A NOVEL APPROACH USING FUZZY LOGIC TO DETECT TRAFFIC CONTROL SYSTEMS

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the
requirement for the award of the
Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (CORE)

by

GUDI VARAPRASAD (19BCE7048)

Under the Guidance of

DR. SOMYA RANJAN SAHOO

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
VIT-AP UNIVERSITY AMARAVATI- 522237

December 2022

CERTIFICATE

This is to certify that the Capstone Project work titled "**A NOVEL APPROACH USING FUZZY LOGIC TO DETECT TRAFFIC CONTROL SYSTEMS**" that is being submitted by **GUDI VARAPRASAD (19BCE7048)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of Bonafied work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Somya Ranjan Sahoo
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR **DEAN**

B. Tech. CSE School Of Computer Science and Engineering

ACKNOWLEDGEMENTS

We would like to express our gratitude to **Dr. G. Viswanathan,** Chancellor, **Dr. Sekar Viswanathan,** Vice president, **Dr. S. V. Kota Reddy,** Vice Chancellor and Dean **Dr. Sudha S V**, School of Computer Science and Engineering, for providing with an environment to work during the tenure of the course.

In jubilant mood we express ingeniously our whole-hearted thanks to **Dr. Saroj Kumar Panigrahy** Program chair and Professor, School of Computer Science and Engineering and our pleasure to express with deep sense of gratitude to **Dr. Somya Ranjan Sahoo**, Assistant Professor, School of Computer Science and Engineering, VIT-AP, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in our endeavor.

It is indeed a pleasure to thank all faculty members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on us with zeal, which prompted the acquirement of the requisite knowledge to finalize our course study successfully. We would like to thank our respective parents for their support.

At last, but not least, we express our gratitude and appreciation to all those who have helped us directly or indirectly toward the successful completion of this project.

Place: Amaravati
Date: 21-12-2022

GUDI VARAPRASAD

ABSTRACT

This project report proposes a fuzzy traffic control system based on fuzzy logic theory. This suggested system may effectively manage traffic junctions' congestion by regulating the time length of the green phase interval. Fuzzy logic may be developed on top of specialized knowledge and integrated with traditional control approaches; however, rather than replacing traditional control methods, fuzzy control systems simplify the implementation process in many circumstances. The primary objectives are to improve traffic safety at the junction, maximize capacity at the intersection, and minimize delays. In order to govern signal timings, a fuzzy logic controller may utilize linguistic and imprecise traffic data. The growing number of automobiles and road shortages cause traffic congestion in many places, affecting efficiency, productivity, and energy losses. The traffic signal controller operating approach at road junctions is a key contributor to this congestion.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Most major cities across the globe have experienced traffic congestion. This traffic congestion may have an impact on the economy and slow down development. It also diminishes output, raises costs, and interferes with people's everyday lives. A large city's traffic problems may be caused by a variety of factors. Among these include an increase in the number of cars, a lack of adequate roads and highways, and the conventional traffic signal system. All of these elements may contribute to traffic congestion at the junction, but the classic traffic signal system is one of the most influential. Traffic lights are ubiquitous characteristics of metropolitan areas worldwide, limiting the quantity of automobiles. The primary objectives are to improve traffic safety at the junction, maximize capacity at the intersection, and minimize delays. In order to govern signal timings, a fuzzy logic controller may utilize linguistic and imprecise traffic data.

Consider the cities of Bangalore and Mumbai. According to a survey analyzing the traffic condition in 416 cities spanning 57 nations, Bangalore has the trounce traffic flow in the world, while Mumbai is near along in fourth place. A drive in Bangalore at peak traffic takes 71% lengthier. It is 65% slower in Mumbai [1].

There are now three conventional approaches that are utilized for the management of traffic, and they are as follows:

1. Conventional traffic signals with stable trackers: These are controlled by timers that are permanently set. A persistent number is placed into the timer. Depending on the timer setting, the lights automatically alternate between red and green in the junctions.

2. Electronic Sensors: The installation of proximity sensors or sensing devices or some loop detectors on the road is an additional innovative technique. This sensor provides information on the road's traffic. The traffic lights are regulated in accordance with the sensor information.

3. Manual Operation: Like the name implies, controlling traffic by hand is necessary. Traffic cops are assigned to a certain region in order to regulate traffic. To manage traffic, traffic police carry a placard, a sign light, as well as whistles.

These traditional approaches have a number of disadvantages. The manual control system demands a significant amount of labour. We cannot have traffic cops manually manage traffic in all sections of a city due to insufficient manpower. Therefore, a better traffic control system is required. Static traffic control employs a traffic signal with a predetermined timing for each phase, which does not react to the actual traffic on the road. When utilizing electronic sensors, such as sensor devices or loop detection systems, accuracy and coverage are frequently in conflict due to the fact that the collection of rising data typically relies on complex and costly technologies, and a limited budget will consequently reduce the number of amenities. Moreover, owing to the restricted operational range of most sensors, a network of services often needs a large number of sensors for comprehensive coverage.

1.1 Objectives

The following are the objectives of this project:

- Develop Fuzzy Based approach to solve traffic congestion problems.
- Have multiple features such as traffic speed control, spawn (vertical lane) rate, automatic traffic light control, random vehicle movements in multiple directions following traffic rules, etc.
- Design and develop a user-friendly GUI for traffic optimization using Fuzzy Algorithm.

The following are the constraints and hypothesis of this project:

- The Python Community is not accessible for debugging source code.
- Fuzzification rules and limitations are prior established.
- Because each vehicle's speed is preset, the problem can only be partially addressed and partially optimized, which is not how real-world situations work.
- Handling erroneous data and the intuitive inference made by human cognition.
- We assume that the vehicle moves in forward direction and not any other ways.
- Stick to the project timeline and keep checking the deadline.

1.2 Background and Literature Survey

Video monitoring and surveillance systems have been widely employed in traffic management in recent years for surveillance, road design, and delivering real-time updates and information to passengers. Artificial Intelligence based surveillance systems can also be employed to estimate traffic volume and count vehicles, which can subsequently be implemented to manage traffic signal timings to streamline traffic efficiency congestion. Our suggested system attempts to provide a fuzzy system based traffic signal controller that can adapt to the present traffic environment. Our suggested solution can manage the traffic congestion at the junction and regulate the time length of the green stage interval effectively. It can determine the green light time length based on the number of cars waiting at the red light stage and the frequency of vehicles during the green signal. The Min-Max inference process and the Centroid Defuzzification approach will be used by the system to generate the crisp value for the green light period time as the outcome. This helps to optimize green signal periods, and traffic is cleared at a far quicker pace than a static system, eliminating undesired delays, congestion, and waiting time, and thereby lowering fuel consumption and pollution.

This report will begin by providing a quick overview of the fuzzy traffic controller before delving into four case studies linked to the fuzzy logic-based traffic management system. Each of these studies shows a unique approach to developing a fuzzy traffic control system. Our suggested fuzzy system is emulated using a Python library with a graphical user interface, which will include the return values of all the fuzzy sets and membership algorithms used in the code for developing our fuzzy traffic management system.

Fuzzy logic allows for the simple and effective solution of complicated and non-linear issues. Fuzzy logic has several applications, including making decisions, control theory, pattern matching, image analysis, and health care.

There are two aspects to system of traffic lights. The first component is the traffic light, and the second component is the controller module. At congested crossroads, traffic lights are proposed to control flow of traffic. A traffic light signal, often known as a stop light, is a tell - tale sign used to manage the passage of cars and people at road junctions, pedestrian crossings, and other sites. Furthermore, traffic lights typically feature three major lights: a red light that means "stop," a green light that means "proceed," and a yellow light that means "stop if feasible." The next section discusses two typical traffic-control strategies: real-time (RT) traffic light algorithms and fixed-time (FT) traffic light control mechanisms. The fundamental idea behind a fuzzy traffic signal controller is to construct a control approach based on human inference. The traffic controller incorporates fuzzy logic theory to give an adaptive green interval reply based on shifting traffic load inputs. To solve the inefficiencies of traditional traffic controllers that have a constant cycle time depending on variable traffic load, a fuzzy logic control method is developed.

Below is a list of references providing an overview of prior study on this issue:
The author from [2] describes a traffic light governed by fuzzy logic that can respond to the present traffic conditions. For major and auxiliary lanes, this system employs two fuzzy controllers with input variables and one outcome. The simulation using MATLAB and VISSIM enhanced traffic conditions at few traffic densities.

The author from [3] discusses numerous traffic signal control system strategies. This research notes that each approach has a basic architecture: selecting input data, acquiring traffic patterns from input data, processing it, determining density, and updating parameters.

Fig. 1 With a growing population and number of vehicles, urban traffic congestion is becoming one of the most significant problems.

For successful implementation of the project, I used Python for Backend, Pygame to design Front end Interface (GUI). Some of the other Python requirements include Numpy, scikit_fuzzy, Scipy libraries, etc.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Section 1.3.1 consists of the Working of Fuzzy Logic Control, Explanation of Fuzzy Logic implementation for Traffic Lights, Fuzzy Rules, Python Libraries.

- Section 1.3.2 contains Project Methodology, GUI Implementation, and Fuzzy Implementation.
- Section 1.3.3 considers the results obtained after the project had been executed.
- Section 1.3.4 summarizes and concludes the report.
- Section 1.3.5 accommodates of codes.

SECTION 1.3.1

A NOVEL APPROACH USING FUZZY LOGIC TO DETECT TRAFFIC CONTROL SYSTEMS

This Chapter describes the proposed software, methodology, software configurations.

However, due to the significant rise in traffic (both public and private), there are several traffic congestions issues today. The traffic controller is one of the most important determinants of traffic flow. Existing traffic management systems are largely static, meaning that they do not adapt to the flow of traffic. A Fuzzy Logic-based Traffic Lights Control System is one of the possible alternative options. A reasoning strategy that matches human reasoning. Fuzzy logic imitates the manner in which people make decisions, which incorporates all intermediate possibilities between the digital values YES and NO.

Our suggested system attempts to provide a fuzzy system based traffic signal controller that can adapt to the present traffic environment. Our suggested solution can manage the traffic congestion at the junction. Fuzzy logic is superior because fuzzy systems may be developed quickly and are readily understood.

It regulates the time length of the green stage interval effectively. It can determine the green light time length based on the number of cars waiting at the red light stage and the frequency of vehicles during the green signal. The Min-Max inference process and the Centroid Defuzzification approach will be used by the system to generate the crisp value for the green light period time as the outcome. This helps to optimize green signal periods, and traffic is cleared at a far quicker pace than a static system, eliminating undesired delays, congestion, and waiting time, and thereby lowering fuel consumption and pollution.

1.3.1.a Working of Fuzzy Logic Control

A fuzzy control system includes the following elements:

a) Rule Base: This includes the fuzzy logic system's rules and membership functions that govern or control decision-making. It also includes the IF-THEN conditions used for conditional programming and system control.

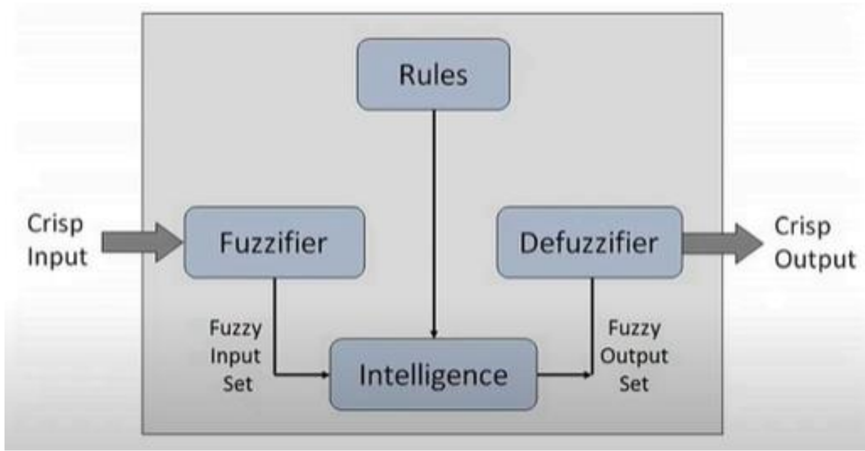

Fig. 2 Fuzzy logic architecture

b) Fuzzifier: This component converts raw inputs to fuzzy sets. The fuzzy sets are processed further by the control system.

c) Inference Engine: This is a device that determines the optimal rules for a certain input. These rules are then applied to the input data to provide a fuzzy output.

d) Defuzzifier: This component converts fuzzy sets to an explicit output (in the form of crisp inputs). Defuzzification is the ultimate phase of a system using fuzzy logic.

1.3.1.b Fuzzy Logic implementation for Traffic Lights (Methodology)

Using certain mathematical models, fuzzy technology may translate the human thought process into an algorithm for controlling the traffic flow system. In this scenario, sensors are assumed to exist at the intersection to be more realistic. Sensors are used to track the quantity of cars. In this case, our system applies fuzzy logic. Fuzzy if-then rules may be used to implement genuine regulations that are comparable to the way that traffic officers would regulate traffic signal lights.

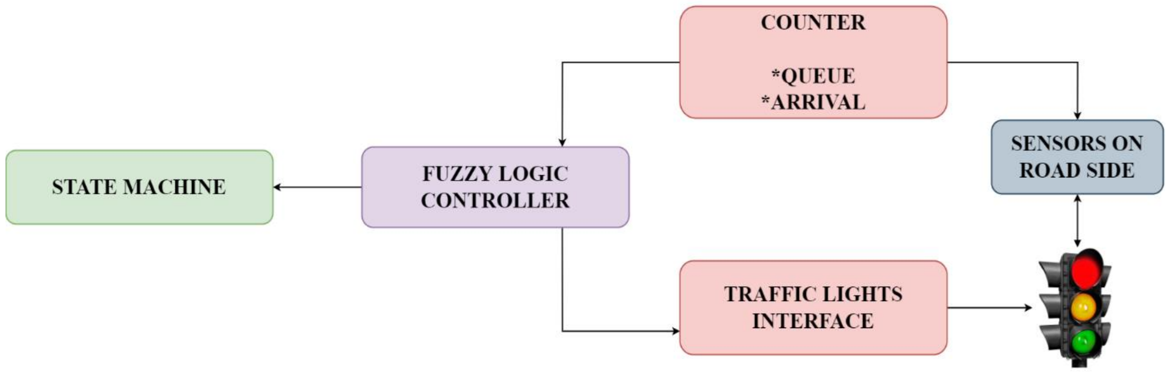

Fig. 3 Working of Fuzzy Logic for Traffic Control System

The controllers of traffic signals are required to alter the cycle period of the green light signal based on the number of arriving cars in order to optimize traffic flow and limit the average waiting time. It can regulate the lights based on the number of vehicles waiting at a red light and correspondingly adjust the timing to release or keep the vehicles. It replicates a crossroads where east-to-west and north-to-south traffic flows. There are sensors on both sides of the road. In a real-world application, these sensors would consist of embedded electromagnetic sensors.

1.3.1.c Fuzzy Rules

A fuzzy rule is an if-then (conditional) expression. IF THEN statements provide the format for fuzzy rules. IF linguistic variables are included, fuzzy sets generate linguistic values. In fuzzy logic, these are the rules of inference that determine the value of an output variable depending on the values of input variables. IF-THEN formulations of the usual expression "IF A THEN B", where A and B are (propositions of) collections including semantic variables. Below are the established Logical rules for the proposed novelty:

- Rule 1: If Arrival is Few Then Extension is Zero.
- Rule 2: If Arrival is Small and Queue is (Few or Small) then Extension is Short.
- Rule 3: If Arrival is Small and Queue is (Medium or Many) then Extension is Zero.
- Rule 4: If Arrival is Medium and Queue is (Few or Small) then Extension is Medium.
- Rule 5: If Arrival is Medium and Queue is (Medium or Many) then Extension is Short.

- Rule 6: If Arrival is Many and Queue is Few then Extension is Long.
- Rule 7: If Arrival is Many and Queue is (Small or Medium) then Extension is Medium.
- Rule 8: If Arrival is Few and Queue is Many then Extension is Short.

1.3.1.d Libraries and Other Software Details

A simulation was constructed from ground using Pygame to imitate real-life traffic. It aids in visualizing the system and contrasting it with the current static system. It has a four-way junction with four traffic lights. Each signal has a timer on top of it, which indicates the time left for the signal to go from green to yellow, yellow to red, or red to green. The number of vehicles that have gone through the intersection or traffic junction is also shown beside each signal. Vehicles such as automobiles, motorcycles, buses, trucks, and rickshaws stream in from all directions. In order to make the simulation more realistic, some of the cars in the rightmost lane turn to cross the crossing. When a vehicle is produced, random numbers are also used to determine whether the vehicle will turn or not. It also features a timer that indicates the time passed since the commencement of the simulation. Various libraries used in this project are:

a) NumPy

NumPy is a library for the Python programming language that extends the language's support for large, matrices and multidimensional arrays and a wide variety of higher level mathematical functions for manipulating them. NumPy is utilized in this project to save needed configuration parameters as array values and reuse them in other Python files. The stored values are scrutinized using fuzzy logic and fuzzy conditions.

b) Pygame

Pygame is a cross-platform collection of Python modules used for game development. It is a collection of graphics and audio libraries developed for use with the Python programming language. Pygame is appropriate for developing client-side applications that may be packaged as a standalone executable. It provides extra functionality on top of the SDL library, allowing you to quickly produce fully functional graphics. In this project, the Front end Interface (GUI) is designed using Pygame to provide a realistic setting of the present traffic system.

c) Skfuzzy

Scikit-Fuzzy or Skfuzzy is a set of fuzzy logic algorithms built in the Python programming language for use with the SciPy Stack. This package of the Fuzzy Logic Toolbox includes a number of helpful tools and functions for computation and projects using fuzzy logic, often known as grey logic. The backend foundation of the whole project is this library. In designing the code for this project, several techniques like as membership, defuzzify, fuzzymath, intervals, image, and filters are used, among others. A comprehensive list of submodules and functionalities facilitates the development of this project and its integration with other emerging technologies.

d) **Other common imports and SciPy libraries:**

i. **glob -** is used similarly to find, locate, and search for all of the files that are present in a system.

ii. **random -** function generates random floating numbers in the range[0.1, 1.0).

iii. **os -** provides functions for creating and removing a directory (folder), fetching its contents, changing, and identifying the current directory, etc.

iv. **enum -** a class in python for creating enumerations, which are a set of symbolic names (members) bound to unique, constant values.

v. **time -** returns the number of seconds passed since epoch. Here it is helpful to compute traffic signal durations.

vi. **threading -** includes a simple-to-implement locking mechanism that allows you to synchronize threads.

vii. **sys -** provides procedures and variables to manipulate Python's runtime environment and interact with the interpreter.

Scientific and technical computers may benefit from the SciPy Python library, which is available for no cost and open source. The scientific and engineering community may use SciPy's optimization, integration, linear algebra, special functions, interpolation, Fast Fourier Transform (FFT), Ordinary differential equation (ODE) solver modules, and signal and image processing.

SECTION 1.3.2

IMPLEMENTATION

1.3.2.a GUI implementation (Frontend)

```python
class TrafficController:
    def __init__(self, surface):
        self.surface = surface
        self.screen_height = Config['simulator']['screen_height']
        self.screen_width = Config['simulator']['screen_width']
        self.traffic_light_body_height = Config['traffic_light']['body_height']
        self.traffic_light_body_width = Config['traffic_light']['body_width']
        self.traffic_light_distance_from_center = Config['traffic_light']['distance_from_center']

        self.traffic_lights = {}

        x = self.screen_width / 2 - self.traffic_light_distance_from_center[0] - self.traffic_light_body_width
        y = self.screen_height / 2 - self.traffic_light_distance_from_center[1] - self.traffic_light_body_height
        self.create_traffic_light(x, y, Lane.left_to_right)

        x = self.screen_width / 2 + self.traffic_light_distance_from_center[0]
        y = self.screen_height / 2 + self.traffic_light_distance_from_center[1]
        self.create_traffic_light(x, y, Lane.right_to_left)

        y = self.screen_width / 2 - self.traffic_light_distance_from_center[0] - self.traffic_light_body_width
        x = self.screen_height / 2 + self.traffic_light_distance_from_center[1]
        self.create_traffic_light(x, y, Lane.top_to_bottom)

        y = self.screen_width / 2 + self.traffic_light_distance_from_center[0]
        x = self.screen_height / 2 - self.traffic_light_distance_from_center[1] - self.traffic_light_body_height
        self.create_traffic_light(x, y, Lane.bottom_to_top)

        self.fuzzy = Fuzzy()

        self.latest_green_light_extension = 0

    def get_traffic_lights(self, double_lane: DoubleLane):
        if double_lane == DoubleLane.Horizontal:
            return [
                self.traffic_lights[Lane.left_to_right],
                self.traffic_lights[Lane.right_to_left]
            ]
        elif double_lane == DoubleLane.Vertical:
            return [
                self.traffic_lights[Lane.bottom_to_top],
                self.traffic_lights[Lane.top_to_bottom]
            ]
        return None
```

1.3.2.b Fuzzy Implementation (Backend)

More Source code can be found here

```
def calculate_fuzzy_score(self, moving_averages):
    traffic_state = self.traffic_ctrl.get_current_active_lane()
    if self.is_extended :
        ext_count = 1
    else:
        ext_count =0

    if traffic_state == DoubleLane.Vertical:
        return self.traffic_ctrl.calculate_fuzzy_score(moving_averages[Lane.top_to_bottom], moving_averages[Lane.left_to_right], ext_count)
    elif traffic_state == DoubleLane.Horizontal:
        return self.traffic_ctrl.calculate_fuzzy_score(moving_averages[Lane.left_to_right], moving_averages[Lane.top_to_bottom], ext_count)
```

Fuzzy.py ×

src > Fuzzy.py > Fuzzy > __init__

```
import numpy as np
import skfuzzy as fuzz
from src.Config import Config

class Fuzzy:

    def __init__(self):
        #to set the images
        setting = Config['fuzzy']['range']
        self.x_behind_red_light = setting['behind_red_light']
        self.x_arriving_green_light = setting['arriving_green_light']
        self.x_extension = setting['extension']
        #set membership functions for A
        setting = Config['fuzzy']['membership_function']['arriving_green_light']
        self.arriving_green_light_few = fuzz.trimf(self.x_arriving_green_light, setting['few'])
        self.arriving_green_light_small = fuzz.trimf(self.x_arriving_green_light, setting['small'])
        self.arriving_green_light_medium = fuzz.trimf(self.x_arriving_green_light, setting['medium'])
        self.arriving_green_light_many = fuzz.trimf(self.x_arriving_green_light, setting['many'])

        setting = Config['fuzzy']['membership_function']['behind_red_light']
        self.behind_red_light_few = fuzz.trimf(self.x_behind_red_light, setting['few'])
        self.behind_red_light_small = fuzz.trimf(self.x_behind_red_light, setting['small'])
        self.behind_red_light_medium = fuzz.trimf(self.x_behind_red_light, setting['medium'])
        self.behind_red_light_many = fuzz.trimf(self.x_behind_red_light, setting['many'])

        setting = Config['fuzzy']['membership_function']['extension']
        self.extension_zero = fuzz.trimf(self.x_extension, setting['zero'])
        self.extension_short = fuzz.trimf(self.x_extension, setting['short'])
        self.extension_medium = fuzz.trimf(self.x_extension, setting['medium'])
        self.extension_long = fuzz.trimf(self.x_extension, setting['long'])

    def get_extension(self, arriving_green_light_car, behind_red_light_car, extension_count):
        behind_red_light_level_few = fuzz.interp_membership(self.x_behind_red_light, self.behind_red_light_few, behind_red_light_car)
        behind_red_light_level_small = fuzz.interp_membership(self.x_behind_red_light, self.behind_red_light_small, behind_red_light_car)
        behind_red_light_level_medium = fuzz.interp_membership(self.x_behind_red_light, self.behind_red_light_medium, behind_red_light_car)
        behind_red_light_level_many = fuzz.interp_membership(self.x_behind_red_light, self.behind_red_light_many, behind_red_light_car)

        arriving_green_light_level_few = fuzz.interp_membership(self.x_arriving_green_light, self.arriving_green_light_few, arriving_green_light_car)
        arriving_green_light_level_small = fuzz.interp_membership(self.x_arriving_green_light, self.arriving_green_light_small, arriving_green_light_car)
        arriving_green_light_level_medium = fuzz.interp_membership(self.x_arriving_green_light, self.arriving_green_light_medium, arriving_green_light_car)
        arriving_green_light_level_many = fuzz.interp_membership(self.x_arriving_green_light, self.arriving_green_light_many, arriving_green_light_car)
```

SECTION 1.3.3

RESULTS AND DISCUSSIONS

The result of the developed Fuzzy system is as follows:

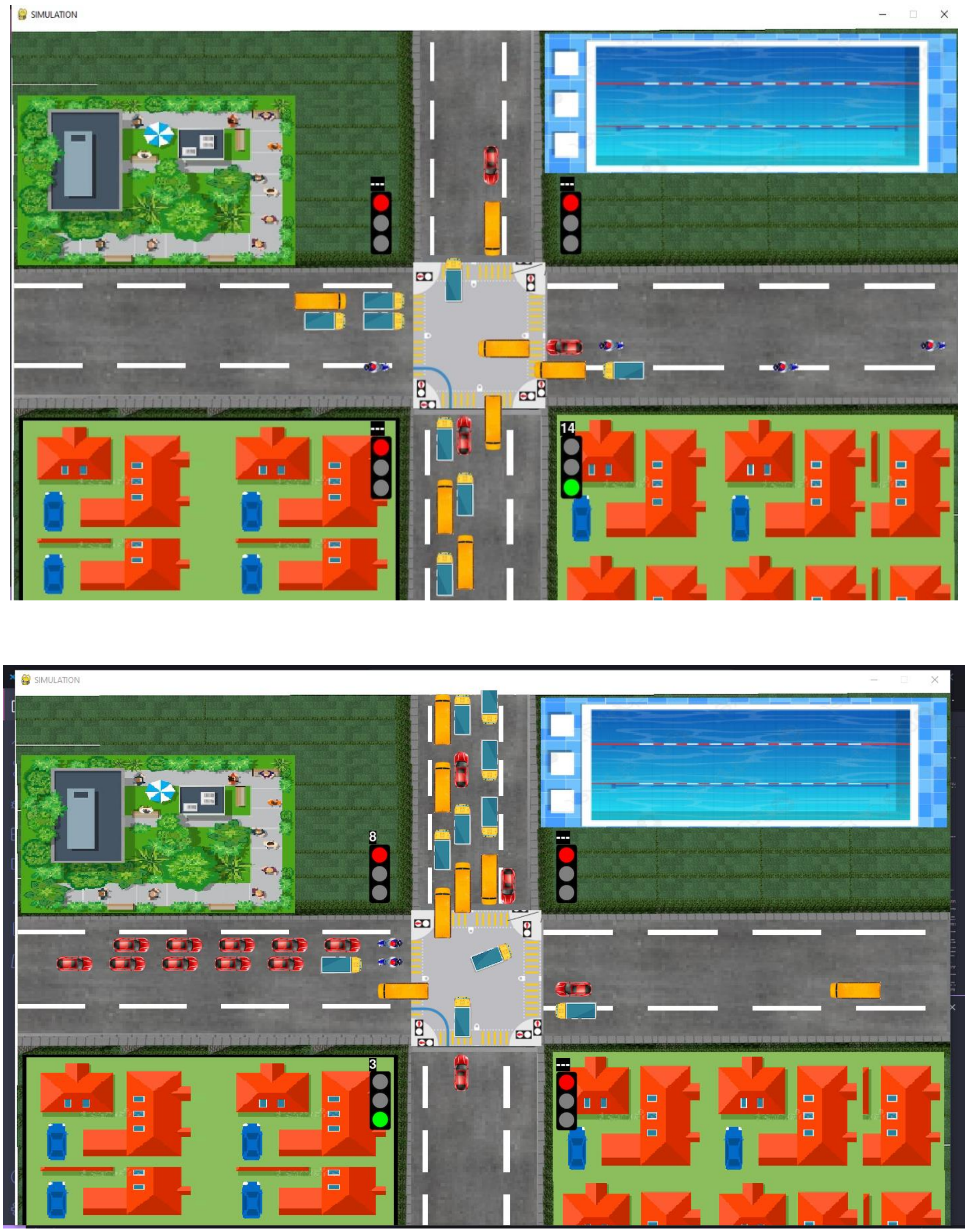

Fig. 4 Working of Fuzzy Logic to Detect Traffic Control Systems

Fig. 5 Working of Fuzzy Logic to Detect Traffic Control Systems

Fuzzy memberships are constructed based on the time. Memberships of F, S, M, M' (Few , Small , Medium , Many) and Z, S, M, L (Zero , Short ,Medium , Long) are arranged; we have the membership functions for related linguistic variables. For our implementation, we have designed triangular shaped membership functions. We employ the flow length and its waiting time, and these values are used to estimate how much the time delay should be.

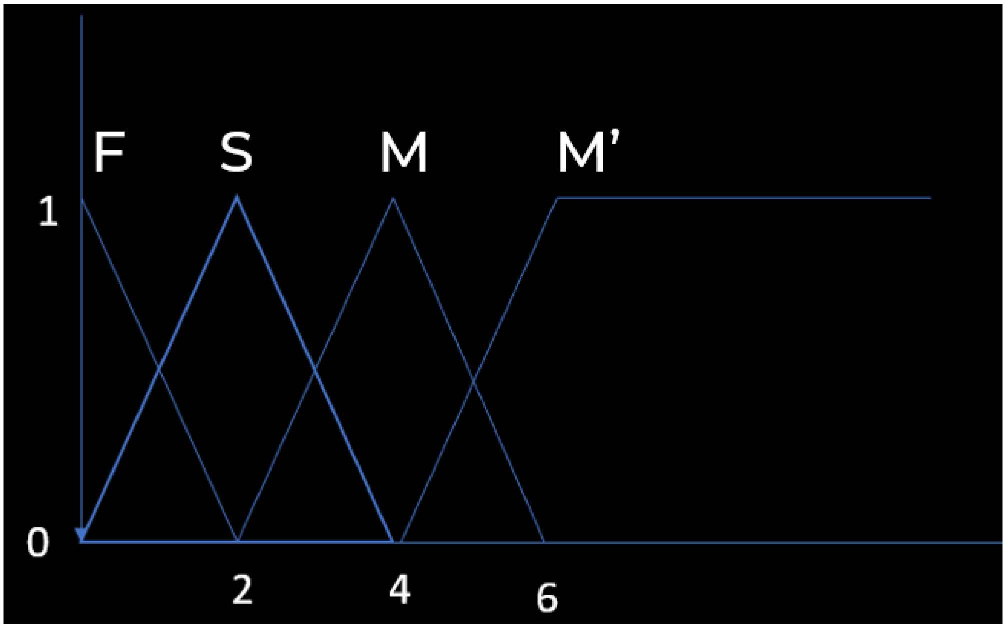

Fig. 6 Input Fuzzy Variable-1 (Arrival { Few , Small , Medium , Many })

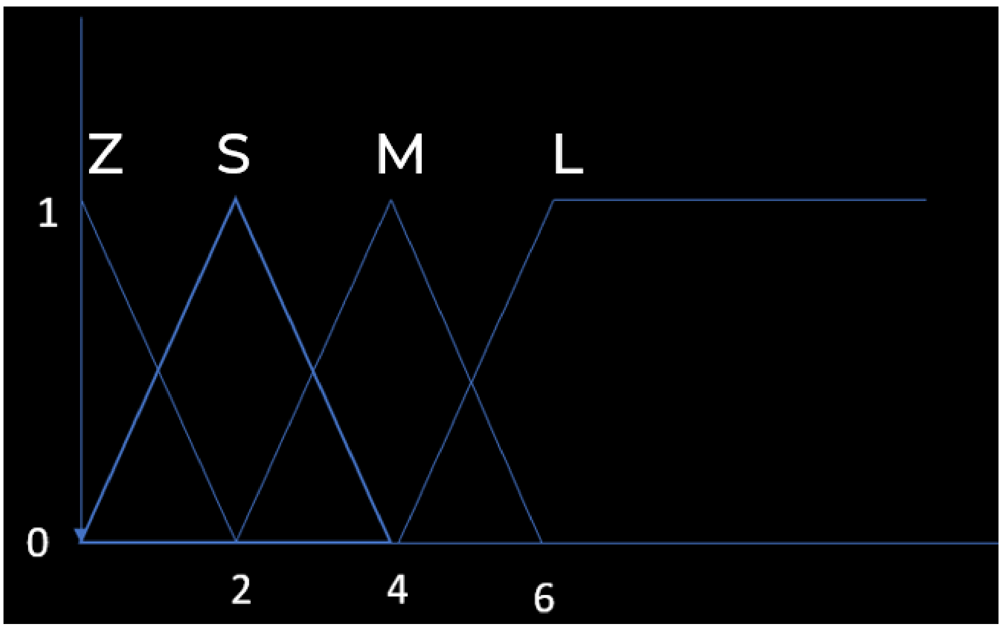

Fig. 7 Input Fuzzy Variable-2 (Queue {Zero , Short ,Medium , Long })

A program has been implemented in Python in order to simulate fuzzy logic controller for testing the controller's effectiveness. The system can contribute to mitigate the traffic congestion problem to maximum possible extent.

SECTION 1.3.4

CONCLUSION AND FUTURE WORK

From the foregoing project, we can infer that fuzzy logic plays a significant role in addressing large problems such as traffic jams. Fuzzy logic is superior because fuzzy systems may be developed quickly and are readily understood. We can resolve difficult challenges. The logic is solid, straightforward, and modifiable to our specifications. It can quickly make judgments since its logic system mirrors human thinking, allowing it to tackle even the most complicated situations with ease. However, fuzzy logic could get muddled since it requires frequent logic updates. A fuzzy logic controller relies entirely on human knowledge, reasoning, and experience. These controllers are incapable of identifying deep learning or machine learning or artificial neural networks. Due to erroneous data, the system's effectiveness is at risk.

Based on the completed project, there are still questions and concerns that may be addressed in next research. The difficulties include Controlling a vast number of intersections concurrently to maintain the uninterrupted flow of traffic, particularly during traffic congestion. Using contemporary UAV-based data collecting and quick mapping techniques to reduce traffic congestion.

The insertion of criteria characterizing emergency scenarios in the event of a traffic collision. The implementation of massive sensor networks using non-traditional variables (drastically changing weather, driver mood, etc.) Optimizing neural networks for traffic description and combining them with fuzzy logic. The usage of driverless robotic vehicles.

SECTION 1.3.5

APPENDIX

[1]. Source code - https://github.com/GudiVaraprasad/traffic-control-fuzzy
[2]. Project Demo - https://www.youtube.com/channel/UCR2gr-sBjqnH0gW-Ca6i7Hw
[3]. Other files - https://github.com/GudiVaraprasad/traffic-control-fuzzy/Files

REFERENCES

[1]. TomTom.com, 'Tom Tom World Traffic Index', 2019. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/ranking/

[2]. A.A.Zaid, Y. Suhweil and M. A. Yaman, "Smart controlling for traffic light time," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, 2017, pp. 1-5, doi: 10.1109/AEECT.2017.8257768.

[3]. Ms. Saili Shinde, Prof. Sheetal Jagtap, Vishwakarma Institute Of Technology, Intelligent traffic management system:a Review, IJIRST 2016.

[4]. Li, Z., Wang, B., and Zhang, J. "Comparative analysis of drivers' start- up time of the first two vehicles at signalized intersections", 2016 J. Adv. Transp., 50: 228– 239. doi: 10.1002/atr.1318

[5]. 'Pygame Library', 2019. [Online]. Available: https://www.pygame.org/wiki/about

[6]. Mihir M. Gandhi, Devansh S. Solanki, Rutwij S. Daptardar, Nirmala Shinde Baloorkar, "Smart Control of Traffic Light Using Artificial Intelligence", 5th IEEE ICRAIE 2020 (IEEE Record#51050), DOI: 10.1109/ICRAIE51050.2020.9358334

BIODATA

Name : Gudi Varaprasad

Mobile Number : 9398589362

E-mail : Varaprasad.19bce7048@vitap.ac.in