Finding Road Lane Lines

Done by: Wael Farag

1. The implemented pipeline:

The following steps describes the implemented pipeline in order of execution:

1. Reading test images: this is done using the "Read_Color_Image()" function. It simply reads all the images in the designated directory "test_images" one at a time in alphabetical order. An example of these images is shown in Figure 1.
2. Converting the color test image to gray: this is done in function "Convert_Color2Gray" using the OpenCV function "ctvColor". Figure 2 shows the image in Figure 1 in gray scale.
3. Filtering the noise: this is done in the "Filter_Blur()" function using the OpenCV function "GaussianBlur" which executes the Gaussian filter algorithm. A kernel size of 5 has been selected. Figure 3 shows the image in Figure 2 after filtering.
4. Edges Detection and Extraction: this is done in function "Edge_Detection" using the OpenCV function "Canny" which executes the well-known Canny algorithm. Figure 4 shows the image in Figure 3 with edges detected. The following table (Table 1) lists the several parameters for the algorithm operation that have been selected after careful tuning and many trial and errors sessions.

Table 1: Canny algorithm parameters.

Parameter	Value
Low threshold	50
High threshold	150

5. The Identification of the region of interest: This is implemented using the " Extract_Region_of_Interest()" function which masks a Trapezoidal area in an image with edges detected (e.g. Figure 4) to produce an image which has only lines corresponding to road lane lines as shown in Figure 5. The vertices of the identified region of interest is shown below in Table 2.

Table 2: Area of interest vertices.

Vertex	X	Y
Lower Left	0	539
Lower Right	959	539
Upper Left	450	330
Upper Right	490	330

6. Detecting straight Lines: straight lines are identified using the "Lines_Detection()" function which implements the Hough algorithm in polar coordinates using the OpenCV HoughLinesP() function.

The resultant Hough line segments are depicted in Figure 6. However, the function not only calculates Hough line segments but also draws extrapolated lines on the image provided as shown in Figures 7 & 8. The parameters that has been found after careful and recursive tuning and used in the Hough algorithm are listed in Table 3.

Table 2: Hough algorithm parameters.

Parameters	Value
rho	1
theta	PI/180
minimum_votes_threshold	15
min_line_length	7
max_line_gap	3

2. The implemented Line-Drawing Function:

The line drawing function "Draw_Lines()" is implemented to connect the line segments for each lane-line (left or right) to produce one solid line that tracks the actual lane-line in the images. The line segments are produced by Hough transform as shown by Figure 7 and the "Draw_Lines()" function is connecting them to be like the ones in Figure 8. The function is achieving that through the following steps:

1. Classification (Left or Right): Classifying all the Hough line-segments as to belong to the left or right lines. This is done based on the slope of the line segment. If the slope is positive and lies between 0.4 => 1.0 then this segment belongs to the left line class, and if it is negative and -0.4 => -1.0 then it belongs to the right line class.
2. For all the classified Left & Right line-segments, the lengths and intercepts (intersections with the x-axis) are calculated and stored along with their slopes.
3. A line fitting technique is used for each class (Left and Right) taking the slope of each line segment as an indication of a good or a bad (noise) line segment.
4. Now the Left and Right lines can be drawn, however to reduce the jitter the information of the previous frame is incorporated to produce an N^{th} order filter (we tested it till 30^{th} order).
5. The FIR equation considered in this implementation is given as $Y_k = a_o*X_k + a_1*X_{k-1} + a_2*X_{k-2} + a_3*X_{k-3} \dots a_n*X_{k-n}$.
6. The resultant lines slopes and intercepts and using the information of "the region of interest" calculated in step 5 of the pipeline, are used to draw the Left line in blue and the right line in red.

3. Shortcoming of the implemented approaches

The following list summaries the identified shortcomings:

1. Even though we used high order FIR Filter to reduce the jitter of the left and right lane-lines, however the results are not good enough especially at light color roads and shades as shown in the “Challenge.mp4” video @ the 4th Second.
2. It only detects the straight lane lines. It may be an advanced topic to handle curved lanes (or the curvature of lanes). We'll need to use perspective transformation and also poly fitting lane lines rather than fitting to straight lines.
3. How the current approach can handle up-hill or down-hill roads, is still a big question? How to determine the region of interest in this case. Needs a study.
4. What about if you have a close car in front of you, how this approach can handle this case.

4. Suggested Improvements

The following list summaries the suggested improvements:

1. Using line-segment length as a differentiator between strong line segments and weak line segments while applying he line –fitting technique (going to work on it in the coming week).
2. More investigation need to be done in the design of the FIR filter: trying higher orders, trying different low pass polynomials like Butterworth, Chebychev, Elliptical … etc.
3. Identification for the type of the lane (dashed or solid) is important in driving regulations, so it needs to be added.
4. Line detection has been widely studied in the literature and Google Scholar search could lead to some new state-of-the-art techniques used by the research community. I would like to revisit this problem after I read and learn more about it.

Figure 1 “solidWhiteCurve” original image.

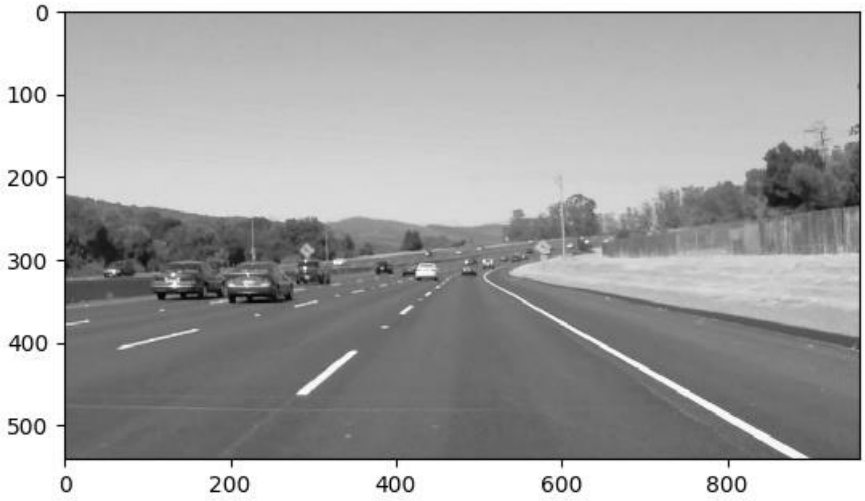

Figure 2 “solidWhiteCurve” in gray scale.

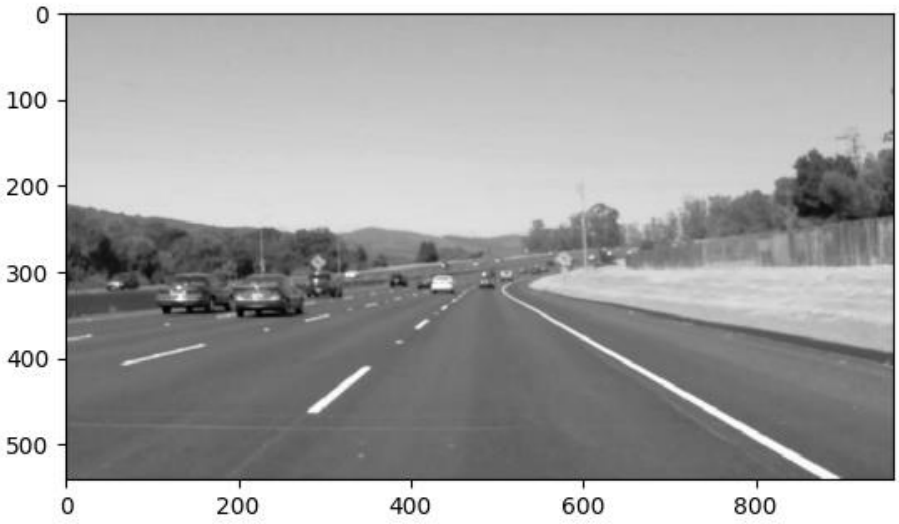

Figure 3 “solidWhiteCurve” after the Gaussian Blur Filter is applied.

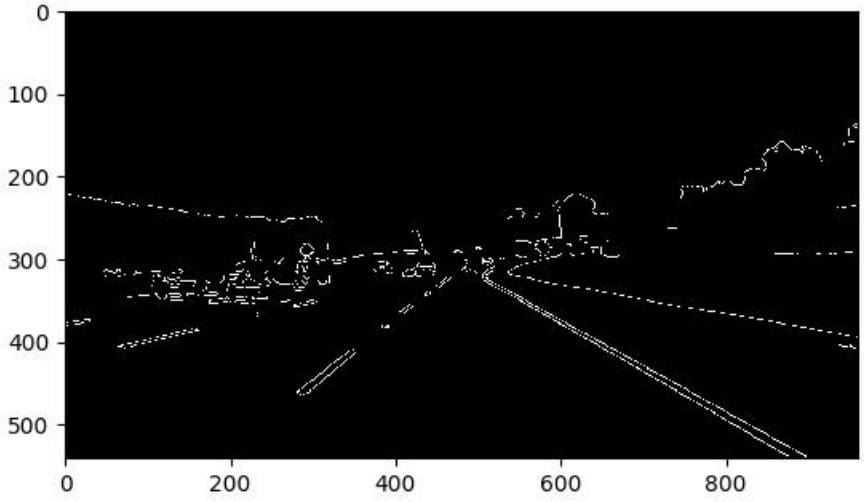

Figure 4 “solidWhiteCurve” after the Cunny algorithm is applied.

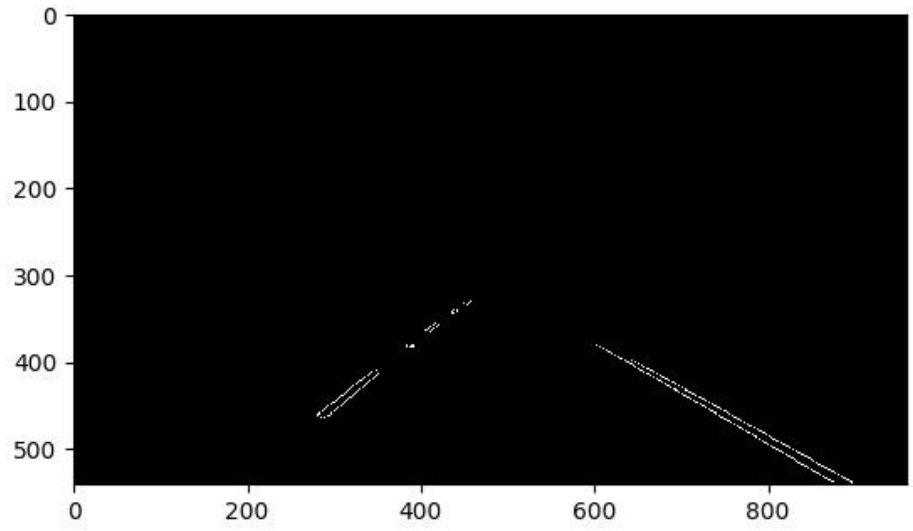

Figure 5 “solidWhiteCurve” after extracting the area of interest.

Figure 6 “solidWhiteCurve” with Hough line segments.

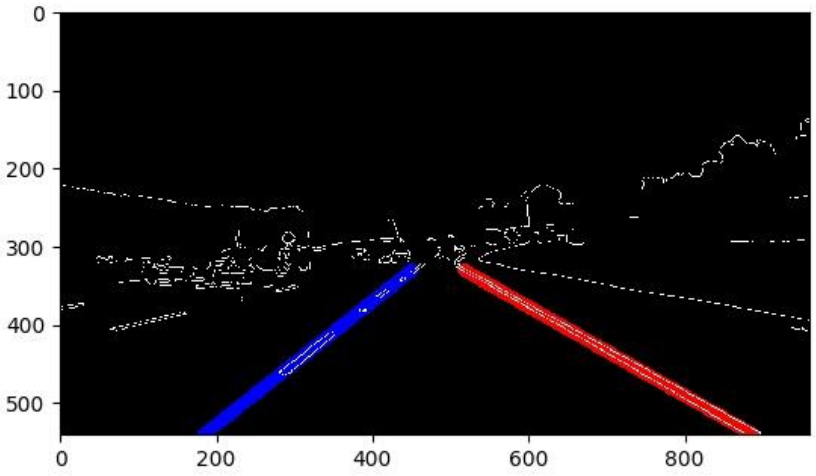

Figure 7 “solidWhiteCurve” after drawing lane lines (blue and red) by extrapolating the Hough lines segments.

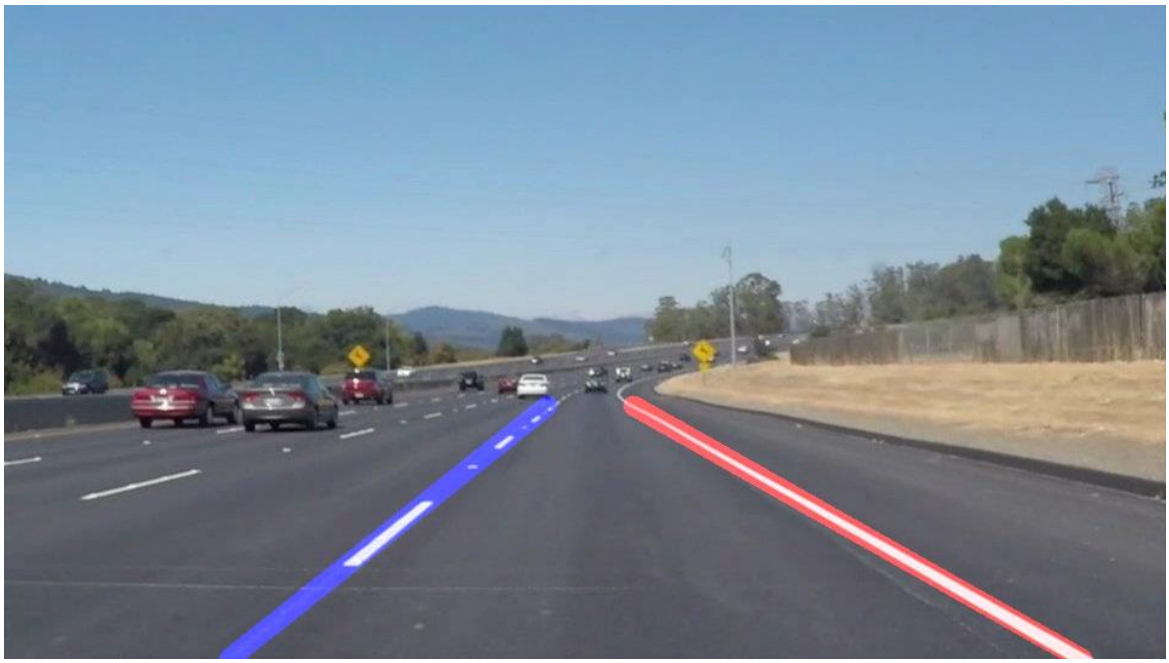

Figure 8 “solidWhiteCurve” after drawing lane lines (blue and red) by extrapolating the Hough lines segments.