Vijay D., Feb 2017 Class
Udacity Car ND

Report P1: Basic Highway Lane Detection

Table of Contents

1. Sequence of Steps to Lane Detection

Note on Code Arrangement: To save time during testing, this process is menu-driven which has been retained in the final code version — helps with loading different images and videos without changing code. The menu appears similar to this:

```
Project: P1, Vijay D.
Basic Lane Detection

PICK A VIDEO: -OR-
___________________________________________
1:  Video w/solid white right lane marker
2:  Video w/solid yellow left lane marker
3:  Video - Challenge (with 2 shadow zones)

AN IMAGE:
___________________________________________
4:  White right lane marker
5:  Yellow left lane marker
6:  White curve marker
7:  Yellow left curve lane marker
8:  Yellow left curve lane marker 2
9:  Challenge image with shadow
10: White car lane switch

Your Choice (Enter:exit): ___
```

Notes:
Run the Python program file named: lanedetection.py on the command line like so: python lanedetection.py
Set the 'debug' flag to 1 inside code to output interim text results and graphically plot Hough lines themselves.
Once the script is running, pressing Ctrl+C ends the video / image being processed.

The sequence used to arrive at an acceptable lane detection, given images (and videos) of highway lanes, is the following:

1. GRAYSCALE: Convert the imported image to grayscale to reduce complexity
2. BLUR: Blur the image to reduce complexity a little more
3. MAKE MASK: Pick the range of white values of the lane markers in the gray-blurred image, and convert to a mask
4. CANNY EDGE DETECTION: Send this ANDed image through the Canny edge detection algorithm
5. MARK ROI: Establish the region of interest in which we want to identify the edges
6. BITWISE AND: Perform a bitwise AND to separate the lane markers (and unfortunately, anything else that might be in the same range of white values).

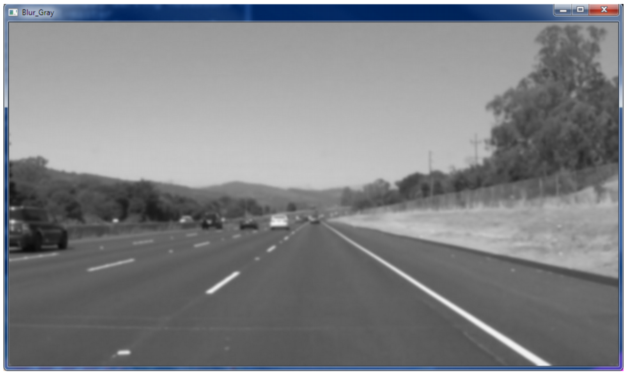
7. HOUGH LINE EXTRACTION:
Send the edge image through the Hough line extraction routine

8. DISCARD OUTLYING LINES:
Discard lines that do not belong to the lanes based on slope

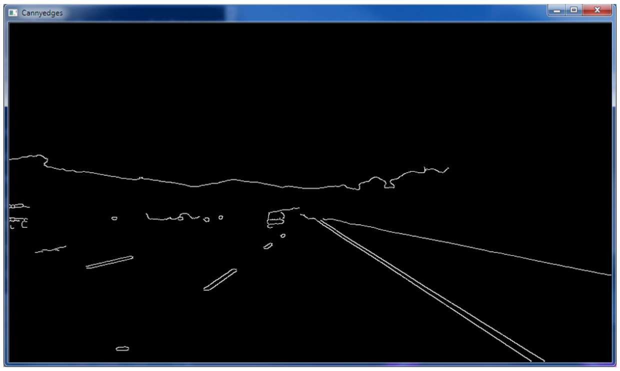
9. EXTEND LINE: Using slope and one of the lane lines, extend this line to the top of the ROI and the max. height of it (in +Y). Do for both sides of the lane.

10. SUPERIMPOSE EXTENDED LINE ON ORIGINAL IMAGE:
Draw the two lane lines on top of the original image using addWeighted() function.

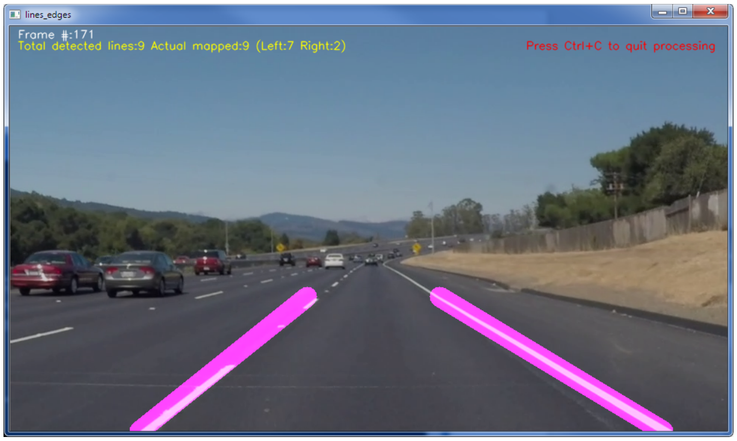
2. Example of lane detection on an image (solidWhiteRight.jpg)



After Step 2



After Step 4



```
Frame #:171
Total detected lines:9 Actual mapped:9 (Left:7 Right:2)
Press Ctrl+C to quit processing
```

After Step 10

3A. Potential Shortcomings With This Method

- A blurry grayscale may not be the best approach to detecting lane markers (an alternative that works to an extent is to convert to a negative image first).
- There are a few parameters in the Canny edge detection and Hough line extraction routines that offer too wide a spectrum to tweak manually. Even after manual tweaking, it may not be general enough to apply to every image.
- Finding and filtering lines (based on slope) through Hough extraction may not be good enough when there are hard turns, T-intersections, etc.

3B. Potential Improvements

- There must be better and more reliable methods for detecting lanes of any color.
- While the OpenCV inRange() function seems to hold promise, it also brings its own shortcomings when filtering out colors.
- Marking the ROI and extending lane lines on either side can be made fully parametric based on image dimensions (it is partially parametric now).