Finding Lane Lines on the Road

Finding Lane Lines on the Road

The goal of this project is to find the left and right lanes on the sides of a car.

Pictures of the road taken from the car will be processed through a pipeline. The output will be the same image with the lane lines highlighted in red.

The same technique will be used to identify lane lines in a video sequence. The video will be considered as a sequence of images and each image will be processed through the pipeline and ensembled back together into a video.

The result of that process will be a video where the lane lines are highlighted in red.

Original image:

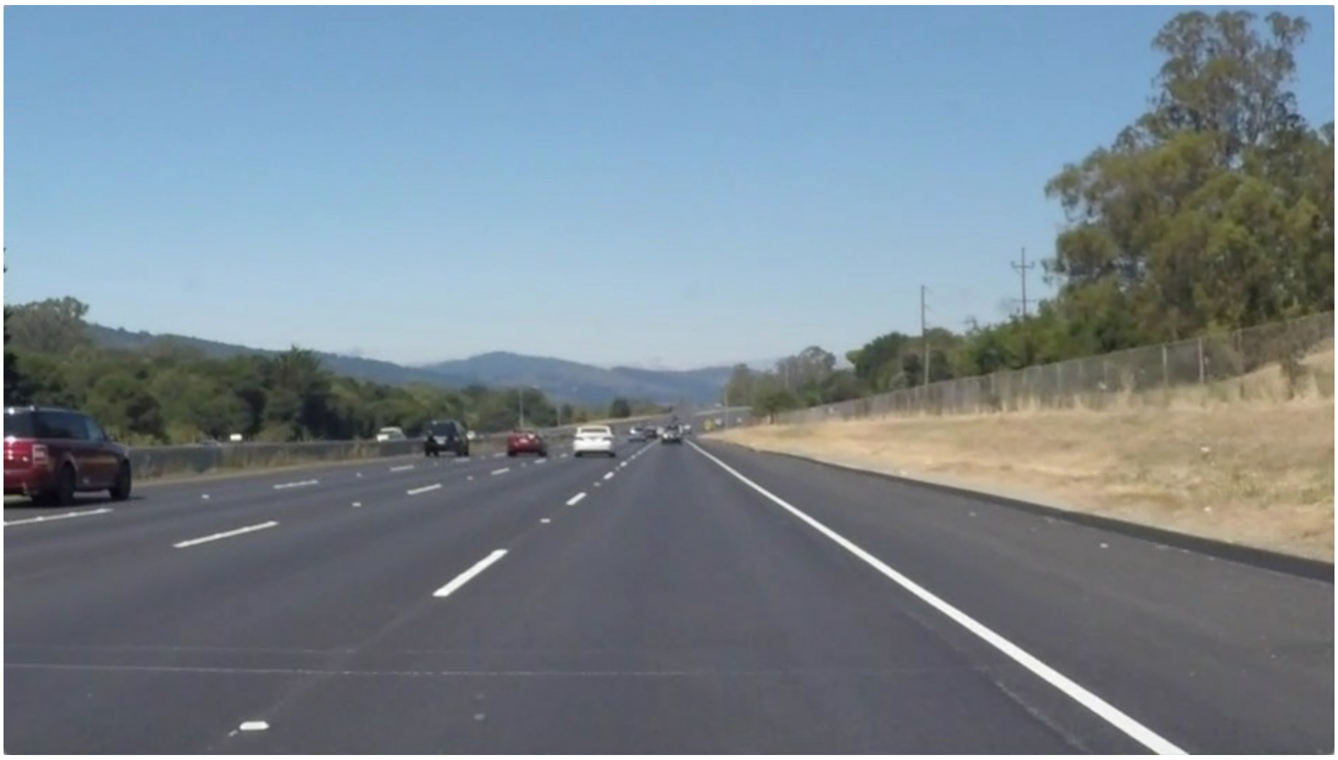
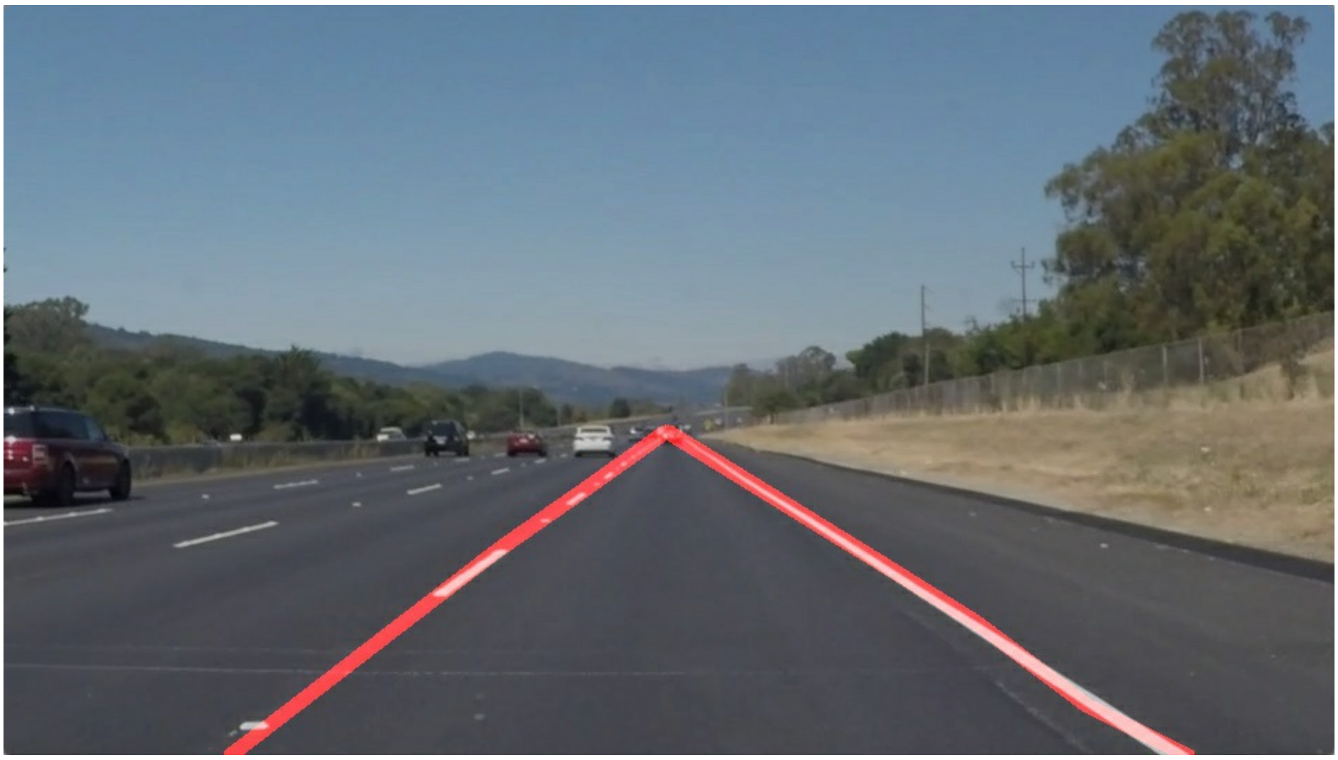


Image processed:



Reflection

My pipeline consisted of 5 steps.

1. First, I convert the images to grayscale. To perform this transformation to the image, I have chosen to use the OpenCV cvtColor function.
2. Then, I apply Gaussian smoothing to the function. Blurring the image is usefull to ignore imperfections in the image when detecting the lines in the next step. To perform this transformation to the image, I have chosen to use the OpenCV GaussianBlur function.
3. The next step in my pipeline uses a canny edge detector to identify edges in the image. The OpenCv Canny edge detector has been used to perform this step.
4. Then, a mask has been applied to the image to reduce the area where the lines should be detected. For instance, it is not very likely to find a lane line on the top of the image, where the sky is usually present.

5. After that, a Hough line transform is used to detected straight lines in the image.
6. Then, the images are grouped into two different sets (left_lane and right_lane) depending on its slope. In addition to that, the images of each set are sorted by its vertical position.
7. For each set, the lowest vertical point is selected and a line from the bottom of the image to that point is drawn. The `x position` of that new line is calculated using linear extrapolation.
8. Finally, a line from the lowest `y positon` to the highest `y position` is drawn per each lane lines set.
9. To prevent errors, lines with a slope lower than 0.3 has not been drawn.

2. Identify potential shortcomings with your current pipeline

One potential shortcoming with my current pipeline is when the lane lines cannot be split into two different sets based on its slope. That might happen in a hairpin curve.

3. Suggest possible improvements to your pipeline

A possible improvement would be to split the line sets using a k-means algorithm.

Also, it could be used a different method to extrapolate the positon of the lane lines between the bottom and the top of the images. Maybe ignoring the lines which are not between the Q1 and Q3 percentille might work.