Canny edge detection is an image processing method used to detect edges in an image while suppressing noise.

The main steps are as follows:

Step 1- gaussian blur

Step 2- determine the intensity gradients

Step 3- non maximum suppression

Step 4- double thresholding

Step 5- edge tracking by hysteresis

…………………………………………………………………………………………………………………………………………………………

The blur removes some of the noise before further processing the image.

I used Sigma 1.4.



An edge occurs when the color of an image changes, hence the intensity of the pixel changes as well.

The magnitude of the image results in the following output:



The image magnitude produced results in thick edges.

Ideally ,the final image should have thin edges. Thus, we must perform non maximum suppression to thin out edges.

Non-maximum suppression works by finding the pixel with the maximum value at an edge.

The result of this is :



We notice that the result from non-maximum suppression is not perfect, some edges may not actually be edges and there is some noise in the image.

Double thresholding takes care of this.

It sets two thresholds, a high and a low threshold. I normalized all the values such that they will only range from 0 to 1. Pixels with a high value are most likely to edge.

For example, you might choose the high threshold to be 0.9, this means that all pixels with a value larger than 0.9will be a strong edge. You might also choose a low threshold of 0.5, this means that all pixels less than it is not an edge and you would set it to 0.

The values in between 0.5 and 0.9would be weak edges, in other words, we do not know if these are actual edges or no edges at all.

This threshold is different per image so I had to vary the values. In my implementation, I found it helpful to choose a threshold ratio instead of a specific value and multiple that by the max pixel value in the image.

As for the low threshold, I chose a low threshold ratio and multiplied it by the high threshold value.

lo, hi = 0.1 \* mmax, 0.8 \* mmax



Now that we have determined what the strong edges and weak edges are, we need to determine which weak edges are actual edges. To do this, we perform an edge tracking algorithm. Weak edges that are connected to strong edges will be actual/real edges. Weak edges that are not connected to strong edges will be removed.

To speed up this process, my algorithm keeps track of the weak and strong edges that way I can recursively iterate through the strong edges and see if there are connected weak edges instead of having to iterate through every pixel in the image