Writeup

The goals / steps of this project are the following:

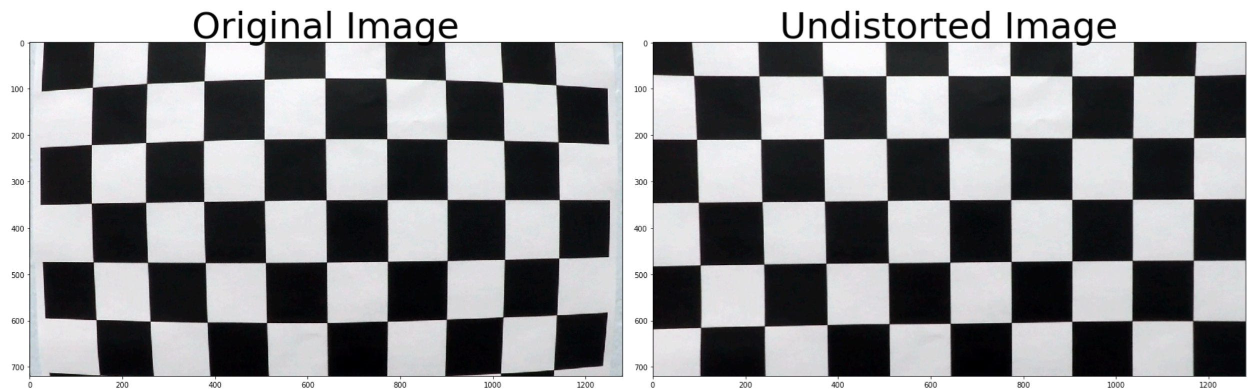
- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- 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.

Rubric Points

Camera Calibration

1. **Have the camera matrix and distortion coefficients been computed correctly and checked on one of the calibration images as a test?**

   Yes sure, I read 20 image files from camera_cal to calculate the image points and object points; after that, use those point to compute the calibration matrix and distortion coefficients; saved the matrix and coefficient and applied them to a test image, here is the result:



Pipeline (single images)

1. Has the distortion correction been correctly applied to each image?

Sure, as first step the function process_image, I undistort the image using precomputed the calibration matrix and distance coefficient.

2. Has a binary image been created using color transforms, gradients or other methods?

Yes, I put these intermediate results in folder output_image as part of my submission.

I applied sobel operation on both X and Y and channel filter after converting it to HSV space.

3. Has a perspective transform been applied to rectify the image?

Yes, I put these intermediate results in folder output_image as part of my submission. The basic idea is to select bottom half of the original image as source points.

4. Have lane line pixels been identified in the rectified image and fit with a polynomial?

Yes sure, see attached intermediate results image. I created a class tracker, which is used for all the frames from the video. For each frame, I used the sliding window method to find the left and right lane; the convolution signal is calculated to find the best window center positions in a thresholded road image. Each window centroid was cached into an array, upon all the window centroids identified, most recent 15 window centroids was taken to fit a second order polynomial and finally identify the left and right lane line by applying the polynomial to the points of left side and right side respectively.

5. Having identified the lane lines, has the radius of curvature of the road been estimated? And the position of the vehicle with respect to center in the lane?

- First, I define conversions in x and y from pixels space to meters,
  ```
  ym_per_pix = 30/720 # meters per pixel in y dimension
  xm_per_pix = 3.7/700 # meters per pixel in x dimension
  ```
- Use them fit a second order polynomial in world space
- Calculate the radius of curvature from the polynomial
- The offset from lane center is calculated by comparing the camera image x center with the center of left lane line and right lane line.

Pipeline (video)

Does the pipeline established with the test images work to process the video?

Yes, video attached.

I was working to images from folder test_image, in order to make same pipeline work for the video, I made some changes to make it better work for video:

- Put the pipeline to a function
- Predefine class tracker so I can reuse it for all the frames from video and reuse the
- result from previous frame.
- Comment out some intermediate codes.

Discussion

- The pipeline has some issues working with challenge video, due to sharp turn (low curvature and shadow too close the isolation strip)?
- Is it possible to optimize class tracker such that it only processes the new window since most windows from current frame are same as previous frame?