Finding Lane Lines on the Road

The goals of this project are the following:

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

1. Pipeline

The pipeline is provided separately as an .ipynb file (P1.ipynb)

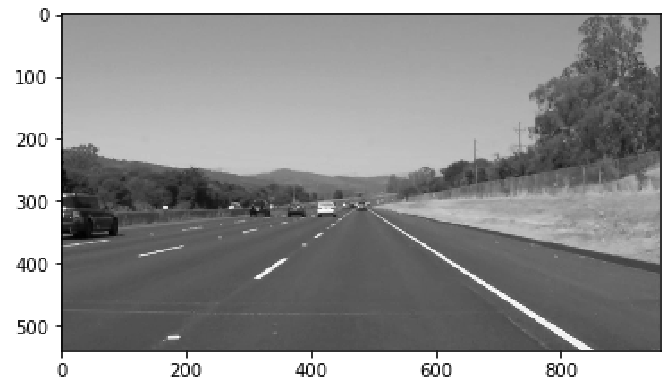
2. Reflection

2.1. Pipeline Description.

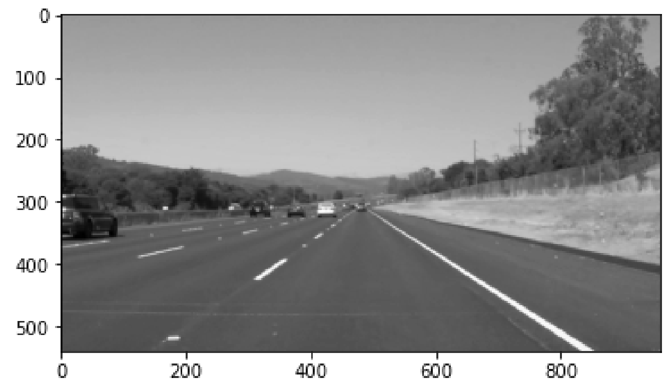
The pipeline implemented in this project consists of the following steps:

1. **Reading an image / Extracting an image from a video file**.
   The read/ extracted image is the starting point from which we will apply computer vision techniques to identify lane lines.

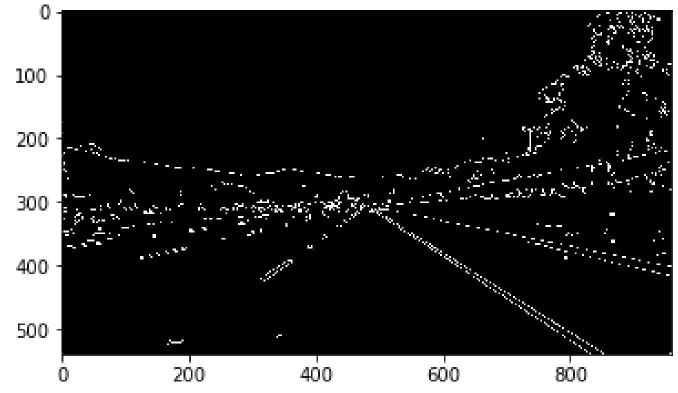
2. **Grayscaling the image**.
   The image is first grayscaled to make the line detection algorithm unbiased in gerard to the line color. The rationale is that the lane lines should be detected regardless of their real or perceived color (i.e. regardless whether they are, say, white or yellow, or whether they are reflecting the ambient light in different ways depending on the meteorological conditions).



3. **Smoothing out the image**.
   This step is necessary to minimise the possibility of identifying spurious lines in the image due to noise or “random” sharp contrasting regions.

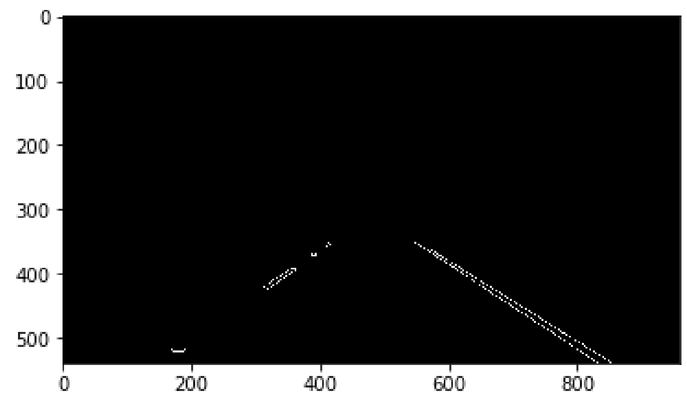


4. **Applying the Cunny Transform**.
   This transform allows us to identify sharp contrasting regions in the image. The result of such transform is a black and white image where only the boundaries of the objects are outlined.

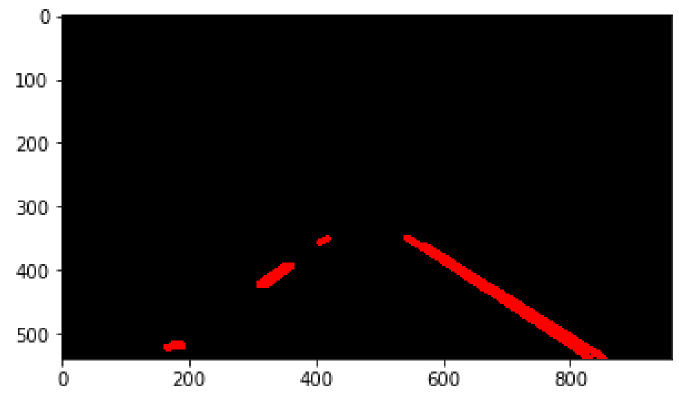


5. **Creating a masked edges image**.
   This step helps to focus on the region of the image where we expect to see lane lines. In the implementation of the pipeline I chose a triangular region covering

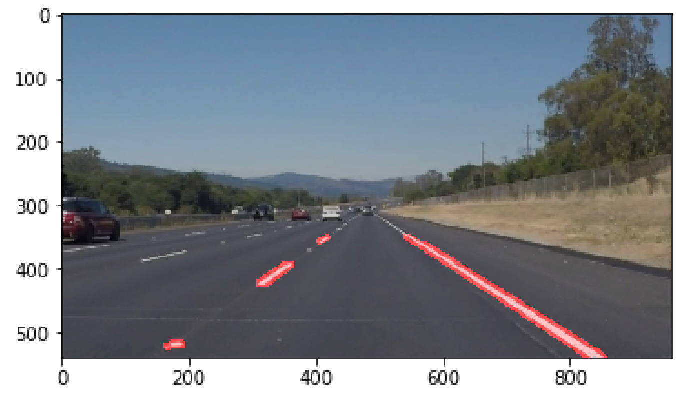
the bottom part of the image.



6. **Identifying lines through the Hough transform**.
   This is the actual algorithm that identify the lane lines from the black and white edge-image generated through the Canny transform. The lines identified by the algorithm are drawn in the mask image.



7. **Overimposing the original image with the mask image**.
   This is the final step of the pipeline where we merge the original image with the mask image containing the lane lines drawn in the previous step.

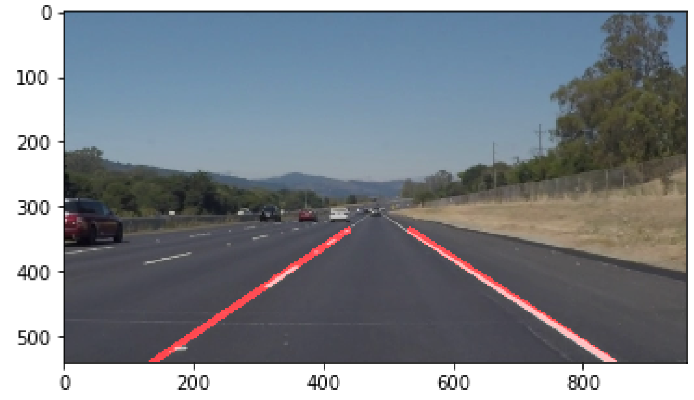


2.2. Improvement of the pipeline

An improvement of this pipeline consists of drawing a continuous line where a dashed lane line is detected. This was done by modifying the function draw_lines().

In the modified version of this function function the different lines detected through the Hough transform were separated according to their slope. Negative slopes were indicating lines/segments belonging to the left lane-line, while positive slopes were indicating lines/segments belonging to the right lane-line.

The positions of the two groups of line/segments were averaged and the top and bottom of each lane-line extrapolated.



2. Identify potential shortcomings with your current pipeline

I noticed that in the video solidYellowLeft.mp4 the lines drawn by the algorithm were flickering at times. This was due to the fact that some of the line segments identified by the algorithm had a very small slope (in absolute value) compared to the others. The likely reason for that is that these segments are spurious, or at least they do not belong to the lane-lines. When averaged with the other identified line segments, these outliers skewed the result. To avoid this problem I filtered out line segments with a slope smaller than 0.4 in absolute value.

This solution however could lead to the algorithm having difficulties in recognising sharp bends.

3. Suggest possible improvements to your pipeline

A possible improvement to the pipeline could consist of filtering out line-segments outliers by checking whether their slope is included in the range of value [AvgM - n *

StdM, AvgM + n * StdM], where AvgM is the average slope of the line segments with positive or negative slope, StdM is the standard deviation and n is an appropriate integer.