Contents

1 Finding Lane Lines on the Road

1.1 Overview

When we drive, we use our eyes to decide where to go. The lines on the road that show us where the lanes are act as our constant reference for where to steer the vehicle. Naturally, one of the first things we would like to do in developing a self-driving car is to automatically detect lane lines using an algorithm.

In this project you will detect lane lines in images using Python and OpenCV. OpenCV means "Open-Source Computer Vision", which is a package that has many useful tools for analyzing images.

1.2 Prerequistes

To run this Notebook use docker

docker pull udacity/carnd-term1-starter-kit

add this line to your .bashrc file

alias car="docker run -it –rm –entrypoint /run.sh -p 8888:8888 -v ~/Documents/CarND:/src udacity/carnd-term1-starter-kit"

After that, you can run the command `car` to start the docker image and the notebook. Just use the link in the terminal.

1.3 Overview

1. Pipelines.

In this project we used two different pipelines, One using just Color Threshold and Another using Canny Edge detection and Hough Lines.

- Color Threshold Pipeline
  - **HLS Convert** To make the color threshold simpler, was used the HLS images.
  - **Color Threshold** Using a White and Yellow threshold to create a mask from the image.
  - **Polyfit** Using the points from the mask it's possible to fit a line.
  - **Weighted image** : To merge the lines in the original image
- Canny Edge and Hough Lines Pipeline
  - **Gray Image** To use the Canny edge detection the image need to be gray
  - **Blur Image** To make the Canny edge less sensible to noise.
    - **Gaussian Blur**: We should specify the width and height of the kernel which should be positive and odd.
    - **Bilateral Filtering**: Filters tend to blur edges. This is not the case for the bilateral filter.

- **Region of interest**:
- **Canny Edge**: Use the Canny algorithm to detect the edges in the image.
- **Hough Lines**: Use the Hough transformation to find the lines in the image.
- **Fit Lines**: To make the lines more clearer was need to extrapolate.
- **Weighted image**: To merge the lines in the original image

The color threshold pipelines was more simple to code and has a nice result. Canny Edge Pipeline is more robust and works fine.

In the future, I'll mix both pipelines and give "some memory" to the algorithm. This way the use in video can be better.