Finding Lane Lines on the Road - Resubmission

The goals / steps of this project are the following:

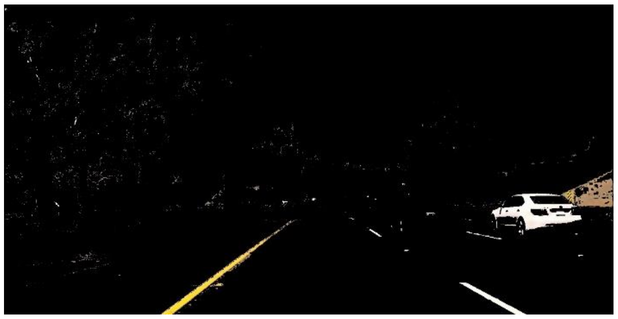
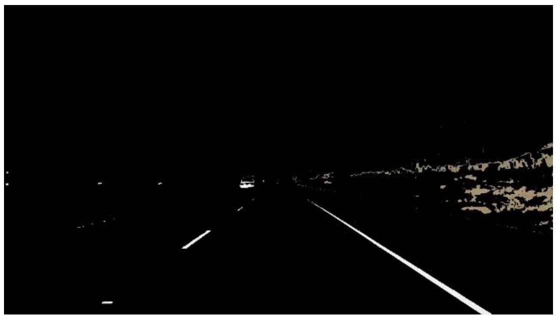
- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report
- **New content for resubmission in Red Text**

Reflection

My pipeline construction consisted of 8 steps. I will demonstrate the process step-by-step by using the following 2 example pictures/images. The image on the right is from the test_images directory, the image on the left is a screenshot from the challenge video, with the tree shadow on the road.

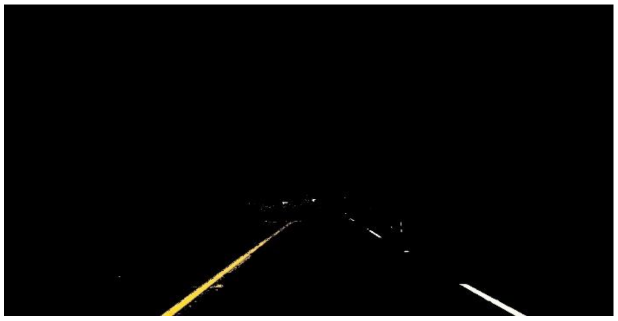
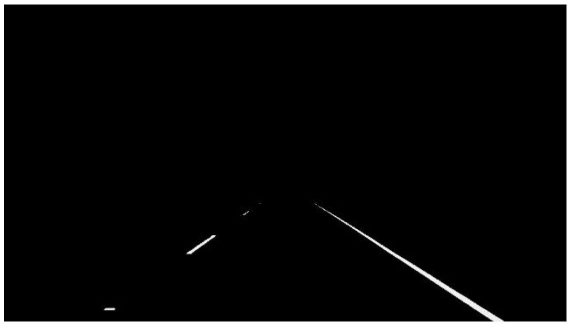
Original images

1. I extracted the Yellow and white lines from the images
   a. I defined ranges for yellow and white colors separately and extracted the pixels that fall in range to those RGB thresholds
   b. Then I combined both yellow and white pixels and used them to mask the main image
   c. Thereby getting an image, which has just yellow and white colors.

After extracting yellow and white colors

2. The next step was to define the region of interest to remove the image segments that are not required

<u>Region of interest defined</u>

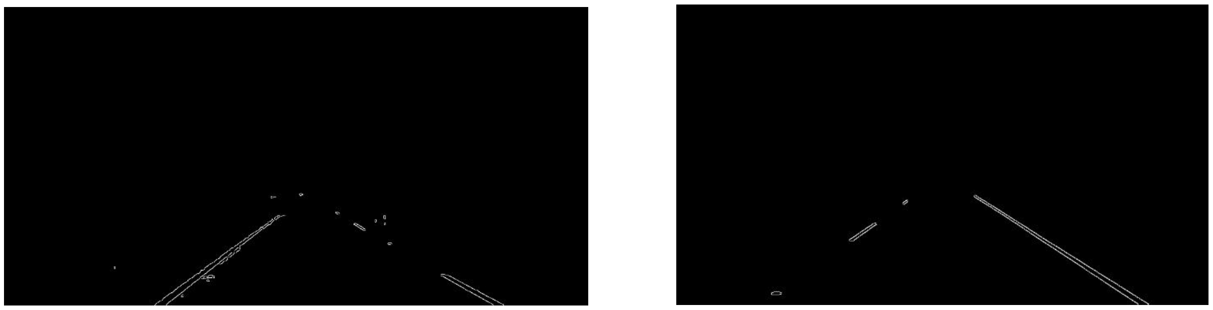
3. Then I converted the processed image to grayscale

<u>Grayscale conversion</u>

4. I used Gaussian blur to smoothen the image so that a roughly clear with consistent points could be obtained
   a. Kernel size of 7 was chosen after assessing images with varying kernel sizes to get the best result

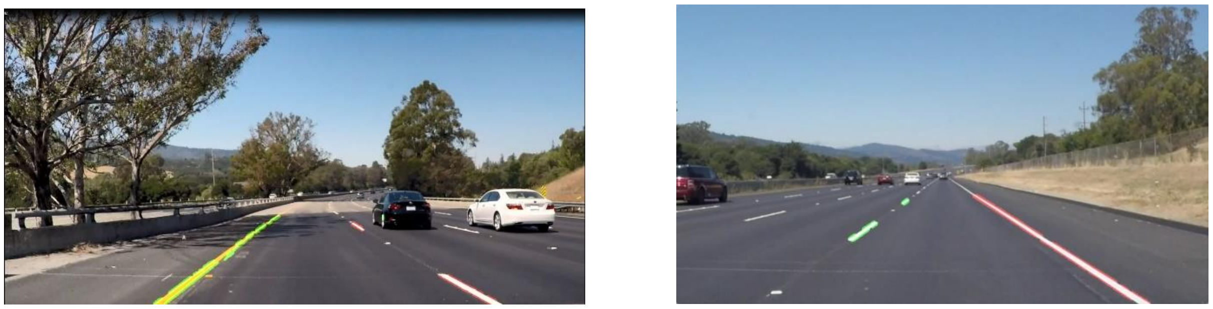
Gaussian blur applied images

5. The Canny filter was applied on the blurred grayscale image to define the canny lines
    a. Lower threshold = 100 and Higher threshold = 200 gave the best results for the images and the videos



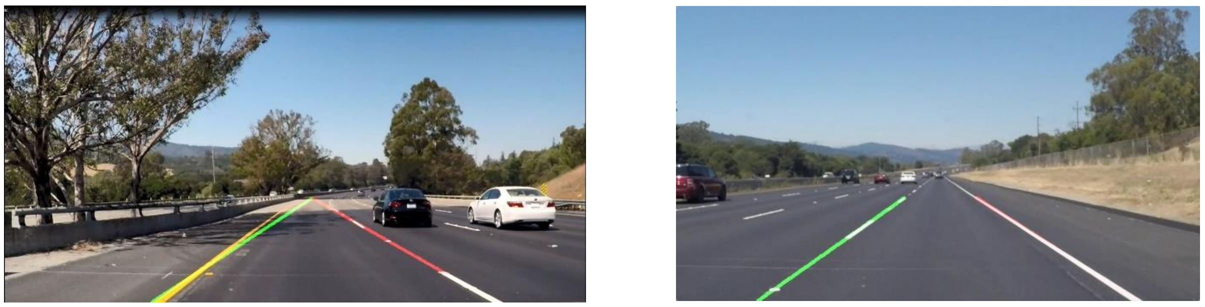
Canny filter applied on above images

6. Hough filter was applied to find hough line coordinates on the canny image



Lines identified using the hough line transform

7. The next step was to find the slopes of all the lines and extract the right and left lines.
    a. If the slope of the hough line coordinates was more than 0.3 then the line was appended to the right lane matrix.
    b. If the slope of the hough line coordinates was less than (-0.3) then the line was appended to the left lane matrix.
    c. Then I merged all the X coordinates of left lines in one array Y coordinates in another array and repeated the same process for the right lines.
    d. Eventually I have a lot of X,Y coordinates that could define right and left lines of a lane.
8. Finally, Polyfit function was used to define a line (1st order) that would fit all the X.Y coordinates for left lane as well as right lane. Then these lanes were added on top of the base image to define the left and right lane markers.



Final lanes projected on the initial images

All the image processing is done in the process_images() function.

The same function can be used to process the images as well as videos

- As per the advice of my reviewer and my mentor, I took average of slopes and intercepts of the polynomial fit I obtained from the Hough Transform images. That really helped me to remove the jittery lane lines and the videos show the lanes to be a lot more consistent than before.

Main GitHub directory: https://github.com/sanyam89/Project_1_finding_lanes_lines_REsubmission/

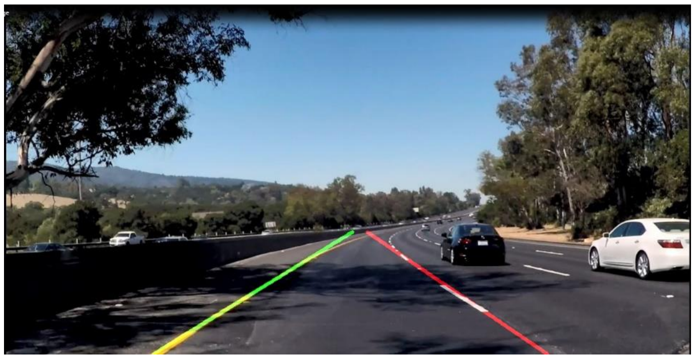
Python code location: **main repository**

The original and processed images are located in this location: **Test_images** repository

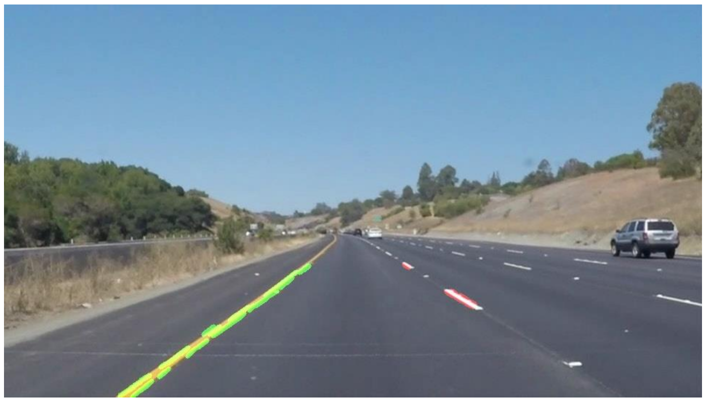
The final processed videos are located in this location: **Test_videos_output** repository

Potential shortcomings with my current pipeline

1. Sometimes it is hard for my pipeline to define the lines correctly, when the road is turning, as shown in the picture below.

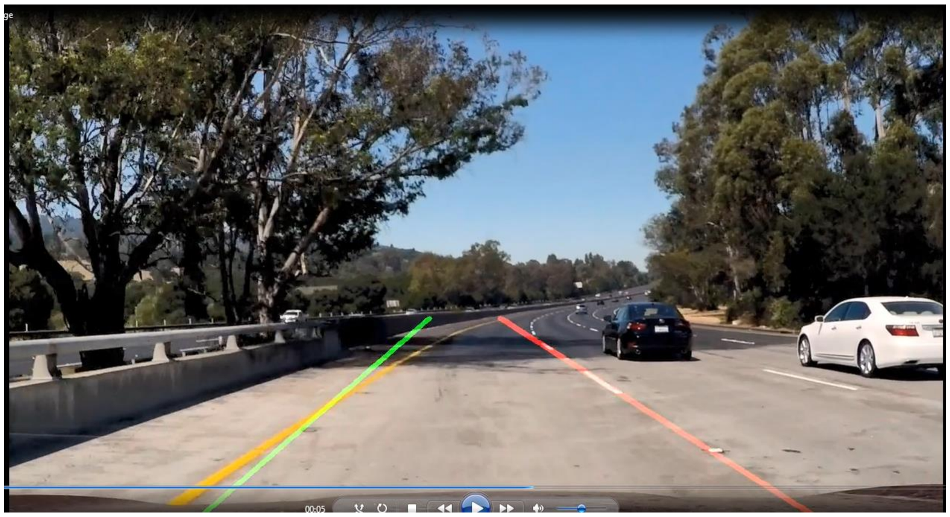


2. Also, the small reflective markers get eliminated from the lane after the hough transform as I am eliminating horizontal lines (<0.3 and >-0.3 slope)



After implementing the average of slopes over multiple frames in the resubmission, this problem was resolved.

3. If there are some shadows on the road, my function is not able to identify the color change effectively.



4. As my code is just focusing on a small portion of the whole image and specific colors, if there is an obstruction on the road, like an animal, my code would not be able to identify it and eliminate it while processing the image.
5. In addition, there are multiple cameras located on an autonomous vehicle, so different regions of interests would need to be defined for all the cameras separately to identify the lane markings effectively.

Possible improvements to your pipeline for future

1. A possible improvement would be to use a higher degree polyfit along giving more weightage to small mane markers closer to the bottom of the screen.
2. Another potential improvement could be to convert RGB to HSL / HSV image and then it might be better to extract the lane lines when there is shadow on the road or the lane lines are not of the specific color regions that we have defined in the code.