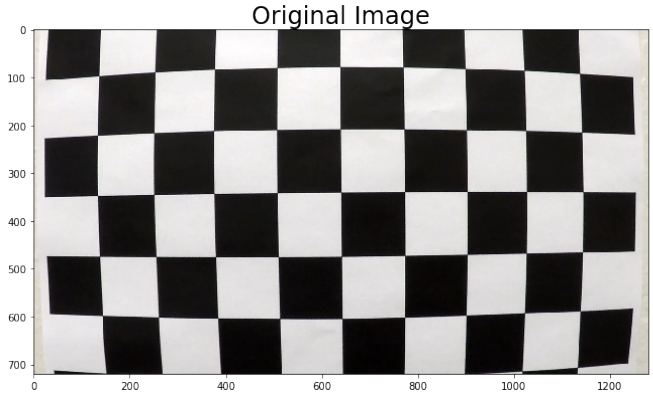
Udacity Advanced Lane Finding Project Report:

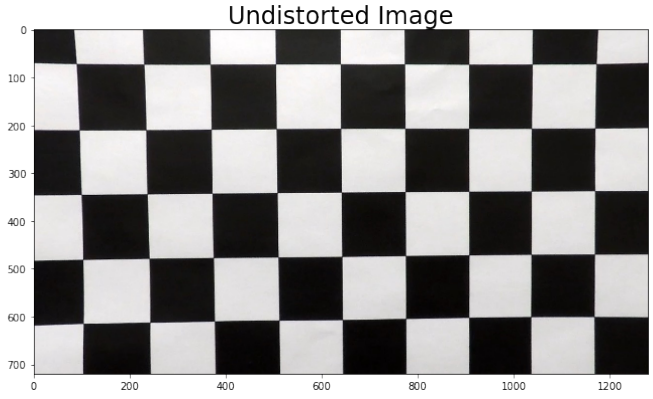
The goal of this project was to perform the following:

1. Obtain camera matrix and distortion coefficients to undistort images using them
2. Transform to bird's eye view
3. Use different color spaces and/or gradient thresholding to obtain an appropriate binary image consisting of mostly the lane lines.
4. Use the sliding window algorithm to detect lane lines in the image and fit quadradic polynomials to them
5. Calculate the curvature of the lane lines and the car offset from the center
6. And Finally draw the detected lane regions back in the original view

Obtaining Camera Matrix and Distortion Coefficients:

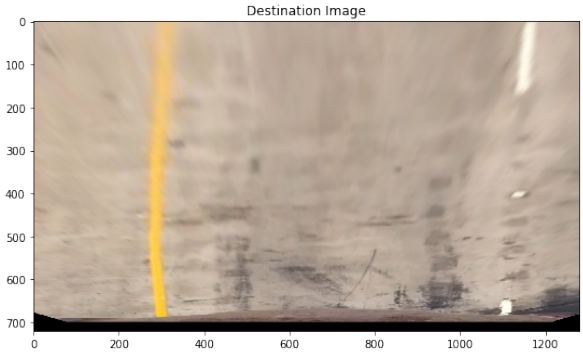
The camera matrix and distortion coefficients are found using the provided images of the chessboards. Once these values are found, any other image taken using the same camera, such as the image of the road, can be undistorted using **cv2.undistored**:





Transforming to Bird's Eye View:

Perspective transformation is found using **cv2.getPerspectiveTransform** and **cv2.warpPerspective:**



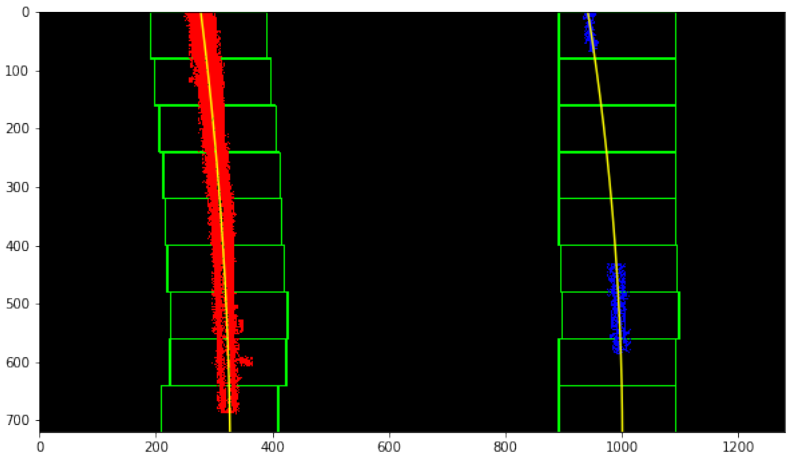
Use of Different Color Spaces and Gradient Thresholds:

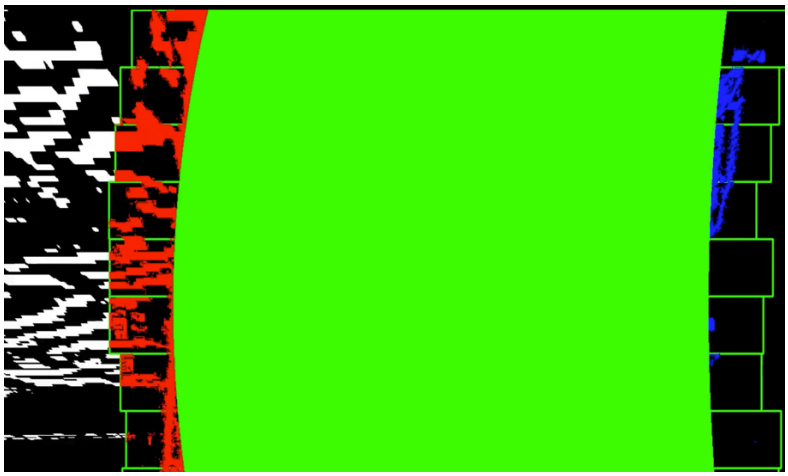
Through trial and error, it was found that applying a threshold to the S channel in the HLS color space gives a good indication of where the yellow lane is located:

However, combining that threshold with magnitude of Sobel filter in both x and y direction, gave an even better result that highlighted both the yellow and the dashed while lane lines:

Sliding Window Algorithm:

After obtaining a good binary representation of lane lines, the Sliding Window algorithm was applied to reconstruct the entire lane by fitting quadradic polynomials to the partial lane pixels:





Curvature Calculation and Offset Detection:

Once fitted quadradic functions are found, curvature of the lanes can be computed using the following formula:

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

The detailed curvature measuring operation can be found in the "measure_curvature" function in the project Jupyter notebook.

The offset from the center is simply the difference (in real world meters) between the center of the video stream and the midpoint of the two detected lane lines. The code for offset calculation is accessible at the end of second definition of "highlight_lane_original" function.

Transforming the Detections Back to the Original View:

To view the result of our pipeline in the original view, we simply apply to inverse of the perspective transform applied earlier:

The entire processed video is saved in "project_video_output5.mp4".

Problems/Issues Faced During the Pipeline Development:

There were two main problems that I faced during the development of the pipeline.

First one had to do with deciding the type of gradient threshold to use. HLS color space seemed to be the natural choice for the color space to work with since the S channel is somewhat resistant to changes in brightness which made it

extremely useful for our application. However when it came to gradient thresholding, it was surprising to see that both the directional gradient thresholds(especially x directional gradient) as well as direction of gradient itself were not as useful as one would assume them to be in this application. Through trial and error, I chose the magnitude of gradient as the most useful gradient threshold when it came to identifying the lane lines.

Second problem has to do with the pipeline occasionally detecting the passing cars as lane lines. After unsuccessfully trying to tune the thresholds numerous time to get rid of such false positives, a better approach was adopted by making sure that the passing car are filtered out during the bird's eye view transformation.

Shortcoming and Suggested Future Improvements:

Even though the currently implemented pipeline is much more robust, compared to the pipeline implemented in the first project. There are still some shortcomings and steps can be taken to further make the algorithm more robust.

The algorithm would likely fail if the cars passing by are too close to our lanes lines as the perspective transform would not be able to filter them out. A suggested solution is to integrate object detection in the pipeline to detect cars and prevent them from being considered in our gradient threshold operation.

Due to a similar reason, the pipeline would also possibly fail if there are cars moving in the same lane front of us. Again, an object detection algorithm, labeling those cars could be utilized to ignore the cars from being considered in our thresholding operation.

It can also be noted that we are only using camera and as such we have no access to depth information. Accessing depth information by utilizing Radars or stereo cameras would provide more accurate information and would enable us to distinguish the road surface from other surfaces.