Advanced Lane Finding Project

Finding Lane Lines on the Road

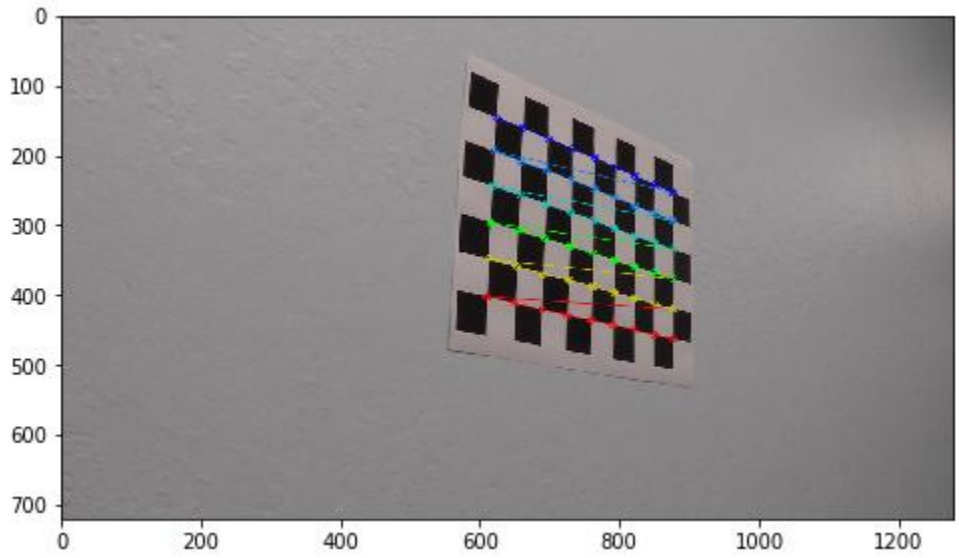
The goals / steps of this project are the following:

- 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.

Reflection

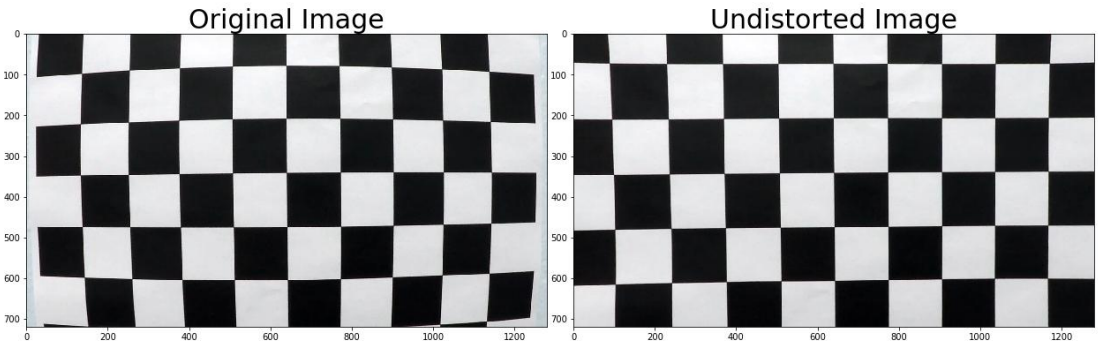
1. Camera Calibration

Step1: use cv2.findChessboardCornersCamera to get the corner objects;



Step2: use cv2.calibrateCamera to get the intrinsic_matrix and distortion_coeffs;

Step3: use cv2.undistort to undistort the image, the contrast is as below:



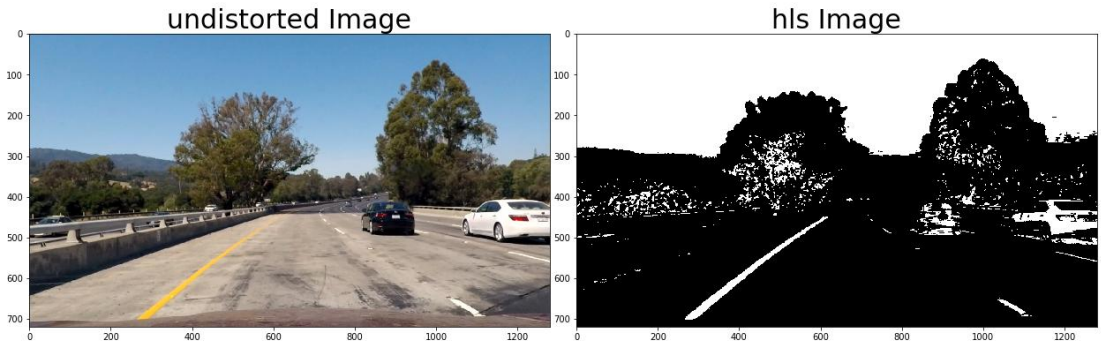
2. Pipeline(test images)

Step1: Distortion correction with the Distortion via camera calibration ,one of the example is as below(the function of cv2.undistort is used):

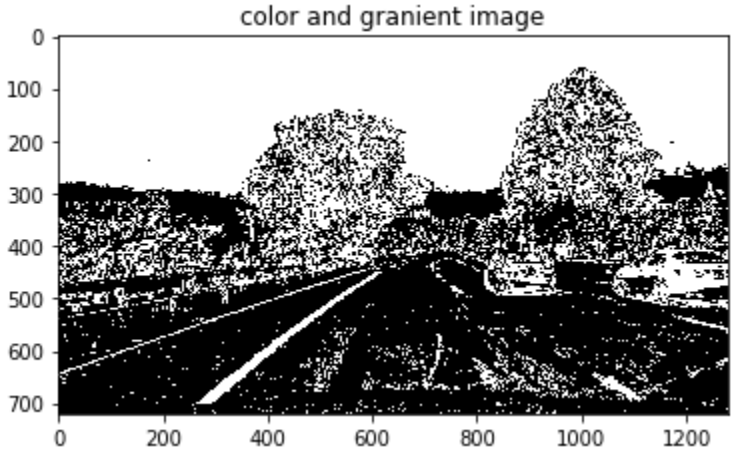
correction cv2.findChessboardCornersCamera to get the corner objects;

Step2: use cv2.Sobel to do edge detection in X-derection; In the image, the bottom line of the guard bar is clear and the left line is wiped off.

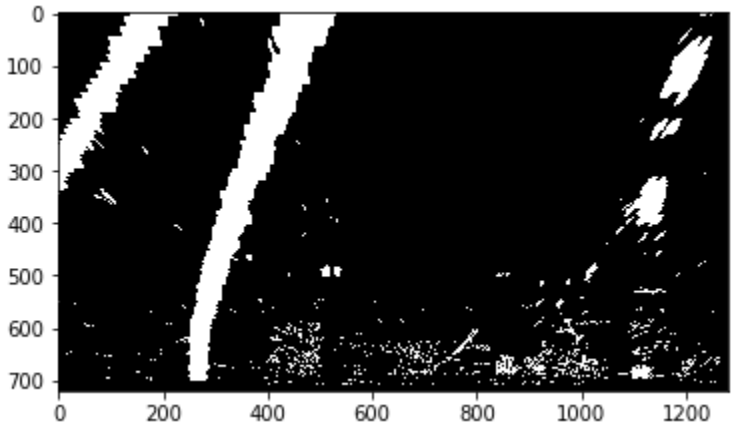
Step3: use HLS transform, the effect is as below, the left line is very clear, and the bottom line of the guard bar is wiped off, but the right line is not distinct.



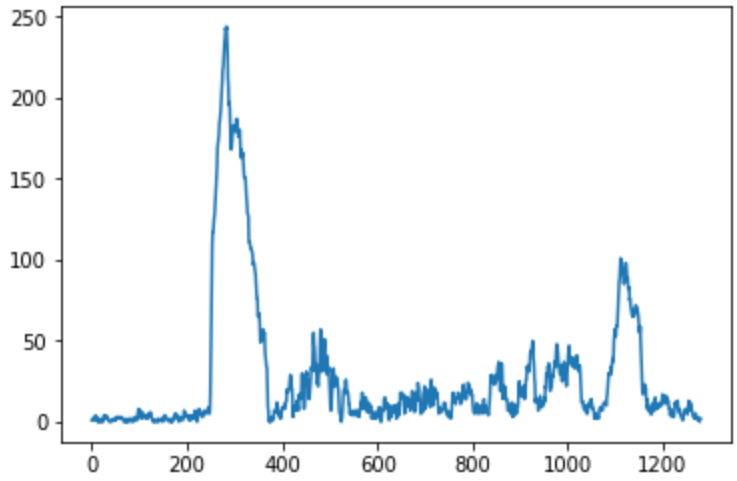
Step4: use HLS transform combined the Xsobel transform, the effect is as below, the left line and right line is clear relatively.



Step5: set the original points are [280,700],[1120,700],[800,510],[510,510], and set the destination points are [270,700],[1110,700],[1110,480],[270,480]. Use cv2.warpPerspectiveuse to get the warped image in birds-eye view.



Step6:calculate the histogram in y direction:

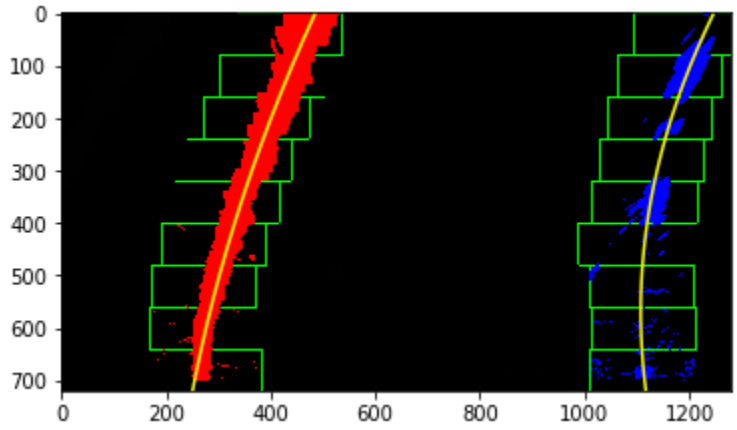


Step7: identify lane-line pixels and fit the line.

First: set the max value and the second max value in x-axis of the histogram as the base points of the left and right line.

Second: set the 9 windows along the y direction, the first windows' base point is set in first step; store the indices of the pixels in the first windows, calculate the mean value of these pixels in x-axis as the second windows of the left and right lines, the third windows, the fourth.....follow the same steps.

Third：with the points of the second step, use np.polyfit to calculate the parameters of the left and right lines. The result is as lelow:



Step8: calculated the radius of curvature of the lane and the position of the vehicle. because the curve is described as :

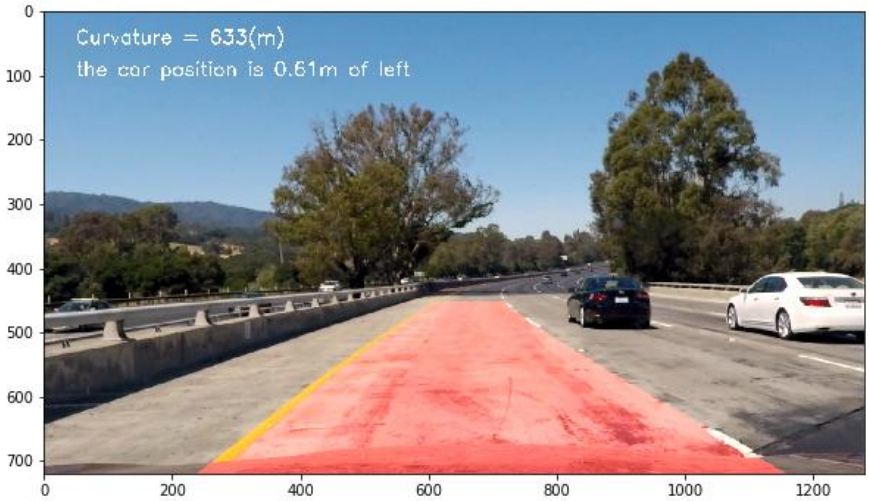
$f(y) = Ay^2 + By + C$

so the curvature is calculated as below(it will be changed to be as meters with the relationship between meter and pixel):

$$R_{curve} = \frac{(1+(2Ay+B)^2)^{3/2}}{|2A|}$$

Assuming the camera is mounted at the center of the car, the offset of the midpoint of the lane from the center of the image is looked as the position of the car(also, it will be changed to be as meters with the relationship between meter and pixel). The result is as below:

Step9: Use the parameters of the left and right lines to set polygon, then use the cv2.getPerspectiveTransform(dst,src) to get Minv , with this matrix, use cv2.warpPerspective to warpe back onto the original image. The result is as below:



3. Pipeline(video)

In video test, to the frames of the video, I used the same pipeline as in the image .

In addition，a diagnose function is developed to check the rationality of the two lines by the parameters of the lines and the distance of the two lines. If the rationality is not ok, used the previous lines.

The result of the video processing is shown in the 'test_video.mp4.

Discussion

1.The problem I faced：

a. The right line of the lane is not very clear from the shadow of the road. if I only use the HLS transformation, most of the time, the right line can be shown very clearly, but sometimes it can be wiped off.

My solution is :

Step1: use the sobel transformation combined with HLS to maintain the right line feature(but the shadow and the bottom line of the guard bar are kept too, so after histogram transformation, the peak position of right line in x axis is not very protruding);

Step2:before draw poly in image, use a function to check the rationality of the two lines by the parameters of the lines and the distance of the two lines. If the rationality is not ok, used the previous lines.

2. The potential shortcomings with my current pipeline

1.First shortcoming:

I think the calculation of the whole image is wasteful of time, I think the ROI is still working.

2.Second shortcoming：

If there is a car running on the line, I think the line detection maybe fails

3.Third shortcoming:

The water on the road or the great shadow are still the great threat to the lanes' feature.

3. Possible improvements to my pipeline

To the above three shortcomings, the possible improvement would be:

1.First shortcoming:

Set the ROI as the previous polygon.

2.Second and third shortcomings:

In addition of the filter method, if the line detection fails , the previous lines would be used.