Advanced Lane Finding

Marcelo Eduardo Pederiva

Report

Camera Calibration

The first step of the advance lane finding is to calibrate the camera distortion. I used all images of the calibration folder to obtain the object points and image points. With this values I could use the cv2.calibrateCamera() to obtain the mtx and dist variables, consequently undistort the image by cv2.undistort() (Figure 1).

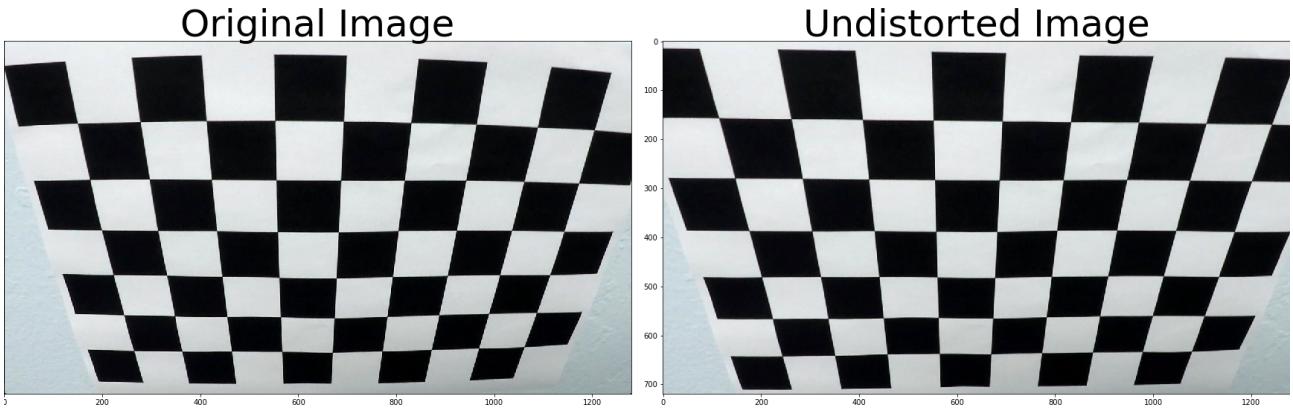

Figure 1 – Camera Calibration

Pipeline

Starting the pipeline, I receive the original image as input and with the calibration variables (mtx and dist) I apply the distortion correction to the image (Figure 2).

Figure 2 – Distortion Corretion

In the second step I apply a filter to the image to transform in a binary image with the line lanes highlighted.

After try to use only Color Transform or Gradients, I observed that each method got a better result in different images. So I combine this two filters to obtain a better result of the lines on the road (Figure 3).

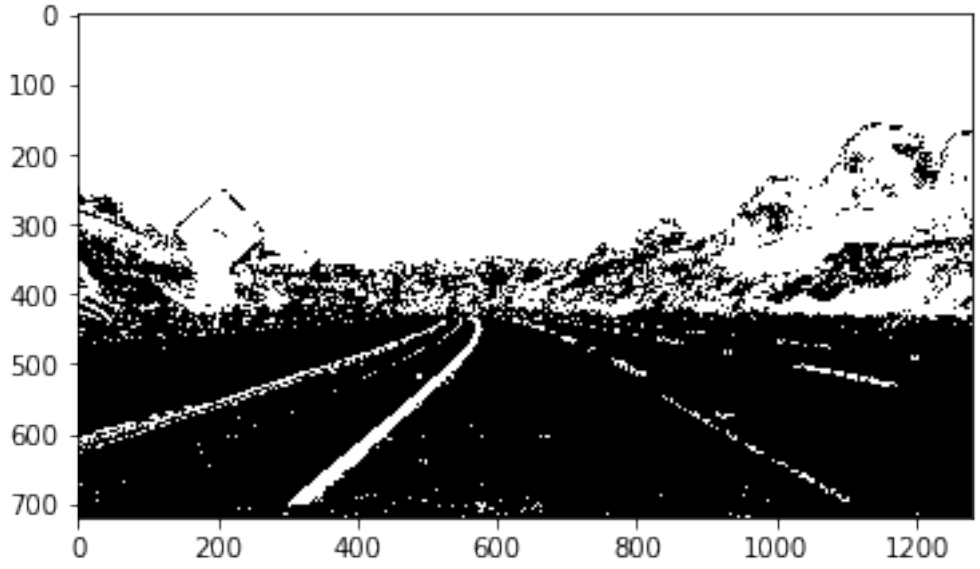

Figure 3 – Binary Image

Now with the image binary I apply the Perspective Transform to observe the curvature of the road.

There are many points to chose in image to transform, I used the Straight Lines images to calibrate the transformation. I try to be precise in the point of the binary image to decrease the error of transform(Table 1)(Figure 4).

Source	Destination
580.815 , 460.748	300 , 0
707.973 , 460.748	970 , 0
243.095 , 691.82	300 , 720
1059.37 , 691.82	970 , 720

Table1 – Source and Destination of Perspective Transform

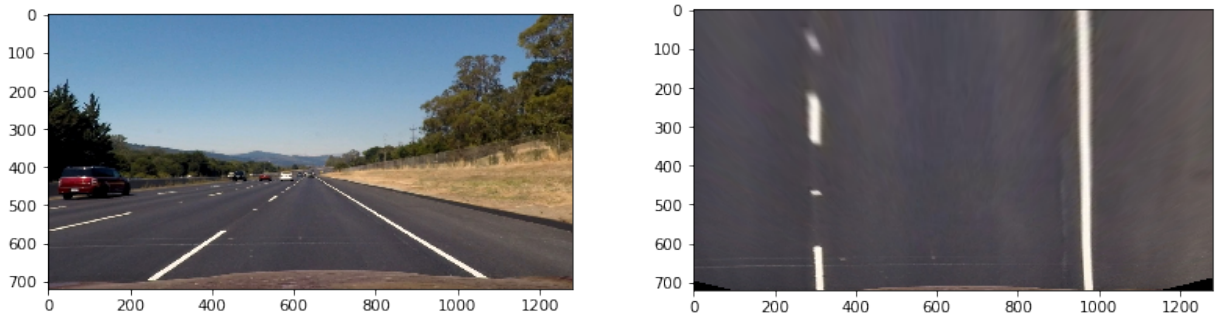

Figure 4 – Example of Perspective Transform in an image of Straight Lines

With the transform I have a warped image of the binary image (Figure 5)

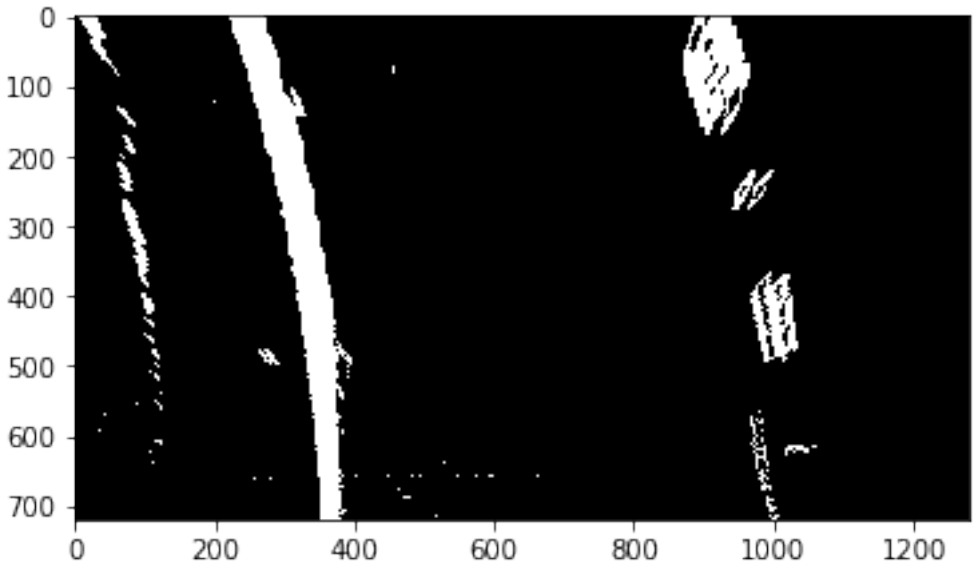

Figure 5 – Warped image of Binary Image

In the next step I use the warped image to find the lane-line and calculate the polynomial that represents each curve(left and right). By measuring the white pixels in the left and right side of the image, I fit a polynomial curve to represent this pixels (Figure 6).

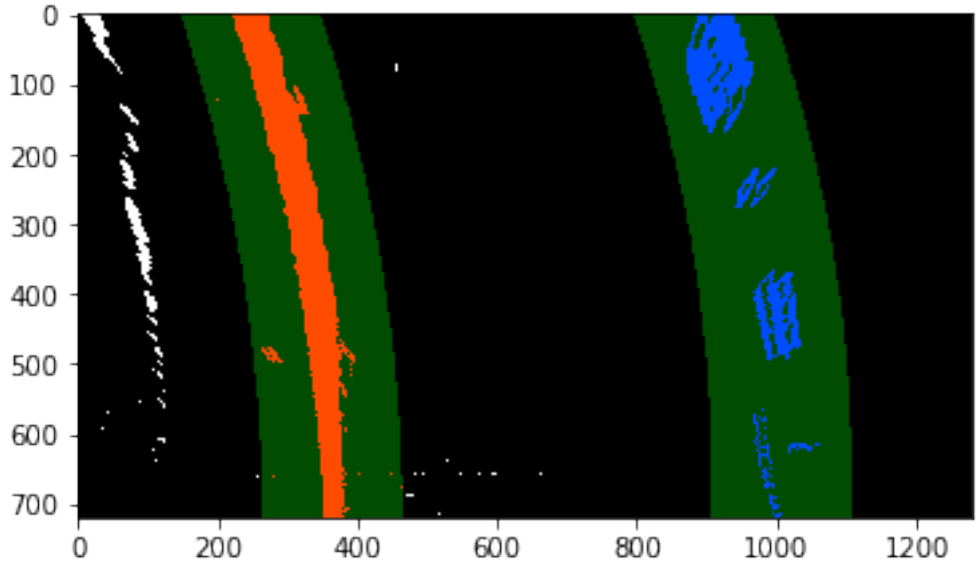

Figure 6 – Lane lines detection

By the values of each polynomial curve, I measure the Radius of the Street by the Equation 1. Furthermore, with the Left Radius and Right Radius in pixel unit I convert the pixel unit to meters unit by the Equation 2, so with the Average of them I have the Radius of Curvature in meters. Additionally, calculating the average of the Left and Right radius in pixel and subtracting this average (in pixel unit) to the Left Radius (pixel unit) I obtain the distance of the left line in pixel units, so I convert this distance to meters and show on the screen (Figure 7).

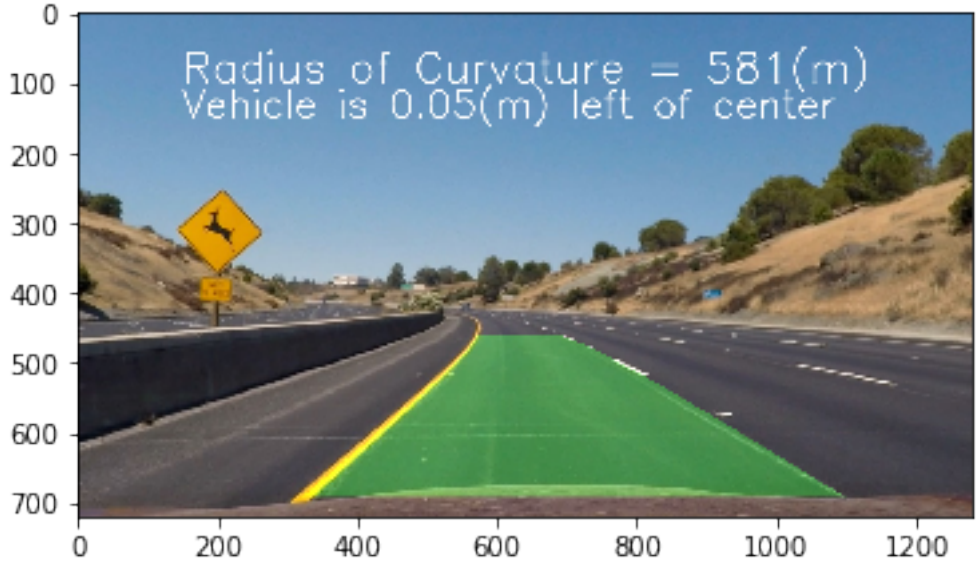

Figure 7 – Final Image

Video

There is link to a video of my final result.

Advance Lane Fiding Video

Discussion

In this project I put a condition to the values of each polynomial. If the one of the curves have a different curvature or if the curvature is too out of the previous I didn't compute this polynomials. I took the average of the last 3 good lines and show this average.

In my results I observe that the radius of curvature looks good by expected, but the distance of the left line have some bad values.

The wall on the left side of the image create a confusion to the detect line, to solve this problem I delete this side of the warped image. However, this solution it's not the smartest way to solve the problem. Maybe changing the Thresholds of the Binary image or filtering better the variables of Detecting Lane script would get a better result from this problem.

Another idea to find a better average of the lanes detecting is to make a variable average with different weight of each lane detected, high weight to the recent lane and low weight to the old lane.