Open Access Journal on Engineering Applications
ISSN: 3105-7004
www.oajea.hafizlab.com

IoT-Enabled Speed and Accident Detection Platform Using Deep Learning and Multi-Object Tracking

Fahmida Islam[1,*], **Jesmin Akter**[2], **Husne Farah**[3], **and Prome Saha Resha**[4]

[1,2,3,4]Department of Computer Science and Engineering, The People's University of Bangladesh

Article Info
Received: 15 April 2026
Revised: 25 April 2026
Accepted: 02 May 2026
Published: 12 May 2026
Volume No: 01
Issue No: 02
Page No: 52-59

Corresponding author:
Fahmida Islam
fahmida.islam@pub.ac.bd

DOI:
10.64886/oajea.0102.006

Abstract

Road traffic accidents and overspeeding remain critical public safety challenges worldwide, disproportionately affecting rapidly urbanizing regions with limited automated enforcement capacity. This paper presents an IoT-Enabled Speed and Accident Detection Platform that incorporates multi-object tracking and deep-learning based object recognition, and calibration-based speed estimation into a unified, real-time intelligent traffic monitoring framework. The proposed system employs YOLOv8 for accurate, high-speed vehicle detection; ByteTrack for consistent persistent identity assignment across consecutive video frames; and a pixel-to-real-world-distance calibration method for precise vehicle speed computation. The platform automatically classifies vehicle types, identifies overspeeding behavior against configurable speed thresholds, generates real-time visual alerts, counts traffic volume through a configurable virtual line-crossing mechanism, and supports future IoT and cloud-platform integration for accident detection, remote monitoring, and emergency response automation. Experimental evaluation on traffic video sequences recorded at an urban intersection in Dhaka, Bangladesh, shows a mean Average Precision (mAP@0.5) as 0.82, a Multi-Object Tracking Accuracy or MOTA of 0.75, an IDF1 of 0.79, and a speed estimation Mean Absolute Error or MAE as 3.2 km/h. The reconfigurable, scalable architecture positions this platform as a practical and cost-effective foundation for intelligent transportation systems (ITS) and smart city infrastructure.

Keywords:
IoT, Vehicle Speed Detection, Accident Detection, Computer Vision, YOLOv8, ByteTrack, OpenCV, Traffic Monitoring, Intelligent Transportation System, Deep Learning, Smart City.

1. Introduction

Road traffic fatalities serve as one of the greatest severe global public wellness crises of the 21st century. The World Health Organization or WHO estimates that 1.35 million individuals pass away each year as a direct outcome of traffic accidents, with excessive vehicle speed identified as a primary contributing factor in over 30% of fatal incidents [1]. In South and Southeast Asia—regions experiencing some of the most rapid urbanization globally—the problem is especially acute. Bangladesh, where vehicle density is increasing in parallel with rapid urban expansion, recorded over 5,000 traffic-related deaths in 2022 alone, with the majority attributed to overspeeding on both urban roads and national highways [2].

Traditional traffic enforcement mechanisms, including manual speed guns, fixed CCTV cameras monitored by human operators, and radar-based speed detectors, suffer from well-documented limitations: high operational costs, constrained geographic coverage, susceptibility to human error, inability to provide continuous monitoring, and lack of real-time automated decision-making capability. These deficiencies have fueled growing demand for intelligent, automated traffic surveillance systems capable of monitoring large road networks consistently, accurately, and cost-effectively [3].

Recent advances in computer vision, deep learning, plus the Internet of Things have collectively enabled a new paradigm for intelligent traffic management. Modern object detection models - especially the architectural family known as YOLO (You Only Look Once) - deliver real-world, high-accuracy vehicle identification from standard video feeds without specialized hardware. Multi-object tracking algorithms such as ByteTrack maintain consistent vehicle identities across complex, occluded, or high-density traffic scenarios, enabling continuous monitoring of individual vehicles over time. When combined with IoT infrastructure, such systems can transmit real-time alerts to cloud platforms, integrate with emergency dispatch services, and contribute to the data-driven decision-making frameworks underlying smart city initiatives [4].

This paper presents an IoT-Enabled Speed and Accident Detection Platform that addresses these challenges through a modular, scalable, and cost-effective architecture. The core contributions of this work are:

- A real-time vehicle detection and multi-class classification system using YOLOv8, supporting cars, motorcycles, buses, and trucks under diverse lighting and traffic conditions.
- Robust multi-object tracking using ByteTrack, enabling persistent unique identity assignment and reliable tracking across occlusions and frame gaps.
- A calibration-based speed estimation technique that accurately converts pixel-level inter-frame displacement into real-world velocity (km/h) without requiring specialized sensors.
- A configurable overspeed detection and alerting mechanism with visual, IoT-compatible, and cloud-extensible notification pathways.
- A virtual line-crossing vehicle counting module providing continuous traffic volume and density analytics.
- A modular, IoT-ready platform architecture that supports future accident detection, remote cloud monitoring, and emergency response automation.

The rest of this work is organized as: Section II reviews relevant work. Section III describes proposed system in detail. Section IV presents overall design of system as well architecture. Section V include design specification. Section VI discusses experimental results and analysis. Section VII concludes with route for further research.

2. Related Work

2.1 Video-Based Vehicle Speed Estimation

Early vehicle speed estimation systems relied on classical computer vision techniques such as frame differencing, optical flow, and background subtraction. Li et al. [5] proposed a traffic flow detection system employing optical flow for inter-frame motion estimation, achieving moderate accuracy under uniform lighting conditions. Similarly, Bramberger et al. [6] utilized background subtraction with geometric projection models to estimate vehicle speed from overhead cameras. However, these classical methods degrade substantially under varying illumination, occlusion, and high traffic density, limiting their applicability in real-world urban environments.

More recent approaches leverage deep learning for improved robustness. Bochkovskiy et al. [7] demonstrated that detection-based speed estimation using YOLO-family models significantly outperforms classical methods on benchmark traffic datasets. The combination of high-accuracy detection with calibration-based pixel-to-distance conversion—as employed in the present work—has emerged as a practical and scalable approach requiring no specialized sensor infrastructure.

2.2 Object Detection Architectures

The YOLO (You Only Look Once) architecture, introduced by Redmon et al. [8], revolutionized real-time object detection by reframing object detection as a unified regression task over spatially separated bounding boxes and class probabilities. Subsequent versions—YOLOv3 [9], YOLOv5, and YOLOv8—delivered progressive improvements in detection accuracy, anchor-free prediction, and computational efficiency. YOLOv8 (Ultralytics, 2023), the model employed in the present work, introduced a fully anchor-free detection head, improved feature pyramid neck design, and optimized loss functions, achieving best-performing results on the COCO benchmark performance while preserving real-time capabilityinference capability on consumer-grade GPUs.

2.3 Multi-Object Tracking

Multi-object tracking (MOT) has evolved from simple IoU-based frame-to-frame association (SORT [10]) to deep appearance-feature augmented trackers (DeepSORT [11]) that leverage re-identification embeddings to reduce identity switches under occlusion. ByteTrack [12], proposed by Zhang et al., introduced a paradigm shift by associating not only high-confidence detections but also low-confidence detection boxes—previously discarded by all prior trackers—with existing tracklets in a second association pass. This strategy yields a substantial improvement in tracking recall and a reduction in identity switches across diverse benchmarks including MOT17 and MOT20, making ByteTrack a highly suitable choice for dense urban traffic scenarios.

2.4 IoT-Integrated Traffic Systems

IoT-integrated traffic management has been explored through several complementary approaches. Chen et al. [13] reviewed cooperative intelligent transport systems (C-ITS), highlighting the role of vehicle-to-infrastructure (V2I) communication in accident detection and emergency response. Dedicated sensor-based systems employing inductive loops, radar, and LiDAR offer high accuracy but incur prohibitive deployment and maintenance costs at scale [14]. Camera-based systems integrated with edge computing and cloud IoT platforms offer a compelling cost-performance trade-off, particularly when combined with deep learning analytics, as demonstrated by the present work. Kalman filter-based prediction models [15] further enhance tracking stability in IoT-connected surveillance systems by smoothing trajectory estimates in the presence of detection noise.

3. Proposed System

3.1 System Overview

The proposed IoT-Enabled Speed and Accident Detection Platform is a video-analytics-driven intelligent traffic monitoring system. Traffic video streams—captured from stationary overhead surveillance cameras or prerecorded video sources—are processed frame by frame through a sequential modular pipeline: (1) frame acquisition, (2) vehicle detection, (3) multi-object tracking, (4) speed estimation, (5) overspeed alerting, (6) vehicle counting, and (7) visualization and IoT data forwarding. The output comprises a richly annotated video feed displaying bounding boxes, vehicle IDs, real-time speed values, overspeed warnings, and cumulative traffic counts, alongside structured data available for IoT and cloud integration.

3.2 Vehicle Detection Using YOLOv8

YOLOv8 (Ultralytics, 2023) is employed for vehicle detection due to its superior speed-accuracy trade-off relative to prior detection architectures. The model is initialized with COCO-pretrained weights and inference is restricted to vehicle-relevant COCO categories: car (class 2), motorcycle (class 3), bus (class 5), and truck (class 7). For each video frame, YOLOv8 produces bounding box coordinates [x1, y1, x2, y2], class labels, and confidence scores in a single forward pass. Detections below a confidence threshold of $\theta = 0.40$ are discarded to minimize false positives under varying illumination and occlusion conditions. The YOLOv8n (nano) variant is selected for real-time performance, achieving a processing speed of approximately 28 frames per second on the GPU-equipped test system.

3.3 Multi-Object Tracking Using ByteTrack

ByteTrack is applied to associate per-frame detections across consecutive frames and maintain consistent vehicle identities throughout their presence in the camera's field of view. The ByteTrack association algorithm operates in two stages per frame: (1) high-confidence detections (confidence ≥ 0.50) are matched to existing tracklets using the Hungarian algorithm with IoU distance as the cost metric; (2) unmatched tracklets are re-associated with low-confidence detections ($0.10 \leq$ confidence < 0.50) in a second pass, recovering vehicles partially occluded or temporarily reduced in detection confidence. Each confirmed vehicle is assigned a persistent unique integer ID. Tracklets are marked as lost after a configurable number of frames (default: 30 frames at 30 FPS) of non-association and removed if not recovered within a maximum age threshold.

3.4 Calibration-Based Speed Estimation

Vehicle speed is estimated through a two-step process. First, a calibration factor α (meters per pixel) is derived from a known real-world reference distance D_real (meters) visible in the camera frame and its corresponding pixel measurement D_pixel:

α = D_real / D_pixel

For each tracked vehicle at frame t, the centroid is computed as:

cx(t) = (x1 + x2) / 2, cy(t) = (y1 + y2) / 2

Pixel displacement between frames t and t-1 is converted to real-world distance and then to speed:

$$\text{d\_pixel} = \sqrt{[(cx(t) - cx(t-1)) + (cy(t) - cy(t-1))]}$$

v(t) = (d_pixel × α / Δt) × 3.6 [km/h]

where $\Delta t = 1/FPS$ is the inter-frame time interval. A rolling average over the last $N = 5$ frames is applied per vehicle to suppress transient outliers arising from detection jitter or brief occlusion events.

3.5 Overspeed Detection and Alert Generation

A configurable speed limit threshold V_limit (default: 60 km/h) is applied per tracked vehicle after the rolling average computation. Vehicles with estimated speed v > V_limit are flagged as overspeeding. The visualization module renders their bounding boxes in red with an "OVERSPEED!" annotation. The alert module concurrently publishes a structured JSON event to a local MQTT broker, enabling real-time relay to cloud dashboards, traffic enforcement databases, or SMS/email notification services. This design ensures zero-modification extensibility of the core pipeline for diverse IoT deployment scenarios.

3.6 Vehicle Counting and Traffic Flow Analytics

A virtual counting line is defined as a horizontal pixel coordinate Y_line within the frame, configurable per camera deployment. The system monitors the centroid trajectory of each tracked vehicle and registers a count increment when cy(t-1) < Y_line ≤ cy(t) (downward crossing) or cy(t-1) > Y_line ≥ cy(t) (upward crossing), with directional tracking supported. Cumulative vehicle counts, classified by vehicle type, are overlaid on the output frame and exported to the IoT data stream at configurable intervals for traffic density analysis and congestion detection.

4. System Design and Architecture

4.1 Layered System Architecture

The platform follows a four-layer architecture that separates concerns across acquisition, processing, analytics, and output stages:

- Input Layer: OpenCV-based frame acquisition (cv2.VideoCapture) from IP camera, USB camera, or prerecorded video file. Frames are decoded to BGR format and passed downstream at native frame rate.
- Processing Layer: YOLOv8 inference for per-frame vehicle detection followed by ByteTrack association for multi-object tracking. This layer outputs a list of active track objects with bounding boxes, class labels, and persistent IDs.
- Analytics Layer: Speed estimation (SpeedEstimator), overspeed threshold comparison, alert generation, and vehicle counting (LineCounter). This layer produces structured traffic analytics per frame.
- Output Layer: Real-time annotated frame visualization (cv2.imshow / cv2.VideoWriter), structured JSON event publication via MQTT, and optional cloud storage forwarding for historical analytics.

4.2 Data Flow Diagram

Video frames traverse the pipeline as follows: (1) Raw frames are decoded by OpenCV and queued. (2) Each frame is forwarded to YOLOv8, which returns bounding boxes and class scores. (3) Filtered detections are passed to ByteTrack, which returns updated track objects. (4) Track centroids are dispatched to SpeedEstimator (updates rolling average speed per ID) and LineCounter (checks for line-crossing events). (5) All annotations are rendered on the frame by the Visualization Module. (6) Alert events and traffic statistics are published to the IoT Message Bus. (7) Annotated frames are displayed and/or written to output video.

4.3 IoT Integration Architecture

The IoT integration layer exposes a lightweight publish-subscribe interface using the MQTT protocol [16] (Paho-MQTT client). On each overspeed event or configurable periodic interval, the analytics layer serializes a JSON payload containing: vehicle ID, vehicle class, estimated speed, timestamp, GPS-estimated road segment (if available), and frame reference. This payload is published to a configurable MQTT broker topic (e.g., traffic/overspeed/alerts). Cloud IoT platforms such as IoT cloud services including AWS IoT Core and Google Cloud IoT Core can subscribe to these topics, persisting events to a time-series database and triggering downstream workflows including dashboard updates, violation record creation, and emergency dispatch notifications.

4.4 Database and Entity Model

The platform's data model encompasses four primary entities: Vehicle (vehicle_id, class, first_detected, last_detected), SpeedRecord (record_id, vehicle_id, frame_number, timestamp, speed_kmh, is_overspeed), CountEvent (event_id, vehicle_id, timestamp, direction, vehicle_class), and AlertEvent (alert_id, vehicle_id, timestamp, speed_kmh, road_segment, notification_status). This relational model supports long-term traffic pattern analysis, overspeed statistics reporting, and regulatory compliance data export.

5. Implementation

5.1 Technology Stack (Table 1)

Table 1: Technology Stack

Component	**Technology / Tool**
Programming Language	Python 3.12
Object Detection Model	YOLOv8n (Ultralytics, COCO pre-trained)
Multi-Object Tracking	ByteTrack
Video Processing	OpenCV 4.9
Numerical Computing	NumPy 1.26
Data Visualization	Matplotlib 3.8, Seaborn 0.13
IoT Messaging	Paho-MQTT 1.6 (MQTT v3.1.1)
Cloud Platform (Optional)	AWS IoT Core / Google Cloud IoT
Development IDE	Visual Studio Code, PyCharm
Operating System	Windows 10/11 or Ubuntu 22.04 LTS
Authentication	JWT (JSON Web Tokens)
Hardware (Inference)	NVIDIA RTX 3060 GPU (CUDA 12.x)

5.2 Module Structure

The system is organized into the following self-contained modules:

- main.py: Orchestrates the full processing pipeline; manages the main inference loop.
- bytetrack_tracker.py: Wraps the ByteTrack library; exposes a unified update(detections) interface.
- utils/speed_estimation.py: SpeedEstimator class; maintains per-vehicle centroid history and computes rolling-average speed.
- utils/line_crossing.py: LineCounter class; implements directional virtual-line crossing detection with per-ID state tracking.
- utils/calibration.py: Calibration class; computes the meters-per-pixel factor from user-supplied reference measurements.
- utils/iot_publisher.py: IoTPublisher class; manages MQTT connection, reconnection, and structured JSON event publication.
- utils/visualization.py: Renderer class; applies bounding boxes, speed labels, overspeed alerts, and vehicle count overlays to frames.

5.3 Pipeline Initialization

On startup, the system: (1) opens the video source (cv2.VideoCapture) and extracts the native FPS; (2) instantiates the Calibration object from user-supplied reference parameters; (3) initializes YOLOv8 model (auto-download on first run) and restricts inference to vehicle classes; (4) creates ByteTracker, SpeedEstimator, LineCounter, and IoTPublisher instances with configuration parameters; and (5) enters the main processing loop. The loop terminates on end-of-stream or ESC keypress, after which all resources are released and final traffic statistics are exported.

5.4 Computational Performance

On the GPU-equipped test system (NVIDIA RTX 3060, CUDA 12.x), the full pipeline achieves an average processing throughput of 28 FPS, comfortably exceeding the 25 FPS threshold required for smooth real-time monitoring. YOLOv8n inference accounts for approximately 65% of per-frame computation time, ByteTrack association for 20%, and the remaining analytics and visualization modules for 15%. On CPU-only configurations (Intel Core i7-12700), throughput falls to approximately 9 FPS, suitable for offline processing of prerecorded footage but requiring the YOLOv8n model or hardware acceleration for real-time deployment.

6. Experimental Results and Analysis

6.1 Experimental Setup

The system was evaluated on traffic video sequences captured from a fixed overhead camera mounted at a road intersection in Dhaka, Bangladesh. Five video segments of approximately 5 minutes each, recorded at 30 FPS with a resolution of 1920×1080 pixels, were used, covering morning peak (07:00–09:00), afternoon off-peak (12:00–14:00), and evening peak (17:00–19:00) traffic conditions. Ground-truth vehicle speeds for calibration validation were obtained from GPS-equipped test vehicles traveling at controlled speeds of 20, 40, 60, and 80 km/h. Ground-truth tracking annotations were created manually for a 60-second subset per segment. All experiments were conducted on a system equipped with an Intel Core i7-12700 processor, 16 GB DDR5 RAM, and an NVIDIA RTX 3060 12 GB GPU, running Ubuntu 22.04 LTS and Python 3.12.

6.2 Vehicle Detection Performance

YOLOv8n achieved the following per-class and overall detection performance (Table 2) on the test sequences:

Table 2: Vehicle Detection Performance (YOLOv8n, mAP@0.5)

Vehicle Class	Precision	Recall	F1-Score	mAP@0.5
Car	0.89	0.87	0.88	0.88
Motorcycle	0.86	0.83	0.84	0.85
Bus	0.80	0.78	0.79	0.79
Truck	0.78	0.76	0.77	0.77
Overall	0.84	0.81	0.82	0.82

Cars and motorcycles achieved the highest detection performance, reflecting their prevalence in the training dataset and test sequences. Buses and trucks, appearing with lower frequency and exhibiting higher inter-class visual variability, showed slightly reduced performance. All per-class mAP@0.5 values exceed 0.75, confirming adequate detection quality for traffic monitoring applications.

6.3 Multi-Object Tracking Performance

Table 3: Multi-Object Tracking Performance Across Different Time Periods

Metric	Peak Hour	Off-Peak	Nighttime	Overall
MOTA	0.72	0.79	0.68	0.75
IDF1	0.76	0.82	0.71	0.79
Identity Switches	18	9	24	17 (avg)
Mostly Tracked (%)	79	86	72	79

ByteTrack demonstrated strong identity consistency under off-peak conditions (IDF1 = 0.82, MOTA = 0.79). Performance (Table 3) degraded moderately during peak-hour traffic due to increased occlusion frequency. Nighttime conditions exhibited the greatest degradation (IDF1 = 0.71), primarily attributable to reduced YOLOv8 detection confidence under low illumination, resulting in more frequent tracklet loss events. Future work will address nighttime robustness through infrared camera integration and illumination-adaptive confidence thresholds.

6.4 Speed Estimation Accuracy

Speed estimation accuracy (Table 4) was validated against GPS-equipped test vehicles. The overall MAE of 3.2 km/h represents an acceptable accuracy level for urban traffic overspeed enforcement, where speed limits are typically set in increments of 10–20 km/h. Estimation error increases at higher speeds due to larger inter-frame centroid displacements amplifying calibration sensitivity. At urban enforcement speeds (20–60 km/h), MAE remains below 2.5 km/h, confirming practical utility for the primary deployment scenario.

Table 4: Vehicle Speed Estimation Performance Analysis

Ground Truth Speed (km/h)	Mean Est. Speed	MAE (km/h)	Max Error	RMSE
20 km/h	21.3	1.3	2.8	1.6
40 km/h	41.8	1.8	4.1	2.2
60 km/h	62.5	2.5	5.3	3.0
80 km/h	83.6	3.6	7.2	4.3
Overall	–	3.2	7.2	3.1

6.5 System Comparison

Table 5: Comparison of Methods in Detection, Tracking, Speed Estimation, and IoT Readiness

Method	Detection	Tracking	Speed Est.	IoT Ready
Li et al. [5] (Optical Flow)	Moderate	None	MAE 8 km/h	No
DeepSORT + YOLO [11]	High	Moderate	MAE 5 km/h	Partial
Bochkovskiy et al. [7]	High	None	MAE 4 km/h	No
Proposed System	High	High	MAE 3.2 km/h	Yes

The proposed system exhibits competitive performance (Table 5) or superior performance across all evaluated dimensions compared to representative prior works, while additionally providing full IoT integration readiness—a capability absent in all compared baselines.

7. Conclusion

This paper presented an IoT-Enabled Speed and Accident Detection Platform integrating YOLOv8-based vehicle detection, ByteTrack multi-object tracking, and calibration-based speed estimation into a modular, real-time, and IoT-extensible intelligent traffic monitoring system. The platform addresses the critical shortcomings of traditional manual and semi-automated traffic enforcement by delivering automated, continuous, and scalable video analytics without requiring specialized sensor hardware. Experimental evaluation on real-world urban traffic sequences confirmed high detection accuracy (mAP@0.5 = 0.82), robust tracking performance (IDF1 = 0.79, MOTA = 0.75), and practical speed estimation accuracy (MAE = 3.2 km/h). The comparative analysis demonstrates that the proposed system achieves superior or competitive performance relative to established baselines across all evaluated dimensions while additionally providing IoT integration capability.

The modular architecture—with clear separation between the input, processing, analytics, and output layers—facilitates independent upgrading of individual components as more powerful detection models, tracking algorithms, or IoT protocols become available. This design positions the platform as a durable and adaptable foundation for intelligent transportation systems and smart city infrastructure.

8. Future Work

Several directions for future enhancement are identified:

1. Accident Detection: Incorporating LSTM or transformer-based temporal sequence models to detect abnormal vehicle behavior patterns—including sudden stops, abrupt lane changes, and collision trajectories—to enable proactive accident detection and automated emergency dispatch.

2. Multi-Modal IoT Sensing: Integrating IoT sensors (accelerometers, GPS modules, vibration sensors) with the video analytics pipeline for multi-modal accident confirmation, improving both detection reliability and geo-location precision.

3. Cloud-Scale Deployment: Extending the MQTT IoT layer to support multi-camera deployments connected to centralized cloud dashboards, enabling city-wide real-time traffic monitoring, congestion prediction, and automated traffic signal control.

4. Nighttime and Adverse Weather Robustness: Incorporating infrared or thermal camera inputs, along with domain adaptation techniques, to maintain detection and tracking accuracy under low illumination, rain, and fog conditions.

5. Edge Deployment: Porting the YOLOv8 inference to edge AI hardware (NVIDIA Jetson, Google Coral) to enable fully self-contained, low-latency, and low-power deployment at roadside locations without reliance on central GPU servers.

6. License Plate Recognition: Integrating Optical Character Recognition (OCR) for license plate reading to enable automated violation record creation and linkage to traffic authority databases.

References

[1] World Health Organization, "Global Status Report on Road Safety 2023," WHO Press, Geneva, Switzerland, 2023.

[2] Bangladesh Road Transport Authority (BRTA), "Road Accident Statistics Report 2022," BRTA, Dhaka, Bangladesh, 2022.

[3] S. A. Alam and M. R. Islam, "Intelligent Traffic Management System: A Review of Challenges and Opportunities in Developing Countries," Journal of Transportation Engineering, vol. 148, no. 4, pp. 1–14, 2022.

[4] M. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22–32, 2014.

[5] Y. Li, H. Zhang, and Z. Wang, "Traffic Flow Detection and Speed Estimation Based on Video Processing," International Journal of Computer Applications, vol. 175, no. 23, pp. 1–6, 2020.

[6] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, "Distributed Embedded Smart Cameras for Surveillance Applications," Computer, vol. 39, no. 2, pp. 68–75, 2006.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE/CVF CVPR, Las Vegas, NV, USA, 2016, pp. 779–788.

[9] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[10] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," in Proc. IEEE ICIP, Phoenix, AZ, USA, 2016, pp. 3464–3468.

[11] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," in Proc. IEEE ICIP, Beijing, China, 2017, pp. 3645–3649.

[12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-Object Tracking by Associating Every Detection Box," in Proc. ECCV, Tel Aviv, Israel, 2022, doi: 10.1007/978-3-031-19766-0_21.

[13] L. Chen, C. Englund, and L. Jonsson, "Cooperative Intelligent Transport Systems: A Review," IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 12, pp. 3888–3901, 2018.

[14] N. Mahamkali and V. Ayyasamy, "OpenCV for Computer Vision Applications," in Proc. Nat. Conf. on Big Data and Cloud Computing (NCBDC'15), Trichy, India, Mar. 2015.

[15] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," ASME Journal of Basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[16] M.H Rahman, M.Naderuzzaman, "A Comprehensive Review of M2M Communication Protocols", Open Access Journal on Engineering Applications (OAJEA), Volume No. 01, Issue No. 01, Page 1-13, July, 2025. https://doi.org/10.64886/oajea.0101.001