Finding Lane Lines on the Road

Project Writeup

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 8 steps:

1. **Read** the image and make a copy. The copy was in order to always have a reference to the original image in order to use it in some of the processing functions.

2. **Make it Grayscale** in order to detect edges easier later to find the lines. Used the cv2 OpenCV library to do this: `cv2.cvtColor()`

3. **Smooth image** with Gaussian Blur function to make it easier for the canny function to detect lines.

4. **Apply the Canny Edge Detection** to create a Gradient Image. This technique highlights the edges on the image, making it much easier to detect the lines.

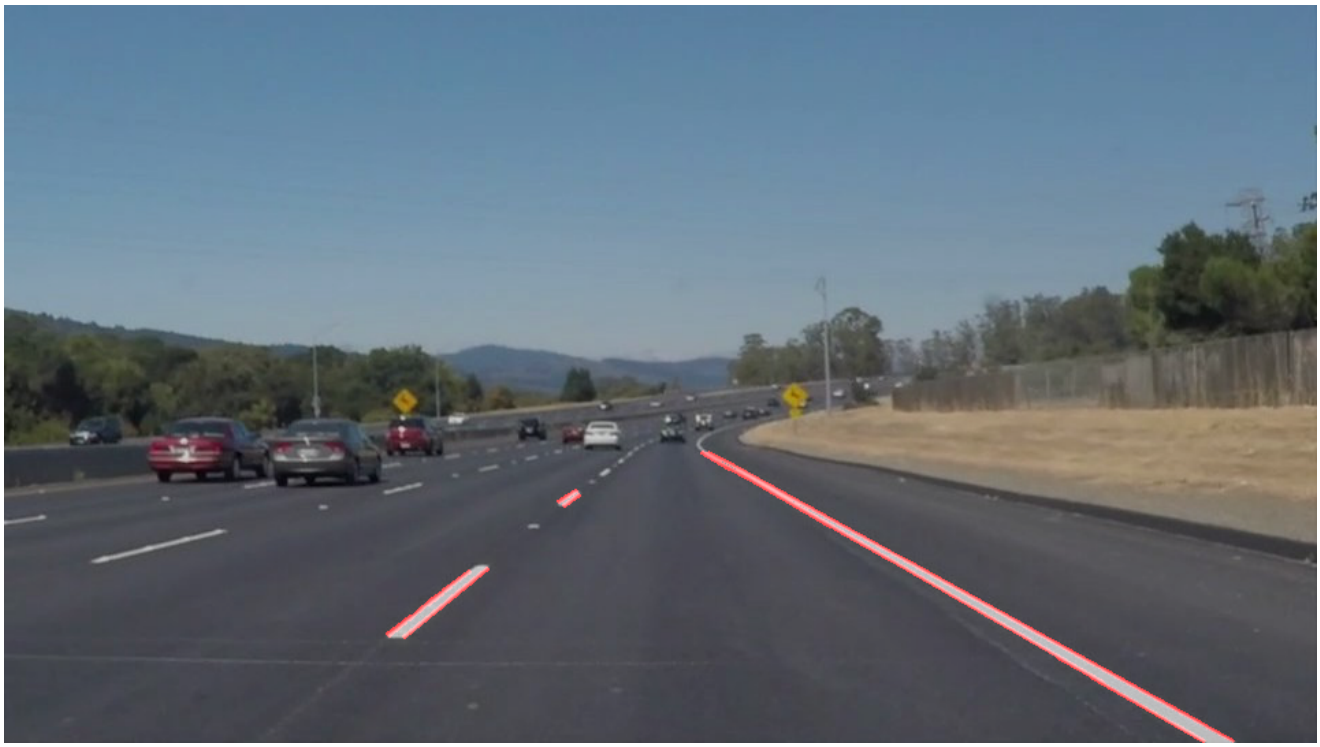
5. **Apply image mask** to remove parts of the image that we are not interested in. Using `cv2.fillPoly()` we create a polygon with a set of defined vertices to only keep the region of interest in the image we want.

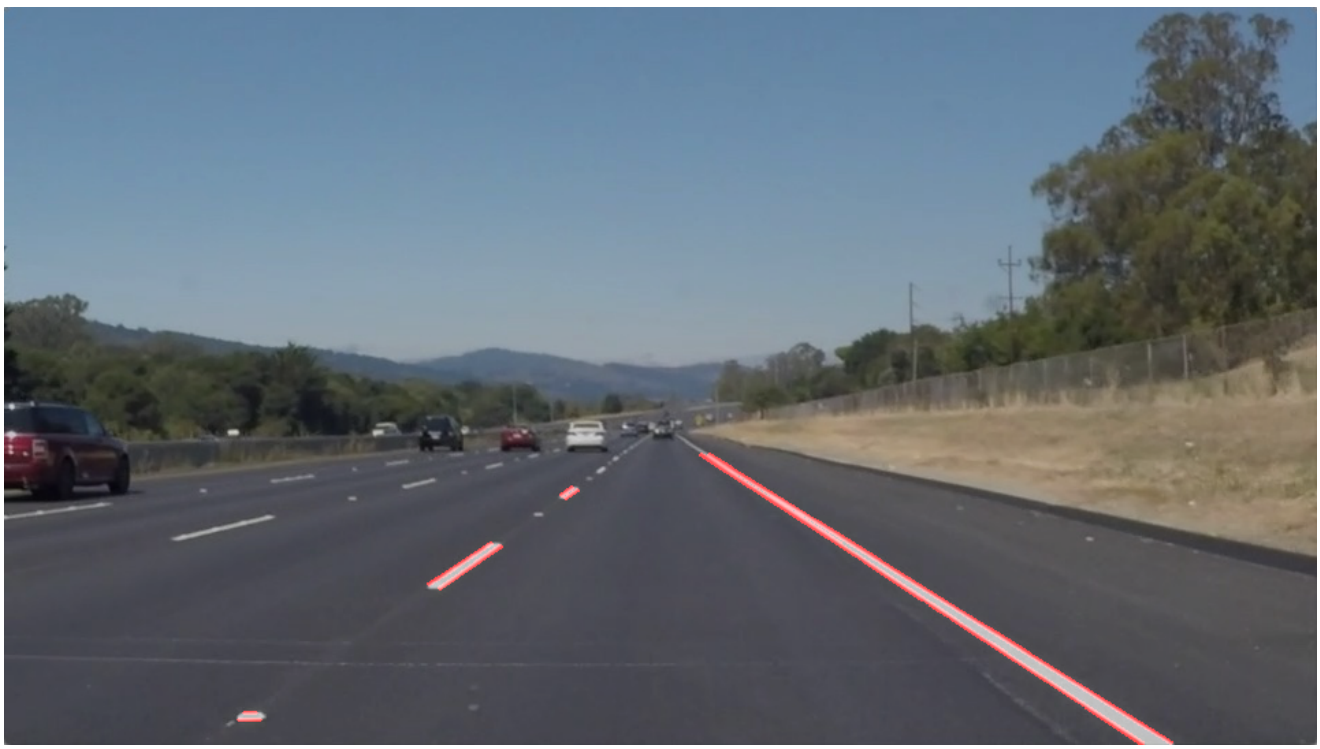
6. **Detect the lane lines with Hough Transform**. This technique allows to find the lines that correspond to the lanes. This technique gives us the points for each line, which we can then use to locate the lanes in the image. By using the parameters

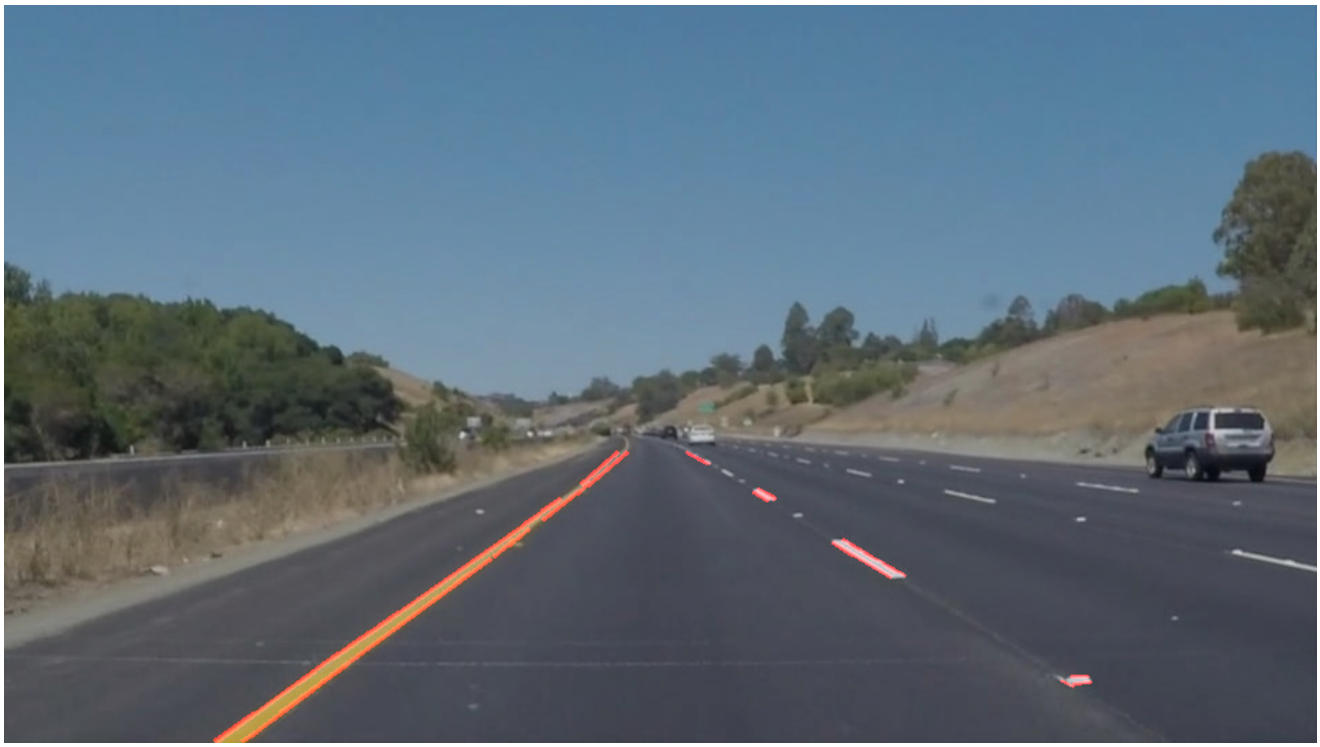
rho, theta, threshold, minimum line length, and maximum line gap, we were able to fine tune the output of the detected lane lines to better detect only the lane lines.

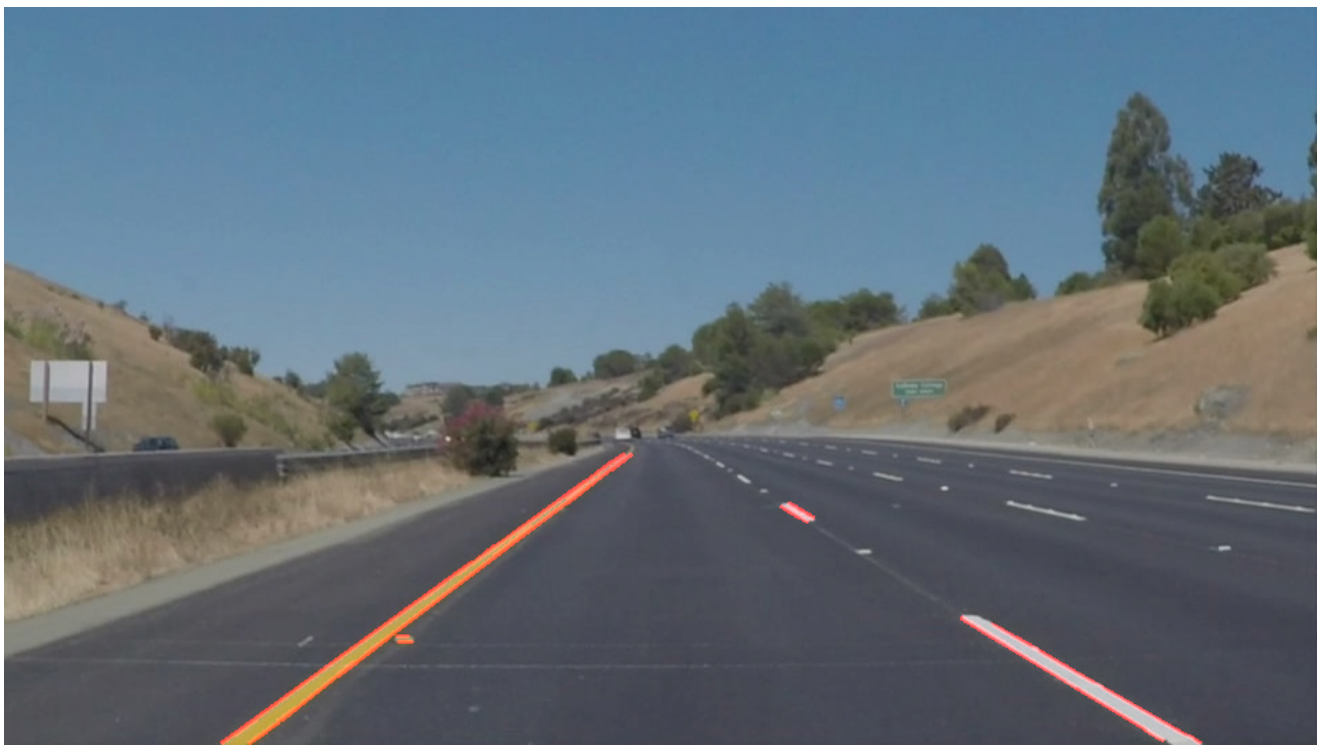
7. **Draw the lane lines on top of the original image**. This steps uses the lines detected by the Hough Transform to draw one continuous line for each side (left and right) on top of the lane lines on the original image. By iterating through each line, I was able to compute the slope of the line to determine if a point belonged to the left or right lane lines. Then the lines were drawn using the slope calculated, the points closest to the top of the image, and the points where Y was equal to the height of the image. Using the equation for a line, I was able to find all the points required. Then using the `cv2.line()` the lines were drawn on top of the original image.

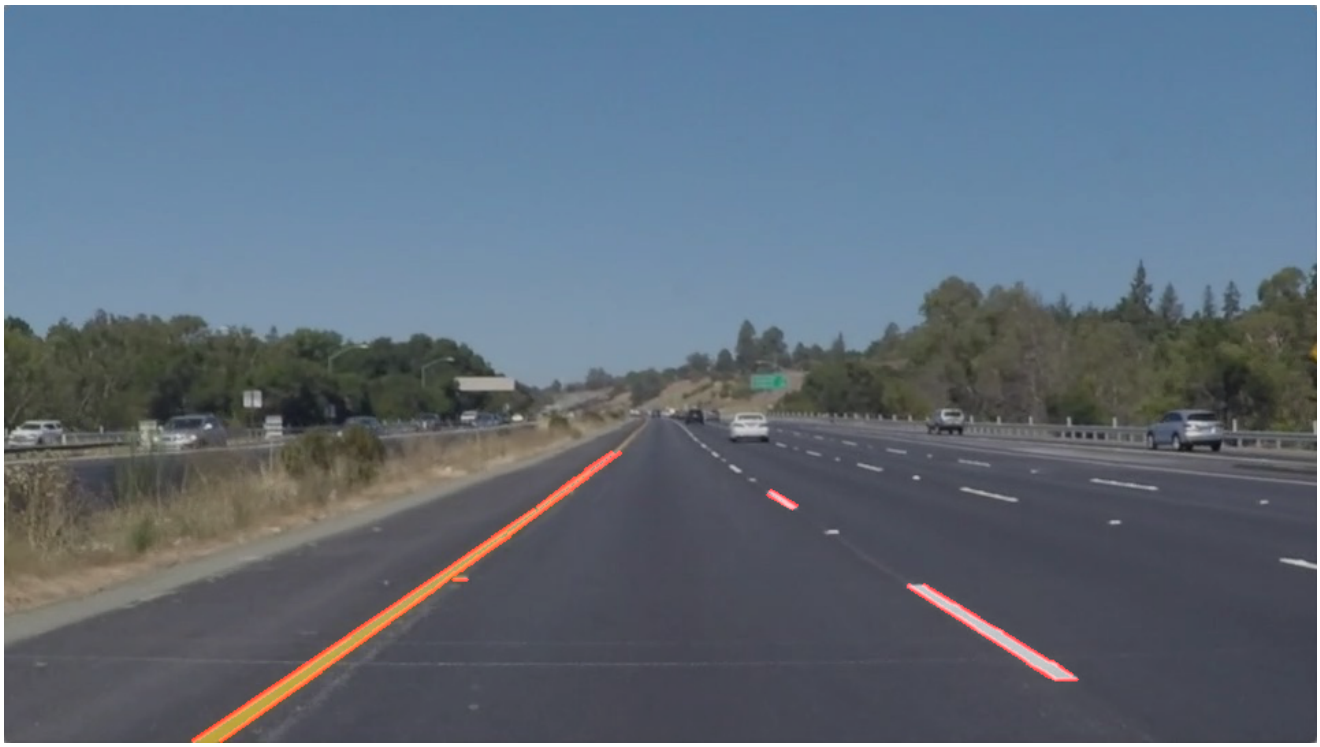
Here are some processed images using the original Draw Lines helper function:

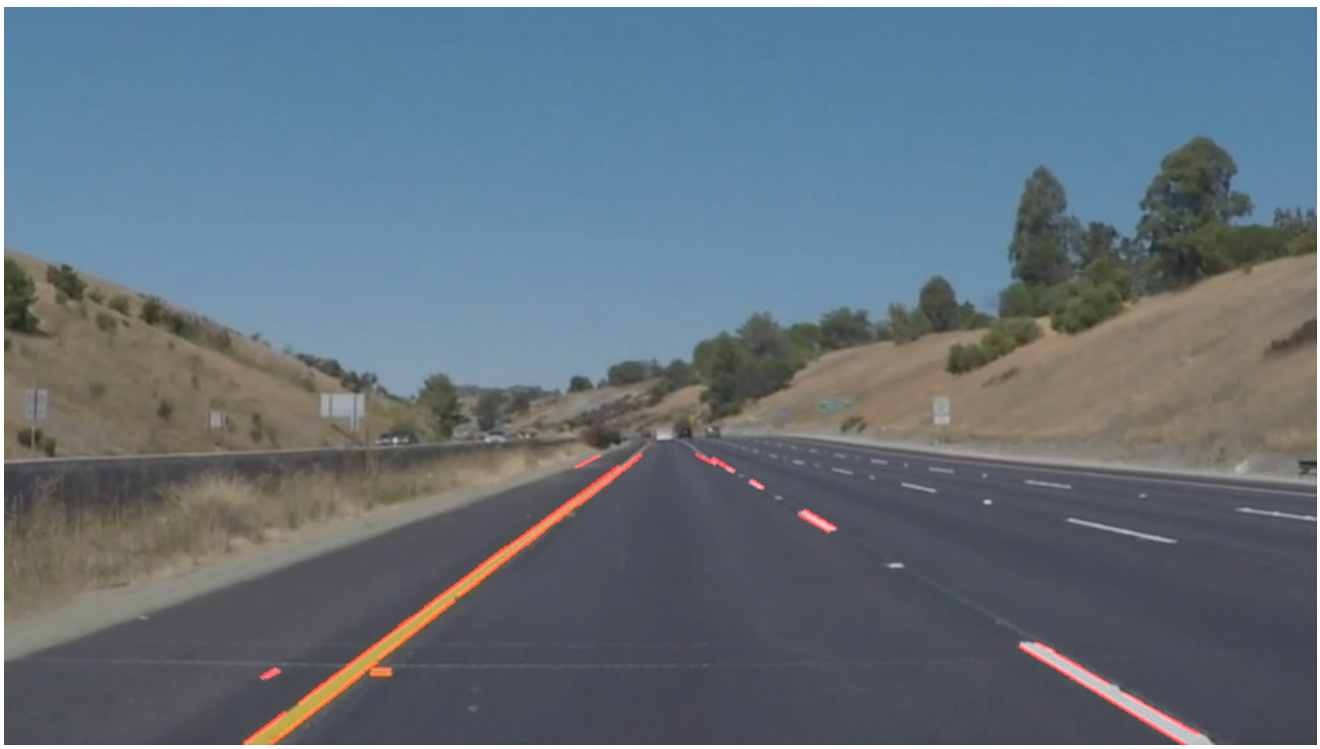












2. Identify potential shortcomings with your current pipeline

One potential shortcoming is that we won't be able to identify the lanes if we don't have the lane lines. Many roads actually don't have lane lines. The road should also be detected in order to protect against having no lines.

Another problem could arise if there are markings or special traffic signs on the middle of the lane. The algorithm would detect these lines and the current pipeline would break.

Also, the mask to select the region of interest is estimated using the image shape, but this might not apply an all cases. Lanes might have different widths, etc...

3. Suggest possible improvements to your pipeline

More effort to find the optimal Hough Transform parameters to better detect the lane lines. It was noted that still some small lines that do not belong to the lanes were still detected.

A better masking technique to avoid detecting any traffic signs on the middle of the lane.