PROJECT DONE BY :

NAME	RRN
MADHAN N	220051601051
HALABI AHAMED	220051601036
MOHAMED ABDUL HAMEED	220051601058
DULKARUNAIN RASEEN	220051601029
JOSHUA REX	220051601043

We all belong to- B.S. Abdur .Rahman crescent Institute of Science and Technology of second year.Pursing B.tech.Electronic and Communication Engineering .

GUIDED BY:
MR.INIYAVAN (A.P,sr.gr)

Problem Statement:

Vehicle Cut-In Detection System

In modern traffic management and autonomous driving systems, the accurate detection and prediction of vehicle cut-ins are crucial for ensuring safe and efficient driving. A vehicle cut-in occurs when another vehicle abruptly changes lanes into the path of the host vehicle. This behavior can pose significant risks if not detected and responded to promptly.

The goal of this project is to develop a robust and real-time vehicle cut-in detection system that can:

1. **Detect Cut-In Events**: Identify instances where another vehicle cuts into the lane of the host vehicle.

2.**Predict Future Movements**:Anticipate the trajectory and behavior of the cut-in vehicle to enable proactive decision-making by the host vehicle.

3.Ensure Accuracy and Reliability: Provide accurate detection under various driving conditions such as varying speeds, weather conditions, and traffic densities.

4. **Minimize False Positives**: Minimize false alarms by distinguishing between intentional lane changes and true cut-ins.

The system should leverage machine learning algorithms for object detection and trajectory prediction, while considering real-time constraints and ensuring minimal computational overhead. The final solution should be validated through rigorous testing scenarios that simulate typical and edge-case driving situations.

UNIQUE IDEA BELIF (SOLUTION):

1. **Vehicle Detection using Contours**: The `detect_vehicle` function uses edge detection (Canny) and contour detection to identify vehicles in a video frame. It draws bounding boxes around detected vehicles.

2. **Distance Estimation**: Although simplified and simulated (`estimate_distance` function), it provides a crucial parameter for calculating Time-To-Collision (TTC).

3. **TTC Calculation**: The `calculate_ttc` function computes the Time-To-Collision based on the estimated distance to the detected vehicle and a predefined relative speed.

4. **Alert Mechanism**: The `alert_driver` function alerts the driver (in this case, prints to the console) when a potential collision is imminent, based on the calculated TTC and a threshold (`ttc_threshold`).

5. **Real-Time Video Processing**: The ‘collision_avoidance_with_video` function processes a video stream frame by frame, detects vehicles, estimates TTC, and alerts if necessary. It uses OpenCV for video capture, frame manipulation, and display.

By enhancing these aspects, the collision avoidance system can become more robust, accurate, and suitable for deployment in real-world autonomous driving or advanced driver assistance systems (ADAS).

FEATURES OFFERED:

1. Vehicle Detection: The code detects vehicles in a video stream using edge detection (Canny) and contour detection techniques. It identifies regions of interest (ROI) where vehicles are likely present based on changes in pixel intensity.

2. Distance Estimation: Although simulated in this example, the code estimates the distance to detected vehicles. This estimation is crucial for calculating Time-To-Collision (TTC) and determining whether a collision is imminent.

3. Time-To-Collision (TTC) Calculation: Using the estimated distance and a predefined relative speed, the code calculates the Time-To-Collision (TTC). This metric helps assess the urgency of the collision threat.

4. Alert Mechanism: When a potential collision is detected (i.e., when TTC is below a specified threshold), the code alerts the driver. In the provided example, an alert message is printed to the console, indicating the estimated TTC.

5. Real-Time Video Processing: The code continuously processes video frames in real-time from a specified video file (`v.mp4`). It uses OpenCV for video capture, frame manipulation (resizing, edge detection, contour detection), and display.

PROCESS FLOW:

1. Initialization and Setup

Import Libraries: The code starts by importing necessary libraries such as `cv2` (OpenCV) and `numpy` for array operations.

`detect_vehicle(frame)`: Performs vehicle detection using edge detection (Canny) and contour detection.

`estimate_distance()`: Simulates distance estimation to detected vehicles.

`calculate_ttc(distance, relative_speed)`: Calculates Time-To-Collision (TTC) based on estimated distance and relative speed.

- `alert_driver(message)`: Prints an alert message to the console indicating potential collision.

2. Main Function (`collision_avoidance_with_video`)

Open Video Capture: Opens a video file specified by `video_path` using `cv2.VideoCapture`.

Video Processing Loop:Read Frames ,Frame Processing

3. Execution:

Example Usage: The `video_path` variable specifies the path to the video file (`v.mp4`) to be processed. This can be adjusted to use different video files for testing.

ARCHITECTURE DIAGRAM:

TECHNOLOGY USED:

1**. OpenCV (Open Source Computer Vision Library):**

- OpenCV is a widely used open-source library for computer vision and image processing tasks.

2**. Python Programming Language:**

- Python is chosen as the programming language for its simplicity, readability, and extensive library support.

.

3**. NumPy (Numerical Python):**

- NumPy is a fundamental package for numerical computing in Python.

4.**Simulation** :

- The system simulates certain aspects like distance estimation (`estimate_distance()` function) and relative speed to calculate Time-To-Collision (TTC).

.

TEAM MEMBER CONTRIBUTION

MADHAN : Prepared code for vehicle cut in detection and creating github**.**

ABDUL HAMEED: He has contributed in simulation of the code through vs code**.**

HALABI AHAMEED:He assisted in preparing the code and uploading the files.

JOSUA REX: He contributed ,the complete out come of the code my leading as a team leader.

DULKARNAI RASEEN: He has contributed preparing ppt and repote and final submission of the project successfully.

We all contribute in great cooperation ,the successful completion of vehicle cut-in detection.

CONCLUTION:

1. **Technological Foundation**: The system relies on technologies such as OpenCV, Python, and NumPy for real-time video processing, computer vision algorithms, and numerical computations. These tools provide a robust framework for implementing vehicle detection, distance estimation, and time-to-collision calculations.

2. **Team Collaboration**: A team of five members, including roles such as project management, computer vision engineering, software development, data science, and quality assurance, is essential. Each member contributes specialized skills to different aspects of the system's development and validation.

3. **Functional Components**: The system's architecture includes components for video capture, frame processing, vehicle detection, simulation-based distance estimation, and alert mechanisms. These components work together to identify potential collision scenarios and alert the driver accordingly.

4. **Simulation and Testing**: Simulation plays a crucial role in validating algorithms and system behavior under various conditions. It helps refine algorithms, optimize performance, and ensure the system meets safety and reliability standards before deployment.

6. **Future Considerations:** For real-world deployment, considerations include integration with vehicle hardware, testing in diverse driving conditions, and compliance with regulatory standards. Continuous monitoring and updates are necessary to adapt to evolving technologies and safety requirements.