Pablo Sauras Perez Reflection.

Term 1 – Project 1 – Finding Lane Lines on the Road

1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

Basic Pipeline

My pipeline consisted of 6 steps:

1. Convert original image to gray scale

2. Apply Gaussian Smoothing with kernel_size = 5 to this gray image.

3. Apply Canny Edge Detection to the smoothed image, with parameters:

low_threshold = 50 and high_threshold = 150

4. Define a Region of Interest and mask the Canny Edge output.

5. Apply Hough Transformation to this masked image, with parameters:

rho = 4
theta = π/180
threshold = 20
min_line_len = 5
max_line_gap = 4

6. Combine the image and the Hough Lines

7. Save resulting figure into an output path (for individual images)

NOTE: For individual images this pipeline is repeated for every input image in the folder "test_images". A new folder "test_images_output" is created and the output images are saved in it.

Steps 1-6 are constitute the pipeline for the videos.

Draw_lines() function modification

I modified the draw_lines() function to draw a single line on the left and right lanes in the following way:

1. Divide left an right lanes

a) Right lanes are those with positive slopes and x-coordinates in the right part of the image (half of the image in the half of the x axis)

b) Left lanes are those with negative slopes and x-coordinates in the left side of the image (half of the image in the half of the x axis)

2. Each part (left and right) will have their corresponding vectors of x and y coordinates.

3. We can find then (for left and right) the corresponding line parameters (polyfit) and the equation of the line (poly1d)

4. With that, we can draw a line that would define the left or the right line as a single line.

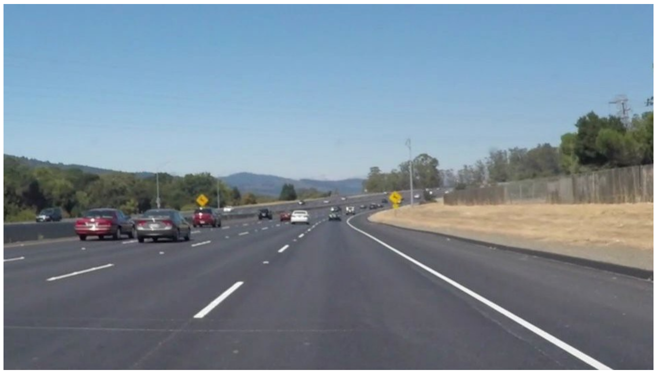
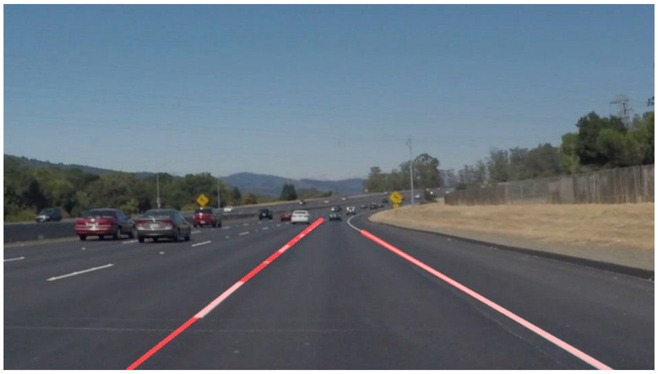
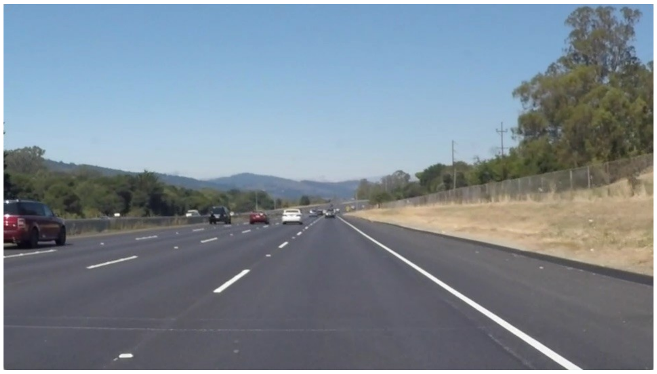
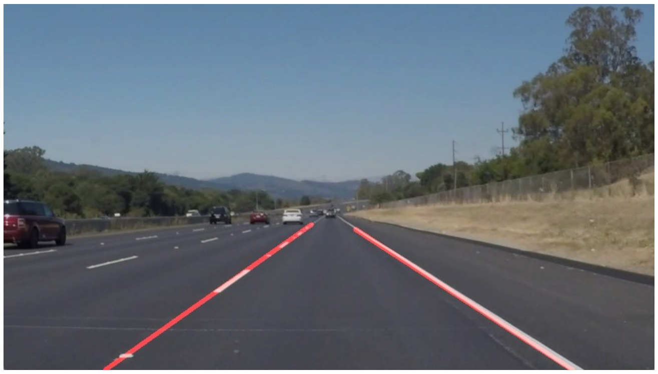
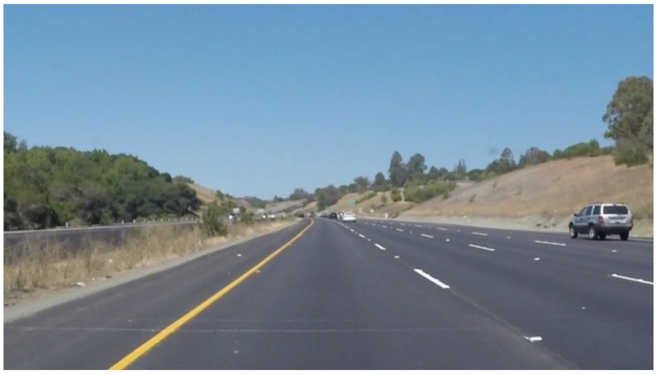
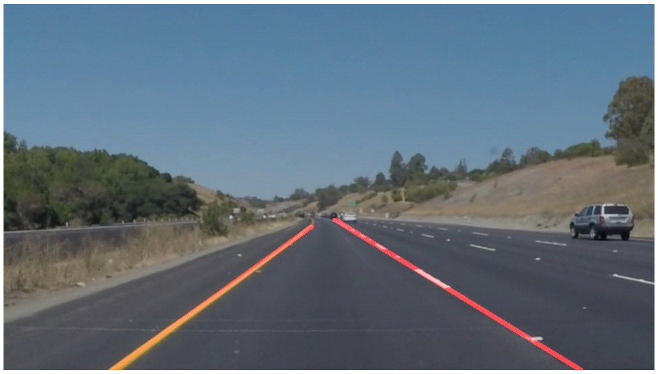
Some Outputs

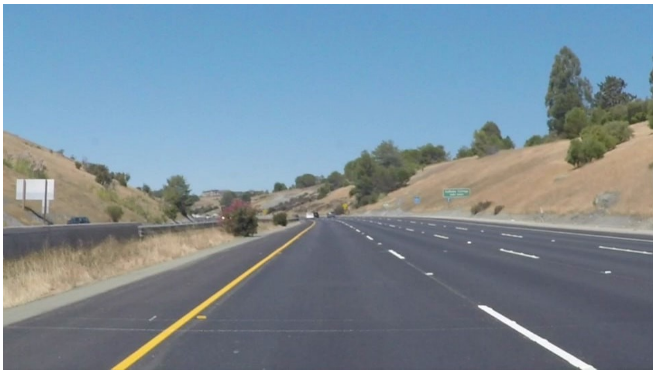
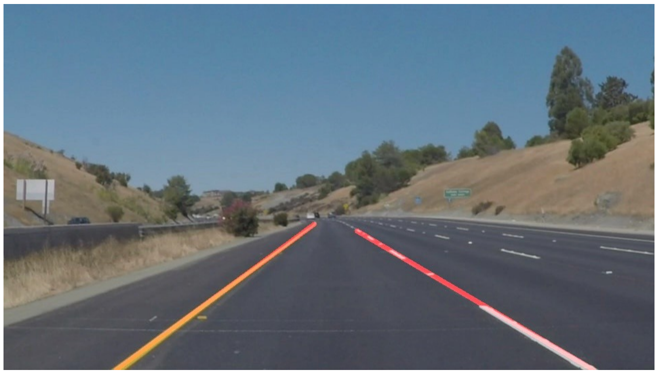
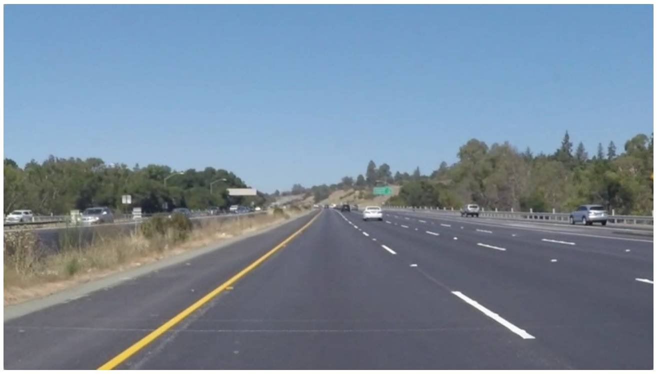
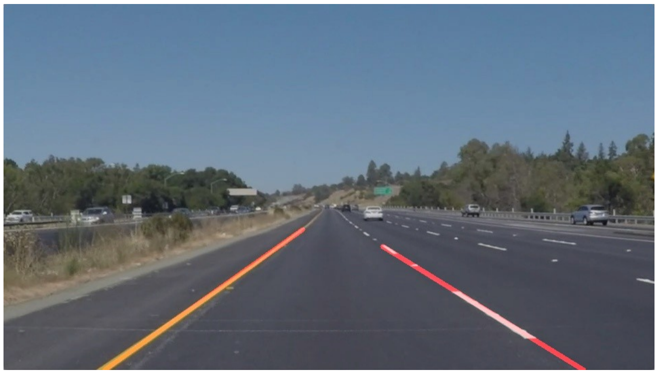
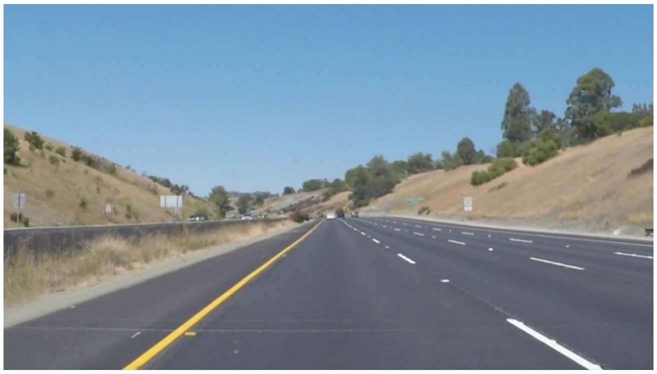
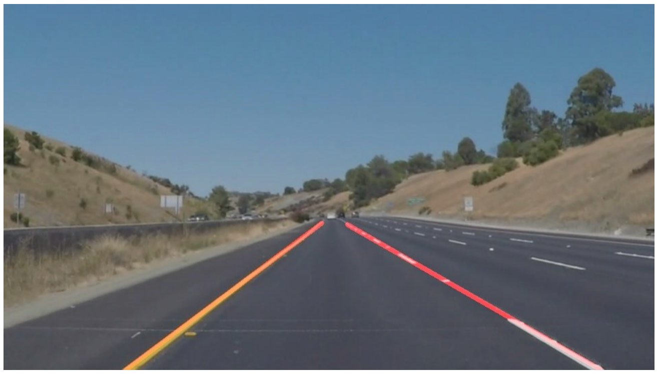
Image outputs are saved in the folder: “test_images_output”

Video outputs are in saved in the folder: “test_videos_output”

Both folders are created if they do not exist.

The following table present some examples of inputs and outputs images.

Input Image	Output Image
	
	
	

2. Identify potential shortcomings with your current pipeline

I have tried the challenge video with no successful result. I think my pipeline has (at least) the following shortcomings:

1. Lighting conditions. My pipeline would not work with differences in the lighting conditions. For example, where there are shades or brighter areas in the road.

2. Weather conditions. In relation with the previous point, this makes me think that in raining conditions, snowing or other weather conditions that may affect the lighting, my pipeline would not work.

3. Curves. As my pipeline is based on lines, I think it would work properly with curves.

4. Road markings. This pipeline would not work when road markings are not properly painted.

5. Vehicle overtaking. I wonder what would happen if a vehicle is switching to my lane. I am also wondering what would happen if I am switching lanes.

3. Suggest possible improvements to your pipeline

1. I think the Canny Edge step could be improved in order to detect lines in different lighting conditions. For this, there may be needed a image preprocessing step.

2. Maybe using polyfit with degree more than 1 would help for curve detection.

3. There are frames in my output where the lines “jump”. Thus, a better filtering may be needed.