Lane and Vehicle Detection for Autonomous Systems

V.E.Sudarsan(22116098), Vaibhav Bothra(22116101)
Akarsh Prasad(22116010), Ayushman Jha(22116025)

***Abstract*—This report discusses the creation of a lane and vehicle detection system designed for autonomous vehicles. Using computer vision methods, the system identifies road lanes and nearby vehicles from real-time video feeds. The main goal of the project is to accurately detect road lanes to assist in autonomous driving, with additional objectives including vehicle detection, traffic density analysis, and calculation of relative speed between vehicles. The report outlines the approaches taken and the technical aspects of the code.**

I. Introduction

Lane detection plays a vital role in the functionality of autonomous vehicles, offering significant real-world applications. It enables vehicles to remain within their designated lane, maintain safe speeds, and manage appropriate distances from other cars, ensuring secure navigation. This system leverages computer vision techniques to identify and monitor road lanes and surrounding vehicles. By analyzing video frames captured by a camera, the goal is to develop a real-time solution that can be integrated into autonomous vehicles.

II. Our Solution

The main objective of this project is to implement a system capable of detecting lane boundaries and surrounding vehicles in real-time. Our approach consists of two primary components: lane detection and vehicle detection. In addition, we also aim to estimate traffic density based on the number of vehicles detected.

A. *Lane Detection*

Lane detection involves identifying the boundaries of the lane in which the vehicle is traveling. Our approach for lane detection is as follows:

- **Image Preprocessing:** The captured frames are converted to grayscale to simplify the data. A Gaussian blur is applied to reduce noise and improve edge detection accuracy. The lanes we focus on are typically white and occasionally yellow, so to separate these elements, we use a yellow and white mask. Following this step, we will be left with the components that are either white or yellow. The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given in Figure 1.
- **Edge Detection:** We use the Canny edge detection algorithm to find significant changes in pixel intensity, which correspond to the edges of the lane. The lanes have sharp intensity changes, and edge detection isolates these lines. The process involves applying the Gaussian filter, computing the gradient of image using Sobel operators in x and y direction, refining the edges with non maximum suppression and applying thresholding to get strong and weak edges.
- **Region of Interest (ROI):** The frame that we are analyzing consists a lot of irrelevant information, like the sky and the trees, which acts as a noise, in relative to our current information, and thus we define a region of interest, which is a trapezium as we observed how the lanes diverge from the driver perspective forming an approximate trapezium. This region of interest, helps us to focus on important parts in our whole analysis.
- **Perspective Transformation:** Analyzing lanes and road features from a standard camera perspective can be complex, particularly when estimating lane curvature or measuring distances. The distortion from the camera's angle makes it harder to interpret lane boundaries accurately. To overcome this, we apply a bird's-eye view transformation using a matrix (Matrix A) that maps key source points to their corresponding top-down positions. This transformation straightens the curved lane lines, making them parallel and easier to detect. By converting the image to this top-down view, we simplify lane analysis, improve curvature estimation, and enhance the overall accuracy of lane detection systems.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

Fig. 1. Gaussian Formula

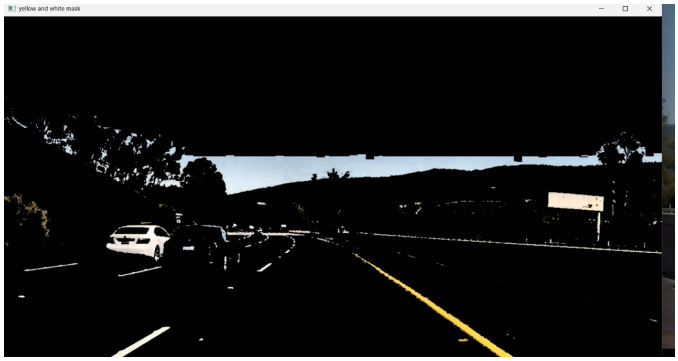

Fig. 2. Yellow and White Mask

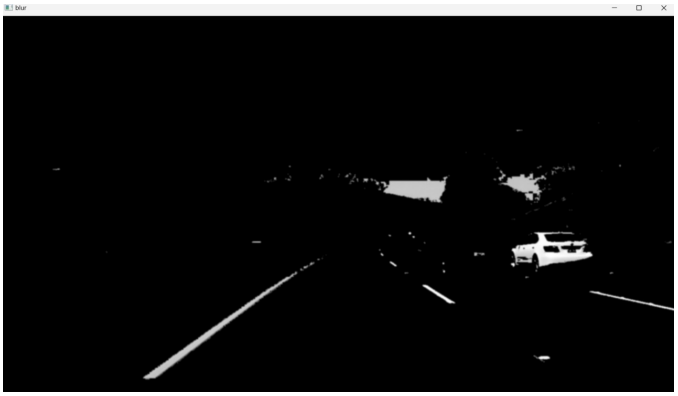

Fig. 3. Gray and Blurred Image

Fig. 4. Edge Detection

Fig. 5. Region Of Interest

Fig. 6. Bird's Eye View

- **Lane Line Detection:** To find the lane pixel points on the top view image a histogram was calculated along the x values which maps the intensities along with the x coordinates. The bottom part of the top view image was considered because that is the part which is nearer to the car and thus, consists of lane pixels that is nearer to the car, which is of more significance since it tells us the immediate lane pixel points.
  After that the histogram is divided into the two parts, by calculating the midpoint of the histogram now, the peak intensity value point on either half is most likely gonna be the lane pixel points on the left and right half, cause lane pixels take a significant part in the binary top view image.
  Before calculating the left and right peak points, we convolve the histogram with a kernel containing all ones acting like the box filter (mean filter kernel) which smoothens out the noise, so as to suppress the secondary peaks or artifacts which may be there, and the primary peaks which are the lane pixels, are identified properly.
  After convolving, the left and right peaks are calculated. After then, we calculate all the nonzero pixel locations in the binary top view image. The pixels which are near the peaks are masked, as they are likely to be the part of lane. And these points are added to the lists as the lane pixels, which is returned at the end of the function.
- **Curvature Calculation:** The curvature of the lane is calculated from the polynomial coefficients that are scaled to real world units that best fits the lane points. This helps in understanding the sharpness of turns. The radius of curvature R for a quadratic polynomial $x = Ay^3 + By^2 + Cy + D$ is calculated as:

$$R = \frac{\left(1 + \left(3Ay^2 + 2By + C\right)^2\right)^{1.5}}{6Ay + 2B}$$

  where:
  - A: The quadratic coefficient of the polynomial.
  - B: The linear coefficient of the polynomial.
  - y: The vertical position at which the curvature is evaluated.
- **Steering Direction** : The steering direction logic comes from the assumption that the dashcam of the car is mounted at the physical center of the car, such that the center of the frame of the camera coincides with the physical center of the car. Then the lane center is calculated by looking at the bottom most points of the fitted lane pixel points, as they are nearest to the car and are of more importance in immediate steering prediction. The car center is calculated by calculating the center of frame width, i.e., FrameWidth/2. Then the difference

Fig. 7. Lane Image with Radius of Curvature

between the car center and the lane center is calculated. So if the lane center lies to the left of the car center this means the car has to steer left so as to align with the center of the lane, and vice versa logic for the right steering. A threshold of 50 pixels was chosen, so if the difference between the lane and car center was less than 50 px, there was no need of steering, and the car should continue straight.

The radius of curvature could also be used to detect the steering direction, by seeing the ROC of the left and right lanes individually, but that, didn't give satisfactory results, moreover it suffered from problems, like infinite radius of curvature for a straight line.

But the radius of curvature value could tell us about the sharpness of the turn, a lower value of ROC corresponds to a sharper curve and a higher value represents a gentle curve.

B. *Vehicle Detection*

To enhance the functionality of the system, we also detect surrounding vehicles using a deep learning approach:

- **Machine Learning Model:** The machine learning model utilizing HOG (Histogram of Oriented Gradients) feature extraction delivered suboptimal performance, achieving only 70% accuracy. Although it detected some vehicles correctly, it also misidentified background elements like trees and road sections, resulting in false positives and compromising the model's overall reliability.
- **YOLOv3 Object Detection:** We use the YOLOv3 model for real-time object detection, which works by dividing the image into a grid. For each grid cell, it predicts bounding boxes by considering the confidence score and class id for all the objects, such as vehicles, allowing for efficient and accurate identification in real-time. If the model finds the object to be class id of "car" and its confidence score is greater than 0.5, it is considered to be the car. It uses NMS (non-maximum suppression) to discard the overlapping of multiple bounding boxes. These models analyze the scene to detect objects based on predefined classes, such as cars, pedestrians, and bicycles.

Detecting other vehicles on the road is essential for tasks such as adaptive cruise control, collision avoidance, and dynamic path planning.

Fig. 8. Vehicle Detection

- **Vehicle Tracking and Relative Speed Estimation:** Vehicle tracking and speed estimation involve identifying and monitoring vehicles across consecutive video frames to measure their movement over time. The process starts with detecting vehicles using an object detection model such as YOLOv3, which provides the bounding boxes and center coordinates of each vehicle. These detected vehicles are then tracked from frame to frame. Speed is estimated by calculating the displacement of the vehicle's center between frames in pixels and converting it into real-world units using a predefined scale (e.g., meters per pixel). This displacement is multiplied by the frame rate to obtain the speed in meters per second, which can then be converted to kilometers per hour. By integrating detection, tracking, and displacement measurement, this approach delivers precise vehicle tracking and real-time speed estimation.
- **Traffic Density Estimation:** The number of vehicles in a specific area is monitored to estimate traffic congestion, categorized as low, medium, or high. This helps drivers and traffic authorities to make better decisions, plan routes, and reduce travel time, leading to smoother, more efficient road conditions for everyone.

Fig. 9. Speed Estimation and Traffic Density Estimation

C. *Relative Speed Calculation*

Relative speed estimation is crucial for understanding the movement of detected vehicles relative to the ego vehicle

(vehile under consideration). In our system, we compute the relative speed based on the displacement of the vehicle's center between consecutive frames.

- The center of each detected vehicle is tracked across frames, and the distance traveled is calculated from the change in position between the frames.
- The relative speed v is given by the formula:

$$v = \frac{d}{\Delta t}$$

  where d is the distance (change in pixel positions) and Δt is the time between frames, yielding speed in pixels per second.
- To convert this speed to real-world units, pixel measurements are calibrated using the camera's field of view and perspective transformation. Assuming the distance per pixel is 0.04 as it depends on the camera and field of view. For general, the ideal range is 0.02 – 0.04.

Fig. 10. Final Output

III. Contributions

- **Integration:** All the different parts of the code are integrated by everyone.
- **V.E. Sudarsan:** Image preprocessing, edge detection, region of interest and Perspective Transformation along with Vaibhav.
- **Vaibhav Bothra:** Perspective Transformation along with Sudarsan, Lane Line Detection along with Akarsh and car detection along with Ayushman and vehicle tracking and relative speed estimation.
- **Akarsh Prasad:** Lane Line Detection along with Vaibhav, curvature calculation and steering direction prediction.
- **Ayushman Jha:** Polynomial Fitting, car detection along with Vaibhav and traffic density estimation.

IV. Applications

- **Autonomous Driving System :** The lane detection is useful for the autonomous driving as it helps the system to stay in the same lane, and also predict the steering direction. The traffic density calculation helps to predict the speed, for instance, if the density is more, then the system drives at a lower speed. Also the relative speed, helps to apply break if the relative speed of the vehicles ahead is decreasing.
- **Advanced Driver-Assistance Systems (ADAS) :** Alerts drivers about lane departure or vehicles in close proximity. *For example, Audi's lane departure warning system vibrates the steering wheel when it detects unintended lane changes.*
- **Traffic Monitoring and Management :** By detecting curves and predicting turns, traffic systems can analyze road conditions in real time. This helps spot areas with congestion or potential hazards, allowing for better management of traffic flow and improved safety on the roads.

V. Conclusion

In this project, we successfully developed a lane detection and vehicle detection system for autonomous vehicles. Our solution combined traditional computer vision techniques with modern deep learning models to detect lane boundaries and vehicles in real-time. The system also included the estimation of traffic density, which adds value to the autonomous driving experience. The contributions of each team member were integral to the development of this project, and the results demonstrate the effectiveness of the combined approach.