



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85098>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Edge AI-Based Vehicle Surveillance System

Sreedevi S¹, Sukhi S Gowda², Sinchana H R³, Sushruth R⁴, Srusti U⁵

Department of Computer Science and Engineering, JNN College of Engineering, Shivamogga, Karnataka, India

Abstract: Urban vehicle numbers have grown far faster than the capacity of traffic police to watch every intersection and parking lane, and the result is a steady stream of hit-and-run incidents, wrong-parking, and double-parking cases that pass unrecorded under conventional patrol- and CCTV-based enforcement. This paper describes an Edge AI vehicle-surveillance pipeline built to close that gap while running entirely on low-cost, offline-capable hardware — a laptop or a Raspberry Pi — with no cloud backend and no purpose-built sensors. Vehicles are localised and classified frame by frame with YOLOv8, and a DeepSORT tracker then follows each detected vehicle through the sequence so that its position can be monitored continuously. A collision is inferred when two tracked bounding boxes overlap sharply, measured through Intersection over Union, together with an abrupt motion change captured by Lucas–Kanade optical flow; if the vehicle involved does not remain at the scene within a five-second grace period, the event is recorded as a hit-and-run. Separately, restricted-zone polygons combined with a configurable stay-duration check are used to catch vehicles parked illegally or blocking a live lane. Whenever any of these violations is confirmed, EasyOCR reads the offending number plate, and the system saves a short evidentiary clip, a still image, and a PDF summary before pushing an SMS, an email, and a dashboard notification to the relevant personnel. Implemented purely in Python and OpenCV, the resulting system is inexpensive to deploy at intersections, parking lots, and campus checkpoints.

Keywords: Edge AI; Vehicle Surveillance; YOLOv8; DeepSORT; Hit-and-Run Detection; Automatic License Plate Recognition; Parking Violation Detection; Optical Flow; Traffic Law Enforcement.

I. INTRODUCTION

As cities grow and the number of registered vehicles keeps climbing, keeping every stretch of road, parking bay, and intersection under continuous watch has become impractical for human enforcement alone. Thousands of vehicles pass through any given junction or campus road on an ordinary day, and inevitably some of them break the rules while no officer or camera operator happens to be looking at that exact moment. Three violation types are especially consequential here: a driver who causes a collision and leaves before help arrives or blame is assigned, commonly termed hit-and-run; a vehicle left standing in a restricted zone that blocks ambulances, fire trucks, or the general flow of traffic, termed wrong parking; and a vehicle stopped alongside one that is already parked, closing off a live lane and inviting rear-end collisions, termed double parking. Foot patrols and passively recorded CCTV footage do little to change this picture, since the footage is reviewed after the fact, if it is reviewed at all, by which time the offending vehicle is long gone and little can be done.

What has changed this picture is the maturity of deep-learning-based computer vision. A single frame can now be scanned for every vehicle it contains — car, motorcycle, bus, or truck — by a detector such as YOLOv8 running at real-time speed; DeepSORT can then stitch those per-frame detections into continuous trajectories, giving each vehicle a stable identity as it moves through the scene; and an OCR engine such as EasyOCR can pull the text off a number plate in well under a second. None of this requires anything more powerful than an ordinary laptop or a Raspberry Pi, and none of it depends on a live internet connection. The stakes are not small: road-traffic collisions remain among the leading causes of accidental death worldwide, and in many regions — India among them — the number of officers available per registered vehicle is far too low to catch more than a small fraction of violations as they happen. Victims of hit-and-run incidents are frequently left without prompt aid and without evidence strong enough to support a prosecution, while wrong- and double-parked vehicles quietly erode both safety and traffic flow every single day.

Yet most surveillance already installed in Indian cities and institutions does little beyond passive recording. It cannot, on its own, spot a hit-and-run as it happens and pull the fleeing vehicle's plate; it does not log wrong- or double-parking with any supporting evidence; it never alerts a guard or a traffic officer automatically; and almost none of it is designed to run on affordable, edge-capable hardware free of cloud servers and dedicated sensors. The system proposed here is built to close precisely that set of gaps. Written in Python and OpenCV, it brings together real-time vehicle detection, continuous multi-object tracking, hit-and-run behaviour analysis, parking-violation detection, automatic plate recognition, evidence preservation, and multi-channel alerting into one pipeline that a standard laptop or Raspberry Pi can run using nothing more than a webcam feed.

The goals set for this system can be summarised as follows: recognise and classify vehicles frame by frame with YOLOv8; keep a persistent, Kalman-filtered identity on every tracked vehicle using DeepSORT; catch collisions by combining Intersection-over-Union overlap with Lucas–Kanade optical flow, and mark an involved vehicle as a hit-and-run offender if it does not stay at the scene within a configurable time window; spot wrong- and double-parking through polygon-bounded restricted zones paired with dwell-time checks; pull the offending plate automatically with EasyOCR; keep a video clip, a snapshot, and an auto-built PDF report as evidence for every confirmed incident; push out SMS, email, and dashboard alerts the moment a violation is confirmed; and do all of this on edge hardware with nothing beyond Python and OpenCV, avoiding cloud servers and specialised sensors altogether. Detecting red-light running or overspeeding, and any use of depth or stereo cameras, is left for a later phase of the project.

II. LITERATURE SURVEY

Twelve recent studies spanning edge-based enforcement, tracking, parking detection, plate recognition, and collision analysis were examined to position the present work. Charran and Dubey [1] place small perception units near intersections so that video is processed on the spot and only decisions, not raw footage, travel between units and connected vehicles over V2X; the design cuts both delay and bandwidth use relative to cloud-centred alternatives, but keeping many such units synchronised when the network is unreliable, and the expense of covering an entire city with them, remain unsolved problems. Patil and Ghule [2] pair edge inference with IoT sensor data to catch speeding, lane-crossing, and signal-jumping, and find that adding sensor readings on top of the camera feed sharpens overall awareness of what is happening on the road, though the extra hardware raises both installation and upkeep cost. Taking a different route, Scandurra et al. [3] mount AI-capable cameras on patrol vehicles themselves, trading the fixed footprint of a stationary camera for mobility that lets a single unit cover several streets in a shift; the price paid is motion blur, camera shake, and lighting that changes from block to block, all of which chip away at detection reliability.

A second cluster of work concentrates on detection and tracking quality. Anisha et al. [4] join YOLOv8 to DeepSORT and Kalman filtering in a pipeline they call DeepYOLO, and show it keeps a vehicle's identity intact through brief occlusion, though the GPU load it carries makes low-power edge boards a poor fit. Djarah et al. [5] go a step further by folding optical-flow-based motion estimates into a YOLOv9/DeepSORT pipeline, which cuts down identity switching in busy scenes at the cost of extra computation that modest hardware may struggle to sustain frame-by-frame. A narrower study [6] applies the same YOLOv8/DeepSORT combination purely to counting vehicles and estimating flow for city traffic management; it holds up well in clear conditions but loses accuracy in fog, heavy rain, or after dark.

Parking enforcement is addressed separately in two studies. Sharma et al. [7] track how long a vehicle occupies a spot and raise a flag once it exceeds an allowed duration, replacing what would otherwise be a manual walk-through, but the method has no way to tell an authorised vehicle from an unauthorised one unless paired with an outside database, and it is thrown off by anything the camera cannot see. Kathait et al. [8] build a similar computer-vision system aimed specifically at illegal parking in no-parking zones and live lanes, automatically generating evidence usable for a citation, though it inherits the same weakness of blind spots and occlusion in busy traffic.

Plate recognition is the focus of two further studies. One survey of deep-learning ALPR methods [9] finds that pairing a detector with an OCR stage reads plates accurately across a wide range of styles and lighting, but accuracy falls once a plate is dirty, damaged, or partly hidden. A newer pipeline built on YOLOv12 and PaddleOCR [10] pushes recognition accuracy close to perfect and supports several languages at real-time speed, but only with the help of accelerators such as an NVIDIA Jetson board, which puts it out of reach of lighter edge hardware. Finally, two studies deal with collisions directly. Adewopo and Elsayed [11] run an ensemble of deep networks to spot accidents and fire off emergency notifications automatically, shortening response time overall, though dense congestion is sometimes mistaken for a crash. An earlier study on optical-flow-based side-collision detection [12] shows that motion patterns between consecutive frames are enough to catch an abnormal impact reliably, but it treats collision detection as a standalone problem rather than one stage within a full surveillance system.

Taken together, these twelve studies trace a shift from YOLOv8 toward YOLOv12 for detection, growing reliance on optical flow to stabilise DeepSORT tracking, and an increasing tendency to specialise — one system for parking, another for plates, another for accidents — rather than to unify. None of them combines edge-friendly detection, resilient tracking, hit-and-run behaviour analysis, more than one class of parking violation, automatic plate reading, evidence preservation, and multi-channel alerting into a single low-power pipeline; most either assume access to a capable GPU or handle only one type of violation. It is this combination that the present work sets out to provide.

III. PROPOSED METHODOLOGY

Nine stages make up the proposed pipeline: reading the video, detecting vehicles, tracking them, analysing collisions for hit-and-run behaviour, checking parking zones, reading plates, preserving evidence, sending alerts, and logging everything to a dashboard and database. Chained together, these stages let an ordinary webcam feed on a laptop or Raspberry Pi turn into an evidence-backed violation alert without leaning on any cloud service for the core detection work.

A. Video Acquisition and Preprocessing

OpenCV reads incoming video — live or pre-recorded, from an ordinary monocular camera — one frame at a time. Every frame is resized and normalised before anything else happens to it, and the areas that matter for later checks, such as a traffic lane, a junction, or a marked parking bay, are drawn out as regions of interest ahead of time.

B. Vehicle Detection using YOLOv8

Every preprocessed frame goes through a YOLOv8 model, which in one pass returns a bounding box, a class label, and a confidence score for each car, motorcycle, truck, or bus it finds. YOLOv8 was picked here because it strikes a workable balance between detection accuracy and inference speed on hardware that has neither a server-grade GPU nor unlimited power, which matters if the pipeline is to keep up with a live feed on a laptop or Pi.

C. Multi-Object Tracking using DeepSORT

Once vehicles are detected, DeepSORT takes over the job of keeping track of who is who from one frame to the next, combining a Kalman-filter motion model with an appearance descriptor so that an identity survives a short occlusion — a vehicle briefly hidden behind another at a junction, for instance, or in a packed parking row — without being reassigned a new ID.

D. Collision and Hit-and-Run Detection

A collision is suspected whenever two tracked bounding boxes suddenly overlap by an amount measured through Intersection over Union, and this signal is checked against Lucas–Kanade optical flow, which should show the same abrupt change in motion if an actual impact occurred. Whenever both checks agree that a collision has taken place, the vehicles involved are watched for a short grace period — set to five seconds in the current build — and any vehicle that pulls away within that window rather than stopping is recorded as having committed a hit-and-run.

E. Wrong-Parking and Double-Parking Detection

No-parking areas and active lanes are marked out as polygons that can be redrawn for each camera view. A tracked vehicle whose position sits inside one of these no-parking polygons is flagged as wrongly parked, and a vehicle that stays inside a live lane next to an already-parked vehicle for longer than a configurable dwell-time limit is flagged for double parking instead.

F. Automatic License Plate Recognition (ALPR)

As soon as a violation is confirmed, the region of the frame containing the offending vehicle's plate is cropped out and handed to EasyOCR, which reads off the alphanumeric text. That reading is then attached to the incident record so the vehicle can be identified afterward.

G. Evidence Collection and Reporting

Each confirmed incident triggers automatic saving of a ten-second clip that brackets the event, a still frame of the vehicle involved, and a PDF report carrying the timestamp, the violation type, and the plate reading, so that every alert sent out is backed by something a human can review afterward.

H. Real-Time Alert Dispatch

Once a violation is confirmed, three notifications go out almost simultaneously: an SMS through Twilio, an email through SMTP, and an in-app notice on a Flask web dashboard, letting whoever is on duty respond without needing to watch the live feed at every moment.

I. Dashboard and Database

A Flask-based dashboard shows the live camera view alongside a running log of past incidents and alerts, with an SQLite database underneath holding every record for later lookup. Since detection, tracking, and violation analysis all run locally without needing a network connection, the only point at which the system needs to go online is to send out an alert — which keeps the whole setup cheap enough for an intersection, a parking lot, or a campus checkpoint.

IV. CONCLUSION

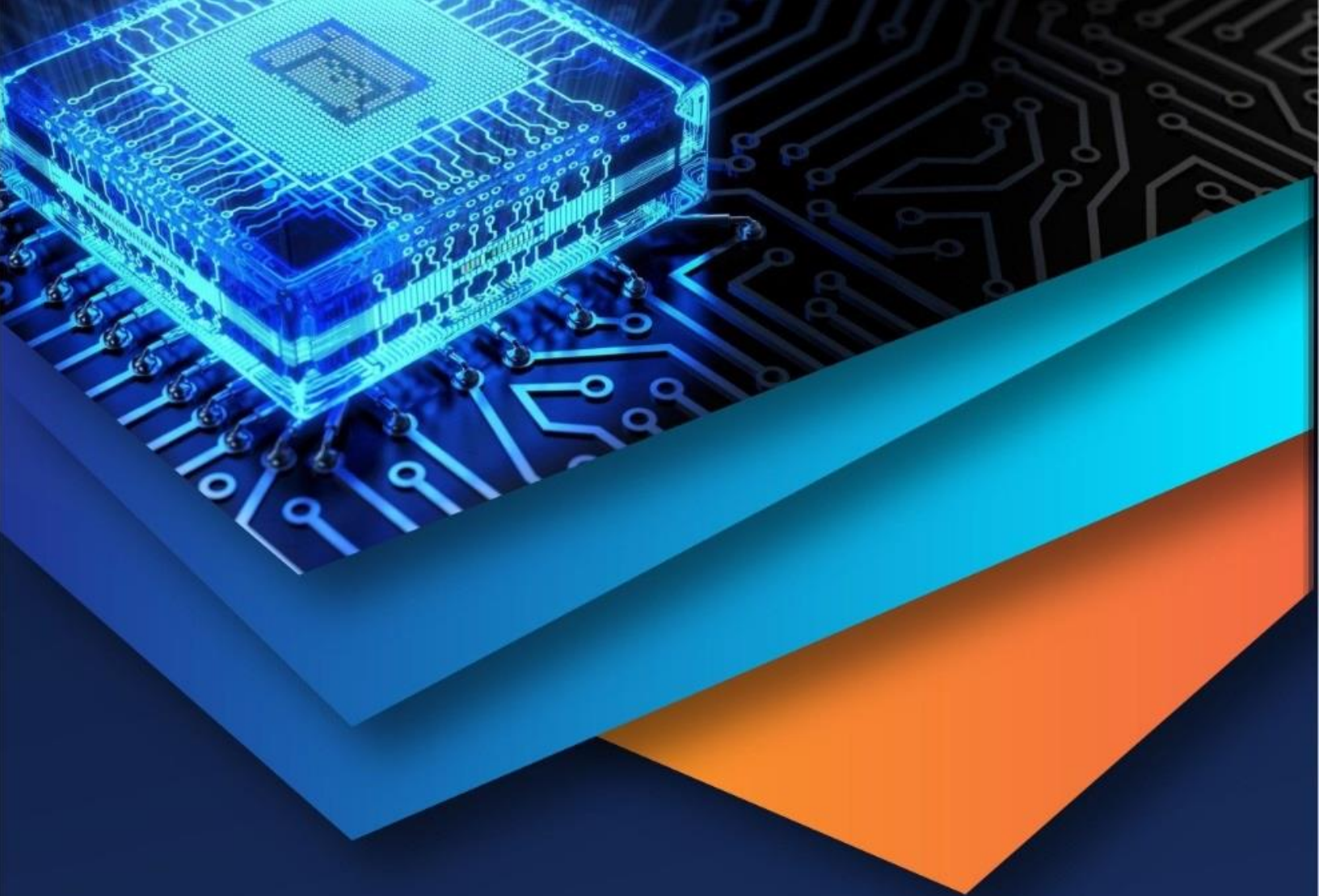
This paper has set out an Edge AI vehicle-surveillance design that ties YOLOv8 detection, DeepSORT tracking, IoU-and-optical-flow collision analysis, polygon-based parking checks, and EasyOCR plate reading into one pipeline able to catch hit-and-run incidents, wrong parking, and double parking as they happen. Because it keeps a video clip, an image, and a PDF report for every confirmed case and pushes SMS, email, and dashboard alerts out immediately, the design cuts down how much manual watching is needed and gives whoever responds a documented case to act on rather than a vague report. Running end-to-end on Python and OpenCV, on nothing more than a laptop or Raspberry Pi and without cloud servers or dedicated sensors, the system is affordable enough to scale across many sites, and later phases could extend it to red-light running, overspeeding, and finer-grained behaviour-anomaly detection.

V. ACKNOWLEDGMENT

The authors are thankful to Mrs. Sreedevi S, Assistant Professor, Department of Computer Science and Engineering, for the guidance and support extended throughout this project. Thanks are also due to Dr. Jalesh Kumar, Professor and Head, Department of Computer Science and Engineering, and to the Principal, JNN College of Engineering, Shivamogga, for their encouragement, and to the college and department more broadly for the resources made available for this work.

REFERENCES

- [1] S. Charran R and R. K. Dubey, "Edge-AI Perception Node for Cooperative Road-Safety Enforcement and Connected-Vehicle Integration," arXiv preprint arXiv:2601.07845, Jan. 2026.
- [2] B. A. Patil and S. Ghule, "Next-Gen Smart Traffic Violation Detection Using Edge AI and IoT for Safer Urban Mobility," *Int. J. Latest Technol. Eng. Manag. Appl. Sci. (IJLTEMAS)*, vol. 14, no. 13, pp. 138–142, Oct. 2025. doi: 10.51583/IJLTEMAS.2025.1413SP030.
- [3] A. Scandurra et al., "Next-gen Traffic Surveillance: AI-Assisted Mobile Traffic Violation Detection System," arXiv preprint arXiv:2311.16179, Nov. 2023.
- [4] N. K. Anisha, M. Prabu, S. P. Krishnendhu, and J. S. Anni, "DeepYOLO—An Efficient Real-Time Vehicle Detection System Using YOLOv8 and DeepSORT Algorithm," in *Lecture Notes in Networks and Systems*, vol. 1515, Springer, Singapore, Jan. 2026, pp. 1–10. doi: 10.1007/978-981-96-8799-2_12.
- [5] D. Djarah, A. Benmakhlouf, G. Zidani, and L. Khettache, "Online Multi-object Tracking with YOLOv9 and DeepSORT Optimized by Optical Flow," *Eng. Technol. Appl. Sci. Res.*, vol. 14, no. 6, pp. 17922–17930, Dec. 2024. doi: 10.48084/etasr.8770.
- [6] "Intelligent Traffic System: YOLOv8 and DeepSORT in Car Detection, Tracking and Counting," *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)*, vol. 12, Mar. 2024. doi: 10.22214/ijraset.2024.59201.
- [7] N. Sharma, S. Baral, M. P. Paing, and R. Chawuthai, "Parking Time Violation Tracking Using YOLOv8 and Tracking Algorithms," *Sensors*, vol. 23, no. 13, p. 5843, Jun. 2023. doi: 10.3390/s23135843.
- [8] S. S. Kathait, A. Kumar, R. Patidar, K. Agrawal, and S. Sawal, "Computer Vision and Deep Learning Based Approach for Violations Due to Illegal Parking Detection," *Int. J. Comput. Appl.*, 2025. doi: 10.5120/ijca2025924477.
- [9] "Advanced Deep Learning Techniques for Automated License Plate Recognition," *MDPI/PMC*, 2024.
- [10] "A Deep Learning-Based System for Automatic License Plate Recognition Using YOLOv12 and PaddleOCR," *Appl. Sci.*, vol. 15, no. 14, p. 7833, 2025. doi: 10.3390/app15147833.
- [11] V. Adewopo and N. Elsayed, "Smart City Transportation: Deep Learning Ensemble Approach for Traffic Accident Detection," arXiv preprint arXiv:2310.10038, Oct. 2023.
- [12] "Towards a Closer Fusion of Active and Passive Safety: Optical Flow-Based Detection of Vehicle Side Collisions," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2011. doi: 10.1109/IVS.2011.5940446.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)