Canny Edge Detector

Om Shri Prasath, EE17B113

Introduction

Canny Edge Detector is an edge detecting operator which uses a series of image-processing steps to extract the edges in the image. It was developed to address the following criterion for good edge detection :

- Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible.
- The edge point detected from the operator should accurately localize on the center of the edge.
- A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

The steps involved in the Canny Edge detector is as follows :

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

In this assignment, there are slight modifications to the above algorithm which makes the coding part easier (albeit with reduction in the edge detection accuracy). The changes to the above algorithm done in the assignment are as follows

- Instead of doing non-maximum supression using interpolation of pixels to find neighbours, we instead directly take the neighbour as the point nearest in the direction of the gradient.
- Instead of doing double thresholding and hysteresis tracking, we just apply single thresholding to find the edges.

Images on applying the Algorithm

Converting the Image to Grayscale

We read the image and convert the image to grayscale, which is a prerequisite for applying the algorithm. The output is as follows :

Figure 1: Original Image converted to grayscale

Gaussian Filtering

On the grayscale image, we apply Gaussian Filtering to remove the noise. In the first case, we use a Gaussian kernel with kernel size $= (5 \times 5)$ and $\sigma = 1.5$. We convolve the image with the Gaussian kernel to get the filtered image. The output is as follows :

Figure 2: Smoothening the Image using Gaussian Filter

Gradient Magnitude and Angle Extraction

On the filtered image, we are going to extract the magnitude and direction of the gradient from the image using the Sobel Filter. First we extract the gradients along the horizontal and vertical direction by convolving the image with the Sobel Filters. The output is as follows :

Figure 3: Gradients Along X and Y Direction

From the gradients, we extract the magnitude by taking the square root of sum of the squares of both gradients (and normalize it to 0-255 range), and the angle by taking the inverse tangent of the Y Gradient by the X Gradient (and adding 180 to bring the angles between the range 0-180 for easier application of next step). The output is as follows (using HSV type display for angle with the angle value denoting Hue and magnitude denoting the Value and Saturation set to 255)

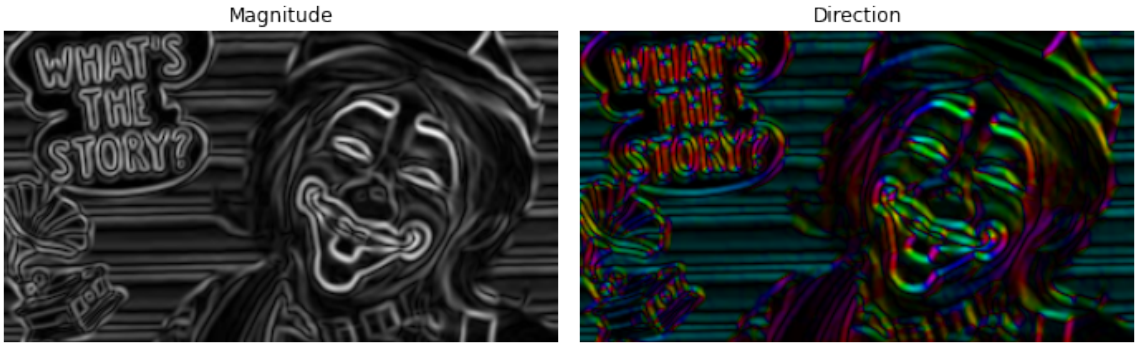


Figure 4: Magnitude and Direction

Non-Maximum Suppression

We thin the edges in the magnitude plot by removing the non-maximums in the direction of the gradients. This is done as follows :

1. We first find two neighbours of the pixel using the below classification :
   - Direction angle is between $0^o - 22.5^o$ or $157.5^o - 180^o$: **Left** and **Right** pixel are chosen as neighbours
   - Direction angle is between $22.5^o - 67.5^o$: **Bottom-Left** and **Top-Right** pixel are chosen as neighbours
   - Direction angle is between $67.5^o - 112.5^o$: **Bottom** and **Top** pixel are chosen as neighbours
   - Direction angle is between $112.5^o - 157.5^o$: **Bottom-Right** and **Top-Left** pixel are chosen as neighbours
2. If the pixel is maximum greater than the neighbours, we will set it as a maximum, otherwise we will remove it.

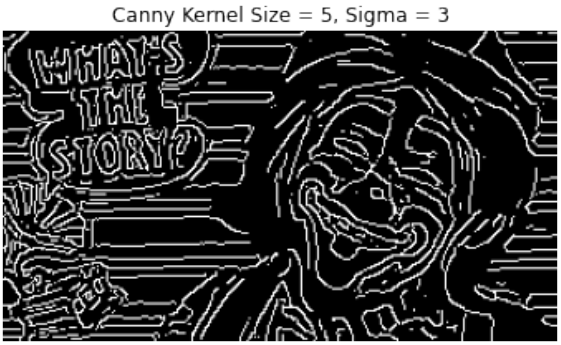
The output of the above algorithm is as follows :

Figure 5: Magnitude vs. Non-Maximum Supressed Output

Single Thresholding

We apply single thresholding on the output of non-maximum supression by setting the pixels with values less than the threshold (which we will keep as the **median** of all the magnitude values) to 0 and those above threshold to 255 to give clear edges as given below :

Thus we have got the Canny Edge output with kernel size = (5×5) and $\sigma = 1.5$ Now we try to apply the same algorith by changing only the σ to 3. The output is as below :



On comparing them, we can see that the output with $\sigma = 1.5$ has less defined features and more slightly bent lines than the one with $\sigma = 3$ due to the slight increase in blurring as a result of increasing σ.

Result

- We implemented a modified version of Canny Edge Detector to extract edges from a image
- On increasing σ the features becomes less defined and the lines become more bent.