Advanced Lane detection

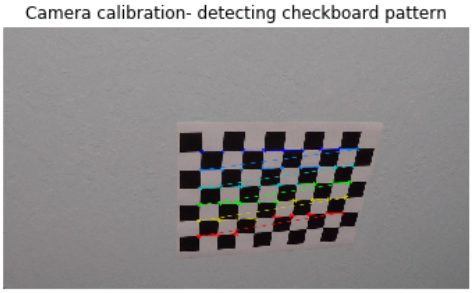
by K.Vikraman

The goals / steps of this project are the following:

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

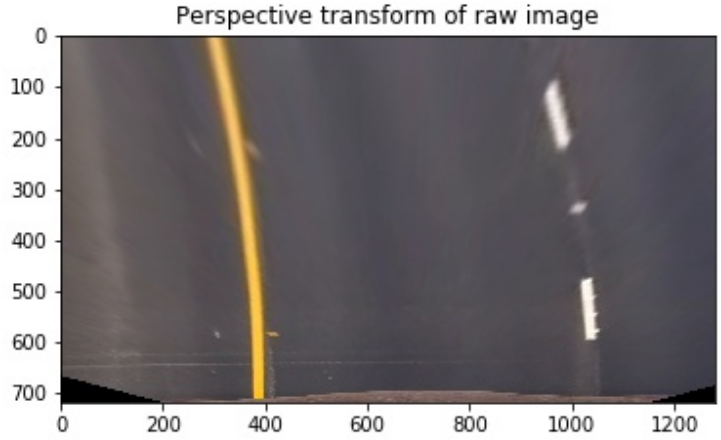
Step 1: Camera Calibration and undistortion

An image from a camera inherits some errors due to the curved shape of lenses. Hence they need to be calibrated. I worked with images of chessboard to calibrate since it is easy to find corners and a chess board provides a flat image.
First I located the corners of the image:



Using the information of corners, I stored the object points in an array (size :6*9,3). With the object and image points I calculated the transformation Matrix to undistort the images

After callibrating, I applied perspective transform to have a bird's eye view of lane :

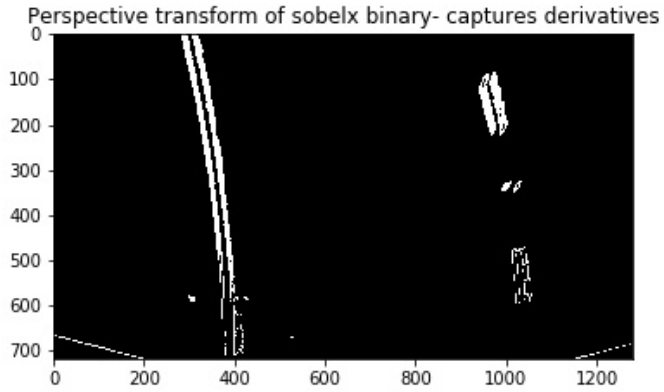


Step 2: Choosing essential features: Color and gradient threshold

Each color channel and gradient provides valuable information about the location of lane pixels. I experimented a lot of color channels and I am discussing the useful features here.

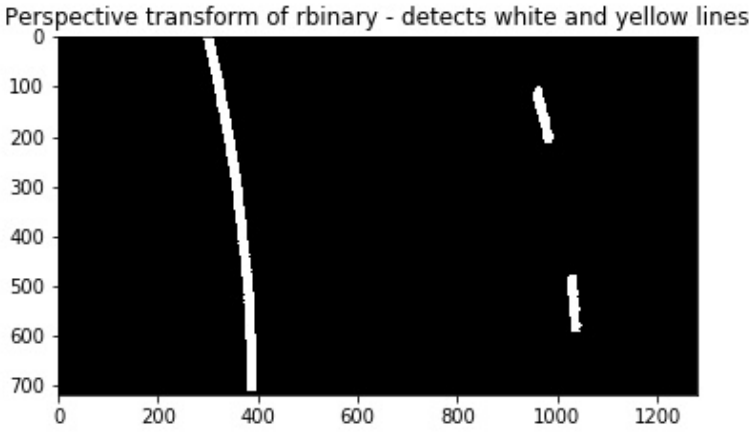
Gradient threshold:

There is abrupt change in the intensity of pixel values in lanes, and hence gradient threshold is useful in identifying lane pixels. However in poor lightning conditions, like shadows, the gradient values are high. In order to address this issue, I carefully chose the threshold values.



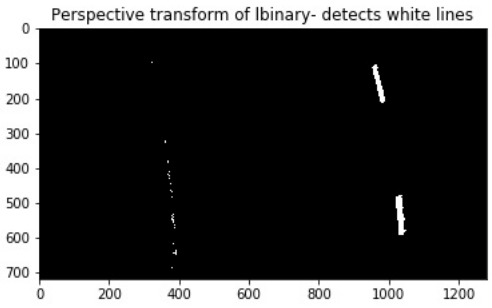
R-Channel :

One of the major components of yellow color is red. Hence to identify yellow lanes, choosing R-channel pixels is useful. In order to remove background noise, I chose threshold values to restrict yellow color in different lighting conditions. In addition, it also detects white lane pixels because white is made of equal values of red, green and blue.



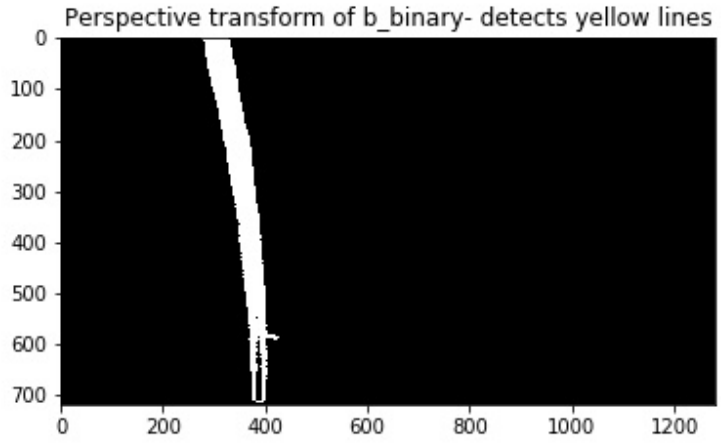
L-channel:

L channel from LUV colorspace helps to detect white lane pixels. Hence, I choose 'L' channel values.



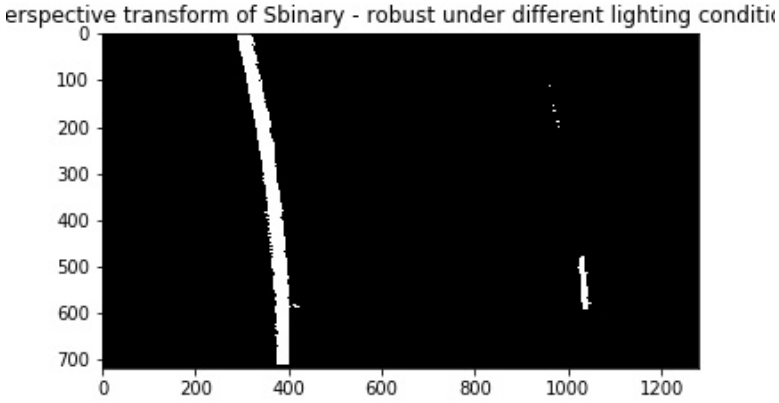
B- Channel :

'B' Channel from Lab colorspace identifies yellow lines with a great accuracy. Hence, I chose 'B' Channel values.

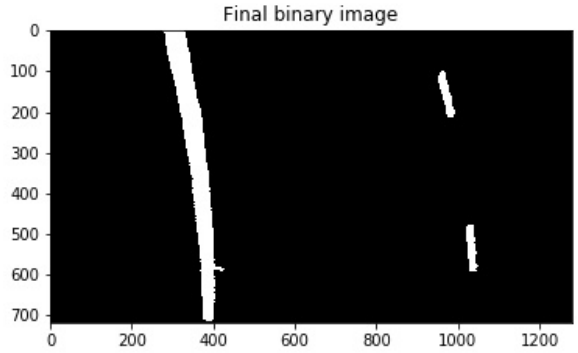


S- Channel :

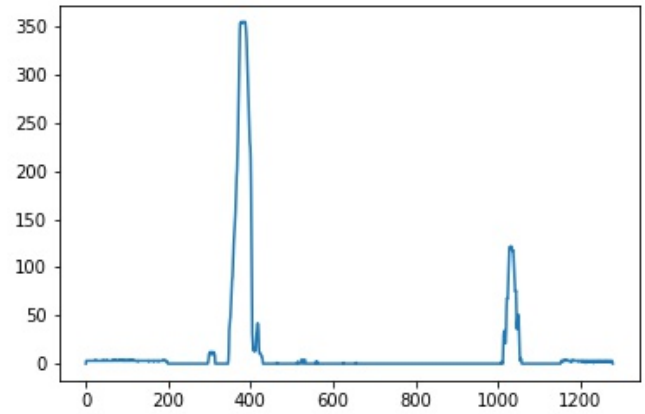
In real life scenarios, we are likely to encounter shadows and darker situations especially during night. To address this issue, I used Saturation values from 'HLS' colorspace. It is robust in identifying lane pixels in different lighting conditions.



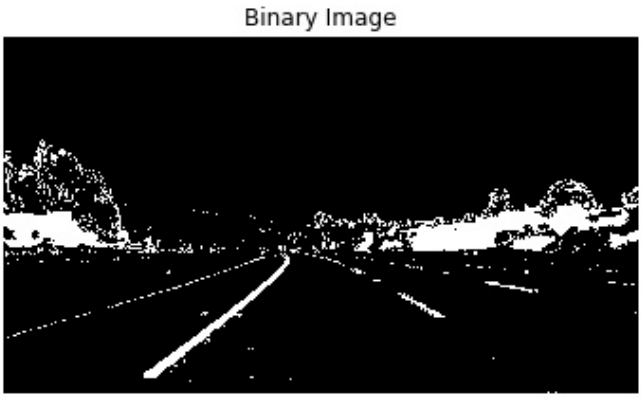
Final output :
The following is the final output after including all the above features



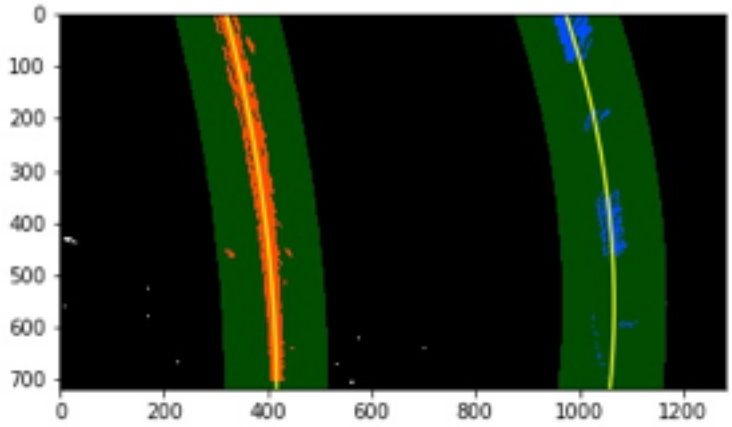
The corresponding histogram :



Binary image of the entire image:



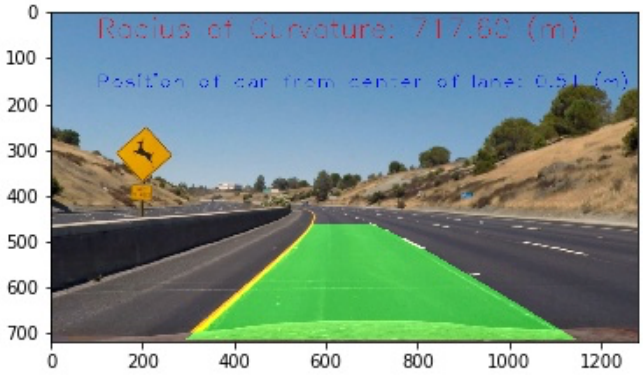
Step 4: Detecting Lane Pixels:
From the histogram we know the approximate locations of lanes. We use sliding windows(in a vertical manner) to find the pixel positions in our image and use polynomial fit to find the approximate second degree polynomial and hence find the lane



Step 5: Find Radius of curvature and off center :
The final step is to calculate radius of curvature using a standard formula for both the lanes. The radius of curvature of the road is the average of radius of left and right lane. I have evaluate the value of Radius of curvature at the bottom of the image which corresponds to maximum value of 'Y'.
Offcenter:

The deviation of the vehicle from the center of a lane can be easily by subtracting the position of car (center of the image) from the position of lane's center(average of position of left and right lane).

Final output



Discussion :

- Since this method relies on manual tuning of threshold parameters, it might fail in some unseen conditions like city trails where the distance between two cars is small and lanes can not be identified and harsh weather conditions when windshield is switched ON.
- Another instance this method might not work is during night. Though, ‘S’ value in HSV colourspace takes care of this issue, we need to test it in a night driving mode and test it in different lightning condition like street lamp, head lamps , etc.