Advanced Lane Line Finding

Udacity self-driving car Nanodegree project 4

• 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.

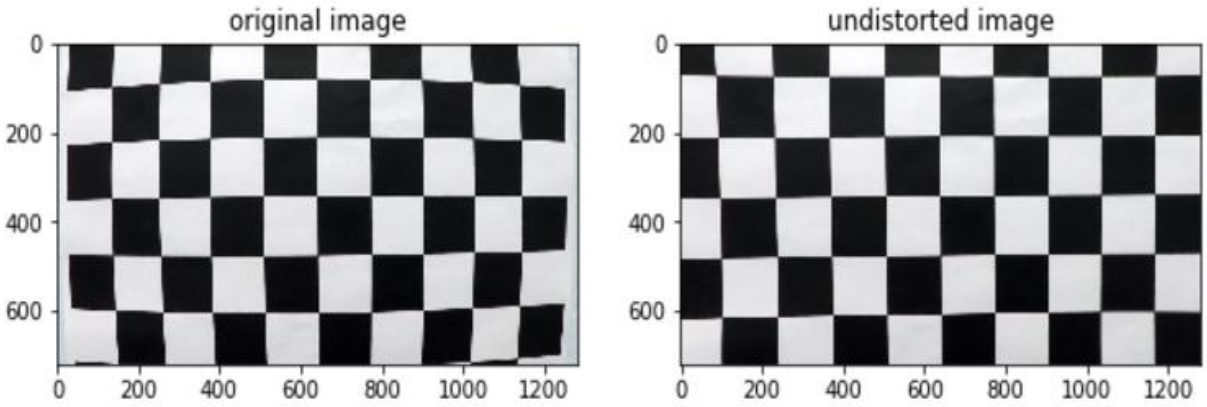
• Warp the detected lane boundaries back onto the original image.

• Output visual display of the lane boundaries and numerical estimation of lane curvature and vehicle position.

I will discuss the detailed implementation of these steps in the following

1.Camera Calibration

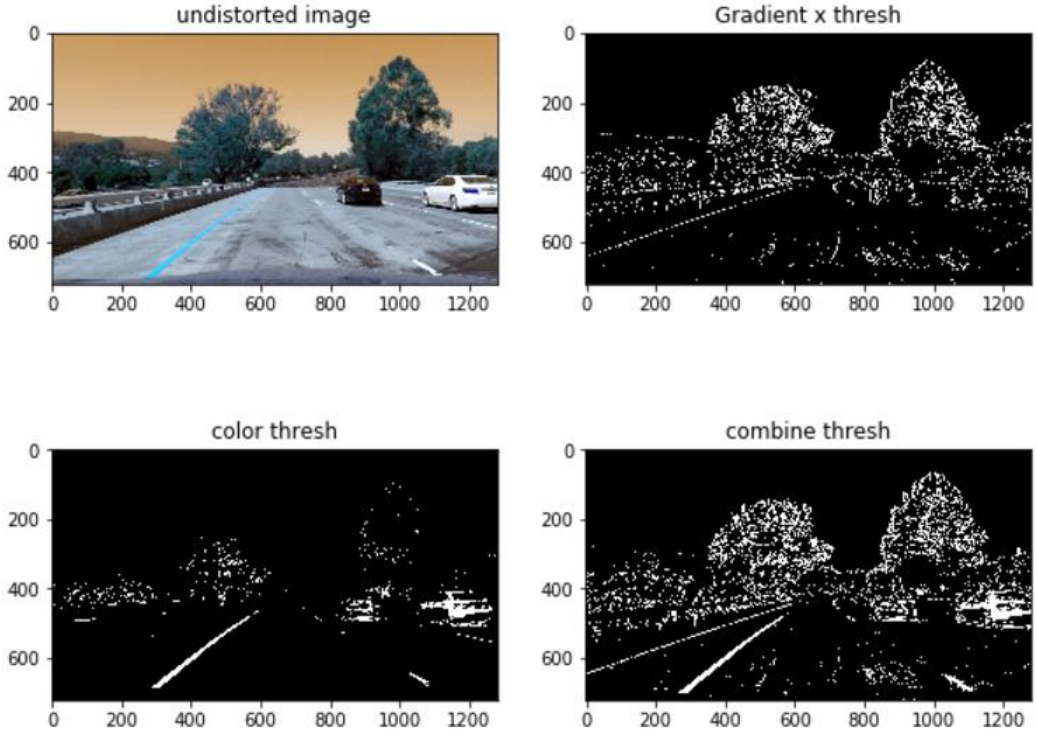
The camera has radial distortion and tangential distortion. In order to correct these optical distortion,we need to compute the camera matrix using several object points and image points.The image points are detected as chess board corners and the object points are just the coordinates of the chess board.I used the opencv function cv2.findChessboardCorner() and cv2.undistort() to perform camera calibration.Here is the result of an example image.



2.Pipeline

Then I use color gradients and color threshold to produce binary images for further detection. There are many methods and parameters to choose such as color gradient along x direction and y direction, RGB color threshold and HLS color threshold. After a series of experiments and parameter tuning,I ended up to choose color gradient along x axis and HLS color threshold along saturation channel.The threshold parameters were (20,75) for sobel x threshold and (150,255) for saturation channel.

The result of a test images is like following:

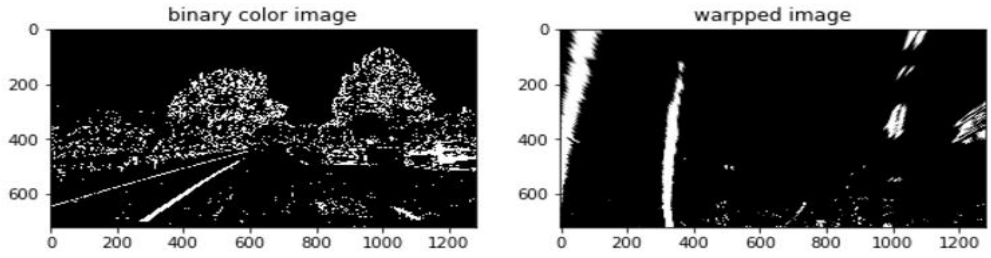


3.Perspective Transformation

In order to fit a polynomial to the lane line, we need to perform the perspective transformation to birds eyes view images.We need four source points and four destination points which represent polygons in source and output image to compute the transform matrix and inverse matrix.Here are the points I choose.

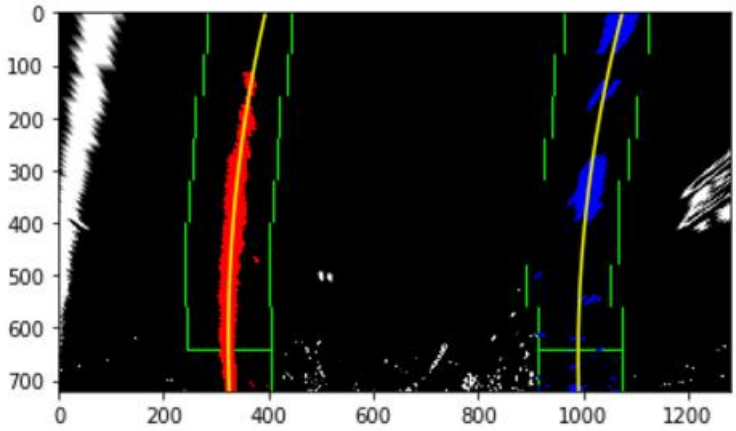
```
src = np.float32([[293, 668], [587, 458], [703, 458], [1028, 668]])
dst = np.float32([[310, 720], [310, 0],[950, 0], [950, 720]])
```

After perspective transform,the warpped image look likes this



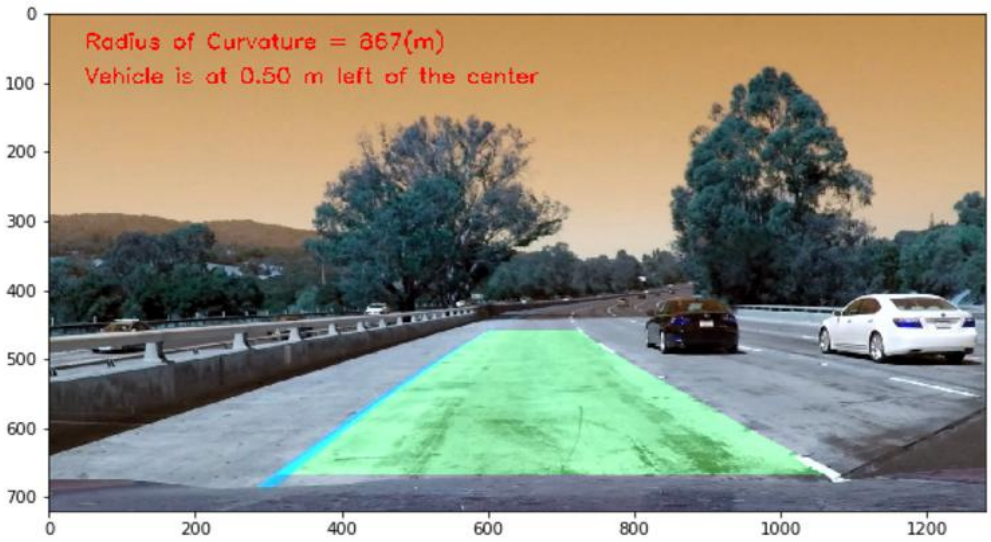
4.Fitting a polynomial to the lane

In order to fit a polynomial to the lane line, I transformed the warpped image to histogram and employed windows slide method to detect the lane line.The maximum peak in the bottom of the histogram is detected as the start of lane line.Then we can search for the curve of lane line and fit a quadratic polynomial.This method output such result in a test image.



5.Compute the curvature radius and the position of the vehicle

After acquiring the parameters of the polynomial, we can use the mathematical formation to compute the radius of curvature of the lane.Assuming the camera was installed in the middle of the vehicle,we can also get the position of the car relative to the center of the lane line.Here is a result of a test image.



6.Video production

Using the pipeline described above,we can produce the output video by applying the processes on each frame.Here is the video result

Discussion

The pipeline discussed above is a simple and fast method to detect lane lines but the result is not robust enough when performing at the challenge video.The shadow line and different lightness condition are quite tricky to be filtered for me.I think more computer vision knowledge is needed to get more robust method.I also think deep learning model is also a good choice for lane line detection as long as enough training data can be acquired.