Finding Lane Lines on the Road

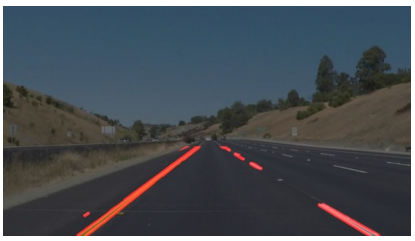
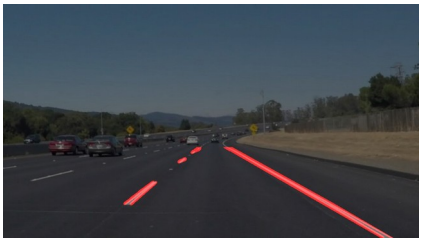
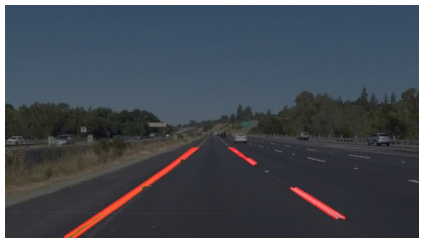
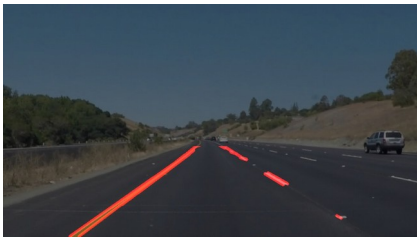
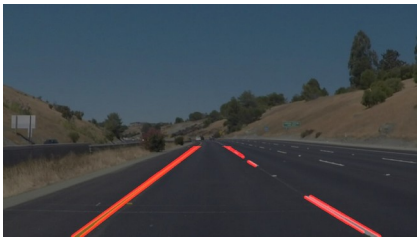
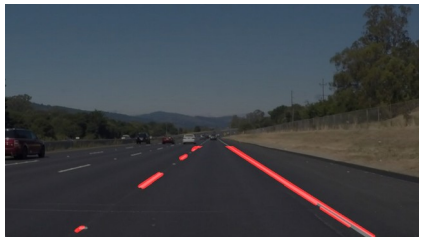
The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report

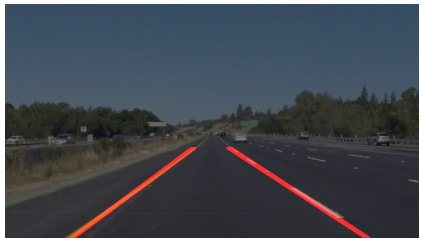
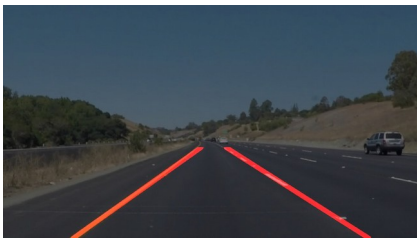
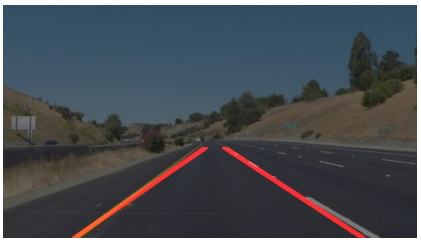
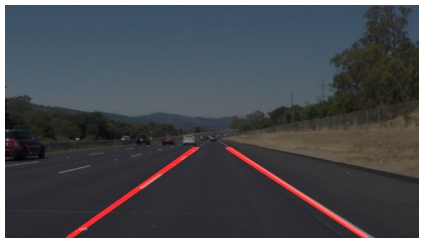
Reflection

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

My pipeline consisted of **6 steps**. First, I **converted the images to gray-scale**, then I applied a **Gaussian/Blur filter** and after I operated on Gaussian image applying the **Canny filter**. The image obtained has been processed in the fourth step by applying **Hough transformation** to find the lines in the image, then I selected the lines inside a **region of interest** obtaining a masked image. Finally, in the last step, I **created a combo image** using the original image and the masked one.

<table>
<tr>
<td> 
whiteCarLaneSwitch</td>
<td> 
solidWhiteCurve</td>
<td> 
solidYellowLeft</td>
</tr>
<tr>
<td> 
solidYellowCurve</td>
<td> 
solidYellowCurve2</td>
<td> 
solidWhiteRight</td>
</tr>
</table>

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by adding a sort of median filter: the filter takes into account all possible good segment in the region of interest (ROI) by watching if the points delimiting the segment are within the ROI. For all good segments, the filter calculates the gradient and the intercept of the straight line passing through the two points of the segment. Then, the filter takes into account the lines with a reasonable gradient coefficient (between 0.1 and 1 for the positive gradient and between -0.1 and -1 for the negative one) and store them in a positive (right lane) or negative (left lane) list (NB: Positive line are on the right due to the fact that the Cartesian coordinates are upside-down). Finally, draw_lines() function operates on the two list by calculating the median of values of each list in order to find the best values for left and right lanes. I also tried with the mean instead of the median but the latter works better.

 whiteCarLaneSwitch	 solidWhiteCurve	 solidYellowLeft
 solidYellowCurve	 solidYellowCurve2	 solidWhiteRight

2. Identify potential shortcomings with your current pipeline

One potential shortcoming is evident when the selected segments, concurring to fill the positive and negative lists, are not well aligned each other. That way draw_lines() function select the representative values of the line not perfectly aligned with the image (as shown in solidYellowCurve2).
Generally the misalignment come from a bad detection of the lines in Hough transformation due to a bad quality of the image, strong lightening inhomogeneities in the image or simply because the lane is curve and not straight.

Another shortcoming come from the fact that the pipeline consider the image as the unique source of data. It doesn't take into account the fact that two different images can be temporally connected and the lane can't change drastically between the two images. It is possible to see the shortcoming between solidYellowCurve and solidYellowCurve2. The two image are very similar but in the latter the right lane is not perfectly aligned. This is particularly evident in the processing of the video. The draw_line() function produces an extremely unstable and jittery lane. It is not possible to work on it.

3. Suggest possible improvements to your pipeline

I stabilized the lane 'prediction' using a buffer to store lanes data between frames. That way draw_lines() function has historical data to support the right choice of the median of each lane parameters. The result is quite good with a buffer's depth set to 20 frames.

Another potential improvement could be a color normalization for that region of the image with strong lightening inhomogeneities.

4. Optional Challenge considerations

The improved draw_lines() function works quite well also on challenge video, after setting the target_resolution to (540,960) in the VideoFileClip interface, because the original file has a resolution about 1280x720.
The only point where is evident the weakness of the function is around the second 4 of the video where the lane is curved and the asphalt color has a drastic change.