Lane Detection

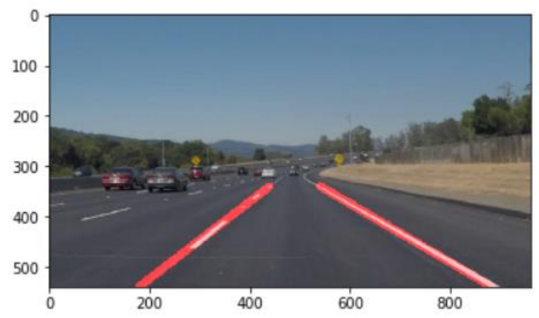
1. Describe pipeline. As part of the description, explain how modified the draw_lines() function.

My pipeline consists of 6 steps. Firstly, the image is converted to grayscale, except the video pipeline. Secondly, the image is smoothed by using gaussian_blur. Thirdly, the canny transform of the thresholds established and the transform is applied. Fourthly, a section of the image where the lane lines are most likely to be is masked. Fifthly, the masked section is hough transformed to pick out continuous lines and two subsections of lines which pass respective slope tests are chosen and converted into a single line each representing their average. Finally, these two lines are drawn on the original image.

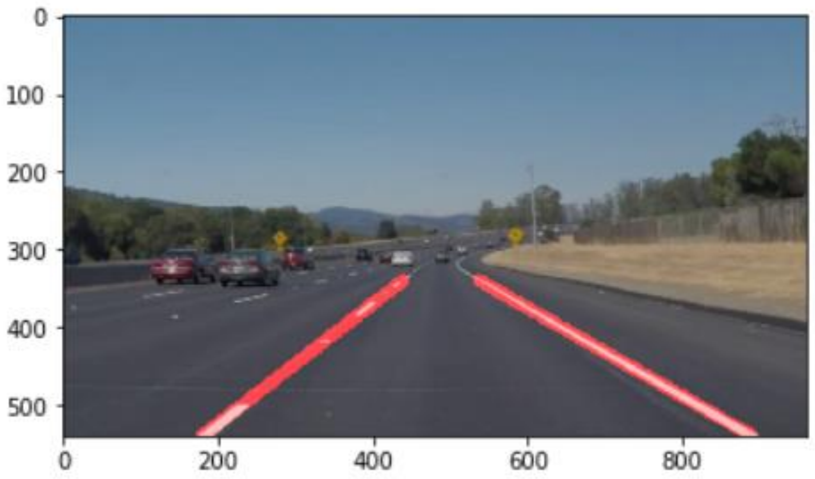
1.1 Reference section

The draw_lines function was heavily modified which is refered from github to create a single lane line for each lane line in the image. Firstly, I created a nested for loop to loop over the lines and grab their coordinates. Then I calculed the slope using the coordinates and filtered the lines into "left" and "right" categories based on their slopes. At the same time I started accumulating their x and y values to be averaged later. Then the slopes and coordinates of each of the categories were averaged and the endpoints of the lane lines were extrapolated using the average coordinate and the slope for each. Finally, the function then draws the lines on the original image. And I have a error that is not effect to result in first, and the others as follows.

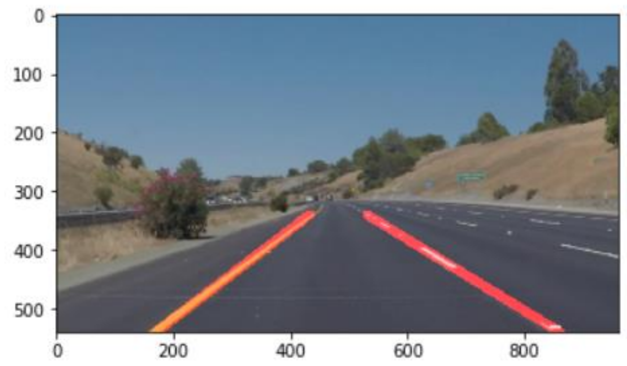
```
    584     def set_data(self, A):

  FileNotFoundError: [Errno 2] No such file or directory: 'test_outputs/solidWhiteCurve-lines.jpg'
```



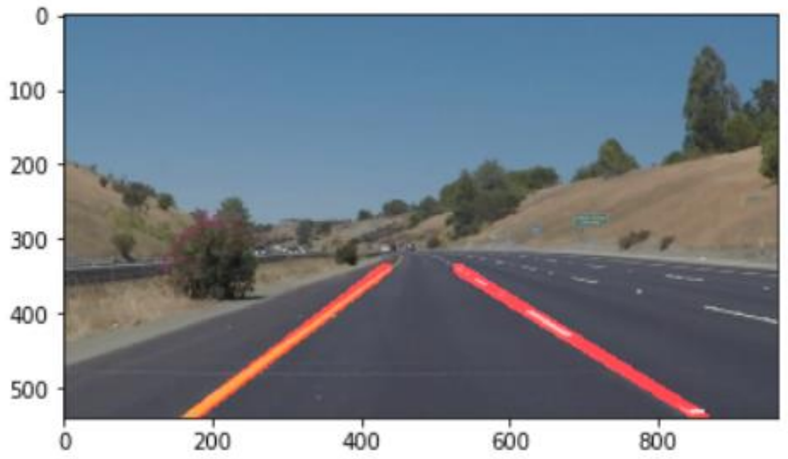
```
[MoviePy] Writing video test_videos_output/solidWhiteRight.mp4

100%|██████████| 221/222 [00:18<00:00, 11.75it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: test_videos_output/solidWhiteRight.mp4

CPU times: user 7.96 s, sys: 1.14 s, total: 9.1 s
Wall time: 20.4 s
```

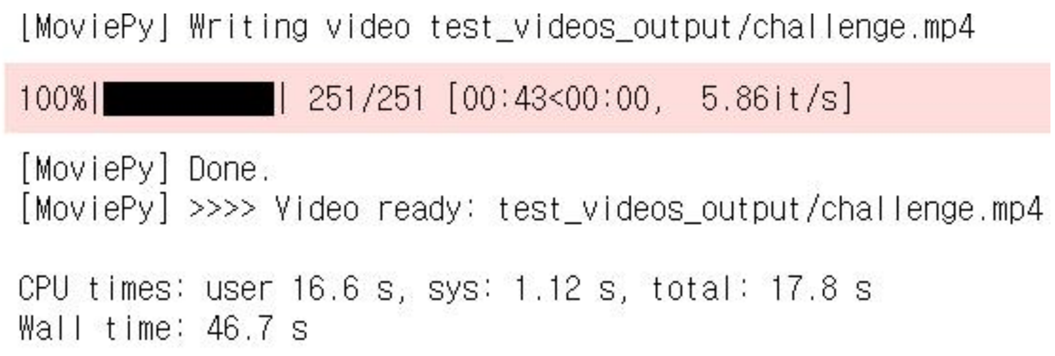


```
100%|██████████| 681/682 [00:59<00:00, 11.36it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: test_videos_output/solidYellowLeft.mp4

CPU times: user 24.8 s, sys: 1.2 s, total: 26 s
Wall time: 1min 1s
```

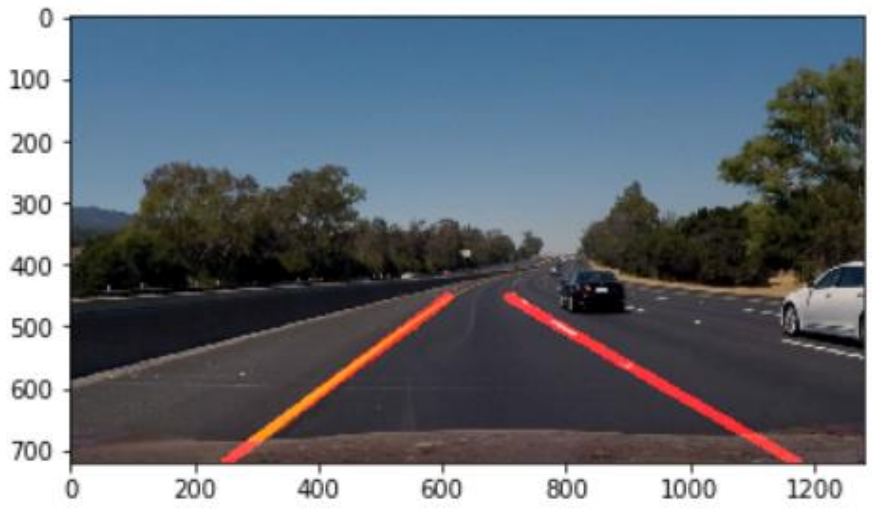


```
100%|██████████| 681/682 [00:59<00:00, 11.36it/s]

[MoviePy] Done.
[MoviePy] >>>> Video ready: test_videos_output/solidYellowLeft.mp4

CPU times: user 24.8 s, sys: 1.2 s, total: 26 s
Wall time: 1min 1s
```





```
[MoviePy] Writing video test_videos_output/challenge.mp4
100%|██████████| 251/251 [00:43<00:00,  5.86it/s]
[MoviePy] Done.
[MoviePy] >>>> Video ready: test_videos_output/challenge.mp4

CPU times: user 16.6 s, sys: 1.12 s, total: 17.8 s
Wall time: 46.7 s
```



2. Identify potential shortcomings with your current pipeline

One potential shortcoming of this method is that it doesn't function if the lane lines are a very similar color to the road, or if the road is very reflective. Another is that it likely will lose a lot of precision in tight turns because there will be far fewer lines which satisfy the slope test. Finally, the processing is quite slow, sometimes taking twice as long as the video it's working from. Whether this is as a result of FFMPEG or the pipeline itself is unclear. Regardless, it predicts lane lines to a good distance.

3. Suggest possible improvements to your pipeline

One possible improvement is to convert all of the yellow in the masked area to white before doing canny and hough transforms to maximize contrast and thus maximize the number of lines. Another would be to allow for it to determine significant curves in order to predict upcoming turns.

Overall though, this pipeline is very successful at the task set for it. It even compitantly completes the optional challenge task which is on a more curved road and with changes in the asphalt.