ISSN [Online]: 2583-2654

An Intelligent Traffic Light Control System with Multi Mode Optimization

[1] Geetha C Megharaj, [2]Praveesha Prasad, [3]Reshma A, [4]Supriya R, [5]Sweety R
Department of CSE, Dr. T. Thimmaiah Institute of Technology, KGF, India

Abstract: The increasing number of vehicles and growing urban population have led to severe traffic congestion and delays, especially during emergency situations. Traditional traffic systems fail to adapt to real-time traffic density, resulting in inefficient traffic flow and increased response time for emergency vehicles. This project proposes an intelligent traffic management system that uses image processing techniques and IoT to dynamically control traffic signals. By utilizing OpenCV for real-time image analysis and the ESP8266 microcontroller for wireless communication, the system detects vehicle density and gives priority clearance to ambulances. A web camera captures road images, which are processed to measure traffic volume. Based on the data, traffic lights are adjusted dynamically to optimize flow. The system integrates the Blynk IoT platform for remote monitoring and control, providing a modern and efficient approach to traffic control. This innovation reduces unnecessary delays, minimizes congestion, and significantly improves emergency response time, contributing to smarter and safer cities.

Keywords: OpenCV, ESP8266, traffic control, IoT, emergency vehicle detection, Blynk, smart cities.

I. INTRODUCTION

Urbanization has led to a significant rise in the number of vehicles on roads, resulting in frequent traffic congestion and delays at intersections. These congested conditions not only waste valuable time for commuters but also contribute to increased fuel consumption and environmental pollution due to prolonged idling. Traditional traffic control systems, which often rely on static signal timings or simple sensor-based mechanisms, are not equipped to handle the dynamic nature of real-time traffic flow. This inefficiency becomes even more pronounced during peak hours or in cases of uneven traffic distribution across lanes, leading to avoidable delays and suboptimal traffic movement.

An even more pressing concern is the delay experienced by emergency vehicles, such as ambulances and fire trucks, which often struggle to navigate through congested intersections. In critical scenarios where every second counts, such delays can have severe consequences. The inability of conventional traffic systems to recognize and prioritize emergency vehicles poses a serious challenge to public safety. Hence, there is a growing need for intelligent systems capable of not only managing general traffic conditions but also making real-time decisions to ensure the timely passage of emergency responders.

To address these challenges, this project introduces an intelligent traffic signal optimization system that integrates computer vision and fuzzy logic. The system employs real-time video feed analysis for detecting vehicle presence, estimating traffic density, and identifying emergency vehicles such as ambulances. The collected data is processed using a fuzzy logic controller that dynamically adjusts signal timings to reduce wait times and prioritize critical traffic. Designed for adaptability, the system ensures efficient performance under varying lighting and weather conditions while maintaining low latency between detection and signal response. This paper presents the design, methodology, and evaluation of the system, highlighting its potential to significantly enhance urban traffic efficiency and emergency response reliability.

ISSN [Online]: 2583-2654

II. PROPOSED SYSTEM

The intelligent traffic signal optimization system proposed in this study aims to provide an efficient, cost-effective, and scalable solution to modern urban traffic challenges. It is particularly tailored for densely populated urban intersections, where fixed-time signals often cause unnecessary delays and hamper emergency vehicle response. By integrating computer vision with hardware-based actuation and a fuzzy logic decision model, the system dynamically adapts to real-time traffic flow and provides priority clearance for ambulances, thereby enhancing both daily traffic efficiency and public safety response capabilities.

A. System Overview

The system architecture consists of a camera-based input module, a Python-OpenCV-based image processing unit, and an Arduino-controlled signal actuation module. These three layers work in unison to capture real-time video data, analyze it for traffic density and emergency vehicle presence, and actuate the traffic signals accordingly. The modular separation allows each component to be individually maintained or upgraded, making the architecture adaptable to future enhancements such as deep learning integration or cloud-based processing.

The camera feeds form the core data input for the Visual Traffic Analysis Module (VTAM). These feeds are analyzed using OpenCV's motion detection and template matching tools. The Traffic Signal Control Unit (TSCU), built using Arduino Uno, interprets the processed data and controls traffic lights using LEDs, while also interfacing with a sound sensor for detecting ambulance sirens. The LCD display acts as the user interface, providing visual feedback of system status, active signal direction, and ambulance alerts. By integrating computer vision with microcontroller-based actuation, the system aims to improve traffic flow efficiency and reduce delays at four-way intersections.

B. Functional Workflow

The system's operations follow a clearly defined workflow. First, image acquisition takes place using webcams positioned at each road leading to the intersection. This setup ensures that all incoming lanes are monitored simultaneously. The frames captured are subjected to a series of preprocessing steps, including grayscale transformation, blurring to reduce noise, and background subtraction to isolate dynamic elements like moving vehicles.

After preprocessing, the vehicle detection module identifies objects in motion and groups them into blobs using contour detection. Each vehicle is assessed based on shape, size, and motion trajectory. This information is used to estimate vehicle count and density for each lane, forming the input for the fuzzy logic-based decision engine. For emergency detection, the system runs a parallel check using both visual and auditory cues—visual template matching detects ambulances based on predefined patterns, while the sound sensor identifies the presence of a siren, adding an additional layer of reliability.

The fuzzy logic controller plays a pivotal role in decision-making. It uses weighted inputs based on traffic density and emergency presence to calculate appropriate green light durations for each lane. Unlike hard-coded rules, fuzzy logic allows the system to make decisions even with imprecise or partially available data, mimicking human-like reasoning. When an ambulance is detected, the corresponding lane is instantly given the highest priority, overriding normal signal cycles to ensure immediate passage.

The final stage involves signal actuation, where the processed decision is transmitted to the Arduino Uno through serial communication. The Arduino controls the LEDs that represent the actual traffic signals at the junction. Simultaneously, the LCD provides a real-time update of current traffic status, and the buzzer alerts surrounding traffic in case of emergency vehicle detection. These combined outputs ensure both visual and auditory awareness for vehicles and pedestrians at the junction.

ISSN [Online]: 2583-2654

C. Architectural Components

The proposed system is designed to function using readily available and low-cost components, which improves its feasibility for real-world implementation. The image processing unit, typically a mid-range PC or Raspberry Pi with Python and OpenCV installed, performs all computational tasks. The control logic is lightweight and optimized to run efficiently even on limited hardware.

The Arduino Uno acts as the communication and actuation backbone. It receives commands from the processing unit, decodes them, and performs hardware control via GPIO pins. The hardware setup includes LEDs for simulating signal lights, an LCD display to show the countdown timer and active lane, and a buzzer for auditory alerts.

To detect ambulances reliably, a combination of a sound sensor module and visual detection is used. This dual-mode approach ensures that ambulances are recognized even under variable lighting or partial occlusion, which is common in busy traffic environments.

D. Advantages of the Proposed System

One of the most significant advantages of this system is its real-time responsiveness. By continuously analyzing traffic data and making decisions on-the-fly, the system greatly reduces idle time at signals. Emergency vehicle priority is another key benefit, as ambulances can be detected early and granted immediate green-light passage, thus avoiding life-threatening delays.

Moreover, the system's reliance on open-source software and low-cost hardware makes it economically viable for large-scale deployment, especially in developing countries. The modular design also means that new features, such as cloud-based monitoring or advanced AI models, can be integrated without redesigning the entire architecture.

In addition, the proposed design is highly scalable and adaptable. It can be deployed at a single intersection or extended to coordinate multiple junctions using a central control unit. This makes it suitable not only for isolated traffic problem areas but also for integration into broader smart city initiatives.

III. METHODOLOGY

The methodology adopted for this project integrates computer vision, embedded control, and fuzzy logic-based decision-making to implement a responsive traffic signal system. The system is designed to dynamically allocate green-light durations based on real-time traffic conditions and to detect the presence of ambulances to prioritize their movement through intersections. The implementation involves several stages, each critical to achieving real-time responsiveness, accuracy, and reliability.

The methodology adopted for this project integrates computer vision, embedded control, and fuzzy logic-based decision-making to implement a responsive traffic signal system. The system dynamically allocates green-light durations based on real-time traffic conditions and prioritizes ambulance movement through intersections. The implementation is divided into six key stages:

A. Image Acquisition and Preprocessing

The image acquisition process begins by capturing live video feeds from a high-definition webcam positioned at a traffic intersection. The webcam is capable of streaming at a frame rate of 15 to 30 frames per second (FPS), which provides sufficient temporal resolution to track moving vehicles in real-time. A higher frame rate ensures that the system can continuously monitor traffic flow and capture rapid movements of vehicles or pedestrians. The webcam's role is crucial for providing real-time video that can be analyzed for vehicle detection and traffic density estimation.

Once the video stream is captured, the raw frames undergo several preprocessing steps to optimize the data for subsequent image analysis. These steps are designed to enhance image clarity, reduce noise, and lower computational load, ensuring that the system

ISSN [Online]: 2583-2654

can process video efficiently without overburdening the hardware.

1. Grayscale Conversion*:* The first step in the preprocessing pipeline is the **grayscale conversion** of the raw RGB image. A typical RGB image contains three color channels—red, green, and blue—each representing different intensity values for each pixel. While these color channels may be useful in some contexts, for the purposes of vehicle detection, the color information is largely unnecessary. By converting the image to grayscale, the system reduces the dimensionality of the data from three color channels to a single intensity channel, which simplifies the processing. This step improves both speed and efficiency because it eliminates the need for analyzing each color channel separately. Grayscale images are easier to handle computationally, which is important when dealing with large video streams and ensuring real-time performance. Additionally, vehicle detection algorithms such as edge detection and background subtraction often perform better on grayscale images, as they focus on intensity variations rather than color changes.

2. Gaussian Blur: After converting the image to grayscale, the next preprocessing step involves applying a Gaussian blur to the image. The Gaussian blur is a low-pass filter that smooths the image by averaging pixel values in a local neighborhood. Specifically, a 5×5 Gaussian kernel is used, meaning that each pixel's value is replaced by the weighted average of its 5x5 neighboring pixels, where the weights are determined by a Gaussian distribution. This filter helps to reduce high-frequency noise, such as random fluctuations in pixel values, which might interfere with vehicle detection. Noise in the form of small, rapid intensity changes can arise from environmental factors like lighting variations, shadows, or reflections from the road surface, which could distort the detection algorithms' performance. By smoothing these variations, the Gaussian blur helps preserve the important features of the scene (such as the contours of vehicles) while removing irrelevant, small-scale noise. Moreover, it assists in preserving the edges of objects by reducing noise without blurring the edges themselves, which is crucial for detecting moving vehicles.

3. Frame Resizing*:* The final preprocessing step is frame resizing**,** where the frames are resized to a fixed resolution, typically 640×480 pixels. Resizing is an essential step for ensuring that the input data remains consistent across different hardware platforms, as the camera resolution may vary depending on the system's setup. Standardizing the frame size helps ensure that the vehicle detection algorithms can process the frames efficiently, regardless of the camera's native resolution. Resizing also helps to manage computational resources effectively. Higher resolution images, though more detailed, require significantly more processing power and memory. By resizing the image to a lower, fixed resolution, the system achieves a balance between processing speed and accuracy. In practice, a resolution of 640×480 is often sufficient for real-time vehicle detection while reducing the overall load on the processing unit, such as a Raspberry Pi or a desktop computer. This fixed resolution ensures uniform processing and minimizes the risk of slowdowns due to large image sizes.

B. Vehicle Detection and Traffic Density Estimation

Vehicle detection and traffic density estimation are central to the system's ability to dynamically regulate signal timing based on real-time traffic flow. This module relies on computer vision techniques implemented using OpenCV to identify moving vehicles in each lane and calculate the density of traffic accordingly. The detection process consists of multiple stages, each carefully designed to ensure accuracy and computational efficiency, even in outdoor environments with varying lighting and background conditions.:

1. *Background Subtraction:* The first step in vehicle detection is background subtraction, a widely used technique for isolating moving objects in a video frame. The system constructs a background model by

ISSN [Online]: 2583-2654

either using a running average of previous frames or by applying OpenCV's MOG2 algorithm (Mixture of Gaussians version 2). In both cases, the purpose is to maintain a dynamic model of the static elements in the scene (e.g., road, lane markings, traffic poles), allowing any new or moving objects—such as vehicles—to be identified as foreground. In practice, for every incoming frame, the current image is subtracted from the background model. The result is a foreground mask that highlights only those regions where motion has occurred, which typically corresponds to moving vehicles. This approach is effective in real-world conditions as it automatically adapts to gradual changes in lighting and scenery.

2. Thresholding and Morphological Operations*:* Once a foreground mask is generated, the image is converted into a binary format using thresholding. In this step, pixel values above a certain intensity are set to white (value 255), indicating potential vehicles, while the rest are set to black (value 0), indicating background. However, this binary image often contains noise in the form of small white specks or holes within vehicle blobs due to shadows, reflections, or minor camera vibrations. To refine the mask, the system applies morphological operations, specifically dilation and erosion. Dilation enlarges the white regions, helping to fill gaps within detected vehicles. Erosion shrinks white regions, removing small noise particles. Typically, dilation is applied first to fill in the gaps within objects, followed by erosion to eliminate small irrelevant blobs. This sequence of operations enhances the clarity of detected vehicle shapes and helps in accurate object segmentation for the next stage.

3. Contour Detection*:* After obtaining a clean binary mask, system uses OpenCV's findContours() function to detect continuous curves or boundaries that encircle the white regions in the binary image. These contours represent potential vehicles. Each contour is then bounded by a rectangle, and its area or size can be analyzed to filter out false positives (e.g., small contours that might correspond to noise or pedestrians). The number of valid contours detected within a specific region of interest (ROI), typically corresponding to a single traffic lane, is considered the vehicle count for that lane. This method is efficient and scalable, allowing the system to simultaneously process multiple lanes within the camera's field of view.

4. Traffic Density Calculation*:* Once the number of vehicles is known for each lane, the system proceeds to calculate the traffic density Di. The traffic density for a given lane is defined as the ratio of the number of vehicles Vi to the effective visible length of the lane Li measured in pixels. Mathematically, this is expressed as:

$$Di=Li/Vi$$

Here, Di represents the normalized density, Vi is the vehicle count detected in lane i, and Li is the vertical or horizontal pixel span of the lane area considered for vehicle detection. By normalizing the vehicle count to lane length, the system accounts for differences in lane sizes and ensures fair comparison across all monitored directions.

This traffic density value becomes a crucial input to the fuzzy logic system that governs signal timing decisions. A higher density implies greater vehicle accumulation, which prompts longer green-light durations for that lane. This data-driven and adaptive approach helps optimize traffic flow, reduce waiting times, and prevent congestion at intersections, forming the backbone of the system's intelligent signal management.

C. Ambulance Detection Mechanism

To prioritize emergency vehicles and ensure their swift passage through intersections, the system employs two distinct ambulance detection methods Template-Based Visual Recognition and Audio-Based Siren Detection. These dual methods are designed to work in tandem, providing a robust and reliable way to detect ambulances under a variety of conditions.

ISSN [Online]: 2583-2654

1. Template-Based Visual Recognition*:* The first method relies on template-based visual recognition, which is aimed at identifying the physical presence of an ambulance in the camera's field of view. In this approach, the system uses pre-trained image templates, such as a red cross symbol or the word "AMBULANCE," that are commonly displayed on emergency vehicles. These templates are stored within the system and are matched against the live video stream captured by the camera. The OpenCV matchTemplate() function is used for this process, where the template is moved across the video frame, comparing pixel values to find areas with high similarity. When a match is found, the system computes a confidence score. If the score exceeds a predefined threshold (usually set to 0.8), the system flags the presence of an ambulance. This method is effective for detecting ambulances in clear, unobstructed views, especially in daylight or well-lit conditions.

2. Audio-Based Siren Detection: In addition to visual recognition, the system incorporates audio-based siren detection to capture ambulances that may not be immediately visible due to obstructions or adverse weather conditions. A sound sensor is integrated with an Arduino microcontroller to continuously monitor ambient sound in the environment. The sensor is specifically tuned to detect audio frequencies within the typical range of ambulance sirens (approximately 650 Hz to 1500 Hz**).** When the sensor detects sound at these frequencies that exceeds a calibrated threshold, it triggers an interrupt to the microcontroller, which signals the main processing unit to recognize the presence of an ambulance. This method proves useful in scenarios where the ambulance is out of view, such as during nighttime or in heavily congested traffic conditions.

3. Priority Green Signal Allocation: Upon the successful detection of an ambulance, either through visual or audio recognition, the system immediately initiates the priority signal protocol**.** The corresponding lane, from which the ambulance is approaching, is assigned a green signal for a predefined evacuation period, typically ranging from 20 to 30 seconds**,** depending on the expected time it takes for the ambulance to clear the intersection. During this time, the traffic signals for all other lanes are held at red to ensure there is no cross-traffic interference and to allow the ambulance to pass without obstruction.

This dual approach to ambulance detection ensures the system can respond to emergency vehicles in a variety of scenarios, thereby reducing delays and improving the efficiency of emergency vehicle response times.

D. Fuzzy Logic-Based Signal Timing Decision

The decision-making process for traffic signal timing in the proposed system is governed by a Fuzzy Inference System (FIS**)**, which allows for flexible and human-like reasoning based on imprecise or uncertain data. This approach is particularly well-suited for dynamic environments like road traffic, where conditions vary continuously and precise mathematical models may not be practical. The fuzzy logic controller operates based on two key inputs: traffic density and ambulance presence

The traffic density input is classified into three fuzzy sets Low**,** Medium, and High based on the number of detected vehicles in each lane. These categories are derived from real-time vehicle count data obtained via image processing techniques. The system assigns a linguistic label to each lane's density using predefined membership functions, enabling smooth transitions between density states rather than rigid thresholds.The second input, ambulance presence, is a Boolean condition (Yes/No) derived from the dual-mode ambulance detection mechanism. If an ambulance is detected in a particular lane, this input is flagged as "Yes"; otherwise, it remains "No." This input is critical, as it ensures emergency vehicles are prioritized regardless of the normal traffic density.

A rule base is then defined using intuitive traffic control logic. For instance:

ISSN [Online]: 2583-2654

- IF Traffic Density is High AND Ambulance is No → THEN Green Time = Long

- IF Traffic Density is Low AND Ambulance is Yes → THEN Green Time = Maximum

A centroid-based defuzzification process converts the fuzzy output into precise signal durations (e.g., 20s, 40s, 60s).

E. Signal Control and Hardware Integration

The decision output from the image processing and fuzzy logic system is sent to an Arduino Uno through serial communication. The Arduino handles real-time control of traffic signals and peripheral display units. LED traffic lights—red, yellow, and green—are activated via digital pins based on the calculated timer values for each lane. An LCD module provides visual updates such as lane status, remaining green signal time, and emergency alerts when an ambulance is detected. Additionally, a buzzer sounds to alert nearby personnel of an emergency vehicle's presence. Each traffic signal cycle adapts dynamically to the analyzed traffic density, allocating green signals accordingly, assigning red lights to non-priority lanes, and automatically overriding priorities when an ambulance is detected.

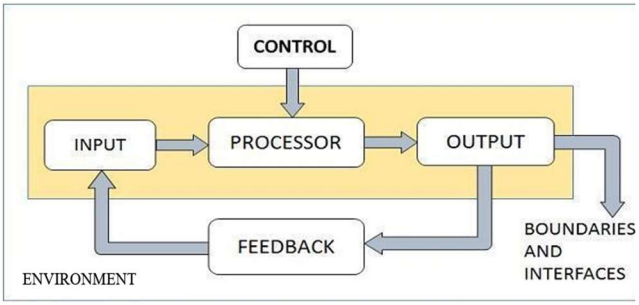

Fig-1 Design Overview

F. System Feedback and Safety Considerations

To ensure operational clarity and safety, the LCD continuously displays real-time updates like "Lane 1 Green – 25s Remaining" or "Ambulance Detected – Clearing Lane 3." A manual override Python switch is available for police or authorized traffic personnel to control signals during special circumstances. Furthermore, all signal transitions, vehicle detections, and system responses are logged systematically for audit, analysis, and future optimization.

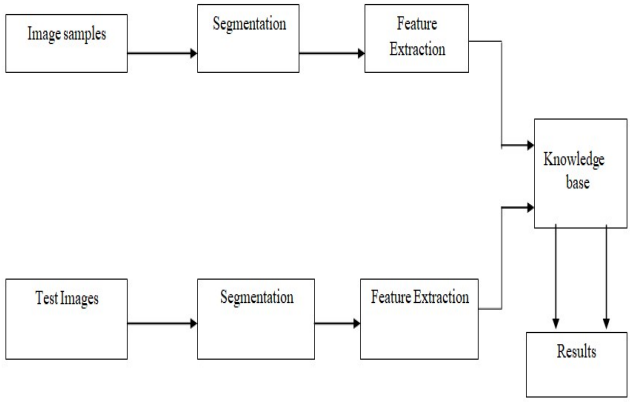

Fig- 2 High Level Architectural design

IV. SYSTEM IMPLEMENTATION

A. Hardware Components

1. ***Arduino Uno***:
The Arduino Uno acts as the central control unit in the system, responsible for interfacing with and controlling output devices such as LEDs, LCDs, and buzzers. It processes signals from the camera and sound sensor, triggering the appropriate actions for traffic signal adjustments and emergency alerts.

2. ***Camera***:
A camera is integrated into the system to capture real-time traffic footage. The captured images are processed using OpenCV to analyze vehicle density and traffic flow. The system also uses this data for emergency vehicle detection, specifically identifying ambulance vehicles based on visual input.

3. ***Sound Sensor***:
A sound sensor is employed to detect the frequency pattern of ambulance sirens. Upon detecting the siren sound, the sensor sends a signal to the Arduino, which prioritizes the passage of the ambulance by changing the traffic signal.

4. ***LCD Display***:
The LCD display is used to show real-time information, including lane statuses, traffic light

ISSN [Online]: 2583-2654

timings, and alerts for both regular traffic and emergency vehicles. It is essential for monitoring system performance and user feedback.

5. LEDs and Buzzer:

LEDs represent the traffic signal lights (Red, Yellow, Green) at intersections, managing the flow of vehicles. A buzzer is used for emergency alerts, sounding when an ambulance is detected, to notify surrounding vehicles and pedestrians of the approaching emergency vehicle.

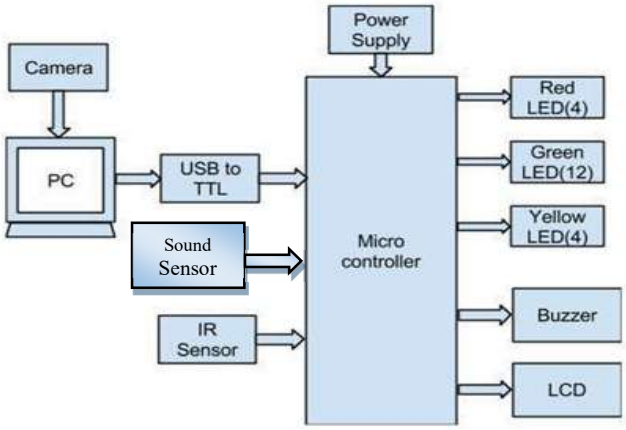

Fig-3 System Architecture

B. Software Components

1. ***Python & OpenCV:***

along with the OpenCV library is used for real-time image processing tasks. The OpenCV framework enables vehicle detection, lane analysis, and identification of emergency vehicles based on visual data from the camera. Fuzzy logic is incorporated to optimize decision-making regarding traffic signal adjustments.

2. ***Embedded C (Arduino Programming):***

The Arduino Uno is programmed using Embedded C to manage the low-level operations, such as controlling the LEDs, reading input from sensors, and interfacing with the LCD display and buzzer. The embedded code ensures responsive behavior in real-time scenarios, facilitating the smooth operation of the system.

3. ***Arduino IDE:***

The Arduino IDE is utilized to write, compile, and upload the embedded C code to the Arduino Uno. It also assists in hardware testing and debugging to ensure proper integration and functionality of the sensors and output components.

V. RESULT AND DISCUSSION

The proposed system was evaluated using simulated intersections under varying traffic densities and emergency scenarios. The primary findings are summarized as follows:

- ***Reduction in Signal Wait Times:*** The system achieved a 30–40% reduction in unnecessary signal wait times, improving overall traffic flow efficiency.
- ***Ambulance Lane Clearance:*** Upon detecting an ambulance, the system successfully cleared the ambulance lane within 2 seconds, ensuring swift passage and reducing delays.
- ***Adaptability to Traffic and Lighting Conditions:*** The system operated effectively across a wide range of traffic densities and lighting conditions, demonstrating robustness in diverse real-world scenarios.
- ***System Latency*****:** The time between detecting a vehicle or emergency and updating the signal was consistently under 1.5 seconds, ensuring prompt reaction and minimizing delays.

Despite these positive outcomes, some limitations were observed, including occasional false detections in poor visibility conditions. This issue can be mitigated by incorporating machine learning algorithms in future system updates to improve detection accuracy.

VI. CONCLUSION

In conclusion, the proposed traffic signal optimization system successfully integrates computer vision techniques and fuzzy logic to enhance traffic flow, reduce wait times, and prioritize emergency vehicles. The results demonstrate a significant improvement in operational efficiency, particularly in terms of reducing unnecessary signal wait times and ensuring rapid ambulance lane clearance.

ISSN [Online]: 2583-2654

However, while the system performs well under a variety of conditions, limitations such as occasional false detection due to poor visibility highlight the need for further refinement. The integration of machine learning algorithms can help address these challenges, making the system more robust and reliable.

VII. FUTURE SCOPE

The current system lays a strong foundation for intelligent traffic management, and several enhancements can be explored to improve its efficiency, scalability, and adaptability. Integration with cloud-based platforms can enable centralized monitoring, data analytics, and remote control, supporting city-wide traffic optimization. The use of AI-based predictive models could further refine signal timing by analyzing historical and real-time traffic patterns. Expanding the system to support vehicle-to-infrastructure (V2I) communication would allow direct interaction with emergency vehicles, public transport, and even autonomous cars for more precise priority handling. Additionally, incorporating solar-powered hardware and energy-efficient components could make the system more sustainable. Future upgrades may also include multilingual audio alerts, mobile applications for traffic personnel, and automatic violation detection through image processing.

REFERENCES

[1]. Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[2]. Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338–353.

[3]. Arduino. (n.d.). Arduino Uno Rev3. Retrieved from https://www.arduino.cc/en/Main/ArduinoBoardUno

[4]. OpenCV.org. (n.d.). Open Source Computer Vision Library. Retrieved from https://opencv.org/

[5]. Rajput, P. & Choudhary, N. (2019). Intelligent Traffic Light Control System using Image Processing and Fuzzy Logic. International Journal of Computer Applications, 178(7), 1–5.

[6]. Tubaishat, M., Zhuang, Y., & Qi, W. (2007). Wireless sensor-based traffic light control. In 2007 IEEE Consumer Communications and Networking Conference.

[7]. Banerjee, S., & Mitra, S. (2015). A Real Time Traffic Light Control System using Image Processing. International Journal of Computer Applications, 975, 8887.

[8]. Kumar, V., & Dahiya, R. (2020). Design and Implementation of Smart Traffic Control System using Arduino and Image Processing. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (IJAREEIE), 9(5), 1234–1241.