ECE 493- Autonomous Vehicles
Assignment 1 – Lane Detection
Shilpan Shah - 20603389

Part 1 – Pipeline

This report covers the pipeline that used to draw lane lines in image/video as the car is driving. The objective of the pipeline is to identify left and right lane that the vehicle is travelling in.

The pipeline is as follows

1. Apply greyscale conversion for easier processing and detecting lanes
2. Apply Gaussian blur to smoothen the images
3. Apply Canny edge detection on the smoothened greyscale image
4. Define a region of interest to remove the lines outside the region
5. Apply the Hough transform to find the lanes within the region
6. Drawing Lines over Lanes

Step 1 – Gray Scaling

First step is to gray scale the image to able to apply other transformations used later in the pipeline. The result is shown below.

Step 2 – Gaussian Smoothing

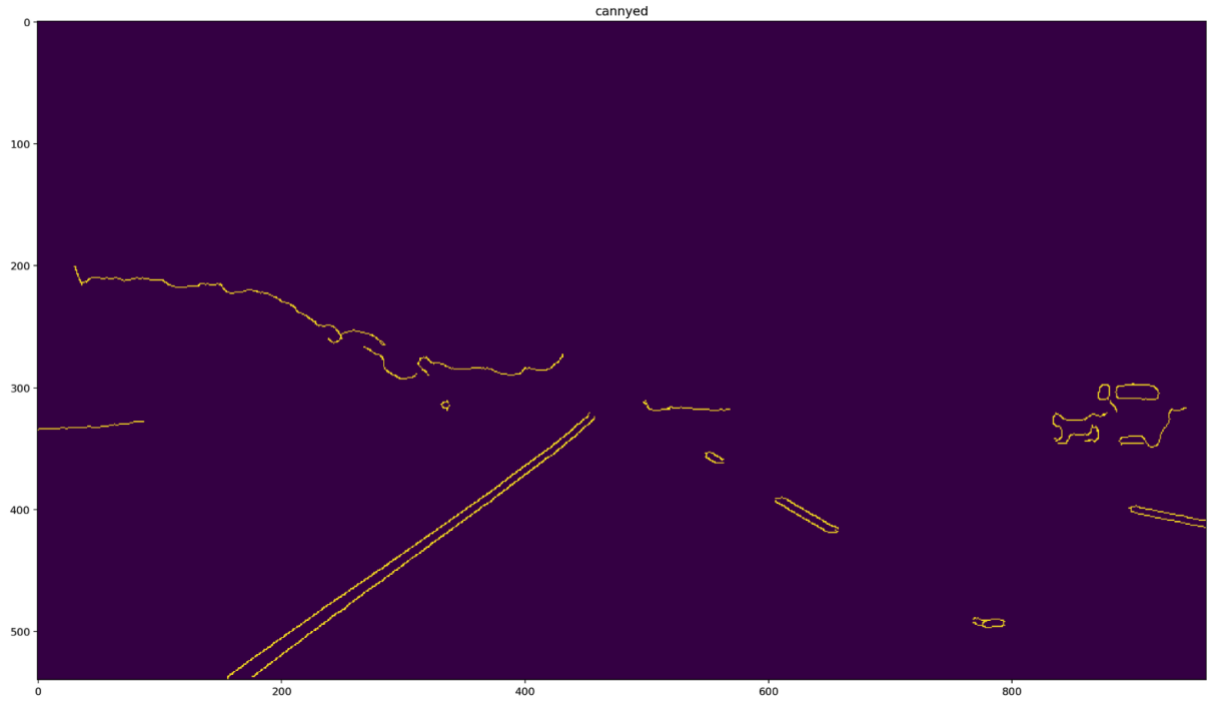
Gaussian smoothing is applied to the greyscale image to reduce noise of the signal within the image. Gaussian filter applies a spatially weighted average across the image for creating a smoothened effect before applying the canny edge detector. The kernel size parameter is tweaked to achieve the best edge detection from canny edge detection step. It is known that the cv2.canny applies its own 5x5 Gaussian filter, however for the purposes of this application is determined insufficient, and an additional Gaussian filter of kernel size 13 is applied. The smoothed result is seen below.

Step 3 – Canny Edge Detection

The gray scaled smoothed image is passed through a canny detection algorithm to determine the lines in the image by finding intensity gradients using horizontal and vertical Sobel kernels. Next, applying a non-maximum suppression to remove any unwanted pixels which may not constitute the edge. Thresholds are then applied to determine potential edges, with use of high and low thresholds for strong and weak edges. Hysteresis thresholds are used to pick up stronger edges by suppressing all the other edges that are weak and not connected to strong edges.

The edges that are above the higher thresholds or within the range of high and low thresholds that are next to edges are kept. The edges with gradients less than low threshold values are considered non-edges and discarded.

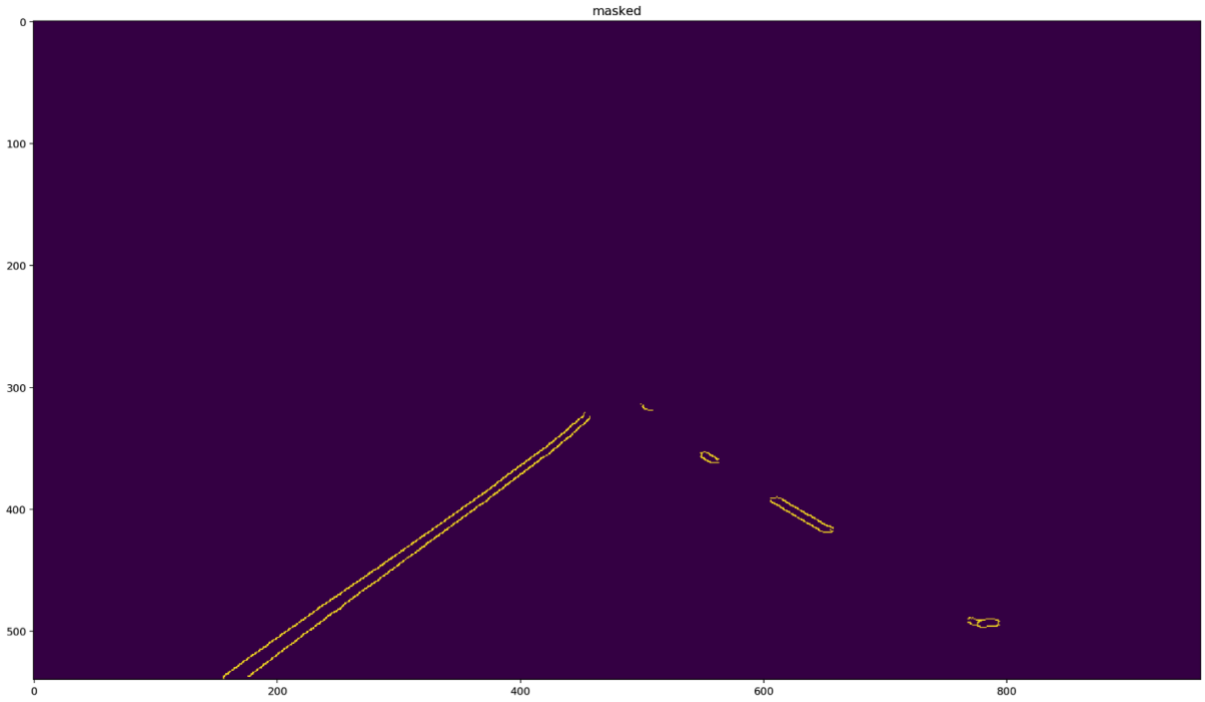
The final canny edge detected result is shown below. The lower threshold is chosen to be 50, and higher threshold is 150.



Step 4 – Region Selection

The region of interest are the points within the defined polygon shape and hence points outside the polygon are removed. The polygon is considered to be a trapezoid shape with wide bottom at the bottom of the image and narrow top in the middle of the image. Selecting a region of interest reduces the lines detected that are sure not to have lanes. This was done using trial and error of finding the dimensions of the trapezoid from the image.

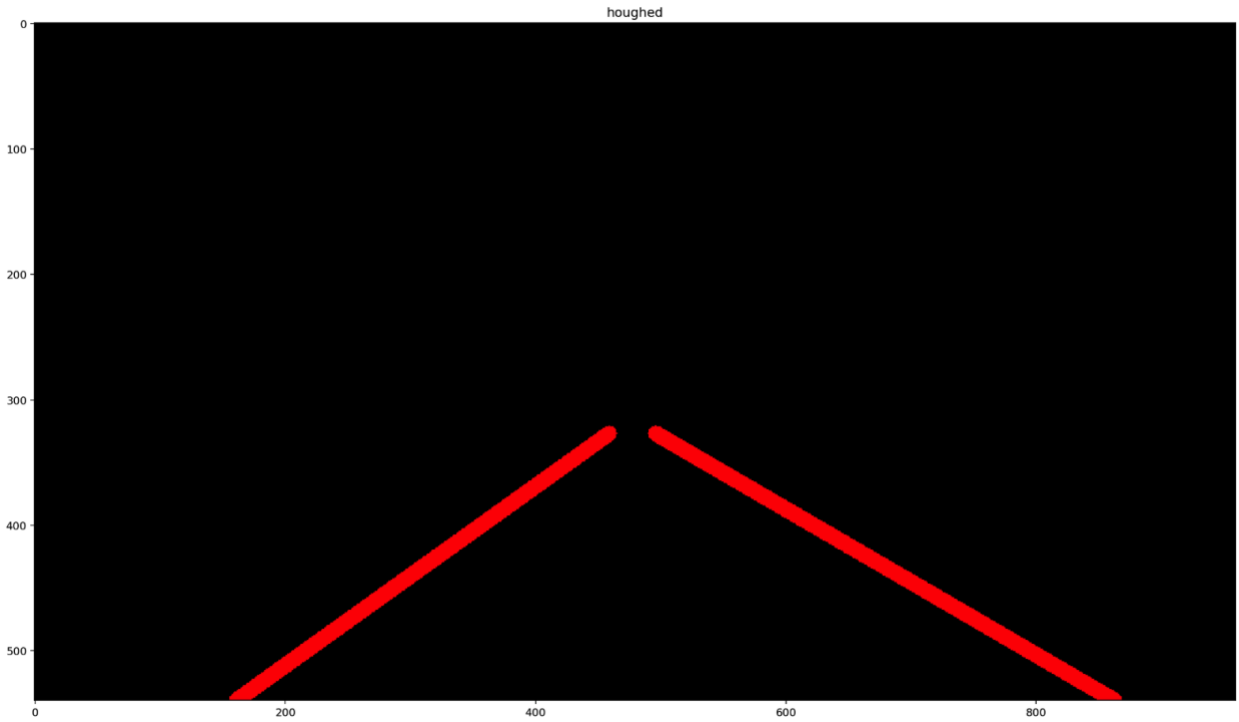
The polygon is illustrated in below.



Step 5 – Hough Transform Line Detection

Hough Transform detects all the lines in the image. It does this by transforming a line equation $y = mx + b$ into the Hough space to represent a line with $\rho = x \cos \theta + y \sin \theta$. Thus, each line is represented as a point (b,m) and the lines that have the same point value corresponding to the same line. Each line is plotted in a ρ vs θ plot and the brightness of areas correspond to the most probable lines. The parameters for Hough transform are chosen to have pixel accuracy (thus rho = 1) and thus the angle array will consist of 180 slots, for angles of 0 to 180. The threshold represents min number of intersection for detecting a line or knows as min vote for a line to be considered. Higher the threshold the less lines will be detected as more points will be needed to get the line. As well, the other paraments *minLinLength* and *maxLineGap* parameters are optimized to achieve an image with least amount of clutter and defined road lanes. *minLinLength* if the min points that can form a line and *maxLineGap* is the max gap between two points to be considered in the same line. These values were tweaked for best results.

The result is shown in the image below.



Step 6 – Drawing Lines Over Lanes

The last step in the pipe line is to combine the result of the Hough Transform with the original image to get an image overlaying the line with image. The Hough Transform call the draw_lines() function which extrapolates two lines (one on the left and one on the right). The draw function first analyzes all the lines passed in from the Hough Transform in terms of their slope. The first stage it to filter all lines that are too big in slope, which would be unrealistic for highway lanes. Next, the lines are split into two parts of the lane (left and right) based on the direction of slope (positive = right or negative = left). After separation by sides, all points in the lines of the left and right sides are used to fit a line through it with an intersection with the bottom of the image, thus necessary to find the y-intercepts. Then, the outliers are found and removed in both the positive set of slopes and negative set of slopes. This is done by finding the 25 and 75 percentile interquartile values and only keeping slopes that are in-between the percentile values in a final array of slopes. Next, the averages of the positive and negative slope and y-ints is found using the mean function. This is an attempt to reduce outliers in the calculation of the slopes that would be used to draw over the lanes, and produce an overall better representation of the lane line. These left and right average slopes and y-ints were used to obtain a linear fit model of the lane lines which are then draw n in red over the original image.

The final result is shown below.


final
0
100
200
300
400
500
0
200
400
600
800

Part 2 – Shortcomings

Possible Shortcoming in the Pipeline:

- If the lane is covered with something, then the lane lines will not be detected. This can happen if a vehicle is changing lanes in front and so no lines will be detected or drawn.
- The effect of shadows and poor lighting will impact the ability of the pipeline to detect lines. This can occur if a tall car or truck is travelling in from of the vehicle and its shadow overlays onto current lane.
- Inability to detect curved lines.

Part 3 – Improvements

Possible Improvements to the pipeline:

- Keep average slope and intercepts of both lane lines so if the line ahead disappears, the line will still be drawn from previous readings.
- The color space can be changed from grayscale to HSV function. This allows to filter colors easier by focusing on hue and saturation (color and color intensity) of the pixel instead of its darkness. Hence, enhance the color in the image.
- Adding function to deal with second or third degree polynomial fit curves for curved lines.
- Better function to detect and kill outliers.