Lane Line Detection

- INSTRUCTOR: Dr. Rajasoundaran

- Team Members:
- Mada Sai Surya-19MIM10095
- Kadiyala Meghanath-19MIM10097
- Abishek Swamy-19MIM10117
- Praneeth Gadipudi-19MIM10044

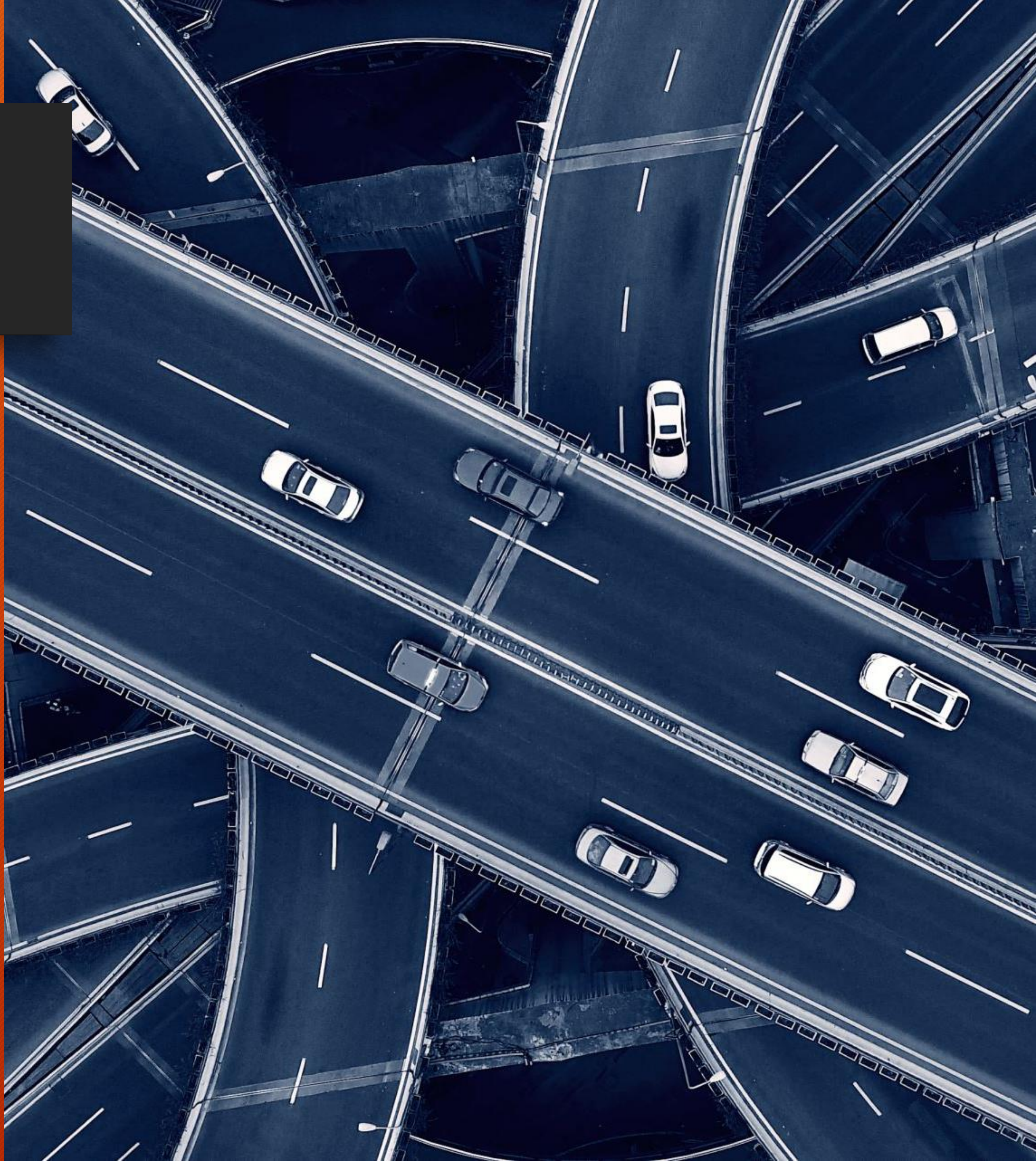
ZEROTH REVIEW

Topics

- Introduction
- Existing work
- Proposed work and Methodology
- Real time usage
- Hardware & Software requirements
- System Architecture

Introduction

- Autonomous Driving Car is one of the most disruptive innovations in AI. Fueled 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.



Existing work

- Jung et al. Developed a lane marking modality using spatiotemporal images which are collected from the video. The spatiotemporal image created by accruing a set of pixels that are mined on a horizontal scan-line having a static location in every frame along with a time axis. Hough transform is applied to the collected images to detect lanes. The system is very effective for short-term noises such as mislaid lanes or obstruction by vehicles. The system obtained the computational efficacy as well as the higher detection rate.

- Wu et al. designed a lane detection and departure warning scheme by determining the region of interest (ROI) in the region near to the automobile. The ROI is divided into non-overlapping chunks and to get the chunk gradients and chunk angles, two basic masks are developed that decrease the computational complexity. The driving situations are classified into four classes, and the departure system is developed with respect to lane detection outcomes. From the experimental outcomes, it is shown that the average lane detection rate is 96.12% and the departure warning rate is 98.60%. However, it takes comparatively high processing time due to computing the vertical and horizontal gradients

PROPOSED METHODOLOGY FOR LANE DETECTION

Our Project uses the latest machine learning algorithms and Python popular package **(OpenCv).**

Methodology:

- **Capturing and decoding video file:** We will capture the video using VideoCapture object and after the capturing has been initialized every video frame is converted 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 colored 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.

- **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.

Real time usage

- Nowadays autonomous driving has become the big trend in the automotive industry, Every Manufacturer is implementing Auto-Pilot mode in their Vehicles. Tesla is the best Example for Self-Driving Car even though there are many Companies manufacturing them.

Hardware & Software requirements

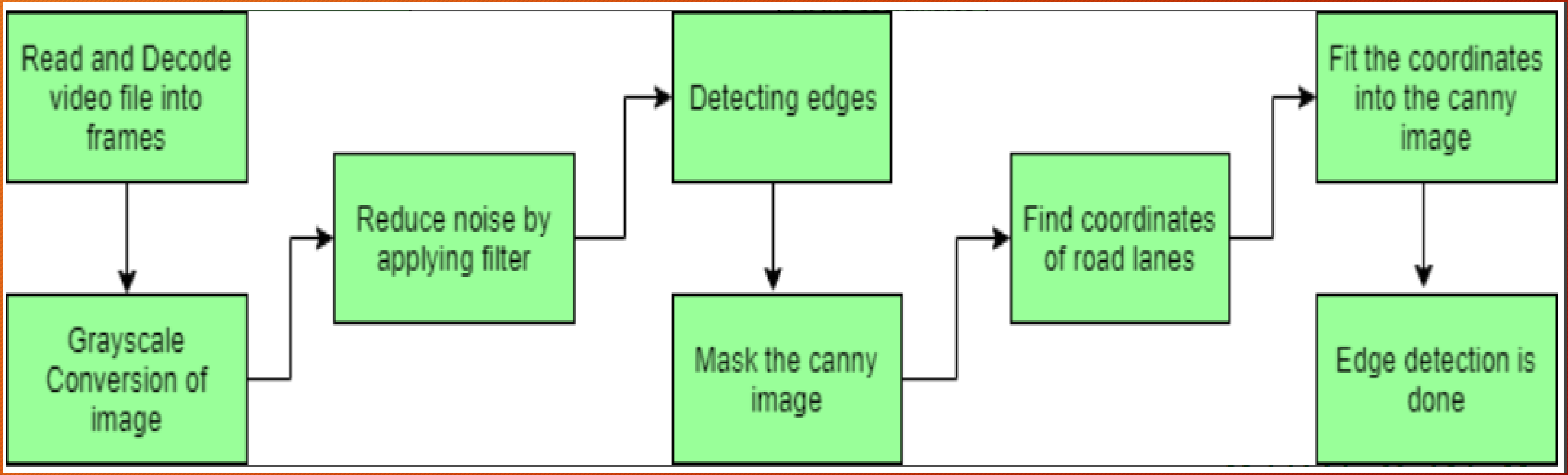
Hardware :

- Min. 4Gb of Ram
- Min. 2 Cores
- Min. 50Gb of SSD or HDD

Software :

- Anaconda (Python Platform)
- OpenCv package
- Jupyter Notebooks

System Architecture


Read and Decode video file into frames
Grayscale Conversion of image
Reduce noise by applying filter
Detecting edges
Mask the canny image
Find coordinates of road lanes
Fit the coordinates into the canny image
Edge detection is done

FIRST REVIEW

Literature review

- Road Accidents are very common these days, to prevent accidents and life loss, automation is required. We cannot completely reduce accidents by implementing automation and using self-driving cars in our daily life but we can reduce as much as possible.
- We implement Lane Line Detection by using different Algorithms like Canny Edge Detector, Hough Line Transformation,etc.

- Tseng et al. [2005] gave a lane marking detection algorithm by using geometry information and modified Hough transform. In that algorithm the captured image was divided into road part and non-road part by using camera geometry information. The color road image was quantized into a binary image. The modified Hough transform with road geometry consideration was used to detect the lane markings. The histogram of intensities was applied to quantize the road image into a binary image. A modified Hough transform method has been developed to detect the lane markings in road image by using the road geometry information. It was time consuming because Hough transform was a full search algorithm in parameter space. It also failed when the lane boundaries intersected in a region which was a non-road part.

Module description

- Modules used:
  - Canny Edge Detector
  - Region of Interest
  - Hough Line Transformation

Canny Edge Detector

- The most used widely edge detection technique in Computer Vision (J Canny 1986)
- Algorithm:
  - Filter image with derivative of Gaussian
  - Find magnitude and orientation of gradient
  - Non-maximum Suppression
  - Linking and thresholding(hysteresis)
    - Define two thresholds: low and high
    - Use the high threshold to start edge curves and low threshold to continue them.

Region of Interest

- The dimensions of the image are chosen which will contain the road lanes and mark it as our region of interest or the triangle. Then a mask is created which is same as the dimension of the image which would essentially be an array of all zeros. Now we fill the triangle dimension in this mask with the intensity of 255 so that our region of interest dimensions are white. Now we will do a bitwise AND operation with the canny image and the mask which will result in our final region of interest.

Below is the Original Image

Here is the Masked image

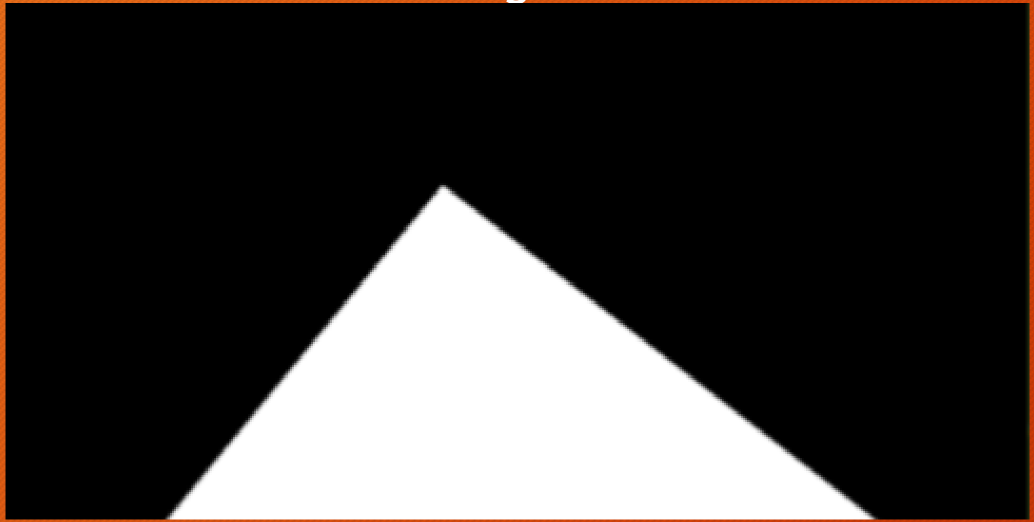
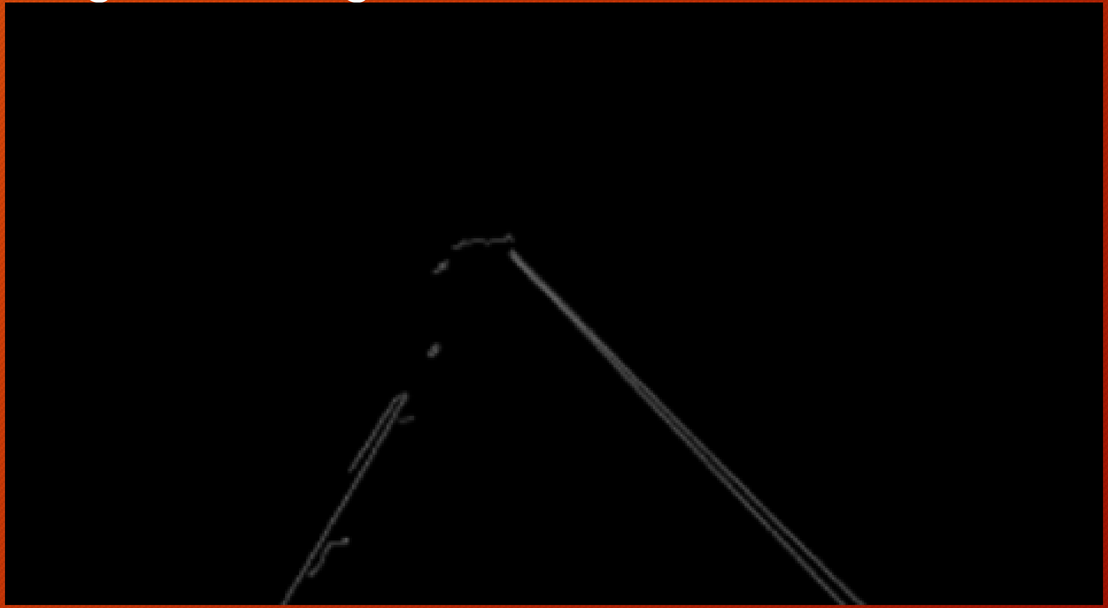


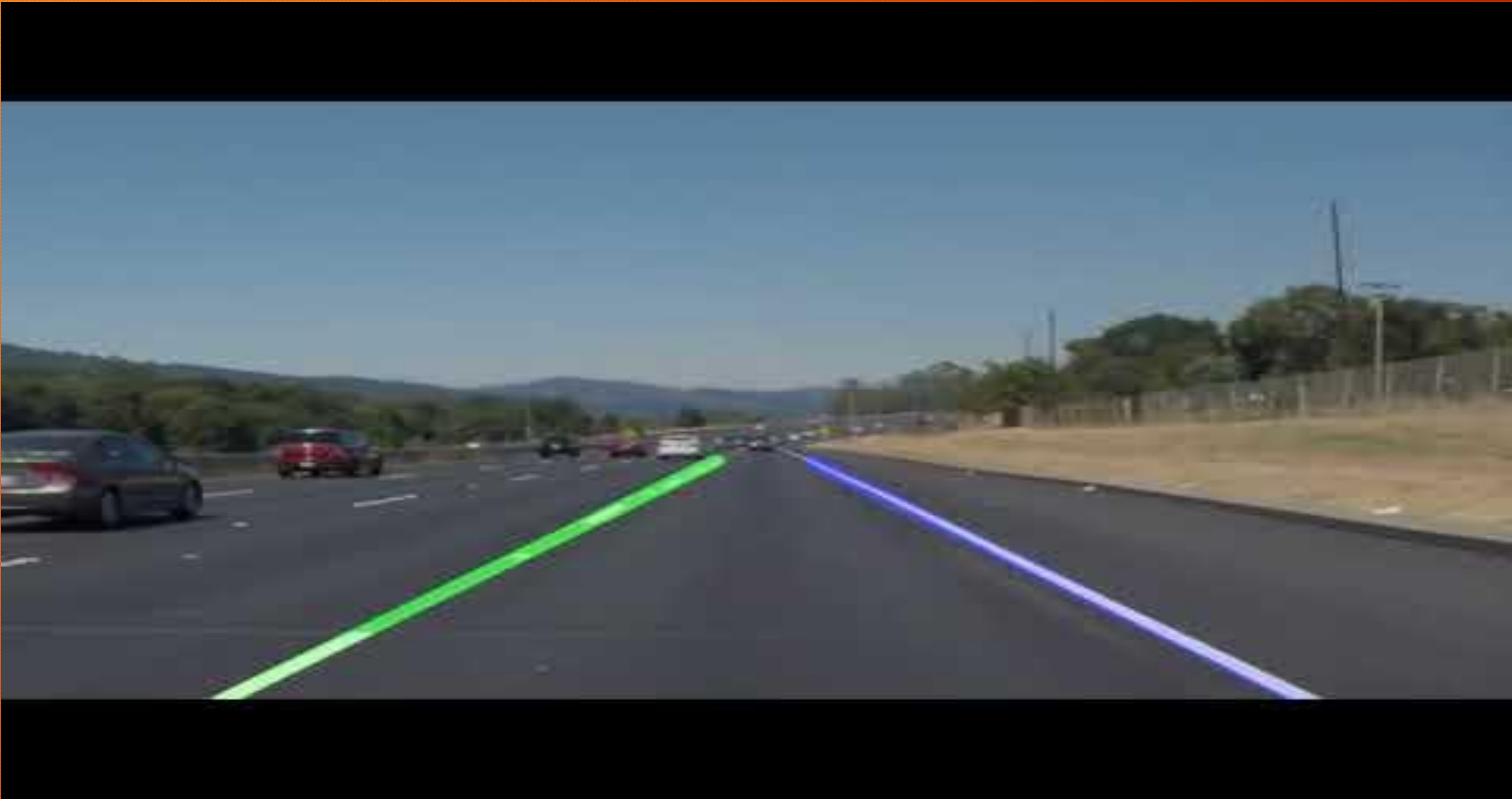
Image of our Region of Interest



Implementation and coding.

```
hsv_img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
mask_white = cv2.inRange(img, (200,200,200), (255, 255, 255))
mask_yellow = cv2.inRange(hsv_img, (15,60,20), (25, 255, 255))
color_mask = cv2.bitwise_or(mask_white, mask_yellow)
masked_img = np.copy(img)
masked_img[color_mask == 0] = [0,0,0]
gray_img = grayscale(masked_img)
kernel_size = 5
blurred_gray_img = cv2.GaussianBlur(gray_img, (kernel_size, kernel_size), 0)
### detect edges ###
low_threshold = 50
high_threshold = 150
edges_from_img = cv2.Canny(blurred_gray_img, low_threshold, high_threshold)
### select region of interest###
imshape = img.shape
vertices = np.array([[(0,imshape[0]),(4*imshape[1]/9, 6*imshape[0]/10), (5*imshape[1]/9, 6*imshape[0]/10),
(imshape[1],imshape[0])]], dtype=np.int32)
masked_edges = region_of_interest(edges_from_img, vertices)
rho = 2
theta = np.pi/180
 threshold = 15
 min_line_len = 10
 max_line_gap = 5
line_img = cv2.HoughLinesP(masked_edges, rho, theta, threshold, np.array([]), minLineLength=min_line_len,
maxLineGap=max_line_gap)
```

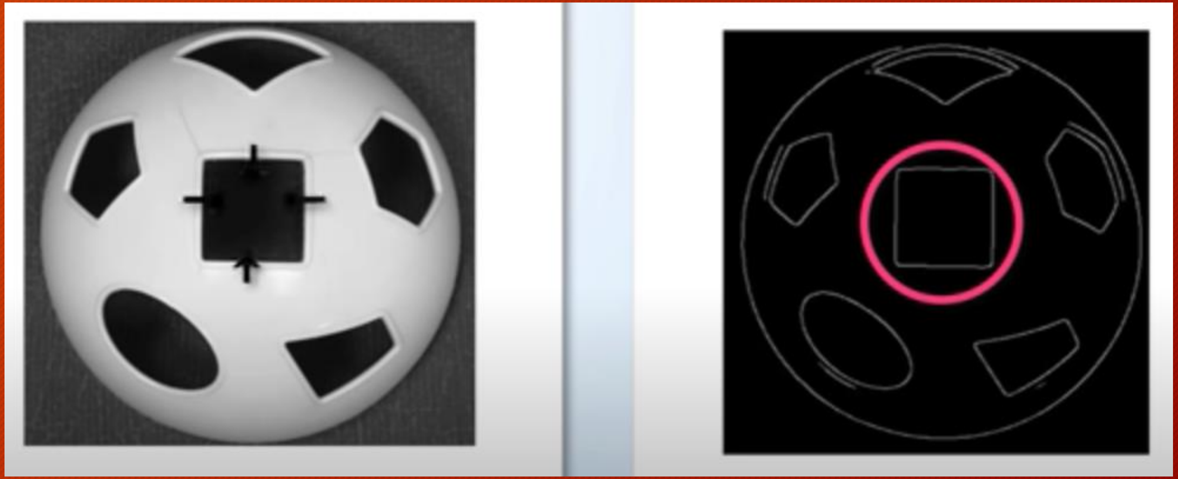
Demo video



Final Review

Module 1 – Edge Detector

- Intensity of Black is : 0 and Intensity of White is : 255
- Gradient : Measure of change in brightness(Intensity Value) over adjacent pixels
  - Stong Gradient : When there is a steep change in Gradient then it is known as Stong Gradient Eg.: When the adjacent pixel changes from 0 --> 255 i.e black to white
  - Small Gradient : Represents a Shallow change. Eg: 0 --> 15 i.e black to dark brown.
- Edge : Rapid change in Brightness i.e Wherever there is a Strong Gradient then the edge is detected



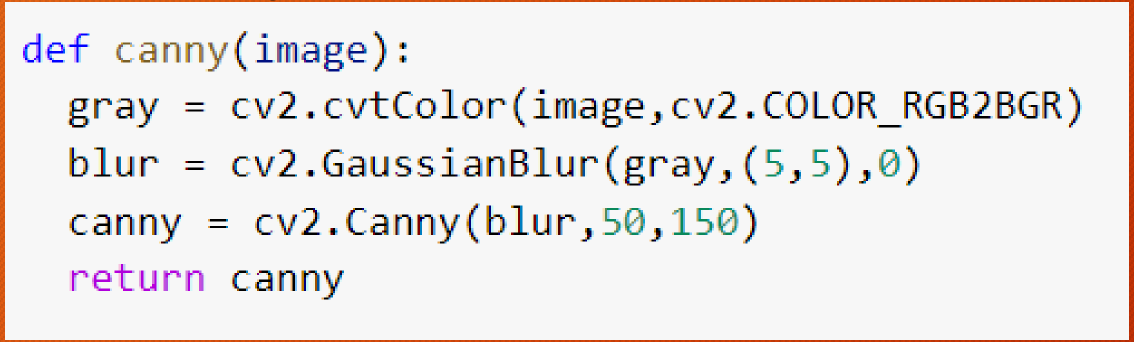
Now Applying the Intensity and Gradient concept in our road image.

Firstly, we convert our road image to Grayscale why? When we convert into Grayscale we would have only one channel of intensity values to deal with and it is easier instead of dealing with three channels of intensity pixels(Red, Blue and Green)

Our second step is to reduce the noise in our road image with the help of Gaussian Filter. What Gaussian filter does is it modifies each pixel value in our image by taking average of all pixels.

Our Final step in Edge detection is to apply Canny Method of Edge Detention. Since we are looking for Strong Gradient this method finds the derivative of pixels. When there is big derivative then there is a strong derivative. When strong Gradient is detected we have an edge.

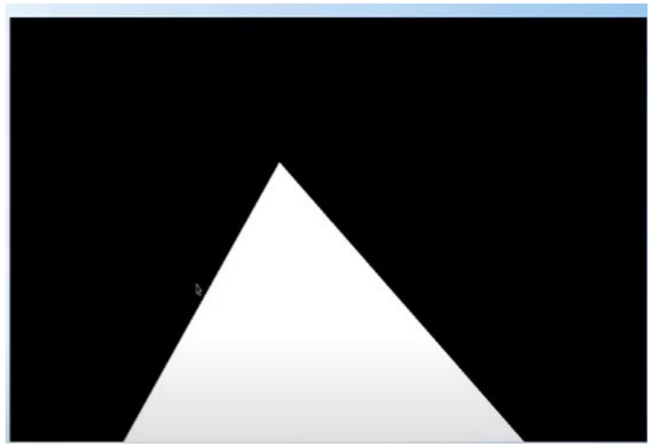
Code for Edge Detector

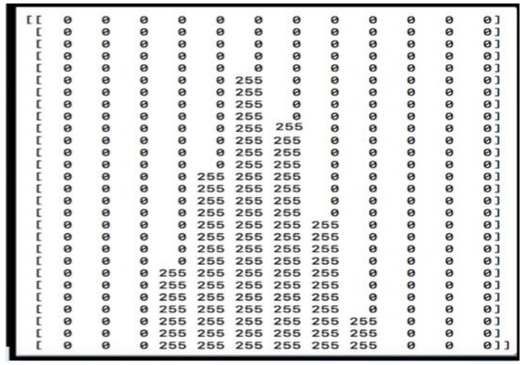


```python
def canny(image):
  gray = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
  blur = cv2.GaussianBlur(gray,(5,5),0)
  canny = cv2.Canny(blur,50,150)
  return canny
```

Module 2-Region of interest

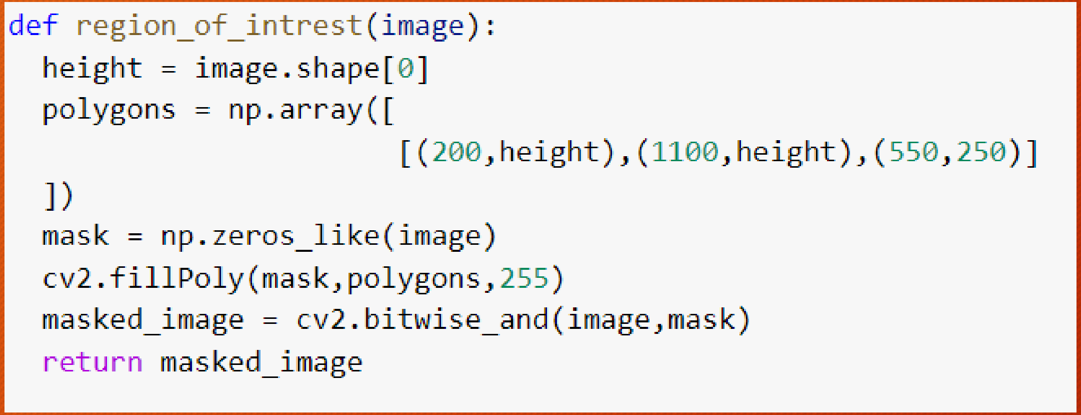
- The dimensions of the image are chosen which will contain the road lanes and mark it as our region of interest or the triangle.
- We need to create a mask, which should same as the dimension of the image which would essentially be an array of all zeros.
- Then we need to fill the triangle dimension in this mask with the intensity of 255 so that our region of interest dimensions are white.
- We need to print the pixel representation of the masked image.
- The binary representation of 0=0000 & for 255=11111111





```
[[   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0   0   0   0   0   0   0   0]
 [   0   0   0   0   0 255   0   0   0   0   0   0]
 [   0   0   0   0   0 255   0   0   0   0   0   0]
 [   0   0   0   0   0 255   0   0   0   0   0   0]
 [   0   0   0   0   0 255   0   0   0   0   0   0]
 [   0   0   0   0   0 255 255   0   0   0   0   0]
 [   0   0   0   0   0 255 255   0   0   0   0   0]
 [   0   0   0   0   0 255 255   0   0   0   0   0]
 [   0   0   0   0   0 255 255   0   0   0   0   0]
 [   0   0   0   0 255 255 255   0   0   0   0   0]
 [   0   0   0   0 255 255 255   0   0   0   0   0]
 [   0   0   0   0 255 255 255   0   0   0   0   0]
 [   0   0   0   0 255 255 255   0   0   0   0   0]
 [   0   0   0   0 255 255 255 255   0   0   0   0]
 [   0   0   0   0 255 255 255 255   0   0   0   0]
 [   0   0   0   0 255 255 255 255   0   0   0   0]
 [   0   0   0   0 255 255 255 255   0   0   0   0]
 [   0   0   0 255 255 255 255 255   0   0   0   0]
 [   0   0   0 255 255 255 255 255   0   0   0   0]
 [   0   0   0 255 255 255 255 255   0   0   0   0]
 [   0   0   0 255 255 255 255 255   0   0   0   0]
 [   0   0   0 255 255 255 255 255 255   0   0   0]
 [   0   0   0 255 255 255 255 255 255   0   0   0]
 [   0   0   0 255 255 255 255 255 255   0   0   0]]
```

- Apply the masked_image on to canny image to only show the region of interest.
- We do this by a bitwise AND operation with the canny image and the mask which will result in our final region of interest.



```
def region_of_intrest(image):
  height = image.shape[0]
  polygons = np.array([
                     [(200,height),(1100,height),(550,250)]
  ])
  mask = np.zeros_like(image)
  cv2.fillPoly(mask,polygons,255)
  masked_image = cv2.bitwise_and(image,mask)
  return masked_image
```

Hough Line Transformation

- This technique is used to detect the straight lines in our region of Interest. Why we need Straight Lines? Straight lines are lanes in which our car is going to travel in-between.
- We know that Straight line in a 2-D graph is represented by Slope : $y = mx + b$. m and b are two intercepts of the equation and if we find change in x and change in y we can get a point that is represented in Hough space. Now in Hough Space we have a point and if we try to draw straight lines, we can draw n number of lines through that point.

THANK YOU