Finding Lane Lines on the Road

Mohit Pandey

2018
January

Chapter 1

Introduction

Finding Lane Lines on the Road

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

Chapter 2

Writeup/Readme

You're reading it.

Chapter 3

Reflection

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

3.1.1 Pipeline

The pipeline to detect lane lines is divided into following modules:

- Reading in the road image for which lines have to be detected. When working with a video, this means reading each frame at a time and then processing it with the steps underneath for pipeline.
- Following reading the image, I convert them to grayscale.
- Thereafter I run them through a Gaussian Blur filter. This smoothens the image and removes some noise. I use a filter size of 5.
- This blurred image then goes through a canny edge filter to get edges.
- After finding the edges, we get the region of interest around the car where we think the lane line may be. Assuming that camera is always mounted at the center on the top of the image, it's safe to assume a triangular region of roughly half the size of the image as our region of interest.
- This roi seperated image is then passed through a method to get Hough transform and we get lines in Hough space.
- We then draw these lines over the image we read in step 1 using *weighted_img()*. In next experiment we improve the drawlines method. This will be described in next bit.

3.1.2 Improvement to draw_lines()

From the output of the Hough transform, we look at the co-ordinates of lines to calculate their slope using the formula
m=$\frac{y_2-y_1}{x_2-x_1}$
If the slope is negative, we assign those set of points and hence the line as left line and for positive slope, we assign it to right points.

3.2 Identify potential shortcomings with your current pipeline

- The current pipeline will fail miserably if we change the position of the camera. Since all the parameters are hard-coded and their is no real science in their finding other then bunch of experiments, this pipeline isn't very powerful.
- This pipeline is also certain to perform poorly in case of varied lighting conditions. Since we worked in grayscale which doesn't account for varied lighting, shadows etc, such scenarios will break the pipeline.
- The ROI is hard-coded and isn't the most optimal across different vehicles.
- Finding optimal parameters for Gaussian filter, Hough transform is very challenging.

3.3 Suggest possible improvements to your pipeline

- Work with other color spaces like HSV, YCrCb which account for changing light intensities.
- Train a CNN model for more accurate detection of lane lines.