaethics. *AI Soc 22, 477–493*, 2008.

[5] Ronald C. Arkin. *Behavior-based Robotics*. THE MIT Press, 1998.

[6] Markus Bernard, Konstantin Kondak, Ivan Maza, and Anibal Ollero. Autonomous transportation and deployment with aerial robots for search and rescue missions. *Autonomous Robots*, 14, 10 2011.

[7] Jessica Burgner-Kahrs, D. Caleb Rucker, and Howie Choset. Continuum robots for medical applications: A survey. 2015.

[8] H Bustince and P Burillo. Vague sets are intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 79, 1996.

[9] Macam S. Dattathreya, Harpreet Singh, and Thomas Meitzler. Detection and elimination of a potential fire in engine and battery compartments of hybrid electric vehicles. *Advances in Fuzzy Systems*, 2012.

[10] Rahul Dixit and Harpreet Singh. Bdd, bnn, and fpga on fuzzy techniques for rapid system analysis. *Advances in Fuzzy Systems*, 2012.

[11] Didier Dubois, Walenty Ostasiewicz, and Henri Prade. *Fuzzy Sets: History and Basic Notions*. Kluwer Academic Publishers, 2000.

[12] P E. Ziółkowskia and Śmierciak. Comparison of energy consumption in the classical (pid) and fuzzy control of foundry resistance furnace. *ARCHIVES of FOUNDRY ENGINEERING*, 12, 3 2012.

[13] Nio Tiong Ghee, S. Kumaresan, and Liau Chung Fan. Fuzzy pid controller to control the ph neutralization process. *Research and Development (SCOReD), Student Conference on*, 7 2002.

[14] E. Grassi and K. Tsakalis. Pid controller tuning by frequency loop-shaping: application to diffusion furnace temperature control. *IEEE Transactions on Control Systems Technology*, 8, 9 2000.

[15] Didier Guzzoni, Adam Cheyer, Luc Julia, and Kurt Konolige. Many robots make short work: Report of the sri international mobile robot team. *AI Magazine*, 18, 1997.

[16] Madan M. Gupta Harpreet Singh, Thomas Meitzler, Zeng-Guang Hou, Kum Kum Garg, Ashu M. G. Solo, and Lotfi A. Zadeh. Real-life applications of fuzzy logic. *Advances in Fuzzy Systems*, 2013.

[17] Jack Hollingum. Robots in agriculture. *Industrial Robot*, 8 1999.

[18] Noboru Kawamura and Kiyoshi Namikawa. Robots in agriculture. *Advanced Robotics*, 3, 4 2012.

[19] Zadeh L.A. Fuzzy sets. *Information Control*, 1965.

[20] Ming Luo M, Youzeng Feng, Tingwei Wang, and Jianguo Guan. Micro-/nanorobots at work in active drug delivery. *Advanced Functional Materials*, 28, 4 2018.

[21] N. G. Hockstein, C. G. Gourin, R. A. Faust, and D. J. Terris. A history of robots: from science fiction to surgical robotics. *Journal of Robotic Surgery*, 1, 1.

[22] Debasish Pal and Debasish Bhattacharya. Effect of road traffic noise pollution on human work efficiency in government offices, private organizations, and commercial business centres in agartala city using fuzzy expert system: A case study. *Advances in Fuzzy Systems*, 2012.

[23] Ramesh C Panda. *Introduction to PID Controllers: Theory, Tuning and Application to Frontier Areas*. InTech, 2012.

[24] Michael Rubenstein, Adrian Cabrera, Justin Werfel, Golnaz Habibi, James McLurkin, and Radhika Nagpal. Collective transport of complex objects by simple robots: theory and experiments. *Proceedings of the 2013 international conference on Autonomous agents and multi-agent*, 14, 5 2013.

[25] Ki-Young Song, Janusz Kozinski, and and Madan M. Gupta Gerald T. G. Seniuk. Excluded-mean-variance neural decision analyzer for qualitative group decision making. *Advances in Fuzzy Systems*, 2012.

[26] K. Sugihara and I. Suzuki. Distributed motion coordination of multiple mobile robots. *Proceedings. 5th IEEE International Symposium on Intelligent Control*, 1990.

[27] Adriana Tapus, Mataric Maja, and Brian Scassellatti. The grand challenges in socially assistive robotics. *IEEE Robotics and Automation Magazine, Institute of Electrical and Electronics Engineers*, 2007.

[28] Vicenç Torra. Hesitant fuzzy sets. *International Journal of Intelligent Systems*, 25, 3 2010.

[29] Wallén, Johanna, and Linköping. *The History of the Industrial Robot*. Linköping University Electronic Press, 2008.

[30] M. Yim, K. Roufas, and Duff. Modular reconfigurable robots in space applicationss. *Autonomous Robots*, 14, 2003.

Appendices

A. PID CONTROLLER CODE

```
#include "Aria.h"
#include <stdio.h>
#include <iostream>
#include <conio.h>
#include <math.h>

using namespace std;

int main(int argc, char** argv)
{
  // =========Initialization of Robot==========//

  Aria::init();
  ArRobot robot;
  ArArgumentParser argParser(&argc, argv);
  argParser.loadDefaultArguments();
  ArRobotConnector
    robotConnector(&argParser, &robot);
  if (robotConnector.connectRobot())
    std::cout << "Robot connected!" << std::endl;
  robot.runAsync(false);
  robot.lock();
  robot.enableMotors();
  robot.unlock();

  //=====Defining the variables==========//

  int act_dis, cur_dis; //Actual distance and current distance of the
   robot from the wall
```

```
int sense7, sense6, sens6_hor; //Variables to store the sensor 7 reading, sensor 6 and horizontal component of sensor 6 reading
act_dis = 500; //Actual distance to be maintained between the right edge and the wall
double kp, kd, ki, e, base_speed; //Parameters of the PID controller
double ei, ed, eprev, output;
ei = 0; //Initializing the ei values
eprev = 0;
double rms, lms; //Left and right motor speeds
double left, right;
int count = 0;

//=====Parameters of the PID controller======//

kp = 0.6;
ki = 0.005;
kd = 0.02;

while (true)
{
  //=========Getting Sonar Readings=============//
  ArSensorReading *sonarSensor[8];
  int sonarRange[8];
  for (int i = 0; i < 8; i++)
  {
    sonarSensor[i] = robot.getSonarReading(i);
    sonarRange[i] = sonarSensor[i]->getRange();
  }
  sense7 = sonarRange[7]; //Sensor 7 readings
  sense6 = sonarRange[6]; //Sensor 6 readings
  sens6_hor = sense6*cos(0.698); //Horizontal component of sensor 6 reading

  if ((sonarRange[6] <= 3000) || (sonarRange[7] <= 3000)) //To ignore readings when the robot is very far
  {
    cur_dis = min(sens6_hor, sense7); //Calculating the current distance of separation
```

```
62    }
63
64    e = act_dis - cur_dis; //The error in each step
65    if (count<5) //To control the rapid increase in ei values , this
    condition makes ei values increase only every five steps
66    {
67      ei += e;
68    }
69    else
70    {
71      ei = 0;
72      count = -1;
73    }
74    ed = e - eprev; //storing the ed value
75    eprev = e; //storing the previous error
76    output = (kp*e) + (ki*ei) + (kd*ed); //Getting the PID output
77    count++;
78    base_speed = 150; //Setting the base speed for the wheels
79    left = 0; right = 0; //Initializing both the left and right output
80    left = output; //Assigning the PID output to the left wheel (by
    trial and error)
81    lms = base_speed - left; //final left motor speed
82    rms = base_speed; //final right motor speed
83    robot.setVel2(lms, rms);
84
85    cout << "Inputs: "<<sense7 << " " << sense6 << " " << sens6_hor <<
    " " << endl;
86    cout << "Outputs: "<< lms << " " << rms << " " << endl;
87    ArUtil::sleep(100);
88
89  }
90  robot.lock();
91  robot.stop();
92  robot.unlock();
93  // terminate all threads and exit
94  Aria::exit();
95  return 0;
96
```

```
97 }
```

B. FUZZY LOGIC CONTROL SYSTEM CODE

```
#include <Aria.h>
#include <stdio.h>
#include <iostream>
#include <conio.h>

using namespace std;

struct membership_fn //a structure to define all the membership functions
{
  double x1, x2, x3, x4, centroid;
  void make_shape(double a, double b, double c, double d)
  {
    x1 = a; x2 = b; x3 = c; x4 = d;  //assigning the x values to the shapes
    centroid = (x1 + x2 + x3 + x4) / double(4); //calculating the centroid of the shapes
  }
};

double get_membership_value(membership_fn func, double input) //a function to get the membership values of the supplied inputs
{
  double output;
  if ((input > func.x1) && (input < func.x2))
    output = (input - func.x1) / (func.x2 - func.x1); //rising edge
```

```
  else if ((input >= func.x2) && (input <= func.x3))
    output = 1;
  else if ((input > func.x3) && (input < func.x4))
    output = (func.x4 - input) / (func.x4 - func.x3); //falling edge
  else
    output = 0.0001;

  return output;
}

struct rule_REF //a structure to define the rules in the rule base of
   Right Edge following
{
  double lms, rms;
  double value;
  void define_rule(double input1val, double input2val, double
   centroid_lms, double centroid_rms)
  {
    value = min(input1val, input2val); //taking the minimum to find the
    value correspoding to a  particular rule
    lms = centroid_lms; //lms output for that rule
    rms = centroid_rms; //rms output for that rule
  }
};

struct rule_OA //a structure to define the rules for Obstacle Avoidance
{
  double lms, rms;
  double value;
  void define_rule(double input1val, double input2val, double input3val
   , double centroid_lms, double centroid_rms)
  {
    value = min(input1val, input2val, input3val);
    lms = centroid_lms;
    rms = centroid_rms;
  }
};
```

```
int main(int argc, char **argv)
{
  //Defining Variables for Right Edge Following (REF)
  double REFinput1, REFinput2, REF_lms, REF_rms;
  double REFin1low, REFin1med, REFin1high, REFin2low, REFin2med,
   REFin2high;
  double REFlms_numerator, REFlms_denominator, REFrms_numerator,
   REFrms_denominator;
  int i, j;
  const int REF_no_of_rules = 9;
  rule_REF REFrulebase[REF_no_of_rules];

  //Defining Variables for Obstacle Avoidance (OA)
  double OAinput1, OAinput2, OAinput3, OA_lms, OA_rms;
  double OAin1low, OAin1med, OAin1high, OAin2low, OAin2med, OAin2high,
   OAin3low, OAin3med, OAin3high;
  double OAlms_numerator, OAlms_denominator, OArms_numerator,
   OArms_denominator;
  const int OA_no_of_rules = 27;
  rule_OA OArulebase[OA_no_of_rules];

  //Input membership functions for Right Edge Following
  membership_fn low_in, medium_in, high_in;
  low_in.make_shape(0, 0, 600, 700);
  medium_in.make_shape(600, 700, 750, 1000);
  high_in.make_shape(750, 1000, 5000, 5000);

  //Input membership functions for Obstacle Avoidance
  membership_fn low_inOA, medium_inOA, high_inOA;
  low_inOA.make_shape(0, 0, 700, 900);
  medium_inOA.make_shape(700, 900, 1000, 1250);
  high_inOA.make_shape(1000, 1250, 5000, 5000);

  //Output membership functions
  membership_fn low_out, medium_out, high_out;
  low_out.make_shape(0, 20, 20, 40);
  medium_out.make_shape(80, 150, 150, 220);
  high_out.make_shape(150, 220, 220, 290);
```

```
91
92  //=========Fuzzy Control Architecture (Initialization)==========//
93  membership_fn control_OA;
94  control_OA.make_shape(0, 0, 600, 700);
95  membership_fn control_REF;
96  control_REF.make_shape(600, 700, 5000, 5000);
97  double min_sens, control_inOA, control_inREF;
98  double final_lms, final_rms;
99
100 //============== Initialisation of Robot =============//
101 Aria::init();
102 ArRobot robot;
103 ArArgumentParser argParser(&argc, argv);
104 argParser.loadDefaultArguments();
105 ArRobotConnector robotConnector(&argParser, &robot);
106 if (robotConnector.connectRobot())
107   cout << "Robot Connected!" << endl;
108 robot.runAsync(false);
109 robot.lock();
110 robot.enableMotors();
111 robot.unlock();
112 ArSensorReading *sonarSensor[8];
113 int sonarRange[8];
114
115 while (true)
116 {
117   //========Getting Sonar Readings=========//
118   for (i = 0; i < 8; i++) {
119     sonarSensor[i] = robot.getSonarReading(i);
120     sonarRange[i] = sonarSensor[i]->getRange();
121   }
122   //=========Right Edge Following (Calculations)============//
123   REFinput1 = min(sonarRange[6], sonarRange[5], sonarRange[4]);
124   REFinput2 = sonarRange[7];
125
126   REFin1low = get_membership_value(low_in, REFinput1);
127   REFin1med = get_membership_value(medium_in, REFinput1);
128   REFin1high = get_membership_value(high_in, REFinput1);
```

```
129
130     REFin2low = get_membership_value(low_in, REFinput2);
131     REFin2med = get_membership_value(medium_in, REFinput2);
132     REFin2high = get_membership_value(high_in, REFinput2);
133
134     //Defining Rule Base
135     REFrulebase[0].define_rule(REFin1low, REFin2low, low_out.centroid,
    high_out.centroid);
136     REFrulebase[1].define_rule(REFin1low, REFin2med, low_out.centroid,
    high_out.centroid);
137     REFrulebase[2].define_rule(REFin1low, REFin2high, low_out.centroid,
     high_out.centroid);
138     REFrulebase[3].define_rule(REFin1med, REFin2low, low_out.centroid,
    high_out.centroid);
139     REFrulebase[4].define_rule(REFin1med, REFin2med, medium_out.
    centroid, medium_out.centroid);
140     REFrulebase[5].define_rule(REFin1med, REFin2high, low_out.centroid,
     medium_out.centroid);
141     REFrulebase[6].define_rule(REFin1high, REFin2low, high_out.centroid
    , low_out.centroid);
142     REFrulebase[7].define_rule(REFin1high, REFin2med, high_out.centroid
    , low_out.centroid);
143     REFrulebase[8].define_rule(REFin1high, REFin2high, high_out.
    centroid, low_out.centroid);
144
145     REFlms_numerator = double(0);
146     REFlms_denominator = double(0);
147     REFrms_numerator = double(0);
148     REFrms_denominator = double(0);
149
150     for (i = 0;i<9;i++) //calculating the output values
151     {
152       REFlms_numerator += (REFrulebase[i].value)*(REFrulebase[i].lms);
153       REFlms_denominator += REFrulebase[i].value;
154       REFrms_numerator += (REFrulebase[i].value)*(REFrulebase[i].rms);
155       REFrms_denominator += REFrulebase[i].value;
156     }
157
```

```
158     REF_lms = REFlms_numerator / REFlms_denominator;
159     REF_rms = REFrms_numerator / REFrms_denominator;
160 
161     //=========Obstacle Avoidance (Calculations)===========//
162     OAinput1 = sonarRange[2];
163     OAinput2 = min(sonarRange[3], sonarRange[4]);
164     OAinput3 = sonarRange[5];
165 
166     OAin1low = get_membership_value(low_inOA, OAinput1);
167     OAin1med = get_membership_value(medium_inOA, OAinput1);
168     OAin1high = get_membership_value(high_inOA, OAinput1);
169 
170     OAin2low = get_membership_value(low_inOA, OAinput2);
171     OAin2med = get_membership_value(medium_inOA, OAinput2);
172     OAin2high = get_membership_value(high_inOA, OAinput2);
173 
174     OAin3low = get_membership_value(low_inOA, OAinput3);
175     OAin3med = get_membership_value(medium_inOA, OAinput3);
176     OAin3high = get_membership_value(high_inOA, OAinput3);
177 
178     //Defining Rule Base
179     OArulebase[0].define_rule(OAin1low, OAin2low, OAin3low, low_out.
    centroid, high_out.centroid);
180     OArulebase[1].define_rule(OAin1low, OAin2low, OAin3med, low_out.
    centroid, high_out.centroid);
181     OArulebase[2].define_rule(OAin1low, OAin2low, OAin3high, low_out.
    centroid, high_out.centroid);
182     OArulebase[3].define_rule(OAin1low, OAin2med, OAin3low, high_out.
    centroid, low_out.centroid);
183     OArulebase[4].define_rule(OAin1low, OAin2med, OAin3med, high_out.
    centroid, low_out.centroid);
184     OArulebase[5].define_rule(OAin1low, OAin2med, OAin3high, high_out.
    centroid, low_out.centroid);
185     OArulebase[6].define_rule(OAin1low, OAin2high, OAin3low, low_out.
    centroid, high_out.centroid);
186     OArulebase[7].define_rule(OAin1low, OAin2high, OAin3med, medium_out
    .centroid, low_out.centroid);
187     OArulebase[8].define_rule(OAin1low, OAin2high, OAin3high, high_out.
```

```
centroid, low_out.centroid);
188  OArulebase[9].define_rule(OAin1med, OAin2low, OAin3low, low_out.
centroid, high_out.centroid);
189  OArulebase[10].define_rule(OAin1med, OAin2low, OAin3med, medium_out
.centroid, high_out.centroid);
190  OArulebase[11].define_rule(OAin1med, OAin2low, OAin3high, high_out.
centroid, low_out.centroid);
191  OArulebase[12].define_rule(OAin1med, OAin2med, OAin3low, low_out.
centroid, high_out.centroid);
192  OArulebase[13].define_rule(OAin1med, OAin2med, OAin3med, medium_out
.centroid, high_out.centroid);
193  OArulebase[14].define_rule(OAin1med, OAin2med, OAin3high, high_out.
centroid, low_out.centroid);
194  OArulebase[15].define_rule(OAin1med, OAin2high, OAin3low, low_out.
centroid, high_out.centroid);
195  OArulebase[16].define_rule(OAin1med, OAin2high, OAin3med,
medium_out.centroid, high_out.centroid);
196  OArulebase[17].define_rule(OAin1med, OAin2high, OAin3high, high_out
.centroid, low_out.centroid);
197  OArulebase[18].define_rule(OAin1high, OAin2low, OAin3low, low_out.
centroid, high_out.centroid);
198  OArulebase[19].define_rule(OAin1high, OAin2low, OAin3med, low_out.
centroid, high_out.centroid);
199  OArulebase[20].define_rule(OAin1high, OAin2low, OAin3high, low_out.
centroid, high_out.centroid);
200  OArulebase[21].define_rule(OAin1high, OAin2med, OAin3low, low_out.
centroid, high_out.centroid);
201  OArulebase[22].define_rule(OAin1high, OAin2med, OAin3med, low_out.
centroid, high_out.centroid);
202  OArulebase[23].define_rule(OAin1high, OAin2med, OAin3high, low_out.
centroid, high_out.centroid);
203  OArulebase[24].define_rule(OAin1high, OAin2high, OAin3low, low_out.
centroid, high_out.centroid);
204  OArulebase[25].define_rule(OAin1high, OAin2high, OAin3med,
medium_out.centroid, high_out.centroid);
205  OArulebase[26].define_rule(OAin1high, OAin2high, OAin3high,
medium_out.centroid, medium_out.centroid);
206
```

```
    OAlms_numerator = double(0);
    OAlms_denominator = double(0);
    OArms_numerator = double(0);
    OArms_denominator = double(0);

    for (i = 0;i<OA_no_of_rules;i++) //calculating the output values
    {
      OAlms_numerator += (OArulebase[i].value)*(OArulebase[i].lms);
      OAlms_denominator += OArulebase[i].value;
      OArms_numerator += (OArulebase[i].value)*(OArulebase[i].rms);
      OArms_denominator += OArulebase[i].value;
    }

    OA_lms = OAlms_numerator / OAlms_denominator;
    OA_rms = OArms_numerator / OArms_denominator;

    //======Context Blending=======//
    /*
    min_sens = min(sonarRange[3], sonarRange[4], sonarRange[5]);
    control_inOA = get_membership_value(control_OA, min_sens);
    control_inREF = get_membership_value(control_REF, min_sens);
    final_lms = ((control_inOA*OA_lms) + (control_inREF*REF_lms)) / (
control_inOA + control_inREF);
    final_rms = ((control_inOA*OA_rms) + (control_inREF*REF_rms)) / (
control_inOA + control_inREF);
    robot.setVel2(final_lms, final_rms);
    */

    //=======Subsumption ============//
    /*
    min_sens = min(sonarRange[3], sonarRange[4], sonarRange[5]);
    if (min_sens<600)
      robot.setVel2(OA_lms, OA_rms);
    else
      robot.setVel2(REF_lms, REF_rms);
    */
    //========Only Obstacle Avoidance==========//
    //robot.setVel2(OA_lms, OA_rms);
```

```
    //=========================================//

    //========Only Right edge following=========//
    robot.setVel2(REF_lms, REF_rms);
    //=========================================//
  }


  // termination (SLIDE 47)
  // stop the robot
  robot.lock();
  robot.stop();
  robot.unlock();
  // terminate all threads and exit
  Aria::exit();
  return 0;
}
```