Advanced Lane Finding

Aims

The main aim of this project is to accurately find the lanes on the road using the camera image mounted in the car. Based on the camera images the road lanes are found using traditional computer vision techniques. The project is performed using the following steps:

1. Computation of the camera calibration matrix and distortion coefficients using given a set of chessboard images.
2. Application a distortion correction to raw images.
3. Using of color transform, gradients, etc., to create a threshold binary image.
4. Application of perspective transform to obtain bird's eye view of the binary image
5. Detection of lane pixels and fitting of the lane boundary.
6. Vehicle position and lane curvature determination.
7. Warp the detected lane boundaries back onto the original image.
8. Displaying the output with determined lane markings, estimated lane curvature, and the vehicle's position

Camera Calibration

The problem with camera images is that it can have distortions. In order to remove the distortions from the camera should be calibrated. The distortions can be removed using distortion coefficients and the camera matrix. The distortion coefficients and the camera calibration can be calculated if a picture taken in the camera contains an object whose position is known. In order to do this chessboard is used. Several images of a standard chess board are taken from different angles and distances. Since points of the corners in the chessboard are known, which are called as object points, they are compared with the corners found in the image, which are called as image points. The opencv library in python provides in built functions to find the corners of the chessboard and to calibrate the camera. The first few cells in the jupyter notebook titled "*AdvancedLaneFinding.ipynb*" shows camera calibration. Several chessboard images are read, "*cv2.findchessboardcorners()*" function is used to find the image points then the object points and the image point are feed to "*cv2.calibrateCameera()*" function to find the distortion coefficients and camera matrix. The distortion coefficients and the camera matrix, which are used later to remove distortions from the road images, are then stored in a pickle file "CameraCalibration.p". The below picture shows an example of distorted chessboard image and the same image of the chessboard with its distortions removed. It can clearly be seen that the radial distortions, which makes the edges of the chessboard curved are removed in the undistorted image.

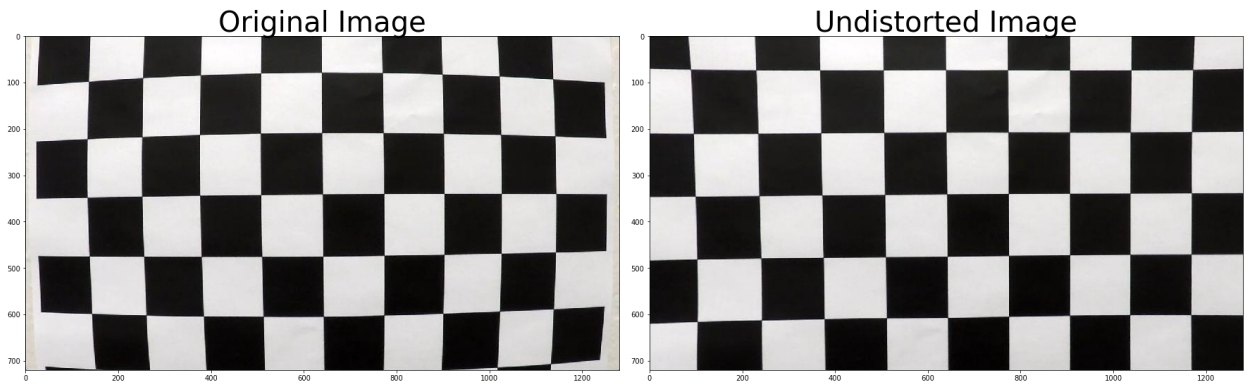


Fig 1: Example of distortion removal from chessboard image

The picture below shows the distortion removal in one of the test images used to find the lanes. There is not so much of the difference between the distorted and undistorted images but there are small differences such as the position of the white car and the shape of car's engine compartment.

Fig 2: Example of distortion removal in one of the test images

Binary Image

In order to detect lanes on the road from the images the features in the image such lines should be extracted. In the first project the lanes were detected using canny edge detections algorithm which was working well for some images but it was not able accurately detect edges in cases such as bad lighting, conditions, images with shadows etc. In this project several methods are used to extract lane markings which are gradient detections, color transformations. In the gradient detections the sobel transform is used to calculate the gradient images along the x and y directions. Based on the gradient of images a predefined values are chosen in the gradient values of the pixels within the range are set to 1 and the rest of the pixels are set to zero giving a binary image. The HLS and HSV color transform are applied and the binary images are obtained for different color channels. Finally a binary image is obtained using a

combination of different methods. The table below shows different methods used for getting the binary image.

Method	Threshold Value	Reason
X gradient	20 - 100	Detects edges in the X direction.
Y gradient	20 -100	Detects edges in the Y direction
Magnitude of X and Y	70 – 100	Detects edges in both X and Y directions
V channel (HSV)	150 – 255	Detects shadows and dark regions in the image
L channel (HLS)	50 – 255	Detects shadows and dark regions in the image
S channel (HLS)	100 – 255	Detects the regions based on the level of color saturation. The road lanes have saturated color so it is useful to detect the lanes, especially continuous lane.

The picture shows the example of the binary image compared to a test image.

Fig 3: Binary image using gradient and color transform

Perspective Transform

When perspective transformation can be applied to an image a new image from a different perspective i.e. different view point can be obtained. Once the binary image showing the road lanes are obtained perspective transform can be applied to get a bird's eye view of the road lanes. This helps to accurately detect the position of the lane and also more information such as curvature of the road, direction of curvature etc. In order to apply perspective transform the transformation matrix should be calculated by choosing four points in the original image and where these points would be located in the transformed image. Then the transformation matrix can be used to warp the image to a new perspective. The opencv library has functions to calculate the transformation matrix "*cv2.getPerspectiveTransform()*"and to warp an image to a new perspective "*cv2.warpPerspective()*". In order to get the points for applying perspective transform the lane detection algorithm in the project "Finding lane lines" is used. When the binary image is feed to this algorithm it detects lines on the image using Hough transforms in a masked region and detects the lane lines, which are a set of four points two for each lane. These four points are the source points for the perspective transform. The destination points are calculated based on the simple assumption that the lane lines are straight and parallel when looked from a bird's eye view. The pictures below shows the result of perspective transform on a test image and the binary threshold image.

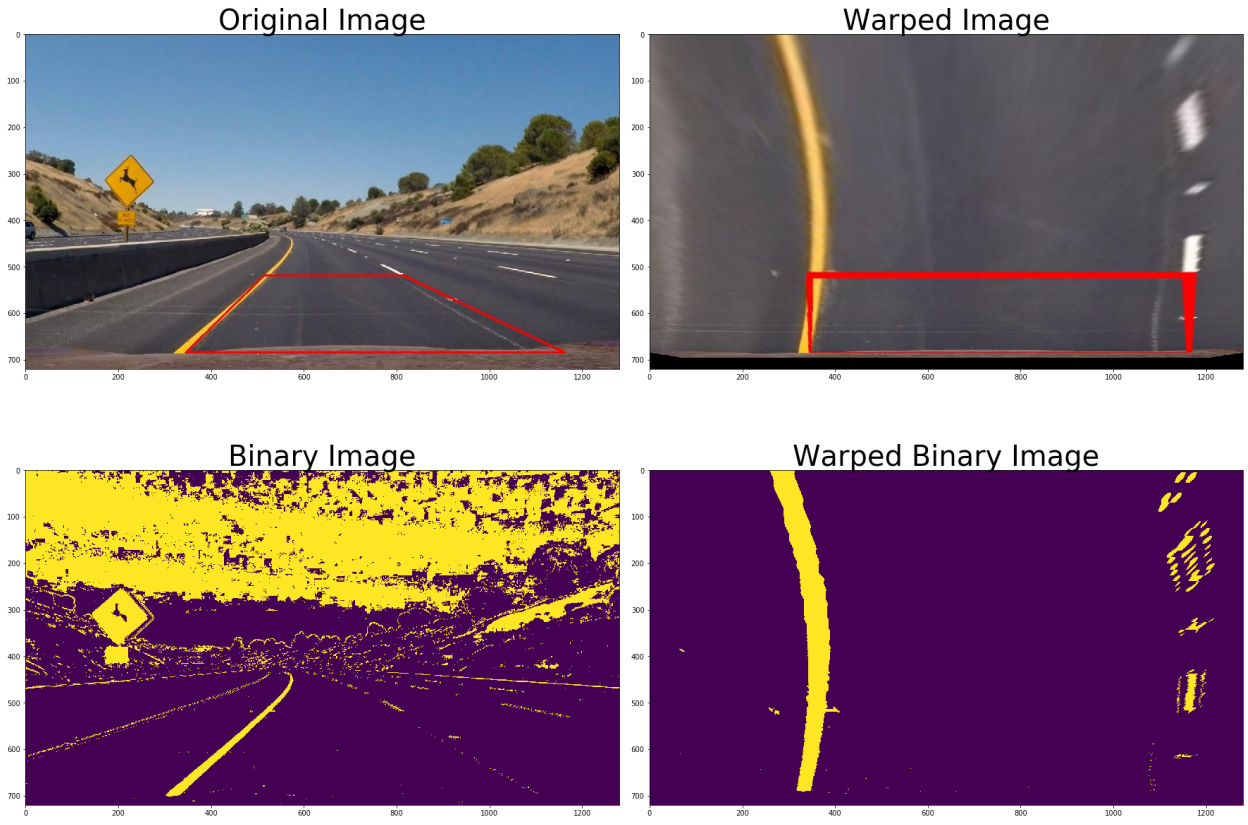


Fig 4: Example of Perspective Transform applied on a test image and binary Threshold image

When the source and destination points are switched in the calculation of the transformation matrix an inverse transformation matrix can be obtained. The inverse transformation matrix is helpful to project the lanes to the original image after its detection in the top view.

Finding Lanes – Sliding Window Method

The perspective transform of the binary image gives the top view of the road lanes. Using the top view the position of road lanes can easily be found. There are two sliding window search methods discussed in the lectures to determine the location of lanes which are histogram based and convolution based method. In this project the histograms based sliding window search is used because it is more robust to noises compared to the convolution method. Ideally the binary image has pixel values equal to one in the locations of lanes and pixel values zero elsewhere. If histogram along x direction, which is the cumulative sum of the pixel values along the x direction, is taken on the lower half of the image the base location of the lanes would correspond to the peak in the histogram. This is the starting point of the sliding window search method based on histogram.

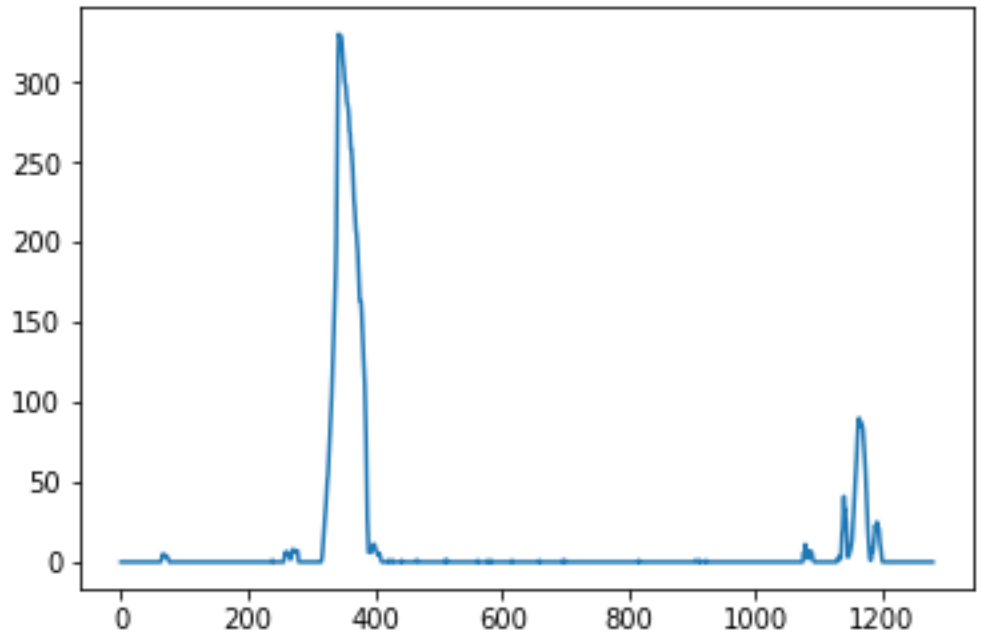


Fig 5: Histogram in lower half of test image along x direction

Using numpy library's methods "*nonzerox()*" and "*nonzeroy()*" the position of non – zero pixels in the x and y direction are determined. The image is then split into two along x direction to determine the position of left and right lane. Both the left and right section of images is split into several vertical sections called windows. In each of these windows the central position of non-zero pixels are determined along x direction. Since position of the lanes in known for the base of the image position of windows is searched from bottom of the image to the top in the vicinity of histogram peaks at the base.

At the end the position of cluster of non-zero pixels forming the left and right lanes are known. Using this information a second order polynomial is fit, which is the location of the lanes in the top view.

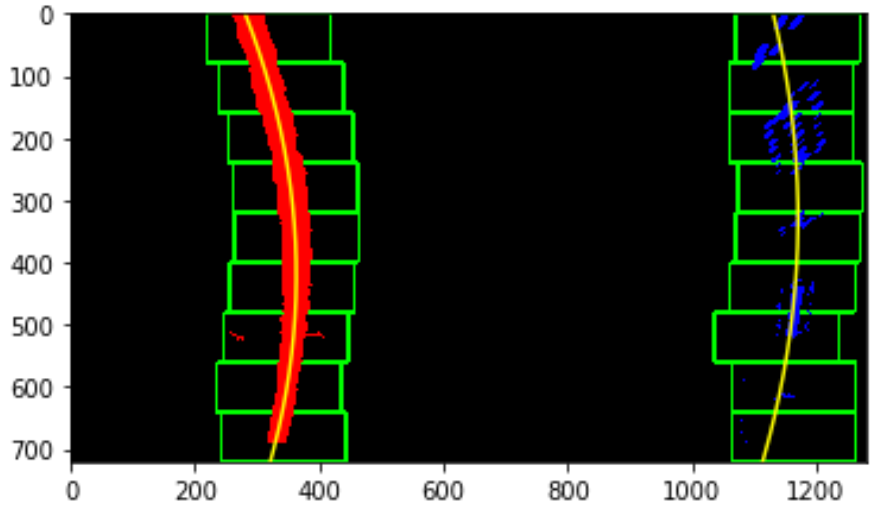
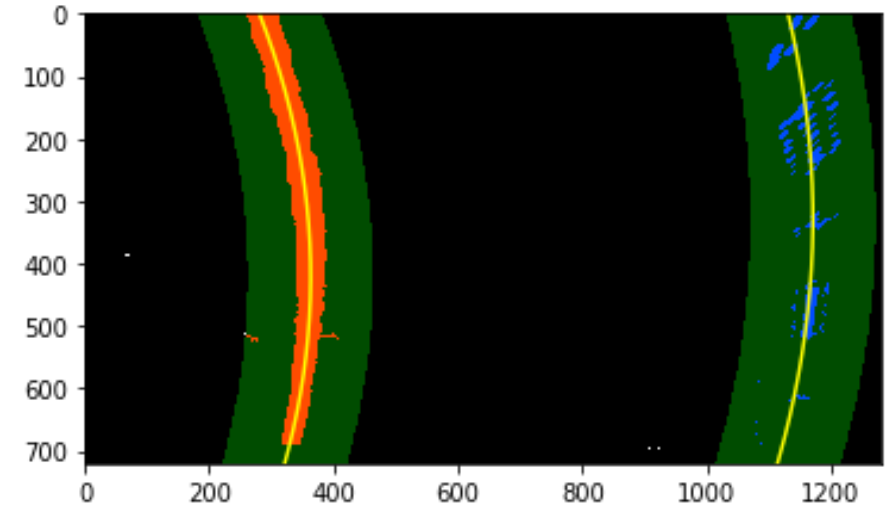


Fig 6: Output of Sliding Window Search Method

The above image shows the output of sliding window search. The algorithm is able to fit a curve for each of the lane. Later the curve found using sliding window method is applied with perspective transform using inverse transformation matrix to change the lanes to original viewpoint. In case of videos the sliding window search is required only in the initial frame. In the subsequent frames since the position of lanes are known for previous frame its location can be searched in the vicinity of lane location in the previous frame.



.

Fig 7: Output of Lane search using the position of lanes in previous frame

Calculation of Radius of Curvature

The radius of curvature is calculated for each of the lanes using the equation of the lane curve. The equation of curve representing the lanes is of form shown in the equation below. It is to be noted that the curve equation has x coordinate as dependent variable which means that the equation gives the location of lanes in X direction for each pixel in Y direction.

$$f(y) = X = aY^2 + bY + c$$

The radius of curvature of a curve is given by the equations:

$$R_{curvature} = \frac{(1 + f^{'}(Y)^2)^{\frac{3}{2}}}{f^{''}(Y)}$$

$$f^{'}(Y) = 2aY + b$$

$$f^{''}(Y) = 2a$$

The radius of curvature for the current image is calculated at the bottom of the image i.e. for Y value equal to the size of image in Y direction.

Position of the Car

The lateral position of the car with respect to the centre of the road can be found using the position of the lanes at the bottom of the image. The center of the image in X direction corresponds to the center of the car since the camera is placed in the center of the car. The center point between the right and left lane position at the bottom of the image is the road center. The difference between the car center and the road center gives the car's lateral position with respect to the road center.

It has to be noted that both the car's position and radius of curvature have to be corrected by a factor to convert from pixels to meters. The correction factor is calculated based on the assumption given in the lecture. That is 720 pixels in Y direction correspond to 30 m in real world and 700 pixels in X direction corresponds to 3.7 m.

Final Output

The final output of this project is the road image where the lanes detected are marked and the radius of the curvature and the car's lateral position is displayed. The road lanes found in the top view is

transformed to original view point using the inverse perspective transformation matrix. The picture below shows the output of project.

Fig 8: Project Output

Applying the Pipeline to the Video

The above discussed pipeline is applied to a video using moviepy library using the function fl_image. The whole pipeline is programmed as a function with raw image as the input argument and the output image as the return argument. For processing of the video the perspective transformation matrix is calculated for first frame and it is used for the entire video. Since the output of the pipeline depends on several parameters such as threshold parameters used for binary image determination, points used in perspective transform etc. the result could be wrong for certain frames. In order to check if the output is correct a function called plausibility check is used. This function is discussed in detail in the section below. The positions of the lanes are determined using sliding window search method for the first frame and also for the frames after the implausible frames. For the rest of the frames the positions of the lanes are determined using the position of lanes in the previous frames. There are situations in the video where the pipeline is not able to detect the lanes accurately due to poor lighting, improper perspective transform. This causes the lane detection to be discontinuous in the video. This is partly reduced using plausibility check function where the previous locations of the lanes are retained for false detection. In order to reduce the effect of noisy frames further the lane positions are smoothened by storing the positions of n frames and displaying the averaged position.

Plausibility Check

The lanes detected are checked for plausibility using the function "*PlausibilityCheck()".* This function return true value only if the following conditions are satisfied in the detected lanes

1. Both the lanes have same direction of curvature.
2. The distance between the left and right lanes at top and bottom of the image are greater than a threshold value.
3. The lanes are approximately parallel which is checked of measuring the distance between left and right lanes at different location in Y direction and comparing them.
4. The lanes are not intersecting each other

The lane detection algorithm is able to accurately detect the lanes in the project video.

Limitations and Further Improvement

The lane detection algorithm in this project is sensitive to the parameters for determination of binary image. The algorithm is not able to detect the lanes properly in the challenge videos. The parameters of the algorithm should be tuned further to make it robust for wide range of road conditions. As mentioned before the points used in the perspective transform is based on the first frame of the video. The algorithm is also sensitive to the points used in the perspective transform. If the points of perspective transform are not chosen properly the lanes in the top view image are not parallel, the top view contains noise such as road edge etc and also sometimes some of the lane segments go missing. There were several attempts taken to dynamically change the points used in the perspective transform but it failed to give proper result. The algorithm should be extended to dynamically choose the points used in the perspective transform.