* **Canny edge algorithm**

This algorithm works by checking for pixel information of the image. In our project, we are using this algorithm for following reasons:

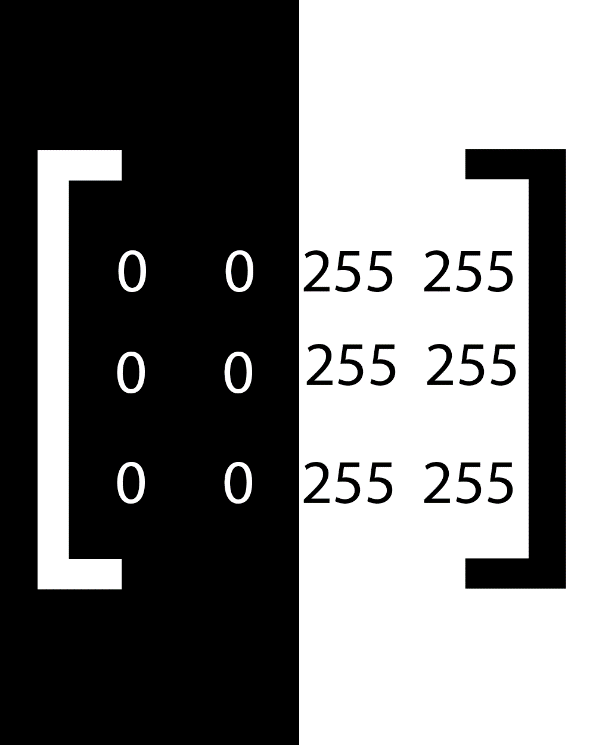
* Reduce noise and unwanted pixel information
* Less time and space complexity making it suitable for real-time image processing
* Can work on any type of image data

**Time and Space Complexity**

* Time Complexity: O(n)
* Space Complexity: O(n2)

Here, n represents the number of pixels in an image

Normally the pixel information is stored in matrix form as shown in the image below:

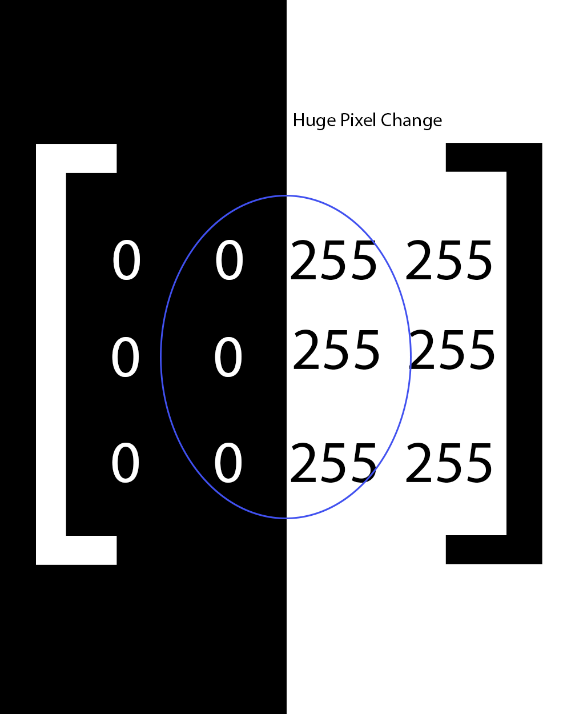
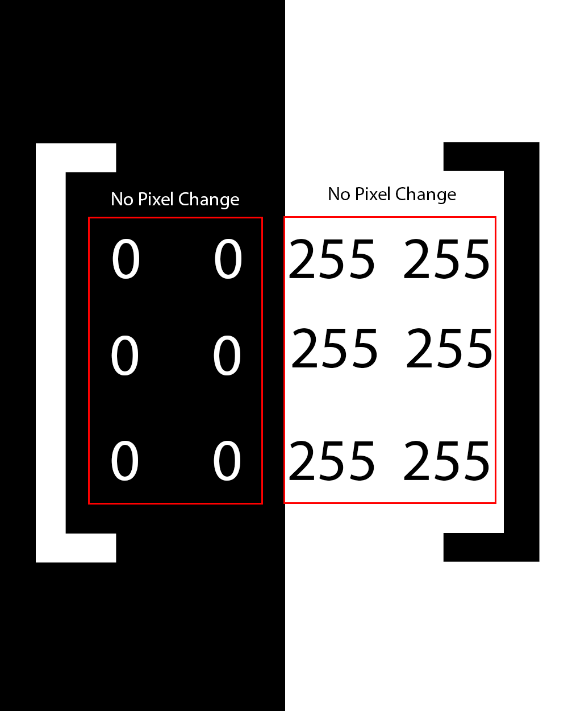


When such image is processed using the “Canny Edge Algorithm”, it checks for pixel change. Any section that has pixel change is considered edge. Pixel can be of two types:

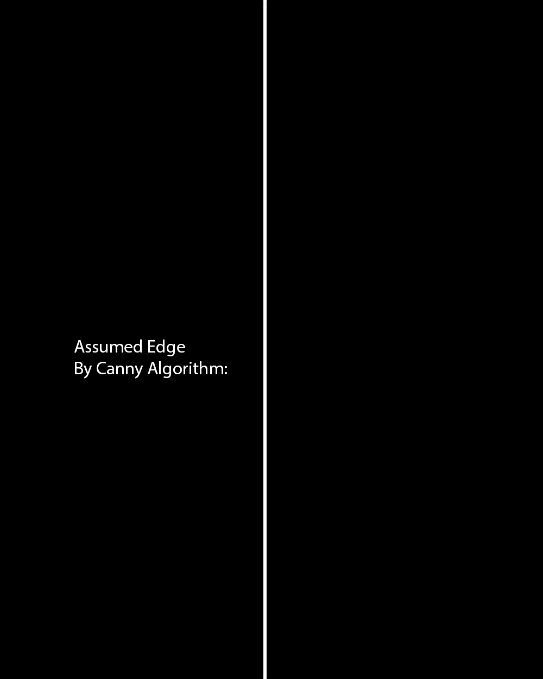
* Edged pixel
* Non edged pixel

The type of edge is derived using the amount of pixel change in the image. There are two arguments namely *Minimum Threshold**(T\_HIGH)*and*Maximum Threshold (T\_LOW)*. Anything above maximum threshold is considered to be stronger edged pixel and anything under minimum threshold is considered to be not edged. Anything between these threshold could either be stronger edge or smaller edge depending upon the neighbor of that specific pixel.

In above image, the first two pixels and last two pixels are same, meaning they do not have change in gradient. Hence, the pixels are not considered edge. However, the middle two pixels[(0,255),(0,255)] have huge gradient shift making it eligible for a strong edge.

****

**Canny edge detection algorithm will give output like this:**

****

**Working Of Canny Edge Algorithm**

Canny edge is a multi-step algorithm. There are mainly 4 steps involved in the pixel summation and edge identification.

**Step 1: Apply Gaussian Filter**

An image has multiple disturbance and noise. Noise, in this case, refers to all the un-necessary pixel information and background objects in the image. Applying gaussian filter to image helps in noise suppression and reduction. The Gaussian filter is a low-pass filter that removes high-frequency noise while preserving the overall shape of the edges in the image.

**Mathematical equation:**

where (x, y) is the pixel location in the image, and σ is the standard deviation of the Gaussian filter.

**Step 2: Compute gradient magnitude and orientation**

This step helps to detect the strength and direction of edges in the image. The gradient magnitude represents the strength of the edge, while the gradient orientation represents the direction of the edge.

**Mathematical equations:**

where D(x, y) is the intensity value of the pixel at location (x, y), and Gx and Gy are the gradient components in the x and y directions, respectively.

**Step 3: Perform non-maximum suppression**

This is step is applied to compare each pixel with its neighboring pixels. For each pixel (x, y), magnitude of its gradient is compared with two of the neighboring pixel in the gradient direction θ(x, y). If the magnitude of the pixel is not on maximum, we set it to 0 meaning that it will be displaying black pixel. If not, we set it to 255 which represents white. It helps reduce any unintended noise that was not reduced or was caused by previous step.

**Step 4: Apply hysteresis thresholding**

As said earlier, Canny Edge accepts two threshold values, T\_LOW and T\_HIGH also known as minimum threshold and maximum threshold respectively. This step deals assigning threshold and checking if pixel matches any of the following condition.

|  |  |
| --- | --- |
| **Condition** | **Assumption** |
| Pixel(x, y) > T\_HIGH | Pixel is a strong edge |
| Pixel(x, y) < T\_LOW | Pixel is a noise, disregard it |
| T\_HIGH > Pixel(x, y) > T\_LOW | If pixel connects to strong edge, it is a strong edge, else not a strong edge |

* **Hough Transform Algorithm**

Hough transform algorithm is line detection algorithm. We are using this algorithm in our project because of following reasons:

* Lane lines can be detection is effective
* Can work great when implemented side by side with Canny Edge Detection Algorithm

**Time and space complexity**

Time Complexity: O(N2 \*R)

Space complexity: O(R^2)

where N is the number of pixels in the input image and R is the resolution of the Hough space.

Here, Hough space is the parameters used to detect any shape in the image. In our case, the shape is a two dimensional line that can be represented using 2D array consisting of x1,y1,x2,y2. These values can be used to calculate slope of the line, which we will discuss later.

**Working of Hough Transform Algorithm**

Hough transform algorithm converts an image into parameterized space where each point in the parameter represents possible line in the image.

**Step 1: Edge Detection**

Edge detection is the first step in this algorithm. and it is not done through this algorithm but through use of other algorithms. In our case, we will be using Canny Edge Detection Algorithm.

**Step2: Hough space**

The Hough transform converts each edge point in the image into a set of lines in a parameter space known as the Hough space. The Hough space is defined by two parameter i.e. slope (m) and y-intercept (b) of the line.

**Step 3: Voting**

In this algorithm, voting is done by the pixel. Voting refers to number of intersection in a Hough Space. If the number of intersection is higher than the threshold set by the developer, the point is considered to be line.

**Step 4: Line Detection**

Once voting is completed, the points in the image with maximum number of intersection is considered to be a line. The line corresponds to straight line in the image.

Finally,

For each edge pixel (x, y) in the binary edge image, we compute the Hough space parameter space point (ρ, θ) as follows:

where ρ is the perpendicular distance from the origin (0,0) to the line passing through (x, y) at angle θ. θ is the angle between the x-axis and the line passing through (x, y).

We can represent the Hough space as a two-dimensional matrix H(ρ, θ), where each element H(ρ, θ) is the number of edge pixels that lie on the line defined by (ρ, θ). Each edge pixel (x, y) in the input image contributes to a sinusoidal curve in the Hough space, and the intersection of these curves corresponds to a line in the input image.

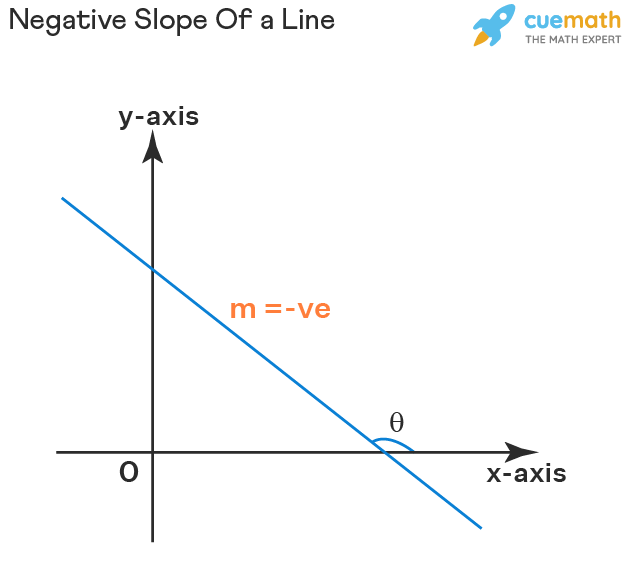
After accumulating all the edge pixels in the Hough space, we can search for local maxima in the Hough space to determine the parameters (ρ, θ) of the lines present in the image.

**Use of Hough transform algorithm in Brainly**

In our project, we need to know when the vehicle should be moving forward, right or left. For that, we need to know the shape of lanes. Hugh transform algorithm provides us with slope that we can use to check if the vehicle should move left, right or straight.

Conditions to be checked:

* Slope is negative (m<0)

When slope is negative, it means that the lane lines are moving from top left to bottom right. This indicates that the vehicle should move to left in order to be in the lane. 

* Slope is positive (m>0)

When slope is positive, it indicates that the lane line is heading from bottom left to top right, it indicates that the lane is curved toward right.

