Finding Lane Lines on the Road

Required Files

CRITERIA	MEETS SPECIFICATIONS
Have all project files been included with the submission?	The project submission includes all required files: • Ipython notebook with code (P1.ipynb in the project folder) • A writeup report (either pdf or markdown). The write is in the file Writeup_Finding_Lane_Lines_on_the_Road.pdf in the zip file. • In addition an html and pdf copies of the notebook are attached as P1.html, P1.pdf in the main project folder

Lane Finding Pipeline

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
Does the pipeline for line identification take road images from a video as input and return an annotated video stream as output?	The output video is an annotated version of the input video.	The pipeline takes images from input video and return the annotated video stream as output. The video output is saved in test_videos_output/solidWhiteRight.mp4, test_videos_output/solidYellowLeft.mp4 files.
Has a pipeline been implemented that uses the helper functions and / or other code to roughly identify the left and right lane lines with either line segments or solid lines? (example solution included in the repository output: raw-lines-example.mp4)	In a rough sense, the left and right lane lines are accurately annotated throughout almost all of the video. Annotations can be segmented or solid lines	The pipe line uses the helper functions given. What process_image function is nothing but the sequence of steps that are used to test the test_images. Also all the forms of the images are saved under test_images_output folder. (carnd-term1-tf11) gvenkat@PT7910:~/SDCND/CarND-LaneLines-P1/test_images_output$ ls -1 . blur_gray_solidWhiteCurve.jpg blur_gray_solidWhiteRight.jpg blur_gray_solidYellowCurve2.jpg blur_gray_solidYellowCurve.jpg

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
		blur_gray_solidYellowLeft.jpg blur_gray_whiteCarLaneSwitch.jpg canny_blur_gray_solidWhiteCurve.jpg canny_blur_gray_solidWhiteRight.jpg canny_blur_gray_solidYellowCurve2.jpg canny_blur_gray_solidYellowCurve.jpg canny_blur_gray_solidYellowLeft.jpg canny_blur_gray_whiteCarLaneSwitch.jpg gray_solidWhiteCurve.jpg gray_solidWhiteRight.jpg gray_solidYellowCurve2.jpg gray_solidYellowCurve.jpg gray_solidYellowLeft.jpg gray_whiteCarLaneSwitch.jpg hough_masked_canny_blur_gray_solidWhiteCurve.jpg hough_masked_canny_blur_gray_solidWhiteRight.jpg hough_masked_canny_blur_gray_solidYellowCurve2.jpg hough_masked_canny_blur_gray_solidYellowCurve.jpg hough_masked_canny_blur_gray_solidYellowLeft.jpg hough_masked_canny_blur_gray_whiteCarLaneSwitch.jpg masked_canny_blur_gray_solidWhiteCurve.jpg masked_canny_blur_gray_solidWhiteRight.jpg masked_canny_blur_gray_solidYellowCurve2.jpg

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
		masked_canny_blur_gray_solidYellowCurve.jpg masked_canny_blur_gray_solidYellowLeft.jpg masked_canny_blur_gray_whiteCarLaneSwitch.jpg
Have detected line segments been filtered / averaged / extrapolated to map out the full extent of the left and right lane boundaries? (example solution included in the repository: P1_example.mp4)	Visually, the left and right lane lines are accurately annotated by solid lines throughout most of the video.	This has been done by extending the draw_lines function. Main idea behind the implementation is to use the slope and intercept concepts and calculate resultant slope and points to extend the line sgements from the bottom of the image to the height of 60% of the images. Roughly around 60% of the height the lines stay separate and tend to intersect if we cross the 60% of height and also aroung the corners.

Reflection

CRITERIA	MEETS SPECIFICATIONS
Has a thoughtful reflection on the project been provided in the notebook?	Reflection describes the current pipeline, identifies its potential shortcomings and suggests possible improvements. There is no minimum length. Writing in English is preferred but you may use any language. For draw_lines function, I have borrowed some code from internet as the implementation (in draw_lines2 function) I had was not clean.