CHAPTER 1

INTRODUCTION

In recent times, the self-driving cars or the autonomous cars are getting more attention, as they are proved to be more convenient and require less human effort for travelling. The challenges related to the autonomous vehicles are discussed, like recognizing the traffic lights, the pedestrians and recognizing the traffic signs using the deep learning methods.

Automatic lane detection to help the driver is an issue considered for the advancement of Advanced Driver Assistance Systems (ADAS) and a high level of application frameworks because of its importance in drivers and passer by safety in vehicular streets. A new computer vision based system is designed in the way that it helps in intelligent automotive by giving the driver assistance. Using a colour based segmentation method the lanes are detected.

1.2 FIGURES

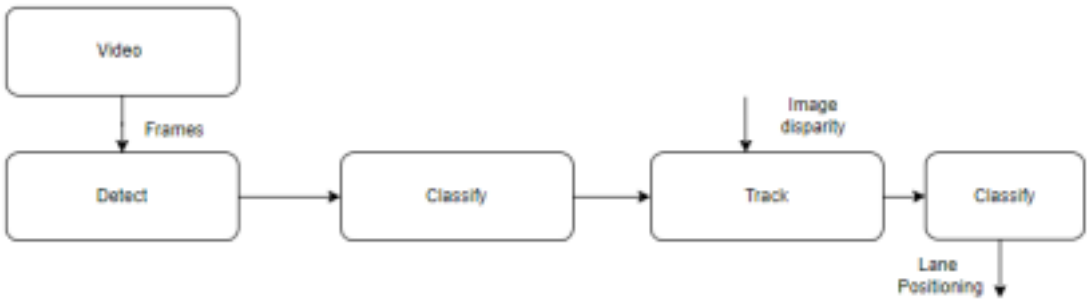


Figure 1.1: Simple Flow chart of lane detection system

1.2 SCOPE

Beyond safety, self-driving systems could unlock other benefits, including traffic awareness, decreased insurance premiums, increased fuel efficiency and infrastructure use efficiencies.

It helps consumers become comfortable with allowing machines to do work for them. Autonomous robots are already becoming integrated into retail, and in the future, they will become an integral part in the operation of many other industries.

With fewer traffic jams and more efficient driving experiences, repair and maintenance costs will decrease, leading to lower insurance premiums. Inherently, this will also increase fuel efficiency.

Still, it will take at least a few more years before both society and the automotive industry are ready for the mass adoption of autonomous cars.

1.3 NOVELTY OF THE IDEA

We will detect the lanes even during vehicles travelling at night time. In this project we also implement to detect vehicles using semantic segmentation. Image processing is done mainly through using LiDAR, but in our project we are using a computer vision based technique where video can be obtained through a normal camera which is highly available in our regular life and by apply various algorithms and mathematical model we build accurate model for lane detection. As per the Government reports 90% of the accidents occurs based on human errors this number can be bought to significantly low using Self driving vehicles.

.

CHAPTER 2

PROBLEM DEFINITION

CHAPTER 2 PROBLEM DEFINITION

In everyday life we see people who are disabled, elderly people who need personal care are not able to drive themselves. Automation reduces the crash of vehicles and most of the accidents occur due to human errors, even sometimes there could be threat to pedestrians.

In order to tackle this problem we have come up with a solution which can be used in automated vehicles in order to detect lanes.

But still, now it is a most challenging problem because of some factors that are faced by lane detection systems like vagueness of lane patterns, perspective consequence, low visibility of the lane lines, shadows, incomplete occlusions, brightness and light reflection. The system detects the lane boundary lines using computer vision-based technologies. we introduce a system that can efficiently identify the lane lines on the smooth road surface.

2.2 OBJECTIVES

Using the Computer vision technique frames of the recorded video used as input, using algorithms we analyze the frames we obtain a video where lanes are detected.

Enabling faster and more accurate mapping to a particular input or an input aspect using Semantic segmentation.

By all the following techniques, at the end of the project we will build a model where we can achieve accurate lane positions and display it on an output video file.

CHAPTER 3

LITERATURE REVIEW

CHAPTER 3 LITERATURE REVIEW

In this paper they proposed the approach to detect lanes, detect and track multiple vehicles for lane change support around the test vehicle. For lane detection, to detect lane in real-time, they used EDLines algorithm which can detect line segments between 10 ms and 20 ms on 2.2 GHz CPU, and EDlines was applied to ROI. Therefore, lane detection method has been implemented in the3.3 GHz Intel CPU and it takes about 13 ms with each the image. With frontal view, our algorithm detects three lane areas, frontal lane, left-side lane. Moreover, both rearside lane were detected with two rear-side cameras by using our method.[1]

In this paper there are two input images. One which is used for classic approach and the other is used for deep learning in our project. The algorithm followed in this paper is to detect lane markings on the road by giving the video of the road as an input to the system by using computer vision technology and primarily designed with the objective of reducing the frequency of accidents. System can be installed in cars and taxis in order to prevent the occurrence of accidents due to reckless driving on the roads. In this project they used Region of Interest(ROI), main objective of ROI is to decrease the portion of an image for speedy calculation and also the size of image can be decremented by ROI generation.[2]

This thesis shows how line and lane detection is possible with the provided hardware. The system can find edges from a camera picture, classify these edges and calculate a virtual line that the car attempted to drive according to. A camera stream was implemented as a proof of concept which displayed the above in addition to an instrument panel which showed the errors the car tried to regulate. However, due to time constrains, this thesis was not able to correctly regulate the errors calculated. To summarize, Inherited hardware from a previous project and built the software required for the car to be able to drive autonomously. The software created for this thesis had no influence from the software used for the previous project.[3]

This paper discusses the results of implementation of lane detection algorithm on toll road Cipularang as parts of self-driving car system. Video image taken using action camera mounted on top of the vehicle, with 1280x720 resolution. Average speed of the vehicle is 100 km per hour. Programming language of image processing using Python 3. Image processing method are a combination of methods of colour region, line selection, canny edge detection, and Hough transform. In this project they used a particular region and trained to get the lane detection model and Object classification is performed using one class linear SVM is given the training data for each object with positive and negative data.[4]

In this paper proposed a lane detection method based on OpenCV. Preprocessing image in the OpenCV environment, adopting LMedSquare (Least Median Square) idea to select the best subset combined with least squares method to piecewise fitting the lane so that it realized automatic identification of lane. This algorithm is suitable for both straight and curve. Simulation shows that this algorithm has well real-time performance, accuracy and robustness. It can meet the requirements of the vehicle system.[5]

CHAPTER 4
PROJECT DESCRIPTION

CHAPTER 4 PROJECT DESCRIPTION

Autonomous Driving Car is one of the most disruptive innovations in AI. Fuelled by Deep Learning algorithms, they are continuously driving our society forward and creating new opportunities in the mobility sector. An autonomous car can go anywhere a traditional car can go and does everything that an experienced human driver does. But it's very essential to train it properly. One of the many steps involved during the training of an autonomous driving car is lane detection, which is the preliminary step.

Capturing and decoding video file: We will capture the video using Video Capture object and after the capturing has been initialized every video frame is decoded (i.e. converting into a sequence of images).

Grayscale conversion of image: The video frames are in RGB format, RGB is converted to grayscale because processing a single channel image is faster than processing a three-channel coloured image.

Reduce noise: Noise can create false edges, therefore before going further, it's imperative to perform image smoothening. Gaussian filter is used to perform this process.

Canny Edge Detector: It computes gradient in all directions of our blurred image and traces the edges with large changes in intensity. For more explanation please go through this article: Canny Edge Detector

Region of Interest: This step is to take into account only the region covered by the road lane. A mask is created here, which is of the same dimension as our road image. Furthermore, bitwise AND operation is performed between each pixel of our canny image and this mask. It ultimately masks the canny image and shows the region of interest traced by the polygonal contour of the mask.

Hough Line Transform: The Hough Line Transform is a transform used to detect straight lines. The Probabilistic Hough Line Transform is used here, which gives output as the extremes of the detected lines.

4.1 PROPOSED DESIGN

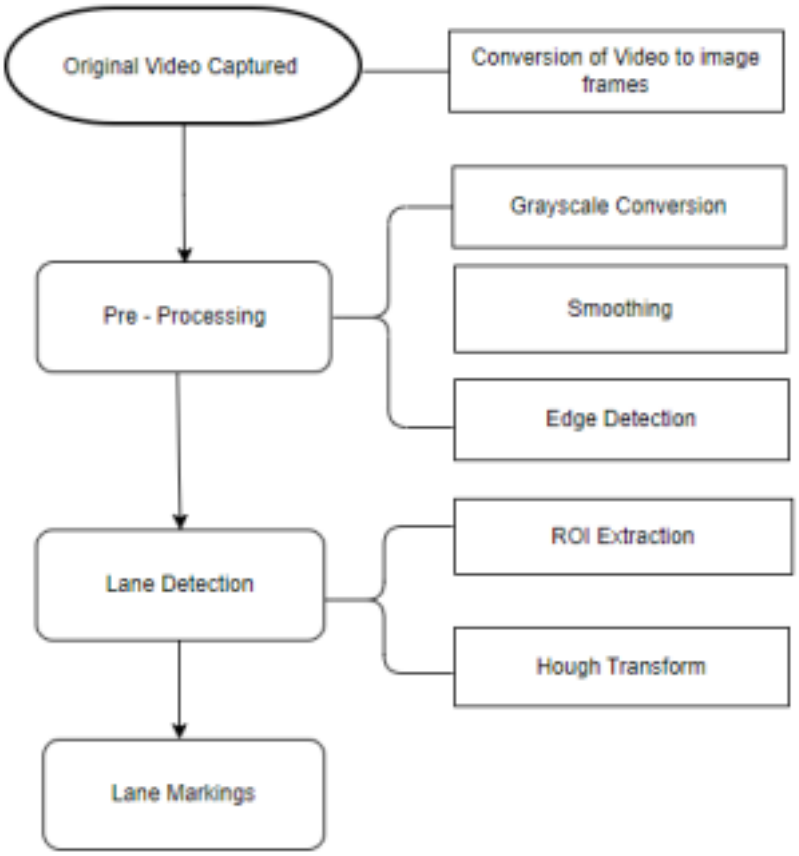


Figure 1.2: Design of the project

4.2 ASSUMPTIONS AND DEPENDENCIES

We'll be taking a recorded video of a road which contains lanes, the recorded video has stability and correct lane markings. The recorded video can either be day or night. Firstly we convert video into frames, and for that frames we will segregate the useful and not useful components. After that we obtain a data in which we could extract road lane lines and this will be further processed to get accurate lane line and this will be marked using any type of colour

which is visible by changing the parameters according to a video we can find best lane detection system.

CHAPTER 5
REQUIREMENTS

CHAPTER 5 REQUIREMENTS

5.1 FUNCTIONAL REQUIREMENTS

We are implementing project on lane detection system using computer vision has functional requirements.

• Captured video stability should be good so that there is less noise and it is easy to get the accurate output.

• Image frame analyzing is a static image frame to mark lanes and use the different data present in the image and apply algorithms on it.

• Required frame of the image should have lane lines so that algorithms can detect the constraints or edges of the lane.

• Video file format for continuous lane detection. Once the image processing is done we will apply the same technique to the video file format also so that it can detect the lanes dynamically.

5.2 NON-FUNCTIONAL REQUIREMENTS

• It is Reliable increases driving safety of automatic driving vehicles.

• It provides Safety to the people who are travelling in vehicles and to pedestrians as well.

• It is Compatible to any recorded video file.

• It also provides Security.

• It provides availability, System should be available at all the time to detect the lanes.

CHAPTER 6
METHODOLOGY

CHAPTER 6 METHODOLOGY

6.1 PROPOSED METHODOLOGY

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize lanes on the road, it needs to be fed vast quantities of lane images and lane-related items to learn the differences and recognize a lane, especially one with no defects.

We'll be performing the following tasks in our project:

a)Greyscale conversion: It is an image conversion technique in digital photography. It eliminates every form of colour information and only leaves different shades of gray; the brightest being white and the darkest of it being black.

b)Smoothing: Smoothing is used to reduce noise or to produce a less pixelated image. Most smoothing methods are based on low-pass filters, but you can also smooth an image using an average or median value of a group of pixels that moves through the image. **c)Edge detection**: Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

d)ROI Extraction: A region of interest (ROI) is a portion of an image that you want to filter or operate on in some way. You can represent an ROI as a binary mask image. In the mask image, pixels that belong to the ROI are set to 1 and pixels outside the ROI are set to 0.

e)Hough transform: The Hough transform is a popular feature extraction technique that converts an image from Cartesian to polar coordinates. Any point within the image space is represented by a sinusoidal curve in the Hough space.

6.2 HARDWARE REQUIREMENTS

1) Intel i5 CPU, with an i7 recommended

2) 8GB RAM, with 16GB recommended

3) 1920 x 1080 resolution display

6.3 SOFTWARE REQUIREMENTS

1) OS: Windows 10 OS.

2) Language: Python3.

3) IDE: MATLAB, Google colab, PyTorch, Visual Studio

CHAPTER 7
EXPERIMENTATION

CHAPTER 7 EXPERIMENTATION

```
    # Convert the frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Gaussian Blur to reduce noise
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Canny edge detection
    edges = cv2.Canny(blurred, 50, 150)

    # Define a mask to only consider the region of interest (ROI)
    mask = np.zeros_like(edges)
    ignore_mask_color = 255
    imshape = frame.shape
    vertices = np.array([[(0, imshape[0]), (700, 700), (300, 350), (imshape[1], imshape[0])]], dtype=np.int32)
    cv2.fillPoly(mask, vertices, ignore_mask_color)
    masked_edges = cv2.bitwise_and(edges, mask)

    # Define the Hough transform parameters
    rho = 1  # distance resolution in pixels of the Hough grid
    theta = np.pi / 180  # angular resolution in radians of the Hough grid
    threshold = 5  # minimum number of votes (intersections in Hough grid cell)
    min_line_length = 30  # minimum number of pixels making up a line
    max_line_gap = 10 # maximum gap in pixels between connectable line segments
```

```
    # Run Hough on edge detected image
    lines = cv2.HoughLinesP(masked_edges, rho, theta, threshold, np.array([]),
                            min_line_length, max_line_gap)
    filtered_lines = []
    for line in lines:
        x1, y1, x2, y2 = line[0]
        measured = np.array([[np.float32(x1)], [np.float32(y1)]])
        predicted = kalman.predict()
        if abs(measured[0] - predicted[0]) + abs(measured[1] - predicted[1]) < 25:
            kalman.correct(measured)
            filtered_lines.append(line)


    # Draw the lines on the frame
    for line in lines:
        for x1, y1, x2, y2 in line:
            cv2.line(frame, (x1, y1), (x2, y2), (200, 300, 100), 3)

    return frame

# Read the video stream
cap = cv2.VideoCapture("night.mp4")
```


Importing the modules

```
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
```

Frames plotting on a graph

```
capture  = cv2.VideoCapture("/Video_1.mp4")
ret, frame = capture.read()
plt.imshow(frame)
plt.show()
```

Grey scaling

```
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
```

Gaussian Blur

```
   blur = cv2.GaussianBlur(gray,(5, 5),0)
```

Canny Edge Detection

```
canny = cv2.Canny(gray, 50, 150)
return canny
```

Sobel edge detection

```
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)

# Compute the magnitude of the Sobel gradients
mag = np.sqrt(sobelx*2 + sobely*2)

# Normalize the magnitude values to 0-255 range
mag = np.uint8(255 * mag / np.max(mag))
```

Region of interest

```
import numpy as np
mask = np.zeros_like(edges)
    ignore_mask_color = 255
    imshape = frame.shape
    vertices = np.array([[(0, imshape[0]), (100, 600), (800, 450), (imshape[1], imshape[0])]], dtype=np.int32)
    cv2.fillPoly(mask, vertices, ignore_mask_color)
    masked_edges = cv2.bitwise_and(edges, mask)
    return masked_edges
```

Hough Transform

```
def hough_transform(img):
  houghLines = cv2.HoughLinesP(img, 2, np.pi/180, 10, np.array([]), minLineLength=250, maxLineGap=5)
  return houghLines
```

Display Lines

```
def displayLines(img, Lines):
  line_mark = np.zeros_like(img)
  if Lines is not None:
    for line in Lines:
      x1,y1,x2,y2 = line.reshape(4)
      cv2.line(line_mark, (x1,y1), (x2,y2), (255,0,0),10)
      return line_mark
```


Kalman filter

```python
def process_frame(frame):
    kalman = cv2.KalmanFilter(4, 2)
    kalman.measurementMatrix = np.array([[1, 0, 0, 0], [0, 1, 0, 0]], np.float32)
    kalman.transitionMatrix = np.array([[1, 0, 1, 0], [0, 1, 0, 1], [0, 0, 1, 0], [0, 0, 0, 1]], np.float32)
    kalman.processNoiseCov = np.array([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]], np.float32) * 0.03
```

```python
    filtered_lines = []
    for line in lines:
        x1, y1, x2, y2 = line[0]
        measured = np.array([[np.float32(x1)], [np.float32(y1)]])
        predicted = kalman.predict()
        if abs(measured[0] - predicted[0]) + abs(measured[1] - predicted[1]) < 25:
            kalman.correct(measured)
            filtered_lines.append(line)
```

CHAPTER 8 TESTING AND RESULTS

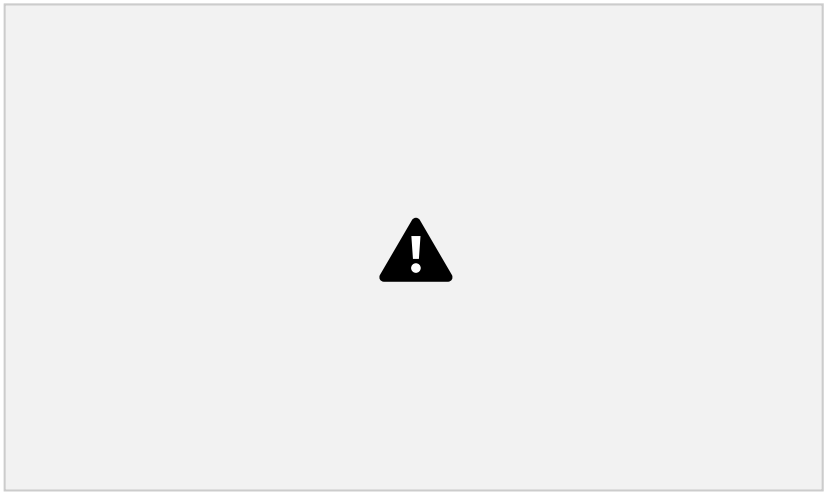


Fig 1.3: Plotting image on a graph

Smoothing Techniques:

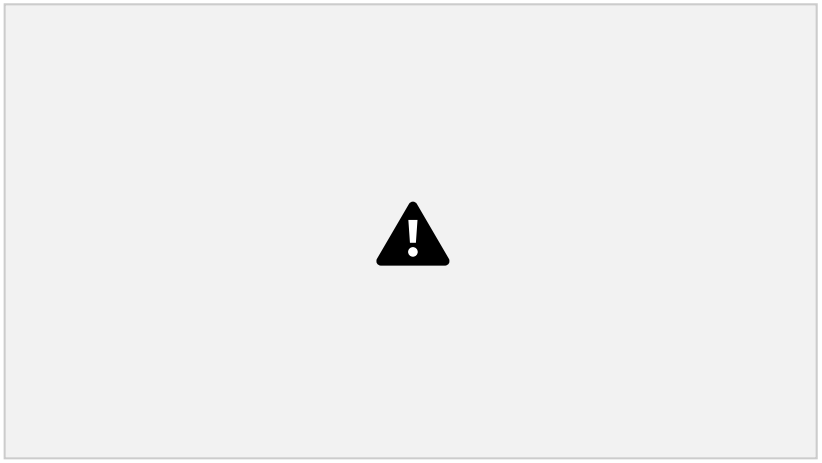


Fig 1.4: Grey Scaling

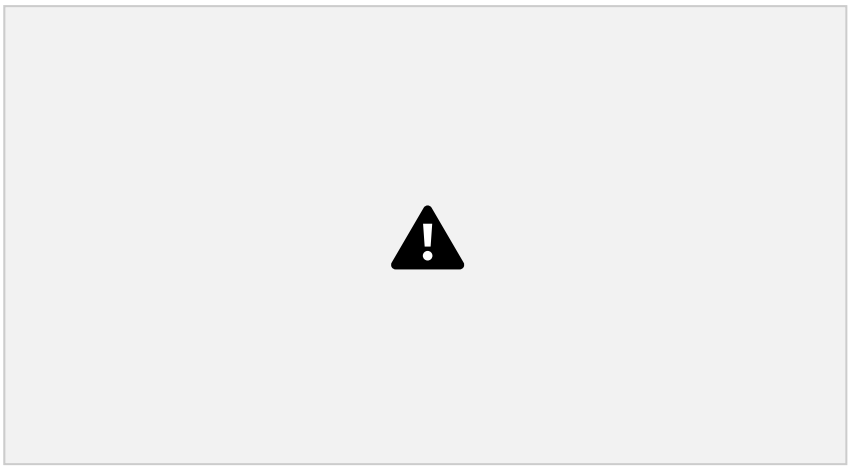


Fig 1.5 : Gaussian Blur

Edge Detection Techniques:

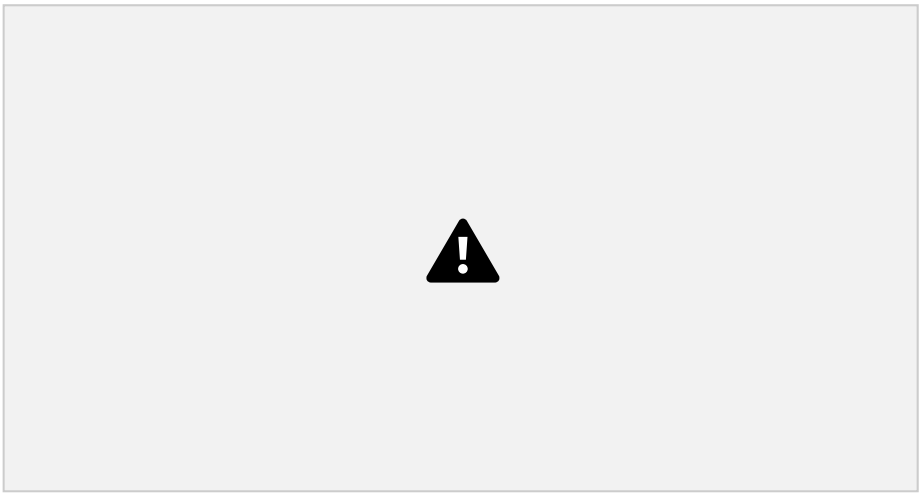


Fig 1.6: Canny Edge Detection

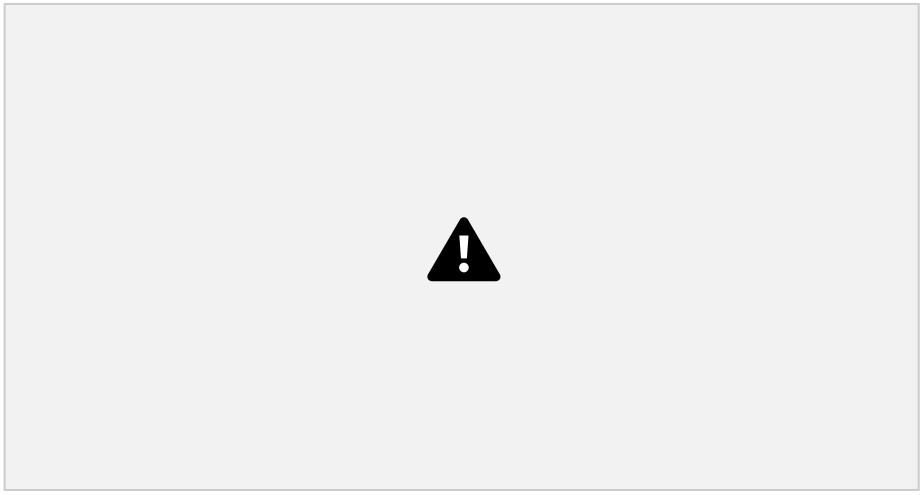


Fig 1.7: Sobel edge detection

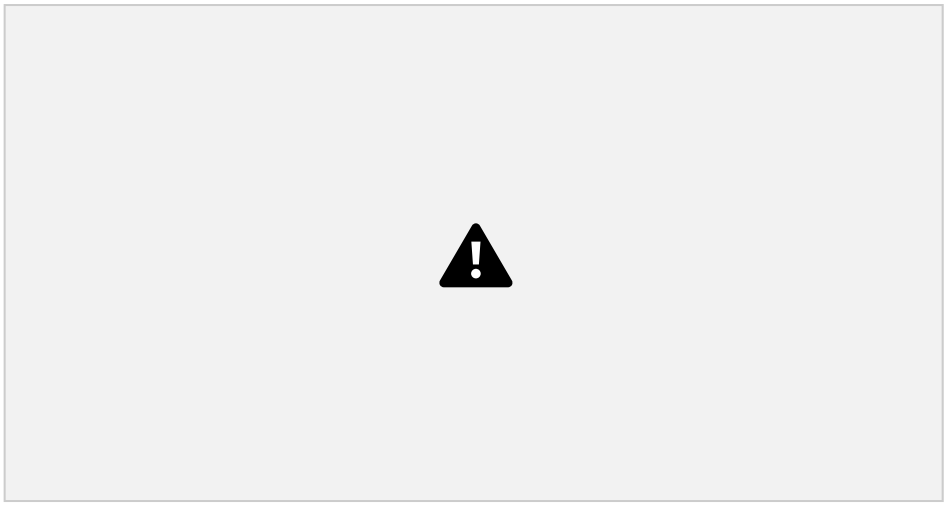
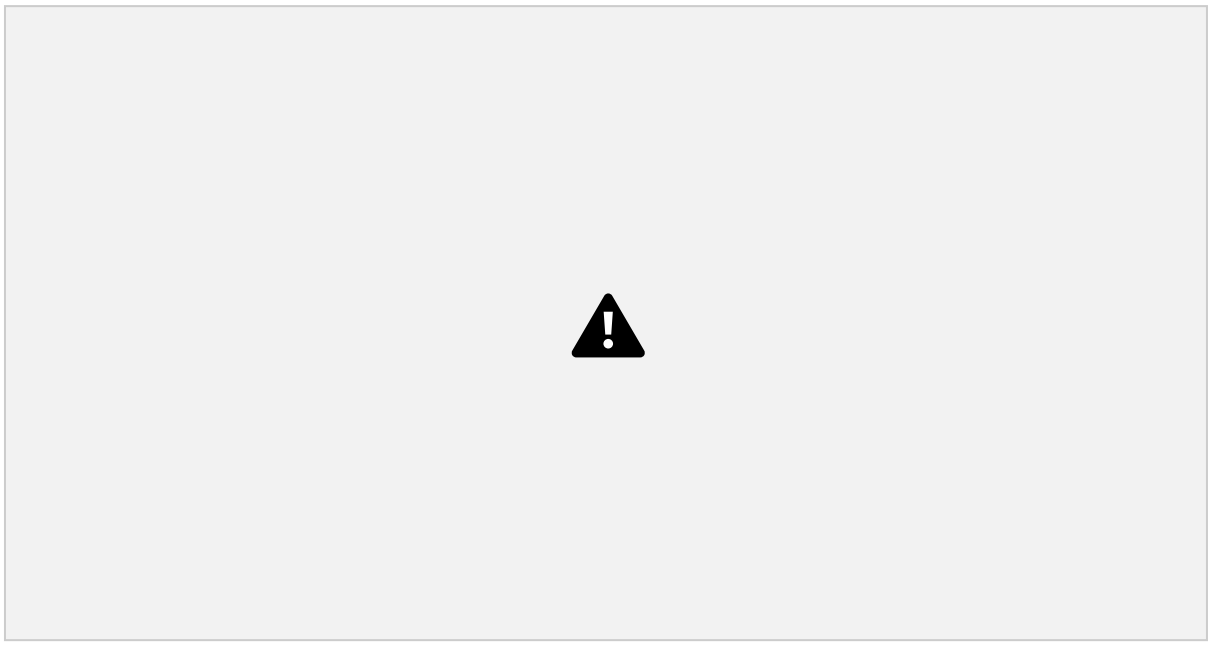
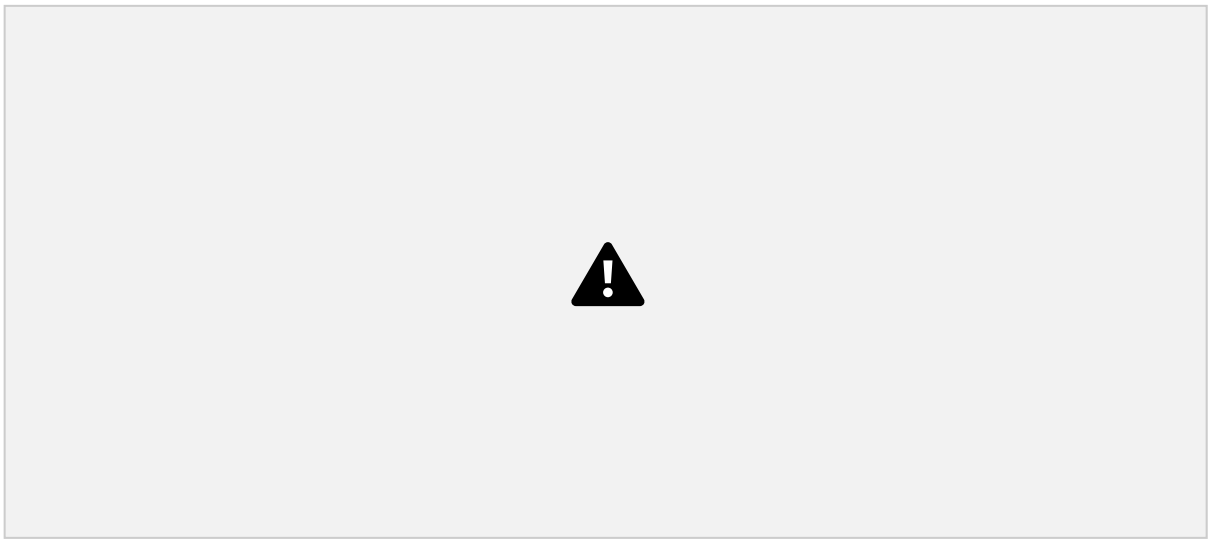


Fig 1.8: Region Of Interest (ROI)



Lane Detection using canny edge detection System



Lane Detection using Sobel edge detection

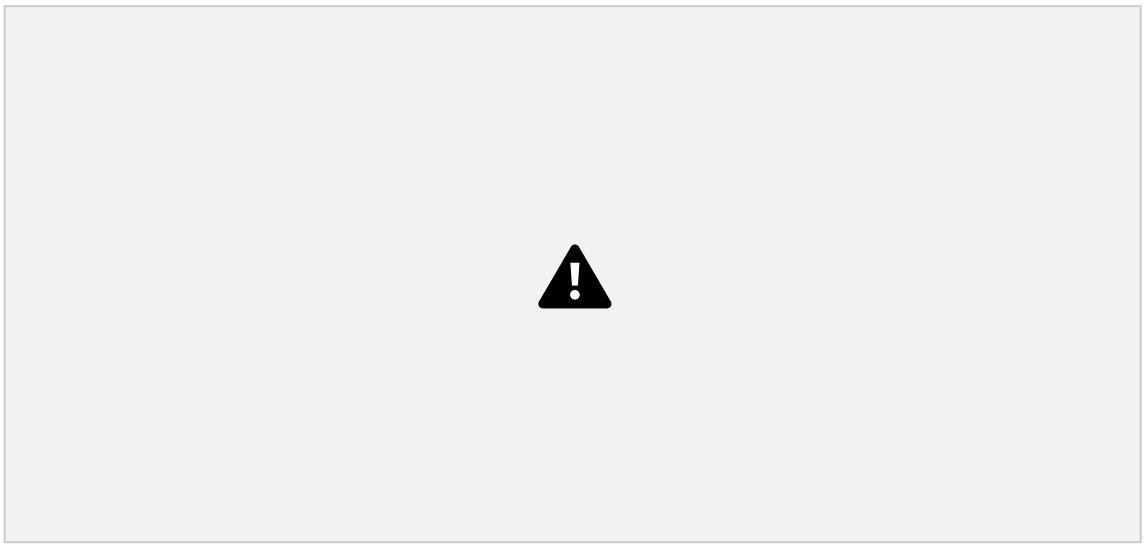


Fig 1.9 Lane Detection using Kalman filter

CHAPTER 9 CONCLUSION

9. 1 CONCLUSION

Lane detection using computer vision has shown promising results, it still faces some challenges, such as dealing with varying lighting conditions, complex road geometry, and occlusions caused by other vehicles or objects.

Overall, lane detection using computer vision is a rapidly evolving field that has the potential to significantly improve the safety and efficiency of autonomous driving and ADAS systems. As computer vision and machine learning techniques continue to advance, we can expect to see even more accurate and robust lane detection algorithms in the future.

9.2 SCOPE FOR FUTURE WORK

Improving accuracy and robustness: While current lane detection algorithms have shown promising results, there is still room for improvement in terms of accuracy and robustness. Future work could focus on developing more advanced computer vision and machine learning techniques to improve lane detection performance, especially in challenging conditions like poor lighting, adverse weather, or complex road geometry.

Real-time implementation: Another important area for future work is the real-time implementation of lane detection algorithms. This involves optimizing algorithms to run efficiently on embedded systems, such as those used in autonomous vehicles or ADAS systems, where low latency and high frame rates are critical.

REFERENCES

[1] Raman Shukla, Rajat Shukla, Sarthak Garg and Pooja Vajpayee "Lane line Detection System in Python using OpenCV" International journal of Innovative Research In Technology (IJIRT) Vol 8, June 2021.

[2] Abhishek Goyal, Mridula Singh and Anand Srivastava "Lane Detection on Roads using Computer Vision" International Journal of Engineering and Advanced Technology (IJEAT) Vol 9, October 2019.

[3] Aditya Subramanian "Car Vision: Lane detection and Following" Thesis carried by Embedded systems department of Infotiv Gothenburg, Sweden June 2017.

[4] Mochamad Vicky Ghani Aziz, Ary Setijadi Prihatmanto and Hilwadi Hindersah "Implementation of Lane Detection Algorithm for Self-Driving Car on Toll Road Cipularang using Python Language" 4th International Conference on Electric Vehicular Technology (ICEVT) October 2-5, 2017.

[5] Xu Yang and Zhang Ling “Research on Lane Detection Technology Based on OpenCV” 3rd International Conference on Mechanical Engineering and Intelligent Systems (ICMEIS) 2015.