॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Project Report

on

AI-Driven Traffic Management System With Emergency Vehicle Prioritization

Submitted by

Project Members

Mehul Vaidya 1032210859

Prakhar Pachauri 1032211387

Riya Gupta 1032211270

Tanvi Khadap 1032211484

Under the Internal Guidance of

Prof. Pramod Mali

School of Computer Engineering and Technology

MIT World Peace University, Kothrud,

Pune 411 038, Maharashtra - India

2023-2024

DEPARTMENT OF COMPUTER ENGINEERING AND TECHNOLOGY

C E R T I F I C A T E

This is to certify that,

Mehul Vaidya

Prakhar Pachauri

Riya Gupta

Tanvi Khadap

of BTech.(Computer Science & Engineering) have completed their project titled "*AI-Driven Traffic Management System With Emergency Vehicle Prioritization*" and have submitted this Capstone Project Report towards fulfillment of the requirement for the Degree-Bachelor of Computer Science & Engineering (BTech-CSE) for the academic year 2024-2025

[Prof. Pramod Mali]	**[Dr. Balaji M Patil]**
Project Guide	Program Coordinator
School of CET	School of CET
MIT World Peace University, Pune	MIT World Peace University, Pune

Internal Examiner:

External Examiner:

Date:

Acknowledgement

We would like to express our heartfelt gratitude to Prof. Pramod Mali, our project guide, for his exceptional guidance, support, and encouragement throughout the course of our Capstone Project, titled “AI-Driven Traffic Management System With Emergency Vehicle Prioritization”. His expertise, insightful feedback, and constructive criticism were instrumental in shaping our work and ensuring its successful completion. We are deeply grateful for his unwavering commitment to our progress.

We also extend our sincere thanks to the Department of Computer Engineering and Technology (DCET), MIT World Peace University, Pune, for providing the necessary resources and a conducive environment for completing this project. We are thankful for the opportunity to work on this project, which has enhanced our learning and skillset. Finally, we would like to express our gratitude to our teammates, for their collaboration, hard work, and dedication in making this project a success.

Name of the Students

Mehul Vaidya
Prakhar Pachauri
Riya Gupta
Tanvi Khadap

Abstract

This project presents an intelligent traffic signal management system that integrates real-time vehicle detection, dynamic signal timing, and emergency vehicle prioritization using computer vision and AI models. Motivated by the inefficiencies of static traffic lights, the proposed system utilizes the YOLO v8 object detection algorithm to monitor vehicle density and identify ambulances. A simulation environment built using Pygame replicates a realistic four-way intersection, where signal durations are adaptively controlled based on traffic flow and vehicle type. In the presence of emergency vehicles, the system dynamically alters signal logic to ensure immediate clearance. The entire pipeline from image input to decision output, is modular and designed for real-time performance. The system demonstrates improved traffic throughput, reduced wait times, and effective handling of emergency scenarios, making it a scalable prototype for smart city applications.

Keywords: Intelligent Traffic Management, YOLOv8, Real-Time Vehicle Detection, Ambulance Priority System, Dynamic Signal Timing, Pygame Simulation, Emergency Response Optimization

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Project Title

AI-Driven Traffic Management System With Emergency Vehicle Prioritization.

1.2 Area

Artificial Intelligence, Computer Vision, Smart Transportation Systems, Embedded Systems, Intelligent Transport Infrastructure, Urban Mobility Solutions.

1.3 Project Introduction and Aim:

Modern urban environments are increasingly plagued by traffic congestion, inefficiencies in signal timing, and the critical challenge of ensuring timely passage for emergency vehicles such as ambulances and fire brigades. With rapid urbanization, the density of vehicles on roads has increased exponentially, making conventional traffic management systems obsolete and reactive. These outdated systems lack adaptability, are not dynamic, and function on static timers, regardless of real-time road conditions or traffic volume.

The aim of this project is to use the power of Artificial Intelligence (AI) and Computer Vision (CV) to create an autonomous, adaptive traffic management system that not only optimizes the traffic flow at signalized intersections but also prioritizes emergency vehicles like ambulances when detected within the vicinity of an intersection. By integrating real-time vehicle detection mechanisms, the system adjusts traffic light timing dynamically, reduces congestion, and provides a clear path for ambulances, thus potentially saving critical minutes during emergencies.

The project introduces a smart simulation-based model that incorporates the following:

- Detection of vehicle density using computer vision techniques.
- Calculation of green signal duration based on the estimated time required for all vehicles to cross.

- Real-time ambulance detection and priority handling, overriding standard signal logic to ensure clear passage.
- Modular design enabling potential integration into real-world smart traffic infrastructure or IoT-based smart city platforms.

This project was developed not just as a technical showcase but also as a solution-driven initiative with real-world applications. It attempts to bridge the gap between theoretical AI models and practical urban problems, proposing a deployable architecture for next-generation intelligent transport systems.

1.4 Need for the Project

- Emergency Response Delay: In urban environments, ambulances often face delays at traffic signals, sometimes stuck behind long queues of vehicles, which can directly impact patient survival outcomes.
- Inefficiency of Static Signals: Conventional signal timing mechanisms do not consider real-time traffic conditions or vehicle density, resulting in inefficient green time allocation.
- Lack of Emergency Prioritization: Most existing traffic systems do not have mechanisms to detect and respond to emergency vehicles dynamically.
- Scalability and Modularity: There’s a strong need for intelligent, modular, and easily adaptable traffic systems that can work independently or integrate with broader smart city ecosystems.

1.5 Overview of Implementation

The project employs a simulated traffic intersection where multiple vehicles are spawned from four directions. The vehicle density is captured using frame analysis, where each vehicle’s presence and movement are tracked using bounding boxes and label data. The AI system computes the estimated time for each vehicle to cross the signal and dynamically allocates green time accordingly. Ambulance detection is incorporated via a class-detection model that overrides the green signal rotation when an ambulance is detected on a non-green signal road.

Key technologies and concepts involved:

- **Computer Vision (OpenCV)** for object detection and spatial calculations.
- **Deep Learning-based YOLO model** for vehicle classification and ambulance identification.
- **Python Simulation Framework** to emulate intersection logic, vehicle spawning, and signal management.
- **Real-Time Signal Adjustment** using data-driven logic based on detected traffic.
- **Ambulance Priority Logic** using global flags and overridden cycle flow.

1.6 Application of the Project

- Smart Cities: Seamless integration with smart traffic infrastructure.
- Urban Traffic Control Systems: Retrofit capability for intelligent traffic controllers.
- Emergency Response Optimization: Helps improve ambulance travel time and reduce mortality rates.
- Simulation-Based Testing Environments: Useful for traffic researchers, urban planners, and AI-based mobility solution testers.

Chapter 2
Literature Survey

Traffic management has been a perennial challenge in urban environments, particularly with increasing congestion and the need for efficient emergency response systems.Recent innovations in machine learning (ML), artificial intelligence (AI), and computer vision have been found to hold great potential in solving these issues, especially with the adoption of real-time adaptive traffic control systems. This literature survey explores key contributions, limitations, and future directions in AI-based traffic management systems, drawing from various methodologies, technologies, and case studies.

2.1 Earlier Work and Contributions

The use of AI in traffic signal control systems can be traced back to efforts to improve static timer-based traffic controls. Early efforts often relied on pre-programmed signal timing, where fixed cycles were used to allocate green, yellow, and red light durations at intersections. These systems were designed to handle average traffic conditions but proved ineffective in fluctuating traffic volumes.

- Traditional Approaches: Previous systems tended to rely on fixed signal timing or basic detectors like infrared (IR) sensors, inductive loops, and pressure pads to sense the presence of a vehicle. However, these methods failed to adjust dynamically to real-time traffic conditions. Ghazal (2016), for example, highlighted a basic model using IR sensors to dynamically adjust green light durations based on traffic volume, but the limitations of this approach became clear over time due to insufficient accuracy and lack of real-time adaptability.
- Use of Video and Image Processing: As technology progressed, systems incorporating video cameras and image processing began to emerge. Anurag Kanungo et al. (2014) introduced an approach that used video cameras and image subtraction techniques to calculate real-time traffic density. This system offered a marked improvement over earlier sensor-based methods by capturing live data from traffic junctions, but it still relied heavily on camera technology

and faced issues in low-light conditions and non-vehicle traffic (e.g., pedestrians and cyclists).

- AI and Machine Learning Integration: As AI developed, computer vision and deep learning algorithms like YOLO (You Only Look Once) started to become a key part of real-time detection and tracking of vehicles. Mihir M. Gandhi (2020) used YOLO to dynamically adjust traffic signals based on vehicle density, with a 23% improvement in vehicle throughput over static systems. This brought about more intelligent traffic management, although the dependency on camera-dependent detection raised challenges concerning privacy, real-world tests, and datasets.
- Edge Computing and V2X: The most recent contributions have incorporated edge computing and Vehicle-to-Everything (V2X) communication. Baoming Wang et al. (2024) emphasized the role of edge computing to reduce latency in traffic data processing and improve real-time decision-making. Similarly, Wei-Hsun Lee and Chi-Yi Chiu (2020) explored the integration of V2X communication in their Smart Traffic Signal Control (STSC) system, enabling Emergency Vehicle Signal Preemption (EVSP) and Public Transport Signal Priority (TSP), which helped reduce congestion and improved emergency response times.

2.2 Limitations and Challenges

While significant strides have been made in AI-based traffic management, several challenges persist in improving the effectiveness and scalability of these systems.

- Real-Time Processing and Scalability: One of the significant drawbacks in previous and even current systems is the issue of scalability and real-time data processing. Most conventional systems have usually faltered when it comes to processing vast amounts of real-time data from various inputs (cameras, sensors, etc.). As an example, Baoming Wang et al. (2024) had written about reducing latency through edge computing, but even the latest models usually falter when scaling across greater city spaces. In addition, cybersecurity issues of V2X communication may threaten system integrity, as Wei-Hsun Lee and Chi-Yi Chiu (2020) noted.

- Limited Integration of Non-Vehicle Traffic: Another significant gap in the current research is the lack of consideration for pedestrians and non-motorized vehicles such as bicycles. Kanungo et al. (2014) noted that their system focused only on vehicles, neglecting pedestrian and cyclist movement. This limitation makes such systems less applicable for urban traffic management where these user groups contribute significantly to traffic dynamics.
- Dependence on Specific Sensor Technologies: A variety of systems remain limited by their reliance on specific technologies like cameras, infrared sensors, or RFIDs. For example, Ghazal (2016) noted that while their system used XBee transceivers for wireless communication, this reliance on a single technology could introduce vulnerabilities such as interference or limited communication range. Similarly, the video-based systems of Kanungo et al. (2014) faced challenges in poor lighting conditions, which remains a common issue for vision-based detection systems.
- Limited Emergency and Multi-Intersection Coordination: Several systems, including Mihir M. Gandhi (2020) and Khalid M. Almuraykhi (2019), have explored emergency vehicle prioritization. However, issues related to multi-intersection synchronization and dynamic rerouting for emergency vehicles remain unresolved. Almuraykhi (2019) pointed out the absence of autonomous traffic flow management for emergencies in certain systems, highlighting the need for integrated city-wide solutions rather than isolated intersection control.

2.3 Proposed Solutions and Future Directions

Several promising solutions have been proposed to address the limitations of current AI-based traffic management systems:

- AI-Driven Multi-Intersection Optimization: Recent research by Qi Jin (2024) used Deep Q-Learning and Graph Convolutional Networks (GCN) to handle multi-intersection optimization in real-time. This approach not only improved traffic flow but also demonstrated potential for adaptive behavioral spatial-temporal light (ABSTLight) control to ensure that green waves are dynamically adjusted to minimize traffic congestion. Future developments could focus on refining this method for pedestrian prioritization and adaptive route planning for emergency vehicles.

- Reinforcement Learning for Dynamic Signal Control: Mihir M. Gandhi (2020) demonstrated the ability of AI to adapt signal timings dynamically based on traffic density. This is further leveraged in reinforcement learning (RL) and Markov decision processes (MDPs) for optimizing signal timing adjustments, anticipating congestion proactively, and making effective use of real-time traffic information. Multi-modal traffic can be explored further in future research, with coordination among vehicles, pedestrians, and public transport.
- Edge Computing and IoT Integration: As discussed in Baoming Wang et al. (2024), incorporating edge computing allows for local processing of real-time data from sensors and cameras, significantly reducing latency. Additionally, the combination of IoT sensors, AI algorithms, and V2X communication holds potential for predictive traffic control and seamless multi-intersection coordination, offering scalability and flexibility for urban environments.
- Addressing Ethical AI and Privacy Concerns: While ethical AI is a relatively new focus in the field, it is increasingly critical. As noted by Prajwal V. et al. (2023), AI systems need to be designed with careful attention to privacy and bias reduction. Future AI systems should incorporate privacy-preserving technologies, such as encryption, and ensure transparent algorithms to prevent discrimination and bias in traffic decision-making.
- Smart City Integration: The integration of these systems into broader smart city frameworks will be key to optimizing urban infrastructure. Wei-Hsun Lee et al. (2020) suggested expanding these models to work in conjunction with smart city IoT systems, enabling real-time coordination across multiple infrastructure layers, including waste management, parking systems, and environmental monitoring.

2.4 Conclusion

The integration of AI in traffic management systems has come a long way, with advancements in machine learning, computer vision, and real-time data processing. Despite the progress, key challenges remain, particularly in terms of scalability, real-time processing, multi-modal traffic coordination, and ethical concerns. As future systems evolve, the focus will likely shift toward more integrated, city-wide solutions that incorporate a wider range of traffic participants, smarter real-time decision-making, and scalable edge-computing solutions. The field promises significant improvements in urban mobility and emergency response, but more research is needed to overcome existing barriers and fully realize the potential of AI-driven traffic management systems.

Chapter 3
Problem Statement

3.1 Project Scope

The scope of this project, titled "AI-Driven Traffic Management System With Emergency Vehicle Prioritization ", is to design, develop, and simulate a smart traffic control system that adapts to real-time road conditions using AI and computer vision techniques. This includes intelligent vehicle detection, dynamic signal timing adjustment, and real-time prioritization of ambulances within urban traffic environments.

The project focuses on solving two critical aspects of urban traffic:

- **Vehicle Density Optimization:** Dynamic traffic light control based on real-time vehicle density estimation to reduce congestion, waiting time, and fuel consumption.
- **Emergency Vehicle Prioritization:** Real-time detection and prioritization of ambulances using camera feeds and AI models, ensuring they get green signals at the right time to avoid delays.

The system will simulate a standard four-way traffic intersection, integrated with vehicle detection modules and logic for signal switching, and will include a mechanism to override standard traffic flow when an ambulance is detected.
This project does not just stop at detection; it actively intervenes in signal flow using learned information from real-time inputs. It is a step toward fully autonomous traffic regulation systems that can reduce human reliance and enhance urban mobility efficiency.

3.2 Project Assumptions

To maintain a focused and achievable development cycle, the following assumptions are made:

- **Clear Camera Feeds**: The system assumes consistent visibility and clarity of traffic camera feeds for effective vehicle and ambulance detection using computer vision models.
- **Detection Accuracy**: It is assumed that the trained detection model (YOLO) can reliably detect vehicles and distinguish ambulances with a high degree of accuracy under various lighting and weather conditions.
- **Intersection Scope**: The simulation and system logic are built on a typical four-directional intersection with fixed entry and exit lanes. Scalability to more complex junctions is outside the current development scope.
- **One Ambulance at a Time**: The logic assumes that only one ambulance approaches the intersection at a time. Multiple concurrent emergency vehicles are not considered in the current implementation.
- **No Interference in Communication**: It is assumed that all system components (detection, control logic, simulation) communicate seamlessly without hardware or software interruptions.
- **Real-Time Execution**: The system is assumed to run with minimal latency, meaning detection and decision-making should occur fast enough to be effective in dynamic traffic.

3.3 Project Limitations

While the proposed AI-driven traffic management system marks a significant improvement over traditional static signal control approaches, it is essential to acknowledge several inherent limitations in its current design and implementation. These limitations are not oversights, but rather reflections of the practical boundaries imposed by the project's scope, resources, and academic context. Understanding these constraints is crucial for situating the contributions of the project realistically and for guiding future enhancements and real-world applicability.

1. **Simulation-Based Implementation**

   The entire system has been conceptualized, developed, and tested within a controlled simulation environment using Pygame. While simulation offers a powerful platform for proof-of-concept validation and rapid iteration, it lacks the unpredictability and complexity of real-world urban environments. Key factors such as:

- Sensor noise,
- Irregular or obstructed camera angles,
- Weather-related visual disturbances (rain, fog, glare),
- Night-time or low-light visibility,
- Non-standard traffic behaviors by human drivers or pedestrians

are either idealized or entirely absent in the current setup. These variables can significantly affect system accuracy and responsiveness during real-world deployment, potentially leading to suboptimal or even hazardous decision-making if not adequately accounted for.

2. Limited Emergency Vehicle Handling

At present, the project's emergency handling mechanism is tailored specifically for ambulances. It relies on visual detection algorithms that identify ambulances based on their appearance in the camera feed. However, this excludes other critical emergency response units such as fire trucks and police vehicles, each of which may differ in size, shape, color, and operational priority. Additionally, the system does not yet accommodate variations in siren signals or flashing lights that are often indicative of emergency presence and urgency. As a result, the solution lacks comprehensive emergency vehicle prioritization.

3. Simplified Intersection Modeling

The current implementation models only standard four-way intersections, which limits the system's generalizability to more complex real-world traffic layouts. It does not yet support:

- Multi-lane or multi-leg intersections,
- T-junctions or staggered crossings,
- Roundabouts and flyovers,
- Intersections with turn-only lanes or reversible lanes.

In real cities, these variations are common and often critical to traffic flow, especially during peak hours or special events. Limiting the simulation to uniform intersections reduces the accuracy of performance evaluations and constrains the adaptability of the system to diverse urban settings.

4. **Training Data Constraints for Vision Models**

The ambulance detection model used in this system has been trained on a moderately sized dataset, containing a limited variety of images. While initial accuracy rates may be promising in the controlled test environment, the model's generalizability and robustness are potentially compromised by the lack of exposure to:

- Diverse ambulance designs across different regions or countries,
- Occlusions by other vehicles or infrastructure,
- Varying lighting and weather conditions,
- Partial visibility (e.g., only part of the vehicle is visible).

In the absence of a large-scale, annotated, and geographically diverse dataset, the model's performance in real-world scenarios may degrade significantly, leading to misclassifications or failure to detect emergency vehicles in time.

5. **Absence of V2X Communication Infrastructure**

Vehicle-to-Everything (V2X) communication is a critical component in the future of intelligent transportation systems, enabling direct communication between vehicles and traffic infrastructure (e.g., traffic lights, road signs). However, the current project does not integrate any V2X capabilities. As a result:

- The system lacks proactive awareness of approaching vehicles.
- Emergency vehicles cannot transmit their presence or route in advance.
- There is no real-time data exchange between neighboring intersections for cooperative signal planning.
- Incorporating V2X would not only enhance detection accuracy but also

significantly reduce system latency and improve coordination across the traffic network.

6. No GPS or Navigation System Integration

The project currently relies solely on vision-based methods for detecting ambulances and assessing their proximity to intersections. However, modern emergency fleets are equipped with GPS systems that can provide:

- Precise location data,
- Real-time estimated time of arrival (ETA),
- Optimal routing information.

The absence of GPS integration limits the system's ability to plan ahead or prioritize intersections dynamically based on the vehicle’s actual trajectory. Vision-based detection is reactive, and while effective within line-of-sight, it may not offer sufficient lead time for preemptive signal adjustment.

7. Lack of Manual Control or Override Mechanisms

Another significant limitation is the absence of manual override capabilities for traffic authorities. In many real-world scenarios such as parades, natural disasters, or signal malfunctions, traffic police officers may need to override the automated system to impose direct control. Without a provision for manual intervention:

- The system becomes a closed loop, potentially unresponsive to contextual human judgment.
- Operators cannot intervene in the case of erroneous decisions or detection failures.
- It may be considered unreliable or inflexible by real-world traffic control authorities.

A hybrid model that allows automated AI operation alongside manual override would be more acceptable for public deployment.

8. **Scalability Challenges and Computational Constraints**

   As the system scales to include more intersections or integrates more sophisticated AI models (e.g., deep reinforcement learning or multi-agent systems), the computational complexity and resource demands will rise significantly. Real-time decision-making under such load requires:

   - High-performance computing hardware,
   - Optimized model architectures,
   - Efficient communication protocols.

   Given the academic and simulation-based nature of the project, such scalability issues are not addressed in full, limiting the direct transferability of this solution to large-scale urban environments.

3.4 Project Objectives

The principal aims of the project are as under:

1. **Simulate a real-world intersection and traffic flow**
   A realistic traffic intersection is simulated using Python libraries like Pygame, enabling controlled testing of dynamic traffic behaviors and signal logic under varying conditions.
2. **Detect vehicles using AI-based computer vision**
   Computer vision models, such as YOLO, are employed to identify and count vehicles from live or simulated camera feeds, ensuring real-time traffic density estimation.
3. **Calculate dynamic green signal timing**
   Signal durations are dynamically adjusted based on detected vehicle count and type, incorporating average crossing times to improve traffic flow efficiency.

4. **Prioritize ambulances using a trained detection model**

   A dedicated AI model identifies ambulances and overrides the current signal sequence, ensuring immediate green signal allocation to reduce emergency vehicle delays.

5. **Visualize system functioning via graphical simulation**

   A live simulation interface displays traffic signals, vehicle movement, and emergency handling, offering an intuitive visualization of system decisions and transitions.

6. **Validate logic through edge-case testing**

   The system is rigorously tested under diverse scenarios, such as sparse traffic, congestion, or mid-cycle ambulance arrival, to confirm its adaptive and reliable behavior.

7. **Offer a modular and extensible framework**

   The project is designed for scalability, allowing future integration of additional vehicle types, communication technologies like GPS/V2X, and edge hardware deployment.

8. **Reduce ambulance response times and support smart mobility**

   By prioritizing ambulances and optimizing traffic flow, the system aims to lower emergency response times and enhance urban mobility with life-saving potential.

Chapter 4
Project Requirements

4.1 Resources

The execution of this project involves coordinated contributions across technical, software, and hardware domains. The resources are categorized into Human Resources, Reusable Software Components, and Software & Hardware Requirements, as outlined below:

4.1.1 Human Resources

This project is executed by a team consisting of four technical contributors with the guidance of an academic guide. Each member is responsible for a critical component of the overall system:

- Prakhar Pachauri – Vehicle Detection Module
  Responsible for the development and integration of the vehicle detection model along with usage of YOLO v8, including dataset collection, annotations for images and training using YOLO frameworks. Also handles the design of algorithms that differentiate between vehicle types eg. - cars, buses, bikes, trucks, and ambulances.
- Tanvi Khadap – Traffic Density Calculation module
  In charge of computing real-time traffic density from detection outputs. Works closely with Prakhar to translate vehicle counts into a meaningful metric for decision-making. Also contributes to dynamic green-time calculation algorithms based on per-lane vehicle volume and prioritization logic.
- Mehul Vaidya – Traffic Light Control Module
  Oversees the core logic responsible for controlling traffic lights dynamically. This includes signal phase management, time-based switching, emergency overrides for ambulances, and integration with the simulated environment. Ensures the logic is modular, maintainable,

and synchronized with detection inputs.

- Riya Gupta – Simulation and Visualization Module
  Develops the real-time visual interface using Pygame. Handles the graphical representation of vehicles, signals, intersections, and live updates during simulation. Coordinates the animation and interaction flow to ensure it accurately reflects backend logic and decision-making.
- Prof. Pramod Mali – Project Guide
  Provided continuous feedback and direction regarding research quality, feasibility. Helped validate literature survey depth, soundness of methodology, and relevance of final output from an academic perspective.

4.1.2 Software Requirements

- Libraries and Frameworks
  - OpenCV (v4.8+) – For image and video frame capture, processing, and display.
  - Pygame (v2.3+) – For graphical simulation of intersection, vehicles, and signal transitions.

- Operating System
  - Windows 10/11 or Ubuntu 20.04+ – Recommended for development, with compatibility for camera and GPU drivers.

- Development Environment
  - Python 3.8+.

4.1.3 Hardware Requirements

- Training Machine
  - Processor: Intel i5 / AMD Ryzen 5 or highe
  - RAM: At least 16 GB to manage large training data sets
  - Storage: SSD with minimum 512 GB for quicker read/write of data sets and simulation files

4.2 Requirements Rationale

The table below outlines specific project requirements along with a clear rationale explaining why each is necessary for the system's success:

Requirements	Rationale
Real-time vehicle detection using computer vision	Enables accurate, lane-wise traffic measurement and dynamic signal allocation, forming the backbone of the intelligent control system.
Separate ambulance detection model	Ensures life-critical response handling by providing emergency vehicle prioritization in the traffic signal logic.
Dynamic signal timing based on traffic density	Improves flow efficiency by avoiding unnecessary waiting at low-density lanes and reduces congestion overall.
Simulation environment using Pygame	Allows for visual feedback and logic validation in a safe, controllable virtual setting before considering real-world deployment.
Modular component structure	Supports maintainability, easier debugging, and future extensibility (e.g., integrating GPS, V2X, or more emergency types).
Python-based stack with OpenCV and TensorFlow	Facilitates rapid prototyping using widely-supported libraries and uses GPU acceleration for detection inference.

Table 4.1: Requirements and Rationale

4.3 Risk Management

This section identifies key risk factors that could affect the project during various phases and categorizes their severity for targeted mitigation:

Risk Factor	Risk Level	Description & Justification
Inadequate Dataset for Ambulance Detection	High	The model's accuracy depends heavily on diverse and annotated ambulance images. A small or skewed dataset can result in misclassification, especially in real-world noisy scenes.
Real-time performance bottlenecks	High	Detection and signal updates must run quickly. If the model or simulation lags, it may affect decision timing and system validity. Optimization and profiling are essential.
Limited model generalizability	Medium	A model trained in one city may not generalize well to traffic visuals from another. Lighting, road markings, and vehicle types vary widely.
Hardware resource limitations	Medium	Especially during training phases, inadequate RAM or lack of GPU can drastically increase training time or make inference impractical.
Synchronization errors in traffic logic	Medium	Bugs in signal switching (e.g., multiple greens or missed transitions) could invalidate test results. Requires careful state management and testing.
External dependencies failure (library bugs, version conflicts)	Low	Using open-source packages means updates or deprecations may introduce compatibility issues or bugs. Proper version control helps mitigate this.
Simulator inaccuracies	Low	The simulation may not capture physical realities (e.g., turning radius, braking lag), limiting fidelity of results. However, it’s acceptable for

		logic validation.

Table 4.2: Project Risk Factors

4.4 Functional Specifications

4.4.1 Interfaces

Interfaces are critical in a modular and scalable system like this traffic management solution. The system integrates multiple independent yet cooperating components, including computer vision models, a simulation engine, a traffic signal controller, and an ambulance priority handler. Interfaces define how these components interact and exchange data.

1. **External Interfaces**

- Camera Feed Input Interface

  Accepts real-time or simulated traffic images or video from street surveillance cameras. The input is piped into the vehicle detection model.
  - Type: Stream / File
  - Format: JPEG/PNG or video (.mp4, .avi)
  - Purpose: Continuous supply of traffic frames for processing.

2. **Internal Interfaces**

   Internal interfaces handle communication between modules:

- Vehicle Detection Module ↔ Signal Timer Logic
- Data passed: Vehicle counts per direction, ambulance presence (boolean), detection confidence levels.
  - Trigger: Every processed frame or set time interval
  - Method: Function call or shared memory structure
- Ambulance Detection ↔ Priority Override Logic
  - Once an ambulance is detected, the priority override function communicates with the main signal controller to update the state machine to green in the ambulance's direction.

3. **Graphical User Interfaces (GUI)**

   Simulation Display Interface (Built using Pygame)

Shows the current state of all roads: vehicle movement, signal color, timers, and ambulances.

4.4.2 Interactions

While end-users like traffic police or civilians won't directly interact with the core system, from a usability standpoint, interactions must be seamless and reliable.

- Operator Starts the Simulation or System
    - The system initializes all modules: traffic simulation, detection model, signal controller.
- Vehicles and Ambulances Enter Simulation
    - Each frame is processed by the detection module.
    - Ambulance detection triggers a flag to the priority logic.
    - Timer controller recalculates green signal durations dynamically.
    - If an ambulance is detected, the corresponding signal is turned green immediately, overriding others.
- Traffic Signal Changes Based on Vehicle Count
    - Each lane's uncrossed vehicles are counted.
    - Total estimated crossing time is calculated.
    - Based on this, signal timers adjust dynamically, ensuring smoother flow.
- Feedback Mechanisms
    - Visual cues in the GUI reflect decisions.
    - Optional logs export can feed into analytics or operator review.

4.4.3 Sustainability

To ensure long-term sustainability of the project:

- Modular Architecture
    - Every module (vision, simulation, control logic) is independent and loosely coupled.
    - Future modules like pollution detection, speed control, accident prediction can be added easily.

- Dataset Reusability
    - The vehicle and ambulance datasets are reusable across multiple urban planning and smart city systems.
- Adaptable to Real-Time Input
    - With minor changes, the system can shift from simulation to live deployment using RTSP feeds.
- Scalable Architecture
    - Can support more intersections, real-time data fusion, and integration with GIS/IoT platforms.

4.4.4 Quality Management

Robustness and correctness are essential:

- Code Quality
    - Adheres to modular design principles.
    - Followed PEP8 guidelines (Python) and used proper exception handling.
- Testing and Validation
    - Each module (vehicle detection, signal control) was tested in isolation and integration.
    - Functional testing ensured ambulance override worked under all timing windows.
- Performance Logging
    - Decision logs (detection time, decision taken) are saved to allow later analysis and validation.
- Version Control and CI/CD Ready
    - The project is maintained on GitHub with commit history.
    - Suitable for integration with automated testing tools in future.

4.4.5 Security

Security is less critical in simulation mode but essential for real deployment:

- Input Validation
    - All external input must be sanitized to avoid runtime errors or

injections.

- Data Integrity
    - Logs and event records must be stored in tamper-proof formats.
- Safe Failovers
    - In case of project failure, the system defaults to traditional timer-based control to avoid accidents.

Chapter 5
System Analysis Proposed Architecture

5.1 Design Considerations

In designing a smart traffic light system that responds dynamically to the traffic volume and gives way to emergency vehicles, some key considerations have been accounted for. These encompass technical, functional, and non-functional aspects, ensuring that the system is not only effective in theory but also adaptable and scalable in practice. The following discussion elaborates on the rationale behind key architectural and implementation decisions:

1. **Real-Time Vehicle Detection as a Core System Requirement**

   Real-time vehicle detection forms the backbone of the project. The optimization of traffic signal timing from the actual vehicle count on each lane would be ineffective without real-time and instantaneous detection of vehicular presence and category. Real-time detection enables the system to respond to these changes in real time, so green light intervals are exactly matched to current demand. Without this, the traffic control logic would either rely on outdated data.

2. **Modular Architecture vs. Monolithic Design**

   The system is divided into distinct functional blocks, such as:

   - Vehicle detection module
   - Density calculation module
   - Traffic signal control logic
   - Ambulance override mechanism
   - Simulation engine

   This modular approach ensures independent development and debugging. Each team member can work on their assigned module without interference. Also, modularity supports reusability and scalability. For example - the ambulance detection module can later be adapted to detect other emergency vehicles like fire trucks.

3. **Simulation-Based Validation**

Implementing and testing the system in a real-world traffic environment is not feasible during development due to constraints. Therefore, the decision to use a simulation environment built with Pygame was made.

4. **Emergency Vehicle Prioritization**

   Emergency response prioritization is an important design aspect, justified by the need to reduce the time of ambulance arrival. It is for this reason that the addition of a second AI model for ambulance detection is provided.

   The inclusion of emergency detection in a standard vehicle model would somehow sacrifice accuracy or add complexity and therefore the system uses a dedicated model trained to identify ambulances. This decoupling of focus enables quicker inference and enhances accuracy.

5.2 Assumptions and Dependencies

1. **System-Level Assumptions:**
   - **Operational and Properly Positioned Cameras:** The model anticipates that each camera at an intersection is working, properly tuned, and oriented so that each receives a high-quality, clear, unobstructed, top-down, or angled shot of the lanes on the road. Non-functioning or wrongly positioned camera feeds can significantly degrade detection accuracy and system performance.
   - **Adequate Light Conditions:** It is assumed that there is adequate ambient light to assure good video capture. Daylight and well-lit city street environments are present, but illumination with poor lights, dense fog, rain, or nighttime runs is not considered part of this simulation. Some future extensions can include night-vision or infrared technologies.
   - **Clear Sight of Ambulances:** The framework requires ambulances to remain within the camera's field of sight and not be obstructed by big objects like buses or trucks. The model assumes partial or complete sight of features unique to ambulances, like the red cross, siren lights, or text signs, in order to make correct classification.
   - **Expected Intersection Configurations:** Only traditional four-way intersections have been considered by the simulation. These intersections are

assumed to be symmetrical and identical, with established and fixed traffic flow patterns. Roundabouts or multi-entry overpasses have not been included in the range of implementations yet.

- **Single Ambulance Conflict Resolution:** The model assumes that, at any instant, there is a single ambulance entering the intersection. Simultaneous ambulance arrivals from multiple directions are not modeled and might need more sophisticated prioritization logic. Synchronized Frame Input: It is assumed that all the camera feeds are frame-synchronized, such that the detection algorithm is provided with stable and sequential data for robust inference.

2. Software Dependencies

- The project relies on certain software libraries, frameworks, and pre-trained models to facilitate the implementation of functional modules. They are as follows:
- OpenCV, or Open Source Computer Vision Library, is being used extensively to capture video frames, pre-process images, track objects, and perform image transformation operations. It acts as the underlying mechanism for handling computer vision operations.
- YOLO Framework: The system utilizes YOLO (You Only Look Once), an object detection system in real time. It allows the system to recognize various types of vehicles and ambulances with high-speed inference.
- Pygame: The graphical simulation and visual interface of the system are achieved via Pygame. It allows real-time rendering of signal state, road features, vehicle movement, and dynamic updates mimicking real traffic flow.
- TensorFlow: Needed by YOLO to carry out neural network-based inference during detection. TensorFlow can support GPU acceleration, so it can do multiple detections at once.

5.3 System Architecture

The system's architectural design is composed of four main modules that collaborate to simulate an advanced traffic management system. Each module performs a specific task while communicating via pre-defined data exchanges and event triggers.

1. Key Elements
    - Detection Module

      This module governs real-time vehicle classification and detection based on pre-trained AI models. It handles video frames from all lanes to count vehicles, identify vehicle types (car, truck, bus, bike), and identify ambulances for priority processing.
    - Simulation Framework

      Developed using Pygame, this module displays the graphical setup of the intersection, including vehicles, traffic lights, and motion dynamics. It displays the system status in real-time, as per the decisions taken by the control logic.
    - Signal Control Logic This module performs real-time computation of green light times depending on the type and density of vehicles. Moreover, it overrides traditional timing procedures when an ambulance is present to allow early transit.
    - Priority Handler

      This logic level manages emergency overrides, decides which lane an ambulance is approaching, and coordinates with the signal control to prefer that direction without cutting down overall flow too much.
2. Data Pipelines and Event Triggers
    - Input Stream: Video stream → Frame extraction → Object detection → Density and priority data
    - Event Triggers: Vehicle counts updates, ambulance detection, and timer expirations trigger re-evaluation of signal state.
    - Control Loop: Every iteration, the simulation engine is refreshed by vision module data, thus resulting in the control logic being calculated and signal state changes.

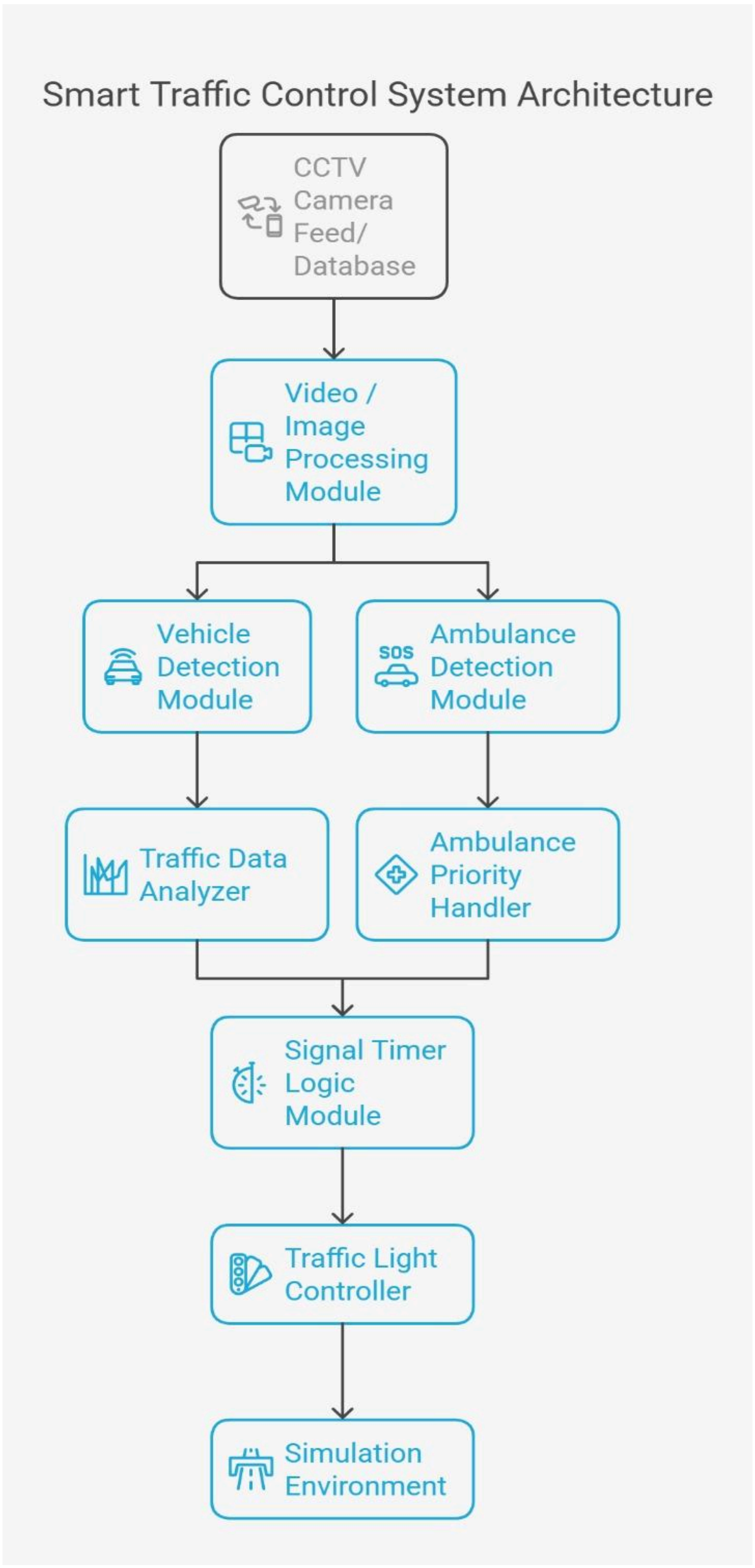

Smart Traffic Control System Architecture
CCTV Camera Feed/ Database
Video / Image Processing Module
Vehicle Detection Module
SOS
Ambulance Detection Module
Traffic Data Analyzer
Ambulance Priority Handler
Signal Timer Logic Module
Traffic Light Controller
Simulation Environment

Figure 5.1: System Architecture

5.4 Modules of the project

1. **Vehicle Detection Module**

   This module uses AI-based computer vision models (YOLOv8) to detect and classify vehicles in real-time. It processes individual frames from each road to count the number and types of vehicles, those being - cars, trucks, buses, bikes, to detect ambulances.

   Key Features:

   - Frame extraction from video feeds
   - Object detection and bounding box classification
   - Ambulance detection using a specialized trained model
   - Outputs passed to the density calculator and priority handler

   Technologies Used:

   - OpenCV, Python, pretrained YOLO models

2. **Density Calculation and Analysis Module**

   This module receives vehicle detection data and computes vehicle density per lane. It also estimates traffic weight by factoring in the type of each vehicle to better approximate the time required for each lane to clear.

   Key Features:

   - Per-lane vehicle count analysis
   - Traffic weight scoring system (car < truck < bus)
   - Handling edge cases such as empty lanes or equal densities
   - Interfaces with signal control logic for green time computation

   Technologies Used:

   - NumPy, Custom scoring algorithms, Python

3. **Traffic Signal Control Module**

   This module controls the dynamic behavior of traffic lights based on density data. It handles time-based signal transitions and integrates ambulance prioritization logic when necessary.

   Key Features:

   - Timer logic for adaptive signal switching
   - Dynamic green time calculation

- Ambulance override capability
- Emergency-safe fallback if input is invalid

Technologies Used:

- Python logic layer, conditional flow statements

4. Simulation and Visualization Module

This module simulates a real-time traffic intersection with dynamic visuals. Built using Pygame, it reflects all operational changes such as signal switching, vehicle movement, and ambulance overrides.

Key Features:

- Graphical rendering of intersection and lanes
- Animated vehicles with directional movement
- Signal display with timer countdowns

Visual feedback for ambulance detection and emergency response

Technologies Used:

- Pygame, Python, coordinate-based vehicle placement

5. Priority Handling and Override Logic

This sub-module processes ambulance detection results and activates a temporary override in the traffic logic. It ensures safe and immediate passage of the emergency vehicle while managing system state transitions.

Key Features:

- Real-time signal override upon ambulance detection
- Blocking of non-priority lanes
- Post-passage reversion to standard timing logic

Chapter 6

Project Plan

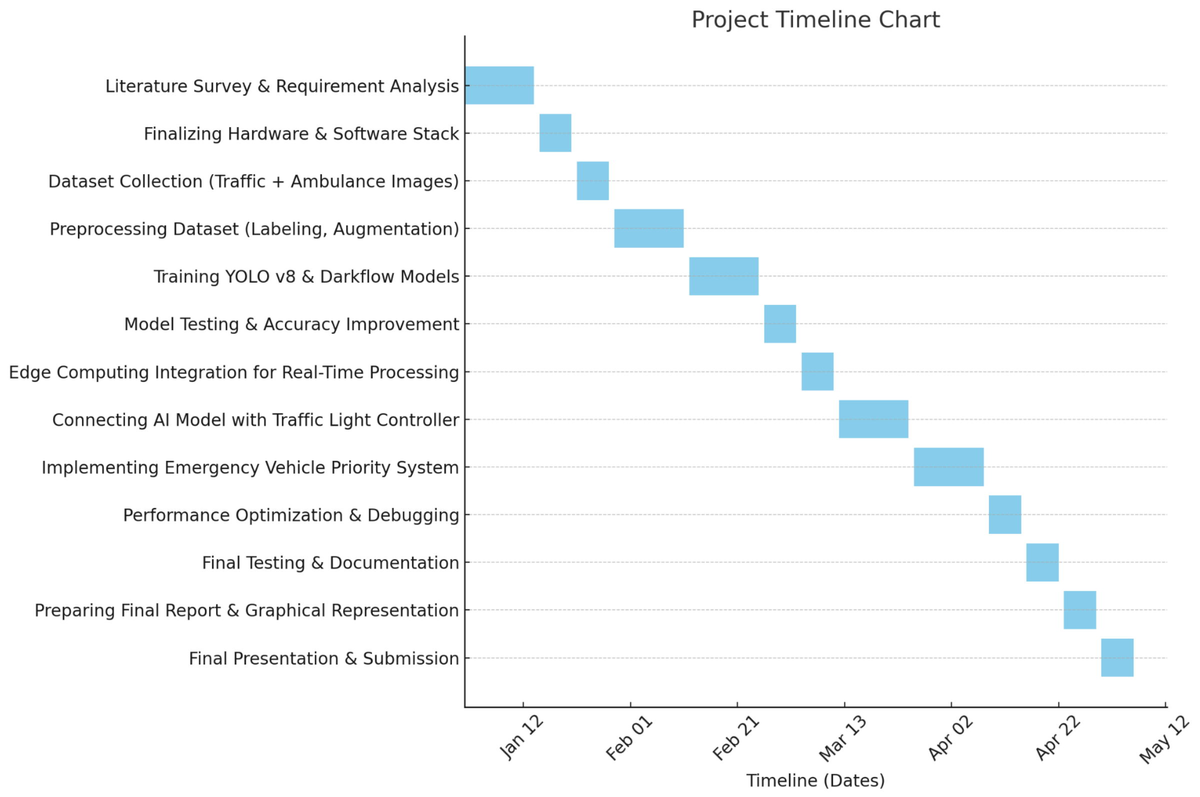


Figure 6.1: Project Timeline

Chapter 7
Implementation

7.1 Methodology

The design philosophy adopted an organized, phase-wise process with the objective of maintaining modularity, optimization, and validity for actual situations. The entire system's workflow can generally be categorized under the following steps:

1. **Problem Understanding & Planning:**
    - Conducted a survey of literature to review existing solutions and identify their limitations.
    - Setting down project objectives such as real-time traffic control and ambulance priority.

2. **Simulation Setup:**
    - Designed a graphical intersection using Pygame to mimic an actual four-way traffic light.
    - Designed the simulation to simulate real-time vehicle traffic flow and traffic light response.

3. **Vehicle Detection Module:**
    - YOLOv8 model for object detection.
    - Applied image input pipeline to constantly process traffic camera frames.
    - Types of vehicles (car, bus, bike, truck) are identified by their coordinates.

4. **Ambulance Detection & Integration:**
    - Trained a dedicated ambulance detection model.
    - Ambulance arrival causes overriding of the present signal cycle immediately.

5. **Dynamic Signal Control Logic:**
    - Depending on the kind and number of vehicles, dynamic signal timing is determined.
    - Green time is determined by the average time taken to cross by vehicle category.

6. **System Actuation and Decision:**
    - Inference from the detection model supplies signal timing algorithms.
    - Priority handler checks for ambulance presence.
    - Final decision alters signal state in the simulation in real-time.

7. **Assessment and Verification:**
    - Edge-case situations (low traffic, high traffic, arrival of an ambulance) were checked.
    - System behavior was verified against expected logic flow.

7.2 Algorithm

7.2.1 Object detection (YOLOv8)

- The light-weight YOLOv8n model was selected because it compromises between detection accuracy and inference speed in real-time scenarios.
- An image captured from a simulated or an actual traffic camera feed.
- Detected bounding boxes, class labels assigned (car, truck, bike, bus, ambulance), and confidence scores.
- Post-processing: Non-Maximum Suppression (NMS) eliminates duplicate bounding boxes and retains the most confident one for each class.

7.2.2 Vehicle Counting & Queue Length Estimation

- The vehicles are categorized and clustered according to the lane/direction they come in within the camera frame.
- The coordinates of each vehicle are inspected to determine its lane position.
- A theoretical queue length is computed as:

- Total Queue Length = ∑(vehicle_count * average_length_per_vehicle_type)
- This is the basis for traffic signal duration calculation.

7.2.3 Signal Timing Calculation Algorithm

- The system computes an individual green signal duration for each direction, depending on the traffic detected.
- Formula applied:
- green_time = base_time + ∑(vehicle_type_weight * vehicle_count)
- Weighting factors are based on mean crossing times for each vehicle type (e.g., bus > car > bike).
- All calculated times are normalized to keep total cycle time in limits.

7.2.4 Ambulance Override Logic

- If an ambulance is spotted in any lane:
  - The control logic overrides the existing traffic light cycle.
  - The ambulance's direction is granted immediate green light access.
  - All other directions are locked to red for now.
  - The system continues to do this until the ambulance leaves the intersection.

7.2.5 State Switching Mechanism

- A hybrid method is applied to signal phase switching:
  - Normal Conditions: Use a round-robin system with dynamic green time.
  - Emergency Conditions: Priority-based override where ambulance-bearing directions are favored over all others.
- System flags and state machines assist with managing active direction, next queue, and emergency lockout durations.

7.3 Other Implementation Details

- **Software Repositories:**
  - OpenCV: Capture of frames, image preprocessing, region masking.
  - Ultralytics: YOLOv8 model central inference engine.
  - Pygame: Real-time visualization and interactive simulation.
- **Utilities:**
  - Logging system tracks vehicle numbers, signal switch history, and override events.
  - Debugging overlays are shown immediately on the simulation canvas to offer visual feedback.
  - Temporary manual override buttons intended for testing non-standard usage scenarios.

7.4 Dataset

- **Source:**
  - Information gathered from a combination of publicly available traffic data sets and manually collected urban traffic clips.
  - Ambulance pictures gathered through web scraping and cell phone photos, subsequently edited for quality.
- **Statistics**
  - General vehicle dataset: 2000 images
  - Ambulance images: 400-500 images
- **Annotation:**
  - Annotated with LabelImg software.
  - Format employed: YOLO (txt files with class index and box coordinates).
- **Splitting:**
  - Dataset divided as 70% for training, 20% for validation, and 10% for testing.
  - Stratified sampling provided good class balance and diverse lighting/background conditions in each split.

Chapter 8
Performance Evaluation and Testing

The performance of an AI-driven traffic management system must be evaluated not merely in terms of its functional correctness, but also in terms of its operational efficiency, responsiveness, and impact on traffic flow and emergency prioritization. This section outlines the key metrics used to assess the system’s performance and provides an in-depth discussion of the outcomes observed during simulation-based evaluations.

8.1 Evaluation Criteria and Key Metrics

To thoroughly evaluate the impact of the proposed system, we have defined the following key metrics:

- **Vehicle Throughput:** The vehicles that successfully moved through the intersection in a specific time interval.
- **Dynamic Green-light Duration:** The ability of the system to dynamically adjust the duration of the green-light to accommodate the variable density of traffic.

Each of these metrics demonstrates how the proposed traffic management solution has the potential for realistic application and robustness.

1. Vehicle Throughput

The throughput of vehicles was used to assess the effectiveness of flow of traffic through the intersection. Under static signal control, vehicles can wait too long while not seeing a full traffic load on the other roadways, and this can lower efficiency of the intersection.

A net increase in throughput confirming that varying the time of the green light based on the count of vehicles improves the overall flow of traffic and lowers congestion.

2. Traffic Signal Adaptability

Another uniquely distinguishing feature of the proposed system is its real-time adaptability. The algorithm continuously assesses vehicle density on each road and applies an adaptation on the green signal length accordingly - using a weighted time model that distinguishes vehicle types.

As an example,

- For a 2-wheeler the weight is smaller than that for a 4-wheeler or heavy vehicle.
- The green time operates on the time required to move the vehicles in each category through the junction.

This model allows the road with the greater accumulation of vehicles to have more green time than a road that has less accumulation of vehicles, thereby allowing lower levels in the system to greater balance the levels of congestion in each direction.

Chapter 9
Deployment Strategies

9.1 Security Aspects

Before deploying the AI-driven traffic management system in a real-world environment, it is essential to ensure that the system is secure, reliable, and scalable. We have considered the following aspects:

- **Error Handling & Fail-safes**: The system includes fallback procedures in case of AI model failures or hardware issues. If the ambulance detection model fails or the video feed is lost, the system defaults to a predefined static signal plan.

- **Scalability**: The modular codebase allows the system to be deployed in multiple junctions. Cloud integration can also be explored in the future for centralized traffic control across a city.

- **Real-time Processing**: Multithreading ensures the system can handle simultaneous tasks like video processing and signal control efficiently without crashing or slowing down.

Chapter 10
Results and Analysis

10.1 Experiment Setup

The system was tested in an environment simulated to mimic a 4-way traffic intersection. The experiment was set up as follows:

1. **Traffic Density Variations**:
   The system was tested under three different traffic densities: low, moderate, and high. Traffic density was randomly controlled by adjusting the number of vehicles generated in each lane.

2. **Ambulance Prioritization**:
   Ambulances were introduced at random intervals, simulating emergency situations, to test the system's capability to prioritize the ambulance over regular traffic. The system's response time, the ability to clear the ambulance's path, and the handling of the signal switching were measured.

3. **Key Performance Metrics**: The following metrics were tracked and analyzed:
   - **Ambulance Response Time**: Time taken to switch the traffic signal to green in the direction of the approaching ambulance.
   - **Traffic Clearance Efficiency**: The number of vehicles cleared per signal cycle.
   - **Average Waiting Time per Vehicle**: Time a vehicle spends waiting at a red light before crossing.

10.2 Discussion of Results

The **ambulance prioritization** feature successfully interrupted the regular traffic Several key insights were gained from the experiment:

1. **Ambulance Prioritization**: The ambulance prioritization feature successfully detected the presence of ambulances and immediately interrupted the regular traffic signal cycle. The green signal was activated in the direction of the approaching ambulance within 1-2 seconds of detection, allowing for rapid emergency response. This ensured minimal delay in clearing the path for the ambulance.
2. **Dynamic Signal Control**: The dynamic signal control of the system varied signal lengths according to traffic volume.During peak traffic hours (high-density conditions), the system extended the green light duration to accommodate more vehicles and reduce waiting times. Conversely, during low-density periods, the green light duration was reduced to improve overall traffic flow and efficiency.
3. **Multithreading for Real-time Operations**: The implementation of multithreading significantly improved the responsiveness of the system. Real-time vehicle detection, signal switching, and emergency response were handled without lag. The use of threads allowed simultaneous vehicle generation, detection, and signal control, which optimized the system's efficiency.
4. **Signal Switching Behavior**: No issues related to signal hang or indefinite green light durations were observed after correcting the signal switching logic in the `repeat()` function. The system reliably switched signals at the appropriate time, ensuring fair distribution of green time across all directions.

10.3 Result Analysis

The system was evaluated based on the following performance criteria:

- **Ambulance Response Time**: The time taken to clear the path for an ambulance once it was detected was measured to be within 1-2 seconds. The system effectively interrupted the regular traffic signal cycle and prioritized the ambulance’s route, minimizing delays.
- **Traffic Clearance Efficiency**: The traffic clearance efficiency was calculated by tracking the number of vehicles cleared per signal cycle. Under high traffic

density, the system was able to clear up to 12-15 vehicles per signal cycle, demonstrating efficient traffic management. During low traffic density, the system cleared approximately 3-5 vehicles per cycle.

- **Average Waiting Time per Vehicle:** The average waiting time for vehicles at red lights varied depending on traffic density In high-density scenarios, the average waiting time was slightly higher, but the system reduced it during peak hours by optimizing green light durations. The mean waiting time was reduced in times of low traffic density by decreasing the green light duration for non-activated lanes.

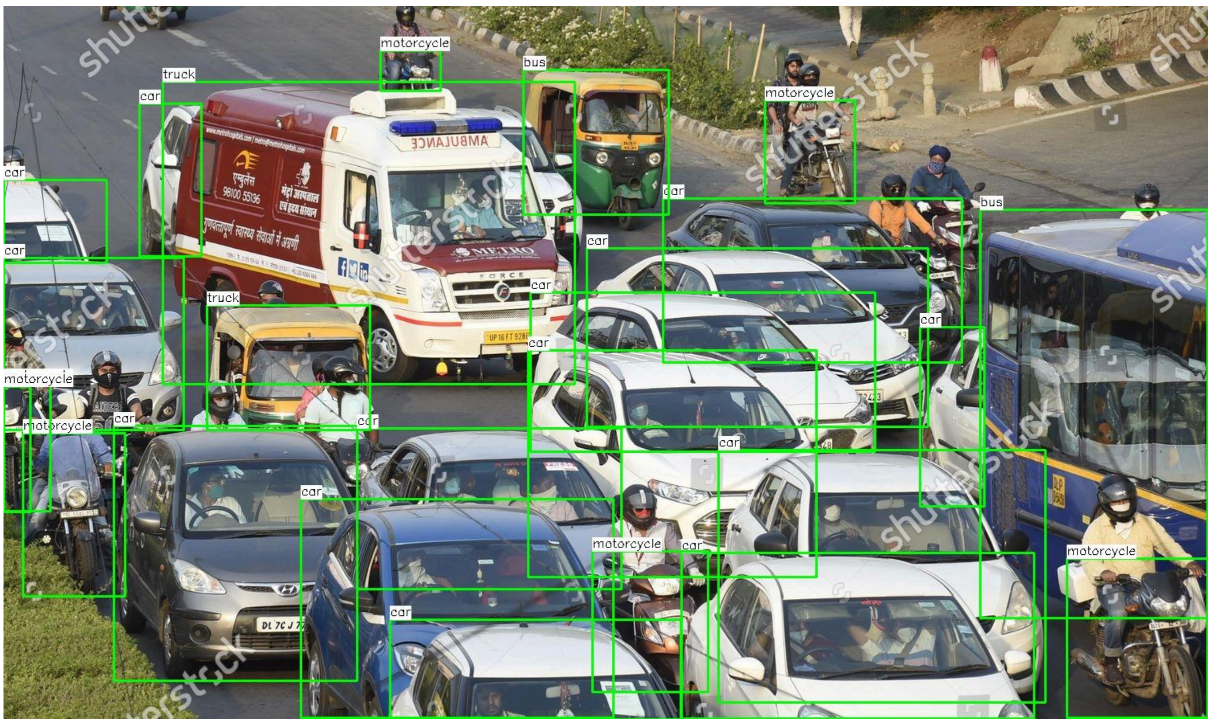


Fig.10.1. Non identification of Ambulance as Ambulance by Old Model

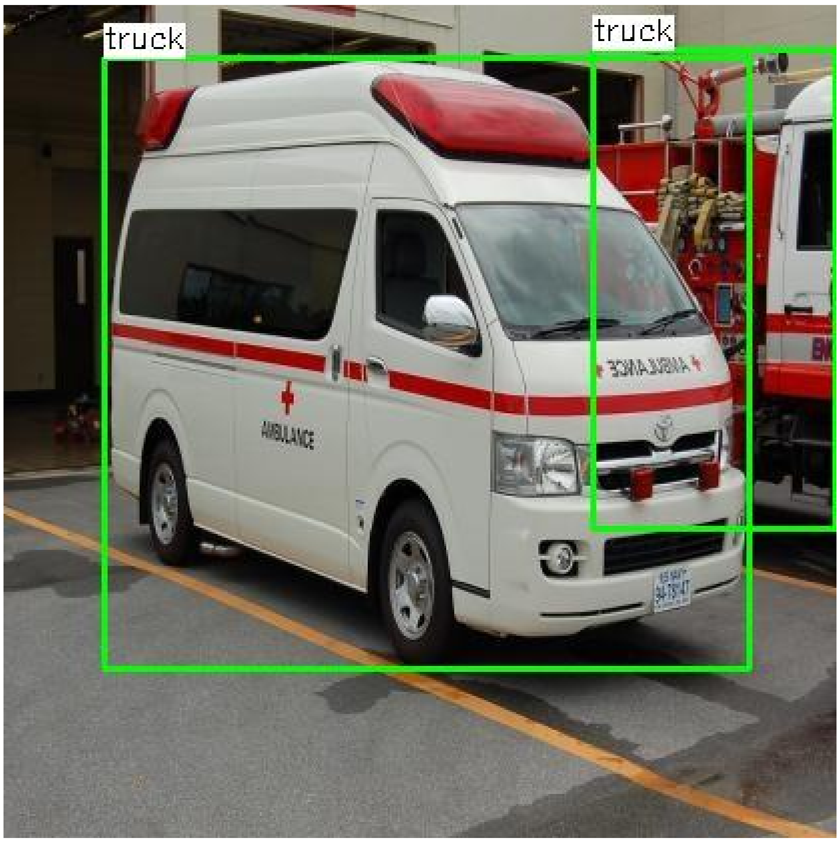


Fig.10.2. Baseline YOLO model detects ambulance as a truck due to lack of specific training data

One critical problem identified in classical object detection systems is not being able to specifically identify ambulances as a distinct class. General-purpose dataset-trained models tend to incorrectly classify ambulances as trucks or vans because there are no ambulance-specific labels.

In our project, we overcame this limitation by developing and training a custom ambulance detection dataset. The dataset consisted of more than 480 labeled images of ambulances from various angles and environments, annotated for emergency vehicle recognition. The model was trained for more than 100 epochs with YOLO architecture, with data augmentation methods such as rotation, scaling, and flipping to improve robustness.

As a consequence:

The model trained specifically on the custom dataset was able to identify ambulances as a separate class, as opposed to earlier models that classified them as trucks.

Ambulance accuracy and detection confidence were greatly enhanced, with high mAP scores obtained during testing.

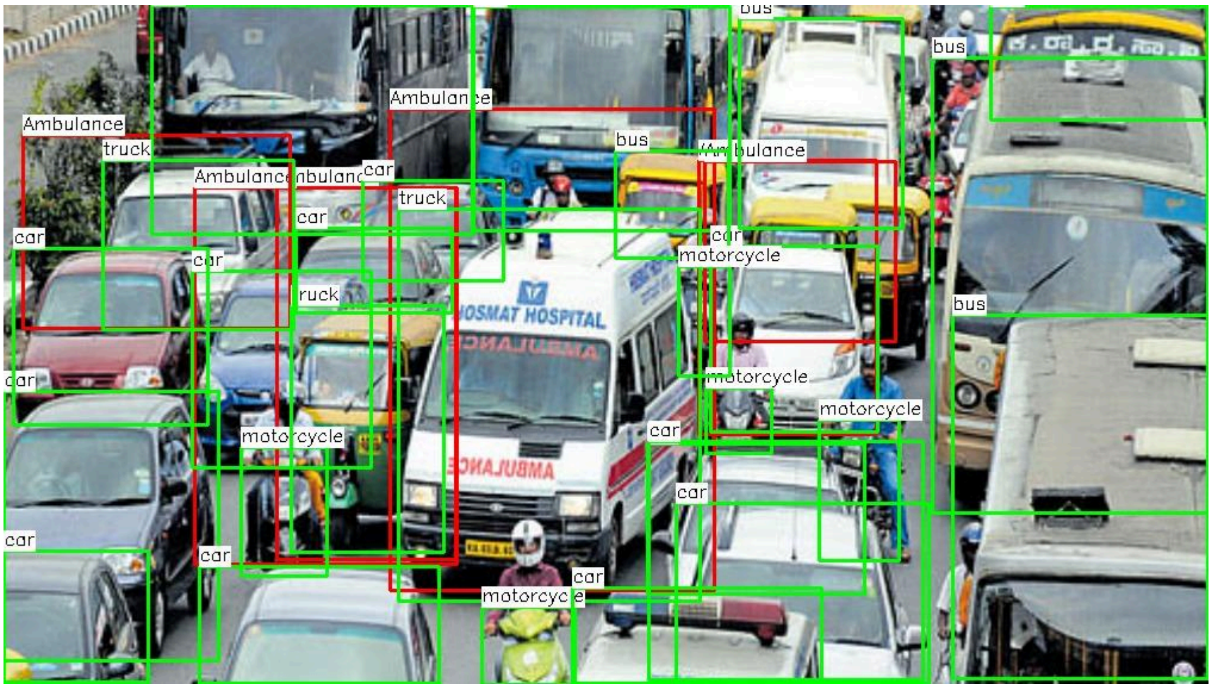


Fig.10.3. Ambulance successfully detected by the enhanced model trained on custom dataset.

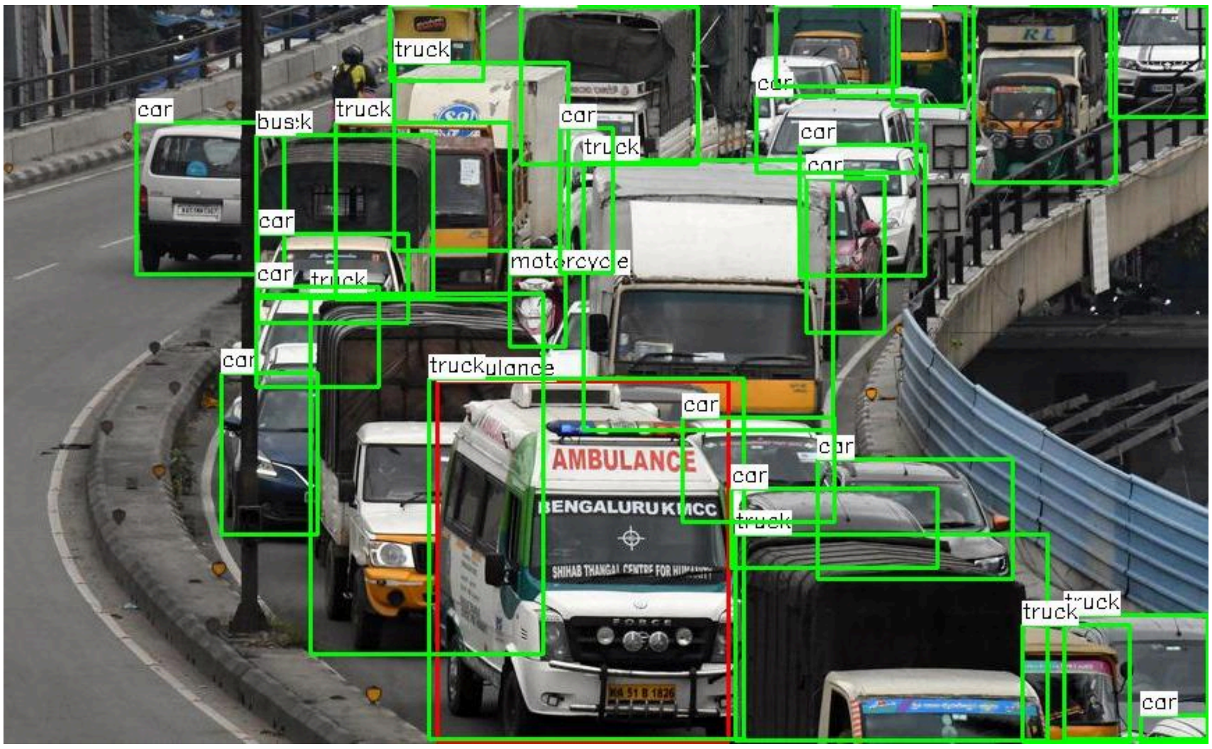


Fig.10.4.Accurate detection: Ambulance recognized correctly using custom-trained YOLO model.

This resulted in more accurate emergency vehicle prioritization in traffic signal control, allowing the system to clear roads for ambulances efficiently.

The enhanced classification ability is a significant leap towards developing AI-driven intelligent traffic systems that can react to actual emergency situations in real time and with accuracy.

10.4. Additional Observations

- **Simulation Efficiency**: The system was able to run continuously without significant drops in performance, even with the introduction of real-time detection and ambulance prioritization.
- **Scalability**: When scaled to a ring-road model with four connected junctions (J1–J4), the system continued to perform optimally, ensuring that each junction communicated and switched signals efficiently based on vehicle density and emergency priorities.

Chapter 11
Individual Contribution

Problem Statement:

To create an intelligent, real-time traffic management system based on AI vehicle and ambulance detection, dynamic signal control, and simulation for enhanced emergency response and urban traffic efficiency.

Name of the Student: Prakhar Pachauri

Module Title**:** Vehicle Detection

Project's Module Objectives - Individual Perspective:

To develop and implement a real-time vehicle recognition system that can detect all types of vehicles, especially ambulances, and provide accurate detection outputs for further traffic control processing.

Project's Module Scope - Individual Perspective:

This module is the cornerstone of the smart traffic system. It processes camera streams, detects and classifies cars, and provides annotated data essential for density analysis and emergency prioritization.

Project's Module(s) - Individual Contribution:

- Applied a two-stage detection pipeline using two YOLOv8 models:
- Pre-trained model for general cars.
- Custom-trained model for ambulances.
- Applied bounding box visualization: red for ambulances, green for others.
- Output passed to density and signal modules.
- Confirmed real-time performance via OpenCV batch processing and execution.

Hardware & Software Requirements:

- GPU-based training/inference hardware

- Python 3.10
- YOLOv8 (Ultralytics)
- OpenCV

Module Interfaces:

- Accepts live/video stream from simulation or real-world cameras.
- Outputs bounding box coordinates and class labels for other modules.

Module Dependencies:

- YOLOv8 model weights
- Detection threshold parameters
- Pre-processing image utilities

Module Design:

- Modular design with load models, infer, and annotation in separate functions.
- Detection threshold configuration for redundancy.

Module Implementation:

- Implemented as Python and Ultralytics YOLOv8.
- Real-time video processing loop with dual-model inference.
- Output images written out with visual verification overlays.

Module Testing Strategies:

- Tested on a variety of traffic scenes and times of day.
- Accuracy metrics verified with labeled ground truth.
- Ambulance detection tested on independent dataset.

Module Deployment:

- Exported as an importable Python module for reuse.
- Linux/Windows system compatible.
- Model weights and configs kept externally for simple update.

Student Name: Tanvi Khadap

Module Title: Density Detection

Project's Module Objectives - Individual Perspective:
To apply a logic that scans vehicle density in various road segments to supply signal control with real-time density information and ambulance notification.

Project's Module Scope - Individual View:
Affects deciding real-time traffic load for each junction lane. Output directly influences how long a signal remains green.

Project's Module(s) - Individual Contribution:

- Split image into zones (left, center, right).
- Estimated vehicles in each zone based on YOLOv8 detection output.
- Dected ambulances and marked their detection.
- Sent total vehicle count and ambulance flag to signal control.

Hardware & Software Requirements:

- Python 3.10
- OpenCV
- YOLOv8
- NumPy

Module Interfaces:

- Input: Bounding boxes from detection module.
- Output: Per-lane vehicle counts and ambulance presence.

Module Dependencies:

- Vehicle detection module
- Image preprocessing utilities
- Lane division logic

Module Design:
Lane zoning based on bounding box coordinates.

Counting logic adaptable to varying input resolutions.

Module Implementation:

- Python logic for inspecting bounding boxes.
- Counted overlapping boxes in their respective zones.
- Generated density arrays for processing.

Module Testing Strategies:

- Used video samples of different traffic densities.
- Compared output with manual count for verification.

Module Deployment:

- Independent module callable in simulation.py.
- Lightweight design to execute on CPU if necessary.

Student's Name: Mehul Vaidya

Module Name: Traffic Light Time Control

Project's Module Goals - Individual Perspective:

Dynamically regulate traffic signals from real-time vehicle density and emergency detection, optimizing normal flow and ambulance priority.

Project's Module Scope - Individual Perspective:

Central decision-making module governing the direction and timing of signal lights. It balances fairness across lanes and quick response for ambulances.

Project's Module(s) - Individual Contribution:

- Implemented decide_signals() function based on traffic counts.
- Managed time-capping for fairness across lanes.
- Managed signal duration based on density output.

Hardware & Software Requirements:

- Python 3

- Threading library

Module Interfaces:
- Input: Vehicle count array + ambulance flag
- Output: Signal direction and timing duration

Module Dependencies:
- Density and detection modules
- Time management logic

Module Design:
- Graph-based design for junction connection
- Modular functions for calculating wait time and direction

Module Implementation:
- Python with NetworkX for path computation.
- Used sleep and thread locking for real-time effect.

Module Testing Strategies:
- Simulated dense vs sparse traffic.
- Randomized ambulance arrival for stress testing.

Module Deployment:
- Designed as a callable function in simulation loop.
- Multithreaded to avoid UI blocking.

Name of the Student: Riya Gupta

Module Title: Simulation

Project's Module Objectives - Individual Perspective:

To design a Python-based simulation to test and illustrate the real-time interfacing of all modules in an artificial traffic intersection environment.

Project's Module Scope - Individual Perspective:

The simulation simulates real-time traffic movement and checks how the system would actually work in a true-life 4-way intersection with dynamic vehicle volumes and emergency situations.

Project's Module(s) - Individual Contribution:

- Designed simulation.py that interfaces all modules.
- Implemented visual signal and vehicle simulation using OpenCV.
- Visualized signal switch according to detection and density.

Hardware & Software Requirements:

- Python 3
- PyGame library

Module Interfaces:

- Inputs: detection and density outputs
- Output: visual traffic simulation with signal overlays

Module Dependencies:

- Vehicle Detection
- Density Detection
- Signal Control

Module Design:

- Centralized loop calling each module sequentially.
- UI window for real-time visualisation.

Module Implementation:

- Python using OpenCV for frame-by-frame rendering.
- Outlined lanes, vehicles, and signal state in real time.

Module Testing Strategies:

- Video feeds with known vehicle flow

- Timed ambulance arrival testing

Module Deployment:

- Integrated with primary execution file.
- Portable with.mp4 or webcam input support.

Applications

This AI-Powered Traffic Management System can be applied to various real-life situations:

- **Urban Traffic Control**: Assists in controlling day-to-day congestion within cities by adjusting signal times to actual traffic movement, enhancing the experience of commuters.

- **Emergency Response Assistance**: Lessens ambulance delays to enable faster transportation of patients to hospitals and improved survival rates

- **Smart City Infrastructure**: Can be integrated into larger smart city systems with IoT-implemented traffic sensors, intelligent surveillance, and centralised control systems.

- **Public Events or VIP Movements**: In events or VIP motorcades, the system may temporarily provide priority to selected routes to ensure smooth flow of traffic.

- **Future Integration with Autonomous Vehicles**: The adaptive signal system can interact with autonomous vehicles to further optimize city traffic operations.

- **Environmentally Friendly**: Through minimizing idle time and congestion, the system can also reduce vehicle emissions, making the air cleaner.

Conclusion

This work showcases the strong combination of artificial intelligence and real-time computer vision technology in combating urban traffic management issues. With the application of the latest YOLOv8 (You Only Look Once) deep learning model, the system accurately detects ambulances and other types of vehicles. The innovation lies in the system's ability to replicate human decision-making in traffic management, but at an enhanced speed, precision, and reliability, ensuring no delays or fatigue.

Key Achievements:

1. **Real-Time Vehicle and Ambulance Detection**

   The system uses YOLOv8 for real-time object detection, accurately identifying vehicles like cars, motorcycles, trucks, and ambulances. The ability to detect these vehicles as they appear on the roads allows the system to react instantaneously. This ensures that emergency vehicles are given priority without delay, greatly enhancing response times and potentially saving lives.

2. **Dynamic Traffic Signal Control Based on Traffic Density**

   In contrast to conventional traffic systems that use static timers, our system adjusts the signal duration according to real-time traffic conditions. It determines the number and proportion of vehicles that are on the road and adjusts the duration of the green signal to provide smooth flow of traffic. This is a dynamic method that minimizes waiting times and eliminates congestion, particularly during rush hours, based on such factors as vehicle size and speed.

3. **Ambulance Prioritization to Improve Emergency Response**

   One of the salient features of this system is the capacity to identify and give priority to ambulances in real time. When an ambulance is identified, the system overrides the existing traffic signal automatically and grants an instant green light to the direction of the ambulance. This automation cuts down emergency response times considerably, replicating the manner in which traffic police officers manually clear roads for ambulances but much more efficiently and uniformly.

4. **Modular, Scalable, and Secure Architecture**

   The system design is modular, and updates, maintenance, and extension of functionality are facilitated. It is possible to introduce new junctions or extensions with minimal disturbance to current functionality. Security provisions are also made such that only approved personnel can access and alter the system's controls to prevent misuse.

5. **Promising Simulation Results**

   The system was tested extensively in a simulated environment replicating actual traffic conditions. Results showed the system's ability to manage multiple traffic signals dynamically, prioritize ambulances efficiently, and clear vehicles in an optimized manner. While still in its prototype phase, these results suggest great potential for real-world applications, offering the promise of smarter, safer urban traffic management.

References

[1] B. Ghazal, K. ElKhatib, K. Chahine, and M. Kherfan, "Smart traffic light control system," in Proc. 3rd Int. Conf. on Electrical, Electronics, Computer Engineering and their Applications (EECEA), Beirut, Lebanon, 2016, pp. 161–166.

[2] K. M. Almuraykhi and M. Akhlaq, "STLS: Smart traffic lights system for emergency response vehicles," in Proc. 2019 Int. Conf. on Computer and Information Sciences (ICCIS), 2019, pp. 1–6.

[3] M. M. Gandhi, D. S. Solanki, R. S. Daptardar, and N. S. Baloorkar, "Smart control of traffic light using artificial intelligence," in Proc. 5th IEEE Int. Conf. on Recent Advances and Innovations in Engineering (ICRAIE), 2020, pp. 1–5.

[4] B. Wang, H. Zheng, K. Qian, X. Zhan, and J. Wang, "Edge computing and AI-driven intelligent traffic monitoring and optimization," in Proc. 2nd Int. Conf. on Software Engineering and Machine Learning (ICSEML), 2024, pp. 225–230.

[5] P. V., A. R., R. Gowda D., and S. G., "AI-driven urban traffic optimization to assess complex traffic patterns for public traffic control and mobility," Int. J. for Research in Applied Science & Engineering Technology (IJRASET), vol. 11, no. 11, pp. 794–798, Nov. 2023.

[6] A. Kanungo, A. Sharma, and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," in Proc. 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, India, 2014, pp. 1–5.

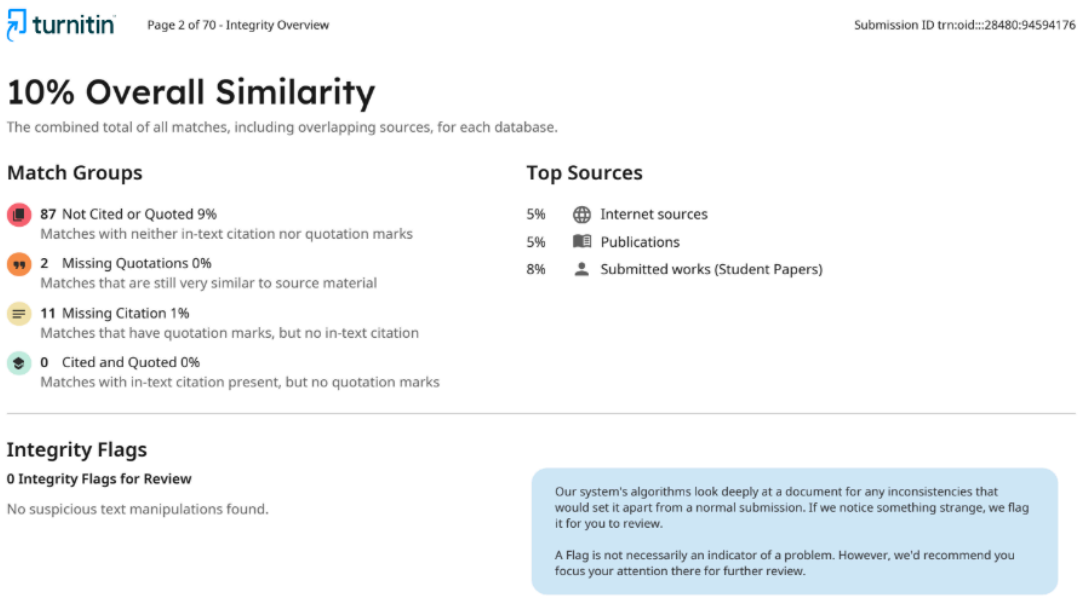
[7] W.-H. Lee and C.-Y. Chiu, "Design and implementation of a smart traffic signal control system for smart city applications," Sensors, vol. 20, no. 2, p. 508, Jan. 2020.

[8] Q. Jin, "Automatic control of traffic lights at multiple intersections based on artificial

intelligence and ABST light," Journal/Conference Name, 2024.

[9] Khushi, "Smart Control of Traffic Light System using Image Processing," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 99-103, doi: 10.1109/CTCEEC.2017.8454966.

Plagiarism Report



turnitin Page 2 of 70 - Integrity Overview Submission ID trn:oid:::28480:94594176

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- 87 Not Cited or Quoted 9%
  Matches with neither in-text citation nor quotation marks
- 2 Missing Quotations 0%
  Matches that are still very similar to source material
- 11 Missing Citation 1%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

5%	Internet sources
5%	Publications
8%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.