Udacity Self-Driving Car Nanodegree Project 2
—Advanced Lane Finding

-Guisheng Wang

Overview

This project is to perform following steps for land detection,

- 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.

The final output video is in the folder of ‘/test_videos_output’

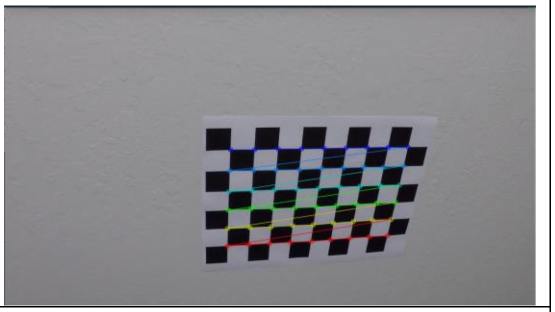
Preparation for pipeline

Camera Calibration

The camera was calibrated using the chessboard images in the folder of '/camera_cal', the result undistorted imaged are in the folder of '/camera_cal_output'

The following steps were performed for each calibration image:

- Convert to grayscale (cv2.cvtColor)
- Find chessboard corners (cv2.findChessboardCorners)
- Get distortion matrices (mtx, dist) with data from all calibration images (cv2.calibrateCamera)
- Undistort calibration images (cv2.undistort)

calibration image	Undistorted image
	
/camera_cal/calibration10.jpg	/camera_cal_output/draw_calibration10.jpg

Pickle Camera Calibration parameter

Pickle parameter (mtx, dist) to file ' camera_cal_data.pkl ' for later use of test_images and project_video.

Initialization for global variables

Many global variables for pipeline, such as line_width, ym_per_pix, xm_per_pix, mask_list, were initialized in this step and also unpickle camera calibration parameter (mtx, dist).

Pipeline for video

There are two main differences between pipeline for test images and video,

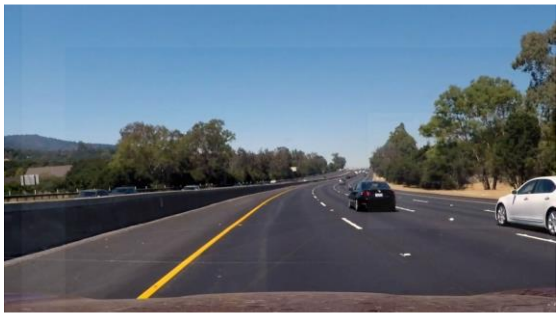
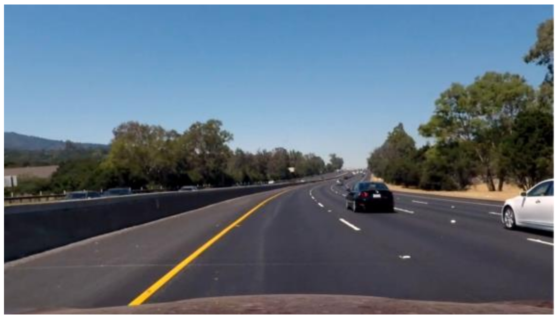
- Two methods (full scan version 'find_lane_pixels' and a quick version 'search_around_poly') to find lane for video.
- Line_class is being used to track, perform sanity check and smooth the result.

Basically, test images are treated as first frame of video, below are locations of original and output files.

	Original file in folder	Output file in folder
Test images	test_images	test_images_output
Project video	test_videos	test_videos_output

Undistort image

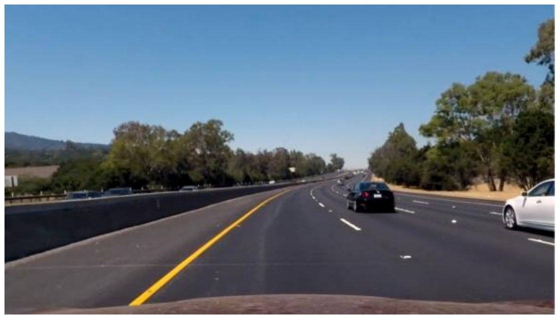
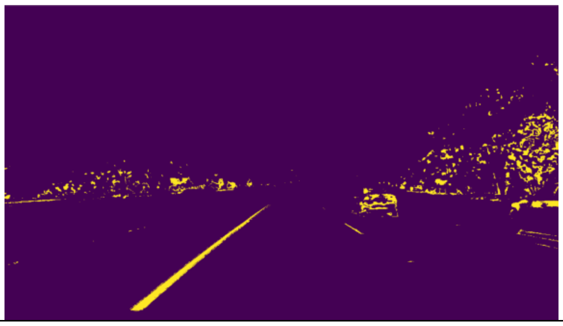
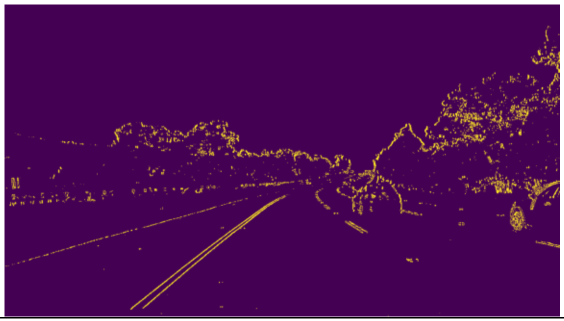
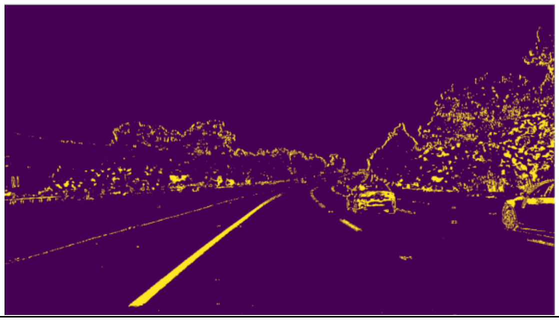
Undistort each frame from video (cv2.undistort) with camera calibration parameter (mtx, dist)

frame#1120 from project_video	Undistorted frame#1120
	

Thresholded binary image.

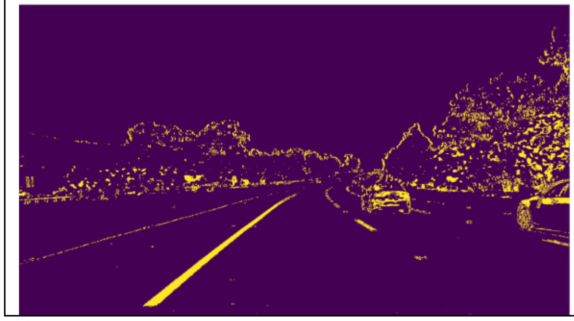
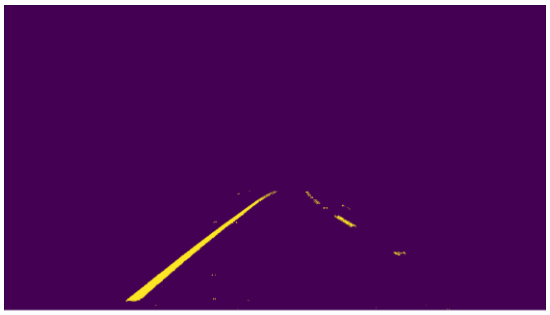
This is a very crucial step for lane fining and there are various threshold method and combination and the parameters are also playing an important roles. I was using a combination of following threshold image

- Absolute horizontal Sobel operator
- Magnitude in both horizontal and vertical directions
- Direction of gradient
- S channel of HLS space

Undistorted frame#1120	s channel
	
Absolute X+Magnitue+Direction	Combined
	

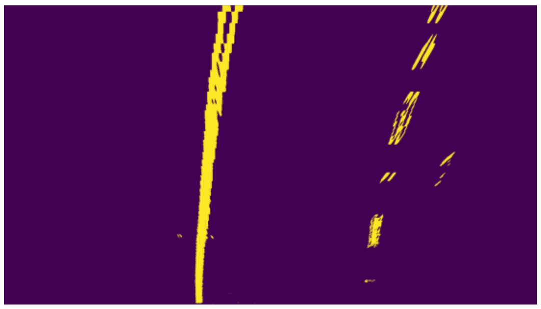
Apply Region of Interest

Apply a mask and discard all other lines outside the region.

Binary	Masked Binary
	

Perspective transform

To get a "bird's eye view" of the lane will enables us to fit a polynomial line to the lane lines. I set up source and destination locations and use cv2.getPerspectiveTransform and cv2.warpPerspective to warp images. At the same time, I saved the inverse matrix to global variable for later use to unwarp the image.

Masked Binary	Warped imaged
	

Polynomial fit

This is the most complicated procedure in this pipeline due to various lane condition, color of the road surface, shadow of trees, etc… There are two functions to do polynomial fit,

- find_lane_pixels

  This function is used for the 1 frame to find the initial position of the lane and also to reset lane position when the 2nd function is not adequate.

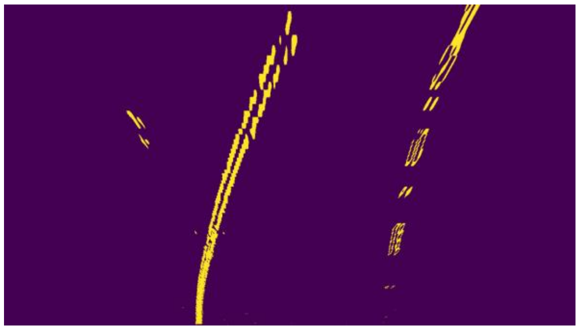
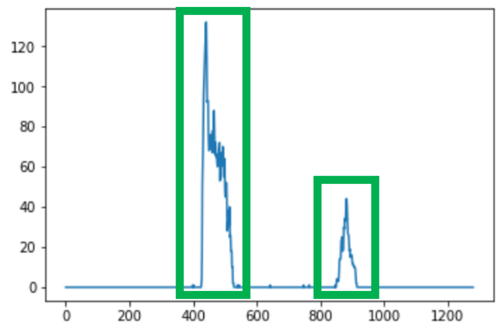
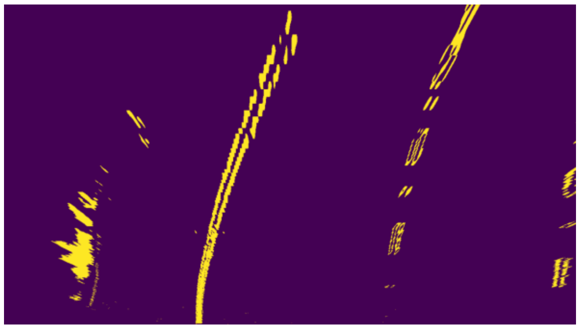
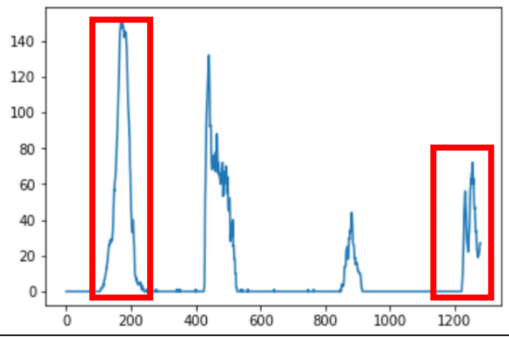
- search_around_poly

  From the 2nd frame of the video, the lane search could focus on the area close to the polynomial from the previous frame. The function will require much less calculation than the 1st function.

Below are some discussion on the details of these two functions.

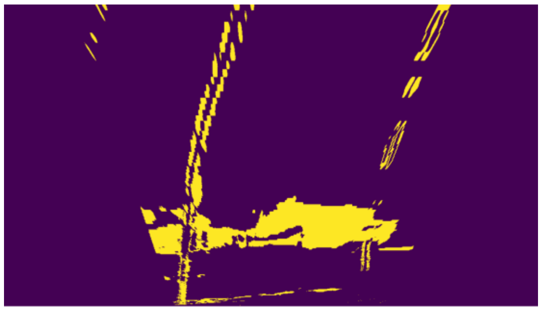
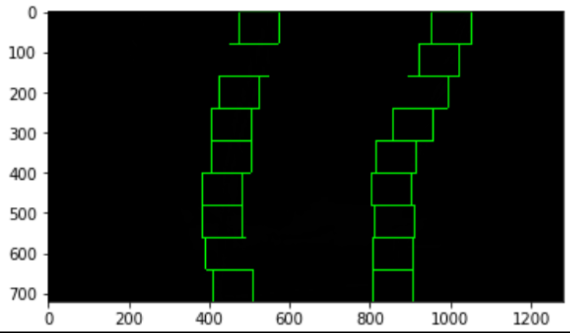
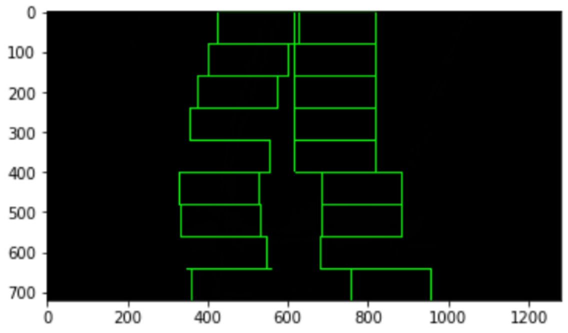
find_lane_pixels- Histogram function.

To use the region of interest improved the accuracy of finding the lane base location. Below is a comparison between to use or not to use the region of interest.

Frame#630 warped image after masking	Histogram find location correctly
	
Frame#630 warped image w/o masking	Histogram find wrong locations
	

Find_lane_pixels- Parameter of Margin

I found that "50" is a much better choice for the parameter "magin" than "100".

Frame#1042 Warped image after masking	Margin=50, good result
	
Frame# 1042 shallow of tree	Margin=100, wrong location
	

Look-Ahead Filter- search_around_poly

The efficiency of the code could be improved by using the function 'search_around_poly' instead of the full scan function of 'find_lane_pixels'. Actually, it does not only improve efficiency but also accuracy for some frame under certain road conditions, such as sharp turn, road surface with light color

Use of Line_class

I created two instances of line( ) class (left_line and right_line) and keep track the parameters (x location at bottom, x-location at top, radius_of_curvature, lane line detected or not, etc) of the previous 5 frames. For each new frame, the code will made comparison of the lane finding result with the previous frame, if there are substance change of X-location or radius of curvature, the code will smooth the result or discard the new result depending on the difference.

Curvature of the lane and vehicle position.

By converting pixel on images to physical distance in meters as below

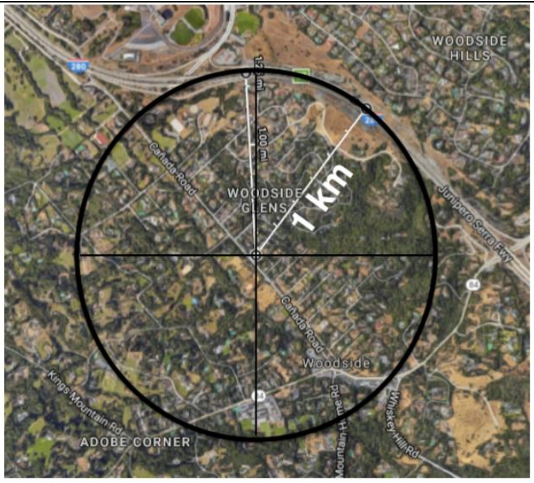
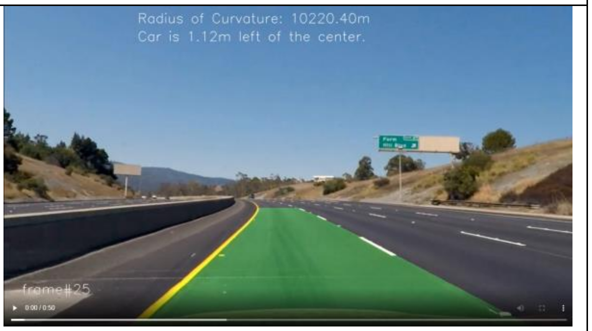
- Curvature calculation is based on ym_per_pix = 150/720

  Per the information provided by the class, the radius is 1km at a left turn curve road, I did several trials with difference parameter and choose 150m/720 pixel .

- Vehicle center calculation is based on xm_per_pix = 3.7/400

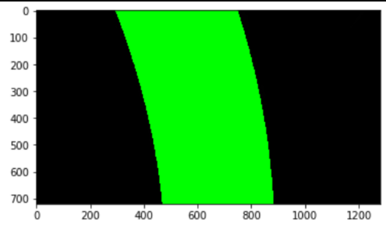
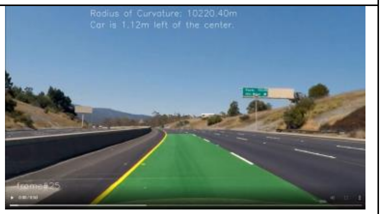
  The value of this parameter is based on the information provided by the class according the US high way regulation and I assumed the camera was mounted at the center of the vehicle which means the vehicle center was at x=640.

The curvature and vehicle center is calculated for each frame.

Actual radius data is 1km	Radius of curvature at Frame#25
 WOODSIDE HILLS 280 1.00 mi Canada Road WOODSIDE GLENS 1 km Junipero Serra Fwy 84 Woodside Kings Mountain Rd ADOBE CORNER Mountain Home Rd Whiskey Hill Rd	Radius of Curvature: 10220.40m Car is 1.12m left of the center. frame#25

Warp the detected lane boundaries back onto the original image.

After detecting the line positions and filling the area with green color between lanes in warped space, I used the inverse Matrix to combine unwarped image and original image together.

Frame#25	Filled line in warped space	Combined
		 Radius of Curvature: 10220.40m Car is 1.12m left of the center. frame#25

Output visual display

With all the steps above for each frame of the project video, the annotated video can be generated with lane area, radius curvature and vehicle position.

The pipeline to generate annotated test image is the same procedure for the 1st frame of the project video.

Discussion

Though the class has provided more function/codes to facilitate the project, there are still lots of work to do to detect the line lane correctly. Below are two key factors,

- Method to generate thresholded warped image

  There are various options/combination of threshold methods, absolute x, absolute y, magnitude, gradient direction, HSL color channel. The binary warped is the crucial input for the lane find function.

- Algorithm to find the lane from the warped image.

  Due to the road conditions, such as the broken lane line, shadow of tree, not all lane line could be thresholded, this brings difficulties to find the lane correctly from warped image. So I have to use the line() class to track line location and make comparison between previous frames and current frame.

There are potential opportunities to make improvement, for examples, we can use dynamic parameter for threshold and lane finding while this will require more iteration.

In short, I did enjoy the whole process of this project, appreciate the effort from Udacity team, my mentor and reviewer. Looking forward to the challenge in the coming projects.