ROAD LANE DETECTION AND OBSTACLE TRACKING

A PROJECT REPORT

Submitted by

MAHJABEEN A

POOJA S

BHARATH M

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY :: CHENNAI 600 025

JUNE 2022

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**ROAD LANE DETECTION AND OBSTACLE TRACKING"** in the bonafide work of "**MAHJABEEN A (2019103542), POOJA S (2019103584)** and **BHARATH M (2019103011)"** who carried out the project work under my supervision.

PLACE : Chennai **SIGNATURE**

DATE : 04.06.2022

Mr. T. M. Thiyagu

SUPERVISOR

Computer Science and Engineering

Anna University, Kotturpuram,

Chennai-600025, Tamil Nadu,

College of Engineering Guindy

ABSTRACT

Lane detection in driving scenes is an important module for autonomous vehicles and advanced driver assistance systems. However, it is a challenging task to improve the robustness of lane detection due to a few environmental factors.

Lane detection is composed of pre-processing, Adaptive Region of Interest (AROI) setting, and lane marking detection and tracking. Kalman filter is employed to track road boundaries detected in the AROI using Progressive Probabilistic Hough Transform (PPHT).

K-means clustering is used to remove the noisy lines with unqualified angles. Once the lane is detected, Object tracking is done using the YOLO algorithm in order to avoid the obstacles on the lane.

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

PHT	-	Probabilistic Hough Transform
PPHT	-	Progressive Probabilistic Hough Transform
AROI	-	Adaptive Region of Interest
YOLO	-	You Only Look Once
Numpy	-	A library consisting of multidimensional array objects and a collection of routines for processing the arrays.
Pandas	-	Converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.
OS	-	It provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.
openCV	-	It is a library of programming functions mainly aimed at real-time computer vision.
sklearn	-	It is a free software machine learning library for the Python programming language.
Darknet	-	A darknet is an overlay network within the Internet that can only be accessed with specific software or authorization, and often uses a unique customised communication protocol.
F1 Score	-	It combines the precision and recall of a classifier into a single metric by taking the harmonic mean.
Precision	-	Precision refers to the number of true positives divided by the total number of positive predictions.
Recall	-	The recall is calculated as the ratio between the number of positive samples correctly classified as positive to the total number of positive samples.
Accuracy	-	It is the ratio of the number of correct predictions to the total number of input samples.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PROBLEM STATEMENT

Lane detection and tracking is an important component of the Advanced Driver Assistance System (ADAS). Lane line detection and identification has become a basic and necessary functional module in the field of vehicle safety and intelligent vehicle navigation, which can not only reduce the occurrence of traffic accidents, but also provide help for indepth research on intelligent traffic. However, effective lane detection is very challenging, due to the following factors: (i) types of roads, such as highways, urban roads, and unstructured roads; (ii) different lighting conditions, e.g., backlighting or low light condition; and (iii) occlusion caused by various obstacles, such as passing traffic, pedestrians, or shadows.

1.2 PROPOSED SOLUTION

We propose a passive type of vision-based lane detection and tracking, where different lighting conditions, and different road types, i.e., straight and curved, are considered. In addition to this, we intend to detect obstacles on the path of vehicles, so as to promote the safety of vehicle drivers better.

1.3 OBJECTIVE

The main objective is to develop a vision-based real-time lane detection and tracking using PPHT, where different lighting conditions, and different road types, i.e., straight and curved, are considered.

We also make use of the YOLO algorithm to detect obstacles on the road and provide a robust object tracking system along with road lane detection.

2. LITERATURE REVIEW

2.1 EXISTING WORK

Real time lane detection and tracking has been one of the most active fields for researchers in the last few years. Pre- processing is the first stage of lane detection that is employed to remove irrelevant noise, get accurate road lane information, and to ensure successful detection in the subsequent steps.

It includes image smoothing using conventional filters such as Gaussian filter [2], [4], [9], and Median filter [9]. In addition, image segmentation is performed where features such as edge and light intensity are used.

After the preprocessing step, the extraction of a Region of Interest (ROI) is performed in lane detection. It is a simple but effective method to reduce redundant image data quantity on the one hand. In fact, if the lane detection is done only in the ROI and not in the whole image, the effect of environmental noise can be reduced.

Previous research works select the ROI as the bottom side of the image, while others use the vanishing point detection technique [3] to define the ROI.

Once the ROI is confirmed, some lane detection approaches apply Warp

Mapping (IPM) [4], to get the bird's eye view of the road image based on the assumption of parallel lane boundaries.

Although the effective detection portion, such as the lane, can account for approximately two-thirds of the area of the image, it may still contain noise. For instance, if the vanishing points were incorrected, processing time may increase and might lead to incorrect ROI extraction. Therefore, these methods cannot meet the real-time requirements of the automated vehicle. In order to reduce the computational complexity, here, we propose to define a simple and adaptive ROI. This is achieved by using a horizon line, dividing the scene into road and sky regions, that is based on the gradient information obtained by the Otsu process.

Methods applied on ROI include two main steps: lane tracking and departure warning. Many methods have been proposed in literature for straight and curved lane detection and tracking. Hough Transform is the most common technique used for straight lane markings detection [2]–[4].

However, it faces serious challenges of computational complexity and high false positive rate [3], [4]. Other detection methods such as hyperbola and parabola are used and presented in [7].

The most widely used tracking methods are Kalman Filter [2] and Particle Filter. Generally, Kalman Filter is used after Hough Transform for tracking issues [2].

Once lane detection and tracking are completed, lane departure warning system alerts the driver for unintended lane departure. However, in

most proposed methods, lane departure can be determined knowing lateral offset, vanishing point position and the optical centre of the camera.

The lateral offset between the primary vehicle's location and its position in various images has to be tracked in real-time.

The vanishing point position is the most common solution used to determine lane departure. In order to find the vehicle's horizontal displacement, the vanishing point position needs to be compared with those of previous frames, but this process is time consuming.

Finally, lane departure is determined based on the centre of a road lane and the location of the optical centre of the camera.

In order to deal with the above-mentioned limitations, in this paper, an adaptive ROI is proposed that uses a horizon line to effectively reduce the computational complexity and remove noisy segments. Moreover, a modified Hough Transform technique (PPHT) is used to enhance memory efficiency and to improve the computational time [3]. In addition, an adaptive threshold is employed for binary images segmentation.

2.2 LIMITATIONS OF EXISTING WORK

The above papers describe various methods for road lane detection. Among these methods, a common issue was facing difficulties in identifying road lanes in diverse situations (when the road surface is not straight). Moreover, there is a higher rate of false recognition amongst certain methods.

2.3 SOLUTION PROPOSED

The proposed approach can stably detect the lanes in different conditions. Compared with baseline architectures which use one single image as input, the proposed architecture can achieve better results because of using multiple continuous frames as input.

The Progressive Probabilistic Hough Transform (PPHT) can avoid false recognitions, and hence performs better than the other methods.

In addition to road lane detection, YOLO algorithm for object detection is added in order to detect obstacles on the road and to avoid accidents. This algorithm is used for its processing speed and good performance.

3. SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

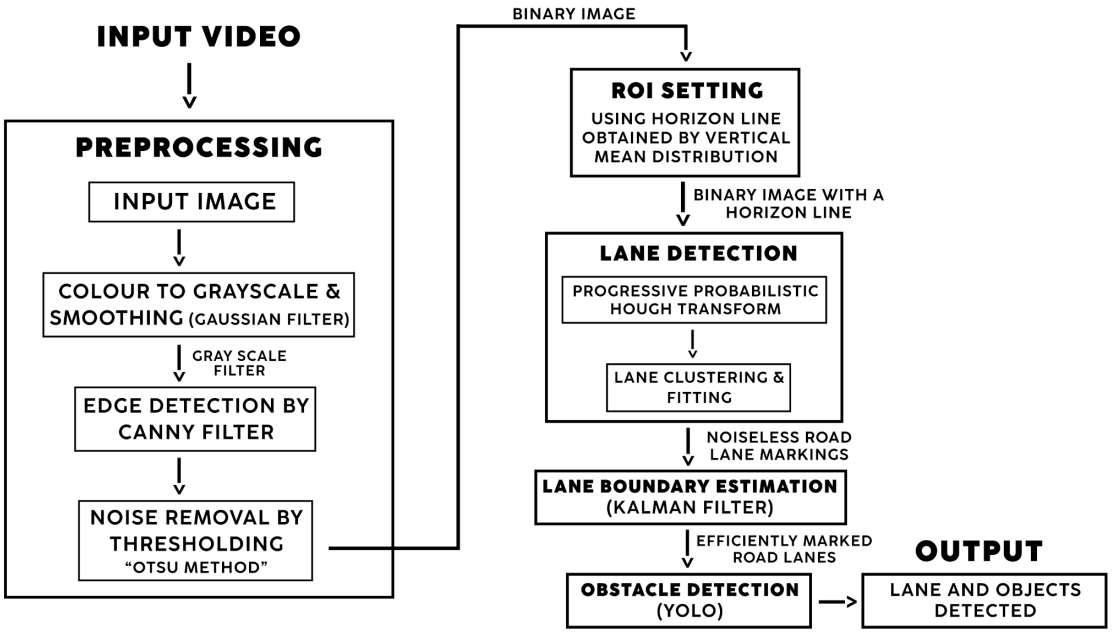


Figure 1 : System Architecture

3.2 SYSTEM REQUIREMENTS

3.2.1 SOFTWARE REQUIREMENTS

- Python 3.x
- Google Drive Storage (10 Gb)
- Python Libraries
    - OpenCV
    - Sklearn
    - Numpy
    - Pandas
    - Matplotlib

3.2.2 HARDWARE REQUIREMENTS

- Intel i3/i5 or Ryzen 3 1200x with minimum 2.4GHZ or equivalent 8GB RAM

4. DETAILED ARCHITECTURE

4.1 LIST OF MODULES

- Image preprocessing
- Region of interest setting
- Lane detection
- Lane boundary estimation
- Obstacle detection

4.1.1 IMAGE PREPROCESSING

In this module, we deal with the preprocessing of the input image so that we get it in our desired form. The output of this module will help us fetch efficient results in this project.

First, we convert the colour of the input image from RGB to grayscale. A grayscale image requires less available space and is faster, especially when we deal with complex computations. Following this, we employ the use of the Canny filter to perform edge detection. Canny filter is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.

After edge detection, Otsu thresholding is carried out to perform automatic image thresholding. In the simplest form, the algorithm returns a single intensity threshold that separates pixels into two classes, foreground and background.

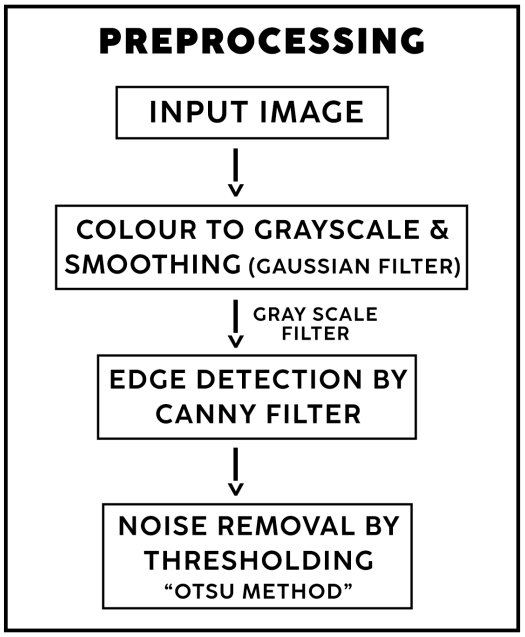


Figure 2 : Pre-processing Module

Input : Road Lane videos

Output : Binary Image

ALGORITHM -

Conversion from RGB to Grayscale :

gray_img = cv.cvtColor(img, cv.COLOR_RGB2GRAY)

blur = cv.GaussianBlur(gray_img, (5, 5), 0)

Edge detection :

edge_det = cv.Canny(gray_img, 50, 100, apertureSize=3)

Otsu thresholding :

ret, thresh1 = cv.threshold(edge, 120, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)

4.1.2 ROI SETTING

In this module, we deal with setting the region of interest for the output of the previous module. ROI (Region of interest) allows us to focus on a selected region only. The remaining part of the image, outside the region of interest is ignored as it is unnecessary in our attempt to detect lanes.
To adjust the size of the ROI, we change the parameter 'Width' and 'Height' to the desired size. ROI can ensure important information will not be lost. Moreover, it can effectively compress image data thereby solving the conflict between compression ratio and image quality.

Figure 3 : ROI Setting Module

Input : Binary Image

Output : Binary Image with a horizon line

ALGORITHM -

Setting the region of interest in our image :

region_of_interest_vertices = [(700, height), (width/2, height/1.37), (width-400, height)]

vertices = np.array([region_of_interest_vertices], np.int32)

mask = np.zeros_like(edge)

match_mask_color = (255)

cv.fillPoly(mask, vertices, match_mask_color)

masked_image = cv.bitwise_and(edge, mask)

4.1.3 LANE DETECTION

In this module, we deal with the detection of lanes present in our input images. For this, we make use of the progressive probabilistic Hough transform (PPHT) algorithm. Unlike the standard Hough transform, the PPHT minimises the amount of computation needed to detect lines by exploiting the difference in the fraction of votes needed to reliably detect lines with different numbers of supporting points.

PPHT does not take into account pre-defined angle thresholds along with the number of voting pixels in the accumulator. This can lead to unwanted lines for various reasons such as insufficient amount of marking lines, blurred images and passing vehicles. These noisy lines with unqualified angles can be removed using K-means clustering technique.

Input : Binary Image with a horizon line

Output : Noiseless Road lane markings

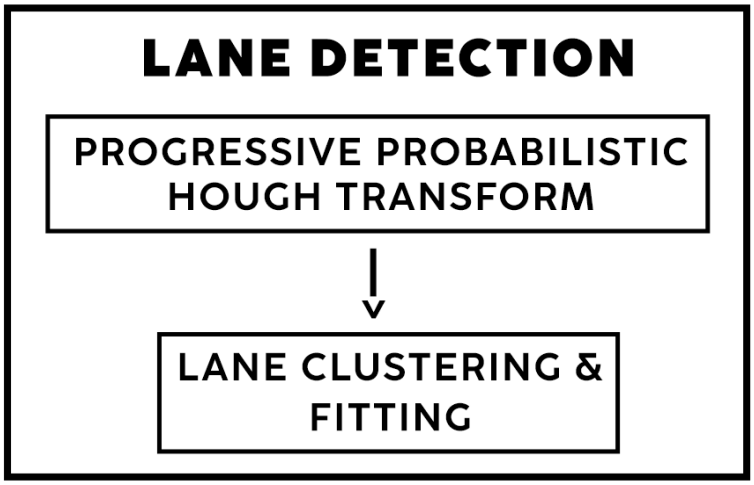


Figure 4 : Lane Detection Module

ALGORITHM -

PPHT algorithm :

PHT :

lines = cv.HoughLinesP(cropped_image, rho=2, theta=np.pi/180, threshold=50, lines=np.array([]), minLineLength=10, maxLineGap=30)

K-Means clustering :

n_clusters=4

kmeans = KMeans(n_clusters=n_clusters, n_init = 4)

kmeans.fit(lines)

*centroids = kmeans.cluster_centers_*

lines = scaler.inverse_transform(lines)

centroids = scaler.inverse_transform(centroids)

4.1.4 LANE BOUNDARY ESTIMATION

In this module, we deal with the estimation of the lane boundaries. The Kalman filter algorithm is used to track the lane boundaries in the given frame.

The algorithm works by a two-phase process. For the prediction phase, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (containing some error) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with greater certainty.

Figure 5 : Lane Boundary Estimation Module

Input : Extracted road lane markings
Output : Efficiently marked road lanes

ALGORITHM -

Kalman filter algorithm :

cv.KalmanFilter(self.state_size, self.meas_size, self.contr_size)
self.kf.transitionMatrix = np.eye(self.state_size, dtype=np.float32)
self.kf.measurementMatrix = np.zeros((self.meas_size, self.state_size), np.float32)
lt = LaneTracker(2, 0.1, 15)

4.1.5 OBSTACLE DETECTION

In this module, we make use of the YOLO algorithm to perform the detection of obstacles found in the given frame. This is an algorithm that detects and recognizes various objects in a picture (in real-time). YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. The algorithm requires only a single forward propagation through a neural network to detect objects.

Figure 6 : Obstacle Detection Module

Input : Efficiently marked road lanes

Output : Obstacles detected

ALGORITHM -

YOLO algorithm :

Divide the input images in various grids

Perform image classification

Predict the class probability of each vehicle present in the image

Detecting objects in video :

!./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights -dont_show /content/drive/MyDrive/video_2.mp4 -i 0 -out_filename /content/drive/MyDrive/video_out2.avi

5. IMPLEMENTATION DETAILS

5.1 IMAGE PREPROCESSING

INPUT

The following video is provided as input to the pre-processing module.

Figure 7 : Frame from input video

OUTPUT

Figure 8 : Grey scale image

Figure 9 : Canny Edge Detection

Figure 10 : Otsu Thresholding

5.2 ROI SETTING

INPUT

Pre-processed video frame is given as an input.

OUTPUT

ROI is calculated and its output is displayed.

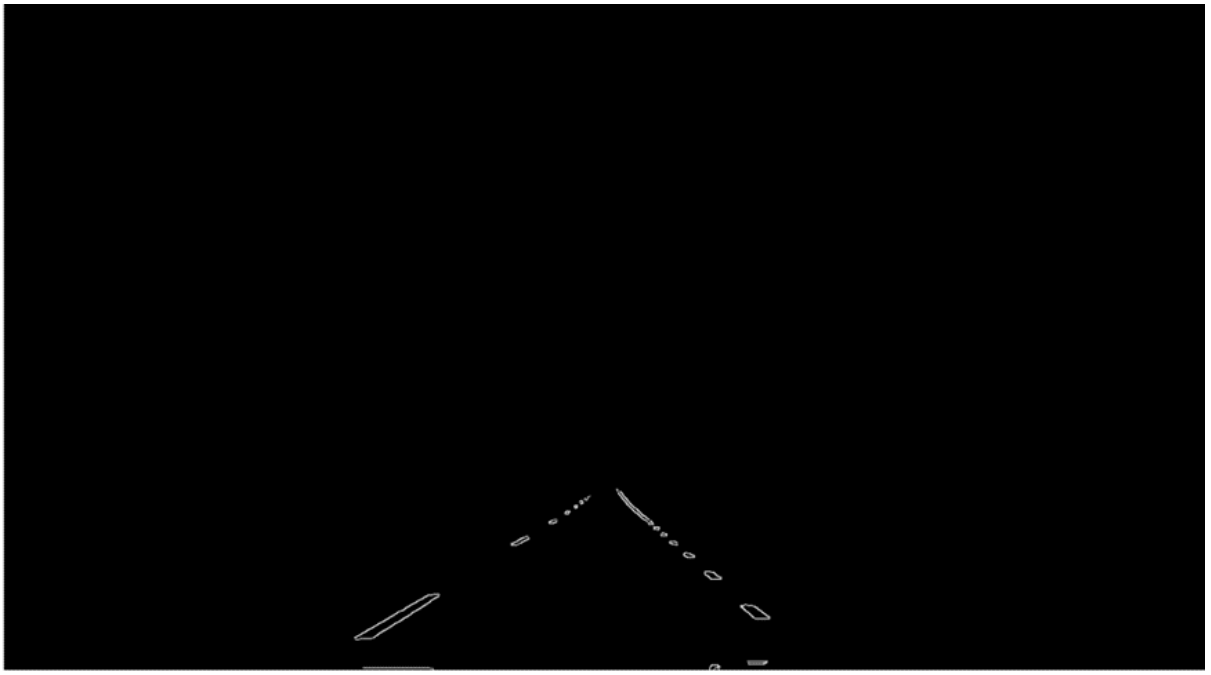


Figure 11 : ROI Output

5.3 LANE DETECTION

INPUT

Binary image with ROI set is given as input.

OUTPUT

Lane detection is done using Probabilistic Hough Transform(PHT). Its output is then passed to k-means classifier which clusters the points.

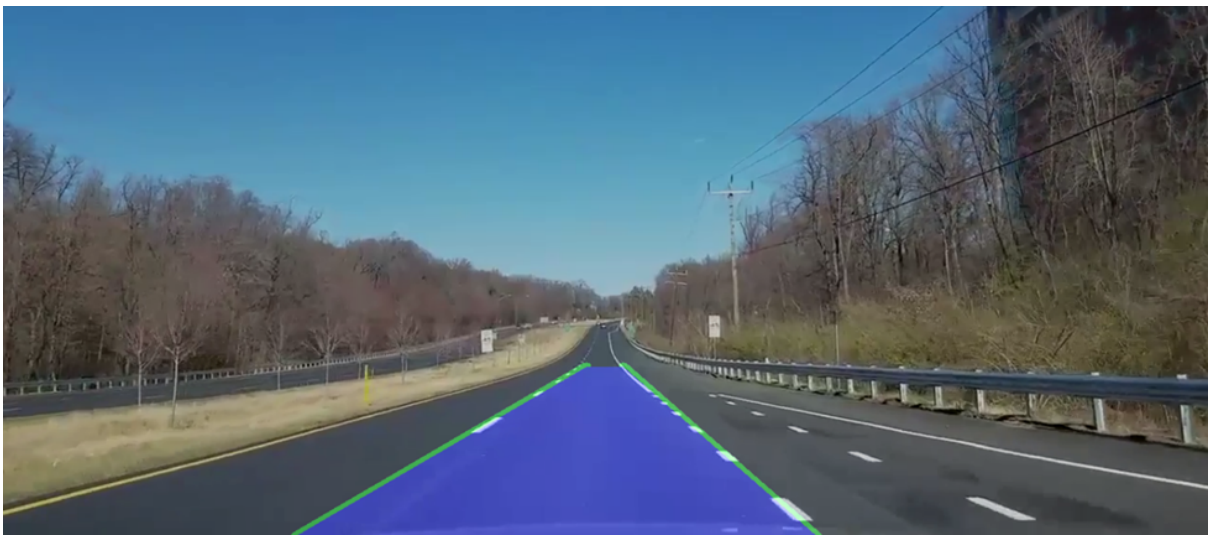


Figure 12 : Lane Detection using PHT

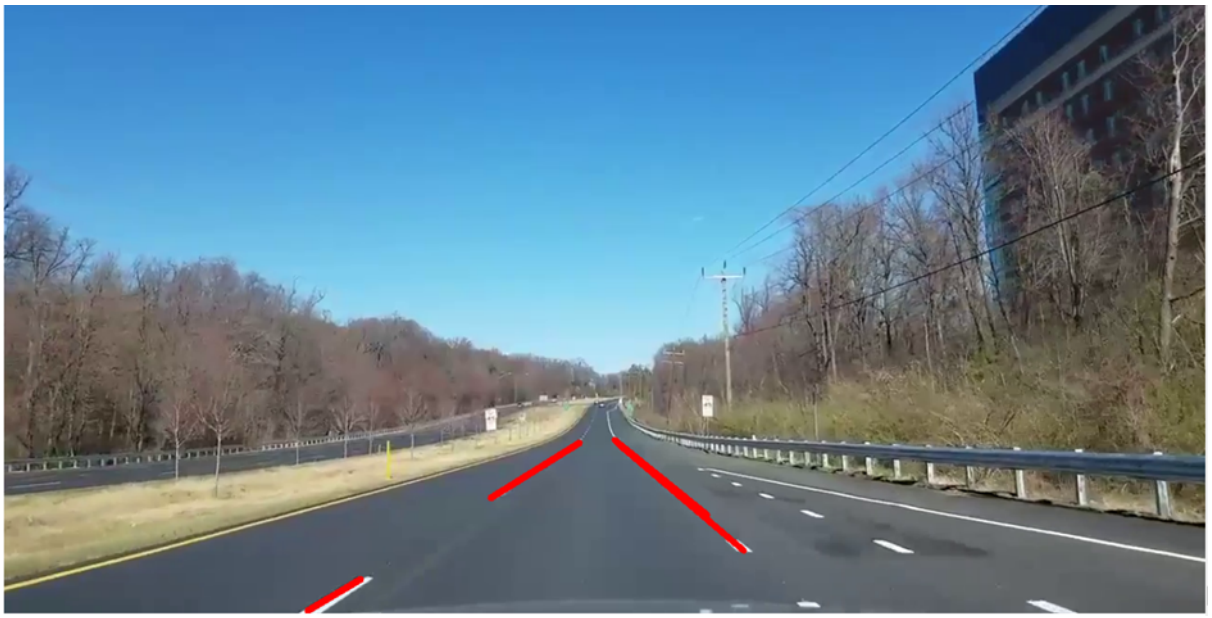


Figure 13 : K-means clustering on PHT output

5.4 LANE BOUNDARY ESTIMATION

KALMAN FILTER

INPUT

Output of the lane detection module is given as input.

OUTPUT

Lane Tracking using Kalman filter is done.

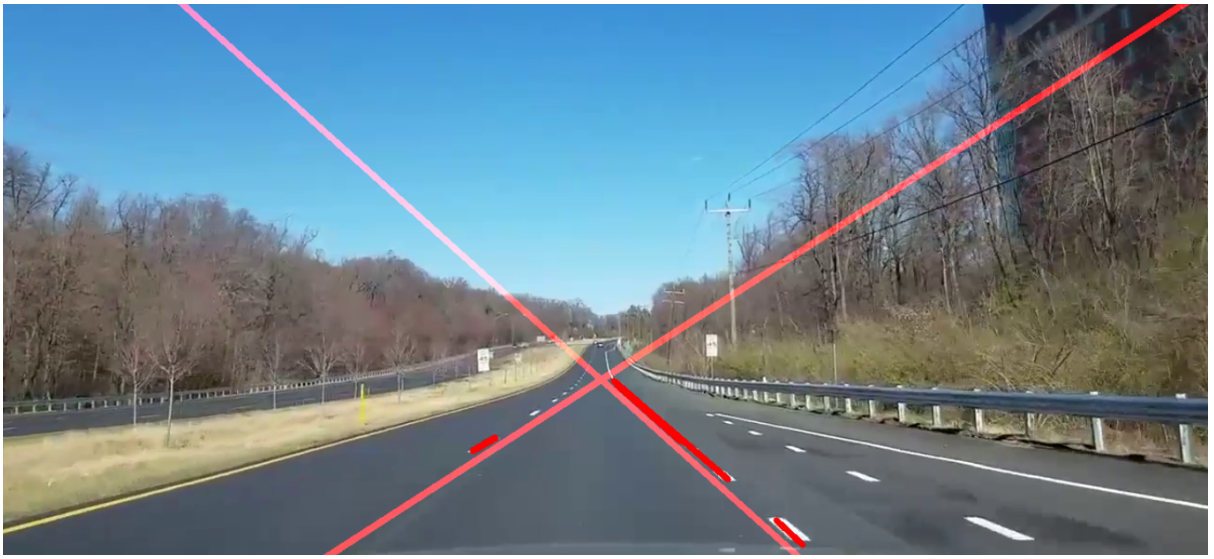


Figure 14 : Lane Tracking using Kalman filter

5.5 OBSTACLE DETECTION

INPUT

Image obtained after lane detection is given as inputs.

OUTPUT

The Yolov4 pre-trained model is used to detect objects.

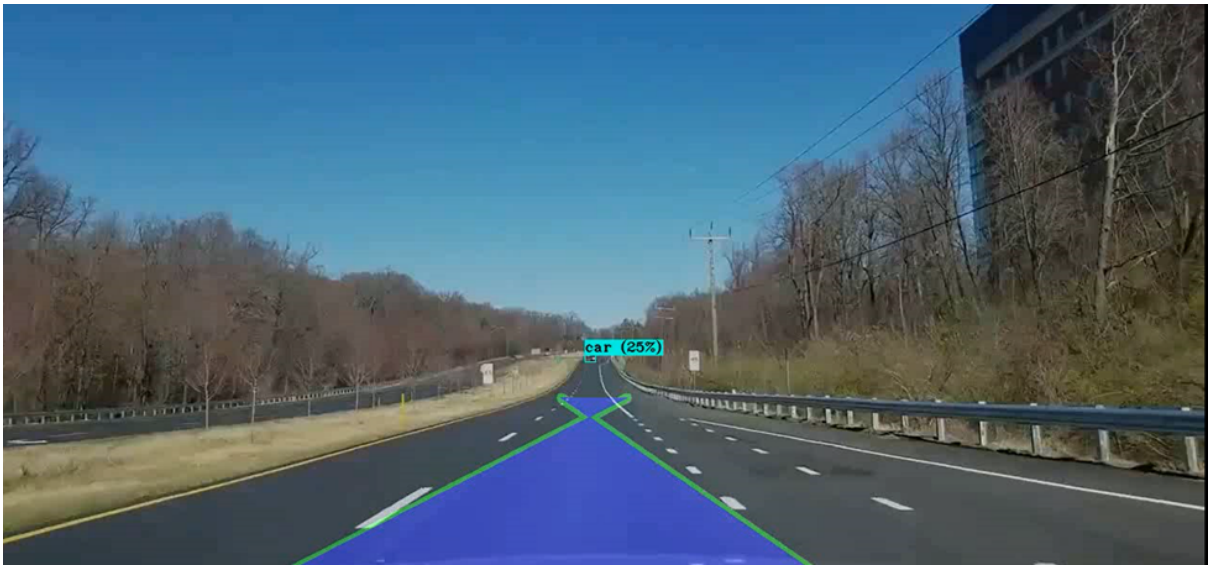


Figure 15 : Obstacle detection using YOLOv4

6. PERFORMANCE METRICS

6.1 ACCURACY

Here accuracy indicates the fraction of predictions that the model made were correct. It indicates how efficiently lane lines are detected.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

6.2 PRECISION

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (True Positives and False Positive).

$$Precision = \frac{TP}{TP + FP}$$

6.3 RECALL

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect positive samples.

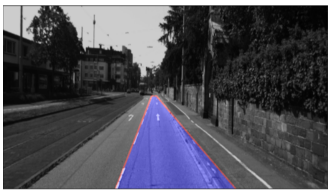
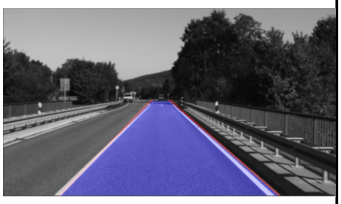
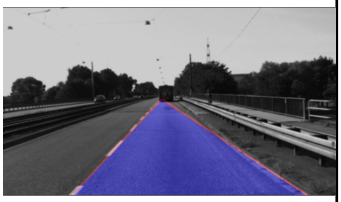
$$Recall = \frac{TP}{TP + FN}$$

6.4 F1-SCORE

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

6.5 METRICS FOR OUR MODEL

Frame No.	Input Frame	Lane detected Frame	Binary output	Metrics
1				Precision : 0.806 Recall : 0.806 Accuracy : 0.963 F1-score : 0.806
2				Precision : 0.736 Recall : 0.736 Accuracy : 0.911 F1-score : 0.736
3				Precision : 0.776 Recall : 0.776 Accuracy : 0.916 F1-score : 0.776
4				Precision : 0.757 Recall : 0.757 Accuracy : 0.912 F1-score : 0.757

OVERALL METRICS

Precision - 0.769

Recall - 0.769

FMeasure - 0.769

Accuracy - 0.925

7. CONCLUSION

Among the various methods implemented for lane detection , a common issue was facing difficulties in identifying road lanes in diverse situations (when the road surface is not straight).
The proposed approach can stably detect the lanes in different conditions. The Progressive Probabilistic Hough Transform (PPHT) can avoid false recognitions, and hence performs better than the other methods. It also reduces computational complexity.
In addition to road lane detection, YOLO algorithm for object detection is added in order to detect obstacles on the road and to avoid accidents. This algorithm is used for its processing speed and good performance.

7.1 FUTURE SCOPE

- Our model is limited to detect only white lanes and can be expanded to detect yellow lanes.
- Although our system works properly on different conditions, there

are still some problems to solve especially incase of shadows present in the road surface. In the future work, we will combine more models of the road to improve the robustness of the algorithm.

- The model can be configured to take real time input by adding the required hardwares and softwares.

8. REFERENCES

[1] Mehrez Marzougui, Areej Alasiry, Yassin Kortli and Jamel Baili, “A Lane Tracking Method Based On Progressive Probabilistic Hough Transform”, 2020.

[2] A. Mammeri, A. Boukerche, and G. Lu, ‘‘Lane detection and tracking system based on the MSER algorithm, Hough transform and Kalman filter,’’ in Proc. 17th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst. (MSWiM), New York, NY, USA, 2014, pp. 259–266.

[3] Y. Kortli, M. Marzougui, B. Bouallegue, J. S. C. Bose, P. Rodrigues, and M. Atri, ‘‘A novel illumination-invariant lane detection system,’’ in Proc. 2nd Int. Conf. Anti-Cyber Crimes (ICACC). Abha, Saudi Arabia: King Khalid University, Mar. 2017, pp. 166–171.

[4] M. Aly, ‘‘Real time detection of lane markers in urban streets,’’ in Proc. IEEE Intell. Vehicles Symp. Eindhoven, The Netherlands: Eindhoven University of Technology, Jun. 2008, pp. 7–12.

[5] Sun, Ziqiang , “Vision Based Lane Detection for Self-Driving Car,” in IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), 2020.

[6] Hiroyuki Komori, Kazunori Onoguchi “Lane Detection based on Object Detection and Image-to-image Translation” in 25th International Conference on Pattern Recognition (ICPR), 2020.

[7] Z. Nan, P. Wei, L. Xu, and N. Zheng, ‘‘Efficient lane boundary detection with spatial-temporal knowledge filtering,’’ Sensors, vol. 16, no. 8, p. 1276, 2016.

[8] R. N. Hota, S. Syed, S. Bandyopadhyay, and P. Krishna, “A Simple and Efficient Lane Detection using Clustering and Weighted Regression”

[9] M. B. de Paula and C. R. Jung, "Automatic detection and classification of road lane markings using onboard vehicular cameras", IEEE Trans. Intell. Transp. Syst., vol. 16, no. 6, pp. 3160-3169, Dec. 2015.