**B. E. PROJECT ON**

# FUZZY METHODS IN AUTOMATED VEHICLES

**Submitted by:**


| | |
|---|---|
| Mehak Malhotra | 2018UIC3016 |
| Mudit Mahajan | 2018UIC3018 |
| Raghav Kaushal | 2018UIC3020 |
| Shloka Gupta | 2018UIC3150 |


**Under the Guidance of:**

Dr Arjun Tyagi

Project-I in partial fulfillment of the requirement for the award of

B.E. in Instrumentation & Control Engineering



**DIVISION OF INSTRUMENTATION AND CONTROL ENGINEERING**

**NETAJI SUBHAS INSTITUTE OF TECHNOLOGY**

**(UNIVERSITY OF DELHI)**

**NEW DELHI - 110078**


**DECEMBER, 2021**

**Department of Instrumentation and Control Engineering**

University of Delhi

New Delhi - 110078

## CERTIFICATE OF ORIGINALITY

We, Mehak Malhotra, 2018UIC3016, Mudit Mahajan, 2018UIC3018, Raghav Kaushal, 2018UIC3020, Shloka Gupta, 2018UIC3150 hereby declare that the Project-Thesis titled 'Fuzzy Methods in Automated Vehicles' which has been submitted to the Department of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, Delhi (University of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering is original and not copied from the source without proper citation. The manuscript has been subjected to plagiarism checks by Turnitin software. This work has not previously formed the basis for the award of any Degree.

Place: Delhi
Date: 19 December 2021

(2018UIC3016) Mehak Malhotra

(2018UIC3018) Mudit Mahajan

(2018UIC3020) Raghav Kaushal

(2018UIC3150) Shloka Gupta


**Department of Instrumentation and Control Engineering**

University of Delhi

New Delhi - 110078

## CERTIFICATE OF DECLARATION

This is to certify that the Project-Thesis titled 'Fuzzy Methods in Automated Vehicles' which is being submitted by Mehak Malhotra, 2018UIC3016, Mudit Mahajan, 2018UIC3018, Raghav Kaushal, 2018UIC3020, Shloka Gupta, 2018UIC3150 to the Department of Computer Engineering, Netaji Subhas Institute of Technology (University of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the thesis work carried out by the students under my supervision and guidance. The content of this thesis, in full or in parts, has not been submitted for any other degree or diploma.

**Prof. Arjun Tyagi**
Professor
Dept. of Instrumentation and Control Engineering
Netaji Subhas University of Technology
(Formerly Netaji Subhas Institute of Technology)
Azad Hind Fauj Marg, Sector-3
Dwarka, New Delhi, PIN-110078

# ACKNOWLEDGEMENTS

We are highly grateful to the Division of Instrumentation and Control Engineering, Netaji Subhas University of Technology (NSUT) for providing us with this opportunity to carry out the project work. The constant guidance and encouragement received from our supervisor, Dr Arjun Tyagi and Head of Department (Division of Instrumentation and Control Engineering) has been of great help in carrying out our present work. Literature and experimentation are greatly acknowledged with reverential thanks. Finally, we would like to express gratitude to all the faculty members of the Division of Instrumentation and Control Engineering, NSUT for their support throughout the course of this project work.

## PLAGIARISM REPORT

Fuzzy Methods in Automated Vehicles

ORIGINALITY REPORT

| 10% | 7% | 7% | % |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | kclpure.kcl.ac.uk<br>Internet Source | 2% |
| 2 | Vineet Kumar, K. P. S. Rana, Jitendra Kumar, Puneet Mishra, Sreejith S. Nair. "A robust fractional order fuzzy P + fuzzy I + fuzzy D controller for nonlinear and uncertain system", International Journal of Automation and Computing, 2016<br>Publication | 2% |
| 3 | www.palgrave.com<br>Internet Source | 1% |
| 4 | digitalcommons.unf.edu<br>Internet Source | 1% |
| 5 | Shreya Bhattacharya, Sujay Ray. "In silico screening and exploration into phenotypic alterations of deleterious oncogenic single nucleotide polymorphisms in HSPB1 gene", Genomics, 2021<br>Publication | 1% |

| | | |
|---|---|---|
| 6 | Zdzislaw Gosiewski, Zbigniew Kulesza. "Virtual collocation of sensors and actuators for a flexible rotor supported by active magnetic bearings", Proceedings of the 14th International Carpathian Control Conference (ICCC), 2013<br>Publication | 1 % |
| 7 | midwestcri.org<br>Internet Source | 1 % |
| 8 | Pravek Dwivedi, Rajesh Gujar. "Impact on Transportation Due to Metro Link Express for Gandhinagar and Ahmedabad (MEGA)", Urbanization Challenges in Emerging Economies, 2018<br>Publication | 1 % |

| | | | |
|---|---|---|---|
| Exclude quotes | Off | Exclude matches | Off |
| Exclude bibliography | Off | | |

# ABSTRACT

When we talk of Boolean logic the truth-value is binary - either 0 or 1. However, when we talk of fuzzy logic, values may also be between 0 to 1. Here, 0 & 1 are the extreme values. Fuzzy logic finds its application in a wide range of engineering fields, from building household washing machines and kitchen appliances to sophisticated car control systems, image processing and space vehicle control. The application we intend to consider in this paper is the automatic car driving system. This paper describes an automotive vehicle control system in which the vehicle keeps a certain distance from the one in front of it and applies brakes using fuzzy logic. If the vehicle detects another object or a vehicle in a certain range, it changes lanes and overtakes the object/vehicle and returns to its original lane. Computers are increasingly taking on driver-related tasks. Among those functions are to maintain reference speed or to keep a safe distance from other vehicles, to improve night vision with infrared cameras, and to build and provide maps. However, many traffic situations remain complex and difficult to control, especially in urban areas. Driving is a category of problems based on the basic systems of logical reasoning and coping with uncertainty. Therefore, in order to upgrade automotive computers to monitor and perform tasks related to visual acuity or driving, we must integrate aspects of human intelligence and behavior so that vehicles can handle driving actuators in a human-like manner.

# CONTENTS

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| CAGR | Compound annual growth rate |
| GA | Genetic algorithm |
| MF | Membership function |

# CHAPTER 1

# INTRODUCTION

The global autonomous cars market reached a value of nearly $818.6 billion in 2019, having increased at a compound annual growth rate (CAGR) of 12.7% since 2015[1]. The market is expected to decline from $818.6 billion in 2019 to $772.8 billion in 2020 at a rate of -5.6%[1]. The decline is mainly due to lockdown and social distancing norms imposed by various countries and economic slowdown across countries owing to the COVID-19 outbreak and the measures to contain it.

The market is then expected to recover and grow at a CAGR of 12.7% from 2021 and reach $1,191.8 billion in 2023. The market is expected to reach $1,642.9 billion in 2025 growing at a CAGR of 17.4%, and $3,195.0 billion in 2030 growing at a CAGR of 14.2%[1].

The manual driving systems involve the risk of accidents, unnecessary wastage of fuels at stop and go points. Soon after the roads started to get populated with automated and semi-automated vehicles there has been a 500% increase in lane capacity, 90% reduction in traffic accident deaths, and 10% improvement in fuel economy[2].

## 1.1 MOTIVATION

While the market share of automated vehicles is growing, there are still many applications where we can leverage the automatic driving technology to find solutions for the betterment of humanity. Mars rover, old age assistant driving, are some of the applications where automated driving techniques are employed, but can be further optimized. By optimization we mean, techniques like Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, etc. can be employed to optimize the components of the automated control system like a phase controller[6][7]. Studies have shown that it is possible for a fuzzy phase controller to steer on a path[3], the motivation behind this thesis work was to use genetic algorithms to optimize a phase controller such that the vehicle can provide the maximum possible ride comfort.

## 1.2 LITERATURE SURVEY

In order to carry out our project we have reviewed multiple research papers which either worked on the same topic as our project or in part worked in our project.

[3] presents a fuzzy logic control system for steering a two-axle vehicle. Two controllers are presented: the steering controller and the velocity controller. Each fuzzy controller is divided into several modules to represent the distributed way in which humans deal with different driving tasks. All of these modules are of the Mamdani Type and use sigmoid or product of sigmoid membership functions. The outputs of the various modules are added together to control the steering angle and the speed of the vehicle, respectively.

[4] addresses the evolution of fuzzy systems for core applications of automotive engineering. It describes the use of fuzzy methods for driver modelling and driver assistant systems.

[5] describes an automatic car driving system in which the vehicle keeps a distance from the vehicle in front of it and applies brakes using fuzzy logic.

[6] describes the use of natural selection methods like GA to optimize parameters of a fuzzy controller under study for integral of absolute error.

[7] proposes a methodology to optimize fuzzy logic parameters based on Genetic Algorithms. The scheme is applied to the problem of electrical signal frequency driving for signals acquisition experiments. The fuzzy logic controller is tuned by Genetic Algorithms until it achieves the optimal parameters. The tuning design approach offers a complete and fast way to design an optimal fuzzy system. Moreover, the results show that the optimized fuzzy controller gives better performance than a conventional fuzzy controller also in terms of rise and settling time.

## 1.3 CHALLENGES AND OBJECTIVES

From the literature survey, we concluded that there is a need to optimize the phase controller in the automated car driving system. The execution time of the program is

very high as it involves a lot of calculation. This time can successfully be reduced through optimization.

The objective of this paper is to design a phase controller such that the car moves on a path in a way that the riding comfort is maximized and the vehicle moves with the minimum possible error.

It is difficult to lay down a definite set of driving rules for a vehicle in a country like India where traffic rules are not adhered to in a lot of cases. Therefore, the data acquired by the sensors can be stored to get further insights in the steering dynamics of the simulation in non-ideal conditions.

The heavy calculations involved in the genetic algorithm evaluation involve the calculation of distance at every instant. We have reduced this repeated calculation to reduce the time taken for its execution.

# CHAPTER 2

# PROPOSED SOLUTION

The following approach tries to optimize the parameters of a fuzzy system by using genetic algorithms. A fuzzy system is attached to the vehicle which is going through a predetermined path. The goal of the vehicle is to reach the end of the path. A comparative analysis is conducted of the fuzzy phase controller and the optimized fuzzy phase controller. The solution is presented as a command line application which creates simulations of the vehicle steering on a track.

## 2.1 PROJECT ARCHITECTURE

The vehicle is taken to be small rectangular object inside a bounded coordinate path in a two-dimensional coordinate system. The object has three sensors - two on either side & one on the front. The distance sensor gives an estimate of how far the obstacles in front, left and right are. We define these distances as $D_f$, $D_L$ and $D_M$ and the range of view as ($D_f$, $D_L$, $D_M$). The range of view and velocity act as an input to a fuzzy phase controller to determine the angle and velocity at which the car vehicle should move.

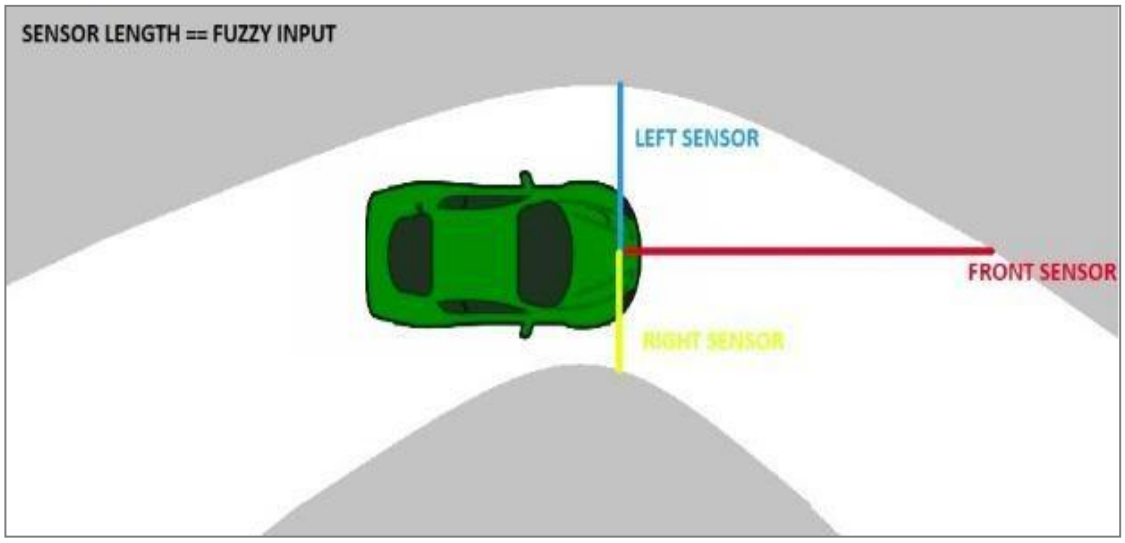


*Fig 2.1: Location of sensors on the car*

### 2.1.1 ROAD GENERATION

Two types of roads were used to test the algorithm:

i. Constant winding path generated by sinusoidal function

```
def generate_sin_path():
    coords1 = []
    coords2 = []
    offset = 200
    num_of_points = 20

    for i in range(constants.SCREEN_WIDTH):
        coords1.append((i, math.sin(0.01*i) * 200 + 250))

    coords2 = lt.apply_translation(0, offset, coords1)
    coords2.reverse()
    polygon = coords1 + coords2

    is_closed = False
    return polygon, is_closed

  #Fig. Code to generate a sinusoidal path
```

ii. Randomly generated closed convex path

```
def generate_convex_polygon():

    coords = [(constants.CAR_POS_X, constants.CAR_POS_Y + 10)]
    num_of_points = 20

    for i in range (num_of_points):
        x = random.randrange(constants.SCREEN_WIDTH)
        y = random.randrange(constants.SCREEN_HEIGHT)
        coords.append((x, y))

    coords1 = ch.gift_wrap(coords)

    scaling_factor = 0.6
    offset_x = 200
    offset_y = 150
    coords2 = lt.apply_scaling(scaling_factor, scaling_factor, coords1)
    coords2 = lt.apply_translation(offset_x, offset_y, coords2)

    polygon = [coords1, coords2]

    is_closed = True
    return polygon, is_closed

#Fig. Code to generate a closed convex path
```

The code snippets shown above will generate a sinusoidal path and a convex path on which the algorithm will be tested. The following images show the preview of the path generated.

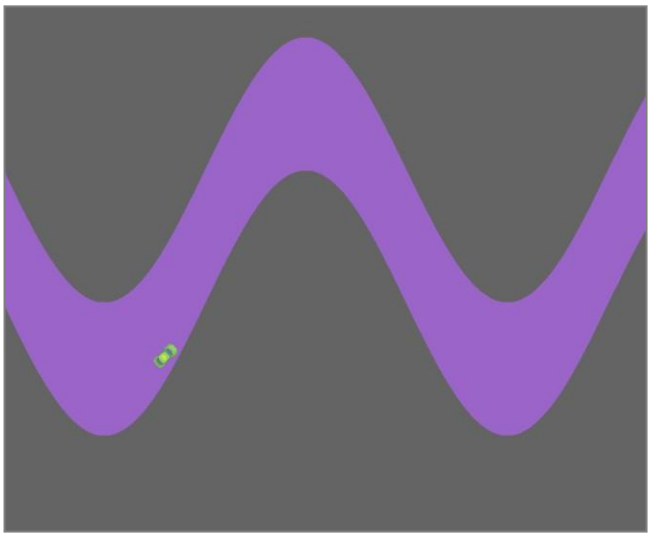

*Fig 2.2: Sinusoidal path*

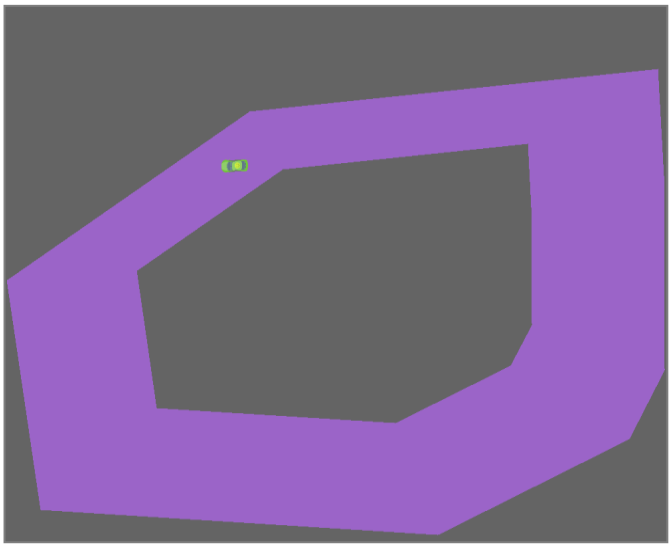

*Fig 2.3: Convex path*

## 2.2 COMPONENTS OF THE PROJECT

The four main components involved in the project are sensors, fuzzy logic, phase controller, and genetic algorithm.

### 2.2.1 SENSORS

Sensors are used in the simulation to calculate the distance between the vehicle and the ends of the road. These distances are used as the input for the fuzzy system. In this case, all the sensors are located towards the front of the vehicle. We distinguish three types of sensors:

i. **Left sensor:** It is present at the left side of the vehicle. It calculates the distance between the vehicle and the end of the road to the left of the vehicle.

ii. **Front sensor:** It calculates the distance between the vehicle and the end of the road in front of the vehicle.

iii. **Right sensor:** calculates the distance between the vehicle and the end of the road to the right of the vehicle.

The following is the method to calculate the intersection point. To calculate the length of the sensor line, we need to find the point on the plane where the sensor line intersects the path. If we fix the rotation of the vehicle, then the front sensor is parallel to the x-axis, and the left and right sensors are parallel to the y-axis. Just like in the picture, we can iterate along the x and y coordinates of the bitmap until we reach the point of intersection. We use a similar idea when vehicle rotation is not fixed.

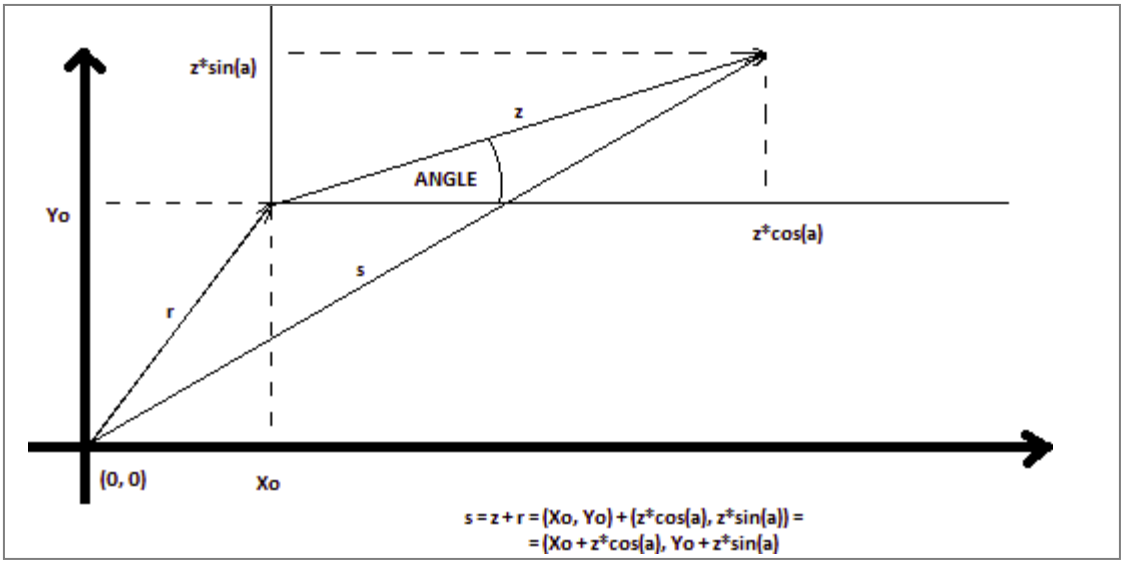


*Fig 2.4: Finding the intersection point of car trajectory and boundary*

i. Let **r = (Xo, Yo)** vehicle position vector, **a** vehicle rotation angle and let **z** end point of the sensor in the system with the reference point (Xo, Yo), where **| z |** vector intensity;

ii. The angle of the vector **z** for the front sensor is equal to the angle of rotation of the vehicle, i.e. **a**:

$$\mathbf{z = (|\, z\, | \cdot \cos(a), |\, z\, | \cdot \sin(a)) = (z \cdot Cos(a), z \cdot \sin(a))}$$

iii. The position of the end point of the sensor **S (z)** is equal to **r * z**;

iv. We can increase the intensity of the vector **z** until the point **S** becomes the point of intersection;

v. In each iteration we calculate **S (z)** as:

$$\mathbf{s(z) = r + z = (Xo + z \cdot \cos(a), Yo + z \cdot \sin(a))}$$

```
def sensor(self, name, screen, angle_direction):
    angle = -self.angle/180*math.pi
    pos_x = self.center_position().x
    pos_y = self.center_position().y
    z = 0
    while(valid_position(int(pos_x), int(pos_y)) and screen.get_at((int(pos_x), int(pos_y))) ==
constants.PATH_COLOR):
        z = z + 1
```

```
            pos_x = self.center_position().x + z*math.cos(angle - angle_direction)
            pos_y = self.center_position().y + z*math.sin(angle - angle_direction)
        sensor_input = str(distance(self.center_position().x, self.center_position().y, pos_x, pos_y))
        return sensor_input
# Fig. Method to calculate the input to the sensors
```

### 2.2.2 FUZZY LOGIC

Fuzzy logic is an approach to variable processing that allows for multiple possible truth values to be processed through the same variable. Fuzzy logic attempts to solve problems with an open, imprecise spectrum of data and heuristics that makes it possible to obtain an array of accurate conclusions.

Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input.

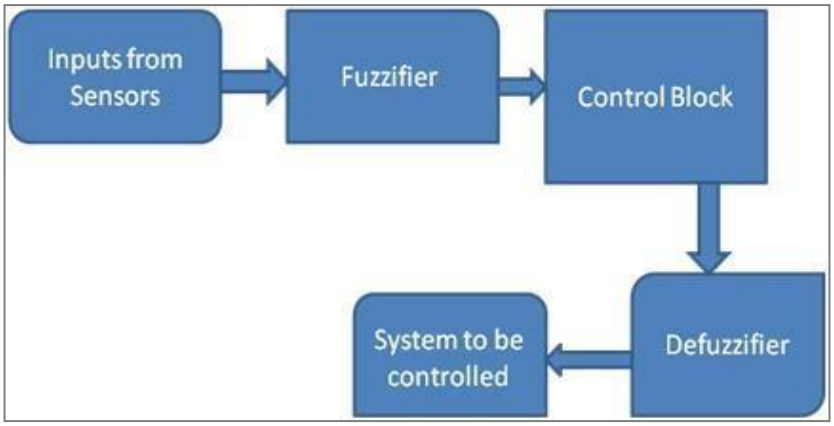


*Fig 2.5: Fuzzy logic control system*

A Fuzzifier transforms the measured or the input variables in numerical forms into linguistic variables.

A Controller performs approximate reasoning based on the human way of interpretation to achieve control logic. The controller consists of the knowledge base and the inference engine. The knowledge base consists of the membership functions and the fuzzy rules, which are obtained by knowledge of the system operation according to the environment.

The Defuzzifier converts this fuzzy output to the required output to control the system.

### 2.2.3 MEMBERSHIP FUNCTIONS

A membership function for a fuzzy set A on the universe of discourse X is defined as $\mu_A$:X → [0,1], where each element of X is mapped to a value between 0 and 1. This value quantifies the grade of membership of the element in X to the fuzzy set A.

Examples of membership functions are:

Triangular: tri (x; a, b, c) = $max\{min\{x - a/b - a, c - x/c - b\},0\}$ Trapezoidal:

trap (x: a, b, c, d) = $max\{min\{x - a/b - a, d - x/d - c, 1\},0\}$ Gaussian: gauss (x:

c, σ) = $exp[-1/2(x - c/\sigma)^2]$

### 2.2.4 FUZZIFICATION

Fuzzification is the process of converting a crisp input value to a fuzzy value that is performed by the use of the information in the knowledge base. Although various types of curves can be seen in literature, Gaussian, triangular, and trapezoidal membership functions are the most commonly used in the fuzzification process.

Methods of fuzzification: Intuition, inference, rank ordering, angular fuzzy sets, neural network.

### 2.2.5 DEFUZZIFICATION

It is the inversion of fuzzification, where the mapping is done to convert the crisp results into fuzzy results but here the mapping is done to convert the fuzzy results into crisp results. This process is capable of generating a non-fuzzy control action which illustrates the probability distribution of an inferred fuzzy control action.

Defuzzification process can also be treated as the rounding off process, where a fuzzy set having a group of membership values on the unit interval is reduced to a single scalar quantity. There are various methods which can be used for the defuzzification,

some of the examples are: Maximum membership principle, centroid method, weighted average method, center of sums.

### 2.2.6 PHASE CONTROLLER

The phase controller is a device which is used to control the input given to a load. In our case, the car acts as a load and the power given by the engine acts as the input to the load.

The phase system consists of 3 input units (a set of characteristic functions), 2 output units and several rules. The distance of the vehicle to the end of the road represents the input data for the phase system. Based on the input data, the characteristic functions for each unit are calculated (fuzzification). By applying the rules, we calculate the characteristic functions for each output unit. Finally, we calculate the output data for each output unit as a centroid (phase shift).

### 2.2.7 GENETIC ALGORITHM

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to the next generation. In simple words, they simulate ‘survival of the fittest’ among individuals of consecutive generations for solving a problem. Each generation consists of a population of individuals and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This chain is the same as the chromosome.

Genetic algorithms are based on an analogy with genetic structure and behavior of chromosomes of the population. Following is the foundation of GAs based on this analogy:

i. Individuals in population compete for resources and mate.

ii. Those individuals who are successful (fittest) then mate to create more offspring than others.

iii. Genes from "fittest" parents propagate throughout the generation, that is sometimes parents create offspring which are better than either parent.

iv. Thus, each successive generation is more suited for their environment.

The population of individuals is maintained within a search space. Each individual represents a solution in search space for the given problem. Each individual is encoded as a finite length vector (similar to the chromosome) of the components. These variable components are analogous to Genes. Thus, a chromosome (individual) is composed of several genes (variable components).

A 'Fitness Score' is given to each individual which shows the ability of an individual to "compete". The individual having optimal fitness score or near optimal are sought. The genetic algorithm is used to maintain the population of n individuals along with their fitness score. The individuals having better fitness scores are given more chances to reproduce than others. The ones with better fitness scores are selected who mate and produce better offspring by combining chromosomes of parents. The population size is static so room has to be created for new arrivals. So, some individuals die and get replaced by new arrivals, eventually creating a new generation when all the mating opportunities of the old population are exhausted. It is hoped that over successive generations better solutions will arrive while least fit die. Each new generation has on average more "better genes" than the individual of previous generations. Thus, each

new generation has better “partial solutions” than previous generations. Once the offspring produced have no significant difference from offspring produced by previous populations, the population converges. The algorithm is said to be converged to a set of solutions for the problem.

Once the initial generation is created, the algorithm evolves the generation using following operators:

i. Selection Operator: The idea is to give preference to the individuals with good fitness scores and allow them to pass their genes to successive generations.

ii. Crossover Operator: This represents mating between individuals. Two individuals are selected using a selection operator and crossover sites are chosen randomly. Then the genes at these crossover sites are exchanged thus creating a completely new individual (offspring).

iii. Mutation Operator: The key idea is to insert random genes in offspring to maintain the diversity in the population to avoid premature convergence.

Fitness score is the number of characters which differ from characters in target string at a particular index. So, individuals having lower fitness value are given more preference.

### 2.3 OPTIMIZATION OF PHASE CONTROLLER

The fuzzy logic controller needs to be optimized using a genetic algorithm, because the genetic algorithm searches from a population of points, not a single point. So, it selects the best solution from a generation (or in our project the velocity and angle at which the cruise comfort is maximum). Genetic algorithms use payoff (objective function) information, not derivatives. They support multi-objective optimization so if we are to control a moving vehicle, we can also maintain the maximum comfort.

We optimize the phase controller by implementation of the Mamdani fuzzy implication [6]. The following classes have been used:

i. MFInput: Represents one characteristic function for input data, ie. calculates affiliation based on input.
ii. MFOutput: Represents one characteristic function for output data, ie. calculates affiliation based on defined rules.
iii. Rule: Represents one phase system rule.
iv. FuzzyInput: An entity that represents a set of characteristic input functions (MFInput).
v. FuzzyOutput: An entity that represents a set of characteristic output functions (MFOutput).
vi. FuzzySystem: A complete phase system consisting of a series of input units (FuzzyInput), a series of rules (Rule), and one output unit (FuzzyOutput).

Calling the fit method simulates the Mamdani process phase of the system for a given input (sensor input). The output value is in the solution attribute.

There are three input units:

i. left_sensor
ii. right_sensor
iii. front_sensor

Each of them has four characteristic functions for each unit separately defined. They represent the distance from the ends of the road.

i. close
ii. midrange
iii. far
iv. very far

The following is an example of an optimized solution with a smaller number of iterations and a smaller population of input units (left, front, right) = (5, 30, 15):

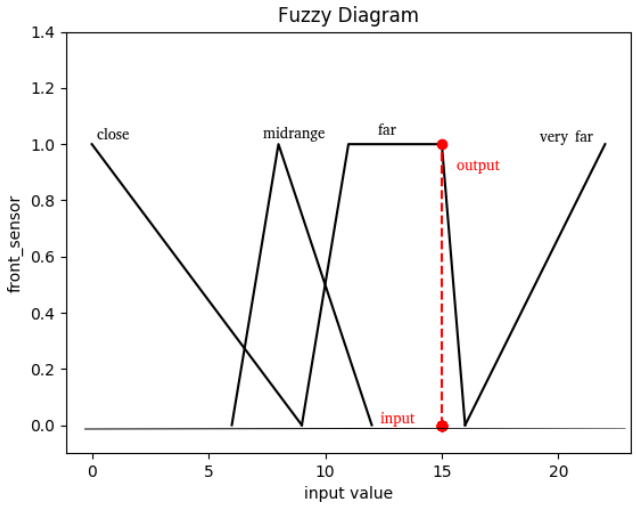


*Fig 2.6: Left sensor input*

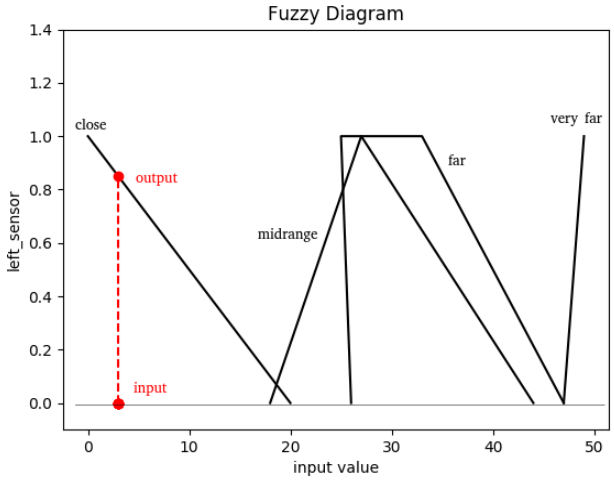


*Fig 2.7: Front sensor input*

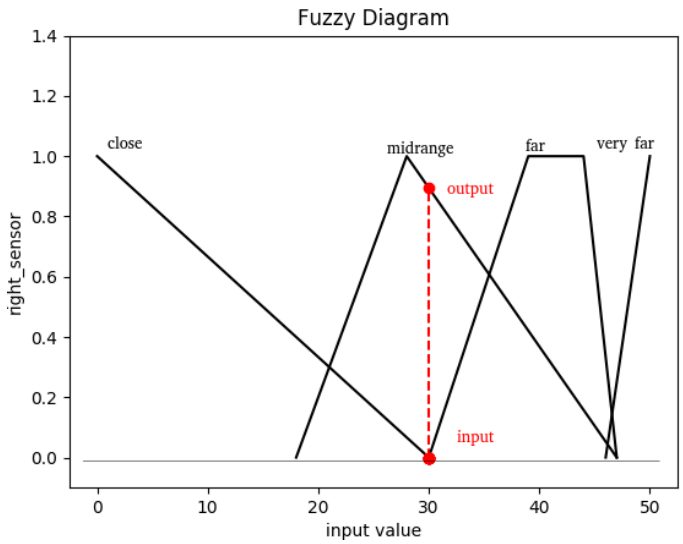


*Fig 2.8: Right sensor input*

There are two output units:

i. velocity
ii. angle

Here, velocity denotes speed, while angle denotes the steering angle.

Velocity consists of four characteristic functions:

i. low
ii. middle
iii. high
iv. very high

Angle consists of five characteristic functions:

i. hard right
ii. right
iii. forward
iv. left
v. hard left

The following is an example of an optimized solution with a smaller number of iterations and a smaller population of input units (left, front, right) = (5, 30, 15):

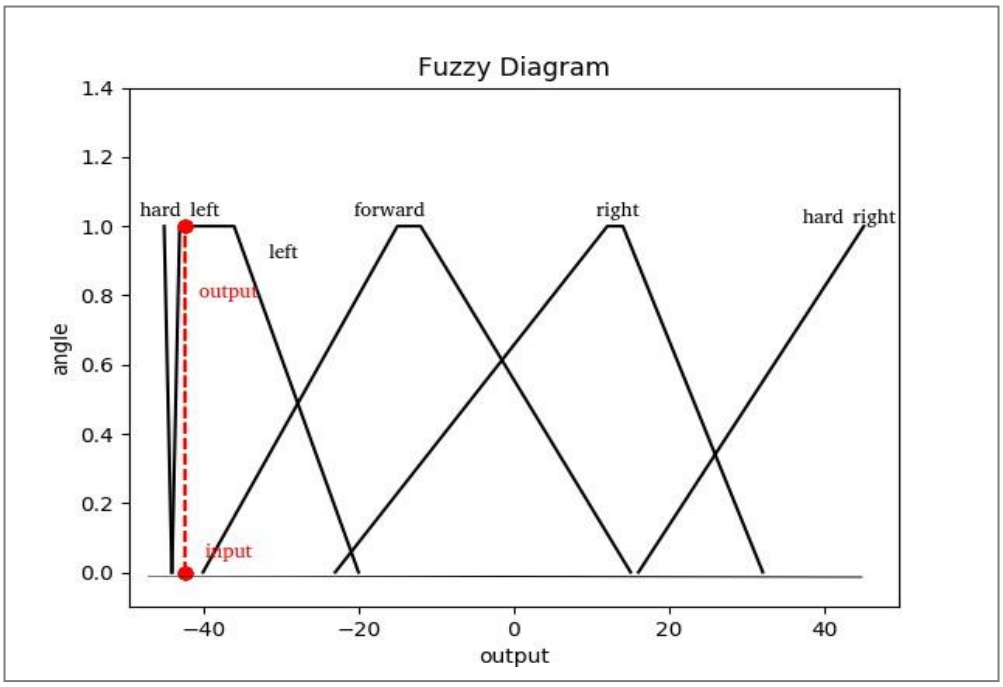


*Fig 2.9: Angle vs Output*

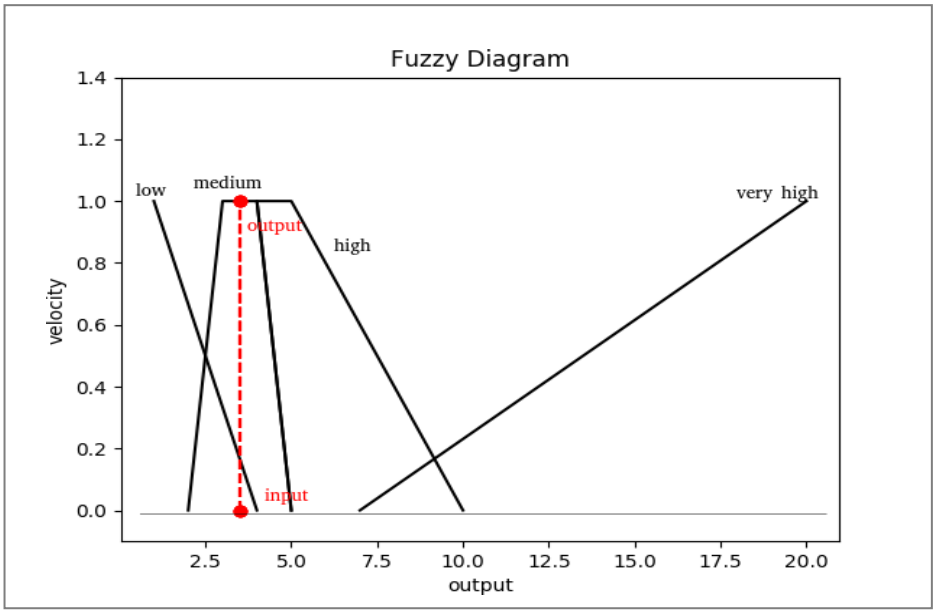


*Fig 3: Velocity vs Output*

The following are the parameters on the basis of which the phase system is generated:

```
ALL_FUZZY_FUNCS = {
    "left_sensor": {
        "name": "left_sensor",
        "left_boundary": 0,
        "right_boundary": 50,
        "mf_names": ["close", "midrange", "far", "very_far"],
        "is_input": True,
    },
    "right_sensor": {
        "name": "right_sensor",
        "left_boundary": 0,
        "right_boundary": 50,
        "mf_names": ["close", "midrange", "far", "very_far"],
        "is_input": True,
    },
    "front_sensor": {
        "name": "front_sensor",
        "left_boundary": 0,
        "right_boundary": 20,
        "mf_names": ["close", "midrange", "far", "very far"],
        "is_input": True,
    },
    "velocity": {
        "name": "velocity",
        "left_boundary": 1,
        "right_boundary": 20,
        "mf_names": ["low", "middle", "high", "very high"],
        "is_input": False,
    },
    "angle": {
        "name": "angle",
        "left_boundary": -45,
        "right_boundary": 45,
        "mf_names": ["hard right", "right", "forward", "left", "hard left"],
        "is_input": False,
    }
}
# Fig. Fuzzy characteristic functions given to the system
```

Phase rules were manually defined by taking some of the existing combinations of rules that seemed to make the most sense.

We already have some information about what the rules should look like (for example, if the right side is close to us, a rule that says to turn sharp to the right obviously doesn't make sense).

The following is a code snippet of the rules used:

```
import  fuzzy
import numpy as np

angle_rules = fuzzy.FuzzyRules(np.array([
    fuzzy.Rule(np.array([left["close"], right["midrange"]]), angle["hard right"]),
    fuzzy.Rule(np.array([left["midrange"], right["close"]]), angle["hard left"]),
    fuzzy.Rule(np.array([left["close"], right["far"]]), angle["hard right"]),
    fuzzy.Rule(np.array([left["far"], right["close"]]), angle["hard left"]),

    fuzzy.Rule(np.array([left["close"], front["close"]]), angle["hard right"]),
    fuzzy.Rule(np.array([right["close"], front["close"]]), angle["hard left"]),
    fuzzy.Rule(np.array([left["close"], front["midrange"]]), angle["right"]),
    fuzzy.Rule(np.array([right["close"], front["midrange"]]), angle["left"]),

    fuzzy.Rule(np.array([left["far"], right["midrange"]]), angle["hard left"]),
    fuzzy.Rule(np.array([left["midrange"], right["far"]]), angle["hard right"]),
    fuzzy.Rule(np.array([left["very far"], right["midrange"]]), angle["hard left"]),
    fuzzy.Rule(np.array([left["midrange"], right["very far"]]), angle["hard right"]),
    fuzzy.Rule(np.array([left["far"], right["close"]]), angle["left"]),
    fuzzy.Rule(np.array([left["close"], right["far"]]), angle["right"]),

    fuzzy.Rule(np.array([left["midrange"], right["midrange"]]), angle["forward"]),
    fuzzy.Rule(np.array([left["close"]]), angle["hard right"]),
    fuzzy.Rule(np.array([right["close"]]), angle["hard left"]),
]))

velocity_rules = fuzzy.FuzzyRules(np.array([
        fuzzy.Rule(np.array([front["close"]]), velocity["low"]),
        fuzzy.Rule(np.array([front["midrange"]]), velocity["middle"]),
        fuzzy.Rule(np.array([front["far"]]), velocity["high"]),
        fuzzy.Rule(np.array([front["very far"]]), velocity["very high"])]))
# Fuzzy Rule declared on general sense such that it seemed to make sense
```

### 2.3.1 GENERATING MEMBERSHIP FUNCTIONS IN INITIAL POPULATION

Each chromosome contains phase systems for angle and speed, as well as fitness.

The most critical part of the algorithm is generating membership functions. Each membership function had to respect the following restrictions:

i. The values for X are within the allowable range
ii. There must be no uncovered part of the X axis within the allowed interval

The affiliation function consisted of 2 sets of coordinates:

i. An X string that takes a value from the interval [left_boundary, right_boundary]
ii. String Y that takes one of the values {0, 1}

Each trapezoidal function is described by 4 pairs of points denoting the critical points of the trapezoid. Similarly, each ramp function is described with 2 pairs of points.

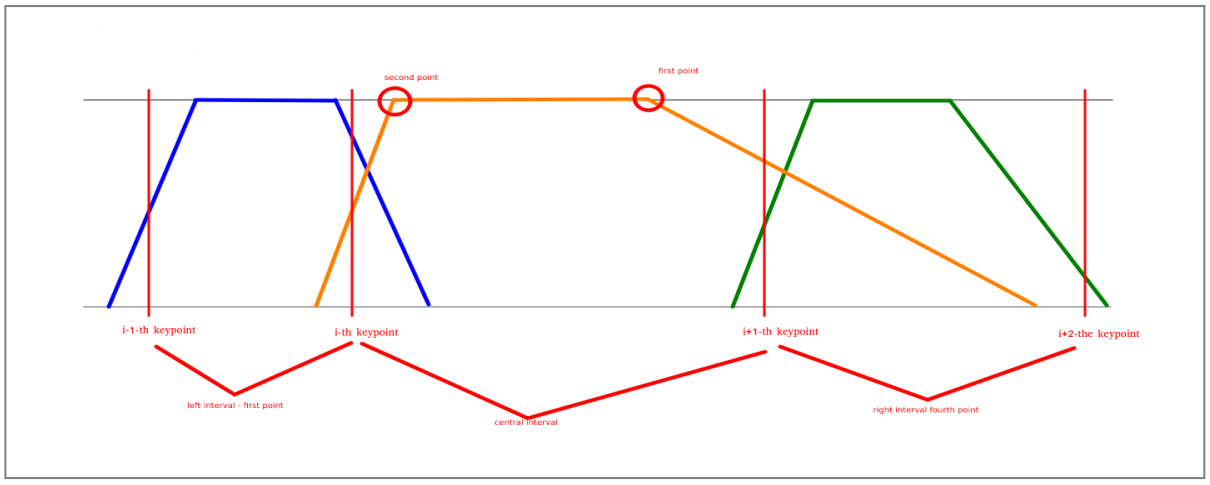


*Fig 3.1: Generating membership functions*

Based on these two sequences X and Y, the initial randomly generated phase system is contrasted.

### 2.3.2 SELECTION AND ELITISM

Tournament selection was used in combination with elitism. Elitism took a certain number of the best individuals in advance, while the others were selected by selection:

TOURNAMENT_SIZE = 10

```
def create_group(population, group_size = TOURNAMENT_SIZE):
    ids = random.sample(range(0, population.size), group_size)
    return population[ids]

def select(population):
    # tournament selection
    group = create_group(population)
    result = group[0]
    for i in range(1, group.size):
        if group[i] < result:
            result = group[i]
    return result
# Code to create group and select individuals
```

### 2.3.3 CROSSBREEDING

Uniform crossing by membership functions was used for each of the output rules of the system phase. The parts of the phase of the system that describe the angle and the parts of the phase of the system that describe the speed are crossed separately:

```
def crossover(p1, p2):
    c1 = copy.deepcopy(p1)
    c2 = copy.deepcopy(p2)
    for i in range(c1.FSAngle.inputs.size):
        fuzzy_output1 = c1.FSAngle.inputs[i]
        fuzzy_output2 = c2.FSAngle.inputs[i]
```

```
            for j in range(fuzzy_output1.inputs.size):
                mf_output1 = fuzzy_output1[j]
                mf_output2 = fuzzy_output2[j]

                r = random.random()
                if r < 0.5:
                    swap(mf_output1, mf_output2)

        for i in range(c1.FSVelocity.inputs.size):
            fuzzy_output1 = c1.FSVelocity.inputs[i]
            fuzzy_output2 = c2.FSVelocity.inputs[i]

            for j in range(fuzzy_output1.inputs.size):
                mf_output1 = fuzzy_output1[j]
                mf_output2 = fuzzy_output2[j]

                r = random.random()
                if r < 0.5:
                    swap(mf_output1, mf_output2)

        return c1, c2
    # Fig. Code to perform crossover on two individual
```

### 2.3.4 MUTATION

Mutation involves generating new values for critical points on an existing function selected for mutation. All the constraints that the function must meet are respected, similar to the generation of the initial population.

### 2.3.5 TRAPEZOIDAL FUNCTION MUTATION

The new function is generated by translating each critical point of the affiliation function along the X axis for a random value from the interval [-MUTATION_SPAN,

MUTATION_SPAN]. Then, the left and right borders were corrected, if they went out of the interval.

### 2.3.6 RAMP FUNCTION MUTATION

The new function was generated so that only the left and right points of the function were shifted by a random value from the interval [-MUTATION_SPAN, MUTATION_SPAN].

```
def mutate(c, mutation_rate = MUTATION_RATE):
    r = random.random()
    if r > mutation_rate:
        return c
    for i in range(c.FSAngle.inputs.size):
        fuzzy_input = c.FSAngle.inputs[i]
        for j in range(fuzzy_input.inputs.size):
            mf_input = fuzzy_input[j]
            for k in range(fuzzy_input.inputs[j].size):
                r = random.random()
                if r < MUTATION_GENOM_RATE:
                    number_of_points = int(mf_input.size)
                    right_boundary = fuzzy_generator.ALL_FUZZY_FUNCS[fuzzy_input.name]["right_boundary"]
                    left_boundary = fuzzy_generator.ALL_FUZZY_FUNCS[fuzzy_input.name]["left_boundary"]
                    shift_value = random.randint(-MUTATION_SPAN, MUTATION_SPAN)
                    if number_of_points == 2 and mf_input.points[0][1] == 1:
                        new_value = mf_input.points[0][0] + shift_value
                        mf_input.points[0][0] = min(right_boundary, new_value)
                    elif number_of_points == 2 and mf_input.points[0][1] == 0:
                        new_value = mf_input.points[1][0] + shift_value
                        mf_input.points[1][0] = max(left_boundary, new_value)
                    else:
                        left = mf_input.points[0][0] + shift_value
                        shift_value = random.randint(-MUTATION_SPAN, MUTATION_SPAN)
                        right = mf_input.points[-1][0] + shift_value
                        if left < left_boundary:
```

```
                    left = left_boundary
                if right > right_boundary:
                    right = right_boundary
                if left > right:
                    swap(left, right_boundary)
                if abs(left - right) <= 2:
                    right += 2
                shift_value = mf_input.points[1][0] + random.randint(-MUTATION_SPAN, MUTATION_SP
                mid_1 += shift_value
                shift_value = mf_input.points[2][0] + random.randint(-MUTATION_SPAN, MUTATION_SP
                mid_2 += shift_value

                if mid_1 > mid_2:
                    mid_1, mid2 = swap(mid_1, mid_2)

                new_xs = [left, mid_1, mid_2, right]

                for t in range(4):
                    mf_input.points[t][0] = new_xs[t]
        return c
    # Fig. Code to perform mutation in one generation
```

### 2.3.7 EVALUATION FUNCTION

Fitness individuals are counted so that the individual would be allowed to ride on the track until one of the following conditions is met:

i. The individual flew off the road
ii. The maximum number of iterations has been reached (the individual never goes out of the way, so he must somehow stop driving)
iii. The unit does not move more than X iterations

As the goal of the individual is to drive as long as possible and as much as possible in the middle of the road, the fitness function is calculated as:

*left_right / iteration + punishment*

*left_right* measures how many individuals walked in the middle of the road, iteration means how long they drove, and punishment is a penalty for those individuals who stopped moving but still did not go out of the way and for individuals who went out of the way.

It remains possible to insert average speed as another parameter for the evaluation function, where the score is better the higher the average score. The smaller the fitness, the better the individual.

```
def evaluate(FSAngle, FSVelocity, road_matrix, memory):
    car = vehicle.Car(constants.CAR_POS_X, constants.CAR_POS_Y, constants.CAR_ANGLE)
    iteration = 0
    past_pos = car.center_position()
    dec = decoder.Decoder(FSAngle, FSVelocity, car)
    dt = TIME_STEP
    total_distance = 0
    punishment = 0
    left_right = 0
    while iteration <= MAX_ITERATIONS:
        car.left_sensor_input, car.front_sensor_input, car.right_sensor_input = get_sensors(car,
road_matrix, memory)
        ds, drot = dec.get_movement_params()
        car.update(dt, ds, drot)
        iteration += 1
        total_distance += ds
        left_right += abs(float(car.left_sensor_input) - float(car.right_sensor_input))
        if iteration % 100 == 0:
            past_x, past_y = past_pos
            curr_x, curr_y = car.center_position()
            if vehicle.distance(past_x, past_y, curr_x, curr_y) < MIN_DISTANCE:
                break
            else:
                past_pos = car.center_position()
        if car.is_idle(iteration) or car.is_collided2(road_matrix):
            punishment = 150
            break
```

```
        return left_right/iteration + punishment
    # Fig. Code to run the evaluation function
```

### 2.3.8 RESULTS

i. Parameters used in training: 5

ii. Population size: 500

iii. Number of generations: 20

iv. Mutation range: 2

v. Probability of chromosome mutation: 0.8

vi. Probability of chromosome crossover 0.2

vii. Probability of mutation of membership function: 0.1

viii. Maximum number of iterations: 500

ix. Penalty for getting off the road: 150

# CHAPTER 3

# SIMULATIONS AND RESULTS

The above concepts were implemented and the vehicle was able to move along the path, maintaining a constant distance from the left boundary and right boundary.

## 3.1 SIMULATION ON A CONVEX PATH

The image shown below is a snapshot taken from the simulation of vehicle steering around the closed convex path.

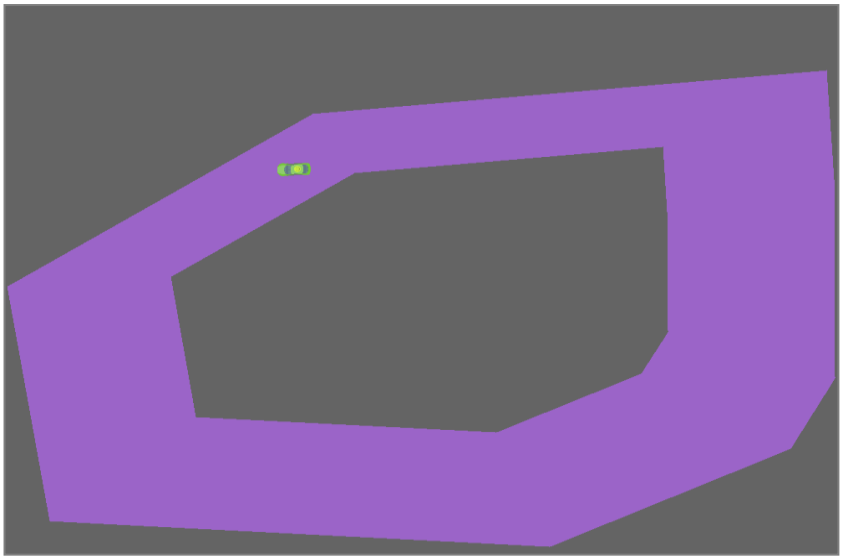

*Fig 3.2: Simulation on a convex path*

### 3.2 SIMULATION ON A SINUSOIDAL PATH

The image shown below is a snapshot taken from the simulation of vehicle steering around a sinusoidal path.

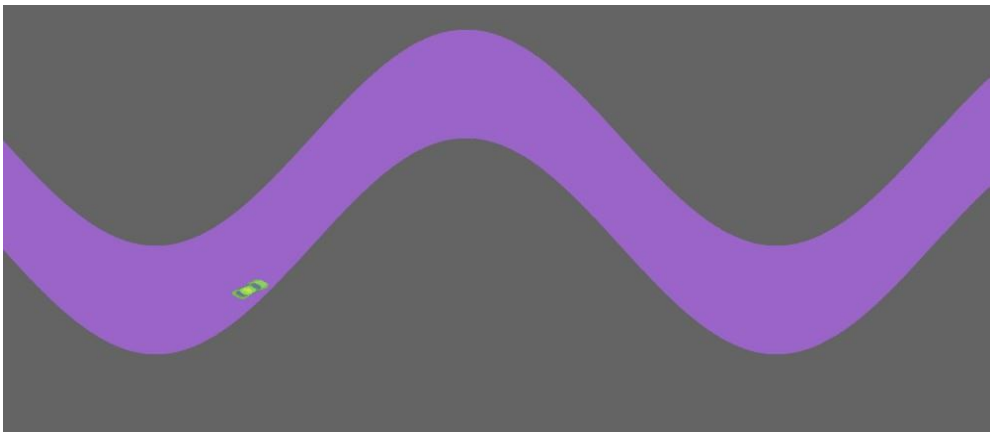

*Fig 3.3: Simulation on a sinusoidal path*

The fuzzy phase controller was further optimized using genetic algorithms. After the algorithm runs for all generations or the evaluation condition is met the simulation was run again. This time we saw an optimized version of the phase controller.

Calculating the distance is an expensive operation. We can round the position of the vehicle and the angle of rotation of the vehicle to an integer. In that case, we have the final number of states in which the vehicle can come. In the map we can store the key that represents the arranged triplet (Xo, Yo, a), i.e., the position of the vehicle and the angle. This key is mapped to the ordered triple (left_sensor_input, front_sensor_input, right_sensor_input). With this optimization, we eliminate double calculations which has resulted in a faster response time of the phase controller.

# CHAPTER 4

# CONCLUSION

As seen in the preceding sections, the aim of this study, which was to optimize the fuzzy phase controller to control the car, was met. The comparative study shows that there has been a significant improvement in the control of vehicles around the track. The optimized phase controller maintains almost a constant distance from both the left and right boundary.

The success of any model depends on the type and quality of the dataset on which it has been trained. The success of automated vehicles in the United States is because of the good and clean dataset. If we were to store the data from the sensors, we can acquire a significant dataset for training the vehicle.

The phase controller which was optimized using genetic algorithms can also be optimized using different heuristic algorithms. A comparative study can be made and a suitable method for optimizing the controller.

# REFERENCES

[1] Laura Wood, “Global Autonomous Cars Market (2020 to 2030)”, Available: https://www.globenewswire.com/, December 28, 2020 (Accessed December 7, 2021)

[2] ThalesGroup, “Benefits of Autonomous Cars” Available: https://www.thalesgroup.com/en, July 2017.

[3] M. Trabia, L. Z. Shi, and N. E. Hodge, “A fuzzy logic controller for autonomous wheeled vehicles,”2006.

[4] V. Ivanov, “A review of fuzzy methods in automotive engineering applications”, August, 2018

[5] V Shinde, R Thorat, T Agarkar, “Automatic Car Driving System Using Fuzzy Logic”, current issue: Vol. 5, Issue 3 , March 2018

[6] Vineet Kumar, K. P. S. Rana, Jitendra Kumar, Puneet Mishra, Sreejith S. Nair. "A robust fractional order fuzzy P + fuzzy I + fuzzy D controller for nonlinear and uncertain system" , International Journal of Automation and Computing, 2016

[7] Danilo Pelusi, “Optimization of a fuzzy logic controller using genetic algorithms’’ , August 2019.

[8] S. K. Alonso, “Mamdani’s fuzzy inference method”,[Online] 2020 Available: http://www.dma.fi.upm.es/recursos/aplicaciones/logica_borrosa/web/fuzzy_inferencia/mamdani2_en.htm

## BRIEF CURRICULUM VITAE

Name: Mehak Malhotra

Roll No. 2018UIC3016

Contact: 9069069400

Email ID: mehakm.ic18@nsut.ac.in

Name: Mudit Mahajan

Roll No. 2018UIC3018

Contact: 7424961361

Email ID: muditm.ic18@nsut.ac.in

Name: Raghav Kaushal

Roll No. 2018UIC3020

Contact: 9315617991

Email ID: raghavk.ic18@nsut.ac.in

Name: Shloka Gupta

Roll No.: 2018UIC3150

Contact: 9971649326

Email ID: shlokag.ic18@nsut.ac.in