Finding Lane Lines on the Road

By Brendan Kam

The goal of this project is to detect lane lines on the road for the provided images and videos using Python and OpenCV. To do so, a pipeline was design to detect the lines and built using the functions introduced in the class.

First the relevant functions were imported into the code. Then the first image was loaded and converted into grayscale using the OpenCV function cvtColor. Figure 1 shows a grayscale conversion of an image provided.

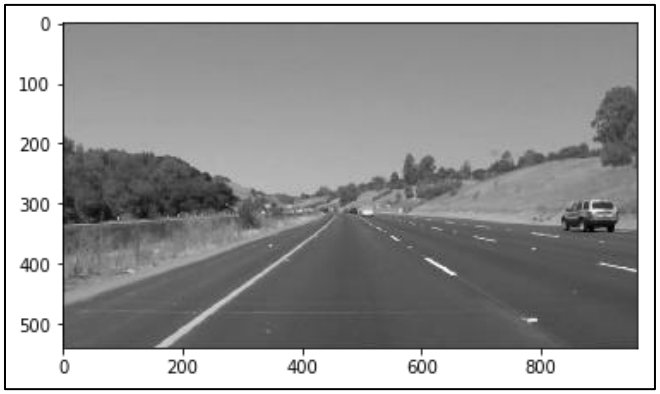

Figure 1: Grayscale conversion of image using OpenCV cvtColor function

Next, the Canny edge function from OpenCV was used to determine the edges in the images. The Canny edge detection method uses a change in gradient of the colors in the image to determine different edges. By adjusting the lower and upper thresholds for the Canny function and the Kernel size for the Gaussian smoothing, the edges of figure 1 was found, which is shown in figure 2.

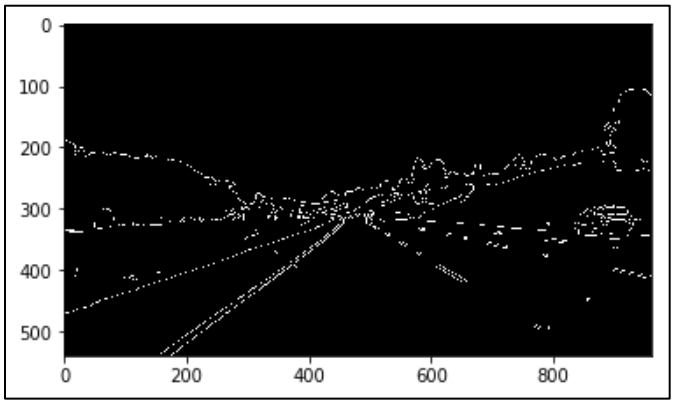

Figure 2: Edges found using Canny Edge Function

The next step in the pipeline was to use the fillPoly function from OpenCV to create a four-sided polygon that only shows the region of interest in the image. After this is done, the Hough Transform function in OpenCV was used to obtain the lines of the edges in the image. The lines obtained from the Hough Transform was then separated into two arrays which represents the lines on the left and right sides of the road respectively.

From the array, the different points of the lines and the gradient of the line was averaged. From the averaged points and gradient obtained, straight lines were drawn. Finally, the addWeighted function in OpenCV was used to overlay the lines drawn with the original image as shown in figure 3 below.

Figure 3: Detected lines on road using pipeline described

The code was then added to a function, to allow the pipeline described above to detect line on the roads in videos. The videos with the detected lines are included in the zip file submitted.

Reflections

A short coming of the pipeline described above is that it uses an average gradient of the lines found in the Hough Transform to extrapolate lines on the road. The pipeline would be able to extrapolate the lines on the road accurately if the car is driving on a straight road. However, if the car is driving a slightly curved road, the extrapolated lines would be inaccurate. A way to improve the pipeline is to use linear regression to draw a line of best fit for the points found in the images or video.

Besides that, this pipeline was only tested on images and videos that is on an open road without any obstacles or objects within its filtered region. If a car or any object was within that region, the Hough transform would return a line which the pipeline would draw as a line on the road. A way

to prevent this might be to use a color filter to filter every object that has a different color from the lines on the road. If the lines on the roads have a different color on the right and left sides, the line on one side can be detected and drawn first followed by the line on the other side and then be blended together on the original image.

As of now, the pipeline described in this report is unable to detect the lines in the "challenge.mp4" video because of the reasons described above. If the modifications suggested were made and a different polygon filter used, the pipeline might be able to detect the lines in the challenge video.

References

The pipeline created is based on the code template provided in "**P1.ipynb**" from CarND-Lane-Lines-P1 GitHub repository. The link to the repository is included below:

https://github.com/udacity/CarND-LaneLines-P1/blob/master/P1.ipynb

Also, this report was written based on the class materials, project rubric and writeup template provided. All of which can be found on the GitHub repository.