GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

DETECTION OF ROAD LANE FOR AUTOMATED DRIVING SYSTEM

THE Final Report of Capstone Project -2

Submitted by

ANAND SHANKAR

(1613107006)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

MS.SWATI SINGH

ASSISTANT Professor

APRIL / MAY- 2020

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **" DETECTION OF ROAD LANE FOR AUTOMATED DRIVING SYSTEM" is the bonafide work of "ANAND SHANKAR (1613107006)" who carried out the project work under my supervision.**

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,

PhD (Management), PhD (CS)

Professor & Dean

School of Computer Science & Engineering

SIGNATURE OF SUPERVISOR

MS. Swati Singh, M.Tech

Assistant Professor

School of Computing Science & Engineering

Abstract

For autonomous vehicle driving technology to move forward from the testing phase to actual self-driving car, safety measures and error minimization plays a key role. This project explains the techniques to outline the road lanes through the lane detection method. The width of the road lanes can be explicitly calculated to define the relative position of the vehicle in the defined lane. Inbuilt camera sensor produces lots of information from the surrounding which are processed through a machine vision system. The advanced system predicts trajectories collected during human handling of the vehicle and employs these to produce automatic tags for training a semantic-based pathway prediction model. Besides, a camera's actual inclination angle and the lane width can be obtained by active normalization. This approach is used to find the lane and width of the road from both sides effectively when there is a hindrance on one side. The drivable route knowledge is necessary particularly in unorganized situations and is crucial for an intelligent transport system to get reliable driving choice.

Table of Contents

LIST OF FIGURES:

Chapter 1.

1.Introduction

Using the machine learning algorithms and python it will predict the width of the road .It will also tell the distance from the roadside and also guide for overtaking the vehicles. The trained machine become the powerful feature in the driverless car as the vehicle get proper assistance in the overtaking and parking also .It can also be used in the cruise control of the car make the more impact of the car and easy in controlling the car. Active safety is currently a key topic in the automotive industry, which fosters the development of Autonomous Vehicle functions. Various advanced driver assistance systems (ADAS) and active safety systems have the potential to improve road safety, driver comfort, fuel economy, and traffic flow by assisting the driver during different driving conditions. It is estimated that human error is a contributing factor in more than 90% of all accidents. In order to save the human lives caused by road accidents, it is hence of interest to develop such systems using modeling and simulation 7tools which is quick and more efficient as compared to the real driving testing. Overtaking is one of the most complex maneuvers with the high risk of collision (75% human error) so the automation of this maneuver still remains one of the toughest challenges in the development of autonomous vehicles. Since overtaking is one of the complex maneuvers and so many factors affect it, the automation of this maneuver has been considered to be one of the toughest challenges in the development of autonomous vehicles. Overtaking involves a great interaction between both longitudinal (throttle and brake) and lateral (steering) actuators. Nowadays, in the field of driver assistance systems and automated driving, development approaches for lateral maneuver control are the very big challenge.

Computerized reasoning in autos has highlighted in numerous exploration tasks and trials have been led since the 1980s when first models for self-ruling autos were displayed via Carnegie Mellon College's Navlab. From that point forward, there have been a lot of innovative headways in the field of self-governing vehicles and Navlab 11, the most recent auto by CMU's Route Research facility, is a 2000 Jeep Wrangler introduced with several pieces of equipment such as GPS, magnetometers, proximity laser censors, and omni-directional camera . Stanley, an autonomous car created by Stanford University in cooperation with Volkswagon won first driver-less car racing challenge known as 'DARPA Grand Challenge' in 2005.

Numerous studies have been led about vehicle robotization and Driver Assistance Systems. A portion of the highlights of a self-governing vehicle incorporates Automatic Cruise Control, Automatic Parking, Collision Avoidance as well as Lane Departure Warning systems. Aside from the said, the potential advantages of autonomous cars incorporate decreased framework costs, expanded security, expanded consumer satisfaction, and a critical lessening in car accidents. . Some in favor of autonomous vehicles additionally trust that conveying robotics to the car will kill more typical wrongdoings like insurance scams and vehicle burglary.

On account of above mentioned benefits, many countries have taken a step forward to bring self-driving cars on the roads for public. UK, in 2014, announced that driver less cars2 will be allowed on public roads and soon a prototype called 'Lutz pod' was launched as UK's first driver-less car [6] [7]. Moreover, In 2017, first autonomous vehicle was demonstrated at Christchurch Airport [8]. However, despite significant amount of benefits, there exist some foreseeable challenges in completely accepting self-driven vehicles. A few opposer believe that widespread adoption of driver-less cars will bring dearth of driving related jobs and will also compromise with the passenger's safety . It is also believed that an autonomous car will likely lead to the loss of privacy and increased risks of hacking attacks and terrorism.

It is understood that the discrepancy between people's beliefs of the necessary government intervention may cause a delay in accepting autonomous cars on the road . This will keep the drivers in control of the

car for some years to come. Therefore, it is important to seek solutions to make their driving experience safer and easier. This thesis presents one such solution - Real time road lane lines detection in different weather conditions. A real time vision-based lane detection system can be used to assist the driver in locating the lanes or warning the driver if the vehicle goes out of lane. In this thesis, a Lane Detection algorithm is presented for real-time detection of road lane lines in various climate conditions. The algorithm runs on a 7 inch display attached to a RaspberryPi 3.0 computer with a Pi camera installed. The entire system is placed on the car's dashboard. The Pi Camera is placed inside the car fixed to the windshield to capture the real time video while driving. The display screen and the RaspberryPi computer is powered by a 12V-to-5V car power supply port. Lane Detection algorithm presented in this thesis implements the concepts of Computer Vision - like gradient change and probabilistic Hough transform, to detect the lane lines in the road images. Moreover, certain filtering techniques, such as filtering based on slope and intercept values, are used to determine the exact location of road lane lines. The algorithm is implemented in Python 2.7 with OpenCV 3.0.

The entire thesis is organized as follows. In Chapter 2, other work related to the lane detection or autonomous driving is discussed. Chapter 3 contains the details of the algorithm used to detect road lane lines in this research. Chapter 4 presents the statistics of the results from the experiments conducted during this research. Chapter 5 presents the conclusion and talks about the future work.

CHAPTER 2.

2.RELATED WORK

Enhancing autonomous driving, especially by performing accurate road lane line detection, has been a major research interest in past few years. Several researchers have performed real time vision based lane detection using various techniques. This chapter discusses few of those research works and related algorithms in brief. Yuan et al established a novel method for tracking road lanes for vision-guided autonomous vehicle navigation. They use an inverse perspective mapping to remove the perspective from the camera and then detect the edges of the road lanes from the inverse perspective mapping images. An algorithm for 'particle filtering' is used to compute the likelihood between all the particles with the edge images, henceforth estimating the three parameters of the real state of road lane lines. This lane detection method is tested in real

road images to achieve reliable results.

P. Mandlik and A.B. Deshmukh presented a Lane Departure Detection System(LDWS) in accordance with Advanced Diver Assistance System(ADAS) to warn the driver when the vehicle tends to depart it's lanes. LDWS is based on the lane identification and tracking algorithm and uses OpenCV implementations of 'Hough Transform' to detect vehicle lane departure on a Raspberry Pi. The experiments are conducted on the images captured using a toy vehicle with a USB camera, 'Intex IT-305WC webcam', mounted on top. Out of many straight lines detected by Hough Transform, the longest straight lines are identified as the lane lines. The results are collected using Intel Core i3 1.80 Ghz processor.

K. H. Lim et al. constituted a real-time implementation of lane detection and tracking system to localize lane boundaries and estimate a linear-parabolic lane model. For experiments, a CCD camera is used to capture video frames and stored in video port buffer of a TMS320DM642 DSP board. Horizon localization is used to discern the sky from road5 in the input image. To recognize lane markings, road pixels are removed from

the road region by performing lane analysis. Once lane boundaries are located, the conceivable edge pixels are scanned to ceaselessly to obtain the lane model. A Linear-parabolic model is used to construct the geometry of the lane and the model parameters are updated with Kalman filtering. X. Du et al. proposed a robust vision-based methodology to deal with challenges during lane detection like shadows, shifting lighting conditions, faded-away lane lines, etc.

The methodology incorporates four key advances: Using a ridge detector, the line pixels are pooled. Then, using a noise filtering mechanism, noisy pixels are removed. After removing noises, a sequential Random Sample consensus is employed to ensure that each lane line in the image is collected correctly. In the final step, a technique parallelism reinforcement is employed to enhance the accuracy of the model. The model is also fit to localize vehicles with respect to the road lane lines.

Q. Truong and B.R. Lee used the principal approach to detect road boundaries and lanes using a vision-based system in the vehicle. The paper presented a methodology to detect and estimate the curvature of lane boundaries. A vector-lane-concept and nonuniform B-spline (NUBS) interpolation method is used to construct the boundaries of road lane lines. Based on the lane boundary, the curvature of left and right lane boundaries are calculated. For experimental purposes, images are captured using a monocular camera. Experimental results are based on real world road images, as presented in the paper.

CHAPTER 3.

3. LANE DETECTION ALGORITHM

3.1 Overview

This section describes in detail the algorithm designed for the real time road lane lines detection. The algorithm is divided into five stages - data collection, data preprocessing, gradient change detection using Sobel, line detection using Hough, and line filtering using slope and intercept values. Subsequent sections describe each stage in detail.

The hardware used for this research is shown in the Fig. 1. A wooden board is used to hold the touchscreen display and the Raspberry Pi computer together. This wooden board is placed on the vehicle's dashboard. A Pi camera, attached to the Raspberry Pi.

Figure 1.A RaspBerryPi Touchscreen

3.2 Data Collection

The data set is collected over multiple drives during varying climate conditions. A Pi camera is used to capture 10 frames per second over an interval of 30 minutes (i.e. 1800 seconds). The camera settings for capturing the data is given in the algorithm section below**.**

For experiments, the collected videos are saved to the local Pi in the memory space8 provided by the attached memory card. The videos collected in h264 format are converted into mp4 and individual frames are extracted from mp4 videos and are converted to jpg images.

The results from the above algorithm are stored in a directory on the local machine and each file is read individually to be processed and saved in the output directory.

3.3 Data Preprocessing

The images collected by extracting frames using the above algorithm are iterated over individually and re-sized. A resided original image can be seen in Fig. 2.

dim = (400*,* 255)

resized = *cv*2.*resize*(*input image, dim, interpolation* = *cv*2.*INT ER AREA*)

The re-sizing is performed using *cv*2.*resize*() method of OpenCV. The resided image then undergoes masking, gray-scaling and Gaussian blurring**.**

3.3.1 Masking

It is understood that in any image containing road (or lane lines), the road surface area is present in the bottom half of the image. Using this knowledge, the region of interest is decided.

Figure 2: Resized original image

The entire y section(vertical height) of the image is horizontally cut into the half and the region lying above that horizontal line is discarded. This was performed using cv2.fillPolly() and cv2.bitwise and() method of OpenCV. This method takes a row number as an input and creates a mask such that the region of interest is only below the given row.

3.3.2 Grayscaling

In the future stages of the presented algorithm, edge detection and Hough transformation, the image is required to be converted into a single color scale, also called as grayscale.

Therefore, the region of interest extracted in the previous stage during masking, is rendered to grayscale image as shown below.

This is performed using cv2.cvtColor() method of OpenCV. This method takes a colored image as an input (RGB) and returns a grayscaled image as an output. Further processing is done on the resulting grayscaled image.

Figure 3: Image after gray scaling

3.3.3 Gaussian Blurring

Once the grayscaling is done, noise reduction is performed on the image. A gaussian kernel is used to blur/smoothen the image. It is done with the help of an OpenCv function, cv2.GaussianBlur(). The width and height of the kernel are required to be defined which should be positive and odd(both are defined as 5 in our case). We should also specify the

standard deviation in X and Y direction, sigmaX and sigmaY respectively (defined as 0 in our algorithm).

This method takes a grayscaled image as an input and returns a blurred image as an output depending on the kernel size. Further processing is done on the resulting grayscale blur(smooth) image. Gaussian blurring is illustrated in Fig. 4.

Figure 4: Image after Gaussian Blurring

3.4 Detecting Gradient Change - Edge detection using Canny Edge Detection in OpenCV

OpenCV puts all the above in single function, cv2.Canny(). We will see how to use it. First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Third argument is aperture size. It is the size of Sobel kernel used for find image gradients. By default, it is 3. Last argument is L2gradient which specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate, otherwise it uses this function: $Edge\_Gradient\ (G) = |G_x| + |G_y|$. By default, it is False.

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

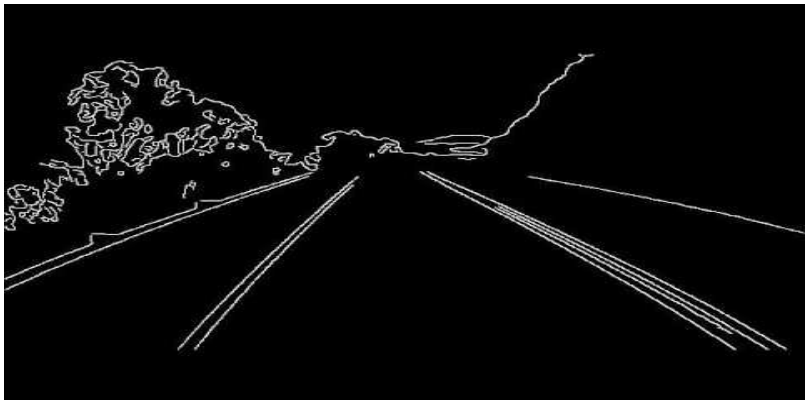

Figure 5: Image with detected edge pixels

3.5 Computing Hough lines based on the detected edges

Hough Transform takes a linear line equation and represents it in the parametric form. Any shape can be detected if it is represented in the mathematical form. A linear line equation is given as

$$y = mx + c$$

While in parametric form, a line is represented as -

$$\vartheta = x\cos\vartheta + y\sin\vartheta$$

where ρ is the perpendicular distance from origin to the line, and θ is the angle formed by this perpendicular line and horizontal axis. Any line can be represented in the form of ρ and θ. The motivation behind this method is to discover the instances belonging to a line shape by a voting procedure. This voting method is performed in a parameter space. [26]

In the given algorithm, this is performed by using an OpenCV method cv2.HoughLinesP().

HoughLines P stands for Probabilistic Hough Transform, which is an optimization of Hough Transform discussed above. In addition to minimum number of votes it also takes two more parameters minimum length of line and maximum allowed gap.

Hough parameters used in the presented algorithm are defined below:

$\rho = 1$

$\theta = pi/180$

threshold = 50

min line len = 60 pixels

max line gap = 400 pixels

Where, threshold is the minimum number of votes required to consider the given edge as a line, minLineLength is the minimum length of line (line segments shorter than this are rejected) and maxLineGap is the maximum allowed gap between line segments to treat them as single line.

An image with detected Hough lines is shown in Fig. 6.

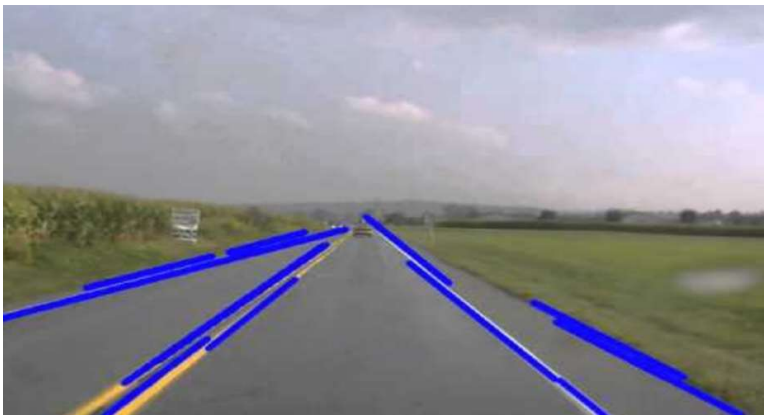

Figure 6: Image with detected lines using Probabilistic Hough Transform

3.6 Filtering resulting lines

As shown in figure .6, multiple lines detected by Probabilistic Hough Transform do not belong to road lane lines. To discard the unwanted lines, two filtering methods are used - filtering using slope value and filtering using intercept value.

The limitation of filtering lane lines based on the slope is that the filtering method considers the parallel lines(lines with the same slope values) as the potential lane lines.

Therefore, another parameter is taken into account to determine the exact lane line position. This parameter is the y intercept value - c in

$y = mx + c.$

To remove the false positives (non lane lines parallel to the lane lines), the y-intercept value for each resulting line is calculated.

The equation of a line is:

$y = mx + c$

Hence, knowing the value of sloe m, the value of c can be found by -

$c = y - mx$

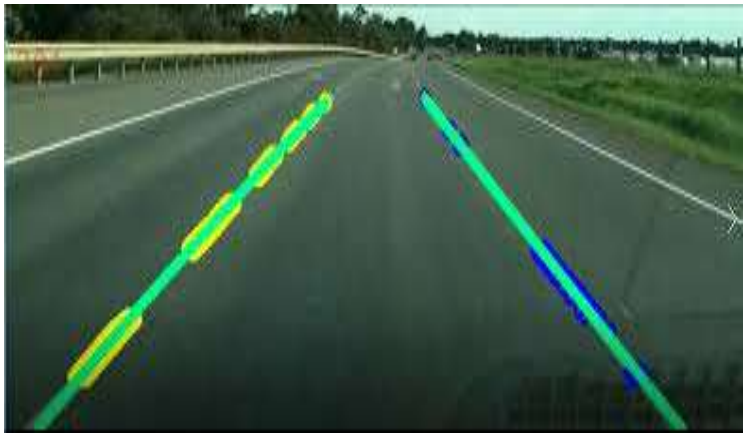

Figure 7. Image after filtering based on slope and intercept values

CHAPTER 4.

4. Block Diagram

: road.

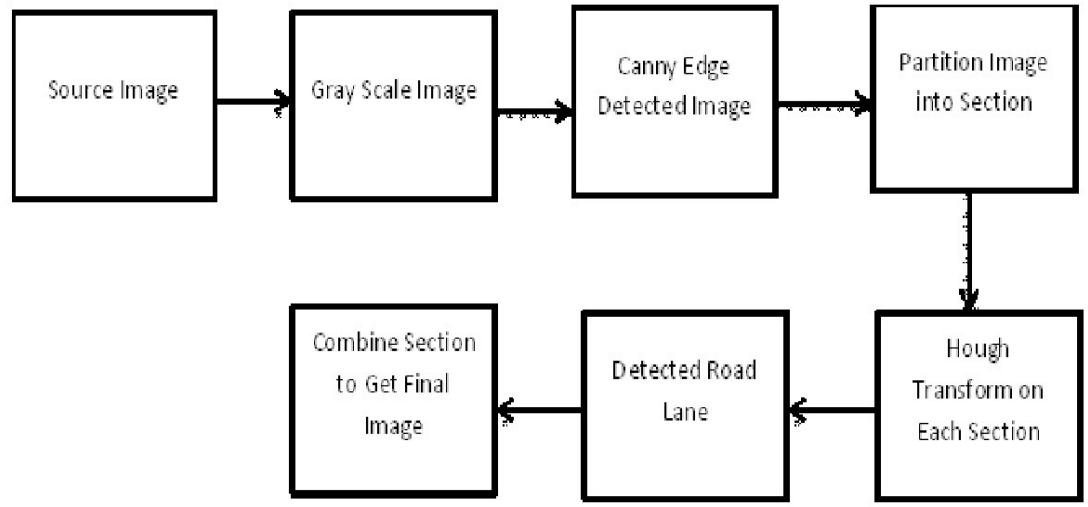

Figure 3. Block Diagram of PHT

Figure 8: Block diagram image of conversion

Chapter 5.

5.Implementation

Program Code:

```
import cv2
import numpy as np
def
make_coordinates(image,line_para
meters):
slope,intercept=line_parameters
print(image.shape)
y1=image.shape[0]
y2=int(y1*(3/5))
x1=int((y1-
intercept)/slope)
17x2=int((y2-
intercept)/slope)
return
np.array([x1,y1,x2,y2])
return
def average_slope_intercept(image,lines):
18left_fit=[]
right_fit
=[]
```

```
for line in lines:
x1,y1,x2,y2=line.resha
pe(4)
parameters=np.polyfit((x1,x2),(y1,
y2),1) slope=parameters[0]
intercept=parameters[1]
if slope <0:
left_fit.append((slope,inter
cept))
else:
right_fit.append((slope,intercept))
19left_fit_average=np.average(left_fit,axis=0)
right_fit_average=np.average(right_fit,axis=
0)
left_line=make_coordinates(image,left_fit_a
verage)
right_line=make_coordinates(image,right_fit
_average) return
np.array([left_line,right_line])
def canny(image):
p
20gray=cv2.cvtColor(image,cv2.COLOR_RG
B2GRAY)
blur=cv2.GaussianBlur(gray,(5,5),0)
canny=cv2.Canny(blur,50,150)
return canny
def display_lines(image,lines):
```

```
line_image=np.zeros_like(i
mage) if lines is not
None:
for x1,y1,x2,y2 in lines:
cv2.line(line_image,(x1,y1),(x2,y2),(255,
0,0),10) return line_image
def
region_of_interest(im
21age): height=image.shape[0]
polygons=np.array([ [(200,height),(
1100,height),(550,250)]
])
mask=np.zeros_like(image)
cv2.fillPoly(mask,polygons,255)
22masked_image=cv2.bitwise_and(ima
ge,mask) return masked_image
cap=cv2.VideoCapture("test2.
mp4") while(cap.isOpened()):
_, frame=cap.read()
canny_image=cann
y(fra me)
cropped_image=reg
ion_
of_interest(canny_i
mage
)
```

```
lines=cv2.HoughLinesP(cropped_image,2,np.pi/180,100,np.array([]),min
Li
neLength=40,m ax LineGap=5)
23averaged_lines=average_slope_intercept(frame,l
ines)
line_image=display_lines(frame,averaged_lines)
combo_image=cv2.addWeighted(frame,0.8,line_
image,1,1) cv2.imshow("result",combo_image)
if cv2.waitKey(1)==ord('q'):
24break
cap.releas
e()
cv2.destroyAllWindows
```

Chapter 6.

6. Output / Screenshot

Figure 9: Input Image

Figure 10: Output Image

Chapter 7.

7. Conclusion

The problem of autonomous highway overtaking was solved. The test protocol for highway overtaking assist was developed which was further used for the development of an automated driving system for the autonomous highway overtaking. The developed test protocol was validated analytically using mathematical equations and the automated driving system was tested virtually. The simulation results were found to be in accordance with the desired host vehicle behavior. The system drives the host vehicle through the selected use cases in a safe and efficient manner, while interacting with target vehicles operating in the traffic environment. The proposed autonomous highway overtaking system has the following characteristics:

Safe, comfortable, and robust: The safety of the developed system was guaranteed by ensuring that the host vehicle remains outside the safe time gap during overtaking. Also, the host vehicle was able to execute the overtaking maneuver without exceeding the safe limits of longitudinal acceleration, heading angle, yaw rate, and lateral velocity, it was concluded that the developed fuzzy controller was suitable for application to the steering control even at a higher speed. Thus, the system is robust enough.

Feasible and modular framework of the autonomous highway overtaking system., the developed system is feasible since it uses the sensors and ADAS systems Also, the development of subsystems is performed independently which might be beneficial to the future optimization and into a real vehicle.

Chapter 8.

8. References

[1] “Automated vs. Autonomous Vehicles: Is There a Difference?”30https://www.autotrader.com/carnews/automated-vs-autonomous-vehicles-there difference273139, 2018. [Online; accessed 25-04-2018].

[2] FENICHE, M., & MAZRI, T. (2019). Lane Detection and Tracking for Intelligent Vehicles: A Survey. 2019 International Conference of Computer Science and Renewable Energies (ICCSRE). doi:10.1109/iccsre.2019.8807727

[3] G. Hegeman, Assisted overtaking: An assessment of overtaking on two-lane rural roads. No. T2008/4, TRAIL Research School Delft, the Netherlands, 2008.

[4] H.Tanveer and A.Sgorbissa, “An Inverse Perspective Mapping Approach usingMonocular Camera of Pepper Humanoid Robot to Determine the Position of Other Moving Robot in Plane” in 15th International Conference on Informatics in Control, Automation and Robotics, IEEE International Conference on ,(2018).

[5] J. E. Stellet, M. R. Zofka, J. Schumacher, T. Schamm, F. Niewels, and J. M. Zöllner, “Testing of advanced driver assistance towards automated driving: A survey and taxonomy on existing approaches and open questions,” in Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on, pp. 1455–1462, IEEE, 2015.

[6] Y.Yenİaydin, K.W.Schmidt, “A lane detection algorithm based on reliable lane markings” in Signal Processing and Communications Applications Conference (SIU), IEEE International Conference on , (2018)