Intel Unnati Industrial Training Program

(Project Report)

Vehicle Cut-in Detection

Submitted by:

Team Arise

Gokul Raj R

Govind Raj R

Karthik R

Sanjay Krishna K

Sreya Krishnan

Problem Statement

The problem statement focuses on creating a sophisticated machine learning model that can identify abrupt vehicle cut-ins and improve autonomous driving. The safety and effectiveness of autonomous cars, which significantly depend on precise and instantaneous identification of nearby objects and their movements, depend on this task.

The processing and interpretation of massive volumes of data in autonomous driving technologies is increasingly dependent on AI-based machine learning algorithms. The detection of sudden vehicle cut-ins, in which two-wheelers, three-wheelers, four-wheelers and pushcarts suddenly join the driving lane, is a significant difficulty in this field. For autonomous cars to navigate safely and to avoid accidents, accurate identification of this cut-in is essential.

Video link regarding the mentioned problem:

https://drive.google.com/drive/folders/1hMGf6OJ9Sz1gWAi01rpUSUaQRZC1rG7J

Solution

Developing an advanced machine learning model for detecting sudden vehicle cut-ins is critical for enhancing the safety and efficiency of autonomous driving systems. It includes the ability for real time detection of surrounding objects accurately.

The solution utilizes the Yolov8 model which is known for its efficiency in real time object detection. It recognises objects of various classes including vehicles. The system can continuously monitor the environment and identify objects and their movements. Using OpenCV, the system captures real-time videos and allows for detection of objects in the vehicle's path. The integration of polygonal region of interest(ROI) which enables detection of object in critical areas further enhances the efficiency of detection.

The SORT algorithm is used track detected vehicles across frames. By analyzing the movement of pattern of vehicles, the system can predict unexpected movements or cut-in.

The implementation of Time-to-collision(TTC) calculations allows to understand the risk posed by the detected vehicles by measuring the distance and relative velocity between the autonomous vehicle and potential cut-ins. If the TTC falls below the certain defined threshold, it generates a warning which allows the autonomous vehicle to react accordingly.

In summary, the combination of detection, tracking and risk assessment contribute to a more safety in driving system.

Github repository link:

https://github.com/Gokul-Raj-R-Coder/Vehicle_cut_in_detection_and_Warnning_system

```
# Polygon ROI coordinates
roi_pts = np.array( object: [[0, 720], [510, 350], [760, 350], [1280, 720]], dtype=np.int32)
```

- The region of interest has to be customised according to your video dimensions.

```
# Check if the vehicle is within the ROI polygon
if (cv2.pointPolygonTest(roi_pts, pt: (x2 - (w // 4), y2 - (h // 4)), measureDist: False) >= 0) or (
        cv2.pointPolygonTest(roi_pts, pt: (x1 + (w // 4), y1 + (3 * h // 4)), measureDist: False) >= 0):
```

- At first the vehicle will only be detected when the centre of the bounding box enters the ROI. To increase the accuracy of the cut in detection we have changed to coordinates to x2-(w//4) , y2-(h//4)

```
# Calculate distance and relative velocity
displacement_vector = (curr_cx - prev_cx, curr_cy - prev_cy)
dist_px = distance.euclidean( u: (prev_cx, prev_cy), v: (curr_cx, curr_cy))
dist_meters = dist_px * PIXEL_TO_METER  # Convert distance from pixels to meters
time_diff = current_frame_time - prev_frame_time
if time_diff == 0:
    continue

# Reference direction from camera center to object center
ref_vector = (curr_cx - 640, curr_cy - 720)
ref_magnitude_px = distance.euclidean( u: (640, 720), v: (curr_cx, curr_cy))
ref_magnitude_meters = ref_magnitude_px * PIXEL_TO_METER  # Convert distance from pixels to meters

# Relative velocity calculation (projected on reference direction)
relative_velocity_mps = (displacement_vector[0] * ref_vector[0] + displacement_vector[1] * ref_vector[
    1]) / (time_diff * ref_magnitude_px) * PIXEL_TO_METER
```

- The Euclidian distance will only provide positive value. So we are unable to get the direction of the relative velocity.

- Displacement Vector:

displacement_vector = (curr_cx - prev_cx, curr_cy - prev_cy)

This represents the change in position of the object between the current and previous frames in pixel units.

- Reference Vector:

ref_vector = (curr_cx - 640, curr_cy - 720)
This vector points from the center of the camera frame (assuming the frame's center is at (640, 720) in pixel units) to the current position of the object.

- Dot Product:

(displacement_vector[0] * ref_vector[0] + displacement_vector[1] * ref_vector[1])
This calculates the dot product of the displacement vector and the reference vector. The dot product projects the displacement vector onto the reference vector, giving a measure of how much of the displacement is in the direction of the reference vector.

- Time Difference:

time_diff
This is the time difference between the current and previous frames in seconds.

- Reference Magnitude:

ref_magnitude_px = distance.euclidean((640, 720), (curr_cx, curr_cy))
This calculates the Euclidean distance from the frame center to the current position of the object, representing the magnitude of the reference vector in pixel units.

```
# Conversion factor from pixels to meters
PIXEL_TO_METER = 0.001
```

This value is used to convert the pixel value to real world distance. The value can be get through camera calibration. The method we use:

Step 1: By using OpenCV, we measure dimension of a non object in the frame.
Step 2: Using the measured dimension in pixels, we can calculate the conversion factor.

$$\text{Conversion Factor} = \frac{0.05 \text{ meters}}{50 \text{ pixels}} = 0.001 \text{ meters/pixel}$$

Challenges faced

1. We where unable to find a good video dataset with satisfying cases
    - So we created a custom video dataset using iPhone 13 pro by placing it on the dash board of a car.
2. Accuracy of the distance and relative velocity calculation
    - By refining the equations for the calculations we were able to achieve a good accuracy. For further implementations hardware can be used to find the distance and relative velocity accurately.
3. Pixel to real world distance conversion.
    - By doing proper camera calibration we can find the pixel to real world distance.

Features offered

- Real time object detection using YOLOv8.
- Employment of SORT algorithm for dynamic object tracking.
- Time-to-collision(TTC) calculation for detected vehicles
- Defines critical area as Region of Interest(ROI) for improving efficiency.
- Generation of real-time collision warning.
- Capability of detecting multiple types of vehicles
- Enhanced situational awareness.
- Improved safety and navigation.

Process flow

4. **Capturing Video Input:**

   - Start the process of capturing video from a camera or video file.

5. **Frame Recovery:**

   - Read frames from the video source continuously in order to process them.

6. **Region of Interest Definition:**

   - To identify possible cut-ins, define a polygonal ROI in the frame that focuses on important regions.

7. **Object Detection:**

   - To identify items in the current frame and obtain bounding boxes, confidence points, and class labels, use YOLOv8.

8. **Filtering Detections:**

   - To concentrate on pertinent items, filter out detections based on confidence thresholds and class (e.g., cars).

9. **Object Tracking:**

   - Track identified objects between frames by using the SORT algorithm, updating their IDs and locations.

10. **Assessment of Collision Risk:**

- Regarding monitored items inside the ROI:
  - Determine the Time-to-Collision (TTC) by dividing the distance by the relative speed..
  - Evaluate the possibility of unexpected cuts.

11. **Generation of Collision Warning:**

- To alert drivers to possible collision hazards, issue visual and/or audio alerts when the TTC falls below a predetermined threshold.

12. **Graphic Overlays:**

- For improved situational awareness, draw bounding boxes, TTC data, and speed measurements on the video frame.

13. **Visual Output:**

- Present the real-time processed video frames with overlays.

14. **Cleanup:**

- When processing is finished, release the video resources and close any open windows. The system's ability to efficiently monitor its surroundings, identify possible threats, and react properly to guarantee safe navigation is ensured by this well-organised flow

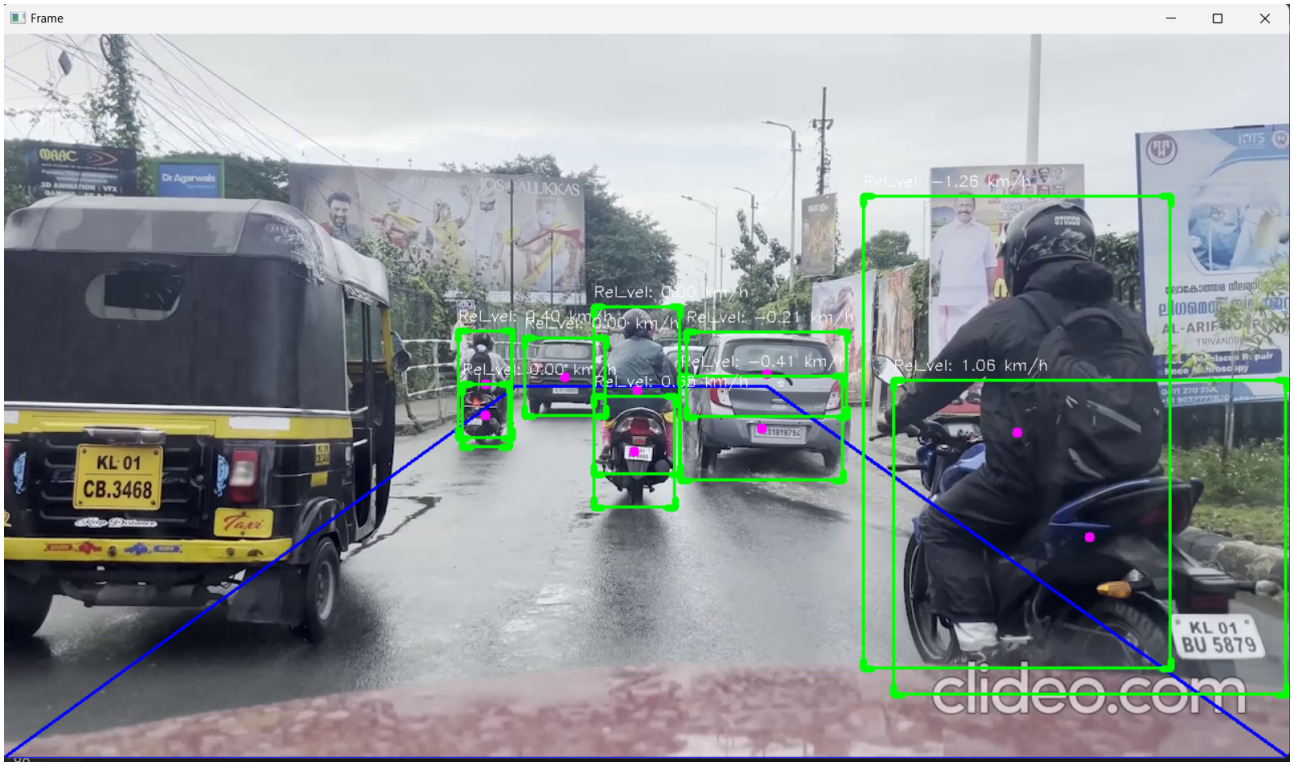
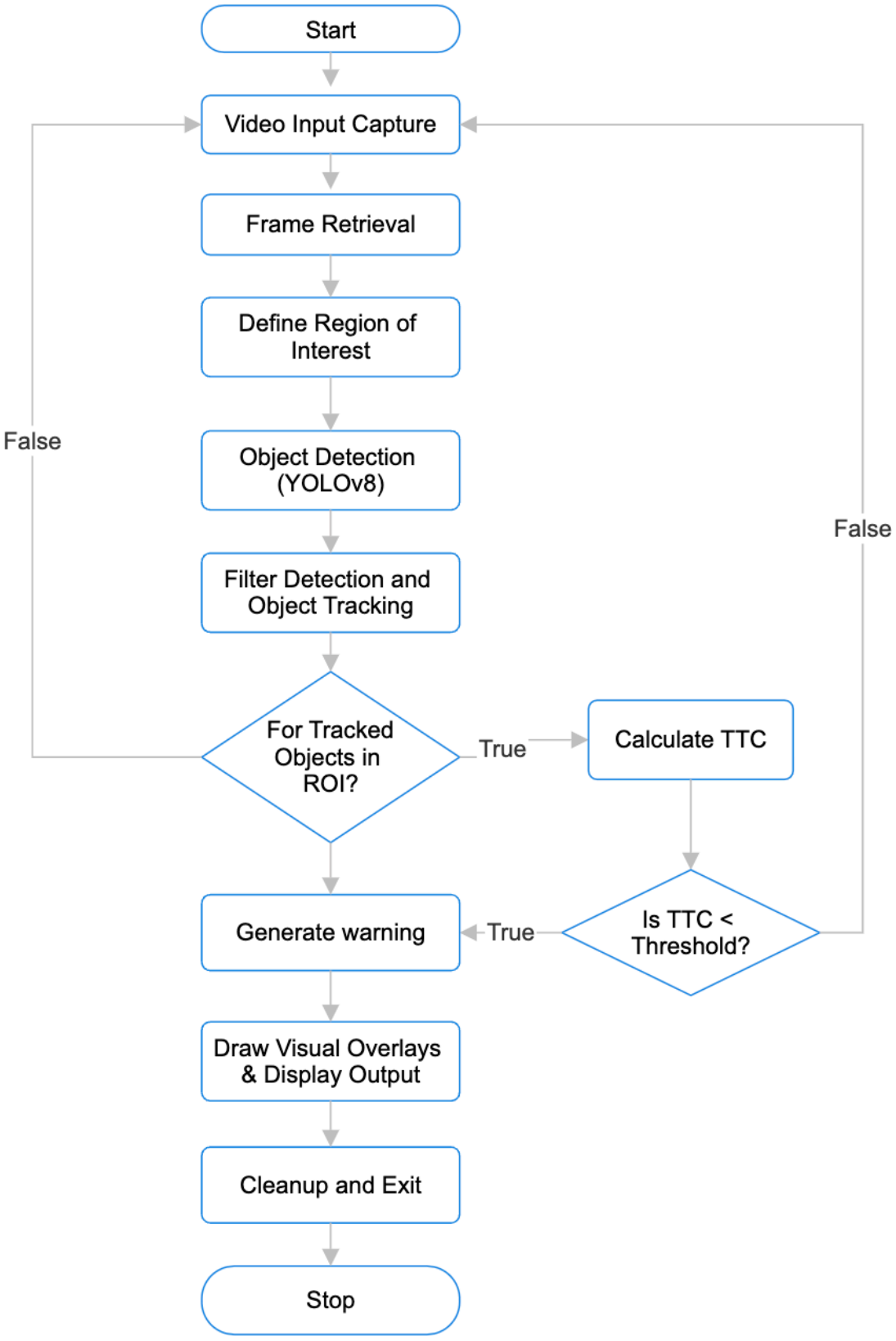


Fig (i) Cut in detection of Two wheelers

Fig (ii) Cut in detection of Four wheelers

Architecture Diagram



Technologies used

- **YOLOv8**: An advanced deep learning model for real-time object detection which enables identification of objects with high accuracy
- **OpenCV**: A computer vision library used for video capture
- **SORT (Simple Online and Realtime Tracking)**: An algorithm for tracking detected objects across video frames.
- **NumPy**: A library for numerical operations which facilitates array manipulation and calculations such as distance and velocity.
- **SciPy**: Specifically, its distance module is used to calculate Euclidean distances for determining relative positions of objects.
- **Python**: The programming language used for implementing the entire solution
- **Computer Vision Techniques**: Various techniques for image processing and analysis.

Team members and contribution

Name	Contribution
Gokul Raj R	Coding
Govind Raj R	Coding
Karthik R	Coding
Sanjay Krishna K	Report writing
Sreya Krishnan	Report writing

Conclusion

To sum up, the vehicle cut-in detection project aims to improve the safety of autonomous driving by utilizing YOLOv8 for precise object identification and SORT for reliable tracking. The technology effectively monitors regions that are vulnerable to cut-ins by creating a Region of Interest (ROI). It uses relative velocity and distance to calculate Time-to-Collision (TTC), forecasting possible collisions by sending warnings when TTC is less than 0.7 seconds. TTC calculations are made with accuracy thanks to the conversion of pixels to real-world distances, and the display of object relative velocities provides further context. Modern object identification, tracking, and predictive analytics are used to greatly improve autonomous driving technology and make vehicle navigation safer and more dependable.