WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

AI-BASED BUS ENTRY AND EXIT LOGGING SYSTEM: REAL-TIME VEHICLE DETECTION, VIRTUAL TRIPWIRE COUNTING AND WEB DASHBOARD FOR CAMPUS GATE MONITORING

Ms. Swetha Krishna U[1], Arshad S[2], Bharanitharan M[3], Dhanarishvanth T[4], Aravindhan D[5]

Department of Artificial Intelligence and Data Science, Bachelor of Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, India.

ABSTRACT :

Managing institutional gate traffic manually leads to long delays, human recording errors, traffic congestion during peak hours and unverified vehicular entries. The AI-Powered Automated Campus Bus Entry & Exit Monitoring System is proposed as an automated solution that combines real-time vehicle detection, multi-object tracking, proximity-based virtual tripwire line- crossing logic and a persistent SQLite database with an interactive web dashboard.

The system captures video frames from a webcam feed, detects target vehicles using YOLOv8, tracks them across consecutive frames, calculates bounding box centre coordinates relative to a virtual horizontal tripwire boundary (LINE_Y = 240) and logs verified ENTRY or EXIT events. Event logs, containing vehicle track labels, movement direction, timestamps and confidence scores, are saved directly into SQLite (campus_gate_logs.db). A responsive web dashboard, served through a Python HTTP server on port 8000, polls Flask backend API endpoints on port 5000 to display real-time counters (Total Entries, Total Exits, Buses Inside Campus) and detailed log inspection tables.

Keywords: Computer Vision, Object Detection, YOLOv8, Object Tracking, Virtual Tripwire, Flask REST API, SQLite Database, Real-Time Dashboard.

INTRODUCTION:

Educational institutions run a large fleet of buses that enter and leave the campus several times a day. At most gates, the arrival and departure of each bus is still written in a register by a security guard. During peak hours, when many buses arrive together, entries are missed, written late or recorded wrongly, and managers cannot say with confidence how many buses are inside the campus at a given moment.

Computer vision offers a practical alternative. A single camera pointed at the gate can be used to detect each bus, follow it from frame to frame and decide, from the direction of its movement, whether it is entering or leaving. Because each decision is stored with a time and a confidence score, the result is an auditable record rather than a handwritten note.

The proposed AI-Based Bus Entry and Exit Logging System combines four parts in one pipeline: YOLOv8 detection, multi-object tracking, a virtual tripwire for direction-aware counting, and a Flask and SQLite backend with a live web dashboard. The dashboard shows Total Entries, Total Exits and Buses Inside Campus, together with a searchable table of every logged event.

OBJECTIVE:

- To detect buses and other target vehicles at the campus gate in real time using YOLOv8.
- To track each vehicle across consecutive frames so that it keeps a stable track label.
- To define a virtual horizontal tripwire (LINE_Y = 240) and detect when a vehicle crosses it.
- To classify each verified crossing as an ENTRY or an EXIT event from the direction of movement.
- To store every event with its track label, direction, timestamp and confidence score in a persistent SQLite database.
- To expose the stored data through Flask REST API endpoints.

1. Redmon et al. (2016) introduced YOLO, which treats object detection as a single regression problem and processes the whole image in one pass. This makes real-time detection practical on ordinary hardware and is the basis of the YOLOv8 detector used in this work.

• To display real-time counters and log tables on a responsive web dashboard.
• To avoid counting the same vehicle more than once for a single crossing.

LITERATURE SURVEY:

1. **Jocher et al. (2023)** released Ultralytics YOLOv8, which provides pretrained detection models and a simple Python interface. It allows vehicle classes such as bus, car and truck to be detected without training a model from scratch.
2. **Bewley et al. (2016)** proposed SORT, a simple online tracker that associates detections across frames using motion prediction. Wojke et al. (2017) extended it as DeepSORT by adding appearance features, which helps keep the same identity after short occlusions.
3. **Zhang et al. (2022)** presented ByteTrack, which associates both high- and low-confidence detection boxes instead of discarding the low-confidence ones. This improves tracking when a vehicle is partly hidden by another.
4. **Bradski (2000)** described OpenCV, the library used for capturing webcam frames and drawing overlays such as the virtual tripwire line and bounding boxes on the video feed.

METHODOLOGY:

1.Frame Acquisition: Video frames are read continuously from a webcam feed placed to view the gate. Frames are passed to the detector in sequence.
2.Vehicle Detection: YOLOv8 returns a class label, a confidence score and a bounding box for each object. A confidence threshold removes weak detections, and only the target vehicle classes are kept.
3.Multi-Object Tracking: Detections of the same vehicle in consecutive frames are linked and given a persistent track label, so that a bus that stays in view is not treated as a new bus in every frame.
4.Virtual Tripwire Logic: For each tracked vehicle the centre of its bounding box is calculated. A virtual horizontal line is fixed at LINE_Y = 240. The system compares the centre position of the vehicle in the previous frame with its position in the current frame. When the centre moves from one side of the line to the other, a crossing is detected, and the direction of the movement decides whether the event is an ENTRY or an EXIT. [Confirm which direction is ENTRY for your camera angle.]
5.Duplicate Prevention: The crossing state of each track label is remembered, so that a vehicle waiting near the line, or a bounding box that jitters across it, is logged once and not repeatedly.
6.Event Logging: Each verified event is written to the SQLite database campus_gate_logs.db with the vehicle track label, movement direction, timestamp and confidence score.
7.Backend API: A Flask application on port 5000 reads the database and returns the counters and log records as JSON.
8.Web Dashboard: A web page served by a Python HTTP server on port 8000 polls the Flask endpoints at regular intervals and updates Total Entries, Total Exits, Buses Inside Campus and the log table without a manual refresh. Buses Inside Campus is calculated as Total Entries minus Total Exits.

EXISTING SYSTEM:

1.Manual Registers: A security guard writes the vehicle and time in a register, which is slow and depends on the guard's attention.
2.Human Recording Errors: Entries are missed, written late or recorded wrongly, especially during peak hours.
3.No Live Count: There is no way to know how many buses are inside the campus without counting them by hand.
4.Congestion at the Gate: Stopping each vehicle to record details causes queues and traffic congestion.
5.Weak Auditing: Paper records are hard to search, easy to alter and cannot be checked against any video evidence.

PROPOSED SYSTEM:

1.Automatic Detection: Buses are detected from the camera feed in real time without any action by the guard.
2.Direction-Aware Counting: The virtual tripwire separates ENTRY from EXIT events using the direction of movement.
3.Persistent Records: Every event is stored in SQLite with a timestamp and a confidence score, so the history can be audited later.
4.Live Dashboard: Gate managers see the counters and the latest events in a web browser as they happen.
5.Low-Cost Setup: The system needs only a webcam and a standard computer, and uses open-source software.

SYSTEM REQUIREMENTS:

Hardware Requirements:

• Processor: Intel Core i5 or higher, or equivalent. [Update to match your test device]
RAM: Minimum 8 GB recommended. [Update]
• Camera: Built-in or external USB webcam.

• Display: Any device with a web browser to view the dashboard.

Software Requirements:

• Operating System: Windows 10/11, Linux or macOS.

• Language: Python 3. Computer Vision: YOLOv8 (Ultralytics), OpenCV.

• Backend: Flask REST API (port 5000).

• Database: SQLite (campus_gate_logs.db).

• Frontend: HTML, CSS and JavaScript dashboard served by the Python HTTP server (port 8000).

• Browser: Google Chrome, Mozilla Firefox or Microsoft Edge.

• [Edit this list so it names only tools actually used in the final version.]

Processing

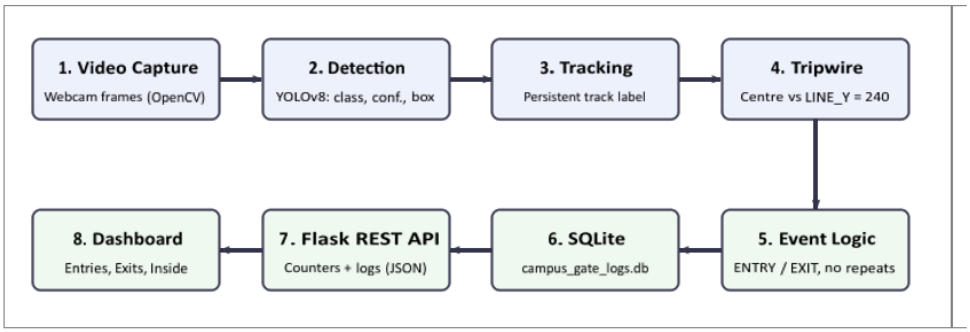


Fig. 1: Processing flow of the AI-Based Bus Entry and Exit Logging System

The webcam frame is passed to YOLOv8 for detection and then to the tracker. The tripwire module compares each tracked centre with LINE_Y and, on a verified crossing, writes an ENTRY or EXIT record to SQLite. The Flask API reads the database, and the dashboard polls the API to refresh the counters and tables.

MODULES:

1.Video Capture Module: Reads frames continuously from the webcam.

2.Detection Module: Uses YOLOv8 to return labels, confidence scores and bounding boxes for target vehicles.

3.Tracking Module: Maintains a persistent track label for each vehicle across frames.

4.Tripwire Module: Calculates box centres, compares them with LINE_Y = 240 and decides ENTRY or EXIT.

5.Database Module: Creates and writes to the SQLite database campus_gate_logs.db.

6.Flask API Module: Serves counters and log records as JSON on port 5000.

7.Dashboard Module: Displays real-time counters and log inspection tables on port 8000.

Event classification used by the system:

Outcome	Rule
ENTRY	Track centre crosses LINE_Y in the entry direction and the crossing is verified.
EXIT	Track centre crosses LINE_Y in the exit direction and the crossing is verified.
No event	Track centre stays on one side of LINE_Y, or the track label has already been logged for this crossing.

Example event log record:


```
{ "id": 1,
  "track_label":
"bus_3",
  "direction": "ENTRY",
  "timestamp": "2026-11-
02 08:15:42",
  "confidence": 0.91 }
```

Dashboard counters:

Counter	Meaning
Total Entries	Number of verified ENTRY events in the database.
Total Exits	Number of verified EXIT events in the database.
Buses Inside Campus	Total Entries minus Total Exits.

APPLICATION SCREENSHOTS:

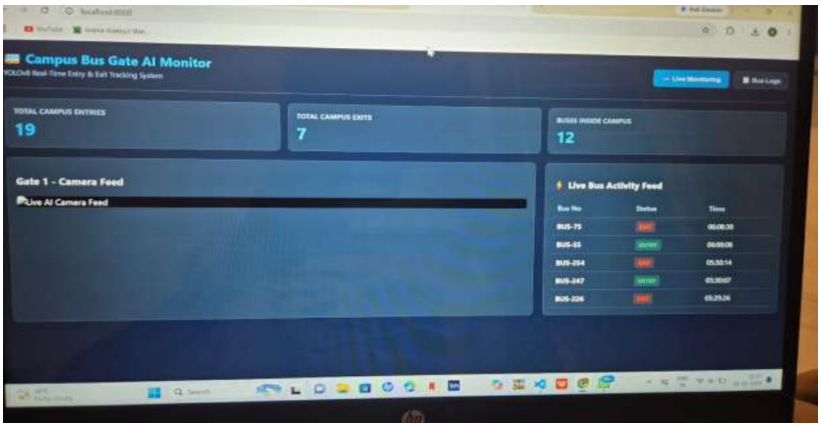


Fig. 2: Live camera feed with virtual tripwire line (LINE_Y = 240)

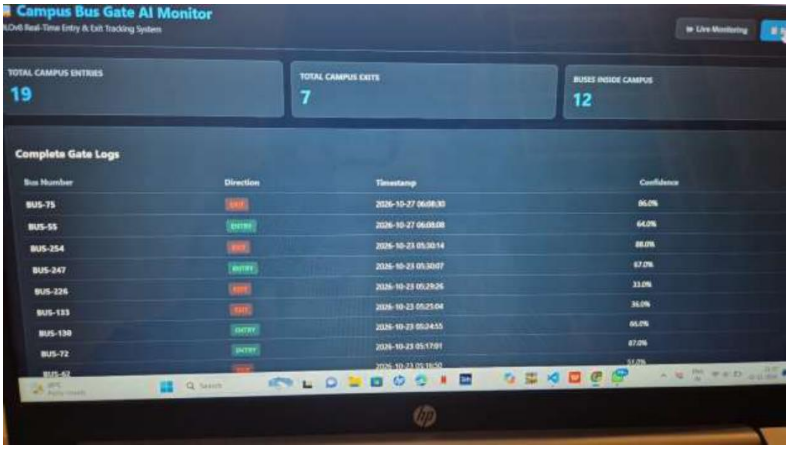


Fig. 3: Vehicle detection with bounding boxes and track labels

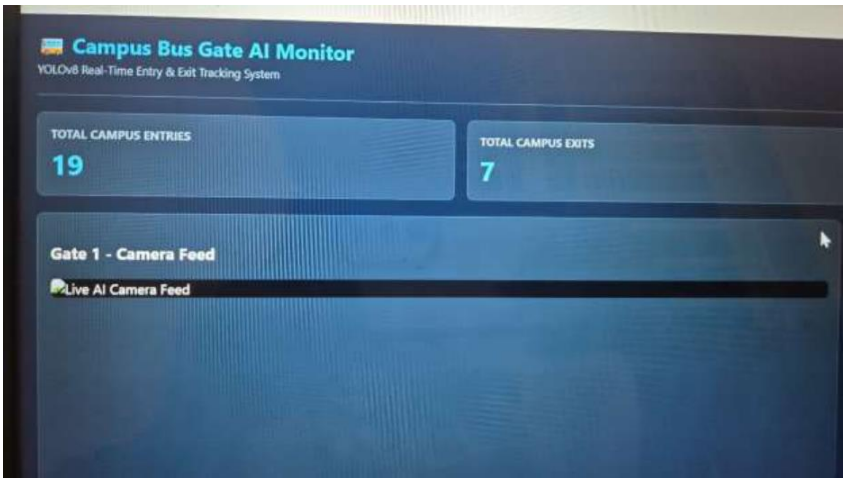


Fig. 4: ENTRY and EXIT event detection



Fig. 5: Web dashboard with real-time counter

TESTING PLAN:

The system is to be tested in the workspace environment with a webcam and recorded or live vehicle movement. The table below lists the planned test cases; results are to be filled in after testing.

No.	Test case	Expected result	Observed result
1	Bus moves across LINE_Y in the entry direction	One ENTRY event logged	[Fill after testing]
2	Bus moves across LINE_Y in the exit direction	One EXIT event logged	[Fill after testing]
3	Bus stops near LINE_Y without crossing	No event logged	[Fill after testing]
4	Two buses cross close together	Two separate events with different track labels	[Fill after testing]
5	Dashboard polling	Counters update without manual refresh	[Fill after testing]
6	Restart of the application	Previous logs remain in SQLite	[Fill after testing]

LIMITATIONS:

•Detection accuracy drops in low light, heavy rain, motion blur and when buses overlap.
•A single horizontal tripwire assumes that vehicles cross the line from top to bottom or bottom to top in the camera view.
•Track labels are assigned by the tracker and do not identify a bus by its registration number.
•A tracking error, such as a lost or swapped label, can produce a missed or duplicate event.
•The system depends on the camera position and angle, and LINE_Y must be set again if the camera is moved.
•The dashboard uses polling, so there is a short delay before a new event appears.

CONCLUSION:

The AI-Based Bus Entry and Exit Logging System shows that a low-cost webcam, YOLOv8 detection, object tracking and a virtual tripwire can replace manual registers at a campus gate. Each ENTRY or EXIT event is stored with a timestamp and a confidence score, and the live dashboard gives gate managers an up-to-date count of the buses inside the campus together with an auditable history.

Future work includes reading bus registration numbers with licence-plate recognition, using multiple tripwires or polygon zones, supporting several gates and cameras, adding alerts for unauthorised vehicles, using WebSockets instead of polling, and moving the database and dashboard to a cloud-hosted service.

REFERENCES:

[1]Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2]Jocher, G., Chaurasia, A., & Qiu, J. Ultralytics YOLOv8. Software, 2023. https://github.com/ultralytics/ultralytics

[3]Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. Simple Online and Realtime Tracking. Proc. IEEE International Conference on Image Processing (ICIWojke, N., Bewley, A., & Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. Proc. IEEE ICIP, 2017.

[4]Zhang, Y., Sun, P., Jiang, Y., et al. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. Proc. European Conference on Computer Vision (ECCV), 2022.

[5]Bradski, G. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

[6]Pallets Projects. Flask Documentation. https://flask.palletsprojects.com

[7]SQLite Consortium. SQLite Documentation. https://www.sqlite.org/docs.html