Lane Detection – Advanced Computer vision methods

The goal of the project is to develop a software pipeline to identify lane boundaries from the repository of front facing camera video, camera calibration images, test road images etc.

The goals / Steps –

1. Import and initialize the packages needed in the project,
2. Compute the camera calibration matrix and distortion coefficients given a set of chessboard images,
3. Apply a distortion correction to raw images,
4. Use color transforms, gradients, etc., to create a thresholded binary image,
5. Apply a perspective transform to rectify binary image ("birds-eye view"),
6. Detect lane pixels and fit to find the lane boundary,
7. Determine the curvature of the lane, and vehicle position with respect to center,
8. Warp the detected lane boundaries back onto the original image,
9. Display lane boundaries and numerical estimation of lane curvature and vehicle position,
10. Run pipeline in a video.

Step 1.

Importing and initializing libraries and packages needed for the project.

- Open CV
- Matplotlib
- Numpy
- MoviePy

Step 2.

Compute the camera calibration matrix using the provided chessboard images.

The function is defined that taken in the chessboard images along with the x and y axis corners.

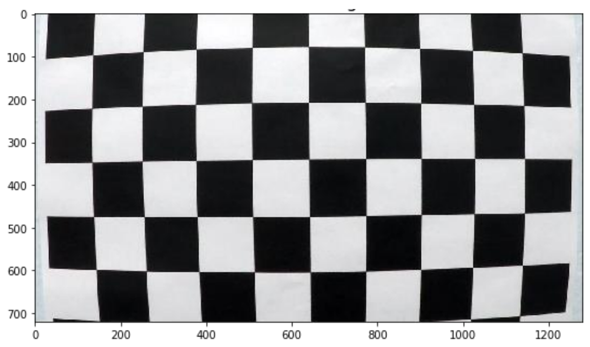
The function uses the cv2.findChessboardCorners & cv2.calibrateCamera function from open CV. The output are the set of camera calibration matrices,

- ret, mtx, dist, rvecs, tvecs

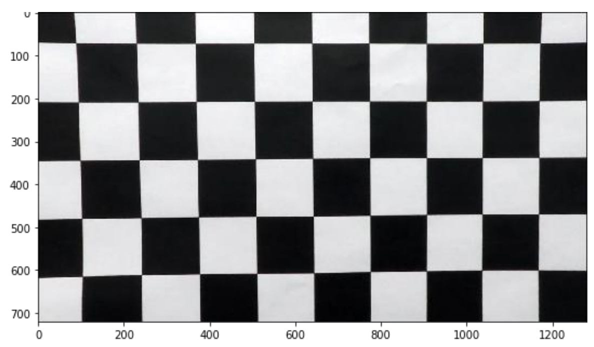
Step 3.

Using the calibration camera calibration function and open cv function – cv2.undistort, the undistorted_img function was defined.

The sample output on the chessboard image looks like –



Original image



Undistorted image

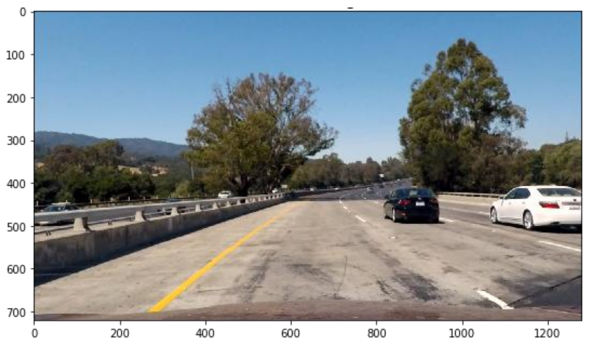
Step 4. Using directional gradient, gradient magnitude, gradient direction and color transform functions to create binary threshold image.

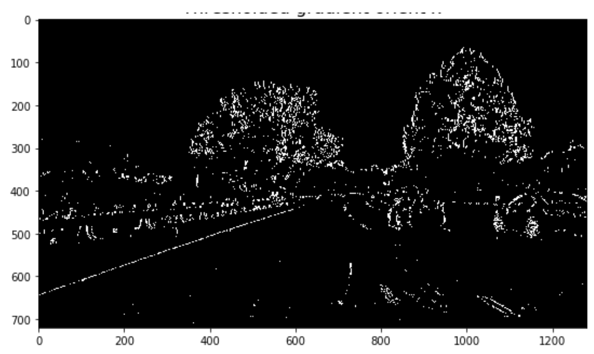
In this step multiple function were defined to determine various gradient parameters (x,y, magnitude, direction and color)

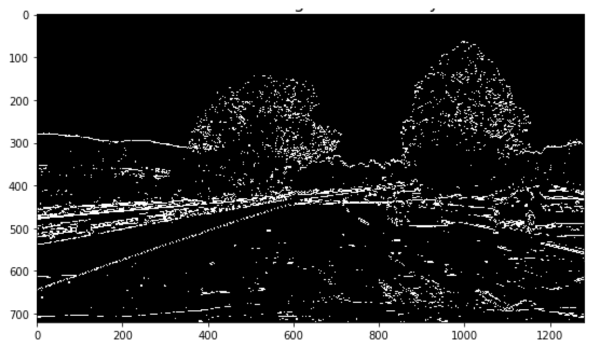
Function –

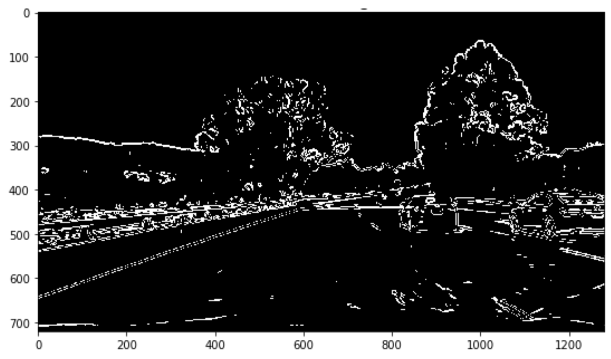
- Calculate directional gradient
- Calculate gradient magnitude
- Calculate gradient direction
- Calculate color threshold
- Combine threshold information to be used to identify lane lines

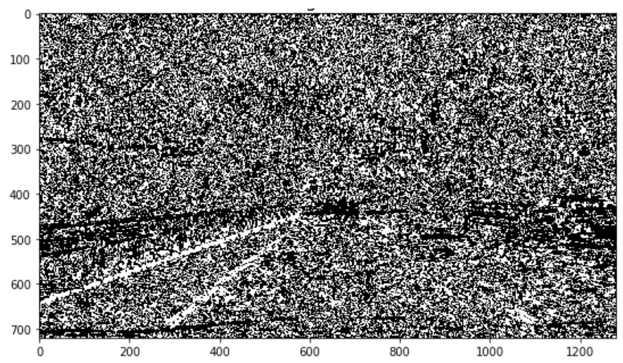
Below are the sample outputs of the functions defined –

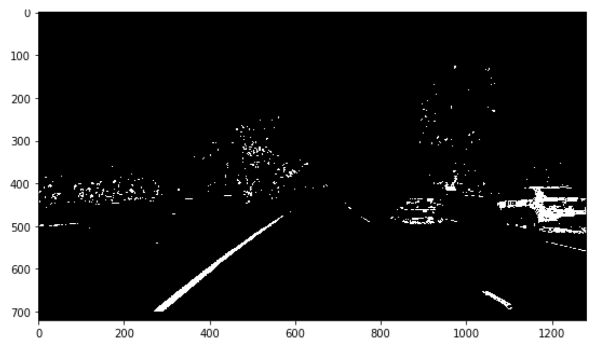

Test image

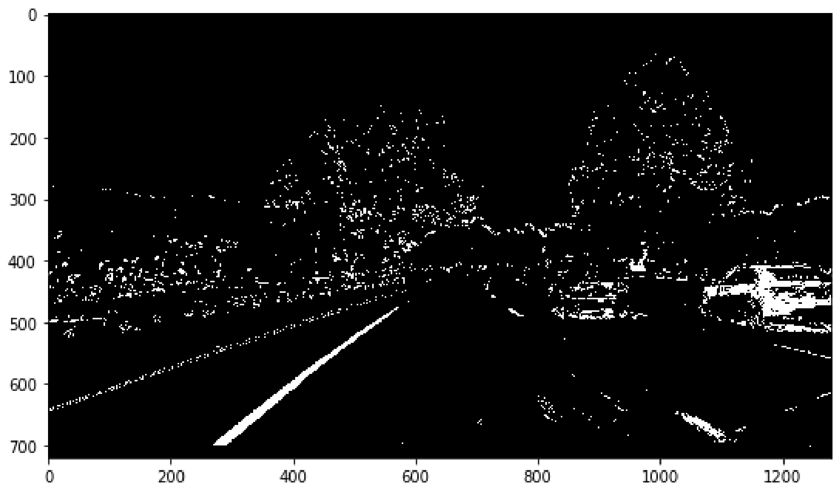

Gradient orientation x


Gradient orientation y


Gradient Magnitude


Gradient Direction

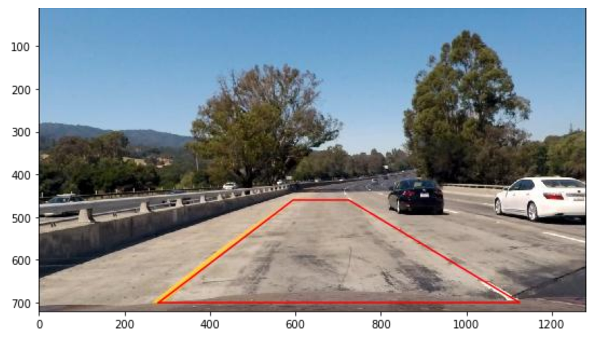

Color Threshold

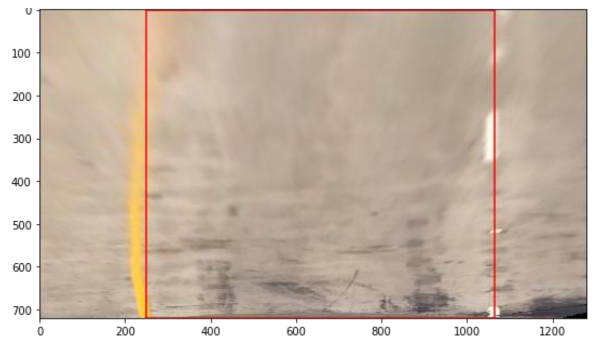

Combined threshold image

Step 5. Apply a perspective transform to rectify binary image.

The function is used to transform the test image into birds eye view image . function inputs the image to be transformed, the coordinates of the targeted section on the source image as well as the image to be transformed.

The open cv function - cv2.getPerspectiveTransform provide the matrices for transformation. Open cv function - cv2.warpPerspective, provide the transformed image.

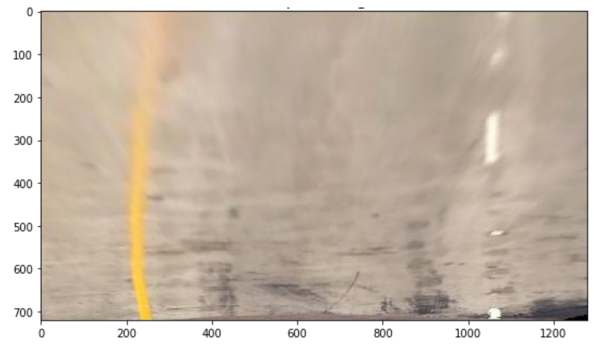

Original image with coordinates

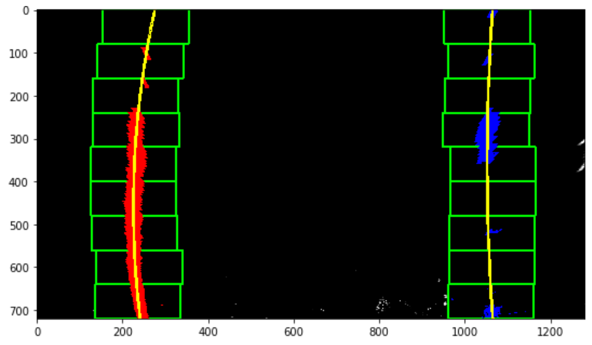

Transformed image with coordinates

Step 6. Detect lane pixels and fit to find the lane boundary,

To detect the lane lines, following steps are followed –

1. Histogram of the lower half of the transformed / warped image.
2. The max values of the histogram from the left and right of the image would provide the identification for the two lane lines.
3. Sliding window technique (as defined in the lectures) is used to map the coordinates of the lane lines.
4. Now, using the coordinates, a second order curve fit is performed to estimates the lane lines as second order polynomials.
5. The function detect_lines defines the above steps.
6. As described in the lectures, once the lane lines are defined in one frame of the video, the lane lines in next frame would be there in the same location, the function detect_similar_lines does that.


Warped image


Detected lane lines

Step 7. Determine the curvature of the lane and the vehicle position

Curvature_radius and car_offset functions are defined to determine the lane curvature and vehicle position.

Step 8 & 9

Warp the detected lane boundaries back onto the original image & Display lane boundaries and numerical estimation of lane curvature and vehicle position on the image.

In this steps, the estimated lane lines are plotted on the original image. Further, the curvature radius and car offset outputs are printed on the original image.

Test image

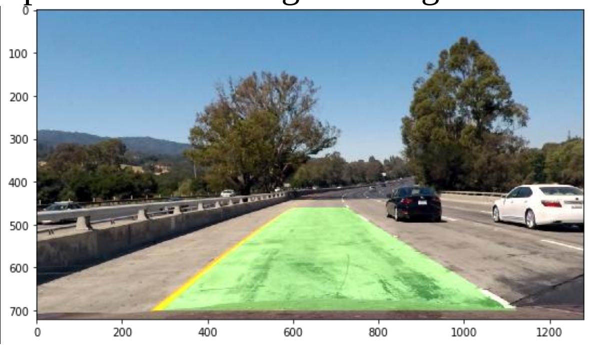
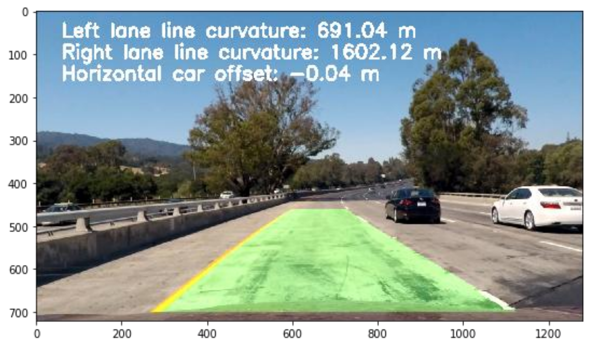


Image with defined lane boundaries


Left lane line curvature: 691.04 m
Right lane line curvature: 1602.12 m
Horizontal car offset: -0.04 m

Output image with lane curvature, car offset printed.

Step 10.

A pipeline function is defined to process the image.

Discussion –

Advanced lane detection algorithm pipeline is generated using advance computer vision algorithms.

Further analysis could be done using deep learning methods of Convolutional Neural nets and semantic segmentation.