Finding Lane Lines on the Road

Submitted by: Sankar Bhavan

1 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

2 Reflection

The pipeline consists of 6 simple steps.

First we start with RGB input image.

1. Convert the image to a gray scale image.

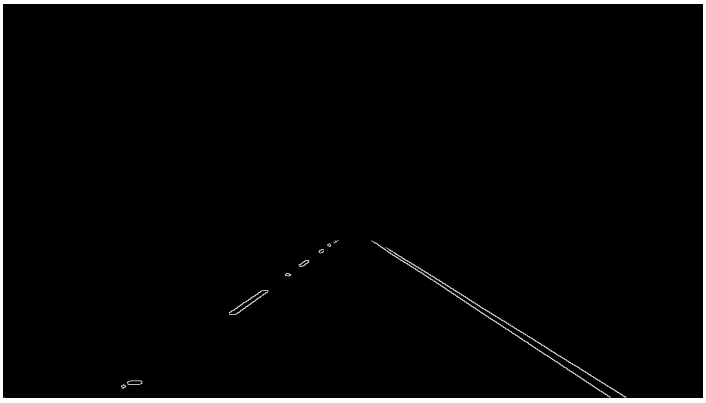
   This is done in-order for the image to be used for later transform functions.

2. Blur the image
   This is done to sooth out unwanted edges that would otherwise be detected by a Canny algorithm.

3. Run the Canny transform function over the blurred image
   This would just highlight sharp variations in colour. Thus highlighting the lanes and may be few other surrounding things.

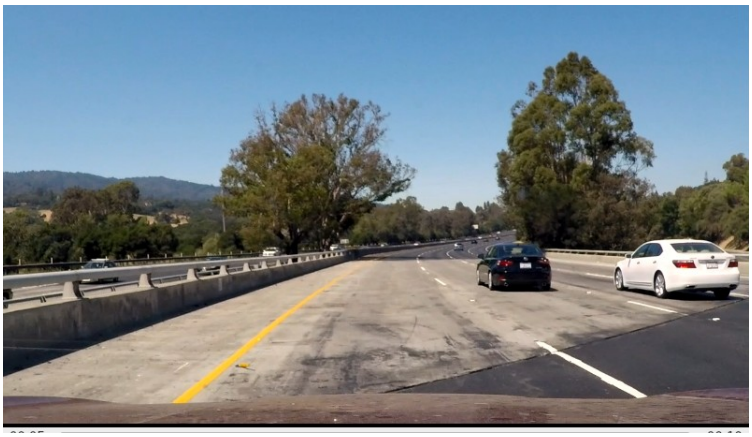
4. Now a mask of area of interest is applied over the the resultant image, to black out everything else but the portion of the road that is off interest.

5. Then Hough's transform is run over the image to identify all the lines in the image, that is big enough to be considered a lane line.



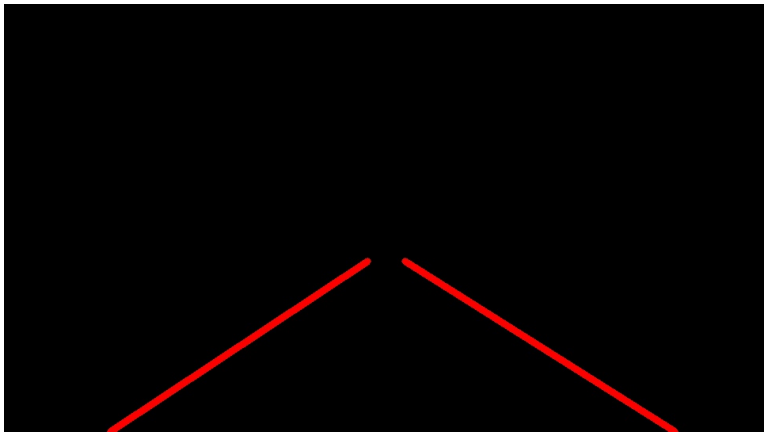
6. Now an **updated draw function** is used to complete the gaps in the lane line, and make it a full line. This is how it is done.
   a) First lines with negative slope and positive slope are packed into two separate arrays. Since both lanes progress in different directions, this is a nice way to separate left lane from right lane.

b) However, there could still be parallel lines, like the front of the car in the challenge sample video. Or even the start of the bridge that was of a separate colour. To avoid these line from interfering from the lane detection, lines with slope less than +/-0.3 is ignored.



c) The resulting image is also broken into two images by applying half of the original mask two times. First the left mask and then the right mask. Now each image should have one lane line only, either right or left.

d) cv2.fitLine API is used to draw a full line that fits each lane image (right and left) to a blank image of the same size



7. Add the image with just the lines with the original image and the lane detection is done. Here is the output image.

8. The same API is used for processing each image in a video.

3 Identify potential shortcomings with your current pipeline

1. The updated “convert to gray function” filters out yellow and white colours to detect the lane in low light conditions and also on the lighter road colour condition. This is still not working great in all conditions. I believe some kind of filtering or averaging function is need to avoid deflections from intended path in this case.
2. Discolorations on the road might also affect the lane detection now, again averaging might be the solutions here.

4 Suggest possible improvements to your pipeline

1. Pipeline include a lot of copies, it can be optimized to make it faster
2. Low visibility conditions, such as bad weather or water on the road might not work good.
3. The pipeline assumes there are no vehicles right in front.
4. Might not really work good for uphill or downhill.