**# CarND-Project2-Advanced Lane Finding**

In this project, I target to consolidate a code file "model-akp.py" that output result images of each stage (such as camera calibration, Distortion correction, ...) and finally write the annotated video of the input video

I sincerely thank Jeremy-Shannon and Mohamedameen, for uploading their great work in Github. I downloaded their work on to my local machine and studied them very thoroughly. Understanding that, greatly helped me in completing my 2nd project in the Udacity nanodegree program

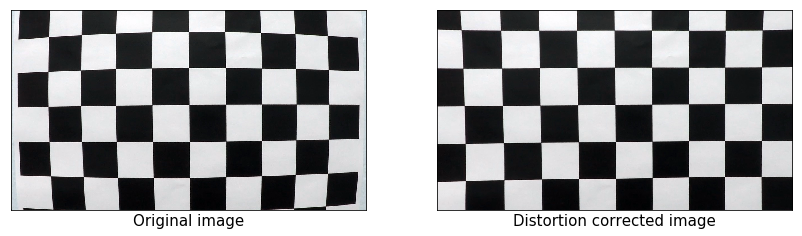
## Pipeline architecture:

The total code has 3 major blocks

1st Block: It contain input (chessboard pictures and test pictures) for calibration and testing. It also includes all the below functions that are finally used in the 3rd block of codes to run the pipeline for video generation

These functions are as follows

* Function “display( )” – this is to define the display format where 2 images are shown side by side, so that both original and modified pictures can be shown together
* Function “calibrate\_camera( )” – 20 pictures of Chessboard is different orientation, residing in folder “camera\_cal” folder, are used to derive camera matrix “mtx”, distortion coefficient “dist”, rotation vector “rvecs” and transverse vector “tvecs”.
* Function “undistort( )” – This function uses as input the above found factors and also the images of chessboard pictures from folder “camera\_cal” and test pictures in folder “test\_images”. The output of this part is side-by-side display of original distorted and resulting undistorted chessboard and test pictures

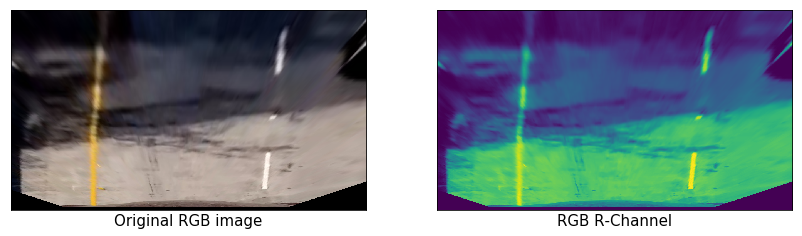




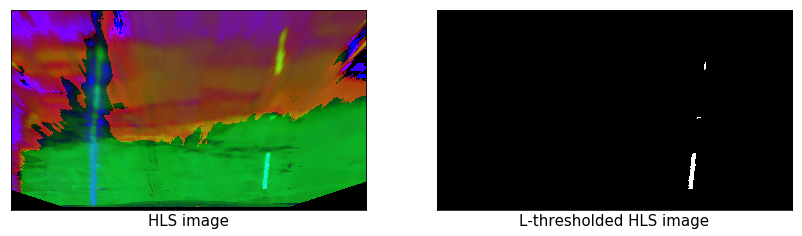
* Function “perspective transform” – This function mainly converts front view into top-down bird’s eye view. Using standard openCV function, 3 items are generated here. They are perspective transform matrix “M”, inverse perspective transform matrix “Minv” and “Warped”. Using these items, some test pictures are displayed side-by-side both their original and warped form as follows:



Further the same warped picture is viewed in different color channels (like R, G & B in RGB; H, S and V in HSV; L, A & B in LAB; H, L & S in HLS), to see which one give the most prominent output. Example of R & RGB combination below



* Function “hls\_l\_thresh( )” – takes the original “HLS warped picture” and show the pixels only above the minimum threshold in the “L channel version of the warped picture”. The comparison outcome is below



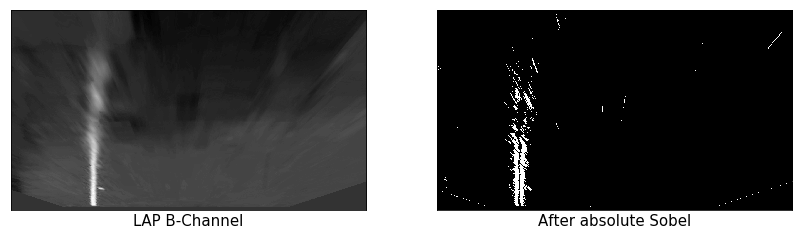
* Function “lab\_b\_thresh” – takes the original “LAB warped picture” and show the pixels only above the minimum threshold in the “B channel version of the warped picture”. The comparison outcome is as below

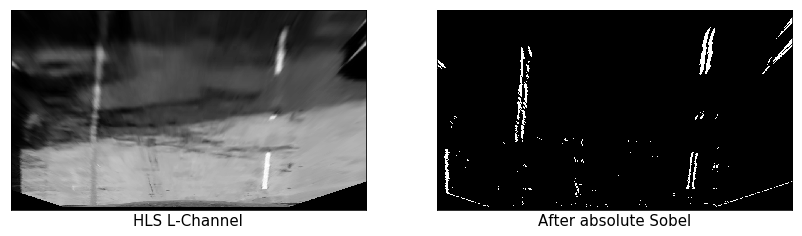


* Function “threshold\_color\_space” – finally combines the effect of thresholding from “L-channel of HLS color space” and “B-channel of LAB color space”. The comparison outcome is as follows

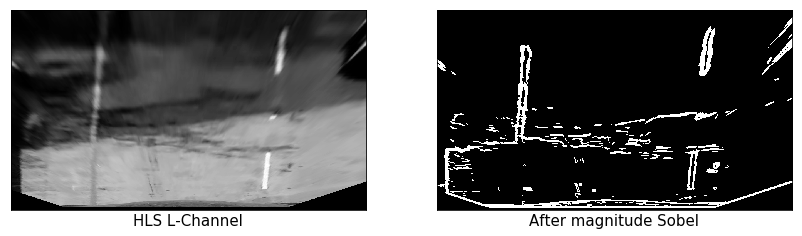


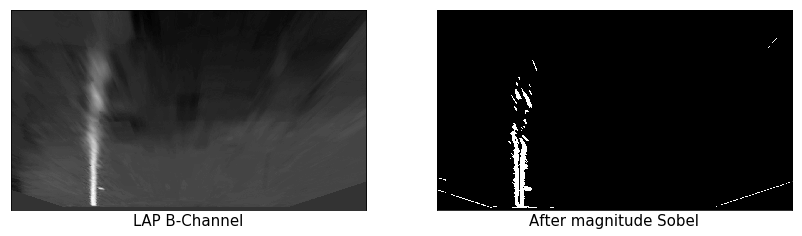
* Function “abs\_sobel( )”absolute sobel – Compares the outcome from “HLS\_L channel version of wraped picture” with “absolute sobel treated version of the same picture”. Also compares the same between “LAB-B channel and its “absolute sobel treated of the same”. Outcome is an “binary” picture based on the differentiation in X direction. But there is a problem. It detects many other things in addition to the two road marking



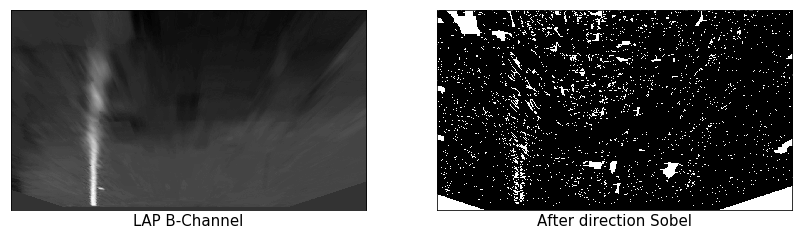


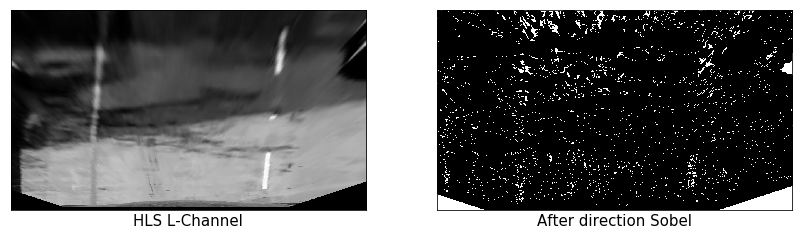
* Function “mag\_sobel” magnitude sobel – Does the same work as above, but this time the result is based on differentiation both in X and Y combining as Sqrt(Xsquare + Ysquare). The 2 comparisons made here are – 1) between “LAB-B channel picture” and “its sobel treated version”. 2) between “HLS-L channel picture” and its “sobel treated version”. Pictures are as follow



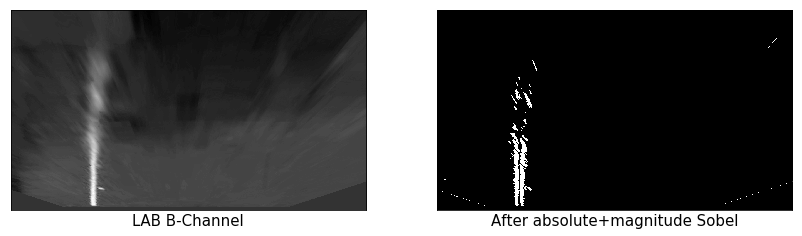


* Function “dir\_sobel” direction sobel – This is also exactly like above, but the result is based on differentiation both in X and Y direction combining as arctan (Y/X). The result is most worst here



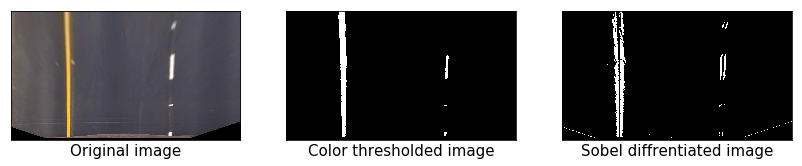


* Function “combined\_sobel” – Here absolute and magnitude sobel differentiators are combined and the result are as follows



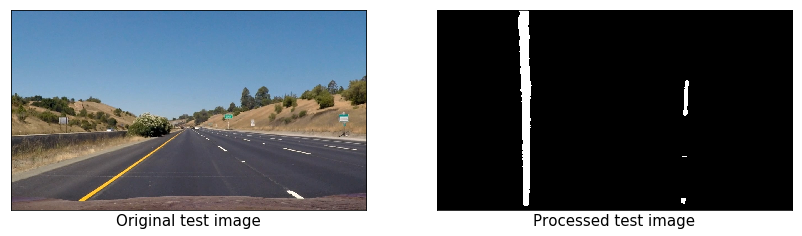


* Functioin “combined\_sobel\_color” – here 3 things are combined – absolute sobel, magnitude sobel and color.



2nd Block: This is for defining the image processing pipeline. This includes various functions like

* Function “image\_process( )” – Here 3 functions defined earlier are put together – 1) undistortion, 2) perspective transform and 3) color space thresholding. When applied on test\_images, following type of result is visible



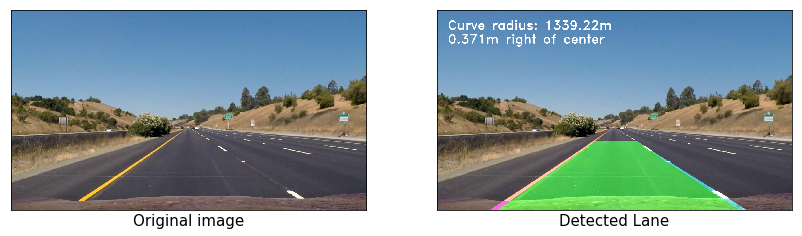
* Function “sliding\_window( )” – For this function, histogram of the image is the base and sides ofthe histogram provides the 2 sides of the window covering the edge in the picture. Thereafter depending upon the moving direction of the edge, incremental windows are slided one over another to trace the edge. Below is a sample applied on one test\_image



* Function “polyfit\_prev\_fit( )” – This is very similar to the previous function, except the base here is a polynomial fitting the edge in one frame. The same polynomial is slightly adjusted as per the minor change in the edge in the next frame and so on. When applied on the test\_images, one sample as follows:-



* Function “curve\_pos( )” – This function provides 1) lane curvature, 2) position of vehicle on the lane. 3.048m in y direction represents 100 pixel and lane line is 10ft (3.048m). Similarly, 3.7 m in x direction represents 378 pixels and lane width is 12 ft (3.7m). When applied on the test\_images, a sample is as follows



* Function “draw\_line( )” – This function draws the lane over the detected lines in the image. The 2 slanting lines in the sample above, are the outcome of this function
* Function “write\_data( )” – This function writes text data relating to curve radius and position of vehicle, also as shown in the sample above

3rd and the last Block: This is for processing video. it includes main parts as follows

* Function “class\_Line( )” - for initialization of key information for the new frame, from the history of the earlier frames. So that the incremental information are passed on to the new frame
* Function “frame\_processor( )” - to link together functions defined earlier. Key functions used here are 1) undistort( ), 2) image\_process( ), 3) sliding\_window( ), 4) polyfit\_prev\_fit( ), 5) draw\_lane( ), 6) curve\_pos( ), 7) write\_data ( ) and finally line ( )
* Finally to write annotated video from input video – here key functions from moviepy library used are 1) VideoFileClip ( ), 2) fl\_image( ) and 3) write\_videofile( )

### Result:

* When the codes are run, all output images come first. Each image to be cancelled for seeing the next image
* -Once all images are shown and cancelled, annotated video generation (of the input video) starts
* For different input video files, the same is provided into the last block of codes above and output annotated video file is saved in the folder
* total 3 videos are processed. they are present in the folder

#### Limitations and improvements

* Annotation of input images worked without any problem
* 2 input video files (project\_video.mp4 and challenge\_video.mp4) worked well and resulted output videos are in the folder
* But the 3rd input video file (harder\_challenge\_video.mp4) did not work well and the output video shows discontinuation of lane lines at several places

## Conclusion

* This was very interesting project. But very difficult to understand all parts for integration, due to so lengthy code lines involved.
* At many places, syntax remains yet to be understood