Advanced Lane Finding
CarND-P4
Ayanangshu Das
May 02,2017

Identifying Lane lines using advanced computer vision techniques

This is in advancement to to project 1 where we needed to locate lane lines more accurately. The pipeline consists of multiple stages which I am going to discuss in this write up.

Calibrating the camera

Camera images sometimes get optically distorted turning straight lines into curved ones. This creates problems in our processing of images as it becomes harder turning 3D images to 2D. Cv2 offers capability to undistort the images with a reference given using chessboard images.

As a first step, we need to take images of chessboards from different angles. The chessboard will have specific corners. Given the number of corners, the cv2.findChessboardCorners( ) can find their locations.

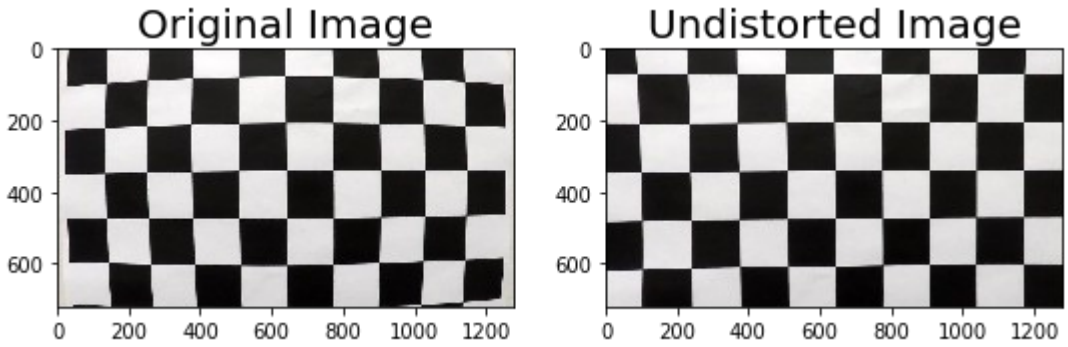
Now having the original image(s) and the identified corner points, cv2.CalibrateCamera ( ) can generate distortion coefficients which can undistort any future images taken from that camera.
For generalization purpose, we need to feed many images by varying angle, distance and orientation.

Pipeline for finding lanes in an image

Once we have calculated the distortion coefficients, we can look at the sample images of roads and start building our pipeline to extract lane lines. The following is a sample image

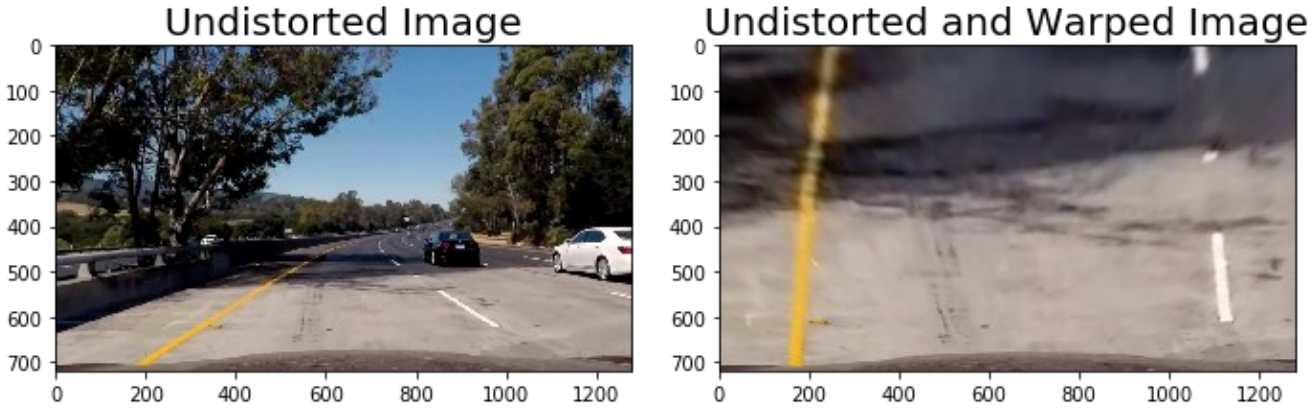
Step 1: Correct camera distortion

Cv2.undistort performs the undistortion



Step 2: Perspective Transform

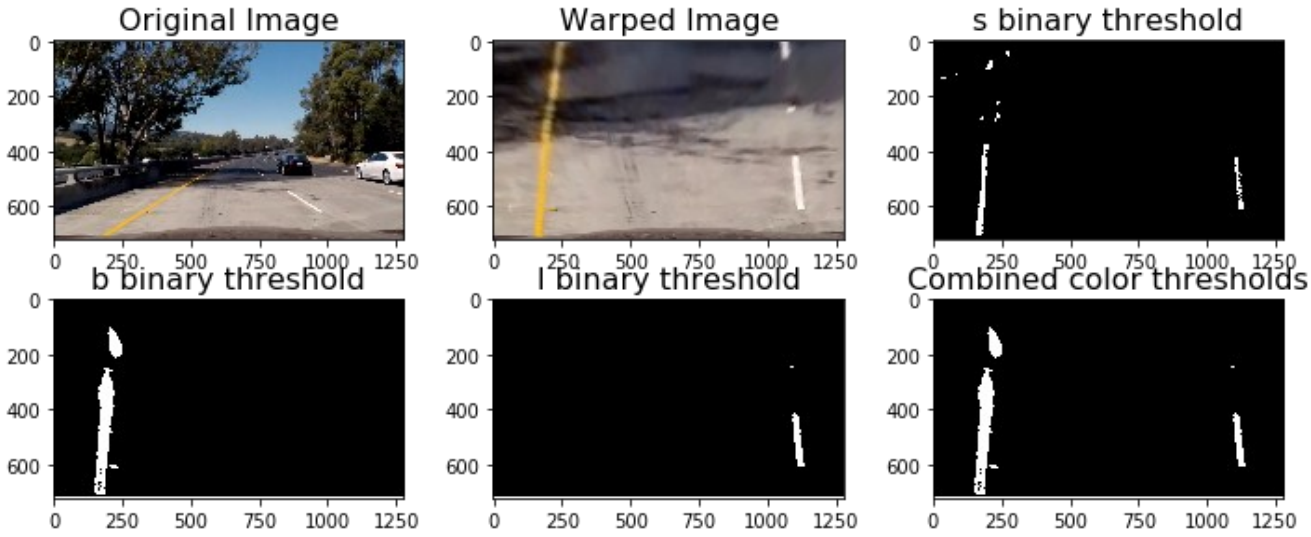
Applying perspective transform makes lane lines appear parallel to each other.
Cv2.getPerspectiveTransform( ) gives the warped view of the original image



Step 3: Image thresholding

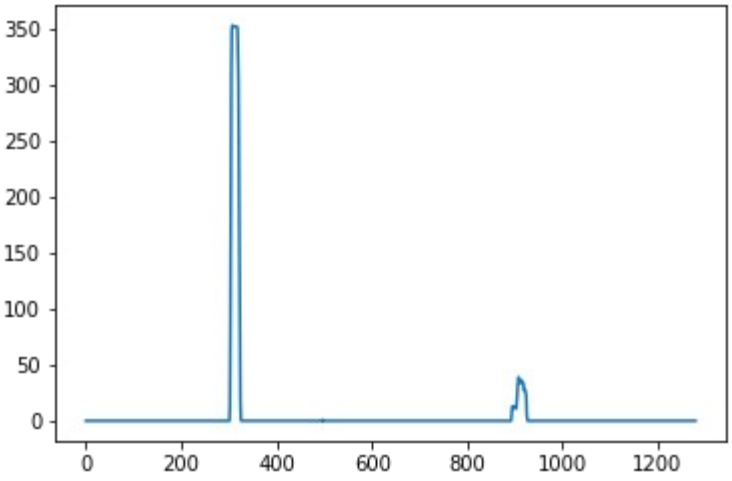
Earlier we have used canny edge detection to detect edges along x and y direction. Sobel operator is at the heart of canny edge detection and we can refine this process according to our need which is done in the function **convert_bin_threshold()**

I have created a combined binary thresholded image using the S channel of HLS color space, L channel from LUV color space and B channel from the lab color space which highlights almost all of the yellow and white lane lines



Step 4: Histogram for potential lane detection

We can take histogram from the x-axis to detect most prominent peaks to identify potentials lanes



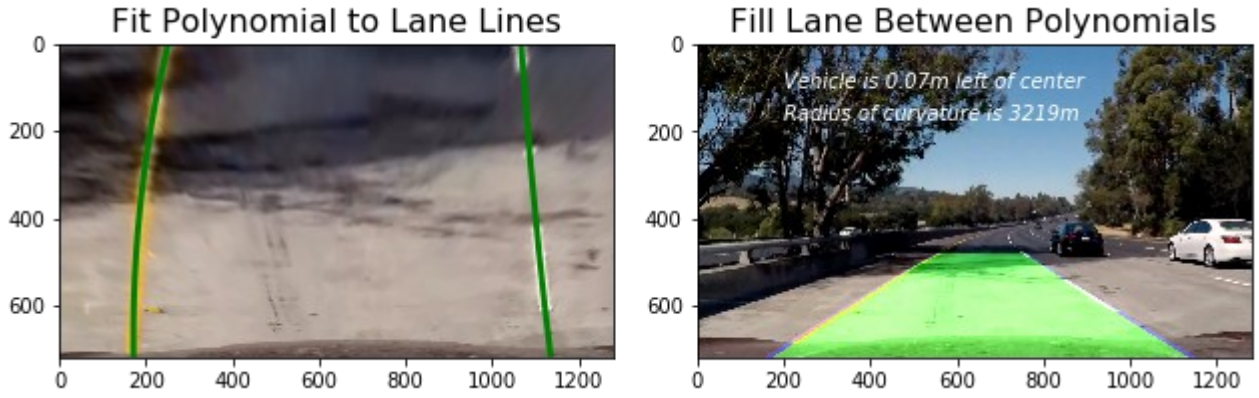
Step 5: Fitting a polynomial to the lane lines by calculating the vehicle position and radius of curvature

At this point I am using a combined binary image to isolate lane line pixels and fit a polynomial to each of the lane lines. The space in-between the lane lines is filled to highlight the drivable area inside the lane. The position of the vehicle is measured by taking the average of the x intercepts of each line. The function fit_polynomial () identifies the lane lines by identifying the peaks in a histogram detecting non-zero pixels in close proximity to the peaks

Step 6: Calculating the radius of curvature and center offset

Since we perform the fit in pixels and the curvature has to be calculated in real world meters, we have to use a pixel to meter transformation and recompute the fit again

The mean of the lane pixels closest to the car gives us the center of the lane. The center of the image gives us the position of the car. The difference between the two is the offset from the center. It is computed in the function **radius_of_curvature()**



Step 7: Pipeline for finding lanes in a video

Video is practically is a stream of images so the techniques learned so far is applied to video as well. As consecutive images in a videos has some relation, it can be exploited to optimize the pipeline for a video rather than brute force search of lanes in each sequential image.

Challenges/ Future Work

1. My current pipeline does not work on the provided challenge video because some frames fail to find any pixels for one of the lane lines. Having feeds from multiple cameras at different angles might help make lane lines always visible.

2. Handling of bad frames needs to be robust
3. The harder challenge video is not working as there are two candidate of right lane line parallel to each other and we are detecting wrong line - needs improvement here.
4. Sudden change of direction also seems to be a problem for my pipeline - need more thought to address that.