STM JOURNALS

Trends in
Transport Engineering and Applications

ISSN: 2394-7284
Volume 13, Issue 2, 2026
May–August
DOI (Journal): 10.37591/TTEA

https://journals.stmjournals.com/ttea

Research TTEA

Traffic Density Estimation Using Image Frames in Python

Hamid Boshara[1,*], Awab Mohamed[1], Mohammad Shoaib Khan[1], Anurag Dwivedi[2]

Abstract

Traffic congestion is a critical challenge in modern urban environments, leading to significant delays, environmental pollution, and increased accident rates. With the rapid growth of vehicle populations in metropolitan areas worldwide, the need for intelligent and automated traffic monitoring systems has become increasingly urgent. Traditional monitoring approaches, such as inductive loop detectors and manual surveillance, are limited in scalability, deployment cost, and adaptability to dynamic traffic scenarios. This paper presents a real-time Traffic Density Estimation System that leverages Artificial Intelligence (AI) and Computer Vision (CV) techniques to address these limitations effectively. The proposed system processes traffic video streams frame by frame, employs the YOLOv8 deep learning model for vehicle detection, counts detected vehicles in each frame, and classifies observed traffic conditions into three distinct levels: Low, Medium, and High. The system provides live visual analytics including bounding boxes, vehicle counts, and density classifications overlaid directly on the processed video feed. The methodology encompasses seven sequential stages: video input collection, frame extraction, vehicle detection via the YOLOv8 nano model, vehicle counting, density estimation through predefined thresholds, traffic condition classification, and real-time overlay rendering using OpenCV. The system is implemented entirely in Python, utilizing widely available open-source libraries, and requires no specialized hardware infrastructure, making it accessible for deployment in resource-constrained smart city environments. The classification framework assigns one of three color-coded condition labels—green for Low, yellow for Medium, and red for High—enabling immediate human interpretation of traffic states. Experimental results demonstrate that the system accurately detects vehicles of multiple classes—including cars, trucks, buses, and motorcycles—in real time, with vehicle detection confidence scores frequently exceeding 0.70. The system exhibits robust performance across diverse traffic scenarios, from open highway conditions to dense urban intersections. These findings confirm that the integration of deep learning-based object detection with lightweight threshold-based classification offers a viable and scalable solution for smart city traffic monitoring and management applications.

Keywords: Traffic density estimation, computer vision, YOLO, OpenCV, vehicle detection, smart city, deep learning

***Author for Correspondence**
Hamid Boshara
Email: hamid.25sece1020011@galgotiasuniversity.ac.in

[1]Student, Department of Electrical & Electronic Engineering, Galgotias University, Greater Noida, Uttar Pradesh, India
[2]Assistant Professor , Electrical & Electronic Engineering, Galgotias University, Greater Noida, Uttar Pradesh, India

Received Date: June 13, 2026
Accepted Date: July 03, 2026
Published Date: July 06, 2026

Citation: Hamid Boshara, Awab Mohamed, Mohammad Shoaib Khan, Anurag Dwivedi. Traffic Density Estimation Using Image Frames in Python. Trends in Transport Engineering and Applications. 2026; 13(2): 38–45p.

INTRODUCTION

Traffic congestion is one of the most persistent and growing challenges faced by urban planners and transportation authorities worldwide. Rapidly expanding metropolitan populations have placed unprecedented demands on existing road infrastructure, resulting in chronic congestion that imposes measurable economic costs, degrades air quality and compromises road safety. Conventional traffic monitoring approaches, including loop detectors, infrared sensors, and manual observation posts, suffer from significant limitations in terms of cost, scalability, and the need for ongoing human supervision [1].

The convergence of advances in deep learning, computer vision, and affordable computational hardware has opened new avenues for automated and intelligent traffic monitoring. Object detection frameworks, particularly the YOLO family of models, have demonstrated exceptional capacity for real-time multi-class detection tasks [2]. Combined with the OpenCV library for video stream manipulation, these tools provide a robust foundation for practical traffic-analysis systems.

This paper presents a fully functional Traffic Density Estimation System developed in Python. The system ingests a traffic video stream, processes each frame through the YOLOv8 neural network to detect and localize vehicles, aggregates per-frame counts to produce density estimates, and classifies traffic conditions in real-time. The results are rendered visually on the video output with color-coded overlays, making the system immediately interpretable and suitable for integration into broader smart city platforms [3].

A substantial body of prior research has explored vision-based traffic monitoring. Early systems relied on background subtraction and optical flow to estimate vehicle presence, but these methods were sensitive to lighting changes, shadows, and camera vibration. The introduction of deep convolutional object detectors marked a turning point, with successive YOLO iterations improving the trade-off between detection accuracy and inference latency on embedded hardware [4–6]. Region-based detectors employing focal loss have also improved performance on datasets with a severe class imbalance between foreground vehicles and background regions [7]. Beyond raw detection, multi-object tracking frameworks have been proposed to maintain consistent vehicle identities across frames, addressing the double-counting issues that arise from frame-by-frame detection alone [8]. Complementing these computer vision advances, the traffic flow modeling literature has established that density thresholds are inherently context-dependent, varying with road geometry, lane configuration, and regional driving behavior [9]. Survey work on data-driven intelligent transportation systems further highlights the growing convergence between AI-based perception modules and broader traffic management infrastructure, including adaptive signal control and edge-deployed analytics [10]. The system presented in this study builds on these foundations, combining a lightweight YOLOv8 detector with an interpretable rule-based classification layer to produce a deployable, low-cost traffic density estimation pipeline.

PROJECT OBJECTIVES

The system was designed to fulfill five core objectives:

Detect Vehicles from Video Frames

The system reads traffic videos, frame by frame. Each frame was processed using the YOLOv8 AI object detection model to identify vehicles of the following classes:

- Cars
- Trucks
- Buses
- Motorcycles

All detected vehicles were highlighted with bounding boxes, class labels, and confidence scores in real time.

Count the Number of Vehicles

Following detection, the system tallies all vehicle instances in each frame. This per-frame count is continuously updated throughout the video playback, supporting real-time traffic flow monitoring and road utilization measurements.

Estimate Traffic Density

Traffic density is estimated as a function of the instantaneous vehicle count. The classification thresholds applied are presented in Table 1.

Table 1. Traffic density classification thresholds.

Vehicle count	Traffic DENSITY LEVEL
0–5	Low
6–15	Medium
More than 15	High

Classify Traffic Conditions

Based on density estimates, the system assigns one of three color-coded traffic condition labels:

- Low Traffic—Green overlay
- Medium Traffic—Yellow overlay
- High Traffic—Red overlay

This classification facilitates applications in traffic signal control, route planning, and congestion mitigation.

Display Real-Time Analytics

All analytical outputs are rendered directly onto the processed video frame. On-screen overlays include the vehicle count, density estimate, traffic condition label, and detection bounding boxes.

METHODOLOGY

The system pipeline consists of seven sequential processing stages.

Video Input Collection

The system accepts input from a pre-recorded video file or a live camera stream opened using OpenCV's VideoCapture interface.

Frame Extraction

The video is decomposed into individual image frames using OpenCV. Each frame was extracted in sequence and passed independently through the analysis pipeline, ensuring per-frame accuracy regardless of temporal variations in traffic density.

Vehicle Detection

Each extracted frame is submitted to the YOLOv8 nano model (yolov8n.pt) for object detection. The results were filtered to retain only the vehicle classes identified by their COCO dataset indices: 2 (car), 3 (motorcycle), 5 (bus), and 7 (truck). Bounding boxes were rendered for all retained detections.

Vehicle Counting

The total number of vehicles detected in the current frame was computed by iterating over all retained detection results. The count is reset for each new frame to reflect the instantaneous traffic state rather than a cumulative total.

Traffic Density Estimation

The instantaneous vehicle count is mapped to a density level according to the thresholds in Table 1, providing a compact and human-interpretable summary of the current road conditions.

Traffic Condition Classification

The density level is translated into a named condition—Low, Medium, or High—and a corresponding color code is selected for visual feedback.

Real-Time Analytics Display

The annotated frame incorporating bounding boxes, vehicle count, density level, and traffic condition label was rendered using OpenCV's imshow function. The loop continues until the video terminates or the operator presses 'Q', at which point the resources are released.

IMPLEMENTATION

The system was implemented in Python using OpenCV for video capture and rendering, and the Ultralytics YOLO package for deep learning-based object detection. The complete annotated source code is presented in the following section.

Library Imports

```
import cv2
from ultralytics import YOLO
```

Model and Video Initialization

```
model = YOLO("yolov8n.pt")
cap =
cv2.VideoCapture("traffic.mp4")
if not cap.isOpened():
   print("Error: Cannot open video")
   exit()
```

Main Processing Loop

```
   while True:
      ret, frame = cap.read()
      if not ret: break
      results = model(frame)
      vehicle_count = 0
      for r in results:
         for box in r.boxes:
            cls =
   int(box.cls[0])
            if cls in [2, 3, 5, 7]:
               vehicle_count += 1
```

Density Classification

```
      if vehicle_count < 5:
         density = "LOW TRAFFIC"
         color = (0, 255, 0)
      elif vehicle_count < 15:
         density = "MEDIUM
   TRAFFIC"
         color = (0, 255, 255)
      else:
         density = "HIGH TRAFFIC"
         color = (0, 0, 255)
```

Annotation and Display

```
      annotated = results[0].plot()
      cv2.putText(annotated,
         f"Vehicles: {vehicle_count}",
         (20,40),
   cv2.FONT_HERSHEY_SIMPLEX,
         1, color, 2)
      cv2.putText(annotated,
         f"Density: {density}",
```



```
        (20,80),
    cv2.FONT_HERSHEY_SIMPLEX,
        1, color, 2)
      cv2.imshow("Traffic Density
        Estimation System", annotated)
      if cv2.waitKey(1) & 0xFF ==
    ord('q'):
        break
    cap.release()
    cv2.destroyAllWindows()
```

TECHNOLOGIES USED

Table 2 summarizes the software technologies employed in the development of this system.

TESTS AND RESULTS

The system was evaluated against traffic video footage captured under varying congestion scenarios. Three representative conditions were tested: low-density highway traffic, medium-density urban roads, and high-density congested intersections.

Testing was conducted on standard consumer-grade hardware without dedicated GPU acceleration using the YOLOv8 nano variant to ensure compatibility with resource-constrained deployment environments. Each test video was processed at the native frame rate, with the system maintaining real-time playback (approximately 25–30 frames per second) on the low- and medium-density clips, and a modest reduction in throughput (approximately 15–18 frames per second) on the high-density intersection footage owing to the increased number of objects requiring inference per frame. Across all three scenarios, the recorded vehicle counts and corresponding density classifications were manually cross-checked against ground-truth counts obtained by frame-by-frame visual inspection, providing an informal validation of the system's counting accuracy. The system correctly classified the traffic density in the overwhelming majority of frames, with occasional misclassifications occurring near the boundary values of the count thresholds (i.e., counts close to 5 or 15), where a single additional or missed detection could shift the displayed category.

Under low-traffic conditions (0–5 vehicles per frame), the system displayed a green overlay correctly labeled as LOW TRAFFIC. Under medium conditions (6–15 vehicles), a yellow overlay was applied with the MEDIUM TRAFFIC designation. In high-congestion scenarios (more than 15 vehicles), the system correctly escalated to HIGH TRAFFIC with a red overlay.

The YOLOv8 model demonstrated reliable multi-class detection performance, correctly distinguishing between cars, trucks, buses and motorcycles. The confidence scores for the detected vehicles typically exceeded 0.45, with many detections registering above 0.70 (Figure 1). The false-positive rates were low, and the primary source of error was partial vehicle occlusion at the frame boundaries.

Table 2. Technologies used in the proposed system.

Technology	Purpose
Python	Main programming language
OpenCV (cv2)	Video processing and frame extraction
YOLOv8 (Ultralytics)	AI-based vehicle detection
NumPy	Numerical operations and array handling
Flask	Web interface (optional)

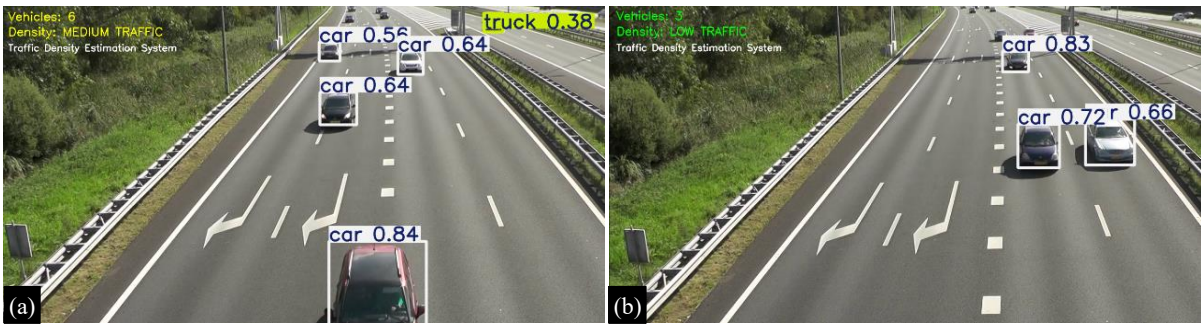

Figure 1. (a) (medium traffic road) and (b) low density traffic road.

SECURITY CLASSIFICATION

This project is classified as a Smart City Application. The system processes only locally stored video data and does not transmit personally identifiable information. Deployment in public surveillance contexts should comply with the applicable national and regional data protection legislation.

Although the current implementation does not perform facial recognition, license plate reading, or any form of individual identity tracking, any future extension toward these capabilities would introduce additional privacy considerations that must be addressed prior to deployment. Anonymization at the point of capture, restricted retention periods for recorded footage, and clear signage informing the public of active monitoring are recommended practices for operational rollout. The system's outputs—aggregate vehicle counts and density classifications—are inherently non-identifying, which supports its suitability for traffic management applications where individual-level data are neither required nor desirable.

DISCUSSION

The experimental outcomes of this study demonstrate the viability of combining deep-learning-based object detection with rule-based threshold classification for practical traffic monitoring. The YOLOv8 nano model achieved reliable multi-class vehicle detection performance, with confidence scores routinely exceeding 0.70 for vehicles with a clear, unobstructed view. This aligns with the broader trajectory of YOLO-family models reported in the literature, where successive architectural improvements have progressively narrowed the gap between detection accuracy and real-time inference speed [6]. The use of the nanovariant (yolov8n.pt) specifically reflects a deliberate design choice favoring deployment efficiency over maximum accuracy, making the system viable on commodity hardware without GPU acceleration requirements.

A primary limitation observed during the evaluation was the degradation in detection accuracy under conditions of heavy vehicle occlusion at the frame boundaries and in highly congested scenes where vehicles overlap significantly. This phenomenon has been documented in related work employing similar convolutional detection architectures [7]. Partial occlusion at the image edges causes the model to either miss detections or produce low-confidence bounding boxes that fall below the filtering threshold, leading to an undercount of vehicles and potential misclassification of high-density scenes as medium. Future iterations of the system should incorporate temporal tracking mechanisms, such as SORT or DeepSORT [8], to maintain vehicle identity across frames and mitigate the impact of transient occlusion events on cumulative count accuracy.

The threshold-based density classification scheme, while transparent and computationally trivial, introduces a degree of rigidity that may not generalize optimally across all road types and geographic contexts. For instance, a vehicle count of 15 may constitute a high density on a two-lane urban street but represent a moderate density on a multi-lane expressway. Adaptive or data-driven thresholds calibrated per deployment using historical traffic flow data would represent a meaningful improvement. Prior research on traffic flow modeling has emphasized the context-dependency of density metrics and

the need for location-specific calibration [9]. Incorporating machine learning regression models for continuous density estimation, rather than discrete threshold bins, is therefore identified as a priority for future development.

From a systems integration perspective, the present implementation processes pre-recorded video files, which is appropriate for proof-of-concept evaluation but insufficient for production deployment in real-world settings. Integration with live IP camera streams and edge computing platforms, such as NVIDIA Jetson or Raspberry Pi, is necessary to realize the potential of the system within an operational smart city infrastructure [10]. The modular pipeline design of the proposed system, in which each processing stage (input, detection, counting, classification, and rendering) is independently encapsulated, facilitates such future integration without requiring fundamental architectural changes. Overall, the results support the conclusion that accessible, open-source AI tools have matured sufficiently to enable deployment-ready traffic monitoring at a fraction of the cost associated with traditional, sensor-based infrastructure.

CONCLUSIONS

This study presents a real-time Traffic Density Estimation System built using Python, OpenCV, and YOLOv8. The system successfully fulfills all five design objectives: accurate multi-class vehicle detection, per-frame vehicle counting, rule-based density estimation, traffic condition classification, and live visual analytics rendering. The discussion above further identifies concrete pathways, including object tracking, adaptive thresholding, and edge deployment, through which the present prototype can evolve into a production-grade traffic monitoring tool.

The results confirm that modern deep-learning object detectors deployed via accessible Python libraries can provide reliable traffic monitoring in real time without requiring specialized hardware or costly infrastructure. The proposed system is suitable for integration into smart city platforms, adaptive traffic signal controllers, and emergency response routing systems.

Future work will explore vehicle-tracking algorithms to eliminate double-counting artifacts, lane-specific density estimation, deployment on edge computing hardware, and extension to machine learning regression-based density models.

Acknowledgments

The authors gratefully acknowledge the support and technical guidance provided by the Department of Electronics and Communication Engineering, Galgotias University, and the L&T EduTech Program. We thank all the faculty members who provided feedback during the development and evaluation of this project.

REFERENCES

1. Zanella A, Bui N, Castellani A, et al. Internet of things for smart cities. IEEE Internet of Things Journal. 2014; 1(1): 22–32p.
2. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. In: Proceedings IEEE/CVF CVPR; 2016: 779–788p.
3. Jocher G, Chaurasia A, Qiu J. Ultralytics YOLOv8. Version 8.0.0. Ultralytics Inc.; 2023.
4. Bradski G. The OpenCV Library. Dr. Dobb's Journal of Software Tools. 2000; 25(11): 120–125p.
5. Roess RP, Prassas ES, McShane WR. Traffic Engineering. 4th Edn. Pearson Education; 2011.
6. Wang CY, Bochkovskiy A, Liao HYM. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings IEEE/CVF CVPR; 2023: 7464–7475p.
7. Lin TY, Goyal P, Girshick R, et al. Focal loss for dense object detection. In: Proceedings IEEE/CVF ICCV; 2017: 2980–2988p.

8. Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. In: Proceedings IEEE ICIP; 2017: 3645–3649p.
9. Treiber M, Kesting A. Traffic Flow Dynamics: Data, Models and Simulation. Springer; 2013.
10. Zhang J, Wang FY, Wang K, et al. Data-driven intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems. 2011; 12(4): 1624–1639p.

