A Real-Time Traffic Signal Management System Using Image-Based Traffic Density Analysis

Abstract

This project presents a real-time traffic signal management system that dynamically prioritizes traffic signals based on image-derived traffic density. Using CCTV traffic frames and computer vision techniques, the system analyzes vehicular density on multiple roads and assigns traffic signals accordingly. The solution is implemented as a full-stack application with a Python (FastAPI + OpenCV) backend and a React-based frontend, simulating real-world intelligent traffic control at an intersection.

Objectives

- To analyze traffic density using image processing techniques.
- To simulate real-time traffic signal control based on congestion levels.
- To design a scalable full-stack architecture for intelligent traffic systems.
- To visualize traffic conditions and signal decisions clearly for users.

System Architecture

The system follows a clear separation of concerns:

Frontend (React):
- Periodically requests analyzed traffic data.
- Displays road-wise traffic images, density levels, signal status, and timers.
- Shows countdown for next system update and network latency.

Backend (FastAPI + OpenCV):
- Reads traffic images from a randomized dataset.
- Computes traffic density scores using image processing.
- Determines signal priority and green/red assignment.
- Returns structured JSON responses to the frontend.

Dataset

The TrafficCAM dataset was used, which contains CCTV traffic frames captured at real-world intersections. A one-time offline randomization of frames was performed to create a unified dataset. During runtime, images are streamed sequentially using a circular index to ensure stochastic yet reproducible traffic simulation without runtime overhead.

Backend Design

Framework: FastAPI

Key Features:
- High-performance asynchronous request handling.

- Image processing using OpenCV and NumPy.
- Real-time traffic density estimation.

API Endpoint:
GET /analyze-traffic

Response includes:
- Processing time in milliseconds.
- Road-wise density scores.
- Traffic level classification.
- Recommended signal assignment (green/red).

Frontend Design

The frontend is built using React with Tailwind CSS for styling.

Features:
- Grid-based dashboard displaying four roads.
- Traffic images for visual verification.
- Green signal assigned to the road with highest density.
- Red signal assigned to remaining roads.
- Countdown timer indicating next update cycle.
- Display of backend and frontend latency.

Traffic Signal Logic

At each refresh cycle:
- The road with the highest traffic density score receives a GREEN signal.
- All other roads receive RED signals.

This mimics real-world priority-based traffic control and ensures clarity in system behavior.

Latency Measurement

Backend latency is measured by recording timestamps before and after image processing. Frontend latency is calculated as the total request-response time. Displaying both metrics provides insight into system performance.

How to Run the Project

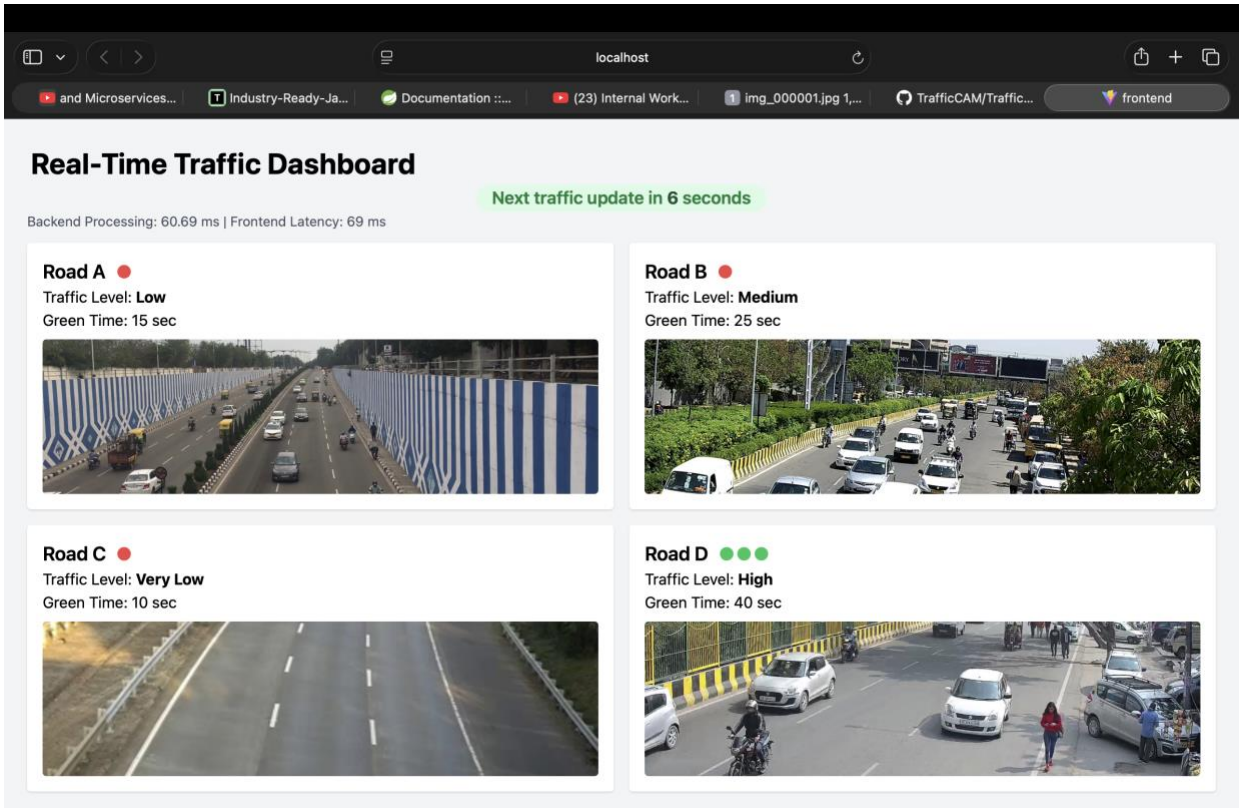
Backend:
1. Create and activate a virtual environment.
2. Install dependencies: fastapi, uvicorn, opencv-python, numpy.
3. Run server using: uvicorn app:app --reload

Frontend:
1. Install dependencies using npm.

2. Start development server using npm run dev.
3. Access dashboard via browser.

Conclusion

This project demonstrates a practical application of computer vision and full-stack development in intelligent transportation systems. By combining real-time image analysis with dynamic visualization, the system provides a clear and extensible foundation for smart traffic signal control.



Module-wise Requirement Satisfaction

Module 1: Problem Identification and Requirement Analysis

The project clearly identifies the problem of inefficient static traffic signal timing at busy intersections. Requirements such as real-time decision making, congestion-based prioritization, scalability, and clarity of visualization were analyzed before implementation. The choice of image-based traffic density analysis directly addresses real-world traffic management challenges.

Module 2: System Design and Architecture

The system follows a modular client-server architecture. The backend (FastAPI) is responsible for data processing and decision logic, while the frontend (React) handles visualization. This clear separation of concerns ensures maintainability, scalability, and extensibility, satisfying system design objectives.

Module 3: Data Acquisition and Preprocessing

Traffic data is acquired from the TrafficCAM dataset consisting of real CCTV frames. A one-time offline randomization script preprocesses the dataset to remove temporal bias and ensure diversity. This satisfies the requirement for realistic and unbiased input data.

Module 4: Algorithm and Logic Implementation

Computer vision techniques using OpenCV are applied to estimate traffic density from images. A deterministic priority algorithm assigns a green signal to the road with the highest density and red signals to all others. The logic is simple, interpretable, and aligns with real-world traffic control principles.

Module 5: Implementation and Integration

The backend APIs are integrated with the frontend using RESTful GET requests. CORS handling ensures smooth cross-origin communication. The frontend dynamically updates signal states, images, and metrics every fixed interval, demonstrating successful full-stack integration.

Module 6: Performance Evaluation and Optimization

Backend processing time and frontend latency are measured and displayed on the dashboard. This provides transparency into system performance and demonstrates awareness of real-time constraints in intelligent transportation systems.

Module 7: User Interface and Visualization

A responsive dashboard visualizes traffic images, congestion levels, signal states, and update countdowns. Visual signal indicators (green and red lights) make the system intuitive and easy to understand, fulfilling usability and visualization requirements.

Module 8: Testing, Validation, and Demonstration

The system is tested across multiple refresh cycles using randomized traffic frames. Signal decisions are visually validated against displayed traffic density, ensuring correctness. The live dashboard serves as a clear demonstration of system behavior.

Module 9: Documentation and Reporting

Comprehensive documentation is provided in this report, explaining objectives, architecture, logic, dataset usage, and performance metrics. The documentation ensures that the project is easy to evaluate and understand by instructors and reviewers.