Advanced Lane Lines SDCND-Term1 Project4

Rubric Specifications:

Required Files:

My project submission includes the following files:

- Project4.ipynb containing the script lane finding pipeline
- Project4_report.pdf summarizing the approach and results
- outputvideo.mp4 file

Approach to the Problem:

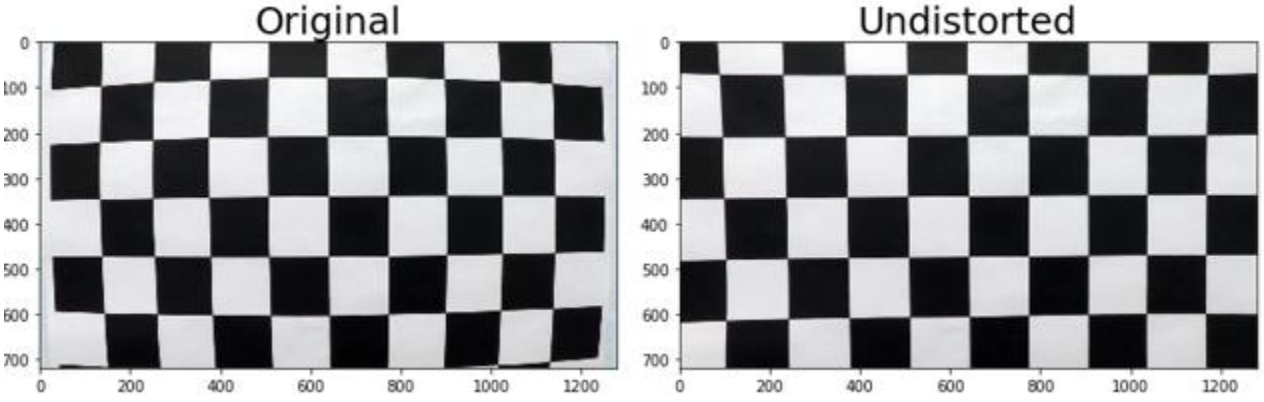
The approach followed for this project is

- Calibrated the camera from chessboard images to get the calibration matrix and distortion coefficients
- Used them to apply distortion correction
- Applied color transforms as thresholds and extract a binary image
- Applied perspective transform to get the birds eye view image
- Visualized histogram and found lane lines using sliding window search
- Detected lane pixels, fixed lane boundaries and drew continuous lane lines window
- Computed the curvature of the lane and offset of vehicle from lane center for each frame and inverted the perspective to real world
- Compiled the final output video with display of lane lines, curvature radius and center offset

Camera Calibration

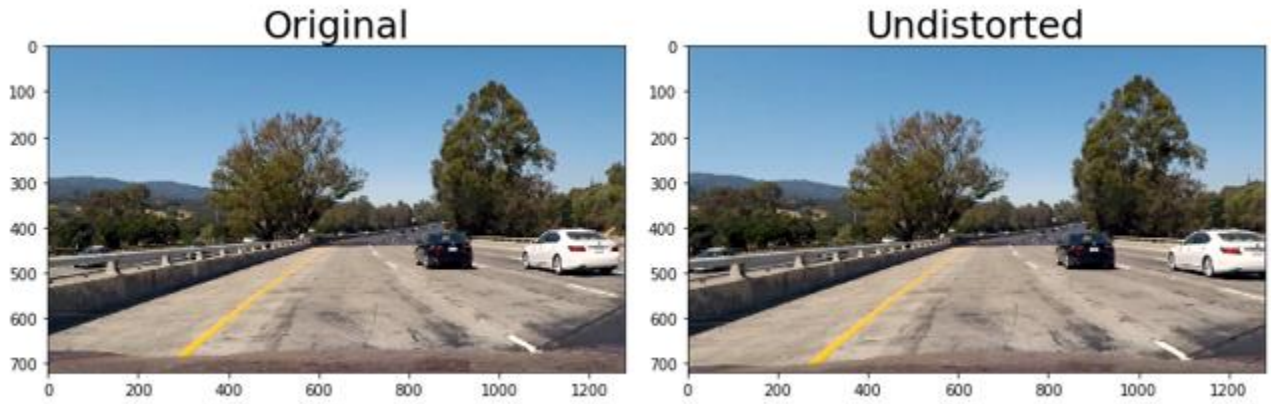
1. Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.

I have done camera calibration in from the line 2 through line 21 in cell 2 of the Project4.ipynb notebook and in the four lines of code cell 3 of the ipython notebook. I have done the preparation of object points from the 6 * 9 chessboard images to find the corner points in real world 3D and then used the cv2.findchessboardcorners() function to map the corresponding points in the image plane. Then with these two points obtained I used the cv2.calibratecamera() function to extract the calibration matrix and distortion coefficients. This distortion matrix is applied to test images using the cv2.undistort() function to undistort the images and produced the following results



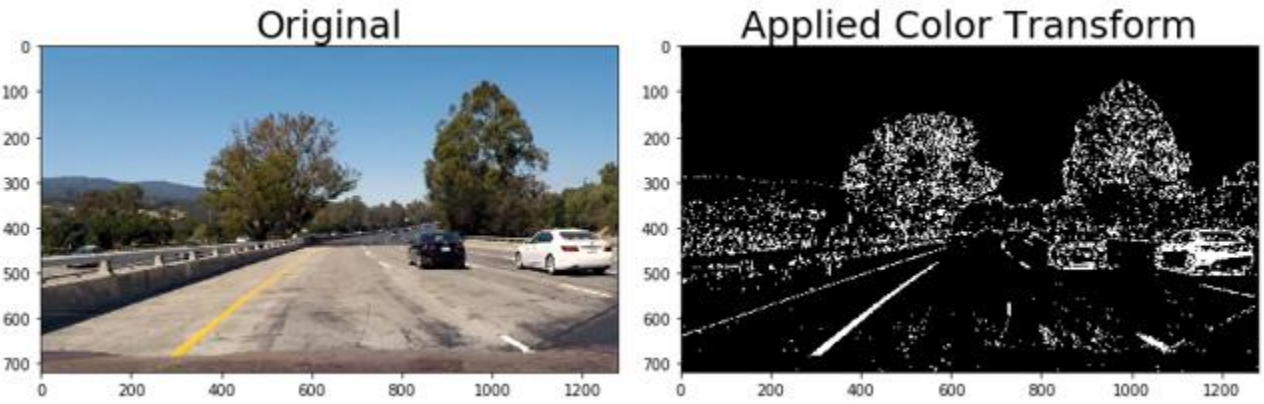
Pipeline (single images):

1.Provide an example of a distortion-corrected image.



2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

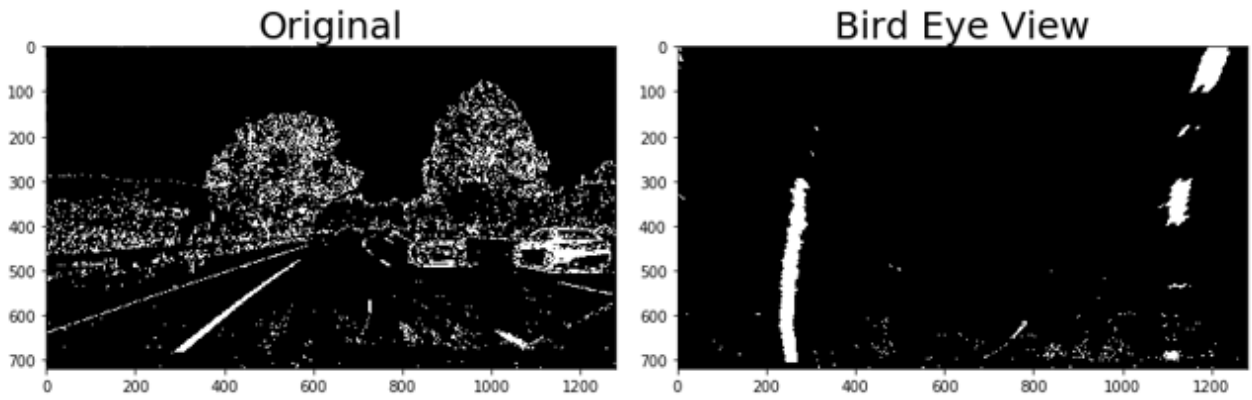
In the fifth code cell of Project4.ipynb notebook, line 2 through 25, deals with the color transforms. I employed the sobel x transform and s-channel of the HLS color space and combined the binary images of sobelx and schannel binary image in the user defined function color_transform() to get the thresholded binary image. The results are



3.Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

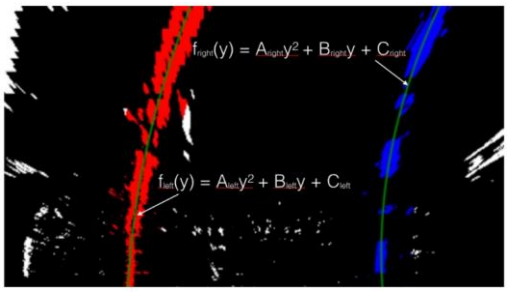
The perspective transformed is being done in the perspective_transform() user defined function. The code is covered in the line 2 through 10 of the seventh code cell in the ipython notebook. The source points and destination points are defined and the cv2.getperspectivetransform() to get the perspective transform matrices. And use the cv2.warpPerspective() function in opencv to get the birds eye view of the image. The points from top left in clockwise

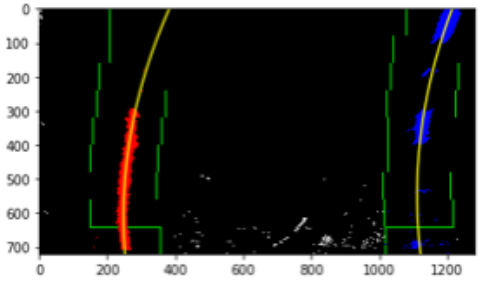
Source Points	Destination Points
585,455	200,0
705,455	1080, 0
1130,720	1080, 720
190,720	200,720

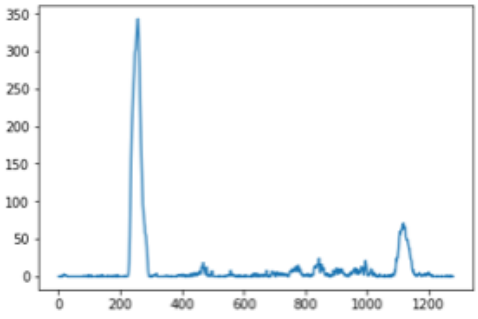


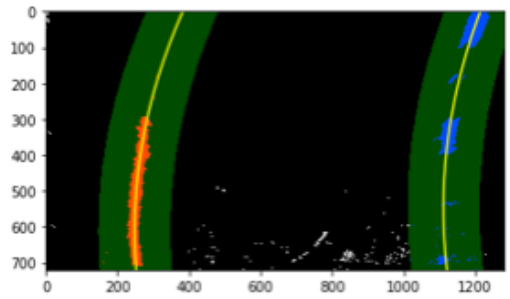
4.Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?

In the code block numbered cell 9th , from the lines 4 through 63 and the code cell numbered cell 10th, the code for identifying lane lines are done. First ,I used histogram to identify the two peak points corresponding to left and right lanes from the bird's eye image. Then used the np.polyfit() function to identify the second order polynomial for the left and right lanes respectively. The polynomials will be similar to the image shown below from the lecture. The sliding window approach is employed and the cv2.fillpoly function is used to extrapolated and draw a continuous lane lines









5.Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.

The radius of curvature is calculated using the formula

$$\text{Radius of curvature} = \frac{\left[1 + \left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{d^2y}{dx^2}\right|}$$

The corresponding code for calculation of the radius and center offset can be found in the Project4.ipynb notebook from the lines 110 through 125 in the last code cell numbered 11 and the center offset of the car is computed by calculating the midpoint of the identified lanes and subtracting the midpoint from the half width of the frame displayed. The meter to pixel conversion used is

```
ym_per_pix = 30/720 # meters per pixel in y dimension
xm_per_pix = 3.7/700 # meters per pixel in x dimension
```

6.Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly

The below image is an example of the frame with lane lines and average curve radius and center offset

Pipeline (video):

1.Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (wobbly lines are ok but no catastrophic failures that would cause the car to drive off the road!).

The link for the project final video is VideoLink

Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

I thought this project would end simpler as most of the code were being given in the lecture itself. As the project progressed it took me some tedious efforts to get it done. The few challenges I faced was finalizing the color transform channels and perspective transform coordinates. After attempting multiple things, I just fixed to keep it simple using sx sobel and s channel

of HLS . Then I tried building the pipeline using user defined function but faced with lot of errors in fixing them. Then I read about the video reader and writer documentation and used that to read the frame by frame images and process the images and build a final video, which seemed little easier for me.

I feel one area of improvement is to make the color transform robust as in the video submitted when the video stream is over the bridges the lane markings and bridge colour is not contrasting and the lane glitches slightly. It runs perfect actross everywhere. This area has to be further worked on in the future.