SMART TRAFFIC MANAGEMENT SYSTEM FOR SMART CITY

Dr. Vasanthamma H[1] , *T Karthik Varma[2], Kartik S[3], Mohammed Shahezan[4],Khaja Moinuddin[5]

[1]*Professor, CS-AIML Department, Proudhadevaraya Institute of Technology, Hosapete,*
[2345]*Students, CS-AIML Department, Proudhadevaraya Institute of Technology, Hosapete*[1]

Abstract

Traffic congestion and delays at intersections are major problems in rapidly growing cities. Conventional fixed-time signals do not adapt to real-time traffic density and cannot provide fast clearance for emergency vehicles. This paper presents a smart traffic management system for a smart city intersection using computer vision, YOLO-based vehicle detection, and adaptive signal timing. Live camera feeds are processed using OpenCV and a trained YOLO model to detect, classify, and count vehicles on each lane. A weighted density score is computed to represent congestion considering different vehicle classes, and the green time is dynamically allocated based on the computed density. The system also detects ambulances and provides priority by overriding the normal timing to create an emergency clearance path. The prototype integrates Raspberry Pi, Arduino Mega 2560, and a desktop system for monitoring and control, and includes a user interface for displaying vehicle counts and allotted time. Simulation and sample real-road experiments demonstrate improved adaptability of signal timing under varying traffic loads and faster response for emergency vehicles.

Keywords: Smart city; Smart traffic management; YOLO; OpenCV; Raspberry Pi; Arduino Mega 2560; Emergency vehicle priority

1. Introduction

Urban areas such as Mumbai and Bangalore have experienced rapid growth in population and vehicle usage, which leads to heavy traffic congestion at intersections and increased travel time. Traditional traffic signal systems usually operate with a fixed timing plan, and the signal duration is not adjusted according to real-time traffic conditions, which results in inefficiencies during peak and non-peak hours.

A smart traffic management system aims to reduce congestion by continuously estimating traffic density and dynamically allocating green time to lanes with higher demand. With the availability of low-cost cameras and edge computing, computer vision based traffic monitoring can replace or complement electronic sensors that are often affected by installation cost, maintenance, and environmental conditions.

In a smart city context, traffic management also requires priority handling for emergency vehicles such as ambulances. Delays in ambulance movement can be life-threatening; therefore, an automated mechanism to detect an emergency vehicle and provide immediate clearance at the junction is required. This work combines YOLO-based vehicle detection and classification, density-based adaptive signal timing, and ambulance priority to support smart city traffic control.

2. Literature Survey

Several approaches have been proposed for adaptive traffic signal control using image processing and intelligent decision methods. Khushi (2019) proposed smart traffic light control using video

processing, where live feed is processed and dynamic timing improves traffic flow compared to fixed timing, but wide deployment can be expensive when additional infrastructure is required. Vogel et al. (2019) presented a camera-based approach combined with fuzzy logic controllers to optimize signal timing and extend green phases under low traffic density conditions.

Ranjith Soman (2020) proposed traffic light control and violation detection using image processing with ANN and fuzzy control, where grayscale conversion, segmentation, and classification are used to estimate traffic conditions and set timers. Recent deep learning based methods, especially one-stage object detectors such as YOLO, provide robust vehicle detection under varying illumination and occlusions and can be integrated with edge devices for real-time operation. Building on these works, the proposed system uses a trained YOLO model to detect vehicles and emergency vehicles and applies a weighted density-based timing strategy for adaptive control.

3. Materials and Methods

3.1. System Architecture

The proposed system uses camera feeds from four approaches of an intersection. Each camera view is processed to detect vehicles and estimate lane-wise traffic density. Raspberry Pi units can be used for capturing and pre-processing the lane images, while an Arduino Mega 2560 is used for interfacing with signal lights and auxiliary display units. The core detection and timing computation is performed using a trained YOLO model and image processing pipeline.

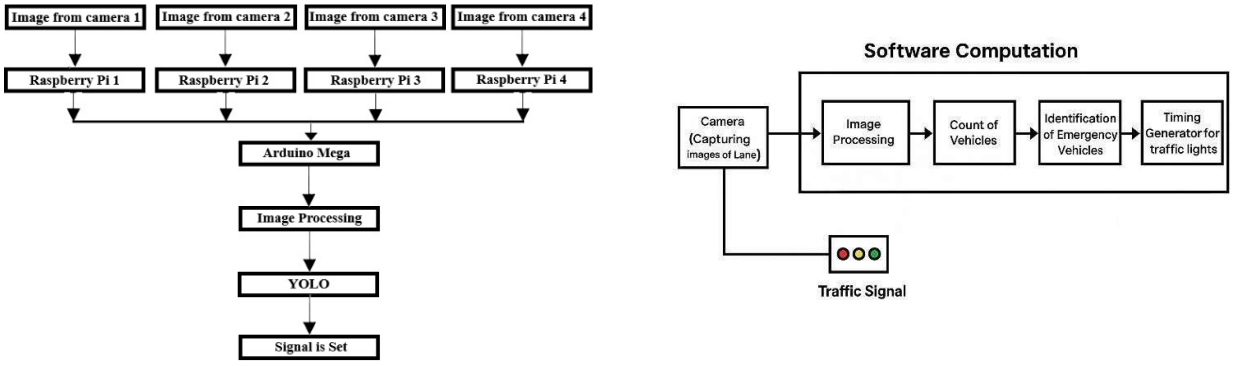

Figure 1. System architecture and software computation pipeline: (a) Multi-camera Raspberry Pi + Arduino Mega integration, (b) Software computation for adaptive signal timing.

3.2. Vehicle Detection and Classification

Vehicle detection and classification are performed using Ultralytics YOLO with a trained model file (best.pt). The implementation reads video frames, applies region-of-interest (ROI) areas for each lane, and runs inference to detect objects. Detected classes include common road vehicles and special classes such as ambulance and police car. The custom class list used in the software contains 21 classes, including car, bus, truck, motorbike, auto rickshaw, and ambulance.

3.3. Density-Based Signal Timing

For each lane, the system counts the detected vehicles and computes a weighted density score. Weights are assigned to vehicle classes to represent their relative road occupancy and impact on congestion. The green time is then allocated proportionally to the lane density while maintaining minimum timing constraints for safety.

Vehicle class	Weight (w)
ambulance	1.1
auto rickshaw	1.5
bicycle	1.6
bus	2.0
car	1.0
minivan	2.0
motorbike	0.8
pickup	2.0
three wheelers -CNG-	1.5
truck	2.5
van	1.1

Table 1. Vehicle class weights used in density computation.

The lane density score is computed as:

$$= \Sigma\_c \, (w\_c \times n\_\{l,c\}) \quad (1)$$

The green time allocation for lane l is computed by normalizing the density score:

$$T\_min + (D\_l / \Sigma\_j \, D\_j) \times (T\_total - 4 \times T\_min) \quad (2)$$

3.4. Emergency Vehicle Priority

Emergency vehicle detection is achieved by identifying the ambulance class in the lane ROI. When an ambulance is detected in a specific lane, the system overrides the normal density-based timing and allocates immediate green time to clear the ambulance path while stopping the remaining lanes. After the emergency vehicle passes, the system resumes adaptive timing based on updated density.

$$bulance \text{ detected in lane } k \rightarrow \text{set lane } k = GREEN, \text{ others} = RED \quad (3)$$

3.5. Implementation Details

The software implementation is developed in Python and uses OpenCV for image processing, Ultralytics YOLO for detection, and a desktop user interface for monitoring. The inference package includes dependencies such as opencv-python, pandas, ultralytics, pyautogui, customtkinter, and pygame. The model training and dataset preparation package is organized around a YOLOv5/PyTorch workflow (hardware package) to generate the trained weight file used during inference. The project can be executed on a desktop system (Intel i3/i5 or higher) and also supports edge deployment using Raspberry Pi for camera interfacing.

Table 2. System requirements for prototype implementation.

Component	Specification
Processor	Intel i3/i5 or higher (3.6–3.7 GHz base clock)
Storage	500 GB hard disk or more
Memory	RAM suitable for image processing (4 GB or higher recommended)
Cameras	Traffic junction cameras / lane cameras for live feed
Controller	Arduino Mega 2560
Edge device	Raspberry Pi
Display	LCD display / monitor (HDMI to D-type converter if needed)
Signal output	Signal light (LED red/yellow/green) and regulated power supply
Software	Windows 11 / Raspberry Pi OS; Python; OpenCV; YOLO (Ultralytics)

4. Results and Discussion

The system was tested using simulation and sample real-road traffic videos. The detector identifies multiple vehicle classes and continuously updates lane-wise counts. Based on the computed density, the green signal is assigned to the lane with higher demand and the timing is adjusted dynamically to reduce waiting time and congestion.

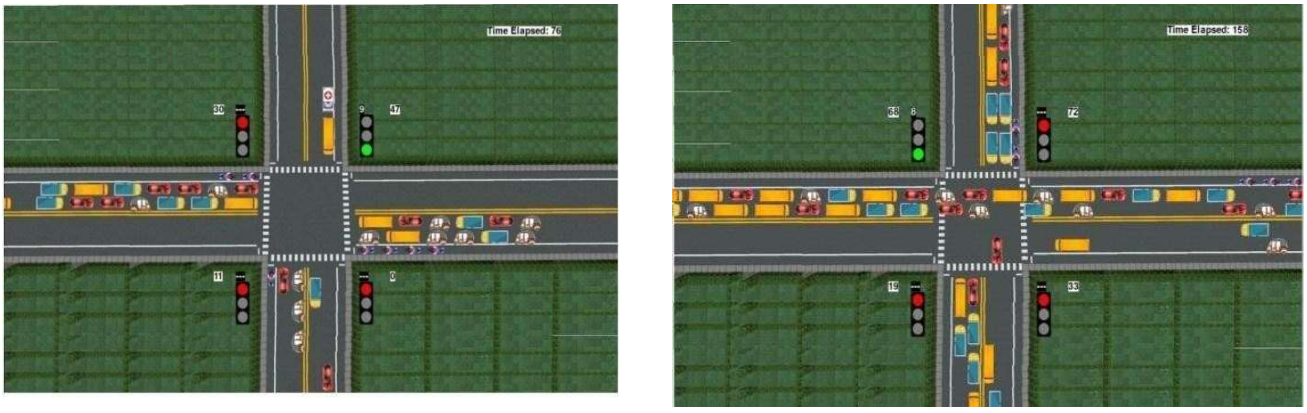

Figure 2. Density-based signal timing in intersection simulation: (a) t = 76 s, (b) t = 158 s.

camera continuously monitors all four lanes at the intersection. Then system counts the number of vehicles in each lane and identify where the traffic is the more. The lane with the higher traffic is then given the green signal, while the other lanes are held on red light. This way, the traffic light adapts in the real time to reduce waiting time and ease congestion.

Table 3. Example green time allocation using Eq. (2) (T_total = 120 s, T_min = 10 s).

Lane	Density score (D_l)	Allocated green time (s)
Lane 1	30	37
Lane 2	47	53
Lane 3	11	20
Lane 4	0	10

Emergency vehicle priority was evaluated by detecting the ambulance class in a lane and triggering a clearance mode. In this mode, the system displays the ambulance lane as GO and sets other lanes to STOP, with the allotted time focused on clearing the ambulance path. After clearance, normal adaptive operation resumes.

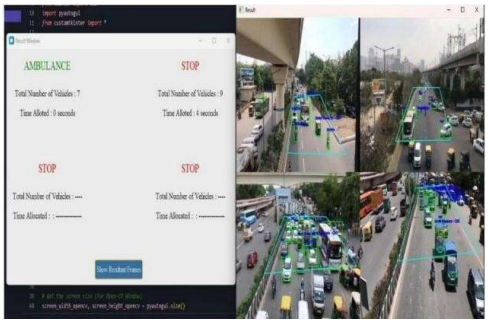

Figure 3. Vehicle detection output and user interface showing ambulance priority.

The prototype demonstrates that computer vision and deep learning can support real-time traffic density estimation without installing intrusive road sensors. The weighted density strategy helps to represent mixed traffic conditions, and the emergency override improves responsiveness for ambulance movement at the junction.

4.1 Prototype Setup Snapshots

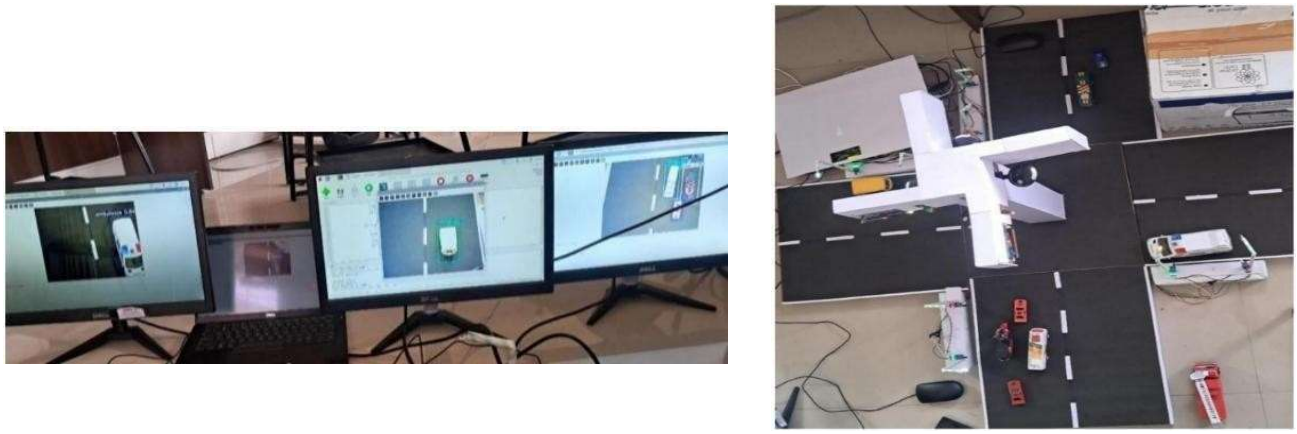

Figure 4. Prototype and testbed setup: (a) multi-monitor system monitoring, (b) physical junction model with sensing and control.

A simulation model is used to detect an ambulance from the camera view. The detection algorithm identifies the vehicle as an ambulance and displays a confidence score beside it. Here, the value 0.84 represents how sure the model is that the detected vehicle is indeed an ambulance. Our smart-city traffic model is shown with four intersecting roads, traffic lights, and a central camera unit. The camera keeps watching all the approaches and checks every vehicle that stops at the signal.

The LCD screen shows the live count (for example, 3 vehicles in one lane and 4 in another), while the small signal beside it indicates the current light status. Each counter is connected to a different road at the junction,
giving real-time vehicle counts that the smart traffic system uses to decide which signal should turn green

Figure 5. LCD alerts for emergency handling: (a) EMERGENCY display, (b) AMBULANCE DETECT IN ROAD 1.

The LCD display shows the message “EMERGENCY”, indicating that an emergency vehicle has been detected on one of the roads. Once the camera identifies an ambulance or fire engine, the controller sends a signal to this unit, which updates the message and alerts the traffic signal beside it. This notification helps the system give immediate priority to the emergency vehicle at the junction.

The LCD clearly displays the message “AMBULANCE DETECT IN ROAD 1.” This means the system has identified an ambulance on Road 1 and flagged it as an emergency. Based on this detection, the controller will give priority to Road 1 by turning its signal green so the ambulance can pass quickly. The detection simulation of vehicle has identified a fire engine from the camera view. The vehicle is highlight with a bounding box and label as “fire-engine” along with a confidence score of 0.69.

5. Conclusion

This paper presented a smart traffic management system using YOLO-based vehicle detection and adaptive signal timing. The system estimates lane-wise density from real-time camera feeds and allocates green time using a weighted density score, which helps to represent mixed traffic conditions. The ambulance detection and priority mechanism provides faster clearance for emergency vehicles by overriding the normal signal cycle when required.

The prototype integrates a practical hardware stack (Raspberry Pi, Arduino Mega 2560, cameras, and signal lights) with a Python-based software pipeline using OpenCV and Ultralytics YOLO. Results from simulation and sample real-road videos show that adaptive allocation improves responsiveness under varying traffic loads and supports emergency clearance. Future work can include multi-junction coordination, cloud-based analytics, and integration with other smart city services.

Acknowledgments

The authors thank Dr. Vasanthamma H (M.Tech., Ph.D.) and the Department of CSE-AIML, Proudhadevaraya Institute of Technology, Hospete, for guidance and support throughout the project work.

The authors would like to acknowledge the support and guidance provided by the project guide and the department, and the facilities provided by the institution for carrying out the prototype implementation

References

[1] TomTom.com, “TomTom global traffic index,” 2019. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/ranking/

[2] K. Khushi, “Smart control of traffic light system using image processing,” in Proc. Int. Conf. Curr. Trends Comput., Electr., Electron. Commun. (CTCEEC), Mysore, India, 2017, pp. 99–103, doi:10.1109/CTCEEC.2017.8454966.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 779–788.

[6] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” arXiv preprint arXiv:1804.02767, Apr. 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[7] J. Tao, H. Wang, X. Zhang, X. Li, and H. Yang, “An object detection system based on YOLO in traffic scenes,” in Proc. 6th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT), Dec. 2017, pp. 315–319.

[9] Ultralytics, “YOLO: Real-time object detection,” [Online]. Available: https://github.com/ultralytics/ultralytics