[Project Writeup] Advanced Lane Lines

PROJECT SPECIFICATION

Advanced Lane Finding

Link to this writeup: https://www.evernote.com/l/AC52yb_mrQxDoLhCt4_zDtIYkjuZYPxfgFw

Writeup / README

CRITERIA	MEETS SPECIFICATIONS
Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.	iPython notebook: Distortion_Correction.ipynb I have added the images in the Distortion_Correction.html file. Readme: writeup.pdf Video: **https://youtu.be/XjQ2p3FoGE4** Find images at: image_samples/*.png For worksheet of how I reached to the pipeline processing_each__image__turn_right.ipynb processing_each__image__turn_right.html Review #1 solutions. Fluctuation in lane lines are rectified by using previous lane lines when error is detected. Lane curvature radius measurement is corrected by multiplying the pixel to meter coefficient.

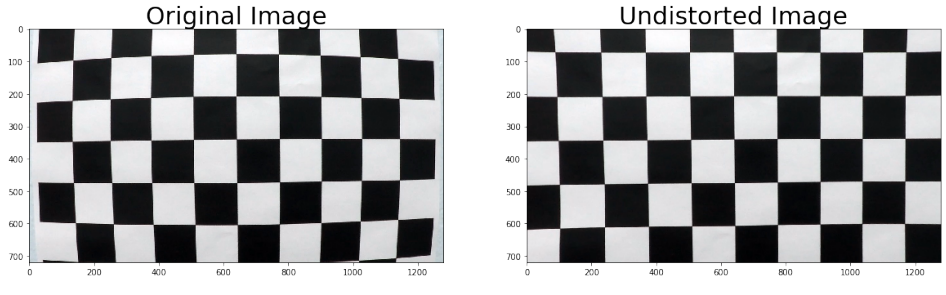
Camera Calibration

CRITERIA	MEETS SPECIFICATIONS
Briefly state how you computed the camera matrix and distortion coefficients. Provide an example of a distortion corrected calibration image.	OpenCV functions or other methods were used to calculate the correct camera matrix and distortion coefficients using the calibration chessboard images provided in the repository (note these are 9x6 chessboard images, unlike the 8x6 images used in the lesson). The distortion matrix should be used to un-

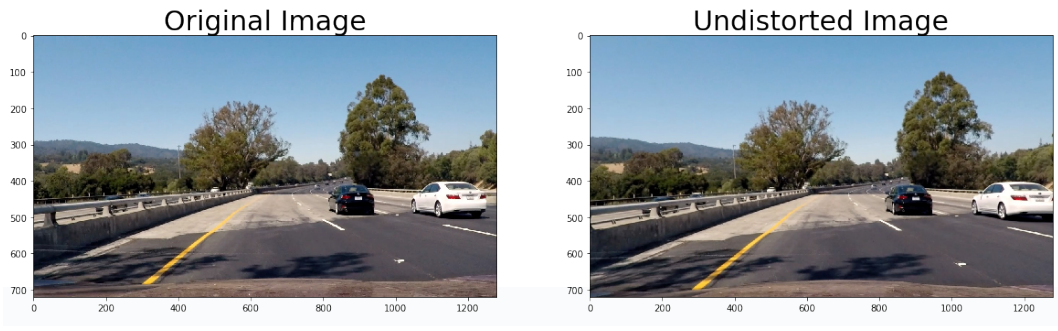
distort one of the calibration images provided as a demonstration that the calibration is correct. Example of undistorted calibration image is Included in the writeup (or saved to a folder).

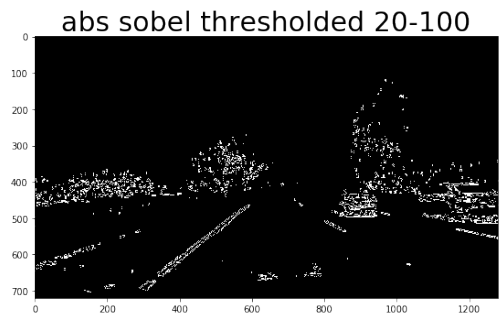
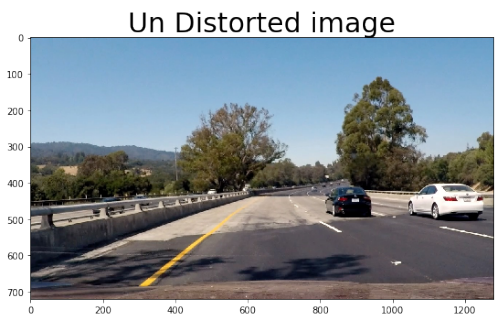
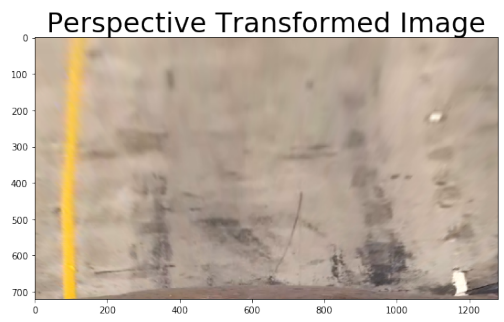
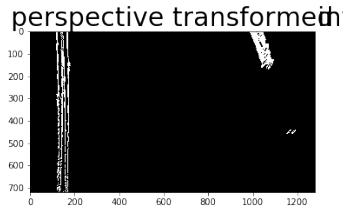
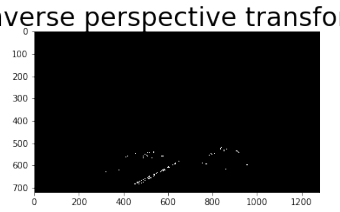
Calibration object points and image points are pickled and added into camera_cal/wide_dist_pickle.p

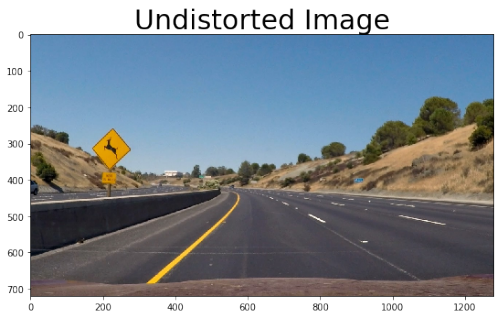
Find this image in image_samples/orig_undistored.png

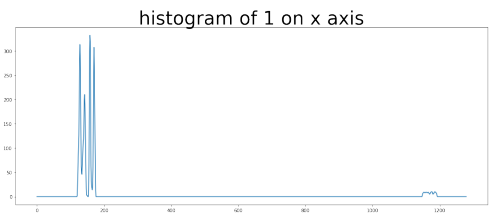
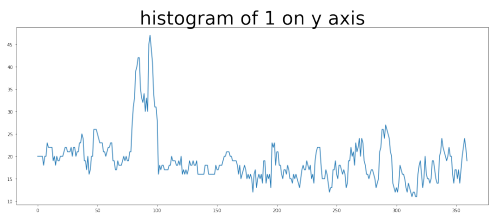


Pipeline (test images)

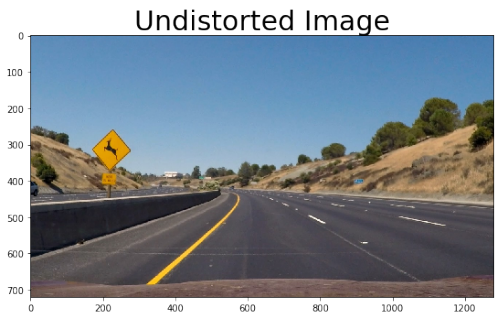
CRITERIA	MEETS SPECIFICATIONS
Provide an example of a distortion-corrected image.	Distortion correction that was calculated via camera calibration has been correctly applied to each image. An example of a distortion corrected image should be included in the writeup (or saved to a folder) and submitted with the project. Find this image in image_samples/orig_undistorted_lane.png 
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.	A method or combination of methods (i.e., color transforms, gradients) has been used to create a binary image containing likely lane pixels. There is no "ground truth" here, just visual verification that the pixels identified as part of the lane lines are, in fact, part of the lines. Example binary images should be included in the writeup (or saved to a folder) and submitted with the project. Pipeline: Undistort image Perspective transform the image get the birds eye view of the lane lines.

	Convert the RGB image to HLS image. (S value gave a good clear view of lane lines and reduced noise.) Retain S values with Threshold between 150, 255 Make a binary image of the S channel, by using S channel for 3 channels. Do a Sobel on X axis of the image The absolute value of the image is captured. Find this image in image_samples/pipeline_applied.png  
Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.	OpenCV function or other method has been used to correctly rectify each image to a "birds-eye view". Transformed images should be included in the writeup (or saved to a folder) and submitted with the project. Perspective transformed Image Find this image in image_samples/perspective_trasnformed.png   Pipeline applied on Perspective transformed Image. An image of inverse perspective image transformation is also included.   
Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?	Methods have been used to identify lane line pixels in the rectified binary image. The left and right line have been identified and fit with a curved functional form (e.g., spine or polynomial). Example images with line pixels identified and a fit overplotted should be included in the writeup (or saved to a folder) and submitted with the project. Methods sliding_window and draw_lane along with compute_rad_curv helped draw the lane line s and compute the radius. Find this image in image_samples/lane_lines_identified_and_radius_computed.png

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.	Here the idea is to take the measurements of where the lane lines are and estimate how much the road is curving and where the vehicle is located with respect to the center of the lane. The radius of curvature may be given in meters assuming the curve of the road follows a circle. For the position of the vehicle, you may assume the camera is mounted at the center of the car and the deviation of the midpoint of the lane from the center of the image is the offset you're looking for. As with the polynomial fitting, convert from pixels to meters.
	
	Find this image in image_samples/histogram.png
	
	Histogram of Lane line pixels are drawn:
	
	  
	
	**def compute_rad_curv(xvals, yvals):
	ym_per_pix = 30/720 # meters per pixel in y dimension
	xm_per_pix = 3.7/700
	fit_cr = np.polyfit(yvals*ym_per_pix, xvals*xm_per_pix, 2)
	y_eval = np.max(yvals)
	curverad = ((1 + (2*fit_cr[0]*y_eval*ym_per_pix + fit_cr[1])**2)**1.5)/np.absolute(2*fit_cr[0])
	return curverad**
	
	This method calculated the radius of curvature. Using the values specific to American road rules.

Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly.	The fit from the rectified image has been warped back onto the original image and plotted to identify the lane boundaries. This should demonstrate that the lane boundaries were correctly identified. An example image with lanes, curvature, and position from center should be included in the writeup (or saved to a folder) and submitted with the project.

Pipeline (video)

CRITERIA	MEETS SPECIFICATIONS
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 image processing pipeline that was established to find the lane lines in images successfully processes the video. The output here should be a new video where the lanes are identified in every frame, and outputs are generated regarding the radius of curvature of the lane and vehicle position within the lane. The pipeline should correctly map out curved lines and not fail when shadows or pavement color changes are present. The output video should be linked to in the writeup and/or saved and submitted with the project. **https://youtu.be/XjQ2p3FoGE4**

Discussion

CRITERIA	MEETS SPECIFICATIONS
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?	Discussion includes some consideration of problems/issues faced, what could be improved about their algorithm/pipeline, and what hypothetical cases would cause their pipeline to fail. Manitude threshold and direction did not work very well for me. Effectiveness of other color transformations might give a good result. Better utilisation of previous polynomial to guess next lane line polynomial might give as a real time performance. In an intersection, the chances of error is very high.