Project Writeup

Anirudh AV (anirudh.av93@gmail.com)

---

1. Camera Calibration

- The chessboard images were loaded, converted to grayscale and the corners were found using find chessboard corners.
- Two arrays were created **objpoints** and **imagepoints** which had the corners in object (board) coordinates and the image (camera world coordinates).
- The found corners were then used as arguments in the calibrate camera to compute the distortion coefficients and the camera matrix.
- The found chessboard corners are shown on the image using drawChessboard corners function.
- The distortion matrix, the distortion co-efficients, the rotation and translation vectors were calculated using the **cv2.calibrateCamera** function.
- The camera matrix and the distortion coefficients were pickled and saved into a file.

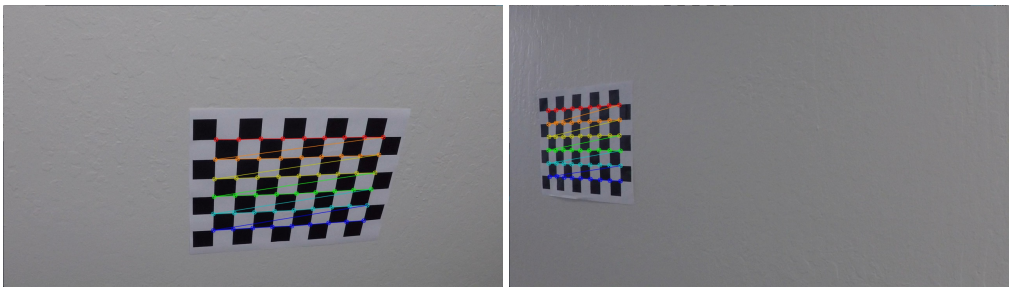

Figure 1: chessboard images pose 1 and 2

- 16 such chess board images were taken and used for calibration and the distortion coefficients and the camera matrix was computed from them.

2. Image Processing Pipeline

- The images were initially undistorted using the coefficients found during camera calibration.
- Then a perspective transform was defined such that the images were rotated and centered in a way that the region of interest was centered.
- The transformation matrix was obtained using the **cv2.warpPerspective** function in openCV.

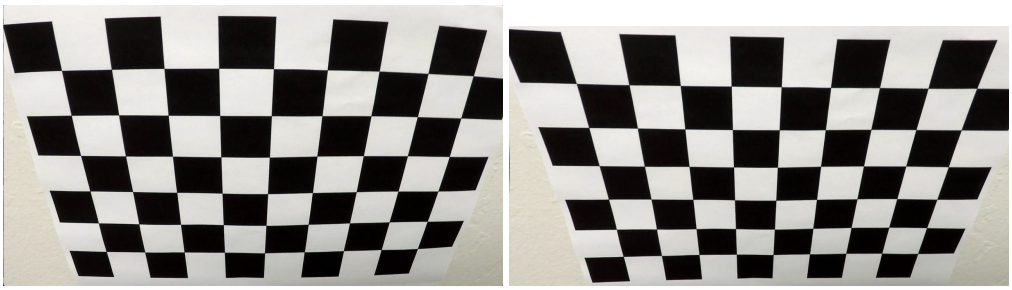

Figure 2: Distorted image (left) , After Undistortion (Right)

Figure 3: Distorted image (left) , After Undistortion (Right)

- Once the images were warped, a binary threshold was applied.
- A variety of thresholds were applied and finally it was shown (Jeremy Shannon) that the HLS l channel and the LAB b channel were really good thresholds to extract the region of interest.
- Then as shown in the udacity classroom, the sliding window fit / or the extrapolation fit was used to identify the lane lines.
- The extrapolate polyfit uses the positional information of the lane lines from the previous image and begins the search from the same region in the next image.
- Keeping track of this lane data accross these images was possible due to the line class, where the **l_line & r_line** instances were used to keep track of the left and right hand lanes.
- The detected lane lines were then drawn back on the original image.
- The radius of curvature was measured as suggested and drawn back on the original image.

Figure 4: Input Test Images

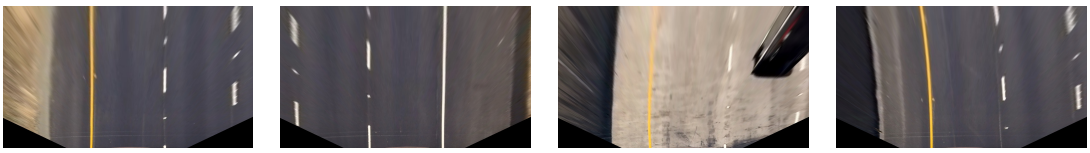

Figure 5: Undistorted and perspective transformed Images

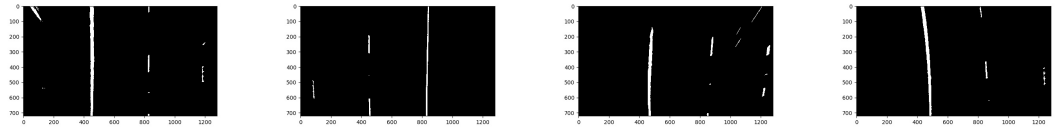

Figure 6: Thresholded binary image

3. Code References

Topic	Line
Undistortion	33
Line Class	38
Threshold	146
Sliding Window	184
Radius of Curvature	334

Table 1: Code references

4. Limitations and Issues Faced

- The first major Issue I faced was selecting a thresholding function in order to identify the pixels belonging to lane lines. After a variety of thresholds and many iterations, A fellow peer's suggestion to use the hls and lab channels and threshold the individual channels, worked for me.
- The second major issue (also a limitation) was the selecting of source and destination points for the perspective transform in the image. I hard coded the points and that reduces a lot of flexibility in the algorithm.
- The Extrapolate polyfit relies upon the sliding window fits results initially, which means that if we want a robust algorithm, the sliding window fit should work on its own without extrapolating the fit. This was a check I was trying to perform and observed a lot of flickering while implementing an algorithm that purelt relies on sliding window fit.
- The undistortion gave me different (valid) results while using a newOptimal camera matrix and using the same camera matrix, I chose the newOptimal camera matrix result by inspection, There should be a more mathematical proof formulated for the choice.

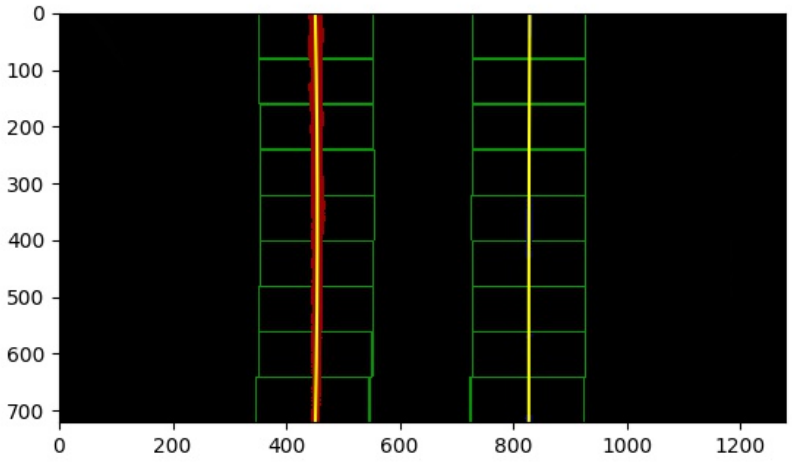

Figure 7: Initial sliding window fit to find lane lines

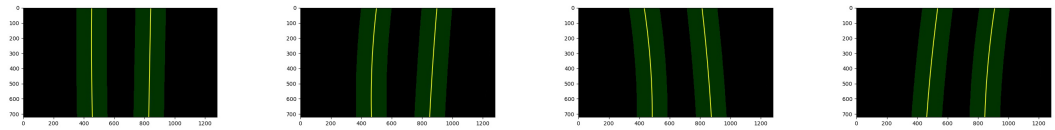

Figure 8: Extrapolated search using the information from sliding window fit in the previous figure

Figure 9: The detected lane lines drawn back on the original image using the cv2.fillPoly function

Figure 10: The measured radius of curvature written back on each image