Problem Statement

- Current State-Of-The-Art (SOTA) solutions for automotive collision avoidance work by:
- Vehicle detection
- Distance estimation
- Time-to-collision(TTC) calculation (using vehicle speed data from GPS or Engine)
- They then warn the driver or apply the brakes (if supported) when the TTC drops below a predefined threshold
- This method has one limitation:
- The vehicle needs to be in front of the driver to be detected
- Vehicles that abruptly cut into the driver's path typically are not considered for collision avoidance

Unique Idea Brief (Solution)

This system enhances vehicular safety by using advanced computer vision techniques. A forward-facing camera continuously captures frames, which are then resized, normalized, and converted to a format suitable for deep learning. The YOLO model detects various vehicle types in real-time, extracting bounding boxes, class IDs, and confidence scores while eliminating redundant boxes through non-max suppression. It calculates the distance to each vehicle using bounding box dimensions and camera parameters, tracking vehicles across frames and detecting potential cut-ins via lateral movements. The system computes the Time-to-Collision (TTC) for each vehicle, focusing on those within a critical distance, and visualizes threats by drawing color-coded bounding boxes with distance and TTC information. Integrated into autonomous or driver-assistance systems, this solution continuously updates its algorithms based on real-world performance data, ensuring accurate detection, distance estimation, and cut-in prediction for improved road safety.

Process flow

1. Frame Capture
 - Continuously capture frames from a forward-facing camera installed on the vehicle.

2. Image Preparation:
 - Resize and normalize the images to prepare them for YOLO model input.
 - Convert images into blob format for deep learning inference.

3. Detection Process:
 - Input the preprocessed images into the YOLO model.
 - Extract bounding boxes, class IDs, and confidence scores for identified vehicles.
 - Use non-max suppression to remove redundant bounding boxes.

4. Distance Calculation:
 - Determine the distance to each detected vehicle using the bounding box dimensions and known camera parameters (focal length and the real-world width of a vehicle).

5. Vehicle Tracking:
 - Track vehicles across frames by matching bounding boxes based on Intersection over Union (IoU).
 - Detect potential cut-ins by observing significant lateral movements of vehicles relative to the host vehicle's path.

6. TTC Computation:
 - Calculate the TTC for each vehicle using changes in their positions across frames and the time intervals between frames.
 - Pay special attention to vehicles within a critical distance (e.g., 3.5 meters) to identify potential hazards.

7. Threat Visualization:
 - Draw bounding boxes around detected vehicles, using color codes to indicate threat levels (e.g., red for high-risk cut-ins).
 - Display distance and TTC information around the bounding boxes.
 - Issue visual or auditory alerts for imminent collision threats based on TTC and distance.

8. System Integration:
 - Incorporate the processed data into the vehicle's autonomous driving or driver-assistance systems.
 - Continuously update and improve detection, tracking, and prediction algorithms using real-world performance data and feedback.

This approach ensures real-time operation, providing accurate detection, distance estimation, and cut-in prediction to enhance the safety and effectiveness of autonomous driving and driver-assistance systems.

Features Offered

1. Real-Time Vehicle Detection
Uses a pre-trained YOLO (You Only Look Once) model to detect various types of vehicles (cars, trucks, buses, motorbikes, bicycles) in real-time from input images.

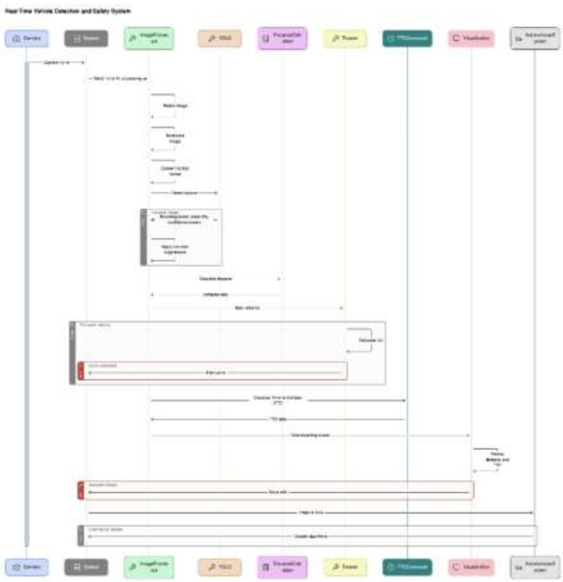
2. Distance Estimation
Estimates the distance to detected vehicles using bounding box dimensions and known camera parameters, improving situational awareness.

3. Cut-In Prediction
Tracks vehicle movements across consecutive frames and identifies potential cut-ins by calculating the Time-to-Collision (TTC) for vehicles within a critical distance threshold.

4. Time-to-Collision (TTC) Calculation
Provides accurate TTC calculations for vehicles posing an imminent threat, enabling timely alerts and interventions.

5. Bounding Box Visualization
Displays bounding boxes around detected vehicles with color coding to indicate potential hazards (e.g., red for critical threats).

Architecture Diagram



Technologies used

1. **Python:** The primary programming language used to develop the vehicle detection and tracking system.
2. **YOLO (You Only Look Once):** A pre-trained model utilized for real-time detection of various vehicle types.
3. **Convolutional Neural Network (CNN):** The deep learning architecture underpinning the YOLO model for accurate object detection.
4. **OpenCV:** A computer vision library used for image processing, including frame capture and preprocessing tasks.
5. **Numpy:** A library for numerical computations, essential for handling image data and camera parameters.
6. **Pandas:** A data manipulation library used for managing and analyzing data collected during the detection and tracking process.
7. **Matplotlib:** A plotting library used to visualize bounding boxes and other relevant information on detected vehicles.

Team members and contribution:

1. **Achal Bajpai -** Worked on the complete project of Vehicle Cut in Detection using Yolov8

T

Conclusion

This project integrates computer vision techniques for real-time vehicle detection and safety prediction to enhance road safety. Using the YOLO model and algorithms for cut-in detection and **time-to-collision (TTC) calculations**, it effectively identifies and predicts hazardous scenarios. This AI-driven solution demonstrates the potential for improving vehicular safety and provides a solid foundation for further development and real-world deployment.

T