Project 1 - Lane Detection - Report

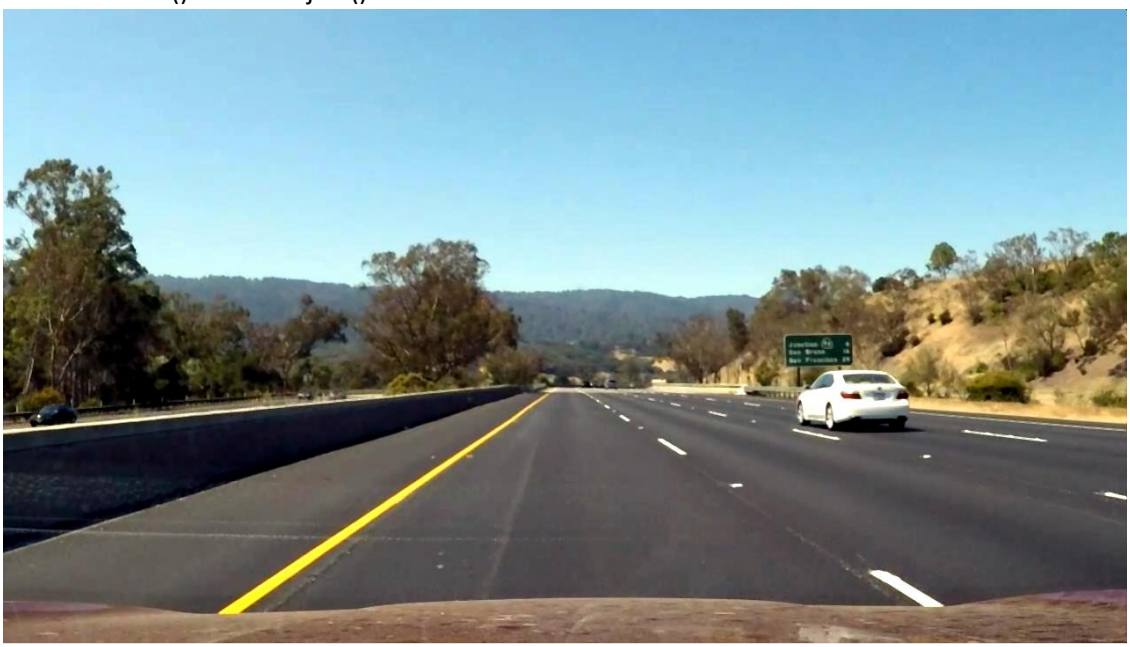
1) Important ideas.

a) **Denoise** and **Enhance** the original pictures in separate RGB channel and regroup them. medfilt2() and imadjust() are the functions I choose to realize that.

Original:

After medfilt2():

After medfilt2() and imadjust():



b) Find color in **HSV** images.
rgb2hsv() to transfer RGB images to HSV images. Tune threshold for yellow and white color. The website, http://colorizer.org/, helps a lot while it offer adjustment of h, s, v channel to see the color.

Note that imadjust() is only used for finding yellow color in the challenge video.
Yellow:

White:

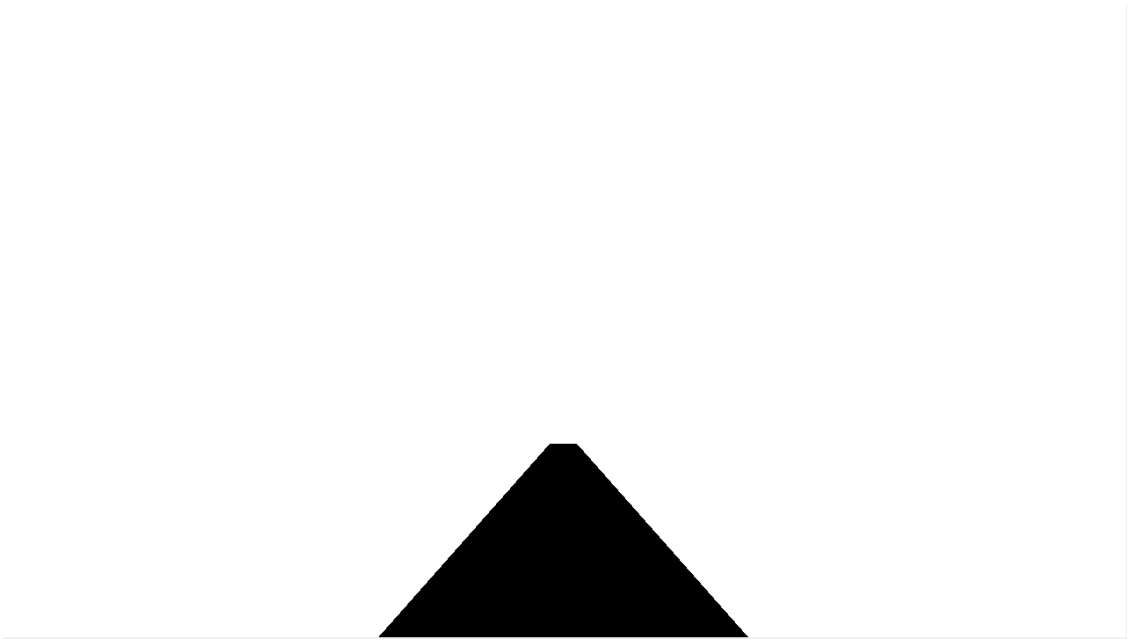
c) Cancel the noise from white and yellow car on road.
bwpropfilt().

d) **Edge detection**.
Using Canny's method.

e) Create **field of interests**.
poly2mask().

Maks1:

Mask2:



f) **Hough Transform**.
hough(), houghpeaks(), houghlines().

g) **Group** the lines to left and right.
Using the theta values in lines struct. Eliminate lines with slope larger than 85 degree.

h) **Linear regression**.

Eliminate the influence by the outliers and refine the left and right lines. In my code, I only choose the top 12 peaks in the Hough transform. The errors are huge after group them into 2 parts. The outliers must be cancelled out. polyfit() is used.

i) Extend the left and right lines to a trapezoid.

j) Smooth the lines by applying a **low-pass filter**.
Sometimes the shadow influent the color a lot and makes the color detection algorithm losing efficacy. I set the threshold to be 96% to cancel the jump points for the left and right lines.

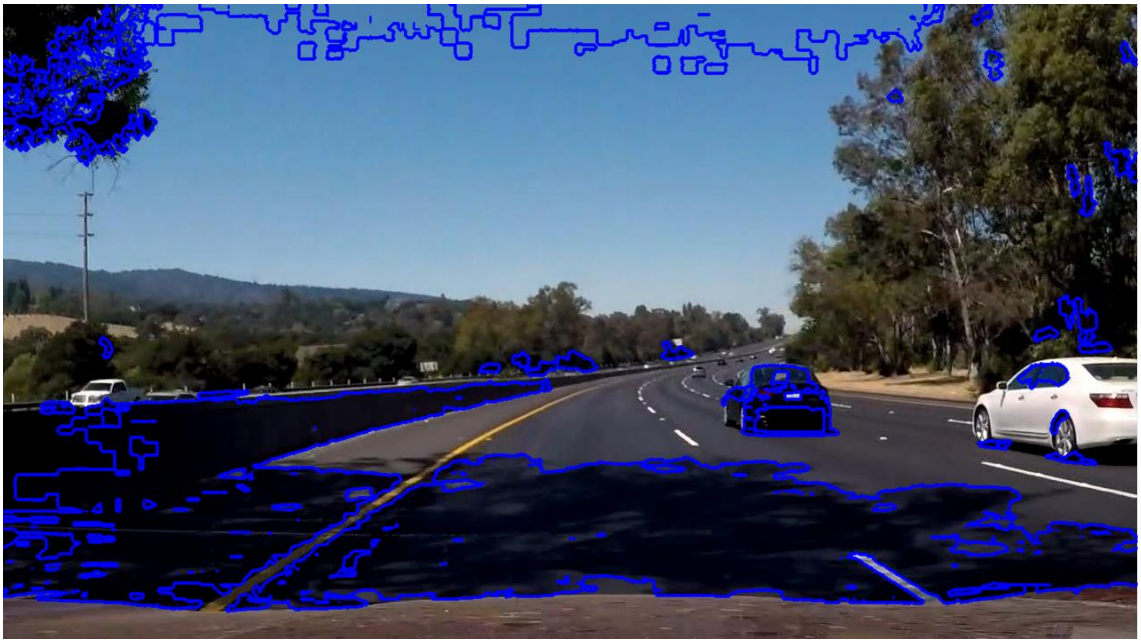
k) Fill the detected area with transparent color.

l) Predict the turns.
I don't find the vanishing point but use the idea of that. Since the vanishing point is determined by the intersection of the left and right lines. And each y value after polyfit is accurate enough so I find the 2 middle points between upper-left, upper-right points and between bottom-left and bottom-right points, respectively. Plotting an arrow by connecting the 2 middle points to show the direction, thus then predicts the turns.

Note that I don't plan to printf a string all the time says "go straight", "turn left" or "turn right". Since driving is not a binary concept – either turn the steering wheel or keep it still. I think an arrow to predict is better because driver or the AI can do subtle changes rather than wait for a while and make a big adjustment.

2) Abandoned ideas.

a) **Shadow detection and removal**.
Success in shadow detection for the regular video while the shadow from the tree covers the yellow line. However, the yellow line cannot be detected as a shadow area even set the threshold to be 0.2, which makes the detected area useless. I didn't try this for the challenge video. But the results will be assumed negative in my opinion.



b) Using middle points of left (or right) lines' points and mean theta value of left (or right) lines to replace polyfit.
I don't know about the linear regression in the first place and tried an algorithm by myself to define the final left and right line. Because the limitation of sample volume, my first results are awful due to the outliers.

3) Results.

a) My code works for both regular and challenge videos (The whole video, not just the simple period of them). You can use my code to process them again to see the results. 2 output videos are for 2 input videos respectively.

b) There's 2 video output from the regular video because I changed the algorithm to define the direction line. It looks so wrong to use the bottom middle point of the trapezoid area so I add a new video using the bottom middle point of the whole image. That looks much better intuitively.

My guess is that the driver drives a little bit to the right in the lane.