Meets Specifications

Congratulations     
Nice job with this submission. Well documented code, modular structure, a very detailed and to the point writeup. It's amazing.  
Keep up the good work !!!  
Well Done and Good luck for next projects. :)

**Writeup / README**

**The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled.**

Steps taken to complete the project are explained in the writeup, with references to where in the code each rubric item was implemented. Good job in identifying the issues in your pipeline and suggesting ways to improve.

**Camera Calibration**

**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).**

**Awesome**

* The chessboard images have been used to calculate the camera calibration matrix and distortion coefficients.
* Chessboard corners are found using cv2.findChessboardCorners
* Camera matrix and distortion coefficients are calculated using cv2.calibrateCamera.
* Good job for demonstrating the undistortion from one of the calibration images.
* Well done for sharing the image in the writeup.

**Pipeline (test images)**

**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.**

**Awesome**

* The undistorted test image shows that the distortion correction has been properly applied to the road images.
* It can be seen that distortion correction is applied in the pipeline as well. Nice work!

**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.**

**Awesome**

* Good job applying a combination of color and gradient thresholds to create a binary image with clearly visible lane lines.
* Well done for sharing the binary image in the writeup.

**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.**

**Awesome**

* Appropriate source and destination coordinates have been selected to warp the images to a bird’s eye view with parallel lane lines.
* Good job using cv2.getPerspectiveTransform() to compute the perspective transform matrix.
* Well done using cv2.warpPerspective() in to warp the binary image into a "birds-eye view".
* Thanks for sharing the output image to show results.

**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.**

**Awesome**

* Nice job using a histogram and sliding window search to find the locations of the lane lines, and using the previous detections to perform a targeted search in subsequent frames.
* Nice explanation of polynomial fit to draw a new lane based on current image or use lanes from previous frames.
* Thanks for sharing the output image to show results.

**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.**

**Awesome**

Great job showing the radius of curvature and vehicle position with respect to center of the lane. The estimates are very reasonable.

**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.**

**Awesome**

Good job warping the lane area back into the original image space and combining it with the original undistorted image

**Pipeline (video)**

**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.**

**Awesome**

The result video looks great! The pipeline is able to accurately map out the true locations of the lane lines throughout the entire video, and doesn't fail in the presence of shadows and pavement color changes. This is a very impressive result!

**Discussion**

**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.**

**Awesome**

* Good job explaining your pipeline and discussing some shortcomings and improvements.
* This research paper <http://airccj.org/CSCP/vol5/csit53211.pdf> goes into how to detect curves and will also help in detecting faded lanes. It uses an extended version of hough lines algorithm to detect tangents to the curve which can help you detect the curve.
* Here is another paper which talks about real time lane detection. <http://www.vision.caltech.edu/malaa/publications/aly08realtime.pdf>