Project: Advanced_Lane_Detection
Name: Dongdong Wu

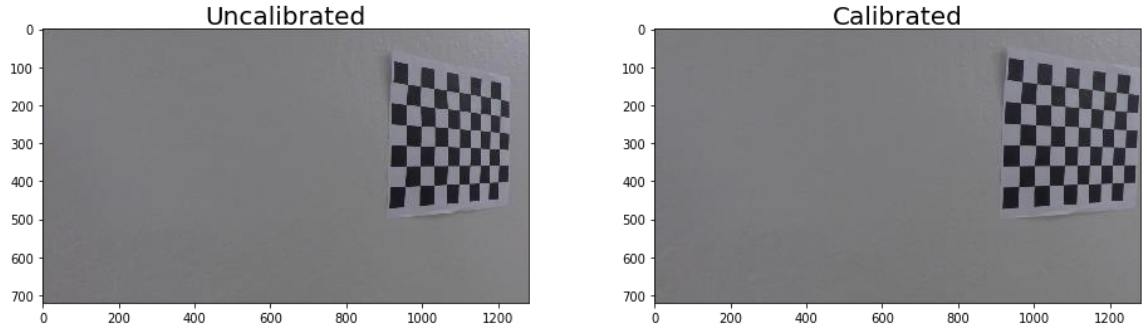
Overview:

This project is part of the Udacity self-driving car program, and much of the code is leveraged from the lecture notes. The pipeline of this project includes:

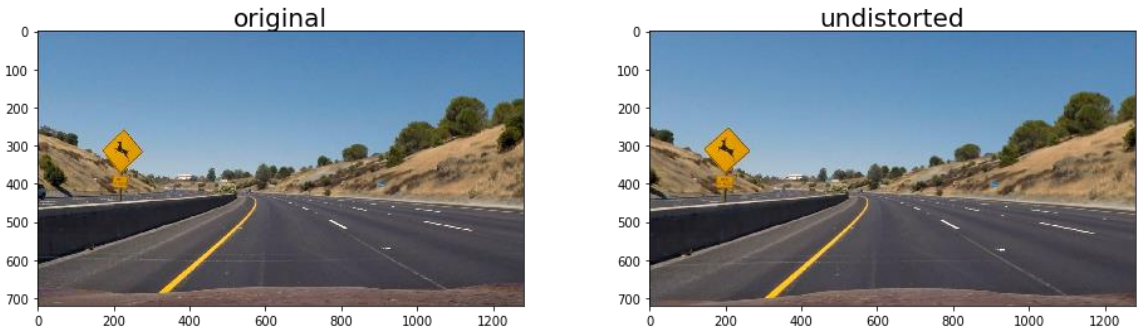
1. Measure calibration matrix and distortion coefficients based on given chessboard images and then apply measured data to rectify raw images.
2. Use gradients threshold on gray image and color threshold on S_channel, to create a threshold binary image.
3. Perspective transform to rectify binary image to birds-eye view image.
4. Detect left and right lane, then do fit to find the lane curve. Based on fitted curve, measure curvature of the lane and car drift to road center
5. paint fitting curve and measure curvature / drift value on original image
6. do lane detection pipeline for each frame of video and generate new video with fitting curve and curvature information

Camera calibration:

Based on randomly selected chessboard image provided, find 9*6 corners by calling Opencv findChessboardCorners() function. The image before and after calibrated likes:



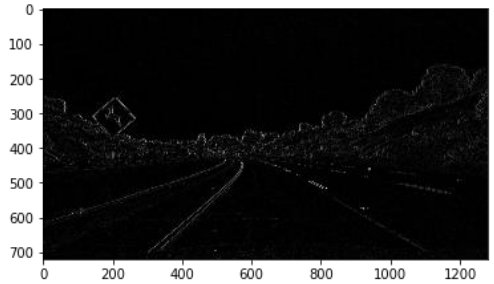
then applied measured calibration matrix and distortion coefficients to raw image provided.



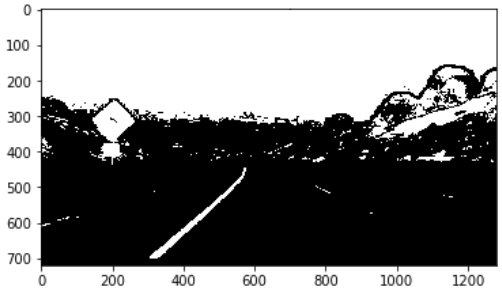
Binary Image by threshold

Based on measure undistorted raw image as input, apply:

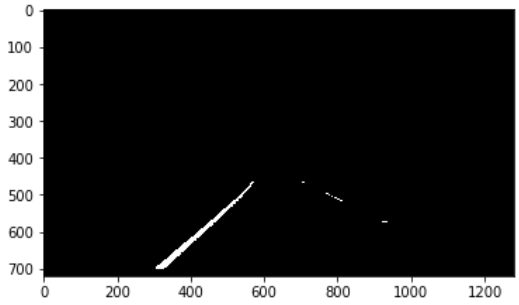
1. Convert image to gray image and Apply Absolute horizontal Sobel operator to get x-gradient image



2. Convert image to HLS image, and by threshold S-channel, create color based image

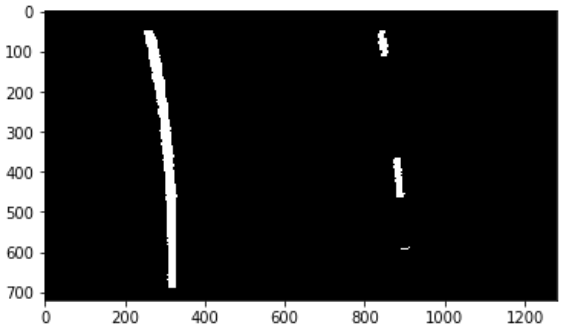


3. limit detect area as mask to avoid interference of other edges, and combine mask + gradient image + color image to get binary lane



Perspective transform

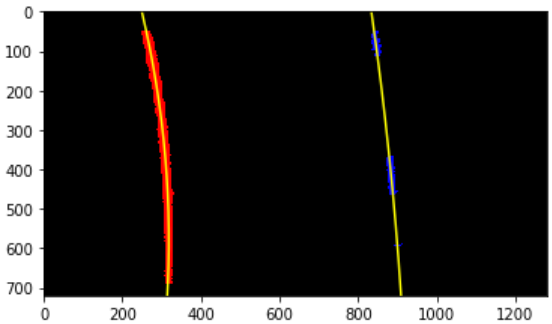
Given binary image of lanes, do perspective transform to get bird-eye view image.



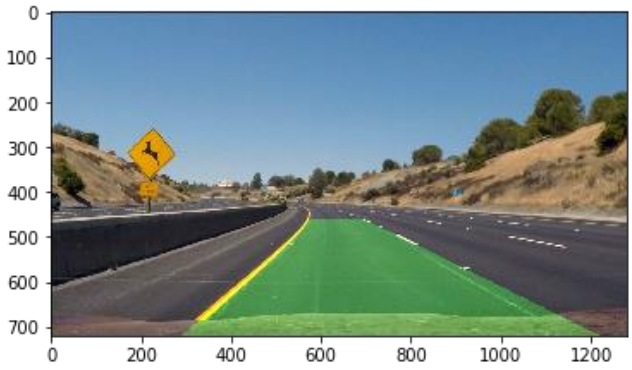
Fit lane and measure curvature

Based on transformed binary image, apply:

1. Calculate a histogram of the bottom half of the image to rough start location of lanes
2. Partition the image into 9 horizontal slices
3. Starting from the bottom slice, enclose a 200 pixel wide window around the left peak and right peak of the histogram
4. Move up horizontal window slices to find pixels that are likely to be part of the left and right lanes
5. Based on estimated lane pixels, fit a 2nd order polynomial



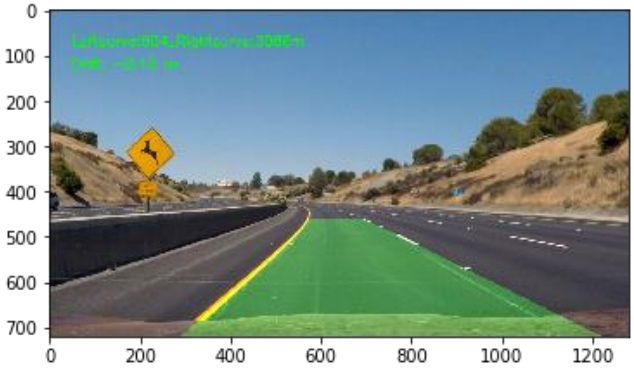
6. Paint fitting lanes to image



Measure curvature and vehicle drift

Given calculated fitting lane, measure lanes curvature for left and right lanes, and by comparing detected lane positions and fitting values, to measure vehicle drift offset to road center

Warp curvature information to image



Apply pipeline for each frame of video

Based on input video, generate output video “project_video_output.mp4” video by go through pipeline above for each frame