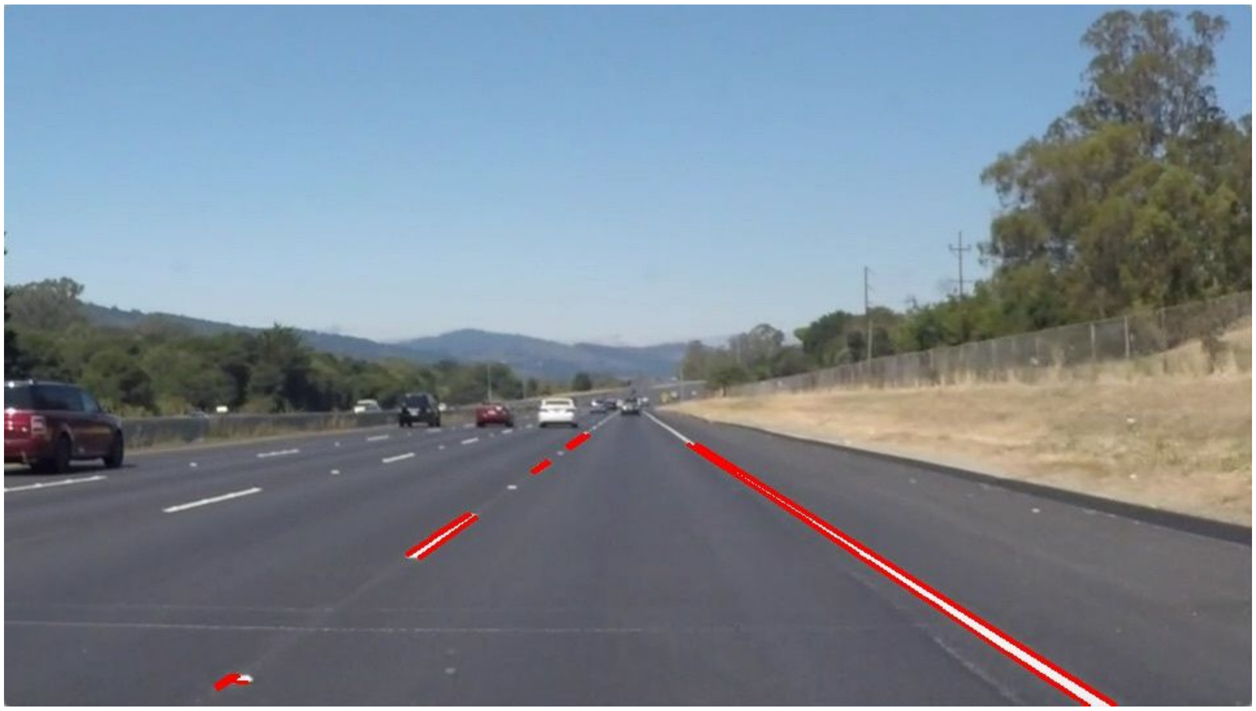
Self Driving Car Engineer Nanodegree
Udacity

Manikandan R
vasumaniram@gmail.com

# Finding Lane Lines On Roads - Project 1



## Objective

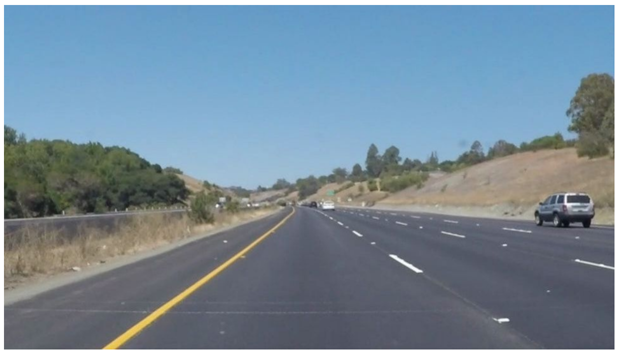
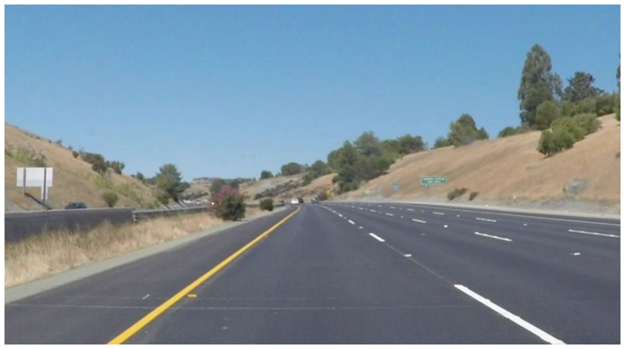
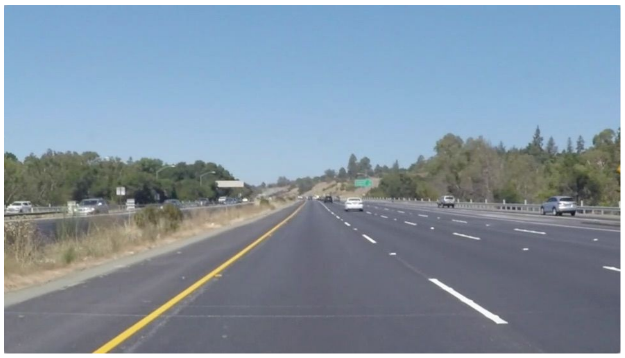
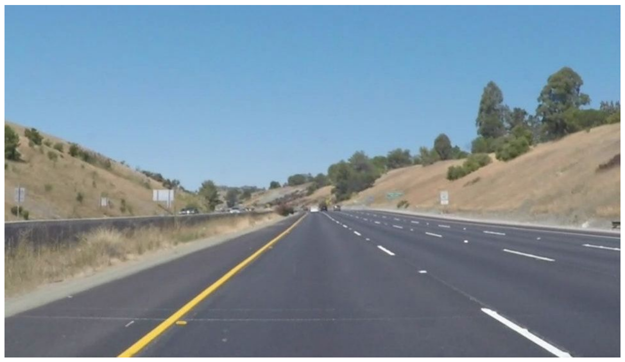
To define a Computer Vision pipeline to detect the left and right lane markings on the road images taken from the front camera of the Car. The same Computer Vision pipeline will be leveraged to detect the lane lines in the video which are the series of road images. The detected lines should be extrapolated to make a single lane line.

### Assumptions

- The right and left lanes will be approximately within the same region in every image as its taken from the front camera of the Car.
- Every image is a color image and will have three color channels (RGB).
- Both right and left markings are visible on every image.

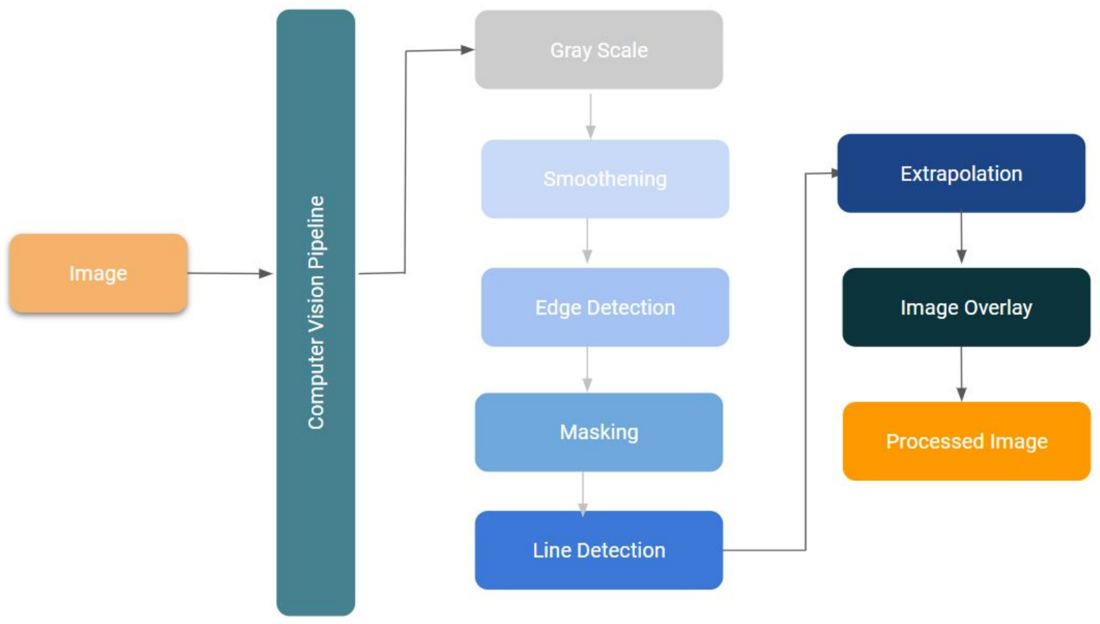
## Input Images:

The input images have both white and yellow lines and both left and right lanes have the discreate lanes.

## 1.Computer Vision Pipeline:

The following image depicts the abstract Computer Vision pipeline used to detect the lane lines on the road images.



Picture 1.1 - Computer Vision Pipeline

The pipeline comprises seven steps to detect the lane lines from the given image.

1. **Gray Scale:** For most of the Computer Vision tasks the color image is not required as those three color channel information are very compute intensive and will not be useful either for most of the use cases. Hence the input image will first be converted into a grayscale image.
2. **Smoothening:** Image smoothening is a preprocessing step which takes care of removing the noise from the image. Have used Gaussian Blur to remove the noise. Gaussian Blur convolves the image with a low-pass Gaussian Kernal. One caveat is, If the image is a low resolution one, applying Gaussian Blur will make it more blur and may not be useful for further processing.

3. **<u>Edge Detection:</u>** Canny Edge Detection has been used to detect the edges of the image. Edge is a drastic change in the pixel values in an image. For example, the white lane lines on the black road will have a huge change in pixel values between the black road and white lane. The edges will be detected based on any color / brightness change in an image. For the Canny Edge detection algorithm, we have to pass the grayscale image and low and high thresholds.All the pixes above the high threshold are considered for detecting edges and all the pixes below the low threshold are rejected. The pixels between the low and high thresholds are used to connect to the strong edges detected by the algorithm. The output of the algorithm is a set of strong edges detected in white on a black background.
4. **<u>Masking:</u>** The objective is to find the left and right lanes from the given image. So its most appropriate to find our Region Of Interest(ROI) in an image and remove the other parts. This is called Image Masking. There are several types of masks we can define based on our requirement. For lane detection, we have used a polygon with four pairs of (x,y) points to define the Regione Of Interest(ROI). The output of the masking is the image with only the Region Of Interest(ROI)
5. **<u>Line Detection:</u>** The next step will be to identify the lines from the Region Of Interest. Hough transform helps to identify the lines from an Image. A line is represented in Image Space as $y = mx + b$ where $m$ and $b$ are the constant coefficients and $m$ being the slope of the line and $b$ representing the point at which the line intersects $y$ when $x$ is zero. In Hough Space, the constant coefficients $m\ and\ b$ are represented instead of $x\ and\ y$. So a line in Image Space will become a point in Hough Space and a point in Hough Space will become a line in Image Space. Voting is the technique to help find the continuous points forming a line. In Hough Space, if $m\ and\ b$ points are intersected by a line, its counted as one vote. If more number of lines are intersecting at $m\ and\ b$, the more votes it gets to be identified as straight line in Image Space.
6. **<u>Extrapolation:</u>** From the videos, either the right or left lane has discrete lines. Based on the detected lines, need to extend the it as a single line. To extrapolate the lines, first need to categorize the lanes into right or left lanes. To achieve this, the slope can be calculated for the lines and based on the slope value, the

categorization could be done.  The image coordinates are placed upside down and the origin of the image starts at the top left corner.

***Left Lane:*** When the $x$ value is increasing the $y$ value is decreasing and hence would result in negative slope values.

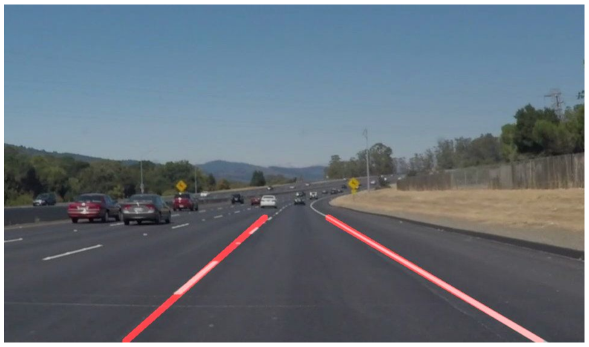
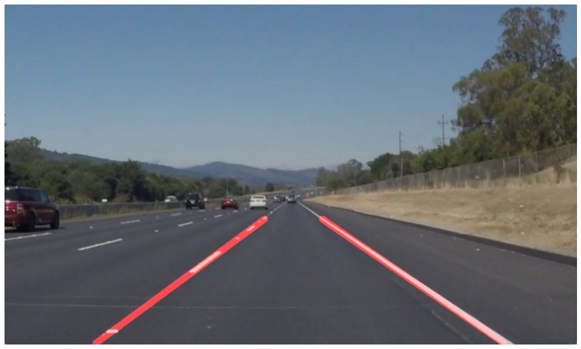
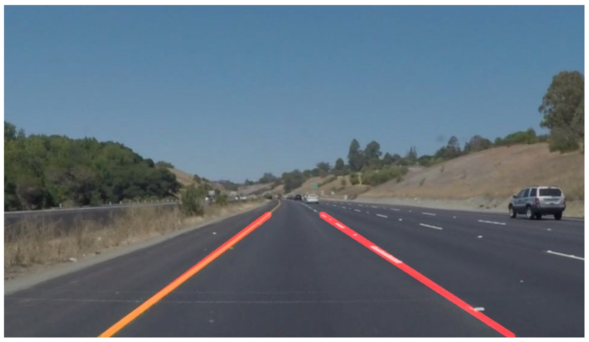
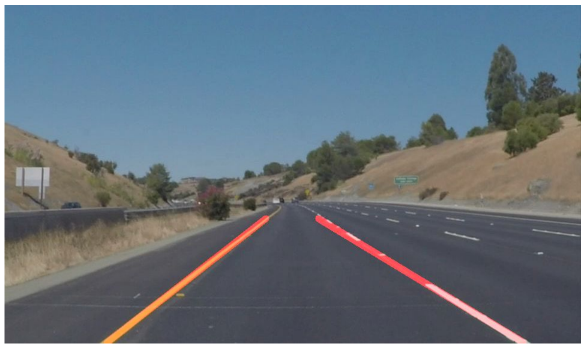
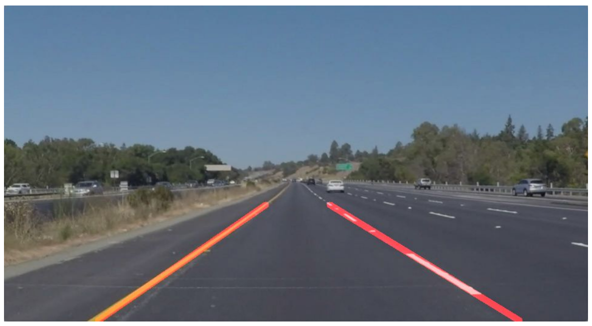
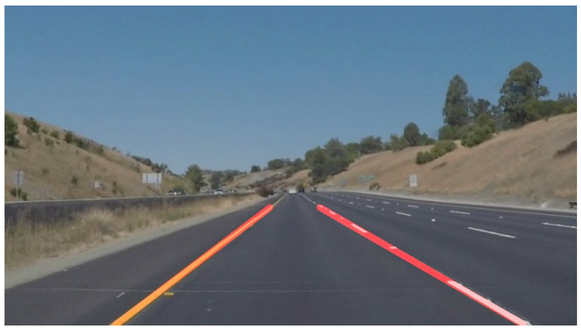
***Right Lane:*** When the $x$ value is increasing the $y$ value is also increasing and hence would result in the positive slope.

If the detected lines have a very minimal slope, those lines could be left off for extrapolation. Based on the left and right lane categorization, the $x$ and $y$ coordintates would be collected for all the lines and would fit them into forming a line equation $y = mx + b$ to further draw a single line.

7. **Image Overlay:** The final part of the pipeline draws the thick lines on the image with the extrapolated line coordinates to make it look like a straight line. Hough Transform will return an empty image on which the detected lines are drawn. Image overlay will take the original image and the Hough Transform returned image and will return the thick straight lines drawn on the input image.

## Output Images:

The following are the output images of the Computer Vision pipeline, marked with left and right lanes.

## 2.Potential Shortcomings In The Pipeline:

- While running the pipeline on the challenge video, it failed to detect the lane lines when,
    1. There is a heavy sunlight laying on the road and the road was not as dark as compared to other frames of the video
    2. When there is a curve in the lane line
- A generic way to mark the Region Of Interest points to detect the lane lines.

## 3.Possible Improvements For The Pipeline:

- Exploring different color spaces to get the edge detection of the lanes in case of sunny or less black roads.
- The extrapolated lines function should also consider the degree of a curve to better fit the lines.
- The frames(especially from Challenge video) which are not detected with lane lines to be taken out for further analysis to find the exact root cause and the improvements could be tested on those frames first.
- Detecting the lanes based on the width of the road which would help to drive even the roads are missing lanes in the road. One possible approach is to understand the length of the road from the HD maps and do the calculations based on it to draw virtual lanes on the Image.

## 4.Source Code And Output Video Links:

**Github Repo** : https://github.com/vasumaniram/SDCND-P1

**Youtube Links :** *White Lanes* - https://youtu.be/GsGiIbnoLi0

*Yellow Lanes* - https://youtu.be/cQOcU0C-ZZw

*Challenge* - https://youtu.be/3b3YJ8xrnGo