Programming Assignment 1

Vijay Prakash Reddy Kovuru

CAP5415 Computer Vision

Oct 04, 2022
Instructor Dr. Yogesh Singh Rawat

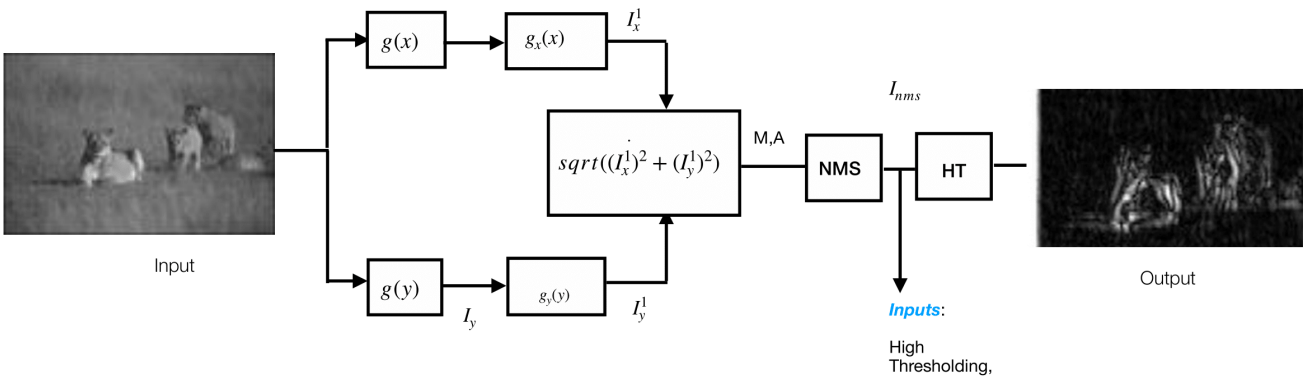
CANNY EDGE DETECTION IMPLEMENTATION

A. Abstract:

Implementing the Canny Edge Detection Algorithm without using any Python built-in functions for Gaussian filter, Convolution, Normalisation, Non-max Suppression, etc., Then Choosing three example grey-scale images from Berkeley Segmentation Dataset and plotting the intermediate and final results of Canny Edge Detection Algorithm. Showing the effect of Standard deviations (σ) in edge detection by choosing three different Standard deviations (σ) values when smoothing.

B. Canny Edge Detection Algorithm:

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. The Canny Edge Detection Algorithm is based on Grey scale Images.



Canny Edge Detection Algorithm

The Canny Edge Detection Algorithm is composed of 5 steps:

1. Reduce noise by smoothing the Image with Gaussian Filter

To remove noise, the image is smoothed by Gaussian blur with the kernel of size 3×1 and 1×3 and $\sigma = 1$. Since the sum of the elements in the Gaussian kernel equals 1 , the kernel should be normalised before the convolution.

2. Compute Derivative (Gradient) of Filtered Image

When the image I is smoothed, the derivatives Ix and Iy w.r.t. x and y are calculated. It can be implemented by convolving I with derivative of 1-D Gaussian kernels.

3. Non-maximum suppression (for Thinning the Edges)

For each pixel find two neighbours (in the positive and negative gradient directions, supposing that each neighbour occupies the angle of $\pi/4$, and 0 is the direction straight to the right). If the magnitude of the current pixel is greater than the magnitudes of the neighbours, nothing changes, otherwise, the magnitude of the current pixel is set to zero.

4. Double threshold

The gradient magnitudes are compared with two specified threshold values, the first one is less than the second. The gradients that are smaller than the low threshold value are suppressed; the gradients higher than the high threshold value are marked as strong ones and the corresponding pixels are included in the final edge map. All the rest gradients are marked as weak ones and pixels corresponding to these gradients are considered in the next step.

5. Edge tracking by Hysteresis Thresholding

Since a weak edge pixel caused from true edges will be connected to a strong edge pixel, pixel w with weak gradient is marked as edge and included in the final edge map if and only if it is involved in the same blob (connected component) as some pixel s with strong gradient. In other words, there should be a chain of neighbour weak pixels connecting w and s (the neighbours are 8 pixels around the considered one).

c. Results:

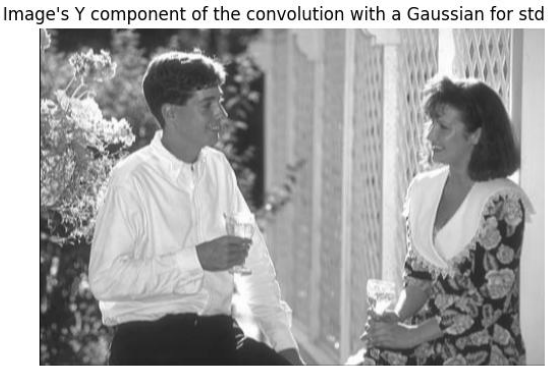
To evaluate the results, a grayscale image from Berkeley Segmentation Dataset (100 Test Images) [Grey][1-25][7] were used. The image were tested using three different Gaussian masks with three standard deviation values $\sigma \in \{1, 3, 5\}$ and zero mean $\mu = 0$.

Starting from top left, I is the input image to the algorithm. Ix and Iy are images blurred by 1D Gaussian filter in x and y directions, respectively. Ix $'$ and Iy $'$ are the results of applying the 1D Gaussian derivative on the the blurred images. The size of the Gaussian mask and Gaussian derivative is set to 3. Ix $'$ and Iy $'$ show the vertical and horizontal edges in the image, respectively. The magnitude of the Image combines the horizontal and vertical edges into one result. To thin out the edges on the output, NMS is applied. Finally, HT is applied to reduce some of the weak edges. The HT, requires weak and strong threshold values which can be tuned to produce the based result based on the image and the application. For the results, the thresholds are set to 20 and 75 pixels. The Hysteresis Thresholding does not have much of an effect on the image because the input image comprised pixels that were already black or white, which indicates that they are either more or less strong.

The results for σ = 1;

1.Input Image (I)

2. Ix



3. Iy

4. Ix'

Y component of the image with the derivative of a Gaussian for std = 1

5. Iy'

Resulting magnitude image for std = 1

6.Magnitude of Image

Canny Edge image after Non-maximum suppression for std = 1

7. After NMS

Image with final output for std = 1

8. After Hysteresis Thresholding

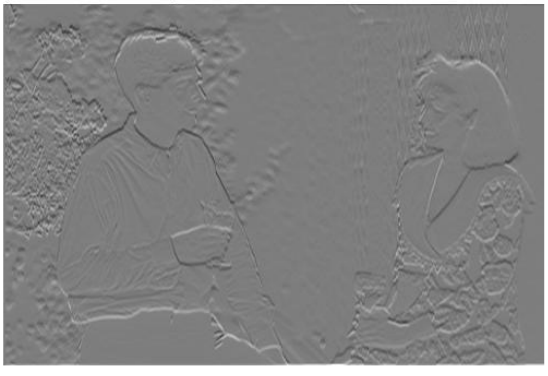
The results for σ = 3;

1.Input Image (I)

2.Ix

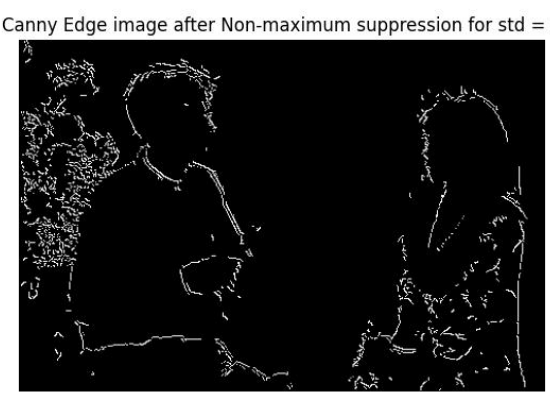
3. Iy

4. Ix'

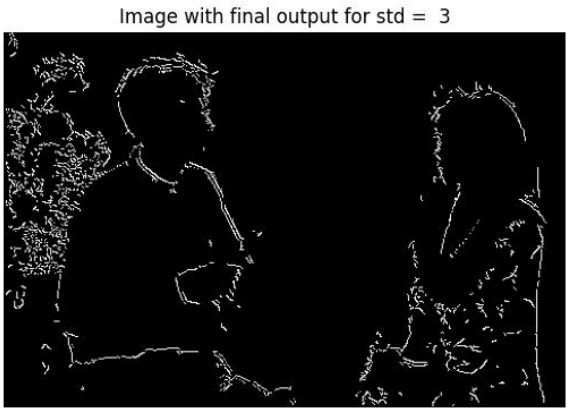


5. Iy'

5. Magnitude



7. After NMS



8. After Hysteresis Thresholding

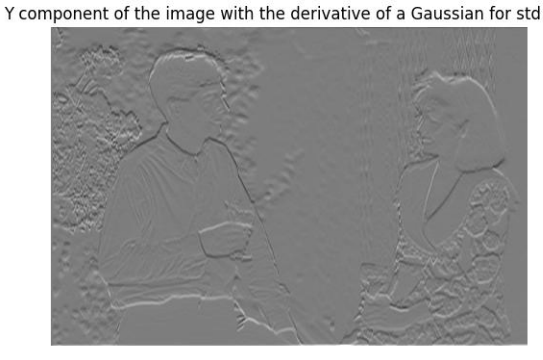
The results for σ = 5;

1. Input Image (I)

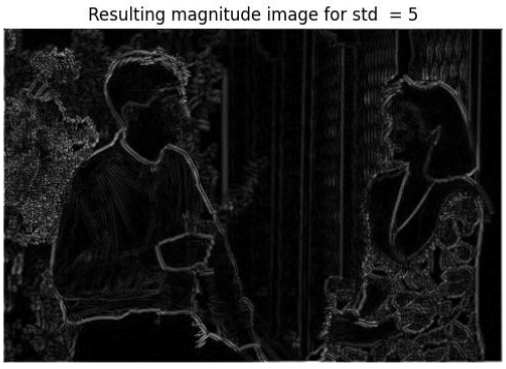
2. Ix

3. Iy

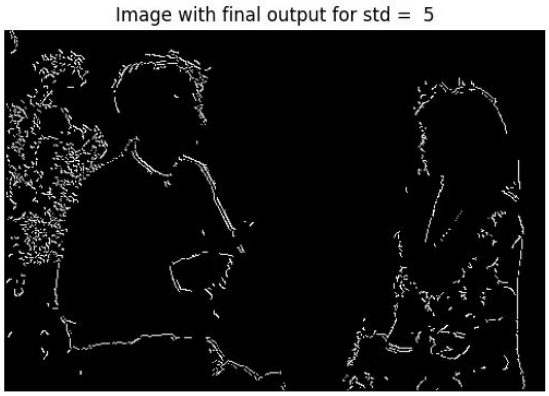
4. Ix'



5. Iy'



6. Magnitude



7. After NMS

8. After Hysteresis Thresholding

Conclusion:

- *Higher sigma values identify larger scale edges and produce a coarse output, whereas lower sigma values produces more edges and detect finer characteristics.*

- *For sigma value = 1 able to extract more edges for all the test Images (test_image1, test_image2, test_image3) and while increasing sigma value smoothening more the edges and missing few edges.*

[Github Code Link]

[Website link for Canny Edge Detection]

Change the theme of the application to Light mode for better view experience.