Writeup Template

This writeup presents the method, challenges and process used to complete the Advanced lanes finding project by implementing computer vision algorithms and tools.

Advanced Lane Finding Project The project document is in the adv_proj.ipynb file.

To complete the project, I followed the objectives in given as a guide

- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- Detect lane pixels and fit to find the lane boundary.
- Determine the curvature of the lane and vehicle position with respect to center.
- Warp the detected lane boundaries back onto the original image.
- Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Camera Calibration

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

The calibration code was given in the example template for the project. I did not substantially change the code but applied it in calibration. To use it, I generated objpoints and imgpoints whice are used to apply distortion correction by using the cv2.undistort() function.

2. Application of distortion correction

Distortion correction is carried out by applying the generated objpoints and imgpoints in the previous section to the test images using the cv2.undistort(img,camMat,distCoef,None, camMat) function where the input arguments to the function are generated as outputs of the cv2.calibrateCamera() function.

3. 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.

I used a combination of HLS, gradient and magnitude transforms in threshold finding. To make the process easier, I designed a GUI tool for this purpose. The GUI tool made easy iteration while changing values for the transforms. The transforms I wrote in this section are the Sobel transform, magnitude, direction, hls and rgb transforms. Although, I wrote the 5, using my GUI tool I was able to get acceptable results using just 4 (the first four).

As part of this process, I had to define a region of interest to help with the thresholding. Although, the pipeline worked well without the region of interest, but the video was distorted when a black car was passing because the car was highly reflective. In order to address this, I modified the region of interest function given to us in the first project by udacity and used it to remove the problem.

Examples of images after thresholding can be found in the output_images/thresh_images folder

Example of the bad video output (without region of interest function) is included in the Project_solution/project_video_out_noROI_30_40s.mp4. The effect of no region of interest can be seen between 30 and 40s of the video

In [ ]: