In this project, I developed and used several functions to process the test images and then the video:

Grayscale()

I used this function to return an image with only one color channel.

gaussian_blur(gray, 3):

I used this function to apply a Gaussian noise kernel with kernel size 3.

canny(gray,50,150):

I used this function with above parameters to apply Canny transform.

region_of_interest(edges, vertices)

I used this function to apply an image mask and keep only the region of the image defined by the polygon formed from vertices (The rest of the image is set to black).

hough_lines(target, 1, np.pi/180, 35, 5, 2)

I then used this function to apply Hough transform with above parameters to the image coming out of canny transform. Also inside this function, another function is called to draw Hough lines in image. Below is this function:

draw_lines():

This function is the most important function which has most codes written. Here the output of hough transform is fed into this function (the start and end coordinates x and y of each line). Then for each line, the slope is calculated and depending on positive or negative value of slope, it will be classified in left or right line. Then the average slopes for left and right lines are calculated. Also, the average coordinates x and y for midpoints of right and left lines are calculated.

Now, the goal is to draw a line which best represents the left lines combined (and then another line which best represents the right lines combined). Since y is almost constant and known at both ends, then x from each end can be calculated from the formula below:

(‘Y_left_end ’ – ‘y_left_average’) = m_left_averag * (‘x_left_end’- ‘x_left_average’)

As we see above, all parameters are known except for ‘x_left_end’ which can be calculated from above equation.

Similarly, another equation can be written to calculated ‘x_left_begin’:

(‘Y_left_average ’ – ‘y_left_begin’) = m_left_averag * (‘x_left_average’- ‘x_left_begin’)

Similarly, for right line:

(‘Y_right_end ’ – ‘y_right_average’) = m_right_averag * (‘x_right_end’- ‘x_right_average’)

(‘Y_right_average ’ – ‘y_right_begin’) = m_right_averag * (‘x_right_average’- ‘x_right_begin’)

After finding coordinates for x and y for start and end points of each right and left lines, we draw lines accordingly into the images. For end point it is considered y=320. for start point it is considered y = 540 (ys for start and end points of left and right lines are almost always constant, and it is x that is changing)

weighted_img(img, initial_img, α=0.8, β=1., γ=0.):

The output of Hough transform (image with lines drawn into it), is fed into this function. (Blank image with lines drawn on it). Initial_img is the image before processing. Initial_img and img are of the same shape. It computes the result image as follows:

initial_img * α + img * β + γ

I then used codes to read images from test_images folder and processed them by above functions and save results as output images in same folder. For images, I didn’t use my codes for plotting a continuous line inside images just to test the functions and codes (although I did it for myself to test the quality of continuous lines on images too, and then applied them to the video) Some images are shown below as examples:

Finally, I developed a function named process_image to process videos by using several functions named above on each image of the video screen. As mentioned earlier, I used continuous lines for left and right this time on videos (unlike the images). I then stored the result as ‘solidWhilteright.mp4’ in ‘test_videos_output’ folder. The result looks good and encouraging☺