Advanced Lane Detection

*Udacity Term - 1*

*Period October to January*

## System & Software Specification

OS - Windows 7

Hardware: Intel  i7 core CPU

Programming language: Python 3.x

 Python Libraries used:

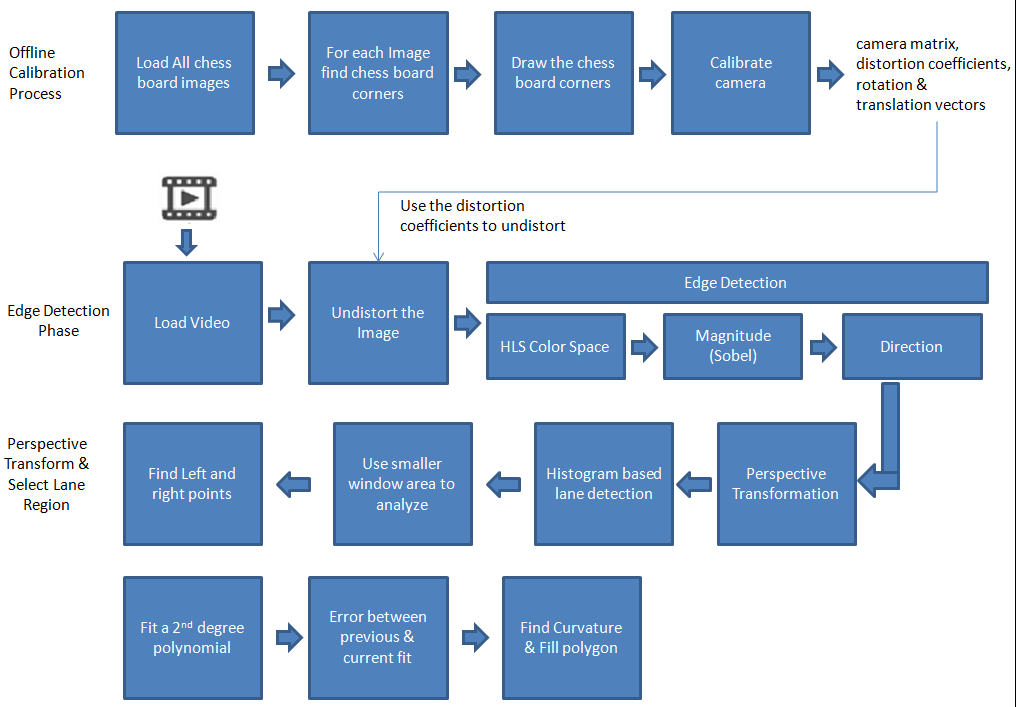
* OpenCV:  library name "cv2" . Used for image processing
* Numpy: Array related functionality
* Matplotlib: used for plotting images
* Math: Used for finding square roots
* Glob: To load images from folder
* moviepy.editor: For loading and creating a video

## Description

Objective: Goal of the project is to calibrate the camera with the images proved by Udacity. Undistort the frames in the video. Apply HLS color conversion and find the edge gradient and magnitude and direction. Apply Perspective transform and find the lane path. Find the lane curvature and print the same on the video.

## Approach

The overall flow of the program is as shown below



### Calibration

The Image provided by udacity is used for calibration of the camera.

The object points are created based on the size of the chess board. [6X9]. All the images in the calibration folder are loaded. For each image the chess board corners are identified using the OpenCV function, cv2.findChessboardCorners. We append these to the object points and find the edges on the image and draw them using cv2.drawChessboardCorners. We append the corners to the imagepoints .

We calibrate the camera using cv2.calibrateCamera .

We get the camera intrinsic parameters, the distortion coefficient, the rotation and translation matrix.

The **camera matrix** is

[[ 1.15694047e+03 0.00000000e+00 6.65948820e+02]

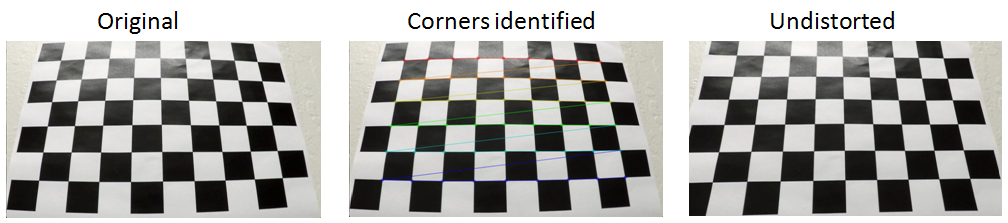
[ 0.00000000e+00 1.15213880e+03 3.88784788e+02]

[ 0.00000000e+00 0.00000000e+00 1.00000000e+00]]

The **Distortion coefficients** are

[[ -2.37638054e-01 -8.54042100e-02 -7.90999575e-04 -1.15882283e-04 1.05726077e-01]]

Sample of the calibration is as shown below.



### Loading Phase

The Video files are loaded using the opencv python library module “moviepy” . All the frames are passed to a function “processFrame” for processing.

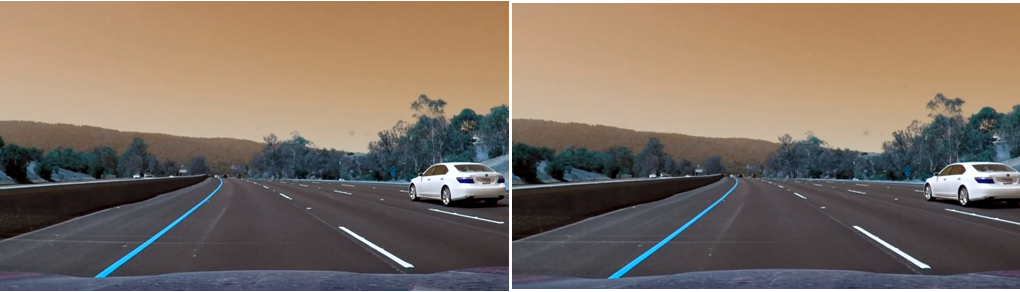
### Pipeline

The below sections describes the different stages in the pipeline.

#### Distortion Correction

The camera matrix and distortion coefficients, identified in the offline process is applied onto each frame in the video.

The OpenCV function cv2.undistort is used for correcting the distortion.



#### Edge Detection

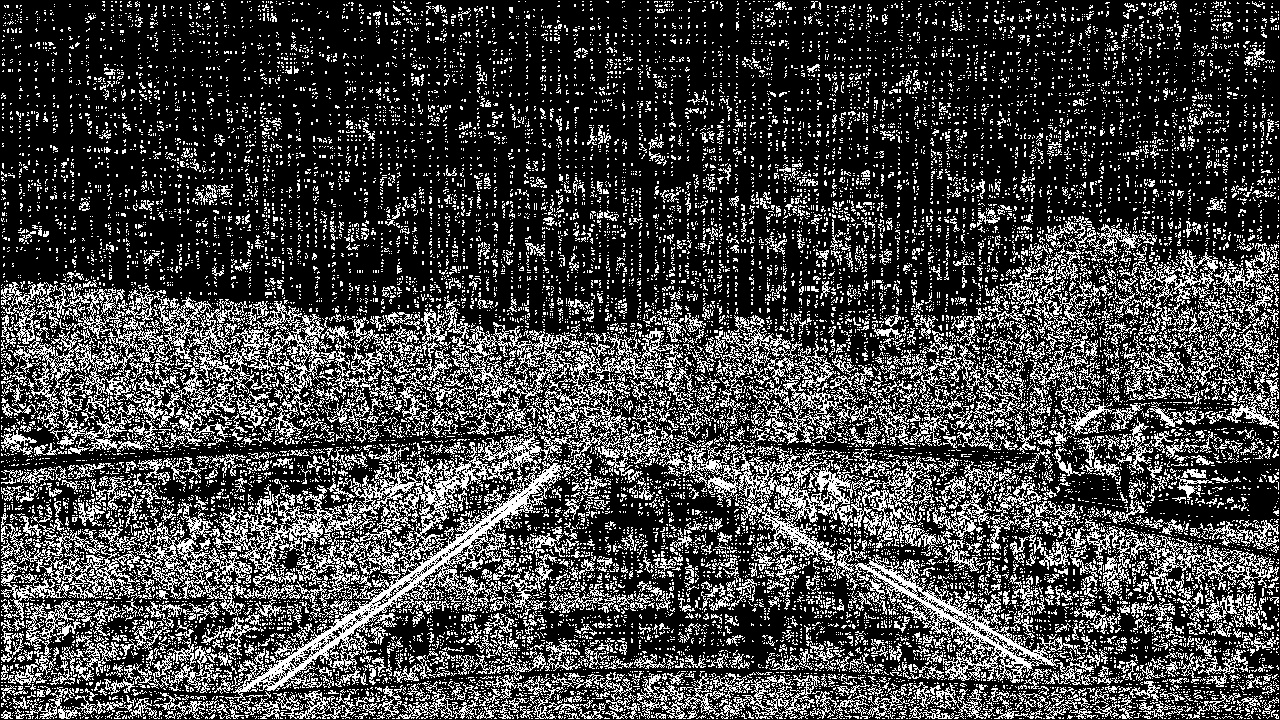
We shall identify the gradient, its magnitude and direction to detect the edges using the sobel operator. The HLS color space is also explored to segregate the yellow and white lanes.

We find the magnitude of the edge gradient using soblel operator.

The output of the Magnitude is shown as below



We find the direction of the gradient and a sample of the output is as shown below



We also convert the frame to HLS space and binaries the same using suitable threshold.



#### Perspective correction

Once the edges are detected the perspective transformation is done.

We identify the transformation matrix and its inverse using the OpenCV function

cv2.getPerspectiveTransform

The source and destination points are as below

Source points (587, 446), (153, 673), (1126, 673), (691, 446)

Destination points (200, 0), (200, 720), (1080, 720), (1080, 0)

The perspective transformation matrix and its inverse are derived and are as below

**M** =

[-0.50772, -1.49582, 951.33 ],

[-3.21965e-15, -1.98816, 914.554],

[-4.98733e-18, -0.00238604, 1 ]

**Minv** =

[0.192187 ,-0.766859 ,518.5],

[1.77636e-15 ,-0.502977 ,460 ],

[-1.73472e-18 ,-0.00120012 ,1 ]



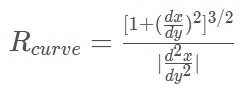
#### Lane Detection

Using the warped image, we use the histogram method to identify lanes. The histogram of the half of the image is taken and peak of the histogram gives us the lane position.

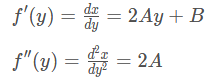
We use sliding window method to identify the pixels of lane.

Using the above, the left and right points are identified and a 2nd order polynomial curve is fitted. And the curvature is estimated.

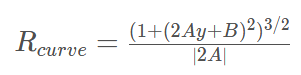
The radius of curvature at any point x of the function x=f(y) is given as follows:



In the case of the second order polynomial above, the first and second derivatives are:

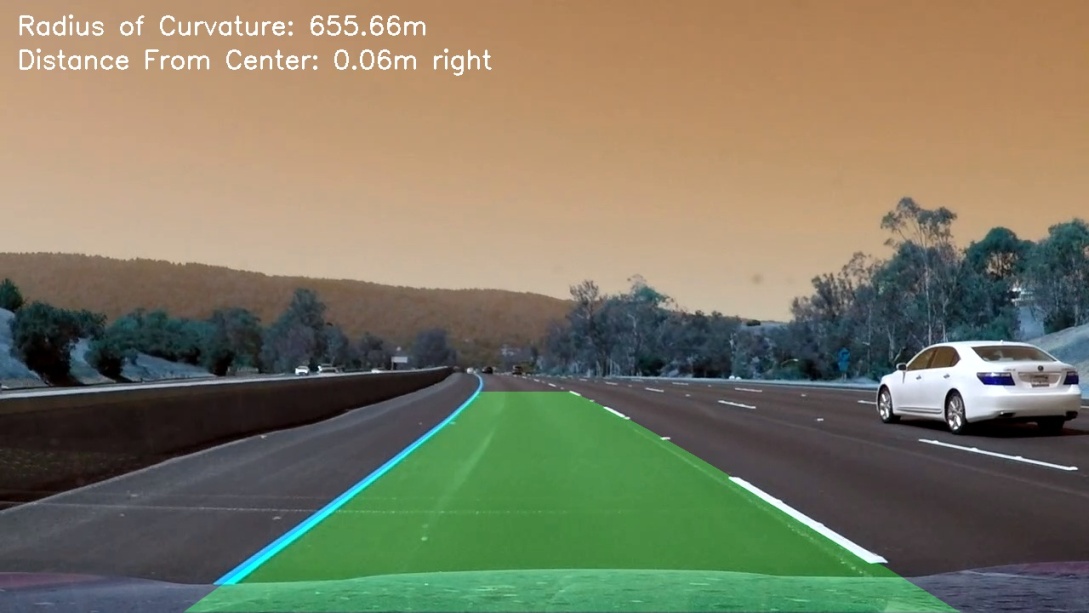


So, our equation for radius of curvature becomes:



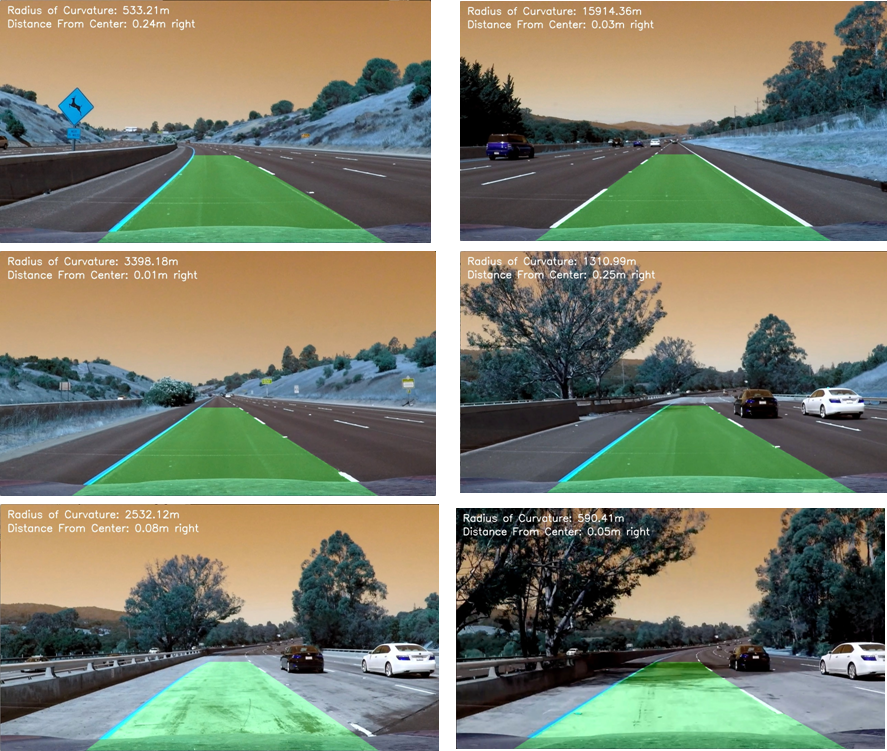
From frame to frame sometimes the lane are not smoothly seen. In order to make the lane more smooth, the current line fit is compared with the previous fit. If the mean error is greater then 0.01, then the previous line fit is taken as 75% weight age

We create blank image and draw the lines on to it. And fill the polygon. We unwarp the image and draw it back to the original image, using the inverse perspective transformation matrix.



## Test Images Output

The output of the pipeline on the test images are shown below





## Improvements

The lane detection may need lot of changes to work for other environmental conditions. We need to also look at places where there are curves in shape of ‘S’.