remichartier / 005_SelfDrivingCarFindingLaneLinesProject

Code Issues Pull requests Actions Projects Wiki Security

master

005_SelfDrivingCarFindingLaneLinesProject / README.md

remichartier Update README.md **History**

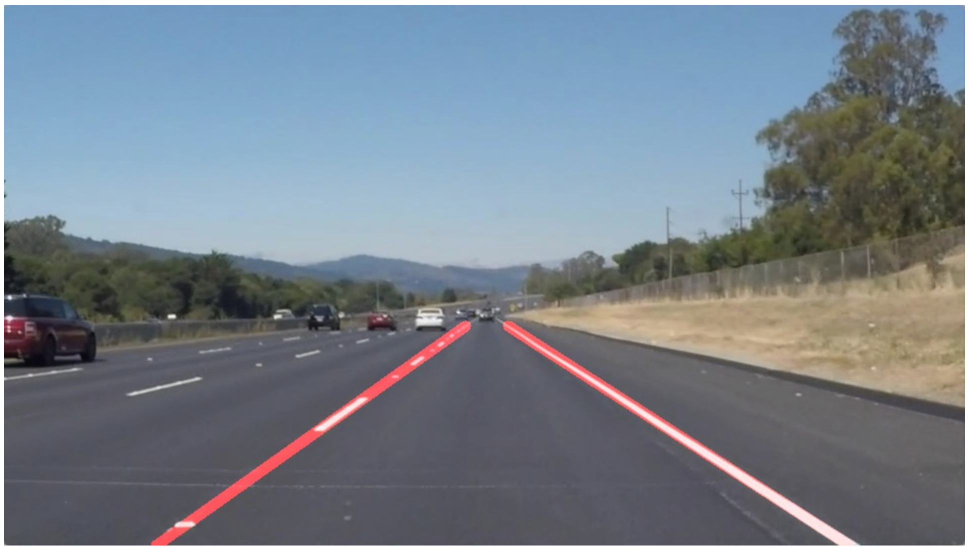
11 contributors

Raw Blame

54 lines (38 sloc) 3.14 KB

Finding Lane Lines on the Road

Udacity CarND



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.

Specifications from Udacity for this project to meet : project rubric

To complete the project, following files/folders are modified and submitted :

- P1.ipynb : Jupyter Notebook file containing project code.
- P1.html : equivalent in HTML.
- P1.pdf : Jupyter Notebook file printed in .pdf.
- writeup.md : containing a brief write up explaining your solution.
- writeup.pdf : equivalent file printed in .pdf.
- readme.md : This current readme.md file.
- readme.pdf : equivalent file printed in .pdf.
- test_images_output : folder with output images + subfolder for each step of the pipeline.
- test_videos_output : folder with output videos.

Note : input images and videos are in folders :

- test_images
- test_videos

Project / Implementation content

- Using the tools you learned about in the lesson to identify lane lines on the road.
- Develop your pipeline on a series of individual images, and later apply the result to a video stream. Check out the video clip "raw-lines-example.mp4" (also contained in this repository) to see what the output should look like after using the helper functions below.
- Once result looks roughly like "raw-lines-example.mp4", get creative and try to average and/or extrapolate the line segments you've detected to map out the full extent of the lane lines.

- Example of the result you're going for in the video "P1_example.mp4".
- Ultimately, Draw just one line for the left side of the lane, and one for the right.
- Brief writeup to complete, in markdown file or a pdf document.
- Tools provided : color selection, region of interest selection, grayscaling, Gaussian smoothing, Canny Edge Detection and Hough Tranform line detection.
- Your goal is piece together a pipeline to detect the line segments in the image, then average/extrapolate them and draw them onto the image for display. Once you have a working pipeline, try it out on the video stream below.
- Ideas for Lane Detection Pipeline Some OpenCV functions (beyond those introduced in the lesson) that might be useful for this project are:
    - cv2.inRange() for color selection
    - cv2.fillPoly() for regions selection
    - cv2.line() to draw lines on an image given endpoints
    - cv2.addWeighted() to coadd / overlay two images cv2.cvtColor() to grayscale or change color cv2.imwrite() to output images to file
    - cv2.bitwise_and() to apply a mask to an image