Lane Line Finding

This work, we would like to introduce a deep learning based model to robustly identify the lane line instead of the original computer vision version to solve the problem for it's lack of robust and strongly based on linear solutions. First we would like to introduce a traditional pipeline to solve the lane line finding problem using computer vision and the weakness of the computer vision solution and how deep learning could have a better solution instead.

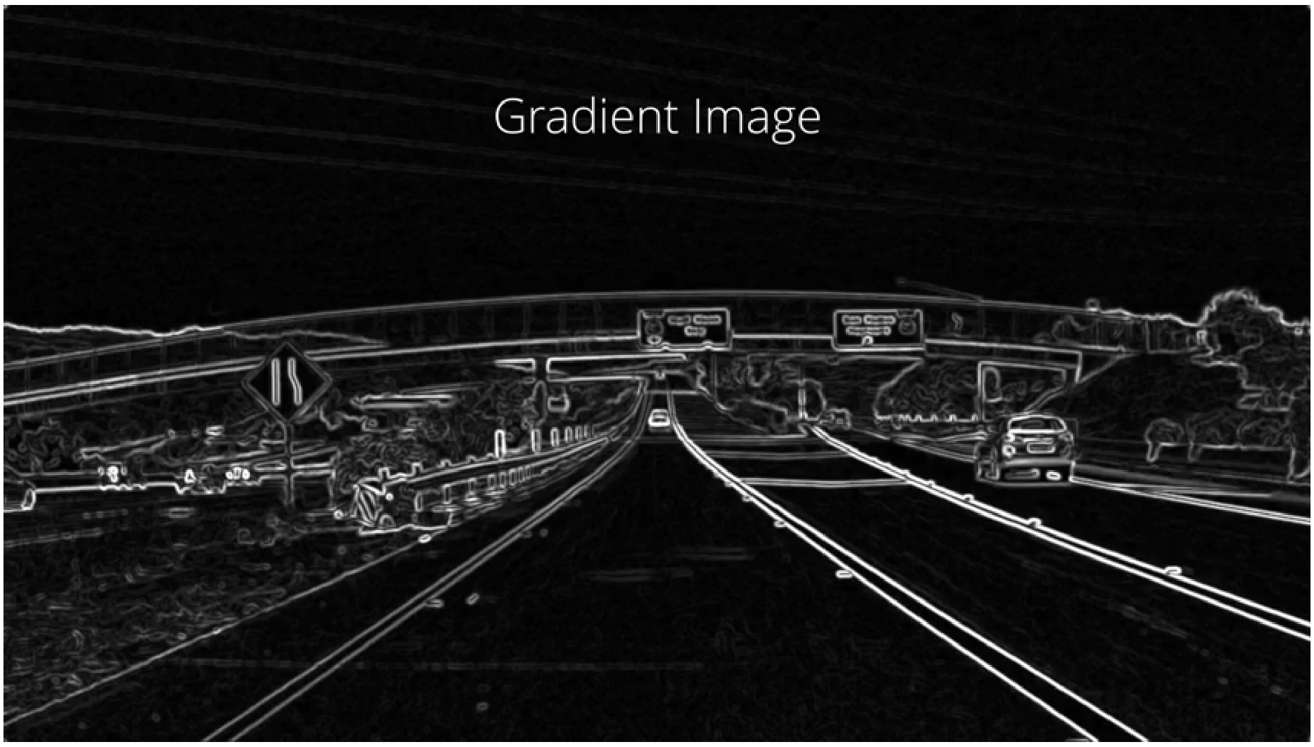
Computer Vision prospective

This section we introduce how to use pure computer vision(without deep learning and other techniques) to figured out how to identify the lane line. The whole process could easily illustrate as:

1. Turn the input into the grayscale.
2. Compute the gradient. To get **all edges**
3. Apply Gaussian blur filter to remove most unwanted edges.
4. Finding the line using Hough Transform.

Canny Edge Detection

Canny edge detection could help us to identify the edge of the object boundary.As for one image, we have got three color channel as input, however it's unnecessary for us to use 3 channel to process, so we would like to turn the image into gray-scale image using opencv tool `cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)` to solve our need. Grayscale image is just a starter, then we would like to compute the gradient.



Thankfully, OpenCV has provided three types of gradient filters or High-pass filters, Sobel, Scharr and Laplacian. We will see each one of them.

Sobel and Scharr Derivatives

Sobel operators is a joint Gausssian smoothing plus differentiation operation, so it is more resistant to noise. You can specify the direction of derivatives to be taken, vertical or horizontal (by the arguments, `yorder` and `xorder` respectively). You can also specify the size of kernel by the argument `ksize`. If `ksize = -1`, a 3x3 Scharr filter is used which gives better results than 3x3 Sobel filter. Please see the docs for kernels used.

Laplacian Derivatives

It calculates the Laplacian of the image given by the relation, $\Delta src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2}$ where each derivative is found using Sobel derivatives. If `ksize = 1`, then following kernel is used for filtering:

$$kernel = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Using Canny

Getting back to the main subject, by using `cv2.Canny(image, low_threshold, high_threshold)` we could perform Canny method easily. After input grayscale images, Canny, a filter like function select a range of gradient that we care about.

Hough Transform

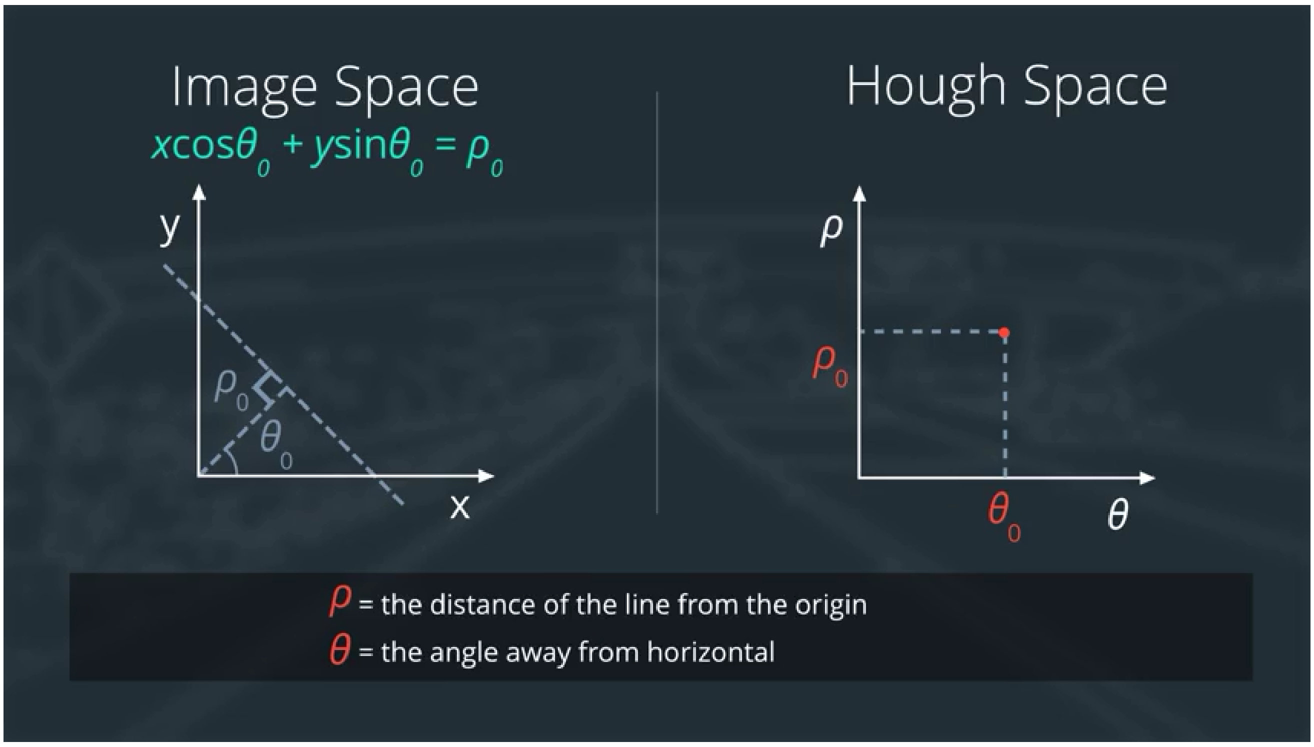


Image Space
$x\cos\theta_0 + y\sin\theta_0 = \rho_0$
y
x
ρ_0
θ_0
Hough Space
ρ
ρ_0
θ_0
θ
ρ = the distance of the line from the origin
θ = the angle away from horizontal

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform. The simplest case of Hough transform is detecting straight lines. In general, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, vertical lines pose a problem. They would give rise to unbounded values of the slope parameter m.

The final result of the linear Hough transform is a two-dimensional array (matrix) similar to the accumulator—one dimension of this matrix is the quantized angle θ and the other dimension is the quantized distance r. Each element of the matrix has a value equal to the sum of the points or pixels that are positioned on the line represented by quantized parameters (r, θ). So the element with the highest value indicates the straight line that is most represented in the input image.

Reference

1. Canny detector, OpenCV document.
2. Hough Transform, Wikipedia
3. Self-driving car, Udacity