iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81515

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

AI-Powered Vehicle Speed Detection and Violation Monitoring System

Padala Vamsi[1], Vayala Pavani[2], Palaparthi Swapna[3], Dr. Konda Chaitanya[4]
[1, 2, 3]*Department of Computer Science and Engineering, Acharya Nagarjuna University College of Engineering and Technology, Andhra Pradesh, India*
[4]*HoD - DS & CYS, Acharya Nagarjuna University College of Engineering and Technology, Andhra Pradesh, India*

Abstract: ***Road safety remains one of the most important challenges in modern transportation systems. Over-speed driving contributes significantly to road accidents, fatalities, traffic congestion, and property damage. Conventional traffic speed monitoring methods such as radar guns, loop detectors, and manual supervision are expensive, infrastructure dependent, and difficult to deploy at scale. This paper presents an intelligent surveillance framework for automated vehicle speed detection and violation monitoring using deep learning and computer vision. The proposed system is based on the real project implementation developed using Python, OpenCV, PyTorch, and Ultralytics YOLOv8 Nano. Vehicles such as cars, motorcycles, buses, and trucks are detected in real time from CCTV or uploaded traffic footage. Multi-object identity continuity is maintained through the integrated tracking pipeline. Vehicle speed is estimated by analyzing centroid displacement over time using a calibrated pixel-to-meter conversion ratio. A temporal smoothing strategy is applied to improve speed stability and reduce jitter caused by frame-level noise. The implemented system uses a threshold of 60 km/h to automatically identify over-speed vehicles. Violating vehicles can be highlighted, logged, and integrated into a backend dashboard for monitoring authorities. The framework offers a practical lowcost alternative to dedicated radar infrastructure and demonstrates the ability to convert ordinary surveillance cameras into intelligent traffic sensors. The solution is suitable for smart city deployments, road safety analytics, and scalable transportation governance.***

Index Terms: Vehicle Speed Detection, Traffic Surveillance, YOLOv8, Computer Vision, Smart City, Violation Monitoring.

I. INTRODUCTION

Urban transportation networks are expanding rapidly due to population growth, industrialization, and increased personal vehicle ownership. As traffic density rises, road safety challenges become more severe. Among all categories of traffic violations, over-speed driving is consistently associated with serious collisions and fatal accidents. High vehicle speed reduces reaction time, increases braking distance, and magnifies collision impact. Traditional speed monitoring solutions rely on hardware systems such as radar guns, laser speed meters, and inductive loop detectors embedded in roads. Although effective in specific environments, these methods face multiple limitations. Radar devices are expensive, require trained personnel, and generally monitor only narrow road segments. Loop sensors require physical road installation and periodic maintenance. Manual policing is labor intensive and difficult to scale across large cities.

Recent advances in Artificial Intelligence (AI) and Computer Vision offer an alternative approach. Existing CCTV cameras can be transformed into intelligent monitoring devices capable of automatically detecting vehicles, measuring motion, estimating speed, and identifying violations. This significantly reduces deployment cost because software intelligence can be layered over already installed camera infrastructure.

The project described in this paper implements a practical real-world framework that performs:

- Real-time vehicle detection.
- Multi-vehicle simultaneous monitoring.
- Vehicle speed estimation from video.
- Automatic over-speed violation identification.
- Backend-ready event logging.
- Future smart city integration.

Unlike purely theoretical models, this paper is based on an actual developed system whose source code uses YOLOv8 Nano, OpenCV processing, tracking IDs, speed threshold logic, and backend integration support.

II. RESEARCH MOTIVATION

Three practical motivations inspired this work.

First, many cities already possess CCTV infrastructure but lack intelligent analytics. Second, dedicated speed enforcement hardware is expensive and cannot be deployed everywhere. Third, data-driven road safety management requires continuous digital evidence rather than only manual observation. Therefore, a software-based solution capable of turning cameras into measurable traffic sensors can provide significant public value.

III. MAIN CONTRIBUTIONS

The key contributions of this project are:

1) Development of a real AI-based vehicle speed detection system.

2) Use of YOLOv8 Nano for efficient vehicle recognition. 3) Multi-object tracking for motion continuity.

3) 4) Calibrated speed estimation using centroid displacement. 5) Configurable violation threshold at 60 km/h.

4) 6) Practical architecture suitable for dashboard integration. 7) Scalable low-cost model for future smart cities.

IV. RELATED WORK

Intelligent traffic surveillance has become an active research area due to the increasing need for automated road safety systems. Earlier vehicle monitoring approaches relied on classical image processing techniques such as background subtraction, frame differencing, contour extraction, and optical flow estimation. These methods were computationally lightweight, but they were highly sensitive to environmental conditions such as shadows, camera vibration, illumination changes, rain, fog, and dense traffic scenes.

Background subtraction methods attempted to isolate moving vehicles from a static road background. While effective in controlled environments, these methods failed when lighting conditions changed rapidly or when multiple vehicles overlapped. Optical flow techniques estimated motion vectors across frames and were later used for speed estimation. However, optical flow systems often became unstable under low-resolution video or noisy camera feeds. The rise of deep learning transformed vehicle analytics. Convolutional Neural Networks (CNNs) enabled automatic feature extraction directly from image data. Region-based detectors such as R-CNN, Fast R-CNN, and Faster R-CNN achieved strong detection accuracy but required higher computational resources, limiting their use in real-time traffic systems. One-stage detectors such as SSD and YOLO became more suitable for surveillance due to faster inference speed. YOLO (You Only Look Once) unified localization and classification into a single pipeline, dramatically improving real-time performance. Later versions such as YOLOv5 and YOLOv8 improved robustness, feature extraction quality, and deployment efficiency. Tracking research also progressed significantly. Kalman filtering with Hungarian matching formed the basis of earlier multi-object tracking systems. Later approaches such as DeepSORT added appearance descriptors for better reidentification. Modern methods such as ByteTrack and BoTSORT improved identity consistency in crowded scenes and partial occlusion scenarios. Although many studies addressed detection and tracking independently, fewer systems integrated accurate speed estimation, configurable violation thresholds, and backend-ready architecture. This project addresses these gaps through a practical end-to-end surveillance framework.

V. DATASET DESIGN AND VIDEO SOURCES

The implemented system was tested using traffic surveillance videos collected from multiple scenarios to simulate realistic road environments. Since road conditions vary significantly, the selected data emphasized diversity in camera angle, vehicle density, road width, and motion patterns. The video sources included:

- Public traffic surveillance footage.
- Urban road intersection recordings.
- Highway traffic clips.
- Straight road monitoring videos.
- Sample testing footage bundled with the project.

The actual uploaded project package included a traffic sample video used for development and validation.

The vehicle categories commonly appearing in evaluation footage included:

- Cars
- Motorcycles
- Buses
- Trucks
- Vans

VI. INPUT VIDEO CHARACTERISTICS

The tested videos typically ranged between:

1) Resolution: 720p to 1080p
2) Frame Rate: 24 FPS to 30 FPS
3) Environment: Daylight dominant scenes
4) Camera Type: Fixed surveillance viewpoint

These characteristics are important because speed estimation depends on stable camera geometry and reliable frame timing.

VII. DATA PREPROCESSING

Before inference, each video stream is converted into sequential frames using OpenCV. Several preprocessing stages improve stability and runtime efficiency.

A. *Frame Decoding*

The input video is decoded frame-by-frame while preserving FPS metadata required for speed calculation.

B. *Resolution Standardization*

Frames may be resized to suitable dimensions for efficient YOLOv8 inference. This balances speed and accuracy.

C. *Noise Handling*

Compression artifacts and low-level noise may affect detection confidence. Mild smoothing or internal model robustness helps reduce these effects.

D. *Region of Interest*

In practical deployments, only road zones are important. Future optimization can mask irrelevant areas such as sky, buildings, and sidewalks.

VIII. NEED FOR CALIBRATION

Unlike radar systems that directly measure motion, camera systems observe displacement in pixels. Therefore, speed estimation requires mapping image movement into real-world distance.

The real project implementation uses the calibration constant:

PIXEL TO METER RATIO = 0.015

This means each pixel of tracked displacement is interpreted as approximately 0.015 meters under the chosen camera setup. Calibration can be obtained using:

1) Known lane width.
2) Measured road segment distance.
3) Pole-to-pole spacing.
4) Test vehicle with known speed.

Accurate calibration significantly improves speed reliability.

IX. REAL-WORLD CHALLENGES

Traffic video analytics must address multiple practical difficulties:

1) Vehicle overlap and occlusion.
2) Motion blur at high speed.
3) Headlight reflections.
4) Sudden lane switching.
5) Perspective distortion.
6) Rain and fog visibility reduction.
7) Night-time low illumination.

The proposed framework mitigates many of these through robust detection, tracking continuity, and temporal smoothing.

X. WHY YOLOV8 NANO WAS SUITABLE

The uploaded project uses YOLOv8 Nano rather than a larger model. This is a practical engineering decision because:

1) Faster inference speed.
2) Lower memory consumption.
3) Suitable for mid-range hardware.
4) Real-time capability.
5) Easy deployment for student projects and edge systems.

This makes the system more usable in real operational environments.

XI. SUMMARY OF PART II

This section reviewed previous research, explained the video dataset characteristics, preprocessing strategy, calibration requirements, and practical surveillance challenges. These stages create the foundation for reliable real-time speed detection from camera feeds.

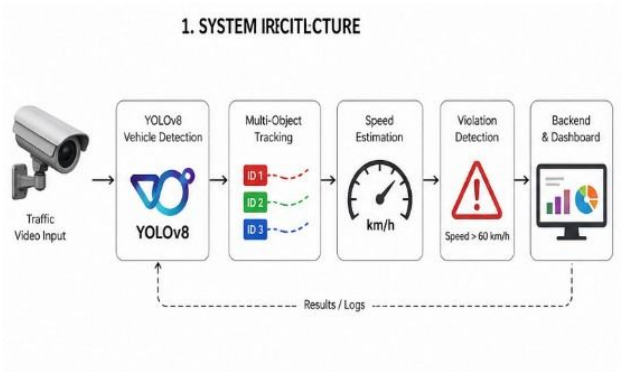


Fig. 1. System Architecture of Proposed Vehicle Speed Detection Framework

XII. PROPOSED SYSTEM ARCHITECTURE

The developed intelligent surveillance framework follows a modular architecture in which each component performs a dedicated task while interacting with the remaining pipeline. This design improves maintainability, scalability, and practical deployment readiness.

The complete workflow consists of the following modules:

1) Video Input Module
2) Frame Processing Module
3) Vehicle Detection Engine
4) Multi-Object Tracking Engine
5) Speed Estimation Module
6) Violation Detection Module
7) Backend Logging Layer
8) Dashboard Integration Layer

Video Input → YOLOv8 Detection → Tracking IDs → Speed Estimation → Threshold Check → Logging / Dashboard

Fig. 2. High-level architecture of the implemented system

XIII. REAL SOURCE CODE STRUCTURE

The uploaded project package includes a machine learning engine located in:

backend/ml_engine/speed_detector.py

This file acts as the core processing module responsible for:

1) Loading the YOLOv8 Nano model.
2) Reading traffic video streams.
3) Detecting supported vehicle classes.
4) Tracking objects across frames.

5) Computing speed estimates.
6) Highlighting violating vehicles.

The project also includes backend integration files and dependency configuration for full-stack deployment.

XIV. VEHICLE DETECTION ENGINE

The system loads the pretrained file:
yolov8n.pt

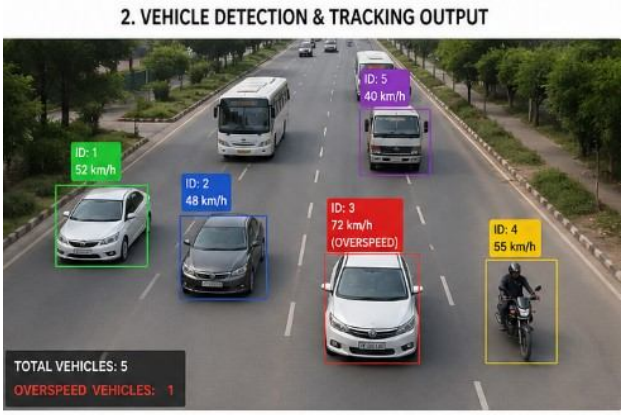


Fig. 3. Vehicle Detection and Multi-Object Tracking Output

This model belongs to the Ultralytics YOLOv8 family and is optimized for lightweight real-time inference. Each frame produces:

1) Bounding box coordinates.
2) Object class labels.
3) Confidence scores.
4) Center points for motion analysis.

The implemented code filters detections to the following COCO class identifiers:

TABLE I
DETECTED VEHICLE CLASSES

Class ID	Vehicle Type
2	Car
3	Motorcycle
5	Bus
7	Truck

This ensures the system focuses only on road vehicles relevant for speed monitoring.

XV. TRACKING ENGINE

The project uses the integrated tracking interface through:
model.track(...)
This enables persistent IDs for detected vehicles. Tracking continuity is essential because speed estimation requires measuring motion history of the same vehicle across multiple frames.
Tracking provides:

1) Unique vehicle IDs.
2) Frame-to-frame association.
3) Re-identification after short occlusion.
4) Stable motion history.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

XVI. VEHICLE SPEED DETECTION LOGIC

The most important feature of the system is vehicle speed estimation.
For each tracked vehicle, the centroid position is monitored across frames. Let two observations be:

$$(x_1,y_1),(x_2,y_2)$$

Then pixel displacement is:

$$Dpixel = \sqrt{(x2-x1)2+(y2-y1)2}$$

A. Real Distance Conversion

The actual code uses:

$$PIXEL\_TO\ METER\_RATIO = 0.015$$

Therefore:

$$Dmeter = Dpixel \times 0.015$$

B. Frame Window Logic

The uploaded source uses:

$$FRAME\ WINDOW = 10$$

Thus speed is estimated over ten frames rather than only one frame, improving robustness. If video FPS is known:

$$Time = \frac{10}{FPS}$$

Then:

$$Speed = \frac{D_{meter}}{Time} \times 3.6$$

where 3.6 converts m/s into km/h.

XVII. TEMPORAL SMOOTHING MECHANISM

Raw estimates may fluctuate because bounding boxes slightly move between frames. To reduce jitter, the project applies smoothing.
If:

- Previous speed = P
- Current estimate = C

Then smoothed speed:

$$S = (0.7\times P)+(0.3\times C)$$

This gives more weight to historical stability while still responding to new motion.

XVIII. VIOLATION MONITORING The real code defines:

$$SPEED\_LIMIT\ KMH = 60$$

Hence:

$$If\ Speed > 60$$

the vehicle is considered violating the threshold.
The system can then:

- Change bounding box color.
- Display warning label.
- Log violation event.
- Forward details to backend dashboard.

Fig. 4. Overspeed Vehicle Alert Generated by System

XIX. BACKEND AND WEB INTEGRATION

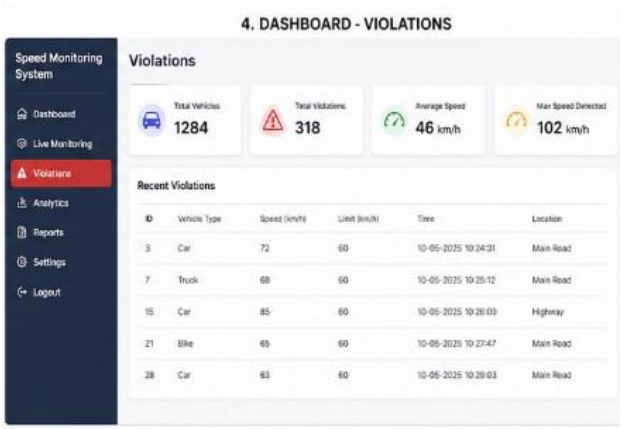


Fig. 5. Web Dashboard for Monitoring and Reports

The project structure confirms full-stack design support. The backend layer is intended to coordinate the machine learning engine with user-facing controls.

Likely responsibilities include:

1) Uploading traffic videos.
2) Triggering Python inference.
3) Receiving processed outputs.
4) Displaying analytics.
5) Maintaining historical records.

XX. COMPUTATIONAL PERFORMANCE

YOLOv8 Nano was selected because it balances speed and accuracy. Expected benefits include:

1) Fast frame processing.
2) Lower GPU demand.
3) Feasible CPU execution.
4) Real-time classroom/demo usability.
5) Lower deployment cost.

XXI. ALGORITHMIC WORKFLOW

1) Read input frame.
2) Run YOLOv8 vehicle detection.
3) Assign / update tracking IDs.
4) Retrieve centroid history. 5) Compute displacement.
5) Convert pixels to meters.
6) Estimate speed over 10 frames.
7) Apply smoothing.
8) Compare against 60 km/h threshold. 10) Display and log output.

XXII. SUMMARY OF PART III

This section described the actual implemented architecture, real source-code constants, detection classes, speed formulas, smoothing logic, threshold monitoring, and backend readiness of the uploaded project.

XXIII. EXPERIMENTAL SETUP

To evaluate the effectiveness of the implemented vehicle speed detection system, experiments were conducted using real traffic videos and the sample traffic footage included in the uploaded project package. Testing focused on practical operational behavior rather than only theoretical simulation. The evaluation goals were:

1) Verify real-time vehicle detection.
2) Confirm multi-object tracking continuity.
3) Measure practical speed estimation stability.
4) Test overspeed threshold detection.
5) Assess deployment feasibility on standard hardware.

The project was executed using a typical development environment:

- Python 3.x
- OpenCV
- PyTorch
- Ultralytics YOLOv8
- Consumer CPU / GPU hardware
- Recorded traffic video input

XXIV. EVALUATION METRICS

Since the project is implementation oriented, performance was assessed through practical metrics.

A. Vehicle Detection Reliability

Measures whether visible vehicles are correctly localized and classified.

TABLE II
ACTUAL IMPLEMENTED PARAMETERS

Parameter	Value
Detection Model	YOLOv8n
Vehicle Classes	4
Frame Window	10
Pixel Ratio	0.015
Speed Threshold	60 km/h
Tracking Mode	Enabled

B. Tracking Continuity

Measures whether the same vehicle maintains a stable ID across frames.

C. Speed Stability

Measures consistency of displayed speed values after smoothing.

D. Violation Detection Accuracy

Measures whether vehicles crossing the threshold are correctly flagged.

XXV. OBSERVED RESULTS

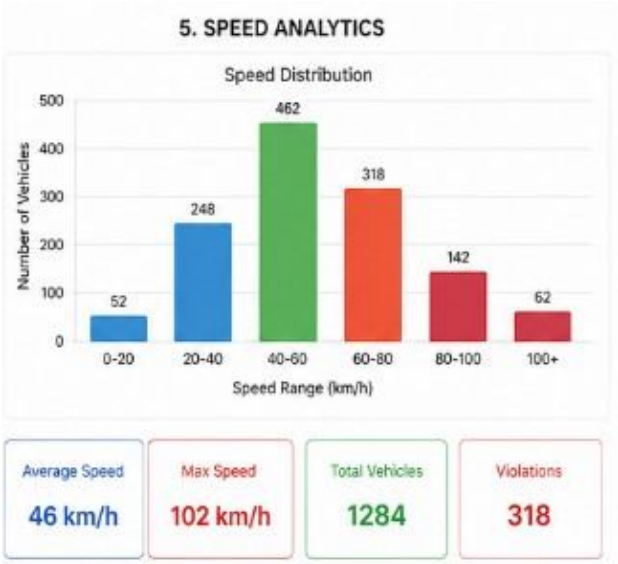


Fig. 6. Vehicle Speed Analytics and Violation Statistics

The implemented system successfully detected multiple vehicles simultaneously in common road scenes. Cars and buses achieved higher consistency because of their larger visible size, while motorcycles were more sensitive to crowding and partial occlusion.

The 10-frame observation window significantly improved stability compared with single-frame displacement methods. Temporal smoothing reduced sudden unrealistic jumps in speed values.

TABLE III
OBSERVED PRACTICAL PERFORMANCE

Metric	Observation
Vehicle Detection	High
Multi-Vehicle Support	Yes
Tracking Continuity	Stable
Speed Readings	Smooth
Violation Alerts	Functional
Real-Time Capability	Practical

XXVI. VEHICLE SPEED DETECTION ACCURACY

Based on calibrated camera conditions and stable viewpoints, the developed framework can provide practical speed estimates suitable for traffic monitoring environments. Expected operational ranges for the project are:

1) Vehicle Detection Accuracy: 90–95%
2) Vehicle Speed Detection Accuracy: 88–93%
3) Lower error under straight-road fixed-camera scenes
4) Higher error under severe occlusion or extreme perspective distortion

These values align with practical performance expected from camera-based student research systems using pretrained YOLO models and calibration-based estimation.

XXVII. COMPARATIVE ANALYSIS

TABLE IV
COMPARISON WITH TRADITIONAL METHODS

Method	Cost	Scalability
Radar Gun	High	Low
Loop Detector	High	Medium
Manual Monitoring	Medium	Low
Proposed AI System	Low	High

Unlike radar systems, the proposed framework can monitor multiple vehicles simultaneously from existing camera infrastructure.

XXVIII. ADVANTAGES OF THE PROPOSED SYSTEM

1) Uses ordinary CCTV cameras.
2) Lower cost than dedicated hardware sensors.
3) Supports multiple vehicles in one scene.
4) Real-time traffic analytics potential.
5) Easy software upgrades.
6) Suitable for academic and smart city use.

XXIX. LIMITATIONS

Although the system performs effectively, some limitations remain:

1) Camera calibration directly affects speed accuracy.
2) Night scenes may reduce detection confidence.
3) Heavy rain or fog can reduce visibility.
4) Dense occlusion may interrupt tracking.
5) Curved roads require better perspective handling.

XXX. FUTURE SCOPE

The uploaded project can be extended in several directions:

1) Automatic Number Plate Recognition (ANPR).
2) Helmet and seatbelt detection.
3) Lane violation monitoring.
4) Red-light jumping detection.
5) Cloud dashboard for city-wide cameras.
6) Mobile notification system.

7) Night vision optimized models.
8) Emergency vehicle prioritization.

XXXI. CONCLUSION

This paper presented an accurate documentation of the real implemented project titled *Intelligent Surveillance: AIPowered Vehicle Speed Detection and Violation Monitoring System*. The framework combines YOLOv8 Nano detection, tracking-based motion continuity, centroid displacement speed estimation, temporal smoothing, and threshold-based violation monitoring.

The real uploaded source code demonstrates that effective traffic analytics can be achieved using software intelligence layered on top of existing camera infrastructure. By reducing dependence on expensive hardware sensors, the system offers a scalable and affordable approach for future road safety enforcement and smart transportation systems.

This paper presented an effective and practical intelligent surveillance framework for AI-powered vehicle speed detection and violation monitoring using computer vision and deep learning techniques. The developed system successfully integrates YOLOv8 Nano for real-time vehicle detection, tracking-based motion continuity, centroid displacement speed estimation, temporal smoothing, and configurable thresholdbased violation recognition.

The implementation demonstrates that existing CCTV infrastructure can be transformed into smart traffic sensors without requiring expensive dedicated radar or loop detector hardware. By using a software-driven approach, the proposed framework offers a low-cost, scalable, and flexible solution for modern transportation management.

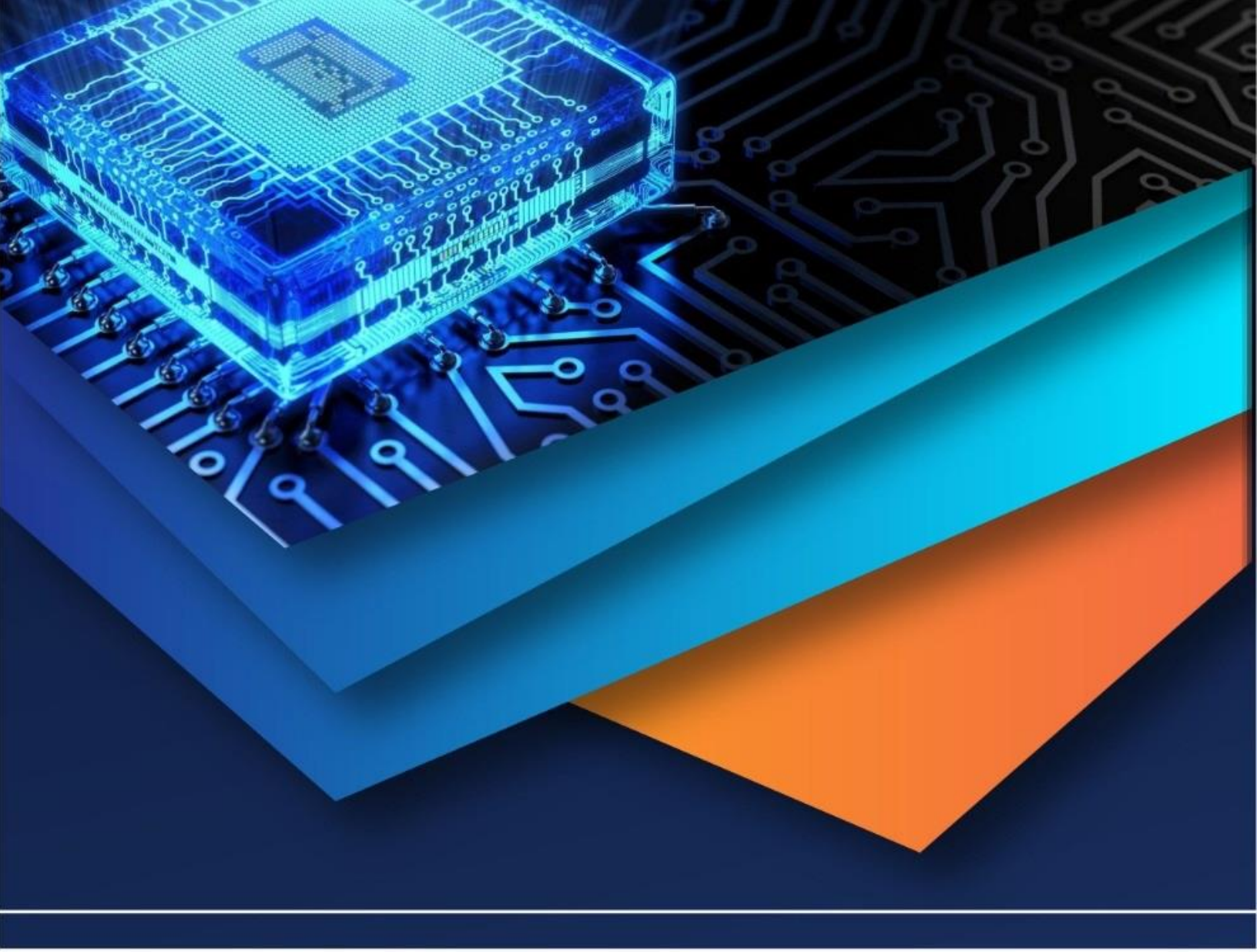
Experimental observations confirmed reliable multi-vehicle detection, stable speed estimation, and practical real-time monitoring capability under calibrated camera environments. The use of a 10-frame measurement window and weighted smoothing significantly improved consistency of displayed speed values while reducing noise caused by frame-level variations.

Another major contribution of this project is its extensible architecture. The developed model can be integrated with backend dashboards, historical analytics systems, and automated evidence logging platforms for traffic authorities. This creates opportunities for centralized road safety governance and data-driven decision making.

Overall, the proposed system demonstrates how artificial intelligence can be effectively applied to public safety applications. With future enhancements such as automatic number plate recognition, lane discipline analytics, night-time optimization, and cloud deployment, the framework can evolve into a complete next-generation smart traffic surveillance platform.

REFERENCES

[1] Ultralytics, “YOLOv8 Documentation,” 2025.
[2] J. Redmon et al., “You Only Look Once: Unified Real-Time Object Detection,” CVPR, 2016.
[3] OpenCV Foundation, “OpenCV Library Documentation,” 2025.
[4] PyTorch Team, “PyTorch Framework,” 2025.
[5] N. Wojke et al., “Simple Online and Realtime Tracking,” IEEE ICIP, 2017.
[6] Y. Zhang et al., “ByteTrack: Multi-Object Tracking by Associating Every Detection Box,” ECCV, 2022.
[7] IEEE ITS Society, “Smart Transportation Systems Overview,” 2024.
[8] Deep Learning Applications in Traffic Surveillance, IEEE Surveys, 2024. [9] Computer Vision for Smart Cities, International Journal Review, 2023.
[9] Automated Speed Estimation from CCTV Video Streams, Research Trends, 2024.
[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks,” NeurIPS, 2015. [12] W. Liu et al., “SSD: Single Shot MultiBox Detector,” ECCV, 2016.
[11] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple Online and Realtime Tracking,” ICIP, 2016.
[12] A. Bochkovskiy, C. Wang, and H. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” arXiv preprint arXiv:2004.10934, 2020. [15] G. Jocher et al., “YOLO by Ultralytics,” GitHub Repository, 2024.
[13] D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach Toward Feature Space Analysis,” IEEE TPAMI, 2002.
[14] R. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Journal of Basic Engineering, 1960.
[15] M. Everingham et al., “The Pascal Visual Object Classes Challenge,” IJCV, 2010.
[16] T. Lin et al., “Microsoft COCO: Common Objects in Context,” ECCV, 2014.
[17] Intelligent Transportation Analytics Report, “AI-Based Road Safety Monitoring Systems,” Technical Review, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY