**Digital Image Processing**

CEP Report



**Group Members**

Syed Nazir Shah(020)

AsadUllah(004)

Mohammad Usama(025)

Class: BCE-7A

Submitted To: Sir Ammar Ajmal

Deparment of Computer Engineering

Bahria University­

Islamabad Campus

**Road Lane Line Detection**

**Introduction:**

The goals / steps of this project are the following:

* Make a pipeline that finds lane lines on the road
* Reflect on your work in a written report

The stacks I used are python, OpenCV, and computer vision algorithms. For example, edge detection via Canny algorithm, line detection via Hough Transformation.

**Open CV:**

* OpenCV, or Open Source Computer Vision library, started out as a research project at Intel. It’s currently the largest computer vision library in terms of the sheer number of functions it holds.
* OpenCV contains implementations of more than 2500 algorithms! It is freely available for commercial as well as academic purposes. And the joy doesn’t end there! The library has interfaces for multiple languages, including Python, Java, and C++.
* The first OpenCV version, 1.0, was released in 2006 and the OpenCV community has grown leaps and bounds since then.

**Lane Detection Pipeline:**

1. Convert original image to grayscale.
2. Isolate yellow from HLS to get yellow mask. ( for yellow lane markings)
3. Isolate white from HLS to get white mask. (for white lane markings)
4. Bit-wise OR yellow and white masks to get common mask.
5. Bit-wise AND mask with darkened image .
6. Apply canny Edge Detector (adjust the thresholds — trial and error) to get edges.
7. Define Region of Interest. This helps in weeding out unwanted edges detected by canny edge detector.
8. Retrieve Hough lines.
9. Consolidate and extrapolate the Hough lines and draw them on original image.

**Code:**

import matplotlib.pylab as plt

import cv2

import numpy as np

def region\_of\_interest(img, vertices):

mask = np.zeros\_like(img)

#channel\_count = img.shape[2]

match\_mask\_color = 255

cv2.fillPoly(mask, vertices, match\_mask\_color)

masked\_image = cv2.bitwise\_and(img, mask)

return masked\_image

def drow\_the\_lines(img, lines):

img = np.copy(img)

blank\_image = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

for line in lines:

for x1, y1, x2, y2 in line:

cv2.line(blank\_image, (x1,y1), (x2,y2), (0, 255, 0), thickness=10)

img = cv2.addWeighted(img, 0.8, blank\_image, 1, 0.0)

return img

# = cv2.imread('road.jpg')

#image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

def process(image):

print(image.shape)

height = image.shape[0]

width = image.shape[1]

region\_of\_interest\_vertices = [

(0, height),

(width/2, height/2),

(width, height)

]

gray\_image = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)

canny\_image = cv2.Canny(gray\_image, 100, 120)

cropped\_image = region\_of\_interest(canny\_image,

np.array([region\_of\_interest\_vertices], np.int32),)

lines = cv2.HoughLinesP(cropped\_image,

rho=2,

theta=np.pi/180,

threshold=50,

lines=np.array([]),

minLineLength=40,

maxLineGap=100)

image\_with\_lines = drow\_the\_lines(image, lines)

return image\_with\_lines

cap = cv2.VideoCapture('test2.mp4')

while cap.isOpened():

ret, frame = cap.read()

frame = process(frame)

cv2.imshow('frame', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

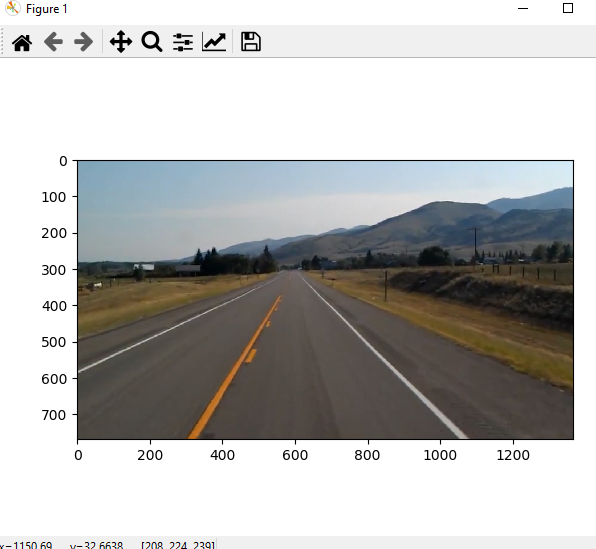
cv2.destroyAllWindows()

**Output:**



## **Convert to grayscale**

Converting original image to grayscale has its benefit. We have to find yellow and white lanes , and converting original image to grayscale increases the contrast of lanes with respect to road.



## **Color Selection**

Here we use OpenCV’s [inRange](https://docs.opencv.org/3.0-beta/modules/core/doc/operations_on_arrays.html?highlight=inrange) to get mask between thresh hold value. After some trial and error, we can find out range for threshold.

For yellow mask:

1. Hue value was used between 10 and 40.
2. We use higher saturation value (100–255) to avoid yellow from hills.

For white mask:

1. We use higher lightness value (200–255) for white mask.

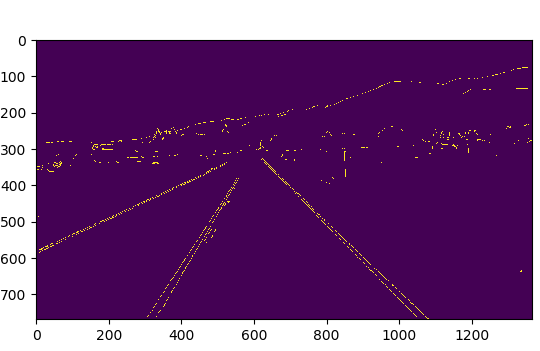
We bit-wise OR both mask to get combined mask.

Below images show combined mask being bit-wise AND with darkened image.

## **Apply Canny Edge Detection**

Now we apply Canny edge detection to these Gaussian blurred images. Canny Edge Detection is algorithm that detects edges based on gradient change. Not that the first step of Canny Edge detection is image smoothing with default kernel size 5.The other steps in Canny Edge detection include:

* Finding Intensity Gradient of the Image
* Non-maximum Suppression
* Hysteresis Thresholding

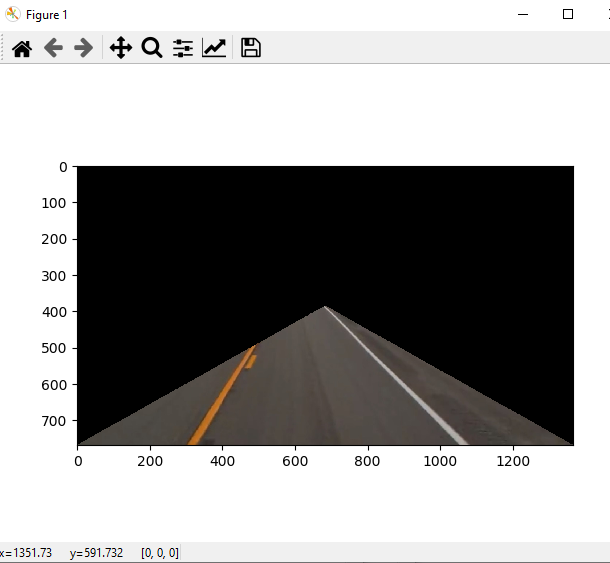


## **Select Region of Interest**

Even after applying Canny Edge Detection, there are still many edges that are detected which are not lanes. Region of Interest is a polygon that defines area in the image, from where edges we are interested.

Note that, the co-ordinate origin in the image is top-left corner of image. rows co-ordinates increase top-down and column co-ordinates increase left-right.

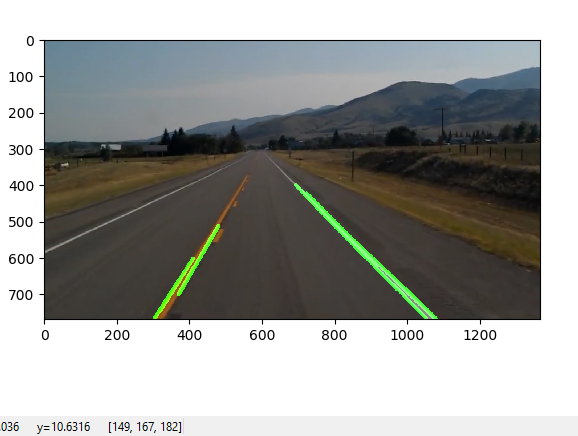
Assumption here is camera remains in constant place and lanes are flat, so that we can “guess” region of interest.



## **Hough Transformation Lines Detection**

Hough Transform is the technique to find out lines by identifying all points on the line. This is done by representing a line as point. And points are represented as lines/sinusoidal(depending on Cartesian / Polar co-ordinate system). If multiple lines/sinusoidal pass through the point , we can deduce that these points lie on the same line.

After finding out Hough lines from Region Of Interest images and then drawn on original images.



## **Shortcomings**

* Hough Lines based on straight lines do not work good for curved road/lane.
* There are many trial-and-error to get hyper parameters correct. Also Region of Interest assumes that camera stays at same location and lanes are flat. So there is “guess” work or hard coding involved in deciding polygon vertices.
* In general, there are many roads which might not be with lane markings where this won’t work.

## **Future Improvements**

* Instead of line, it would be beneficial to use higher degree curve that will be useful on curved road.
* Even when we used information from previous frames, just averaging lanes might not be very good strategy. may be weight average or some form of priority value might work.