Real-time Vision-based Autonomous Driving Assistance System

Tanvir Ahmad (ID: 111074403) *Anh-Huy Dinh (ID: 111434103)*

Code is available at https://github.com/anhhuy-dinh/AVCAS

1. Introduction

Autonomous vehicles and advanced driver assistance systems (ADAS) require real-time lane detection and object recognition and tracking to ensure safe navigation. Lane departure and object collision are common causes of accidents, emphasizing the need for an efficient vision-based system. Traditional lane detection methods struggle in challenging conditions such as low visibility, varying road textures, and occlusions. Additionally, real-time object detection is essential for identifying obstacles, pedestrians, and vehicles to make informed driving decisions. Real-time multiple objects tracking is also important because it allows determining the direction of vehicle movement, which helps predict unexpected lane changes or potential collisions.

The primary goal of this project is to design and implement an Advanced Vehicle Collision Avoidance System (AVCAS) capable of processing video input, such as dashcam footage, in real-time. The system is engineered to produce an annotated output stream, featuring bounding boxes around detected road users, estimated distances to these objects, clearly marked lane boundaries, and timely collision warnings.

The development of this system is unique and crucial in intelligient vehicle technology. A search of recent articles on Google Scholar reveals that while there is considerable research in each of these individual components (e.g., lane detection, object detection, and tracking), few works integrate all three approaches into a unified, real-time solution for autonomous vehicles. Most studies either rely entirely on deep learning models (such as YOLO or LaneNet) or classical vision-based techniques like the Hough Transform for lane detection. However, these methods often struggle with real-time performance, handling occlusions, or robustness in varying environmental conditions.

This project is novel because it uniquely integrates multiple state-of-the-art approaches into a single pipeline that balances the strengths of classical and deep learning methods. The Ultra-Fast-Lane-Detection ensures reliable performance in well-defined conditions, while the YOLO model offers powerful real-time object detection, and ByteTrack adds an efficient multi-object tracking system to handle occlusions and re-entries. This combined system offers a more comprehensive, efficient, and adaptable solution for autonomous driving assistance than any single approach on its own.

From the perspective of innovation, the proposed approach addresses key limitations identified in recent literature, such as handling partial occlusions in object tracking and ensuring robust lane detection in complex road scenarios. By integrating these technologies into a single system optimized for real-time performance, the proposed project fills a gap in current research. Most existing solutions either focus on improving one component (e.g., lane detection accuracy or object tracking precision) or lack real-time deployment readiness.

This project stands out because it can be extended to various applications in autonomous driving, smart city systems, and other vehicle-related technologies. Given the rise in autonomous vehicles and the growing interest in intelligent transportation systems, the proposed project has significant real-world implications, especially in improving safety, efficiency, and reliability.

2. Initial Approach

2.1. Overview

The project's initial phase focused on establishing a foundational ADAS pipeline using a combination of computer vision techniques. As outlined in the project proposal , the initial system architecture comprised:

- **Lane Detection:** The Hough Transform algorithm was employed for detecting lane markings. This classical computer vision technique is known for its ability to identify lines in images based on edge information.
- **Object Detection:** YOLOv4 was selected for detecting traffic objects such as vehicles and pedestrians.
- **Object Tracking:** ByteTrack was integrated to maintain the identity of detected objects across consecutive frames, enabling persistent tracking.

2.2. Implementation Details

For object detection, pre-trained weights for the YOLOv4 model, originally trained on the COCO dataset, were utilized. These weights were provided by the authors of the Darknet framework (refer to this GitHub repository for downloading: https://github.com/AlexeyAB/darknet) . The lane detection component using the Hough Transform and the object tracking component using ByteTrack did not necessitate separate model training in this initial stage, as they are algorithmic approaches that operate on the output of the object detector or image features directly . The complete initial system was implemented and tested on a sample road traffic video sourced from the internet to evaluate its baseline performance.

2.3. Preliminary Results and Limitations

Figure 1. Frame 0

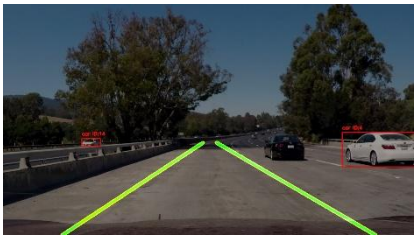

Figure 2. Frame 5

Figure 3. Frame 10

Video Information		Timing Statistics		Method Breakdown	
Length	10.04s	Total processing time	242.24s	Land detection	9.11ms/frame
FPS	25	Average time/frame	958.78ms	Object detection	948.92ms/frame

Total frames	251	Processing speed	1.04 FPS	Object tracking	0.73ms/frame
				Other operations	0.01ms/frame

Preliminary results indicate:

- Figure 1-3 shows that the Hough transform for lane detection and ByteTrack perform quite well, as the two lane lines are detected accurately, and the two vehicles on the right side maintain their fixed IDs after 10 frames, even though they are not detected continuously.
- The processing speed of the end-to-end model across all frames of a 10.04-second video is quite slow (242.24s). This will impact real-time performance when applied to data collected via a webcam or integrated camera in real-time.
- The Hough Transform fails to detect curved lane patterns, and it performs lane detection based on manually set colors (specifically white and yellow). This is ineffective in cases like mountain pass roads with fully curved lanes and varying lane colors.
- Although the YOLOv4 model detects objects fairly well, it does not maintain consistent detection states. In some frames, cars are not identified despite occupying a large area and being clearly visible to the naked eye (e.g., frame 5).

These limitations underscored the need for a more advanced, deep learning-centric approach to achieve the project's goals of real-time performance and robust perception.

3. Proposed Approach: AVCAS (Advanced Vehicle Collision Avoidance System)

3.1. Rationale for Change

The initial approach, while demonstrating basic functionality, suffered from high latency and limitations in both lane and object detection accuracy, making it unsuitable for a real-time ADAS . To address these shortcomings, particularly the poor real-time performance, a shift towards state-of-the-art (SOTA) deep learning-based models was adopted . These models offer superior accuracy and can leverage GPU acceleration for significant speed improvements, which is crucial for reducing latency .

3.2. System Overview

The revised system, named Advanced Vehicle Collision Avoidance System (AVCAS), is designed as a comprehensive driver assistance tool. AVCAS integrates multiple computer vision components:

- **Object Detection and Tracking:** Employs YOLOv11 for detecting various road users (cars, trucks, buses, pedestrians, motorcycles, bicycles) and ByteTrack for robust tracking.
- **Lane Detection:** Utilizes a modified UltrafastLaneDetector for accurate identification of lane markings.
- **Distance Estimation:** Implements monocular camera-based techniques to estimate distances to detected objects.

- **Visualization and Warnings:** Provides real-time trajectory visualization for tracked objects and issues visual and textual warnings for objects within a critical proximity.

The system processes dashcam footage and generates an annotated output video displaying bounding boxes, estimated distances, lane overlays, and warning indicators.

3.3. Object Detection and Tracking (YOLOv11 with ByteTrack)

The core object detection is performed by YOLOv11, a highly efficient and accurate real-time object detection model. The choice of YOLOv11 was driven by its fast processing capabilities, high accuracy, and flexibility. Another important reason for the choice is that YOLOv11 is the SOTA method having the best performance in road detection scenarios and having real-time capabilities. YOLOv11, integrated via the Ultralytics library (https://github.com/ultralytics/ ultralytics), was fine-tuned on the BDD100K dataset to detect road-relevant objects (cars, trucks, buses, people, motorbikes, bicycles). ByteTrack, also from Ultralytics, ensures robust multi-object tracking, maintaining consistent IDs despite occlusions. The model leverages GPU acceleration for fast inference, achieving high accuracy and flexibility in complex scenarios.

3.4. UltrafastLaneDetector for Lane Detection

Lane boundaries are identified using the UltrafastLaneDetector model, chosen for its speed and accuracy, particularly in complex real-world conditions. The UltrafastLaneDetector, trained on the CULane dataset, uses a ResNet-18 backbone and a modified parsingNet model. We increased the griding number to 400 and used 39 row anchors to improve detection of curved lanes. The model processes frames at high speed, producing color-coded lane overlays with 50% transparency for visualization.

3.5. Distance Estimation

The distance estimation module employs monocular vision to calculate the distance to detected objects using the pinhole camera model. The method, implemented in the **SingleCamDistanceMeasure** class, relies on the principle that an objects apparent size in the image is inversely proportional to its distance from the camera. The distance D (in meters) is computed as:

$$D = \frac{f \cdot W_{real}}{W_{pixel}}$$

where f is the cameras focal length (in pixels), W_{real} is the known real-world width of the object (e.g., 1.8 m for cars, 0.8 m for people), and W_{pixel} is the objects width in the image (in pixels). The focal length is pre-calibrated using a reference object at a known distance. The system uses predefined object widths for each class (e.g., cars, trucks, motorbikes) to estimate distances accurately.

The module processes YOLOv11 detections, extracting bounding box widths and class IDs to compute distances. Results are visualized with color-coded indicators (red for distances < 0.5 m, yellow for 0.5 – 2 m, green for > 2 m) and text overlays displaying the distance in meters. This approach enables real-time situational awareness, though it assumes known object sizes and clear visibility.

3.6. Visualization and Warnings

The visualization system integrates:

- Bounding boxes with class-specific colors and track IDs.
- Trajectory lines with fade effects for tracked objects.
- Lane overlays with reduced transparency.
- Warning indicators for objects closer than 0.5 meters.
- FPS and timestamp displays.

4. Evaluation

4.1. Used Datasets

Two datasets were used:

- **BDD100K**: Used for YOLOv11 fine-tuning, this dataset includes 100,000 driving videos with annotations for road objects (cars, trucks, buses, people, motorbikes, bicycles). The ***preprocess_bdd100k_yolo.py*** script converted the dataset into a YOLO-compatible format, filtering relevant classes and splitting data into training (80%) and validation (20%) sets. The data is available here: https://bair.berkeley.edu/blog/2018/05/30/bdd/
- **CULane**: Used for training UltrafastLaneDetector, this dataset contains 133,235 images with lane annotations. The ***lane_utils.py*** script implemented a CULaneDataset class to load and preprocess images, supporting 39 row anchors for dense lane point detection. The data is available here: https://xingangpan.github.io/projects/CULane.html

4.2. Training Setup

- **YOLOv11**: Fine-tuning was performed using the finetune_yolo.py script on a single NVIDIA RTX 3060 GPU with PyTorch 2.6 and CUDA 12.4. The process started with pretrained COCO weights (nano-version: yolo11n.pth) and used the BDD100K dataset. Key hyperparameters included a learning rate of 0.01 (with cosine annealing), batch size of 16, and 50 epochs. Data augmentation techniques (e.g., random flips, rotations, color jitter) were applied to enhance robustness. The model optimized a combination of box loss, classification loss, and DFL (Distribution Focal Loss) using the SGD optimizer with a momentum of 0.9. Validation was performed every 5 epochs to monitor mAP@50.
- **UltrafastLaneDetector**: Training was conducted on the CULane dataset using the same GPU setup. We cloned the original code repository of UltrafastLaneDetector (https://github.com/cfzd/Ultra-Fast-Lane-Detection/tree/master) and used the code ***train.py*** for training on our custom configuration. To improve accuracy, we applied modifications as follows:
    - We increases the griding number to 400 in ***configs/culane.py***:
    - We modified the row anchors to use 39 poiints in ***data/constant.py*****:** [100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150, 155, 160, 165, 170, 175, 180, 185, 190, 195, 200, 205, 210, 215, 220, 225, 230, 235, 240, 245, 250, 255, 260, 265, 270, 275, 280, 285, 287]

  Training ran for 50 epochswith a batch size of 32, learning rate of 0.05, and Adam optimizer.

4.3. Results

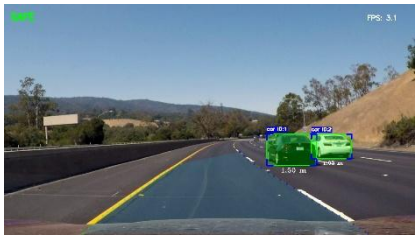

Figure 4. Frame 0

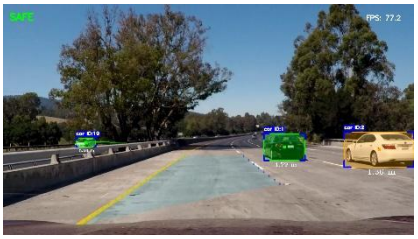

Figure 5. Frame 5

Figure 6. Frame 10

Video Information		Timing Statistics		Method Breakdown	
Length	10.04s	Total processing time	5.03s	Land detection	8.98ms/frame
FPS	25	Average time/frame	19.97ms	Object detection & tracking	5.6ms/frame
Total frames	251	Processing speed	49.92 FPS	Distance Estimation	0.08ms/frame

Compared to the initial approach, AVCAS reduced processing time by 97%, enabling real-time performance. YOLOv11 fine-tuning on BDD100K improved detection consistency, addressing YOLOv4 missed detection and processing time. UltrafastLaneDetector deep learning approach outperformed the Hough Transform in terms of processing time and curved lanes detection. However, in some cases with blurred lanes, the UltrafastLaneDetector failed to draw the line, requiring more optimization.

5. Strengths and Limitations

5.1. Strengths

- **Real-time Performance**: Achieves 50 FPS, suitable for real-time applications.
- **Robust Detection**: YOLOv11 and UltrafastLaneDetector handle complex scenarios, including curved lanes and occlusions.
- **Comprehensive Visualization**: Integrates lanes, objects, distances, and warnings into a clear dashcam-style output.

5.2. Limitations

- **Limited environment:** We trained the model for highway and sunny day environment only. Performance may decrease in bad weather and different environments.
- **Distance Estimation Accuracy:** Single-camera setups inherently provide less reliable distance measurements, particularly for distant or overlapping objects.
- **Hardware-Dependent Performance:** Real-time operation may be challenging on low-powered devices due to computational load.

- **Tracking Stability Issues:** Object tracking and trajectory accuracy degrade if detection is intermittent or objects are briefly occluded.

6. Future Work

- **Sensor Fusion:** Combine visual data with LiDAR or radar to improve robustness and accuracy.
- **Advanced Depth Estimation:** Use stereo vision or deep learning methods for more accurate distance measurement.
- **Expanded Object Detection:** Train models to recognize a wider range of road objects and hazards.
- **Real-time Optimization:** Enhance performance through lightweight model architectures and hardware optimization.
- **Improved Tracking Algorithms:** Adopt robust tracking methods (e.g., DeepSORT) for more stable trajectory tracking.
- **Adaptive Learning:** Integrate adaptive learning strategies to continuously enhance model performance under new and changing conditions.