**Report on Automatic Lane Detection in Images taken by a self driving car**

Author: Sundar Ram Swaminathan

**Problem Statement**

To detect the left and right lane markers closest to the car in the given images and to find out the x-intercept of the lines with the bottom of the image

**Approach**

For each image in the given folder, I performed the following operations in order

1. RGB to grayscale conversion

2. Removing the top half of the image in order to reduce search space, since we know that the road is at the bottom half of all images

3. Gaussian smoothing to remove noise

4. Canny Edge Detector to detect the images (thresholds were set by trial and error)

5. After detecting edges, I traced the contours of these edges. There are unwanted contours which are very small and so I use a Contour Area filter to remove those contours whose area is less than 50 pixels (trial and error)

6. The next step is figuring out the contours which are in the shape of a line. I used contour approximation to separate those contours which look like lines.

7. After separating these contours, I fit a straight line to each of these contours such that it extends to the whole width of the image

8. Next step was the most difficult part for me: to detect which line is the correct line that detects the left and lane markers out of the multiple lines in the image

For this, initially I find the slopes of each line. The slope of the left lane is negative and that of the right lane is positive. The slope of the lines on the farthest left are more negative and the slope increases as we move towards the right side of the image

I use a list to store the slope of each line as well as the coordinates of the endpoints. So I arrange all the slopes in ascending order and choose the two slopes that are nearest to the zero crossing. The one to the left of the zero crossing is the left lane marker and the one on the right side of the zero crossing is the right lane marker.

The last step is finding the x-intercepts of the lines with the bottom of the image. I find the line equation with the two points we have and then find the x-intercept by substituting the other values in this line equation.

I also tested for corner cases:

* If there are no lanes found, I increment the arc length multiplier in order to better approximate the contours as a line and redo the line fitting process again for each contour
* When there is just a single line detected: I try to figure out whether it’s a left lane marker or a right lane marker using the slope value and draw that particular line.
* If there are two or more lines detected near the left lane and none near the right lane, I draw only that lane which is nearest to the car (meaning the least negative slope value). And similarly, vice versa for the opposite side.
* When there are no lines to be found in the image, for now I say the x intercept is None, but since we know that the images are continuous, we can exploit this fact to safely assume that the x-intercepts of the left and right lanes would be more or less the same as the ones detected in the previous image.

**Further improvements possible**

* I tried using binary and Otsu thresholding which did not work out well. I would want to play around with the Adaptive Gaussian thresholding techniques and try to adjust the parameters so that I am more reliably able to detect the lanes on the road.
* Contrast adjustment: Since there are huge brightness variations, we can compensate for that by extracting the luminance channel of the image and equalize the histogram
* I also want to incorporate distance based thresholding for the lines detected, if there are two lines detected which are close, then keep only one of them. The sorting method using slope that I have implemented now was very effective but still having another distance based thresholding would always be helpful and make it more efficient.
* There were many disconnected edges of the lines, I would want to try to connect these edges using morphological operations or try combining sobel vertical edge detector with the Canny edge detector to more reliably get the vertical edges
* I would want to create a slider GUI where you can change parameters for different features such as edge detection thresholds – MinVal and MaxVal, Gaussian Blur kernel size, erosion kernel size, dilation kernel size, and we can figure out which is the most optimal point for accurately detecting the lane markers
* If still I am not able to find the lanes in an image, I would assume that the lanes are more or less near the same location as they were in the previous image.