Writeup: AdvanceLane finding

Introduction

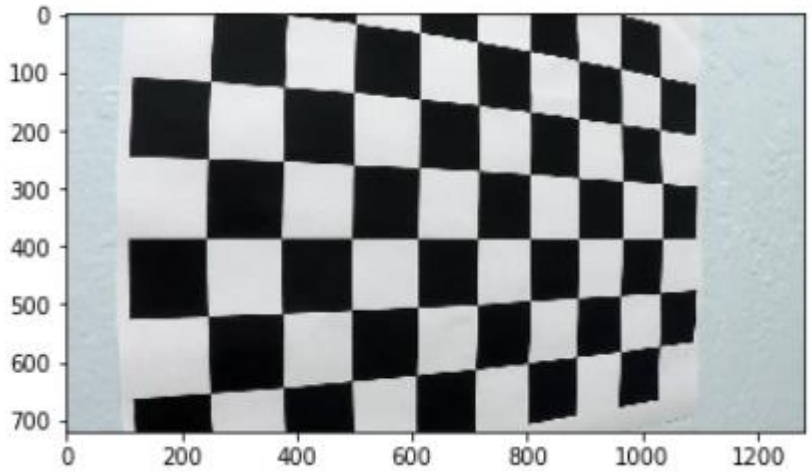
The goals / steps of this project are the following:

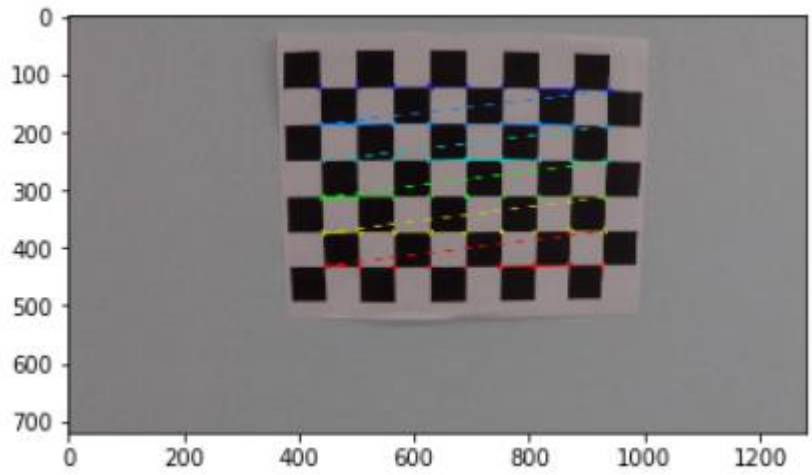
- Compute the camera calibration matrix and distortion coefficients given a set of chessboard images.
- Apply a distortion correction to raw images.
- Use color transforms, gradients, etc., to create a thresholded binary image.
- Apply a perspective transform to rectify binary image ("birds-eye view").
- 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.

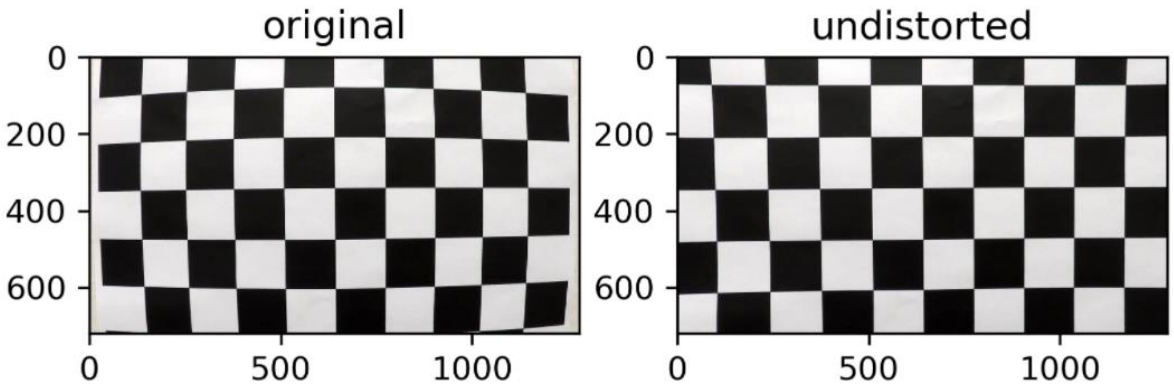
Camera Calibration

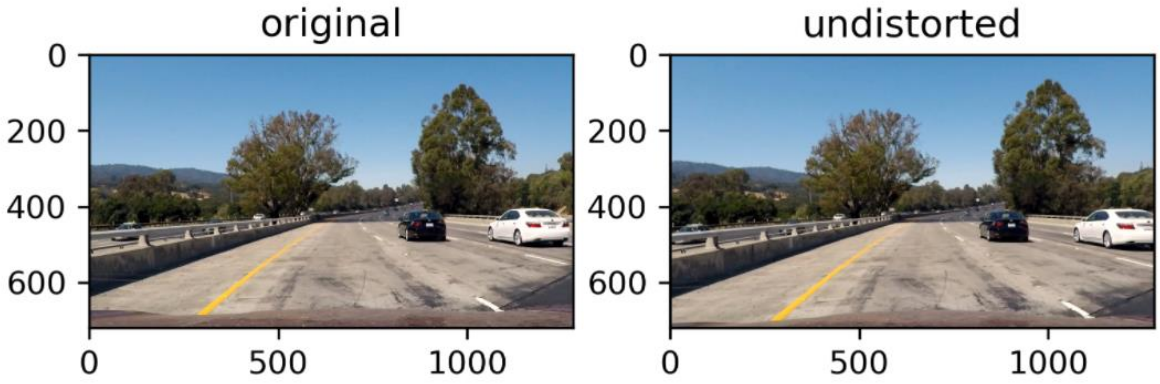
OpenCV functions were used to calculate the camera calibration matrix and distortion coefficients.

- Find the chessboard corners on images provided from project using findChessboardCorners function.
- Based on corners objects, we get camera calibration matrix using calibrateCamera function.
- Transform to the undistorted images using undistort function with camera calibration matrix.









Color space and Threshold

I combined several threshold methods(i.e., HLS color transform, absoulute/magnitude gradients and direction gradients) to create a binary images containing the lane pixels. There is no ground truth here, I tested threshold method with manual parameter tunning. When I calculate the gradient, I directly used converted color space image besides using color thresholded binary image. I tested Gray, HLS and HSV color space, and using HLS color space's S-channel was the best for me.

Here are the some results below:

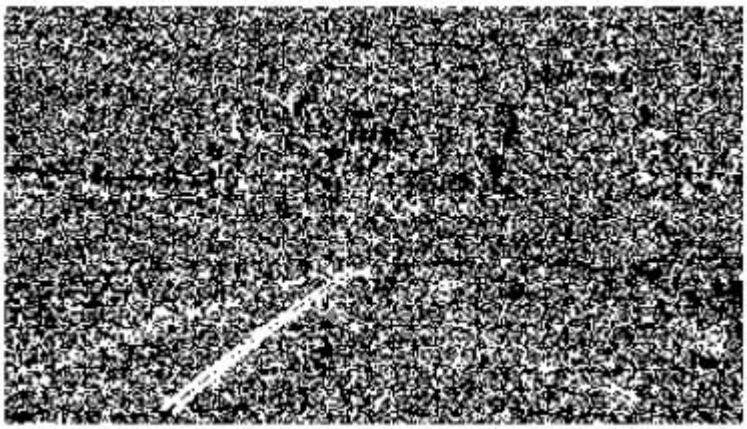
Gradient X

Gradient Y

sobelx_binary

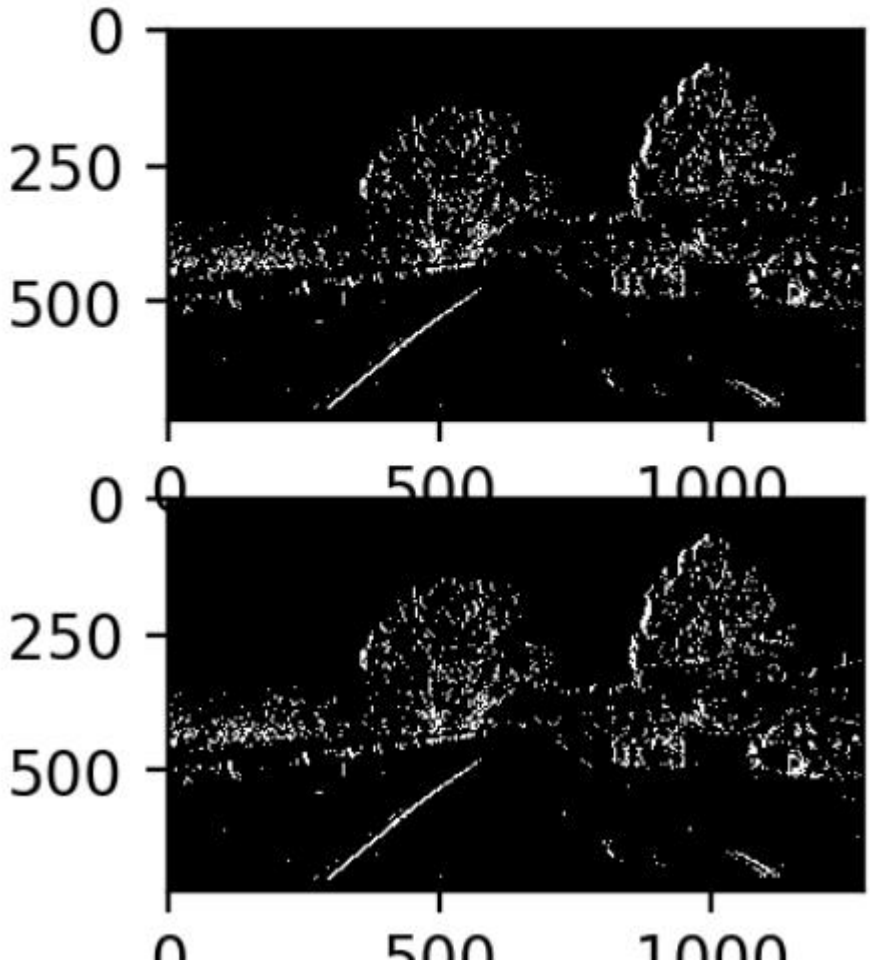
Gradient magnitude

Gradient direction



Yellow threshold

White threshold

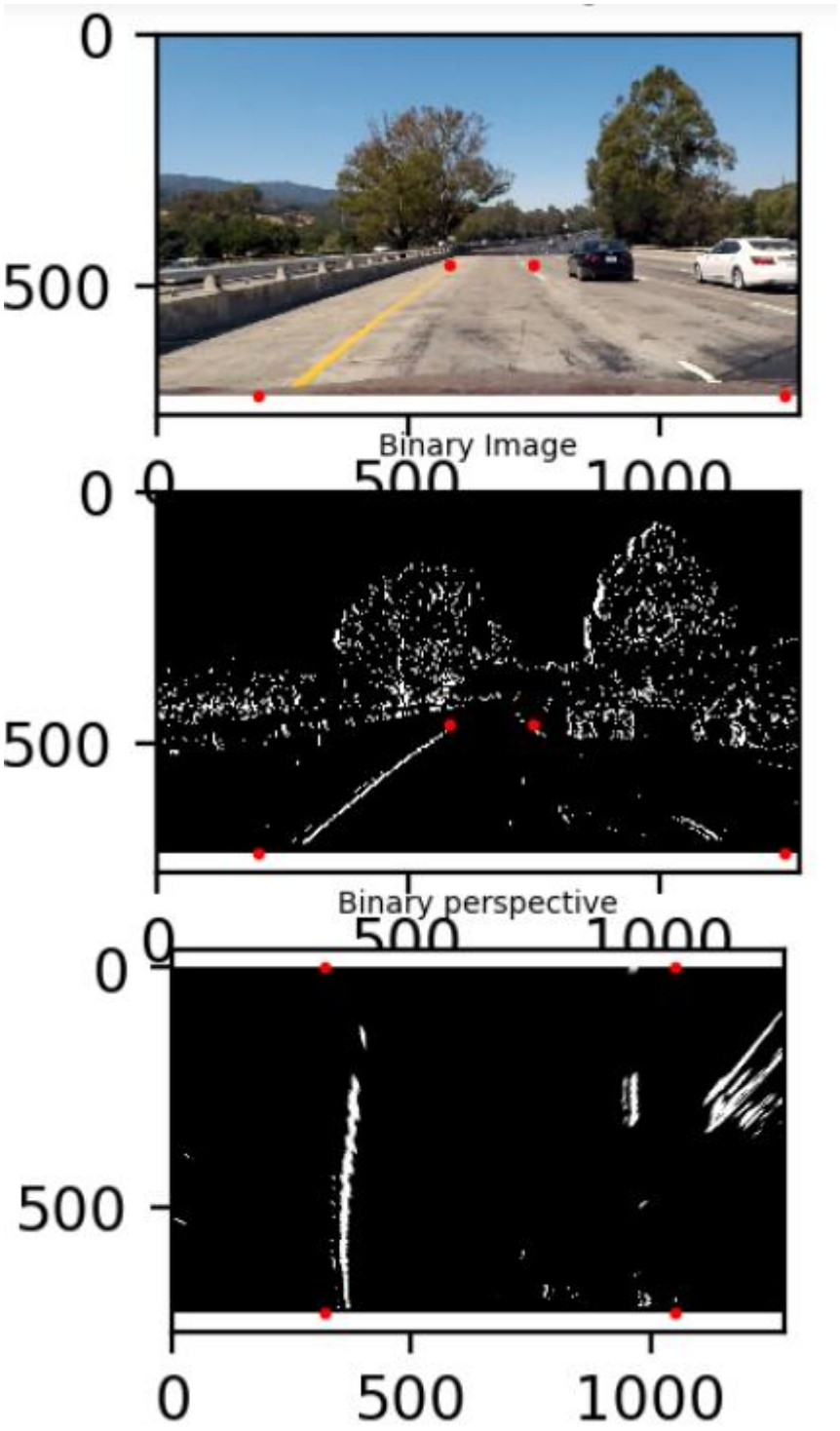


Perspective Transform

I used OpenCV function to correctly rectify each image to a bird-eye view.

- Define the source 4 points of rectangle shape onto the road image. I used the straight-lane image to set perspective source points correctly.
- Find the perspective transform matrix using cv2.getPerspectiveTransform function.

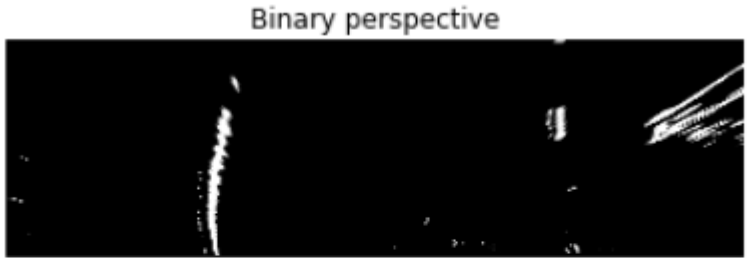
- Based on perspective transform matrix, we get warped images using cv2.warpPerspective function.

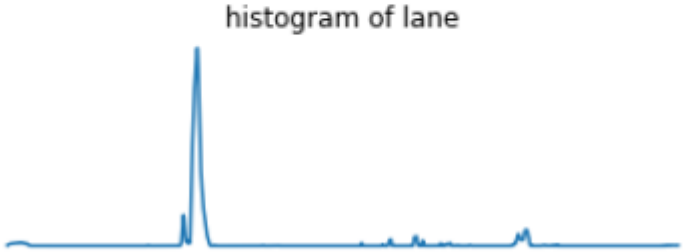


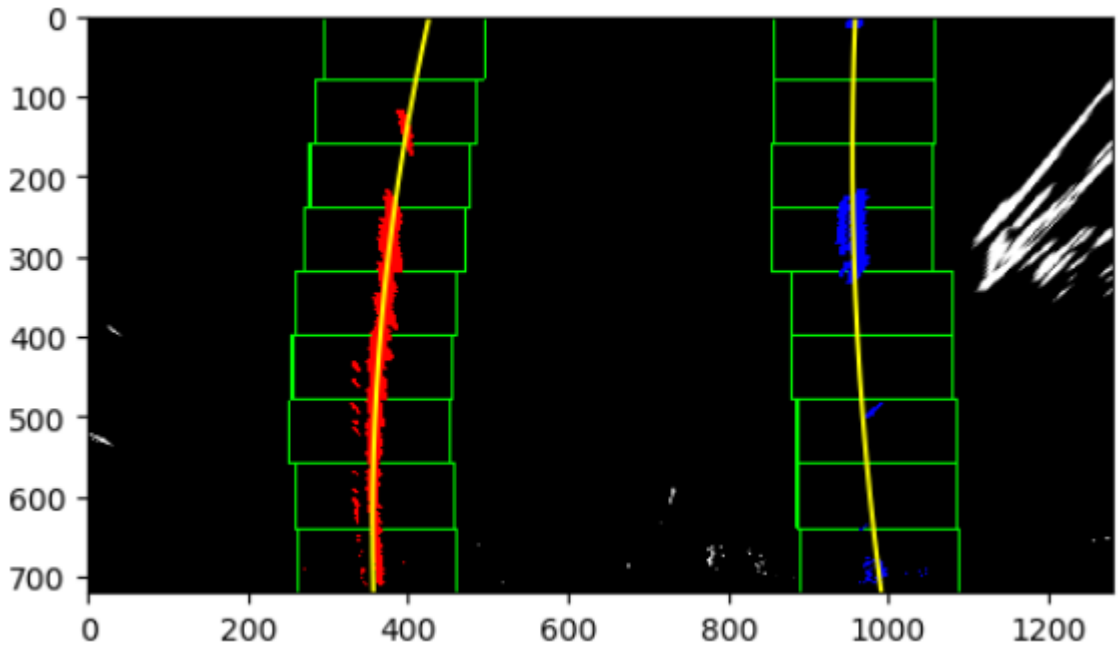
Lane detection and Curve Fitting

Here the idea is to take the measurements of where the lane lines are and estimate how much the road is curving and where the vehicle is located with respect to the center of the lane.

- First I take a histogram along all the columns in the lower half of the images.
- With this histogram I used the sliding window approach to find the places around the line centers in each discrete windows.
- 9 windows used. Margin was 100 pixels.
- Based on each windows center points, It can fit a polynomial to those pixel points.
- Finally, we can get a smooth poly-fitted curve lines.







Calculate Radius of curvature and Position of Vehicle

The radius of curvature at any point x of the function x = f(x) is given as follow:Using this equation for radius of curvature, we can get poly fitted curve's radius. But calculated curve radius based on pixel values, so the radius is not the same as real world values in meters.

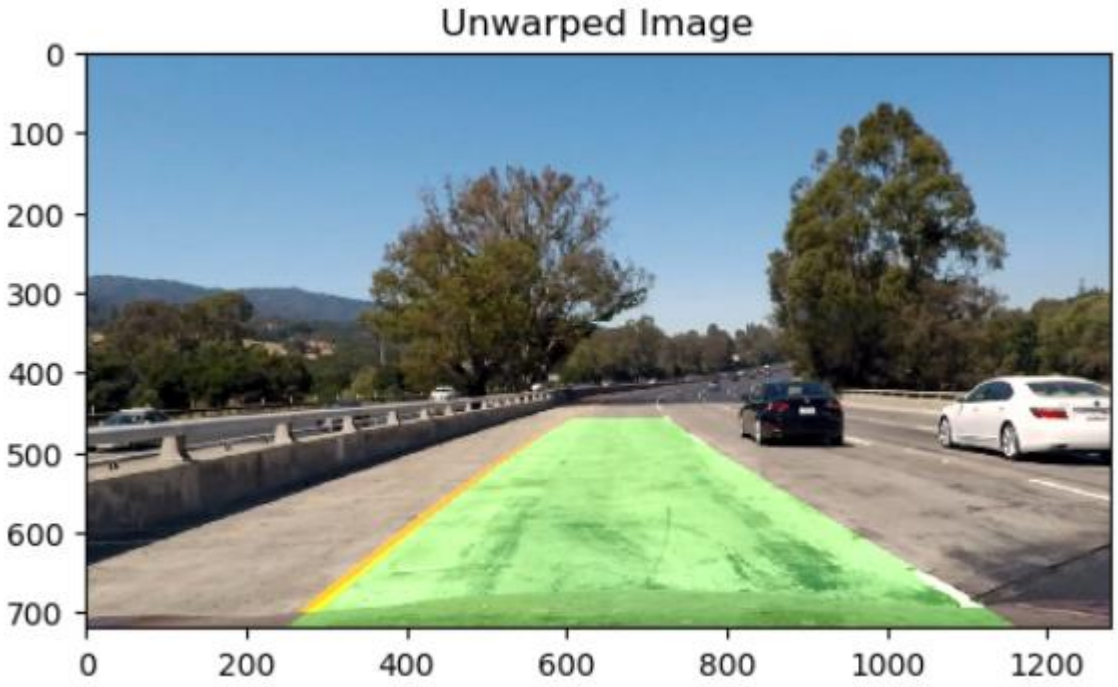
$$R_{curve} = \frac{[1 + (\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

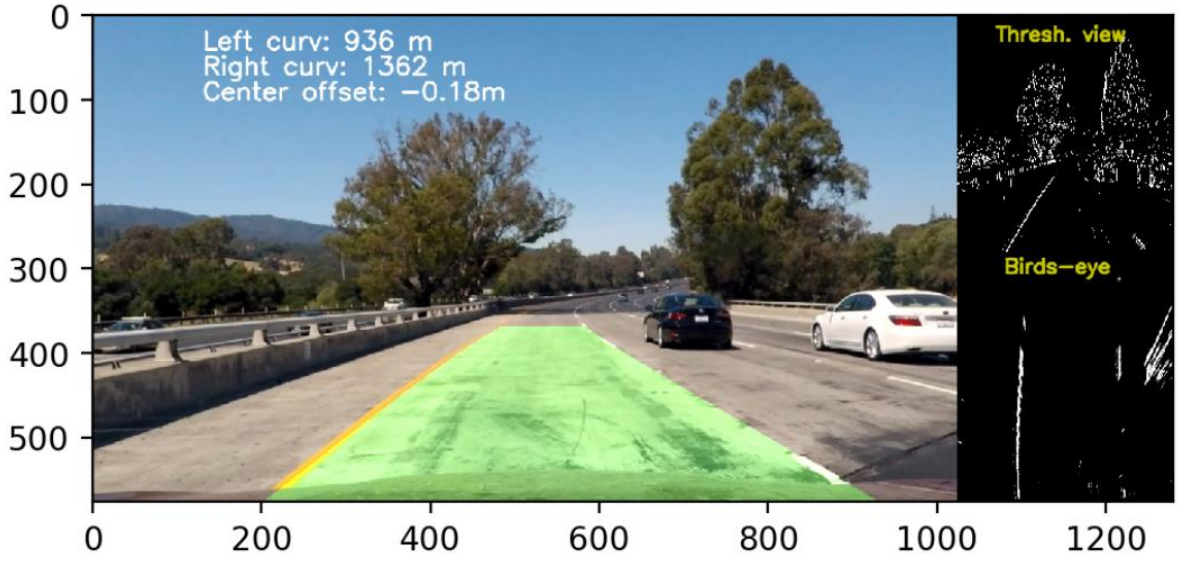
To convert radius to physical value from pixel value, we assume that the lane is about 30 meters long and 3.7 meters wide. Finally, we can get physical value of curve radius using those conversion factors.

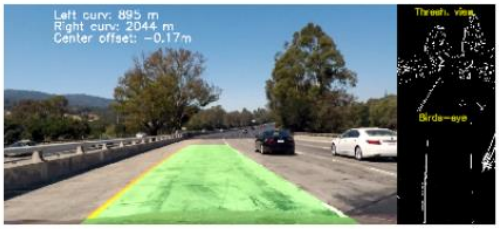
- First we assume that camera is mounted on center of the vehicle.
- To estimate the center offset from the each lane, we can calculate the y-intercept value from the fitting coefficient.(y = ax^2 + bx + c)
- Comparing the estimated center position and middle pixel of the image in y-axis, we can find the center offset values.

Warp Back image and overlay

- The images from the rectified region has been warped back to the original image and plotted to identify the lane boundaries.
- Bird views, threshold views and curve radius values overlayed onto original image.


Unwarped Image
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


Left curv: 936 m
Right curv: 1362 m
Center offset: −0.18m
Thresh. view
Birds−eye
0
100
200
300
400
500
0
200
400
600
800
1000
1200



Result (image, video)

- The image processing pipeline was applied to find the lane lines from images and video.
- The output image here created as a new image where the combination of lanes regions, curve radius and center offset.

`Results Link :` `https://view5639f7e7.udacity-student-workspaces.com/tree/CarND-Advanced-Lane-Lines`

Conclusion

Using advanced computer vision algorithm (Undistortion, Color space transform, Gradient threshold, Sliding windows and Perspective projection), we can easily get the lane detection program. But it's not enough to adapt in real situations. In different roads, weathers and brightness, there are chance to fail detecting lane lines. To come up with this shortcoming, we may use deep learning based feature extraction technique(CNN) in various situation.

- Issues
  - Dashed lane in the far distance from vehicle was not detected well. Because of this, most curve radius of dashed lane has wrong directional curvature. To come up with this I think need to improve the binary threshold function.
  - Updates: I updates threshold binary function based on udacity reviewers suggestions(Yello and White color threshold). The lane detection performance was improved in final result video.
- Improvements
  - Test different color space (YUV, RGB)

- Test gradient threshold through CNN extracted feartures. CNN is powerful fearture extractor from images. I think we may have different results from conventional computer vision technique using pretrained CNN feature as inputs to gradient threshold calculator.