

Vijay D., Feb 2017 Class  
Udacity Car ND

# Report P1: Basic Highway Lane Detection

## *Table of Contents*

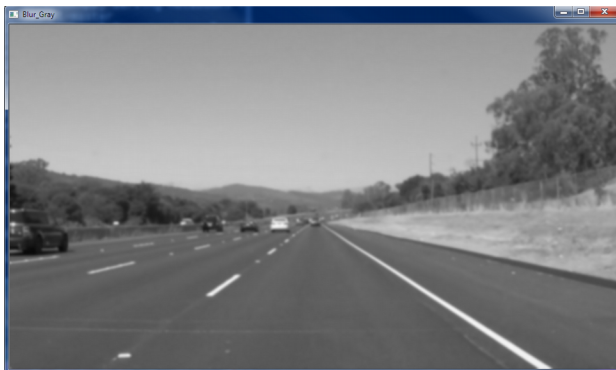
1. Sequence of steps to lane detection	2
2. Example images	2
3. Potential Shortcomings, Improvements	3
<i>Short Notes</i>	4

## 1. Sequence of Steps to Lane Detection

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.

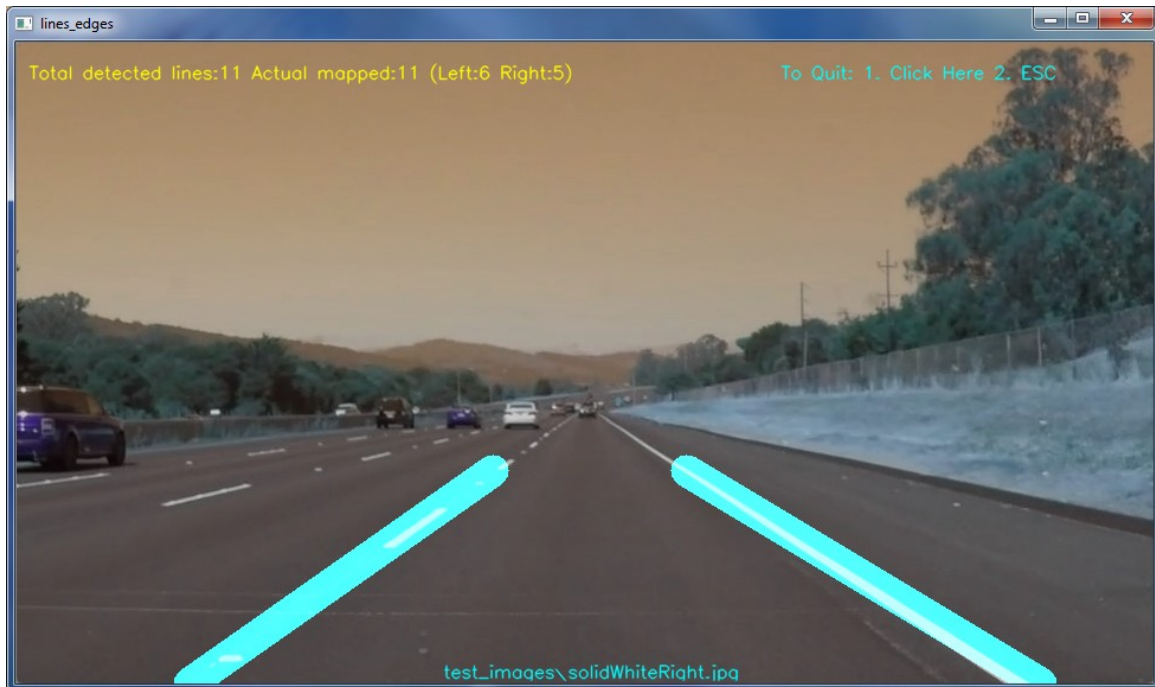
## 2. Example of lane detection on an image (solidWhiteRight.jpg)



**After Step 2**



**After Step 4**



**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 will require exploratory work 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 adequate in the presence of 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.
- While my script is partially successful in processing the challenge video, converting the images to the HSV color space might offer better filtering and performance.
- Marking the ROI and extending lane lines on either side can be made fully parametric based on image dimensions (it is partially parametric now).

## Short Notes

a. To save time during testing, this process is menu-driven which has been retained in the final script version – helps with loading different images and videos without changing code. Script, when executed, displays a menu similar to that shown below (type 1-10, or press Enter to exit):

```
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): ____
```

- b. Run the program with '-d 1' to turn 'debug' flag ON to output interim text & graphics results.
- c. To quit after processing begins, click inside the graphics window and press ESC to quit