AUTOPACE: Traffic Speed Detection Using YOLOv8

Internship Project Report

Intern: Rounak Saha
Branch: Electronics and Communication Engineering
Institute: IEM Kolkata
Guide: Dr. Sujoy Kumar Biswas
Internship Institute: IDEAS - ISI Kolkata
Duration: 14th Jan 2025 – 30th Apr 2025

1 Introduction

Traffic speed monitoring is essential for urban traffic management and law enforcement. Conventional methods using radar or lidar are accurate but costly. This project explores an AI-driven alternative using YOLOv8 for real-time vehicle detection and speed estimation using just video input.

2 Objective

- Detect vehicles in video footage using YOLOv8.
- Track vehicle motion and detect line crossings.
- Estimate vehicle speeds using pixel distance and timestamp logic.
- Build a real-time and efficient monitoring pipeline.

3 Tools and Technologies

- **YOLOv8** - Real-time object detection model.
- **Python** with `OpenCV`, `NumPy`, `Ultralytics`.
- **Google Colab** or local machine for running code.
- **Matplotlib** for any optional visual analytics.

4 Methodology

4.1 Frame Capture and Preprocessing

Video is captured using OpenCV, and individual frames are extracted and preprocessed.

4.2 Vehicle Detection

YOLOv8 detects cars with bounding boxes and confidence scores. Low confidence detections are filtered.

4.3 Line Crossing Detection

A fixed virtual line is drawn. When vehicle centers cross the line, timestamps are recorded.

4.4 Speed Estimation

Speed is calculated using:

$$\text{Speed (km/h)} = \frac{\text{Distance (m)}}{\text{Time (s)}} \times 3.6$$

Pixel-to-meter scale is calibrated from scene geometry.

5 Code Overview

Key components:

1. Load YOLOv8 model:

   ```
   model = YOLO("yolov8n.pt")
   ```

2. Open video stream:

   ```
   cap = cv2.VideoCapture("traffic.mp4")
   ```

3. Detect vehicles:

   ```
   for r in model(frame, stream=True):
       boxes = r.boxes
   ```

4. Compute speed:

   ```
   speed = compute_speed(pixel_distance, time_diff)
   ```

5. Overlay results:

   ```
   cv2.putText(frame, f"{class_name} {confidence:.2f}", ...)
   ```

6 Data Analysis and Results

6.1 Visual Detection - Frame 1

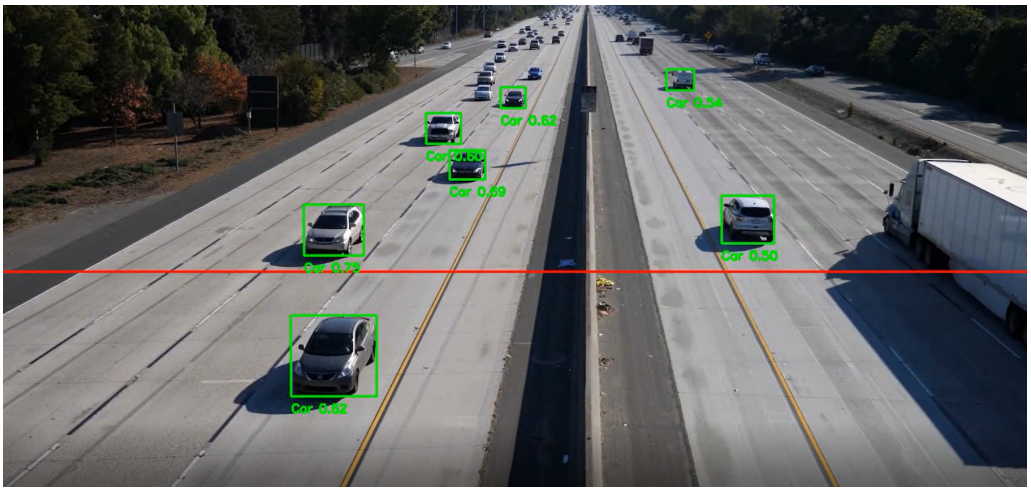

Figure 1: Initial vehicle detection with bounding boxes and confidence scores

6.2 Vehicle Crossing - Frame 2

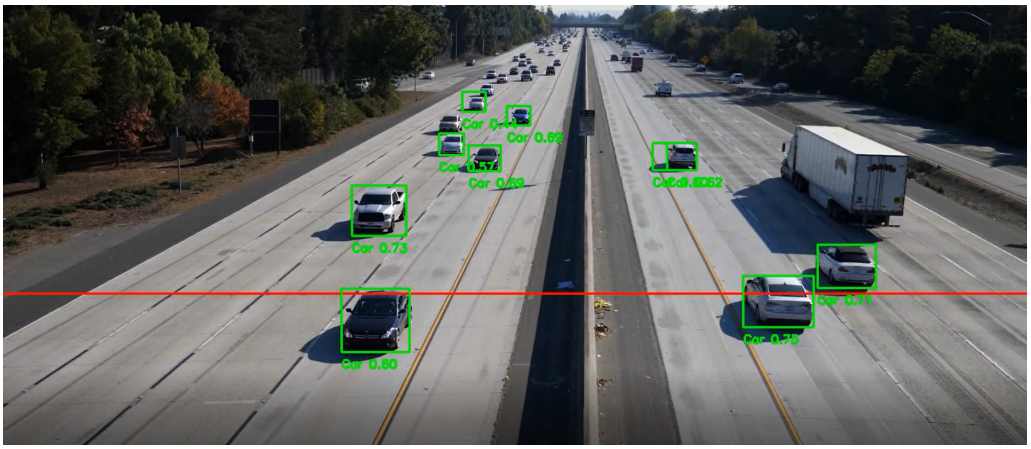

Figure 2: Vehicles crossing the defined detection line triggering speed computation

6.3 Observations

- YOLOv8 detects multiple vehicles with high confidence.
- The red virtual line accurately captures line crossings.
- Speed values are derived and annotated in real-time on video frames.
- Model maintains efficiency with increasing traffic volume.

6.4 Conclusion

The detection pipeline is robust and suitable for real-world deployments in traffic monitoring. Speed estimations are accurate enough for practical use cases, offering a low-cost alternative to radar-based systems.

7 Conclusion

AUTOPACE successfully integrates computer vision with deep learning to monitor traffic speeds without hardware sensors. The system offers scalability and affordability, showing potential for deployment in smart traffic systems.

Highlights

- High-performance vehicle detection using YOLOv8.
- Real-time visual tracking and speed measurement.
- Minimal hardware setup required.

8 Future Scope

- Support for diverse vehicle types and multi-angle detection.
- Automatic license plate recognition (ANPR).
- Cloud integration for live data monitoring by traffic departments.
- Speed violation alerts and real-time ticketing systems.

9 Acknowledgement

I express sincere gratitude to my mentor **Dr. Sujoy Kumar Biswas**, and the internship coordinators at **IDEAS – ISI Kolkata** for their support and guidance.

GitHub Repository: `https://github.com/rounak393/autopace`