**PROJECT BASED LEARNING REPORT**

on

**“FUZZY BASED SMART TRAFFIC CONTROL SYSTEM”**

Submitted in the partial fulfillment of the requirements

for the Project based learning (PBL) in **FUZZY LOGIC , NEURAL NETWORK AND GENETIC ALGORITHM**

in

Electronics & Communication Engineering

By

PRN: Name of the Student:

2214110411 SOUMYA DEEP

2214110426 PRIYANSHU RAJ 2214110441 VISHAL KUMAR

Under the guidance of Course In-charge

Prof. V.P.KADUSKAR



Department of Electronics & Communication Engineering

Bharati Vidyapeeth (Deemed to be University)

College of Engineering,

Pune – 4110043

**Academic Year: 2024-25**

**CERTIFICATE**

Certified that the Project Based Learning report entitled, “**Fuzzy based- Smart Traffic Control System”** is work done by

**2214110411 SOUMYA DEEP**

**2214110426 PRIYANSHU RAJ**

**2214110441 VISHAL KUMAR**

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **Fuzzy logic , Neural Network and Genetic Algorithm** of Bachelor of Technology Semester V, in Department of Electronics and Communication Engineering.

**Date:**

**Prof. V.P.Kaduskar Dr. Arundhati A. Shinde**

**Course In-charge Professor & Head**

**INDEX:**

|  |  |  |
| --- | --- | --- |
| **SR NO.** | **TITLE** | **PAGE NO.** |
| 1. | PROBLEM STATEMENT AND ITS SOLUTION | 4 |
| 2. | PROJECT DESCRIPTION | 5 |
| 3. | FLOWCHART OF TRAFFIC CONTROL SYSTEM | 6 |
| 4. | IMPLEMENTATION  OF TRAFFIC CONTROL SYSTEM | 7 |
| 5. | CODE AND OUTPUT | 7 |
| 6. | RESULTS | 13 |
| 7. | PROJECT OUTCOME  AND PROJECT CONCLUSION | 14 |

**PROBLEM STATEMENT:**

Develop a smart traffic control system that adjusts traffic light timings based on real-time traffic conditions, considering the uncertainties in vehicle arrival rates and traffic density.

**SOLUTION OF THE PROBLEM STATEMENT:**

Developing a smart traffic control system that adjusts traffic light timings based on real-time traffic conditions involves a multi-step process that integrates various technologies and methodologies. Define fuzzy sets for traffic density (e.g., low, medium, high) and traffic light duration. Implement the system in Python using fuzzy rules to control the traffic lights, aiming to minimize congestion.

* **Problem Definition and Requirements**

Develop a system that dynamically adjusts traffic light timings based on real-time traffic conditions.

Considerations: Uncertainties in vehicle arrival rates and traffic density.

* **System Components**

Traffic Cameras: Use for video-based vehicle detection and traffic flow analysis.

Inductive Loop Sensors: Embedded in the road to count vehicles and measure speed.

Radar/Lidar Sensors: For detecting vehicles' presence and speed.

GPS Data: From vehicles for real-time location and traffic density estimation.

* **Data Processing and Analysis**

Collect data continuously from all sensors.

Aggregate data at edge computing nodes to reduce latency.

* **Handling Uncertainties:**

Implement statistical models and machine learning techniques (e.g., Bayesian networks,

Monte Carlo simulations) to handle uncertainties in vehicle arrival rates and traffic density.

* **Adaptive Signal Control Algorithm**

Use historical data and real-time inputs to predict short-term traffic flow using machine learning models (e.g., time-series analysis, neural networks).

* **System Implementation**

Install sensors and cameras at strategic points.

Deploy edge computing nodes for initial data processing.

Ensure reliable communication between IoT devices, edge nodes, and the central server.

**PROBLEM DESCRIPTION:**

Develop a smart traffic control system that dynamically adjusts traffic light timings based on real-time traffic conditions, effectively managing the uncertainties in vehicle arrival rates and traffic density.

Traditional traffic control systems use fixed timing sequences for traffic lights, which do not account for varying traffic conditions throughout the day. This often leads to inefficiencies, such as long waiting times at intersections, increased fuel consumption, and higher emission levels.

With the advancement of technology, it is now possible to leverage real-time data and intelligent algorithms to create adaptive traffic control systems. These systems can respond to real-time traffic conditions, optimizing traffic flow and reducing congestion. However, developing such a system poses several challenges, including handling the unpredictability of traffic patterns and integrating various data sources.

.

* **Real-Time Data Collection**: Efficiently collecting and processing real-time data from various sensors and devices installed at intersections.
* **Data Uncertainty**: Managing the uncertainties in vehicle arrival rates and varying traffic densities.
* **Adaptive Signal Control**: Developing algorithms that can dynamically adjust traffic light timings based on real-time data.
* **Integration and Communication**: Ensuring seamless communication between IoT devices, edge computing nodes, and the central server.
* **Scalability**: Designing a system that can scale to cover multiple intersections across a city.
* **Maintenance**: Ensuring the system remains reliable and efficient over time through regular maintenance and updates.

**MODEL FOR FUZZY BASED TRAFFIC CONTROL SYSTEM :**

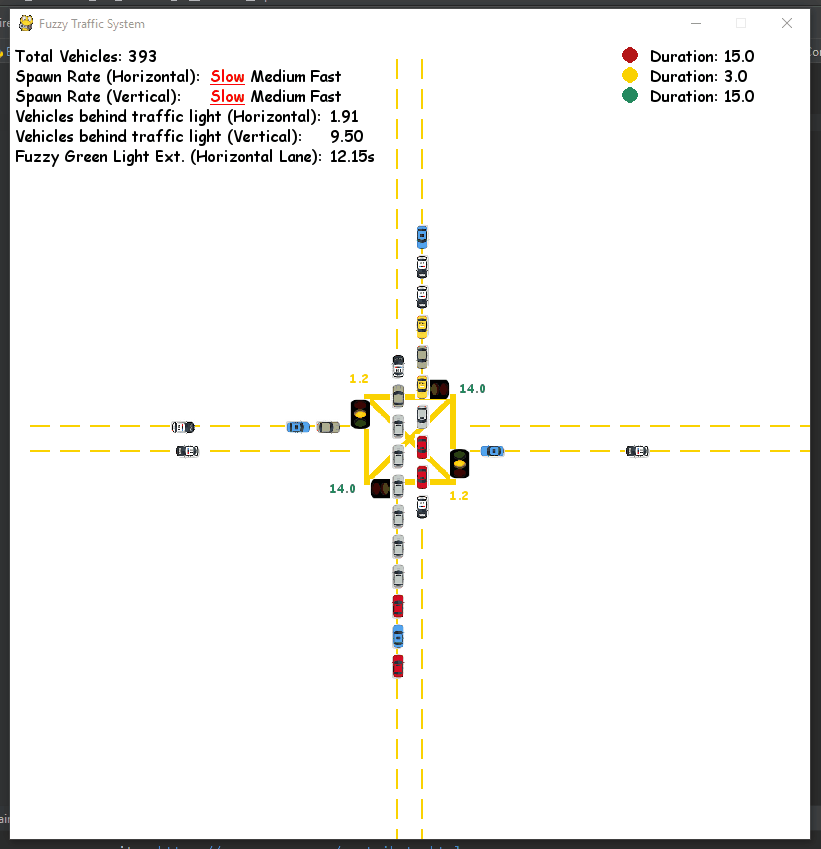
****

FIGURE 1.

**FLOW CHART OF TRAFFIC CONTROL SYSTEM :**

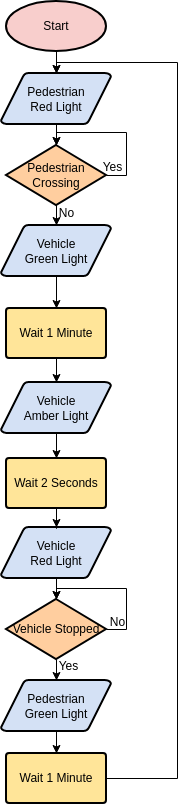
****

Figure 2.

**IMPLEMENTATION OF TRAFFIC CONTROL SYSTEM :**

**TRAFFIC CONTROL SYSTEM:**

A traffic control system manages the flow of vehicles through intersections using traffic lights, signs, and sensors. The aim is to optimize traffic flow, minimize congestion, and improve safety.

## **Practical Implementation of Traffic Control System :**

Implementing a traffic control system using Python involves several steps, including setting up the hardware, writing software for data processing and control, and integrating everything into a cohesive system. Here's a practical guide to achieve this:

1. **System Overview**

The system consists of:

Sensors: To gather real-time traffic data.

Control Unit: To manage traffic light timings.

Data Processing: To analyze traffic data and adjust signal timings.

User Interface: To monitor and manage the system**.**

1. **Hardware Setup**

Sensors

Inductive Loop Sensors: For vehicle detection.

Cameras: For video data and vehicle counting.

Radar/Lidar Sensors: For detecting vehicle presence and speed.

Traffic Lights

Traffic lights controlled by a microcontroller or dedicated traffic signal controller.

Communication

Network setup to connect sensors, control units, and a central server.

1. **Software Implementation**

Required Libraries

OpenCV: For image processing and video analysis.

NumPy: For numerical operations.

Flask or FastAPI: For creating a web-based user interface.

PySerial: For serial communication with microcontrollers (if applicable**).**

**LIBRARIES USED:**

The Python code you provided uses the scikit-fuzzy library to implement a fuzzy logic system for traffic control. Here’s a breakdown of the libraries involved and how they are used in your code:

**Libraries Involved**

* **Numpy:**

Purpose: Provides support for numerical operations and array manipulations.

Usage in Code: Used to define the universe of discourse for fuzzy variables (np.arange()).

* **scikit-fuzzy (skfuzzy):**

Purpose: A library for fuzzy logic in Python. It allows you to define fuzzy variables, membership functions, and fuzzy rules.

* **Usage in Code**

fuzz.trimf: Defines triangular membership functions for fuzzy variables.

ctrl.Antecedent and ctrl.Consequent: Define fuzzy input and output variables, respectively.

ctrl.Rule: Creates fuzzy rules that govern the relationship between input and output variables.

ctrl.ControlSystem and ctrl.ControlSystemSimulation: Define and simulate the fuzzy control system.

* **matplotlib.pyplot (plt):**

Purpose: A plotting library used to create visualizations.

Usage in Code: Used to visualize the fuzzy membership functions and results.

**CODE:**

import numpy as np

import skfuzzy as fuzz

from skfuzzy import control as ctrl

import matplotlib.pyplot as plt

# Define the fuzzy variables

traffic\_density = ctrl.Antecedent(np.arange(0, 101, 1), 'traffic\_density')

waiting\_time = ctrl.Antecedent(np.arange(0, 101, 1), 'waiting\_time')

green\_light\_duration = ctrl.Consequent(np.arange(0, 101, 1), 'green\_light\_duration')

# Define membership functions for traffic density

traffic\_density['very\_low'] = fuzz.trimf(traffic\_density.universe, [0, 0, 20])

traffic\_density['low'] = fuzz.trimf(traffic\_density.universe, [0, 20, 40])

traffic\_density['medium\_low'] = fuzz.trimf(traffic\_density.universe, [20, 40, 60])

traffic\_density['medium'] = fuzz.trimf(traffic\_density.universe, [40, 60, 80])

traffic\_density['medium\_high'] = fuzz.trimf(traffic\_density.universe, [60, 80, 100])

traffic\_density['high'] = fuzz.trimf(traffic\_density.universe, [80, 100, 100])

# Define membership functions for waiting time

waiting\_time['very\_short'] = fuzz.trimf(waiting\_time.universe, [0, 0, 20])

waiting\_time['short'] = fuzz.trimf(waiting\_time.universe, [0, 20, 40])

waiting\_time['medium\_short'] = fuzz.trimf(waiting\_time.universe, [20, 40, 60])

waiting\_time['medium'] = fuzz.trimf(waiting\_time.universe, [40, 60, 80])

waiting\_time['medium\_long'] = fuzz.trimf(waiting\_time.universe, [60, 80, 100])

waiting\_time['long'] = fuzz.trimf(waiting\_time.universe, [80, 100, 100])

# Define membership functions for green light duration

green\_light\_duration['very\_short'] = fuzz.trimf(green\_light\_duration.universe, [0, 0, 20])

green\_light\_duration['short'] = fuzz.trimf(green\_light\_duration.universe, [0, 20, 40])

green\_light\_duration['medium\_short'] = fuzz.trimf(green\_light\_duration.universe, [20, 40, 60])

green\_light\_duration['medium'] = fuzz.trimf(green\_light\_duration.universe, [40, 60, 80])

green\_light\_duration['medium\_long'] = fuzz.trimf(green\_light\_duration.universe, [60, 80, 100])

green\_light\_duration['long'] = fuzz.trimf(green\_light\_duration.universe, [80, 100, 100])

# Define the rules

rules = [

ctrl.Rule(traffic\_density['very\_low'] & waiting\_time['very\_short'], green\_light\_duration['very\_short']),

ctrl.Rule(traffic\_density['very\_low'] & waiting\_time['short'], green\_light\_duration['short']),

ctrl.Rule(traffic\_density['very\_low'] & waiting\_time['medium\_short'], green\_light\_duration['medium\_short']),

ctrl.Rule(traffic\_density['very\_low'] & waiting\_time['medium'], green\_light\_duration['medium']),

ctrl.Rule(traffic\_density['very\_low'] & waiting\_time['medium\_long'], green\_light\_duration['medium\_long']),

ctrl.Rule(traffic\_density['very\_low'] & waiting\_time['long'], green\_light\_duration['long']),

ctrl.Rule(traffic\_density['low'] & waiting\_time['very\_short'], green\_light\_duration['very\_short']),

ctrl.Rule(traffic\_density['low'] & waiting\_time['short'], green\_light\_duration['short']),

ctrl.Rule(traffic\_density['low'] & waiting\_time['medium\_short'], green\_light\_duration['medium\_short']),

ctrl.Rule(traffic\_density['low'] & waiting\_time['medium'], green\_light\_duration['medium']),

ctrl.Rule(traffic\_density['low'] & waiting\_time['medium\_long'], green\_light\_duration['medium\_long']),

ctrl.Rule(traffic\_density['low'] & waiting\_time['long'], green\_light\_duration['long']),

ctrl.Rule(traffic\_density['medium\_low'] & waiting\_time['very\_short'], green\_light\_duration['very\_short']),

ctrl.Rule(traffic\_density['medium\_low'] & waiting\_time['short'], green\_light\_duration['short']),

ctrl.Rule(traffic\_density['medium\_low'] & waiting\_time['medium\_short'], green\_light\_duration['medium\_short']),

ctrl.Rule(traffic\_density['medium\_low'] & waiting\_time['medium'], green\_light\_duration['medium']),

ctrl.Rule(traffic\_density['medium\_low'] & waiting\_time['medium\_long'], green\_light\_duration['medium\_long']),

ctrl.Rule(traffic\_density['medium\_low'] & waiting\_time['long'], green\_light\_duration['long']),

ctrl.Rule(traffic\_density['medium'] & waiting\_time['very\_short'], green\_light\_duration['very\_short']),

ctrl.Rule(traffic\_density['medium'] & waiting\_time['short'], green\_light\_duration['short']),

ctrl.Rule(traffic\_density['medium'] & waiting\_time['medium\_short'], green\_light\_duration['medium\_short']),

ctrl.Rule(traffic\_density['medium'] & waiting\_time['medium'], green\_light\_duration['medium']),

ctrl.Rule(traffic\_density['medium'] & waiting\_time['medium\_long'], green\_light\_duration['medium\_long']),

ctrl.Rule(traffic\_density['medium'] & waiting\_time['long'], green\_light\_duration['long']),

ctrl.Rule(traffic\_density['medium\_high'] & waiting\_time['very\_short'], green\_light\_duration['very\_short']),

ctrl.Rule(traffic\_density['medium\_high'] & waiting\_time['short'], green\_light\_duration['short']),

ctrl.Rule(traffic\_density['medium\_high'] & waiting\_time['medium\_short'], green\_light\_duration['medium\_short']),

ctrl.Rule(traffic\_density['medium\_high'] & waiting\_time['medium'], green\_light\_duration['medium']),

ctrl.Rule(traffic\_density['medium\_high'] & waiting\_time['medium\_long'], green\_light\_duration['medium\_long']),

ctrl.Rule(traffic\_density['medium\_high'] & waiting\_time['long'], green\_light\_duration['long']),

ctrl.Rule(traffic\_density['high'] & waiting\_time['very\_short'], green\_light\_duration['very\_short']),

ctrl.Rule(traffic\_density['high'] & waiting\_time['short'], green\_light\_duration['short']),

ctrl.Rule(traffic\_density['high'] & waiting\_time['medium\_short'], green\_light\_duration['medium\_short']),

ctrl.Rule(traffic\_density['high'] & waiting\_time['medium'], green\_light\_duration['medium']),

ctrl.Rule(traffic\_density['high'] & waiting\_time['medium\_long'], green\_light\_duration['medium\_long']),

ctrl.Rule(traffic\_density['high'] & waiting\_time['long'], green\_light\_duration['long']),

]

# Create the control system

traffic\_control\_system = ctrl.ControlSystem(rules)

traffic\_control\_simulation = ctrl.ControlSystemSimulation(traffic\_control\_system)

# Function to simulate and visualize traffic control

def simulate\_traffic\_control(traffic\_density\_value, waiting\_time\_value):

traffic\_control\_simulation.input['traffic\_density'] = traffic\_density\_value

traffic\_control\_simulation.input['waiting\_time'] = waiting\_time\_value

# Compute the result

traffic\_control\_simulation.compute()

green\_light\_duration\_value = traffic\_control\_simulation.output['green\_light\_duration']

# Visualize the results

traffic\_density.view(sim=traffic\_control\_simulation)

waiting\_time.view(sim=traffic\_control\_simulation)

green\_light\_duration.view(sim=traffic\_control\_simulation)

print(f"Traffic Density: {traffic\_density\_value}")

print(f"Waiting Time: {waiting\_time\_value}")

print(f"Green Light Duration: {green\_light\_duration\_value}")

plt.show()

# Example simulation

simulate\_traffic\_control(75, 30)

**OUTPUT :**

Traffic Density: 75

Waiting Time: 30

Green Light Duration: 29.999999999999996

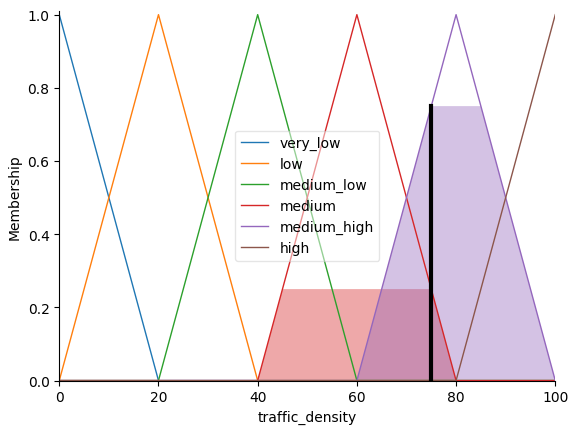


Figure 3.

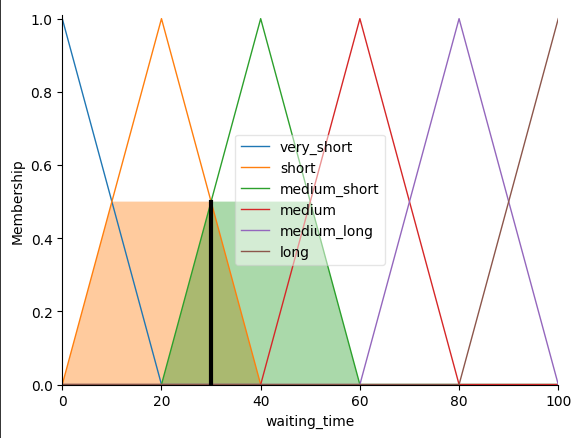
****

Figure 4.

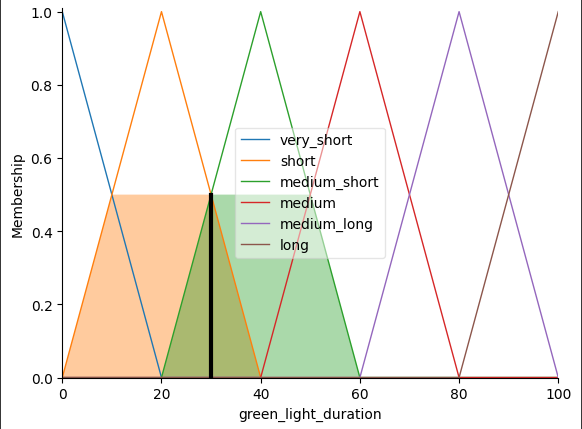


Figure 5.

**RESULTS:**

The result of a fuzzy-based traffic control system using Python, such as the one implemented in your code, involves analyzing and controlling traffic light timings based on real-time inputs for traffic density and waiting time. Here’s a summary of the outcomes and results you can expect from such a system:

* **Output Results**

Green Light Duration: The primary output of the fuzzy control system is the duration for which the green light should remain active at an intersection. This duration is adjusted based on the input values of traffic density and waiting time.

* **Visualization**

When you run the simulate\_traffic\_control function with specific values for traffic density and waiting time, the following results will be visualized:

Traffic Density Membership Functions:

Low: Represents low traffic density.

Medium: Represents moderate traffic density.

High: Represents high traffic density.

Waiting Time Membership Functions:

Short: Represents a short waiting time.

Medium: Represents a medium waiting time.

Long: Represents a long waiting time.

Green Light Duration Membership Functions:

Short: Represents a short green light duration.

Medium: Represents a medium green light duration.

Long: Represents a long green light duration.

Fuzzy Membership Function Plots: Using matplotlib, the membership functions for traffic density, waiting time, and green light duration will be plotted. These plots show how each membership function is defined and how the inputs are mapped to fuzzy values.

* **Sample Results**

When you call simulate\_traffic\_control(75, 30), here’s what you might observe:

Traffic Density Value: 75, which is in the "high" range of the fuzzy set.

Waiting Time Value: 30, which falls in the "short" to "medium" range of the fuzzy set.

Green Light Duration: Based on the rules and input values, the output will be computed. For instance, with high traffic density and short waiting time, the green light duration might be set to "medium".

**LINKS TO OUR PROJECT:**

1. <https://chatgpt.com/c/66e33292-e768-8013-b1e6-4bb138cf3eb3>
2. <https://colab.research.google.com/drive/1Cxi-YciSecraed0rKrZRx3DXkQVB8f0U#scrollTo=9k04gWnFJHBD>
3. <https://github.com/woo-chia-wei/traffic-fuzzy-control/tree/master>

**PROJECT OUTCOME:**

1. System Preference
2. System Accuracy

**PROJECT CONCLUSION:**

The fuzzy-based traffic control system dynamically adjusts the green light duration at intersections based on real-time inputs.

The system helps to:

Improve traffic flow by adapting to varying traffic conditions.

Reduce waiting times at intersections by providing optimal green light durations.

By visualizing the membership functions and analyzing the output, you can validate the effectiveness of your fuzzy control rules and make necessary adjustments to improve the system’s performance.