# Assignment 1 - Lane Detection

ECE 493
Sarim Hasan Zafar
20511043 - shzafar@edu.uwaterloo.ca

## Part A

The pipeline function takes in each frame and processes it to locate the lanes. The steps below provide an overview of how the function is able to locate the lanes and return an image with both the left and right lanes drawn on them.

```
Step 1 : Copy the original image as to not destroy any original data
Step 2 : Define a region of interest in the shape of a trapezium
Step 3 : Extract the region defined in Step 2
Step 4 : Use color filtering to generate a mask with only white and yellow colors
Step 5 : Apply Gaussian filter to this mask to remove any noise
Step 6 : Use Hough Transform with defined parameters to extract lines in the image
Step 7 : Use filtering to remove all the lines that are not part of the lane
Step 8 : Use linear regression to find the extrapolated lines
Step 9 : Draw the lines over the original image and return
```

### *Defining the ROI*

Region of interest (ROI) is an approximated area of the frame where the lanes are most likely to be detected. Selecting an ROI before performing any vision based operations ensures removal of unnecessary local features and minimizes the search area for faster computation. The ROI region is an approximation and has been calibrated separately for each window. It is to be noted that the ROI region can be calibrated based on the camera location in the vehicle. The figure below shows the original image and the following image containing only the ROI. In the following frame the ROI has been cropped down to $bottom_{left} = (193, 513)$ , $bottom_{right} = (768,\ 513)$, $top_{right} = (672,\ 324)$ , $top_{left} = 288,\ 324$

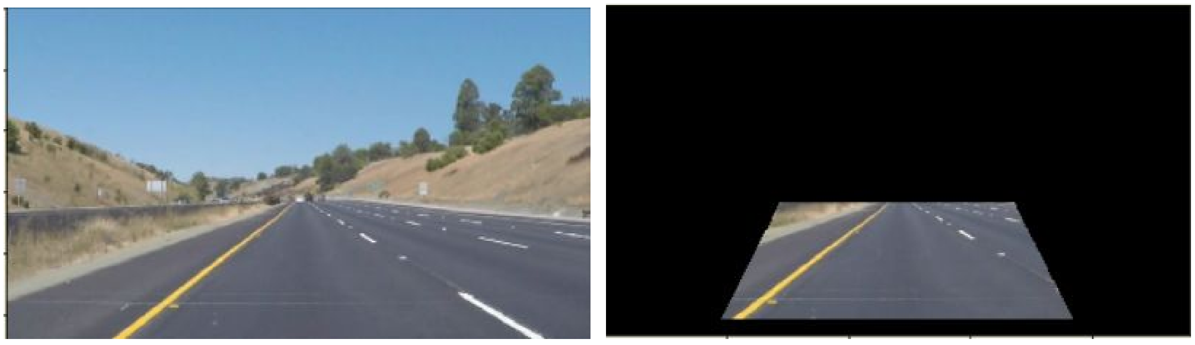
Figure 1 : Original image on the left. Image on the right only shows the ROI

### *Removing noise using Gaussian Blur*

The image must first be smoothed in order to maintain a unique consistency throughout the image. Thus, a Gaussian Blur (size = 5) is used to remove any noise from the frame.

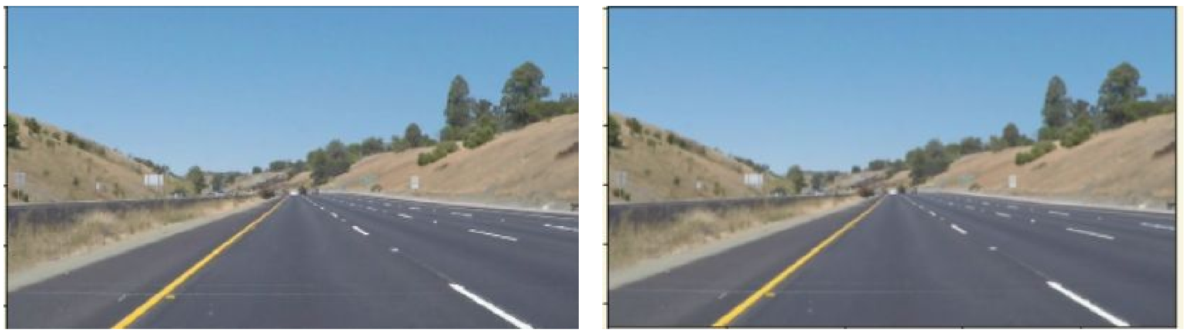
Figure 2 : Original image on the left. Smoothed image on the right using gaussian filter

### *Generating the Color Filtering Mask*

In order to filter the image based on colors the denoised image first had to be converted from the RGB space to HSV space. After which a mask for yellow color was created with a lower and an upper threshold in the HSV space as follows $lower_{yellow} = [20,\ 100,\ 100]$, $upper_{yellow} = [30,\ 255,\ 255]$. A separate mask for the white color was created by first converting the image to grayscale and then using the lower and upper thresholds of $lower_{white} = 220$, $upper_{white} = 255$. The two masks were then combined together using the bitwise OR operation and the resultant mask was then bitwise AND with the grayscale image to generate the filtered mask. As can be seen below the filtered mask only contains colors white and yellow from the original image.

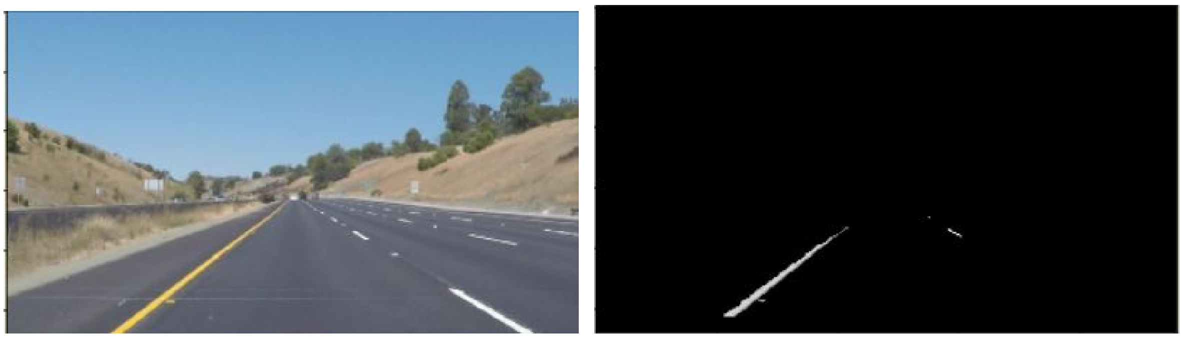
Figure 3 : Comparing the original image with the color filtering mask of yellow and white

As can be observed from the binary mask above that it only contains pixels which are either yellow or white in the original image. Using color scales to filter out lanes is both fast and highly accurate. The process is also robust against shadows and changing intensity levels of the frames.

## *Using Hough Transform to detect lines*

Hough transform is a technique that is used to detect lines in a binary image. Hough transform maps each point of interest (white pixels in the binary image) from cartesian space to angle-distance parameter space, given by the equation $\rho = x.cos(\theta) + y.sin(\theta)$. It then looks for all intersections in the hough space thus indicating collinear points. Probabilistic Hough Transform is an inbuilt function provided by OpenCV which is an improvement on the classical hough transform. It has 5 main tuning parameters. The first two parameters $\rho$ and $\theta$ are distance and angle resolution of the accumulator in pixels and radians respectively. These resolution values are the incremental discretization of the entire Hough Space defined by the dimensions of the image. Selecting a smaller resolution would mean that each point of interest has to be evaluated over a greater number of points which would drastically reduce the speed of the algorithm, while selecting a larger resolution would mean skipping over important large amounts of data. In the algorithm $\rho = 5$ and $\theta = \pi/180$ have been used.. These values of $\rho$ and $\theta$ ensure that step value does not exceed any valuable points of interest while maintaining a decent runtime speed.
The next three parameters are threshold, minimum line length and maximum line gap. These parameters can be customized to remove noise from the image as well as ensuring that only lines with certain criterias are selected. After much tuning the threshold value was selected to be 7. This means that only lines with threshold value greater than 7 votes would be included as a line. A higher threshold means less number of lines, while a lower threshold does not get rid of noise. Hence, it was necessary to choose a median value as to not exclude any necessary line information.  The minimum line length parameter has been tuned to only select lines with a certain number of pixels. Since, the lanes in a video are pretty much consistent it is easier to define this parameter based on the previous knowledge from the frames. Similarly, lane knowledge has also been used to set the maximum line gap parameter. This parameter ensures that only lines of certain length are selected. The selected parameters are as max_line_gap = 8 and min_line_len = 10.

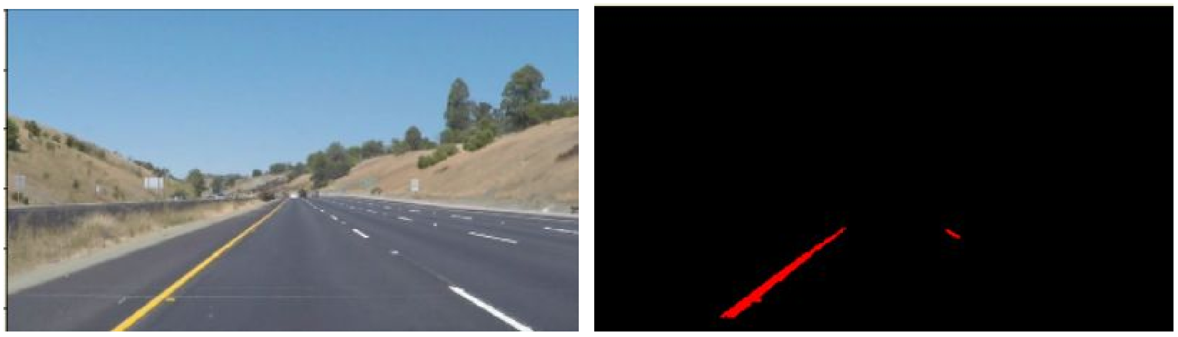
Figure 4 : Original image on the left. Hough Lines drawn on the right in red.

## *Filtering and Extrapolating lines*

As can be seen from Figure 4 that the lanes have been detected yet they have to be extrapolated in order to represent them correctly. The lines were extrapolated using linear regression. In order to do so, all the lines from the hough transform containing a start and an endpoint were collected together and a polyfit algorithm was run through them. Lines which

were parallel to either the X or the Y axis were filtered out as they would skew the regression results. Similarly, all the lines on the left side of the image but with a positive slope and all the lines on the right side of the image with a negative slope were removed from the data points. This was done to ensure that these slopes do not skew the final results. Once the start and end points of the line was found, its slope was computed. If the slope was negative, then all the points on the left side of the midsection of ROI were collected together. Similarly, if the slope was positive then all the points on the right side of the midsection were collected together. This filtering reduced the number of error points in the final data set.
After the data set was built, a linear regression algorithm was run and the slope and intercept of both datasets on either side was calculated. The slope was then used to find the X coordinates at two different Y points, and a line was drawn.

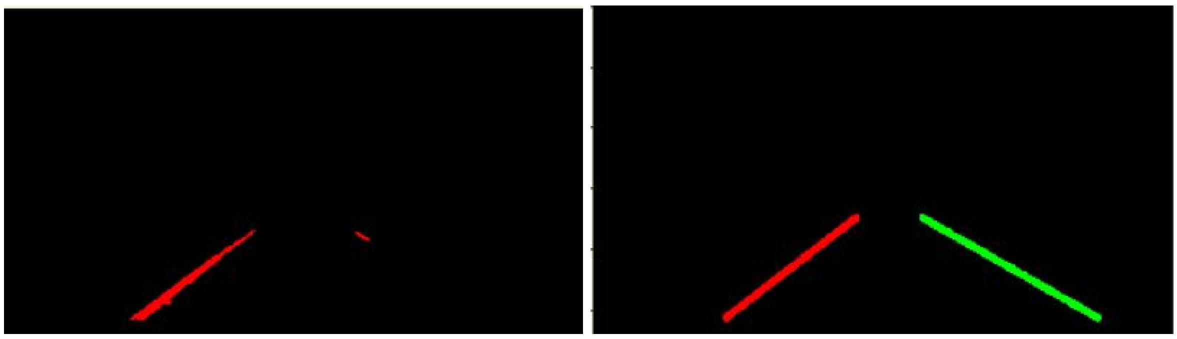
Figure 5 : Output from Hough Lines on the left. Extrapolated lines on the right

### *Drawing over the actual frame*

As a concluding step, these lines were then drawn on top of the actual frame to mark the left and right frames. This was done using the *weighted_image* function provided.

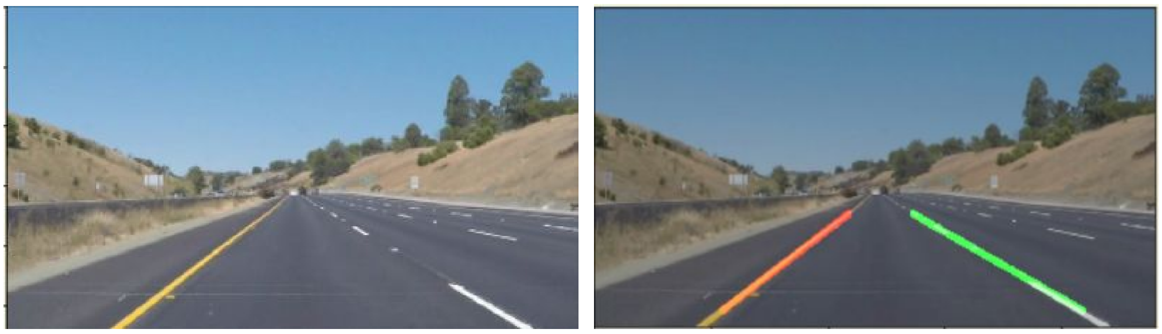
Figure 6 : Original image on the left. Output image with marked lanes on the right.

## Part B

The following shortcomings were observed during the entire process -

1. Manually defining the region of interest - The ROI in each of the test images and videos were manually calibrated. This is a major shortcoming as the ROI needs to be adaptive in nature, since it cannot be readjusted based on every vehicle, scenario or data set.

2. Using color masks to detect lanes - There is a major shortcoming in using color masks when detecting lanes which is the visibility of these lanes during different weather and light conditions. Colored lanes can easily be detected in clear and bright conditions but can be harder to find during snow, rain or cloudy weathers. Another problem is using color detection at night, and especially in low-lit areas where the color values would change drastically based on the lighting conditions.
3. Using Linear Regression for extrapolation - Linear regression is a fast method for extrapolation but it heavily depends on the previous schemes for lane detection. If the dataset is corrupted with noise as in the case mentioned below, the linear detection fails to correctly extrapolate the lines. The noise in the frame arises due to the detection of surrounding lanes and the white car which significantly bias the final slope.

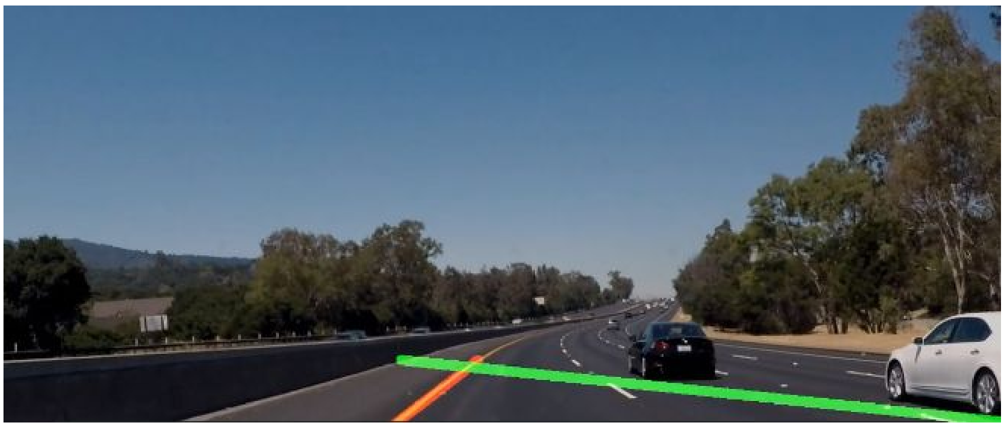
Figure 6 : Right lane is highly skewed due to the presence of noise in the frame

## Part C

The following improvements can be made to the algorithm -

- Implementing adaptive region of interest for lane detection. Using information such as road edges and surrounding vehicles can be used to define a region of interest.
- Using Deep Learning for detecting lanes in an image. Such algorithms are robust to different conditions and do not rely heavily on classical methods of computer vision.
- Using improved data fitting algorithms such as RANSAC instead of Linear Regression. RANSAC most importantly gets rid of outliers and would avoid bias issues as shown in Figure 6.