CAP5415 Programming Assignment #1 Report

Mayank Kumar

Step 1. Read in the image. Call it *I*.

Fig. 1 Input Image

Step 2. Create a 1-D gaussian mask. I have considered 0 mean, standard deviation = 1 and kernel size = 11

We get a mask with the following values:

$$G = [0.053,\ 0.241,\ 0.398,\ 0.241,\ 0.053]$$

It is the same for vertical and horizontal direction.

Step 3. Now we create the mask for derivative of gaussian. I use the same parameters as above.

We get :

$$G_x = [0.107,\ 0.241,\ 0,\ -0.241,\ -0.107] = G_y$$

Again, it is the same for both directions, except for its shape.

Step 4. We convolve I with G to get I_x, the blurred image in x-direction that is, $I_x = G * I$ and similarly for I_y. Only we traverse column-wise for I_y.

Fig. 2 I_x (Left) and I_y (Right)

We then convolve I_x with G_x to get I_x'. and similarly for I_y'.

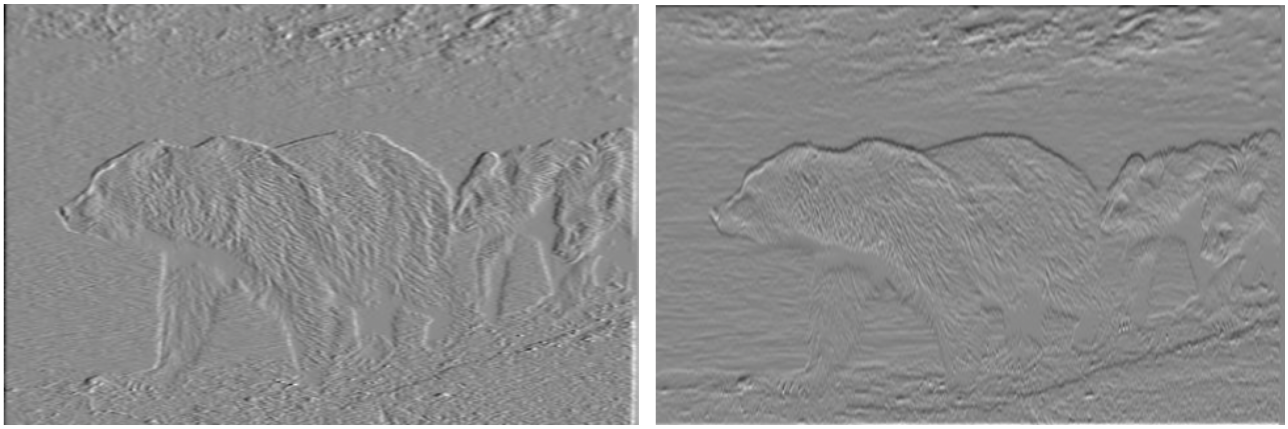


Fig. 2 I_x' (Left) and I_y' (Right)

Step 5. Now we calculate the Magnitude matrix of the edge response from the previous step using the following equation:

$$M(x,y) = \sqrt{I'_x(x,y)^2 + I'_y(x,y)^2}.$$

Using this, we get the following plot:

Fig. 4 Magnitude response

Step 6. Now we perform non-maximum suppression. But before that, we get the orientation of the edges for each pixel by the formula:

$$\theta(x,y) = \arctan\frac{I'_y(x,y)}{I'_x(x,y)}$$

Once we have the orientations, for each pixel, we will look at its neighbors along its orientation. For simplicity, I have divided the angular plane in 8 sectors of 45^o each.

So for example, if the orientation θ is between -67.5^o and -22.5^o or between 112.5^o and 157.5^o. , we consider the neighbor on top-left and bottom-right to compare with.

Once we get the intensities of the pixel neighbor, if either of them is bigger than our current pixel, we suppress the current pixel to 0.

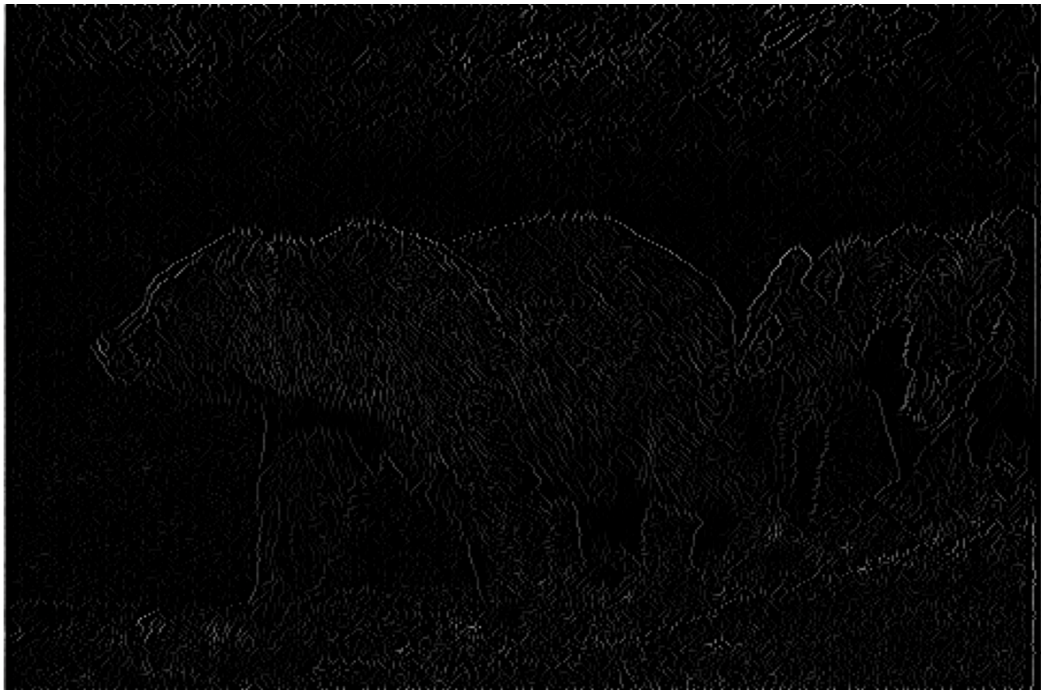


Fig. 5 After Non-maximum suppression

Step 7. Now, we perform Hysteresis Thresholding. For this I am considering the lower and upper thresholds as 0.1 and 0.2. All pixel values above the upper threshold will be made *strong* by setting them to 255. All pixels below the lower threshold will be set to 0. And all pixels between the thresholds are marked as *weak* pixels and set to value 105.

We then calculate the Connected Components for the double-thresholded using the OpenCV library. If any of the *weak* pixels is present in the same connected component of a *strong* pixel, we make them strong as well. This way, the edges are linked together.

We get the following plot:

Fig. 6 Final output of the Canny Edge detection algorithm

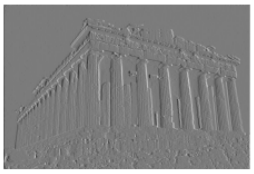
img = 67079.jpg sigma=0.5

(a) Original Image

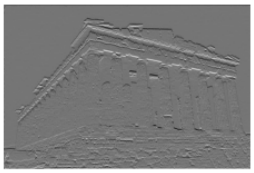
(b) Gaussian blurred in x

(c) Gaussian blurred in y

(d) Gaussian blurred derivative in x



(e) Gaussian blurred derivative in y



(f) Magnitude of gradient

(g) Nonmax Suppression

(h) Double Thresholding

(i) Hysterisis

img = 67079.jpg sigma=0.75

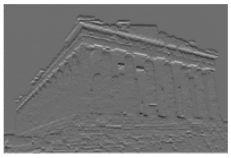
(a) Original Image

(b) Gaussian blurred in x

(c) Gaussian blurred in y

(d) Gaussian blurred derivative in x

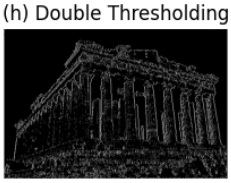
(e) Gaussian blurred derivative in y



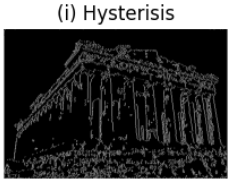
(f) Magnitude of gradient

(g) Nonmax Suppression

(h) Double Thresholding



(i) Hysterisis



img = 67079.jpg sigma=1

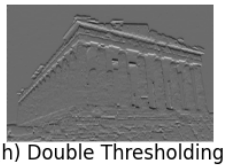
(a) Original Image

(b) Gaussian blurred in x

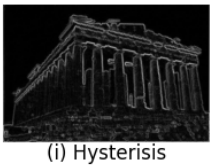
(c) Gaussian blurred in y

(d) Gaussian blurred derivative in x

(e) Gaussian blurred derivative in y



(f) Magnitude of gradient



(g) Nonmax Suppression

(h) Double Thresholding

(i) Hysterisis

img = 22090.jpg sigma=0.5

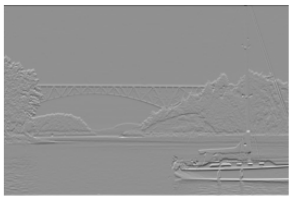
(a) Original Image

(b) Gaussian blurred in x

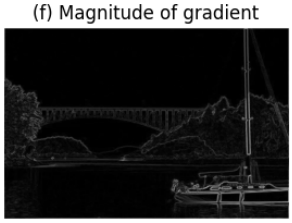
(c) Gaussian blurred in y

(d) Gaussian blurred derivative in x

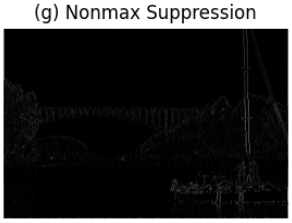
(e) Gaussian blurred derivative in y



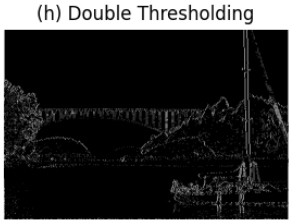
(f) Magnitude of gradient



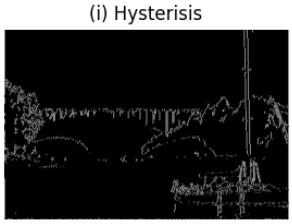
(g) Nonmax Suppression



(h) Double Thresholding

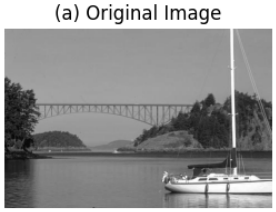


(i) Hysterisis



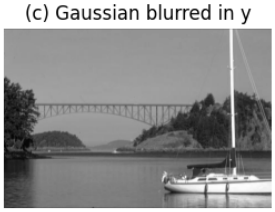
img = 22090.jpg sigma=0.75

(a) Original Image



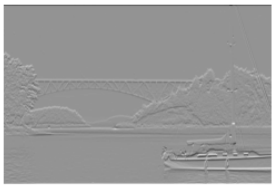
(b) Gaussian blurred in x

(c) Gaussian blurred in y

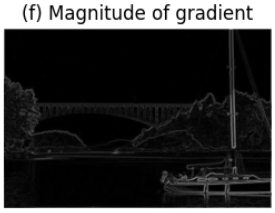


(d) Gaussian blurred derivative in x

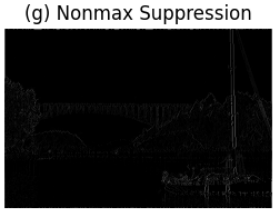
(e) Gaussian blurred derivative in y



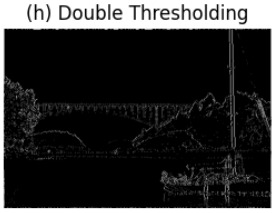
(f) Magnitude of gradient



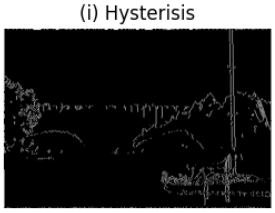
(g) Nonmax Suppression



(h) Double Thresholding



(i) Hysterisis



img = 22090.jpg sigma=1

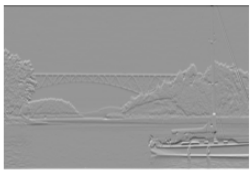
(a) Original Image

(b) Gaussian blurred in x

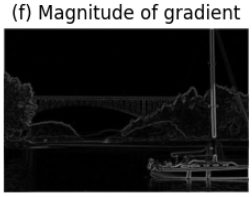
(c) Gaussian blurred in y

(d) Gaussian blurred derivative in x

(e) Gaussian blurred derivative in y

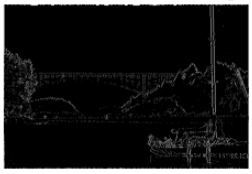


(f) Magnitude of gradient

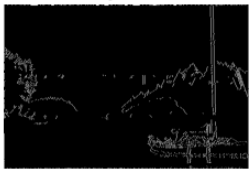


(g) Nonmax Suppression

(h) Double Thresholding



(i) Hysterisis



img = gray1.jpg sigma=0.5

(a) Original Image

(b) Gaussian blurred in x

(c) Gaussian blurred in y

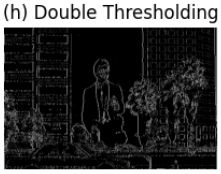
(d) Gaussian blurred derivative in x

(e) Gaussian blurred derivative in y

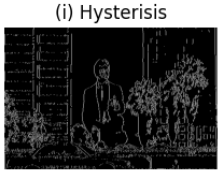
(f) Magnitude of gradient

(g) Nonmax Suppression

(h) Double Thresholding



(i) Hysterisis



img = gray1.jpg sigma=0.75

(a) Original Image

(b) Gaussian blurred in x

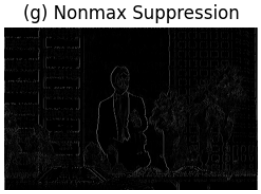
(c) Gaussian blurred in y

(d) Gaussian blurred derivative in x

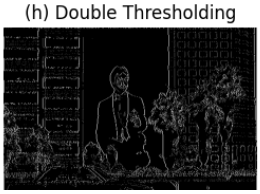
(e) Gaussian blurred derivative in y

(f) Magnitude of gradient

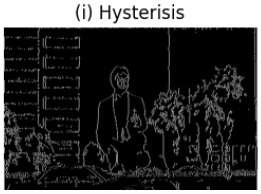
(g) Nonmax Suppression

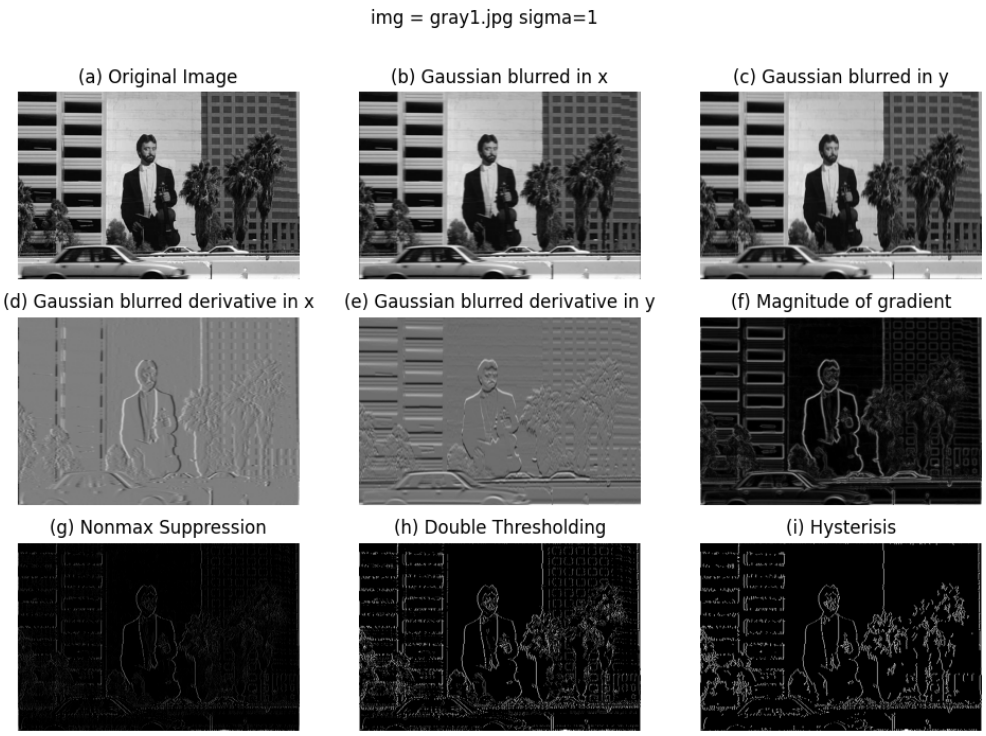


(h) Double Thresholding



(i) Hysterisis





My observation is that as we increase the value of σ, we lose the fine grain edge. Also, the localization of edges becomes less precise as sigma increases. It becomes more challenging to pinpoint the exact location of an edge pixel due to the blurring effect.

So in my opinion, sigma = 0.5 was the most optimal for most images.