Finding Lane Lines on the Road

- Name: Manas Mukherjee
- Course: Self-driving car nanodegree.
- Date: Jan 20, 2020

Reflection

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

My pipeline consisted of 5 steps. First, I converted the images to grayscale, then I applied Gaussian smoothing using a kernel of size 7 to suppress noise and spurious gradients. After smoothing the image, I used the 'Canny' edge detector to find strong edges based on the two threshold parameters. Since I am only interested in the lanes, I used a polynomial mask to focus on the edges found on the road. I then applied the 'Hough' transformation method on the masked image and found the possible line-segments which represent lanes.

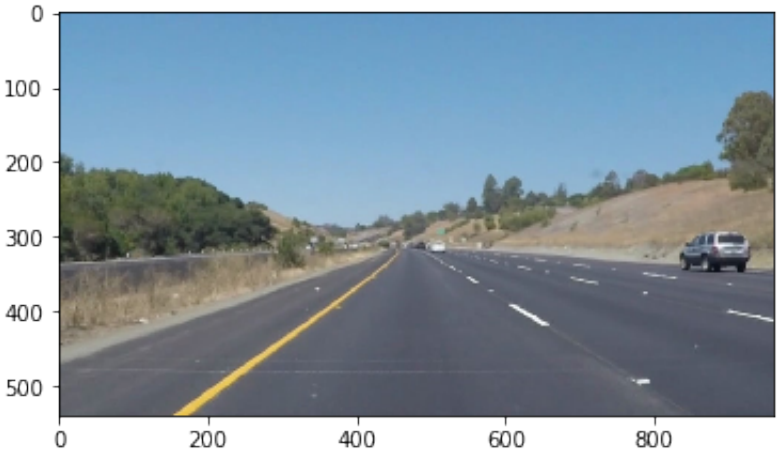
In order to draw a single line on the left and right lanes, I modified the draw_lines() function by identifying the slope and intercept of each line segment, and then taking a weighted(by using the length of the line-segment) average of them. The lines with negative and positive gradients are processed seperately. Here the line-sements with a negative gradient form the left-lane and the lines with +ve gradient form the right-lane. To draw the line, it was required to identify the two endpoints of the left and right lanes. Here, y1 is equal to the height of the input image ie. img.shape[0]. It represents the y coordinate of the end-point which touches the bottom part of the image. y2 is equal to a fraction(0.6 - found by trial and error) of the height, ie img.shape[0]*0.6. This point represents the y corrdinate of the other end of the lanes.

Using the y1 and y2 values, line specific gradients and intercepts, the x coordinates of the two endpoints(x1 and x2) were identified. Then those two lines were plotted on the image using the line() method of the OpenCV library.

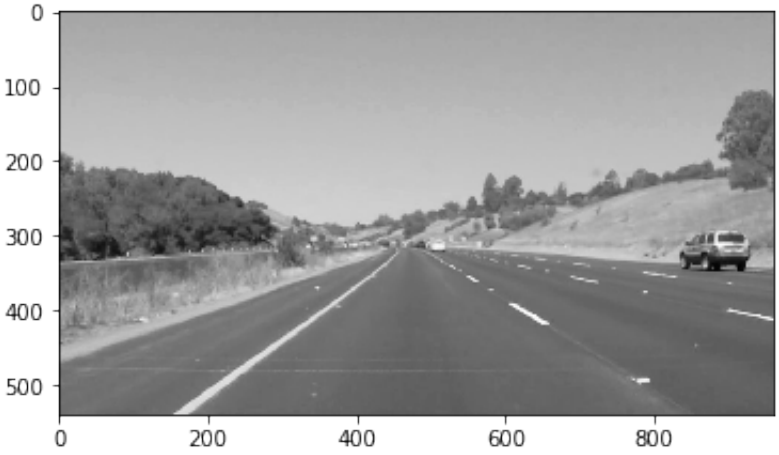
Here is an example tranformation of an input image in the different phases of the processing pipeline.

image:

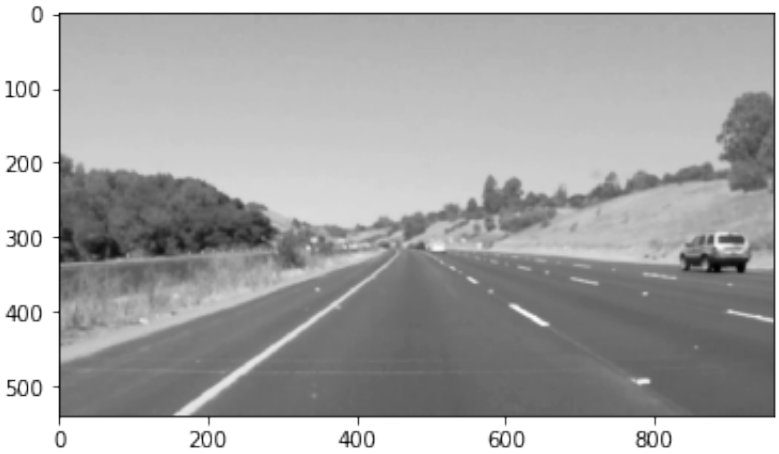
- 1. Original Image



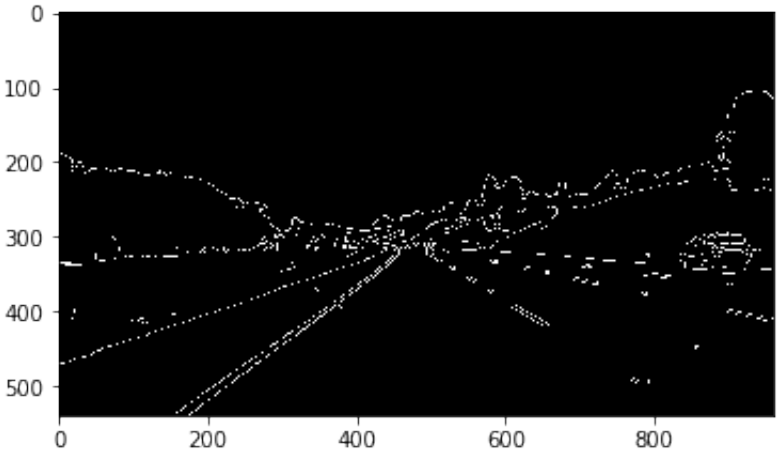
- 2. Grey Scale



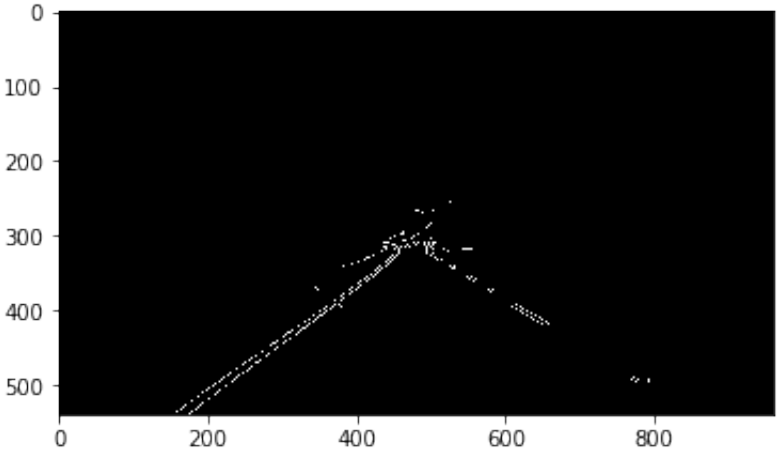
- 3. Smoothing using gaussian kernel



- 4. Edge detection using Canny algo



- 5. Hougs transformed



2. Identify potential shortcomings with your current pipeline

- The current approach may not work if the road or the lanes are curved.
- It might not be easy to find an appropriate mask to suppress the noises (edges other than the lanes).
- If the video is not stable, the reference frame would be changed in every video-frame(image). In that case, the plotted lanes might not be correct.

3. Suggest possible improvements to your pipeline

- A possible improvement would be to use some other advanced techniques to mask(dynamic shape) the unnessary lines.
- To find the curved lanes, it might need a 'curve detector' instead of simple line detector.
- Proper care needs to be taken to make sure, the input video is stable and all the image-frames are processed w.r.t the same reference frame.