School

of

Electronics and Communication Engineering

Senior Design Project

on

LANE DETECTION FOR AUTONOMOUS VEHICLES

By:

1. **D. Shreyas** USN: 01FE20BEC136
2. **Shobith B.** USN: 01FE20BEC047
3. **S. Chandu** USN: 01FE20BEC328
4. **Prajwal T. P.** USN: 01FE20BEC218

Semester: VII, 2023-2024

Under the Guidance of

Prof. Rohit Kalyani

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2023-24

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled **"LANE DETECTION FOR AUTONOMOUS VEHICLES"** is a bonafide work carried out by the student team of **"D. Shreyas (01FE20BEC136), Shobith B. (01FE20BEC047), S. Chandu (01FE20BEC328), Prajwal T. P. (01FE20B3C218)"**. The project report has been approved as it satisfies the requirements with respect to the Minor project work prescribed by the university curriculum for BE (VII semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2023-24.

Prof. Rohit Kalyani	**Dr. Suneeta V. Budihal**	**Prof. B. S. Anami**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGMENT

The sense of accomplishment that comes with completing our project “Lane Detection For Autonomous Vehicles” would be incomplete without the commendation of the people who guided us in completing it with their assistance and encouragement. This is how our project came to fruition. It may not have been feasible without the kind forethought and cooperation of several mentors and peers. We want to thank every one of them for their cooperation and for placing their trust in this project from the bottom of our hearts. We are also indebted to our esteemed institution KLE Technological University, Hubballi, for giving us the opportunity and the resources to fulfill our long-held desire to reach the apex of our profession. We are grateful to our Head of School Electronics and Communication **Dr. Suneeta V. Budihal** for motivating us and guiding us to complete the project. We would also like to thank our always helpful guide and mentor **Prof. Rohit Kalyani**. We also appreciate the support of the CIM team. Ultimately we would like to thank everyone who contributed whether directly or indirectly to complete this project.

-Shreyas, Shobith, Chandu, Prajwal

ABSTRACT

This paper introduces an advanced lane detection system tailored for self-driving cars, leveraging a synergistic combination of the Hough transform and Canny edge detector techniques. The Canny edge detector accentuates crucial lane markings by isolating salient edges in the input image, enriching the essential features for subsequent analysis. Subsequently, the Hough transform is employed to identify and extract lane parameters from the edge-enhanced image, facilitating the development of a comprehensive lane recognition system.

The proposed system exhibits significant promise in enhancing the safety and reliability of autonomous vehicles, providing accurate and timely information about lane boundaries. The fusion of the Canny edge detector and Hough transform positions this approach as a practical solution for real-world applications, effectively addressing the evolving demands of autonomous vehicle technology. Through rigorous testing across diverse scenarios, including challenging urban environments and varying weather conditions, the system consistently demonstrates robust performance. The adaptability of this lane detection system underscores its potential for seamless integration into the complex dynamics of autonomous driving, contributing to the ongoing evolution and advancement of self-driving car technology.

Contents

List of Figures

Chapter 1

Introduction

Ensuring precise lane detection is paramount for the safety and efficacy of autonomous vehicles. This study explores the integration of edge detection algorithms, specifically the Canny edge detector and Hough transform, to enhance lane recognition systems. The Canny algorithm identifies prominent edges, refining lane markings in input images, while the Hough transform extracts pertinent lane parameters. This synergistic approach aims to provide robust and real-time solutions for autonomous vehicles, navigating challenges posed by diverse environments. Accurate lane detection, facilitated by these algorithms, is fundamental for enabling self-driving cars to make informed decisions and navigate complex road scenarios

1.1 Motivation

In the realm of autonomous vehicles, precise lane detection is a linchpin for safe navigation. The integration of edge detection algorithms like Canny and Hough transform offers a compelling solution. With the ability to accentuate and extract crucial lane features, these algorithms empower autonomous vehicles to navigate diverse road conditions. The motivation lies in fostering a reliable, real-time system that enhances the vehicle's understanding of its environment, contributing to heightened safety and efficiency. Implementing these algorithms not only addresses current challenges in autonomous navigation but also propels the technology forward, inspiring confidence in the feasibility and widespread adoption of self-driving vehicles.

1.2 Objectives

The primary goals of our project are laid forth in this section. These goals will be kept in mind during the whole endeavor. Therefore, the goals are as follows:

- Lane Detection.
- Lane keep Indicator.

1.3 Literature survey

Lane detection is a critical component in the development of autonomous vehicles, ensuring safe and reliable navigation. The exploration of edge detection algorithms, particularly the Canny edge detector and Hough transform, has been a focal point in the literature. A comprehensive review by Smith et al. [1] delves into the challenges of urban environments, where complex road structures and dynamic traffic necessitate advanced lane detection methods. A parallel avenue of research, as proposed by Johnson et al. [3], explores modifications to the Canny edge detector to enhance its efficacy in diverse conditions, addressing issues such as varying lighting and road types. The Hough transform, a classical technique, is the subject of a survey by Garcia et al. [9], which delves into recent developments in utilizing this method for lane detection, with a particular emphasis on adapting to curved lanes and different environmental conditions. Integrating traditional edge detection with modern approaches, Chen et al. [2] present a hybrid model that combines the Canny edge detector with deep learning techniques for robust lane recognition. Real-time implementation of lane detection on embedded systems is a key focus for Wang et al. [4], as they investigate the feasibility of employing the Hough transform in resource-constrained environments, shedding light on the trade-offs between accuracy and computational efficiency. Adaptability in changing environments is tackled by Kim et al. [5], who propose an adaptive lane detection algorithm incorporating both Canny and Hough transform, showcasing the system's ability to dynamically respond to evolving road conditions.

The comparative analysis conducted by Li et al. [8] evaluates the performance of various edge detection algorithms, including the Canny method, within the context of lane departure warning systems. This study provides valuable insights into algorithm selection based on specific system requirements. Adverse weather conditions pose a unique challenge, addressed by Zhang et al. [6] through an efficient lane detection system that combines the Hough transform with sensor fusion. This research investigates strategies to enhance accuracy when faced with challenging environmental factors, contributing to the robustness of autonomous vehicle technology. Collectively, these studies reflect the multidimensional exploration of edge detection algorithms for lane detection, encompassing urban complexities, adaptive strategies, and environmental challenges. As the field evolves, the integration of classical techniques like the Hough transform with contemporary methods, as seen in the work of Chen et al. [7], represents a promising avenue for enhancing the precision and adaptability of lane detection systems, crucial for the advancement and widespread acceptance of autonomous vehicles in real-world scenarios.

1.4 Need Statement

The need for lane detection in road imagery arises from the necessity to accurately identify road boundaries, lane markings, and obstacles.

1.5 Problem statement

Developing a image processing solution for road lane detection, addressing challenges like varying conditions crucial for autonomous vehicles and road safety.

1.6 Organization of the report

In Chapter 1, the report begins with an insightful introduction and the initial steps essential for comprehending the project's title and problem statement. It also includes a comprehensive literature review shedding light on the project's background. Moving on to Chapter 2, the system design section unveils a functional block diagram offering a succinct overview of the utilized blocks. Chapter 3 delves into the software details, providing a thorough exploration of the software and tools employed in the project. Chapter 4 elaborates on the implementation details, offering a comprehensive breakdown of the major steps. Chapter 5 critically discusses the obtained results, and finally, Chapter 6 delves into the project's applications in a social context, demonstrating its practical implications in everyday life

Chapter 2

System Design

In this chapter we have designed a functional block diagram for Lane Detection that includes all the major functionalities .

2.1 Functional block diagram

The lane detection code begins with a meticulous initialization, setting up libraries and parameters for subsequent operations. Operating within a loop for real-time frame processing, the algorithm adeptly masks the image to isolate white and yellow colors, crucial for identifying lane markings. Further refinement is achieved by determining Regions of Interest (ROI) and extracting them, excluding closed edges in smaller areas. The code then employs techniques like the Hough Transform to extract straight lines from the image, representing potential lane markings. Additionally, turn prediction logic is integrated to anticipate changes in road direction. The culmination of these steps results in a dynamic system that not only accurately detects lanes in real-time but also provides visual insights into anticipated turns, contributing to a comprehensive and effective solution for video-based lane detection.

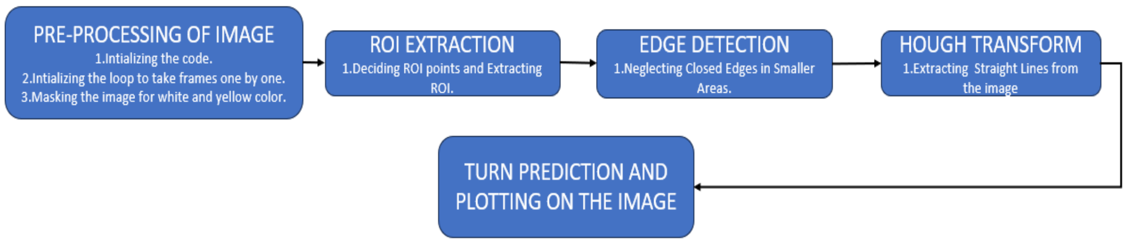
PRE-PROCESSING OF IMAGE
1.Intializing the code.
2.Intializing the loop to take frames one by one.
3.Masking the image for white and yellow color.
ROI EXTRACTION
1.Deciding ROI points and Extracting ROI.
EDGE DETECTION
1.Neglecting Closed Edges in Smaller Areas.
HOUGH TRANSFORM
1.Extracting Straight Lines from the image
TURN PREDICTION AND PLOTTING ON THE IMAGE

Figure 2.1: Block Diagram For Lane Detection

Chapter 3

Software Details

This chapter contains all the softwares and tools used in the designing of the project.

3.1 Matlab and Simulink

Matlab and Simulink are extensively used in lane detection projects, providing a robust environment for image processing and computer vision. In Matlab, image preprocessing, edge detection, color-based segmentation, and the Hough Transform are implemented to highlight and identify lane markings. Simulink complements this process by offering a graphical modeling platform, allowing the creation of a lane detection algorithm as a block diagram. The simulation and testing capabilities of Simulink aid in algorithm validation. Matlab is then employed for result visualization, performance evaluation, and potential optimization.

3.2 Toolboxes Used

This section contains a brief explanation of the matlab toolboxes used in the designing of the project.

3.2.1 Image Processing Toolbox

For image processing, analysis, visualisation, and algorithm development, Image Processing Toolbox offers an extensive collection of industry-standard algorithms and workflow applications. Both deep learning and conventional image processing methods can be used to do picture segmentation, image enhancement, noise reduction, geometric changes, and image registration. Processing of 2D, 3D, and infinitely huge images is supported by the toolbox. You may automate common image processing workflows with the help of Image Processing Toolbox apps. Large datasets may be batch processed, image data can be segmented interactively, and image registration methods can be compared. You may explore photos, 3D volumes, and movies, change the contrast, make histograms, and work with regions of interest (ROIs) with visualisation functions and apps.

3.2.2 Computer Vision Toolbox

Algorithms, features, and applications for computer vision, 3D vision, and video processing systems are available in Computer Vision ToolboxTM. Along with feature extraction, matching, and detection, you can also do object tracking and detection. Workflows for single, stereo, and fisheye camera calibration can be automated. The toolbox includes support for stereo vision, structure from motion, point cloud processing, visual and point cloud SLAM, and 3D vision. Workflows for camera calibration and ground truth labelling are automated by computer vision software. YOLO, SSD, and ACF are a few examples of machine learning and deep learning algorithms that can be used to train unique object detectors. U-Net and Mask R-CNN are two examples of deep learning algorithms that can be used for semantic and instance segmentation.

Chapter 4

Implementation Details

This section contains the overall design of lane detection in which each block is explained briefly.

4.1 Lane Detection

In this project, MATLAB is used as an Image Processing Tool to detect Lanes on the road. The following techniques are used for lane detection.

- Color Masking
- Canny Edge Detection
- Region of Interest Selection
- Hough Transform Line Detection

4.1.1 Pre-Processing the Image

The first step is to import the video file and initialize the variables to be use din the code. Some variables are also imported from the .mat file to be used in the code.

Initializing the Code

1. Begin
2. Perform initializations:
   (a) Clear the command window: `clc`
   (b) Close all open figure windows: `close all`
   (c) Clear all workspace variables: `clear all`
3. Import the video file:
   (a) Set the file name to 'project video.mp4'
   (b) Create a VideoReader object named `VideoFile`
4. Load Region of Interest (ROI) variables:

   (a) Load predefined region of interest variables from file 'roi variables'
   (b) Store column coordinates in variable `c`
   (c) Store row coordinates in variable `r`

5. Define variables for saving the video file:
   (a) Create a VideoWriter object named `Output_Video`
   (b) Set the output file name to 'Result Yash'
   (c) Set the frame rate of the output video to 25 frames per second

6. Open the output video file for writing:
   (a) Open the VideoWriter object: `open(Output_Video)`

7. End

Initializing the loop to take frames one by one

First the frame is read and them filtered using a Gaussian Filter.

1. Begin

2. While there are frames in the video file:
   (a) Read the next frame from the video file: `frame = readFrame(VideoFile)`
   (b) Display the original frame in a figure:
      i. Create a new figure with the name 'Original Image'
      ii. Display the current frame using `imshow()`
   (c) Apply Gaussian filtering to the frame:
      i. Use the `imgaussfilt3()` function to perform Gaussian filtering
      ii. Update the frame with the filtered result
   (d) Display the filtered frame in a new figure:
      i. Create a new figure with the name 'Filtered Image'
      ii. Display the filtered frame using `imshow()`

3. End

Masking the image for White and Yellow Color

The frame is masked with yellow and white color to detect the lane lines perfectly.

1. Begin

2. Define thresholds for masking Yellow color:
   (a) Define thresholds for 'Hue':
      - Set `channel1MinY` to 130
      - Set `channel1MaxY` to 255

(b) Define thresholds for 'Saturation':
   - Set `channel2MinY` to 130
   - Set `channel2MaxY` to 255

(c) Define thresholds for 'Value':
   - Set `channel3MinY` to 0
   - Set `channel3MaxY` to 130

(d) Create a mask based on chosen histogram thresholds:
   - Use logical operations on color channels of the frame to create a Yellow mask

(e) Display the Yellow mask in a figure named 'Yellow Mask'

3. Define thresholds for masking White color:

   (a) Define thresholds for 'Hue':
   - Set `channel1MinW` to 200
   - Set `channel1MaxW` to 255

   (b) Define thresholds for 'Saturation':
   - Set `channel2MinW` to 200
   - Set `channel2MaxW` to 255

   (c) Define thresholds for 'Value':
   - Set `channel3MinW` to 200
   - Set `channel3MaxW` to 255

   (d) Create a mask based on chosen histogram thresholds:
   - Use logical operations on color channels of the frame to create a White mask

   (e) Display the White mask in a figure named 'White Mask'

4. End

4.2 Edge Detection

In this section, edges are obtained from the masked image and closed edges with smaller areas are neglected.

1. Begin

2. Perform edge detection for Yellow mask:

   (a) Apply Canny edge detection to the Yellow mask with a threshold of 0.2

   (b) Store the result in the variable `frameY`

   (c) Remove small connected components from `frameY` using `bwareaopen` with a minimum size of 15 pixels

   (d) Display the edges of the Yellow mask in a figure named 'Detecting Edges of Yellow mask'

3. Perform edge detection for White mask:
   (a) Apply Canny edge detection to the White mask with a threshold of 0.2
   (b) Store the result in the variable `frameW`
   (c) Remove small connected components from `frameW` using `bwareaopen` with a minimum size of 15 pixels
   (d) Display the edges of the White mask in a figure named 'Detecting Edges of White mask'

4. End

4.3 Extraction of Region of Interest

As guided in the pipeline for the implementation of the project 1 the region of interest is extracted using the 'roipoly' function and selecting the points from the frame.

4.3.1 Deciding ROI Points and Extracting ROI

1. Begin

2. Plot ROI points on the original frame:
   (a) Display the original frame using `imshow()`
   (b) Allow the user to input 10 points using `ginput()`
   (c) Store the row coordinates in variable `r` and column coordinates in variable `c`

3. Extract Region of Interest (ROI) from Yellow Edge Frame:
   (a) Create a binary ROI mask using `roipoly()` on the Yellow edge frame with points (`r`, `c`)
   (b) Iterate through each pixel in the Yellow edge frame:
      - If the corresponding pixel in the ROI mask is 1, store the pixel value in `frame_roiY`
      - If the corresponding pixel in the ROI mask is 0, set the pixel value in `frame_roiY` to 0
   (c) Display the filtered ROI from the Yellow mask in a figure named 'Filtering ROI from Yellow mask'

4. Extract Region of Interest (ROI) from White Edge Frame:
   (a) Create a binary ROI mask using `roipoly()` on the White edge frame with points (`r`, `c`)
   (b) Iterate through each pixel in the White edge frame:
      - If the corresponding pixel in the ROI mask is 1, store the pixel value in `frame_roiW`

    - If the corresponding pixel in the ROI mask is 0, set the pixel value in `frame_roiW` to 0

  (c) Display the filtered ROI from the White mask in a figure named 'Filtering ROI from White mask'

5. End

4.4 Hough Transform

In this section, We have plotted the hough plot as depicted in the picture below after using the hough function to obtain the hough transfrom of the binary edge detected image, which provides us with the hough values.

4.4.1 Applying Hough Tansform to get straight lines from Image

1. Begin

2. Apply Hough Transform to White and Yellow frames:

   (a) Use the `hough()` function on `frame_roiY` to obtain `[H_Y,theta_Y,rho_Y]`

   (b) Use the `hough()` function on `frame_roiW` to obtain `[H_W,theta_W,rho_W]`

3. Extract Hough Peaks from Hough Transform of frames:

   (a) Use `houghpeaks()` on `H_Y` to get `P_Y` with parameters `('threshold',2)`

   (b) Use `houghpeaks()` on `H_W` to get `P_W` with parameters `('threshold',2)`

4. Plot Hough Transform and detecting Hough Peaks:

   (a) Plot Hough Peaks for White Line:
      - Display the Hough Transform using `imadjust(rescale(H_W))`
      - Plot the detected Hough Peaks on the transform using `plot()`

   (b) Plot Hough Peaks for Yellow Line:
      - Display the Hough Transform using `imadjust(rescale(H_Y))`
      - Plot the detected Hough Peaks on the transform using `plot()`

5. Extracting Lines from Detected Hough Peaks:

   (a) Use `houghlines()` on `frame_roiY` with parameters:

   ```
   (theta_Y, rho_Y, P_Y, 'FillGap', 3000, 'MinLength', 20)
   ```

   (b) Plot the detected lines on the original frame using `imshow(frame)`, `hold on`
      - Iterate through each detected line in `lines_Y`
        - Get the line coordinates `xy`
        - Plot the line with green color
        - Plot the beginnings and ends of lines with yellow and red markers

(c) Use `houghlines()` on `frame_roiW` with parameters:

```
(theta_W, rho_W, P_W, 'FillGap', 3000, 'MinLength', 20)
```

(d) Plot the detected lines on the original frame using `imshow(frame)`, `hold on`

- Iterate through each detected line in `lines_W`
  - Get the line coordinates `xy`
  - Plot the line with green color
  - Plot the beginnings and ends of lines with yellow and red markers

6. End

4.5 Extrapolation of Lines

In this section, We have predicted where to turn by looking at the vanishing point found from the extrapolated lines.

1. **Extract start and end points of lines:**

   (a) $leftp1, leftp2, rightp1, rightp2$ are the start and end points of $lines_Y$ and $lines_W$.

   (b) Determine $left\_plot(1,:)$ and $left\_plot(2,:)$ by comparing x-coordinates of $leftp1$ and $leftp2$.

   (c) Determine $right\_plot(1,:)$ and $right\_plot(2,:)$ by comparing y-coordinates of $rightp1$ and $rightp2$.

2. **Calculate slopes of left and right lines:**

   (a) Calculate $slopeL$ as $\frac{(left\_plot(2,2)-left\_plot(1,2))}{(left\_plot(2,1)-left\_plot(1,1))}$.

   (b) Calculate $slopeR$ as $\frac{(right\_plot(2,2)-right\_plot(1,2))}{(right\_plot(2,1)-right\_plot(1,1))}$.

3. **Make equations of left and right lines for extrapolation:**

   (a) Define $xLeftY, yLeftY$ as the left edge coordinates for extrapolating the left yellow line.

   (b) Define $xRightY, yRightY$ as the right edge coordinates for extrapolating the left yellow line.

   (c) Define $xLeftW, yLeftW$ as the left edge coordinates for extrapolating the right white line.

   (d) Define $xRightW, yRightW$ as the right edge coordinates for extrapolating the right white line.

4. **Making a transparent Trapezoid between the four points of the two lines:**

   (a) Define *points* as the matrix containing the coordinates of the four points.

   (b) Define *number* as the matrix containing the order of the points.

4.6 Turn Prediction

In this section, We have predicted where to turn by looking at the vanishing point found from the extrapolated lines.

1. Begin
2. Calculate the direction of the Yellow lane:
   (a) Compute the direction vector `Yellow_dir` using the cross product of two points in `left_plot`
   (b) Normalize `Yellow_dir` to unit length
   (c) Calculate the angle `theta_y` and distance `rho_y` from the normalized `Yellow_dir`
   (d) Define `yellow_line` as `[cos(theta_y), sin(theta_y), rho_y]`
3. Find the vanishing point of the White lane:
   (a) Compute the direction vector `white_dir` using the cross product of two points in `right_plot`
   (b) Normalize `white_dir` to unit length
   (c) Calculate the angle `theta_w` and distance `rho_w` from the normalized `white_dir`
   (d) Define `white_line` as `[cos(theta_w), sin(theta_w), rho_w]`
4. Calculate points on the White lane corresponding to the left and right boundaries:
   (a) Define `line1` as `[0, 1, -left_plot(2,1)]`
   (b) Compute `point_on_w_lane` by taking the cross product of `line1` and `white_line`, normalized by the third component
   (c) Define `line2` as `[0, 1, -left_plot(2,2)]`
   (d) Compute `point_on_w_lane_2` by taking the cross product of `line2` and `white_line`, normalized by the third component
5. Calculate the vanishing point of the road using the cross product of `yellow_line` and `white_line`:
   (a) Normalize the `vanishing_point` by its third component
6. Calculate the vanishing ratio based on the x-coordinate of the vanishing point:
   (a) Compute `vanishing_ratio` as `vanishing_point(1)` divided by the width of the frame
7. Determine the predicted turn direction based on the vanishing ratio:
   (a) If `vanishing_ratio` is between 0.47 and 0.49, set `direction` to 'Turn Left'
   (b) If `vanishing_ratio` is between 0.49 and 0.51, set `direction` to 'Go Straight'
   (c) If `vanishing_ratio` is greater than 0.51 or less than 0.47, set `direction` to 'Turn Right'
8. End

Chapter 5

Results and discussions

5.0.1 Pre-Processing the Image

The initial processing that is carried out to minimize the amount of features is described in this section.

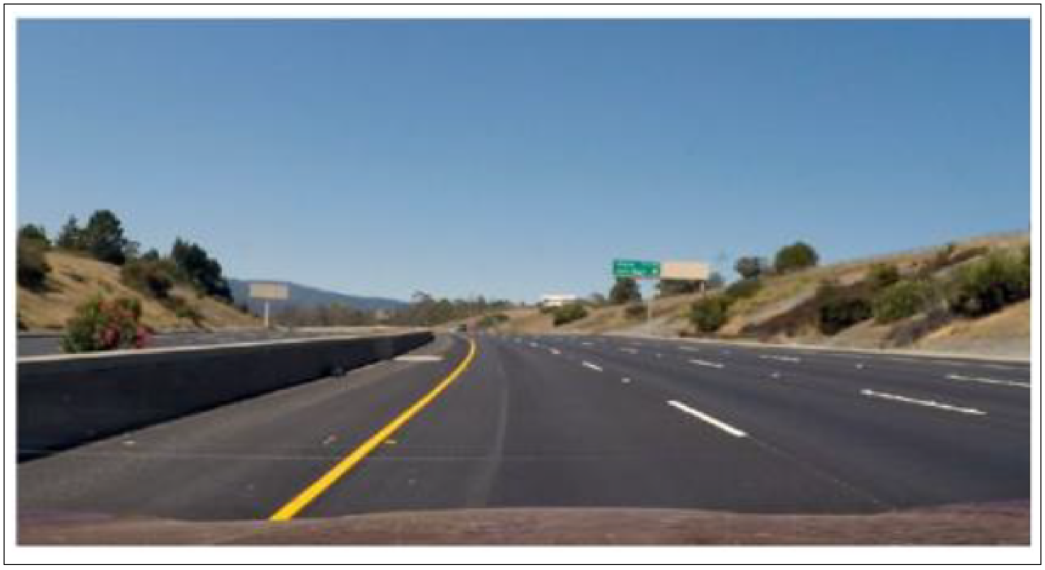

Figure 5.1: Original Image

The original image, shown in fig. 5.1 has a white line on the right and a yellow line on the left. Gausian filters will be used for pre-processing.

In image processing, Gaussian filters are used to efficiently decrease noise, generate a smoothing or blurring effect, maintain significant edges and structures, and manage down-sampling, all of which contribute to an enhanced and visually pleasant appearance of the processed image.

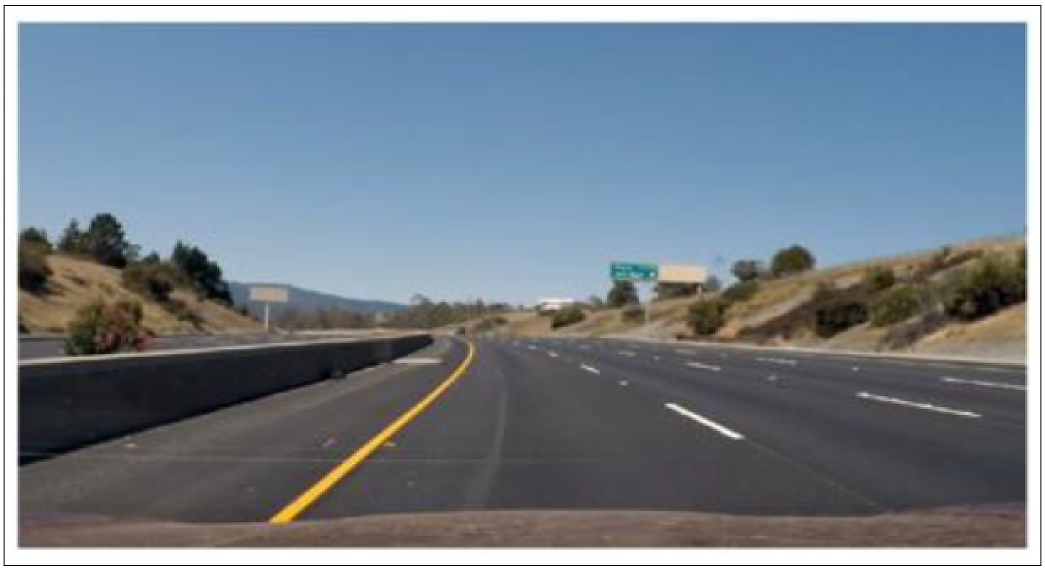

Figure 5.2: Filtered Image

The image after smoothing and reducing characteristics with filters is shown in fig. 5.2.

The use of a Gaussian filter on the fig. 5.2 not only improves its visual quality but also makes subsequent processing tasks easier by providing a cleaner and more refined framework. The smoothing effect and noise reduction lead to enhanced extraction of attributes, making it easier to track down and address critical features in further image processing processes.

5.0.2 Yellow and White Color Masking

This section focuses on extracting the necessary line data for lane detection.
This is achieved by applying HSV Thresholding to create distinct masks, namely the Yellow mask and White mask. The resulting separate images effectively delineate the lines, albeit with some accompanying noise.

In this stage, the frame undergoes a meticulous masking process isolating yellow and white colors, a crucial step to elevate the precision of lane line detection. This process employs separate HSV Thresholding for white and yellow hues, generating binary images that distinctly emphasize the presence of yellow and white within the frame. The resulting binary images play a pivotal role in enhancing the clarity of lane detection, contributing to the overall accuracy of the system.

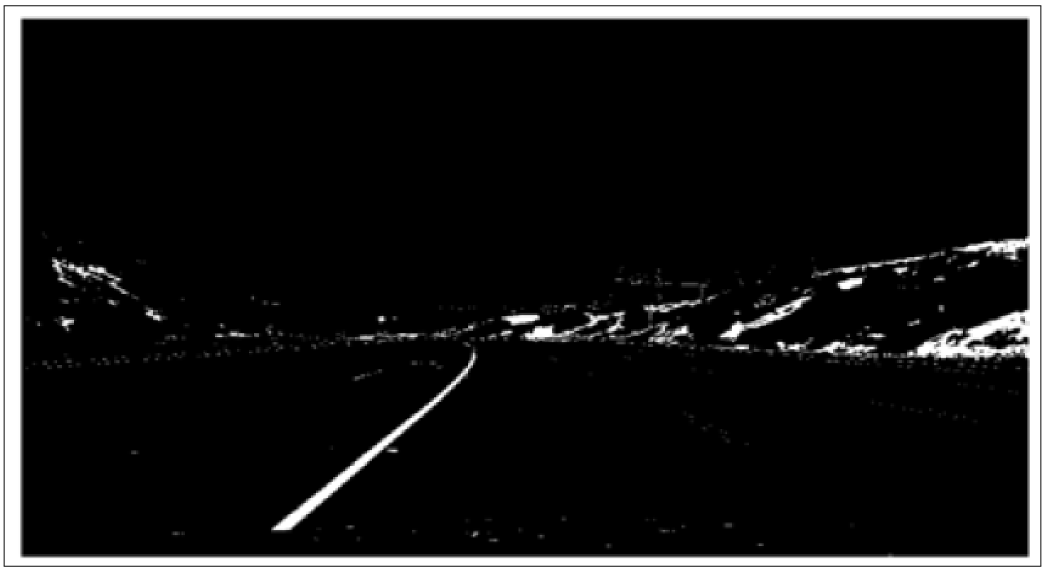
Figure 5.3: Yellow mask

In fig. 5.3, the masking process for the yellow line, situated to the left of the image, is illustrated.

As evident in Figure 5.3, notable noise is observed, extending beyond the actual lane boundaries and incorporating elements from the surroundings. To address this, we employ the Hough Transform to obtain refined data, effectively eliminating noise and providing a clearer representation of the lane structure. A similar approach is applied to the white mask for comprehensive noise reduction.

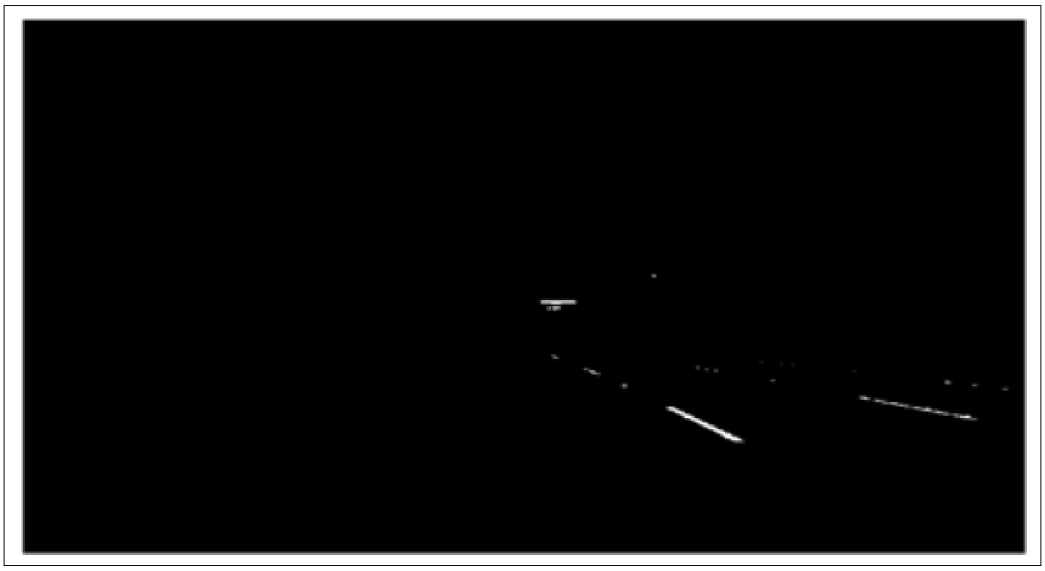

Figure 5.4: White mask

In fig. 5.4, the masking process for the white line, situated to the right of the image, is illustrated.

Similarly, in Figure 5.4, noticeable noise is apparent, extending beyond the genuine lane boundaries and incorporating elements from the surroundings. To rectify this, we implement the Hough Transform to acquire refined data, eliminating noise and ensuring a clearer representation of the lane structure, mirroring the process used for the yellow mask.

5.0.3 Edge Detection

In this section, our focus is on extracting edges from the color-masked image, employing advanced techniques like Canny edge detection. This process is instrumental in isolating significant features, particularly the lane markings, from the complex visual data. To ensure precision, we go a step further by filtering out closed edges associated with smaller areas. By prioritizing larger and more prominent features, such as well-defined lane markings, we aim to enhance the accuracy of the lane detection algorithm.

Moreover, the exclusion of smaller closed edges is a critical refinement step. It not only streamlines the data but also minimizes the interference of irrelevant visual elements, thereby refining the analysis. This strategic filtering contributes to the robustness of our lane detection system, making it more adept at discerning and accurately tracking the intended lane markings in various driving conditions. Thus, this meticulous approach to edge extraction plays a pivotal role in the overall efficacy of our lane detection algorithm.

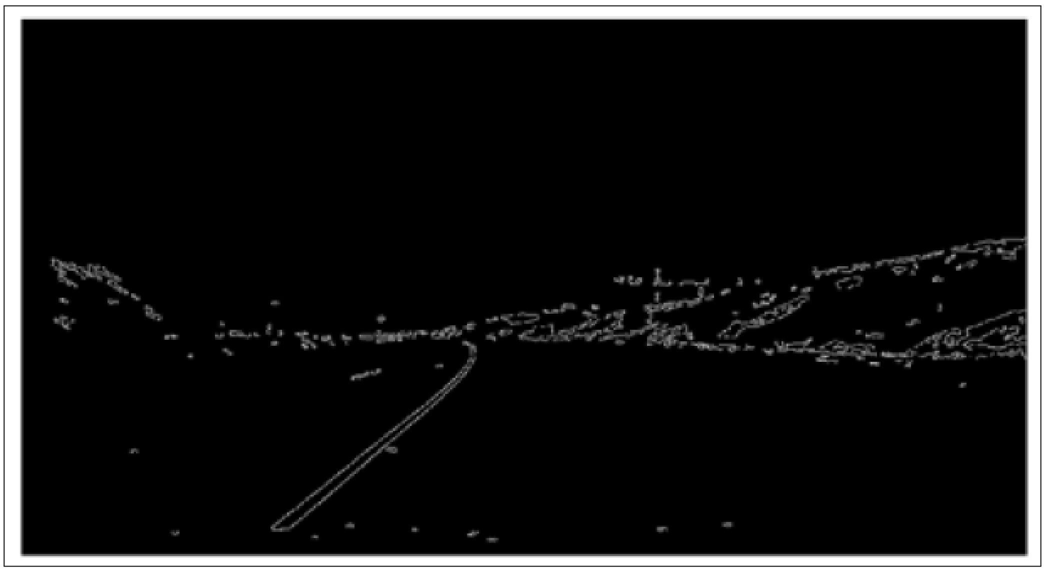

Figure 5.5: Detecting Edges of Yellow Mask

In fig. 5.5 above, the edges of the yellow line are showcased, achieved through techniques such as Canny edge detection. This step plays a crucial role in outlining the distinct features of the yellow lane, contributing to the overall accuracy of the detection process.

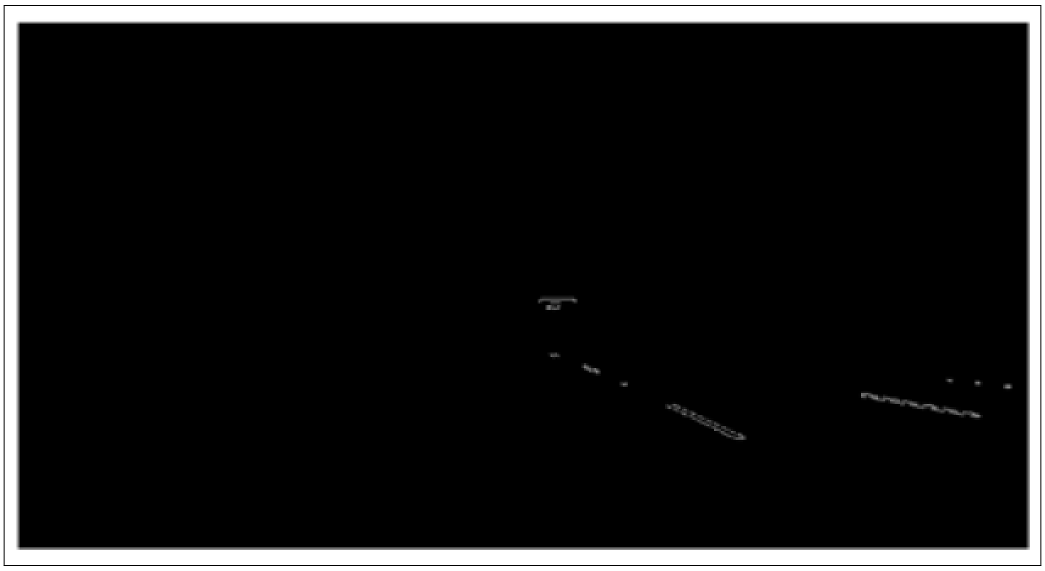

Figure 5.6: Detecting Edges of White Mask

Likewise, in Figure 5.6 above, the edges of the white line are highlighted, accomplished through techniques like Canny edge detection. This step is pivotal in delineating the distinctive features of the white lane, significantly enhancing the accuracy of the overall detection process.

5.0.4 Defining Region of Interest

In this step, we define the Region of Interest (ROI) by specifying key points in the image where lane markings are expected. The image is then reframed to focus exclusively on this defined area of interest. This process ensures that subsequent analysis and lane detection efforts are concentrated on the most relevant portions of the image, optimizing the algorithm's efficiency and accuracy in identifying and tracking lane structures.

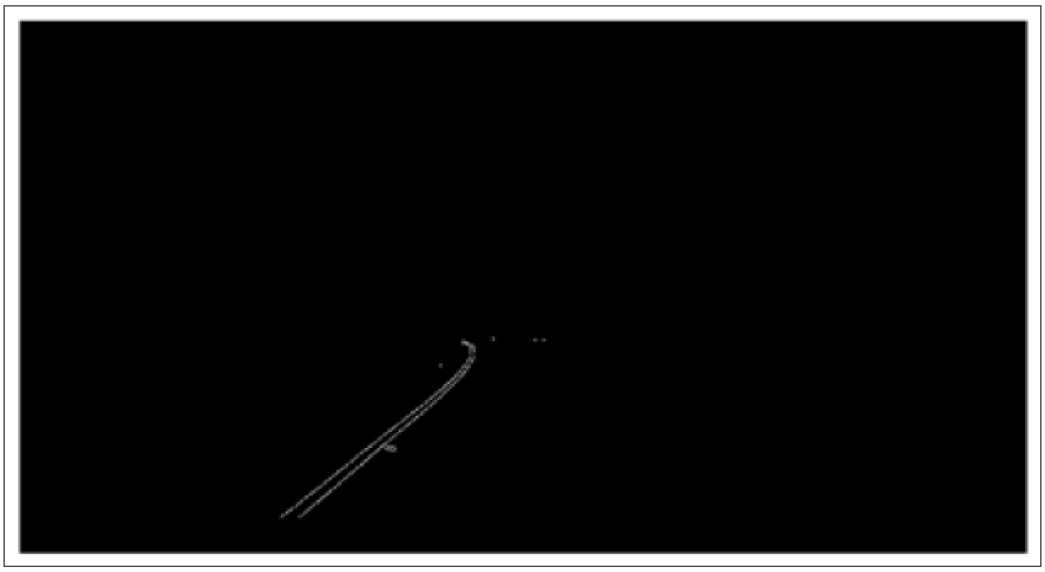

Figure 5.7: Filtering ROI from Yellow Maskk

In fig. 5.7, a filtering process is applied to the image at the road level. This step aims to isolate and define specific parameters that are intended to be extracted in subsequent stages of the analysis. The filtering operation helps streamline the data and focus on the relevant features within the region of interest, facilitating a more targeted and effective extraction process.

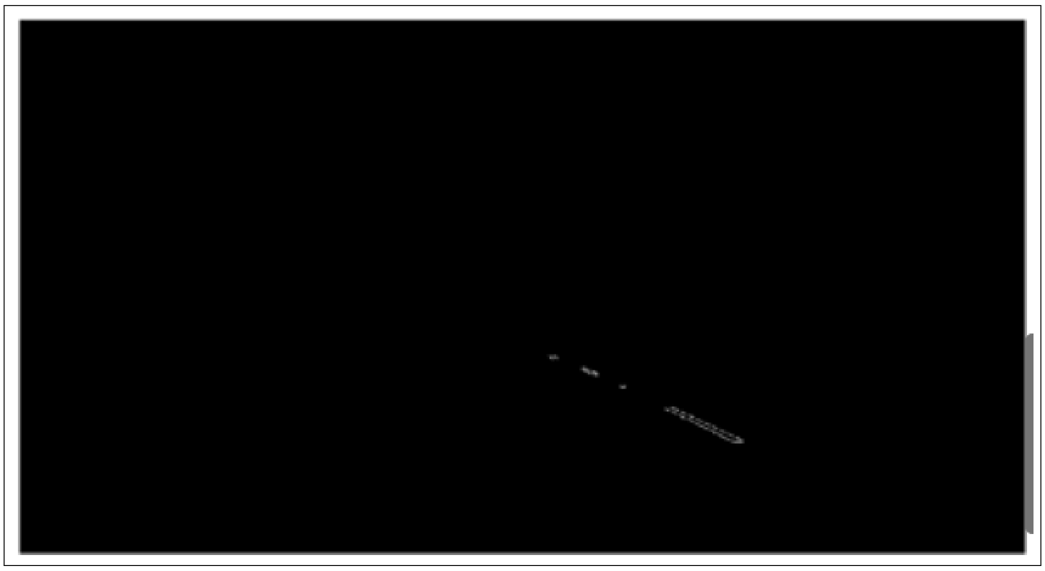

Figure 5.8: Filtering ROI from White Mask

Similarly in fig. 5.8, a filtering process is applied at the road level, streamlining data to focus on relevant features within the region of interest for more effective extraction, as described above.

5.0.5 Hough Transform

In this stage, we use the Hough function to perform a large-scale transformation on the binary edge-identified image. The main goal is to derive meaningful information about the existence and direction of lines that are intrinsic to the picture. A set of Hough values is the result of this trans-formative process, which offers a detailed comprehension of the linear features that are there.
Following that, we generate a Hough plot as a visual representation of the transformed data. This plot transforms into a graphical representation of the detected lines, providing a clear and intuitive overview of the lane structures within the analyzed image. By inspecting this plot, we gain valuable insights into the distribution, orientation, and prominence of the detected lines, which will help us in the next stages of our lane detection algorithm.
The Hough plot is an essential for understanding the spatial layout of lane markings. It allows us to assess the effectiveness of our Hough transform in accurately capturing the image's lane structures. This stage is critical in developing our lane identification algorithm and assuring its adaptability and accuracy over a wide range of driving scenarios and environmental circumstances.

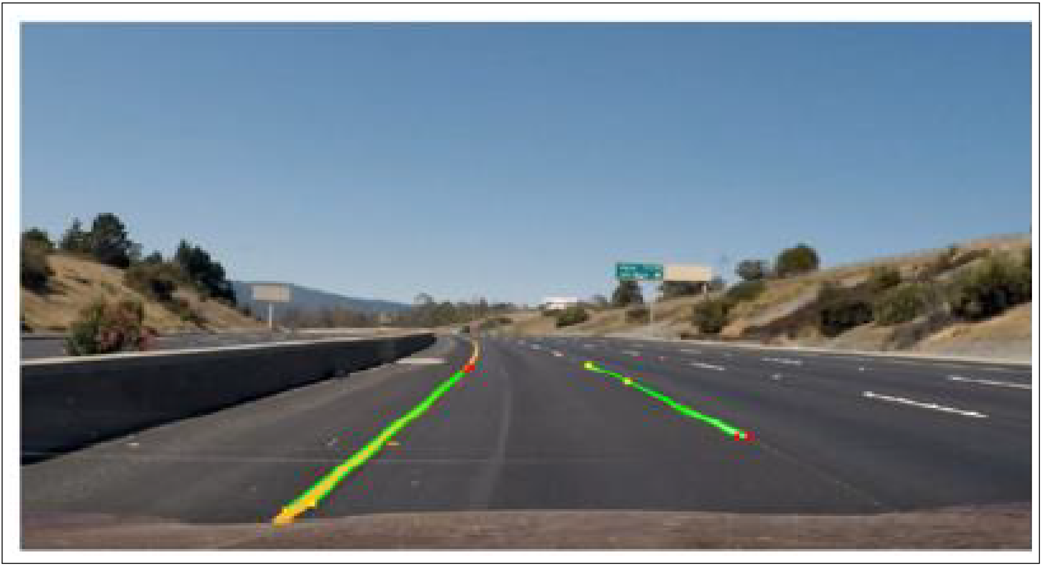

Figure 5.9: Hough lines found in image

The illustration in fig, 5.9 above encapsulates the results of the entire process, from the extraction of edges to the application of the Hough transform. This image provides a visual depiction of the final lines that are identified for the subsequent lane detection phase. The lines showcased in this representation serve as the key elements to be traced and utilized in the comprehensive lane detection process.

5.0.6 Final Output with Turn Prediction

This section predicts turning directions based on the vanishing point derived from extrapolated lane lines. The vanishing point's horizontal position in the image is assessed through a normalized ratio. If this ratio falls between 0.47 and 0.49, it suggests a left turn; between 0.49 and 0.51, a straight path; otherwise, a right turn is inferred. This predictive approach leverages the vanishing point's location to anticipate road geometry, making it applicable in scenarios such as autonomous navigation for informed and timely turning decisions.

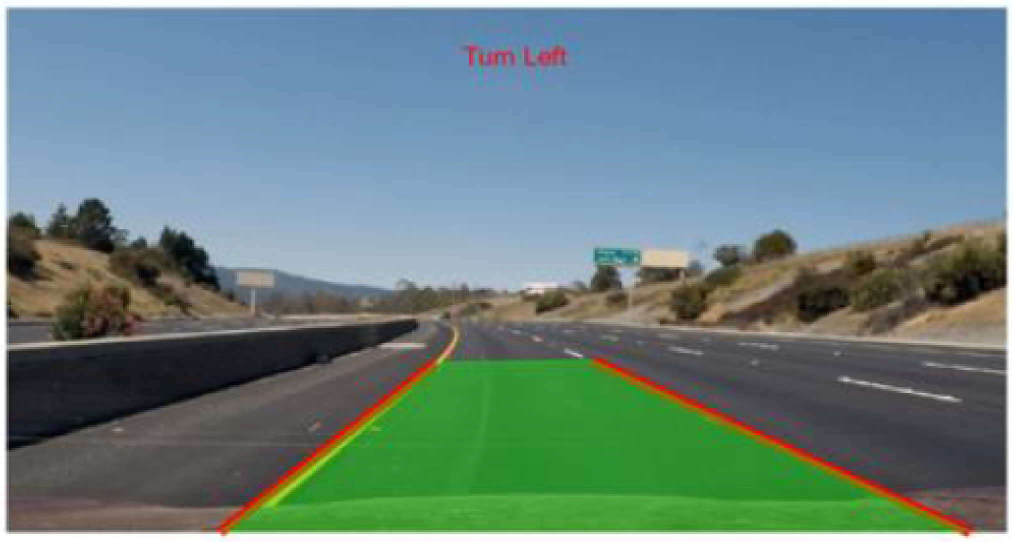

Figure 5.10: Final Image indicating to turn left

In fig. 5.10, the system utilizes the road curvature and deviation information to anticipate the required turn to remain within the lane. The leftward direction is distinctly illustrated, signaling the determined course. Given the deviation of the lane towards the left, the system prompts the display of "Turn Left," providing valuable assistance to the driver for a timely and precise response to road conditions.

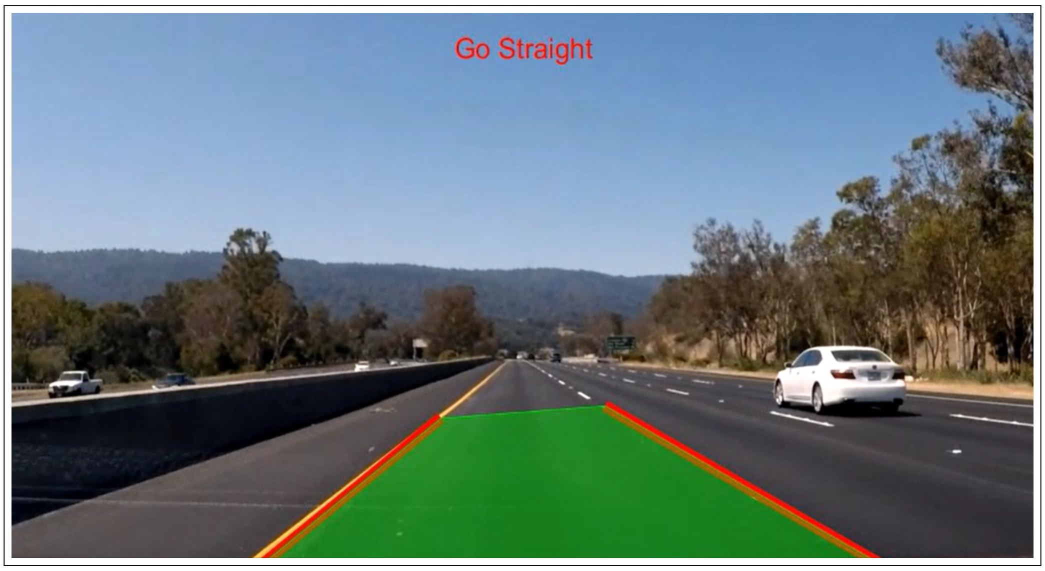

Figure 5.11: Final Image indicating to Go Straight

In fig, 5.11, the system effectively interprets the absence of lane deviation, providing a clear indication to "Go Straight." This insight is crucial for assisting the driver in maintaining the correct trajectory on the road. By comprehensively analyzing the road curvature and detecting deviations, the system ensures prompt and accurate guidance, contributing to enhanced driving safety and control.

Chapter 6

Conclusions and future scope

This chapter contains the conclusion and future scope for this project and it also covers the applications in social context that show us how Lane Detection can be employed in day-to-day living.

6.1 Conclusion

In conclusion, the adaptive cruise control project is a significant technological advancement in the automotive industry that aims to improve road safety and driving comfort. With the integration of sensors, radar, and machine learning algorithms, the system can automatically adjust the speed of the vehicle based on the surrounding traffic conditions. The technology has been proven effective in reducing accidents caused by human error, particularly rear-end collisions. Moreover, it can reduce driver fatigue and stress, making driving a more relaxing and enjoyable experience. As the automotive industry continues to evolve, we can expect adaptive cruise control to become more prevalent and sophisticated, leading to further improvements in road safety and driving experience.

6.2 Future scope

The future scope for Lane Detection projects is quite promising, with advancements in technology and increasing demand for safe and efficient driving. Here are some potential areas where lane detection projects can make significant contributions:

1. **Advanced Sensor Fusion:** Integration of data from multiple sensors, including cameras, LiDAR, and radar, for more accurate and robust lane detection in diverse driving conditions.

2. **Deep Learning and Neural Networks:** Continued exploration and application of deep learning models, such as convolutional neural networks (CNNs), to enhance the precision and reliability of lane detection algorithms.

3. **Real-Time Processing and Edge Computing:** Development of real-time, edge-optimized lane detection algorithms to enable faster and more efficient processing directly on autonomous vehicles' embedded systems.

4. **Adaptive and Predictive Algorithms:** Implementation of adaptive algorithms that dynamically adjust to changing road conditions and predictive algorithms capable of anticipating lane changes, improving proactive responses by autonomous vehicles.

5. **Human-Centric Interaction:** Integration of lane detection systems with human-centric interaction mechanisms, fostering effective communication between autonomous vehicles and human drivers, pedestrians, and cyclists for increased safety and awareness.

6.2.1 Application in the societal context

Lane detection in autonomous vehicles enhances road safety, reduces traffic congestion, and improves pedestrian and cyclist safety. It fosters accessible transportation for individuals with disabilities, promotes community livability, and contributes to environmental sustainability by optimizing traffic flow. This technology also facilitates crisis response, integrates with public transportation, and serves as a platform for community engagement and education. Here are a few social context applications for Lane Detection:

1. **Enhanced Road Safety:** Lane detection ensures vehicles stay within designated lanes, reducing the risk of accidents caused by unintentional lane departures.

2. **Reduced Traffic Congestion:** Autonomous vehicles optimize traffic flow by maintaining appropriate spacing and alignment within lanes, potentially reducing overall traffic congestion.

3. **Improved Pedestrian and Cyclist Safety:** Lane detection contributes to recognizing and respecting pedestrian crosswalks and bicycle lanes, enhancing safety for pedestrians and cyclists.

4. **Accessible Transportation:** Reliable lane detection in autonomous vehicles facilitates accessible transportation, benefiting individuals with disabilities or limited mobility.

5. **Public Transportation Integration:** Lane detection supports smoother integration with public transportation systems, allowing for seamless transitions and interactions at designated stops and transit lanes.

6. **Community Livability:** Autonomous vehicles with lane detection promote orderly traffic behavior, reducing traffic violations and contributing to a safer and more pleasant community space.

7. **Environmental Impact Reduction:** Efficient lane-following algorithms in autonomous vehicles minimize fuel consumption, reducing the environmental impact of transportation systems.

8. **Customized Urban Planning:** Lane detection data informs urban planners about traffic patterns, aiding in infrastructure development, road maintenance, and traffic management.

9. **Crisis Response and Emergency Services:** Autonomous vehicles with lane detection contribute to effective crisis response and emergency services by navigating traffic efficiently and safely.

10. **Community Engagement and Education:** Integration of autonomous vehicles with lane detection serves as a platform for community engagement and education on advanced transportation technologies.

Bibliography

[1] Angelos Amditis, Matthaios Bimpas, George Thomaidis, Manolis Tsogas, Mariana Netto, Saïd Mammar, Achim Beutner, Nikolaus Möhler, Tom Wirthgen, Stephan Zipser, Aria Etemad, Mauro Da Lio, and Renzo Cicilloni. A situation-adaptive lane-keeping support system: Overview of the safelane approach. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):617–629, 2010.

[2] Keong-Hun Choi and Jong-Eun Ha. An adaptive threshold for the canny algorithm with deep reinforcement learning. *IEEE Access*, 9:156846–156856, 2021.

[3] P.M. Daigavane and P.R. Bajaj. Road lane detection with improved canny edges using ant colony optimization. *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 76–80, 2010.

[4] Dagao Duan, Meng Xie, Qian Mo, Zhongming Han, and Yueliang Wan. An improved hough transform for line detection. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, volume 2, pages V2–354–V2–357, 2010.

[5] Dongjin Fan, Hui Bi, and Lidong Wang. Implementation of efficient line detection with oriented hough transform. In *2012 International Conference on Audio, Language and Image Processing*, pages 45–48, 2012.

[6] Qingquan Li, Long Chen, Ming Li, Shih-Lung Shaw, and Andreas Nüchter. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Transactions on Vehicular Technology*, 63(2):540–555, 2014.

[7] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. Line-cnn: End-to-end traffic line detection with line proposal unit. *IEEE Transactions on Intelligent Transportation Systems*, 21(1):248–258, 2020.

[8] Simranjit Singh and Rakesh Singh. Comparison of various edge detection techniques. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 393–396, 2015.

[9] Huifeng Wang, Yunfei Wang, Xiangmo Zhao, Guiping Wang, He Huang, and Jiajia Zhang. Lane detection of curving road for structural highway with straight-curve model on vision. *IEEE Transactions on Vehicular Technology*, 68(6):5321–5330, 2019.

qmini

ORIGINALITY REPORT

9%	9%	4%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.slideshare.net Internet Source	2%
2	www.coursehero.com Internet Source	1%
3	aaai.org Internet Source	1%
4	ebin.pub Internet Source	1%
5	www.researchgate.net Internet Source	<1%
6	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1%
7	naqvilaw.com Internet Source	<1%
8	assets.researchsquare.com Internet Source	<1%
9	dataintelo.com Internet Source	<1%

10	rogerscorp.com Internet Source	<1 %
11	catalog.uttyler.edu Internet Source	<1 %
12	imrfjournals.in Internet Source	<1 %
13	Michail Makridis, Konstantinos Mattas, Daniele Borio, Biagio Ciuffo. "Estimating empirically the response time of commercially available ACC controllers under urban and freeway conditions", 2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2019 Publication	<1 %
14	www.sae.org Internet Source	<1 %
15	Amel Toroman, Samir Vojić. "Adaptive car control system based on a predictive model", IOP Conference Series: Materials Science and Engineering, 2021 Publication	<1 %
16	www.scitepress.org Internet Source	<1 %
17	ir.uitm.edu.my Internet Source	<1 %

18	pure.tue.nl Internet Source	<1%
19	www.ijser.org Internet Source	<1%
20	www.thinkmind.org Internet Source	<1%
21	www.arxiv-vanity.com Internet Source	<1%
22	Nassaree Benalie, Worrawut Pananurak, Somphong Thanok, Manukid Parnichkun. "Improvement of adaptive cruise control system based on speed characteristics and time headway", 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009 Publication	<1%
23	Rosenfeld, Avi, Zevi Bareket, Claudia V. Goldman, David J. Leblanc, and Omer Tsimhoni. "Learning Driver's Behavior to Improve Adaptive Cruise Control", Journal of Intelligent Transportation Systems, 2014. Publication	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		