**Comp 558 Assignment 1 Description**

**Yan Miao**
**260711311**

General description about functions and files:

make2DGaussian.m : (Question 1(a))
Implement a function "make2DGaussian(N,sigma)" that takes the size N of the square filter and the standard deviation "sigma" of the Gaussian and returns a 2D Gaussian matrix which can be used for filtering an image.

myConv2.m : (Question 1(b))
Implement a function "myConv2(image,filter)" that performs a 2D convolution, similar to the Matlab builtin function conv2.
It takes an image and a filter as inputs, and returns the convoluted image.

edge_detect_with_gradient.m : (Question 2 & 5)
Implement a function "edge_detect_with_gradient(image,sigma,input_threshold)" that performs edge detection on an image by first calculating the local differences (used as gradient) in both x and y directions, then calculating the gradient magnitude and using a threshold to plot edge pixels in a binary image.
It takes an image, a standard deviation "sigma" for Gaussian, and input_threshold as inputs. It returns a binary image which shows the edges of the input image.

edge_detect_with_MH.m : (Question 3)
Implement a function "edge_detect_with_MH(image,filt_size,sigma,threshold)" that performs edge detection on an image using Laplacian of Gaussian (LoG).
It takes an image, a filter size,a standard deviation "sigma" , and a threshold for Gaussian as inputs. It returns a binary image which shows the edges of the input image.

A1_MAIN.m : (Question 4 and test codes for functions)
It contains test code for questions 1~4 (code for question 5 is in "edge_detect_with_gradient.m"). It also creates an image for question 3 and contains the solution code to question 4.

# Question 1

a) In this part, we implement our own 2D Gaussian based on the following formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The Gaussian centers at (M+1,M+1), where M = (N-1)/2. N is the size of the Gaussian.

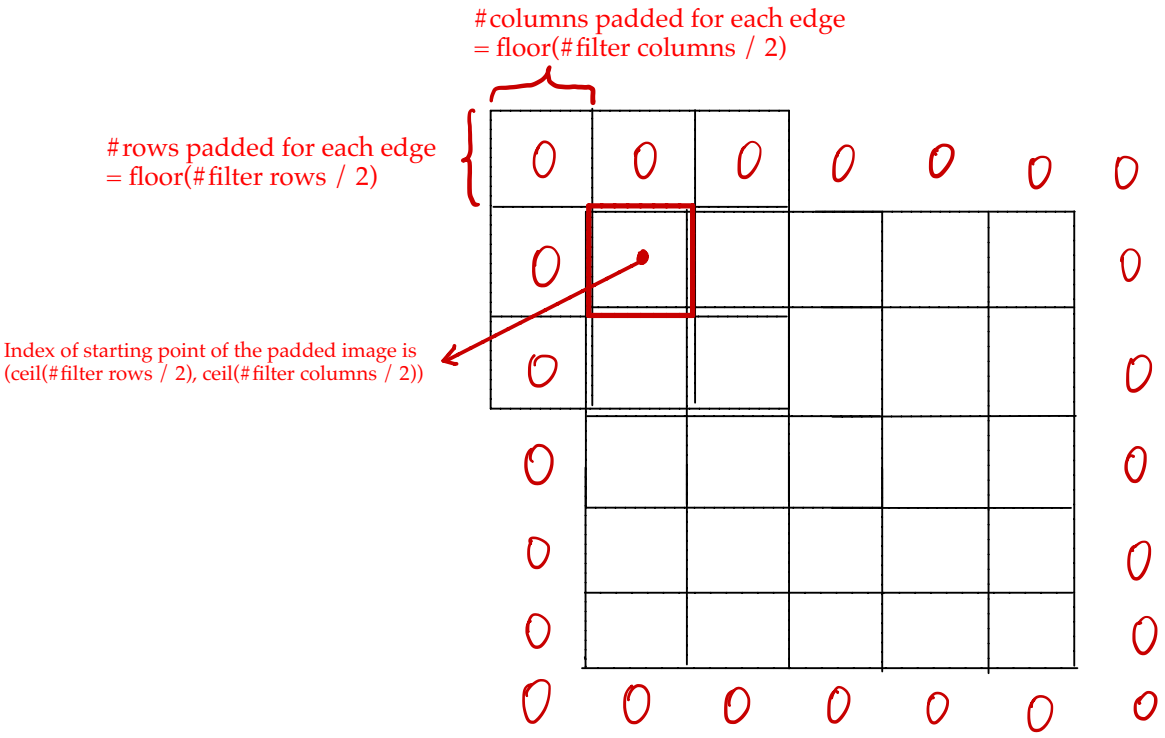
In this case, the x and y in the formula are treated as distance to the center of the Gaussian.

We also need to normalize the filter so that the sum of values in the filter becomes 1.

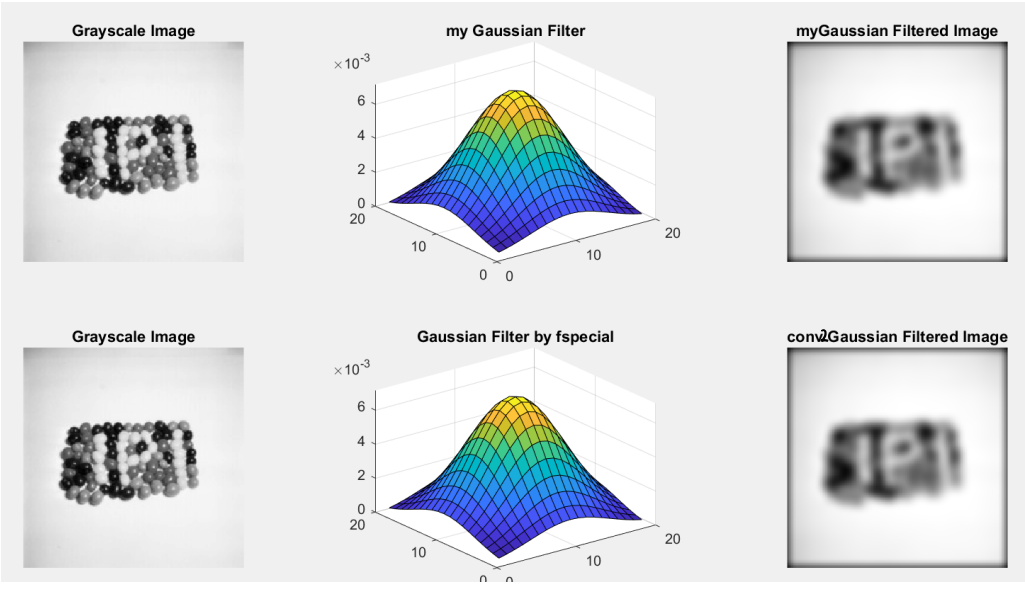
b) In this part, we implement our own 2D convolution.

The following is a visualization of my way of padding and performing convolution.

We use a 3x3 filter and a 5x5 image for demonstration.




The outcome shows as following:




This is a comparison between the filter created by my function and the filter created using fspecial.

# Question 2

In the part, we compute local differences in both x and y directions, and use them to compute the image gradient magnitude and orientation. Then we use the magnitude to perform edge detection. The code is based on the following formulas:

$$\nabla I(x,y) \equiv \left(\frac{\partial}{\partial x} I(x,y), \frac{\partial}{\partial y} I(x,y),\right)$$

$$\approx \left(\frac{1}{2} I(x+1,y) - \frac{1}{2} I(x-1,y), \quad \frac{1}{2} I(x,y+1) - \frac{1}{2} I(x,y-1)\right)$$

$$\| \nabla I(x,y) \| \equiv \sqrt{\left(\frac{\partial}{\partial x} I(x,y)\right)^2 + \left(\frac{\partial}{\partial y} I(x,y)\right)^2}$$

$$\approx \frac{1}{2} \sqrt{\begin{matrix} \big(I(x+1,y) - I(x-1,y)\big)^2 \\ + \big(I(x,y+1) - I(x,y-1)\big)^2 \end{matrix}}$$

$$\theta = \tan^{-1}\left(\frac{\frac{\partial I(x,y)}{\partial y}}{\frac{\partial I(x,y)}{\partial x}}\right)$$

Original image:


Green channel image:

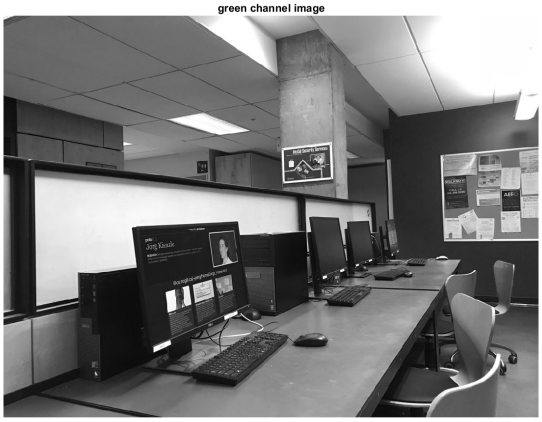
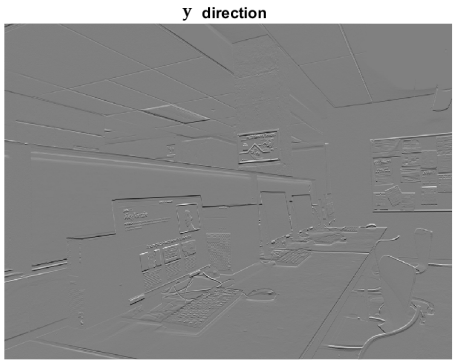
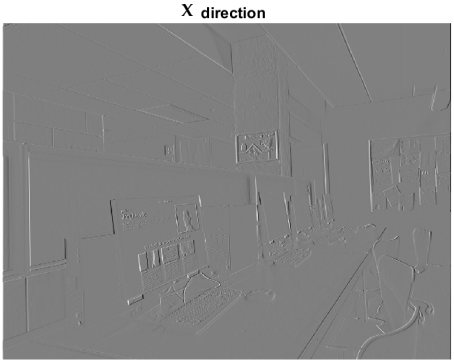



Image gradient in x and y directions:

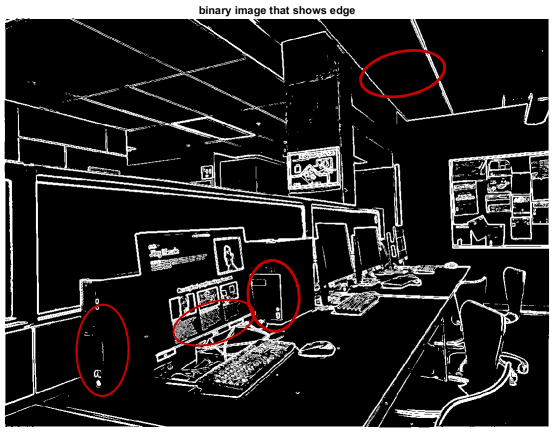


**—> Edge maps with different sigma and discuss the differences**

Edge image (sigma = 0.4, best value after tuning)


Edge image (sigma = 0.8)




Edge image (sigma = 1.6)


As we can see from the 3 images on the left, the top one with the sigma of 0.04 shows most details of the edges of the original image. When the sigma gets multiplied by 2 and 4, the edge image prints out fewer and fewer edges. The reason is that the greater the sigma is, the greater the original image is smoothed. Since the image is smoothed, difference of intensity values among the neighbor pixels is reduced. Thus, values of gradient magnitude become smaller. As a result, less gradient magnitude goes beyond the threshold, and fewer pixels in the binary image become edge pixels. So this is the reason that we see fewer edges as sigma gets greater.

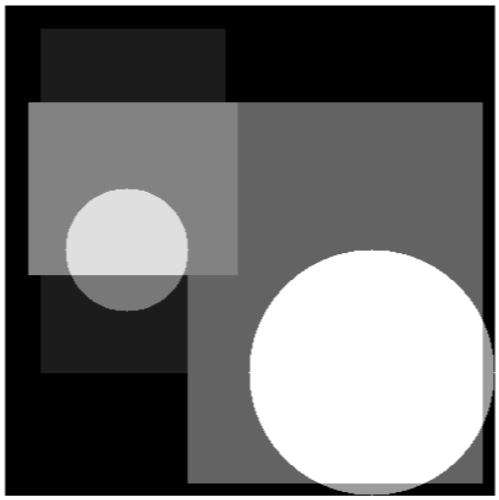
(The parts circled in the image indicates loss of edges.)

# Question 3

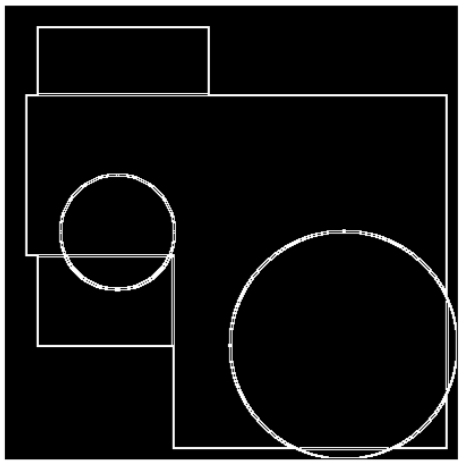
In the part, we use a “second derivative method” to detect edges. We first create a LoG (Laplacian of Gaussian) filter and convolve it with the image of rectangles and circles. Then we find zero crossings in the filtered image and use them to identify edge pixels. The following is the formula of 2D Laplacian of Gaussian:

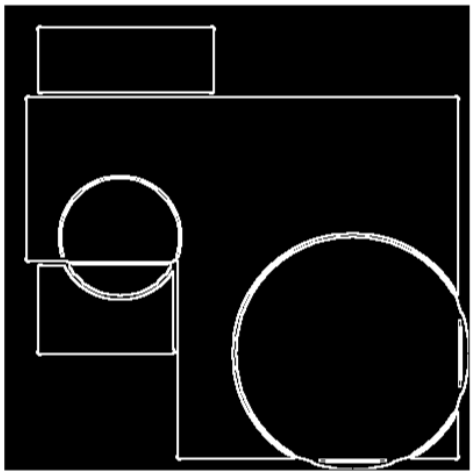
$$\nabla^2 G(x, y, \sigma) \equiv -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

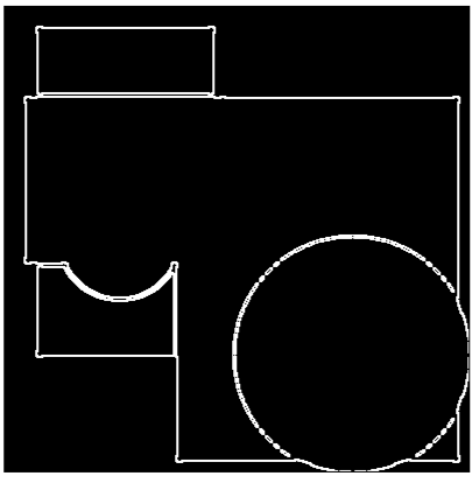
Original image:



Edge image (sigma=0.53) Edge image (sigma=1.65) Edge image (sigma=1.9)







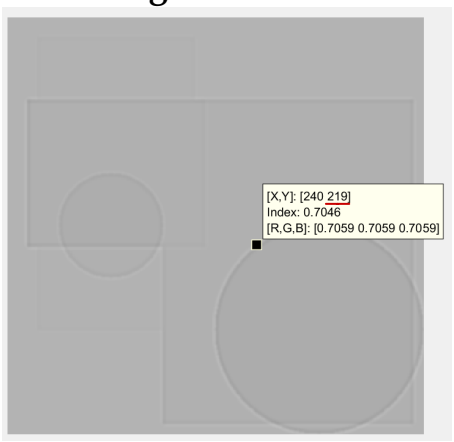
The above are 3 images that shows Marr-Hildreth under different sigma.
As we can see, when sigma gets greater, Marr-Hildreth edge detection forms connected, closed contours at some places in the image. And also, it does a poor job at localization as sigma increases. We will discuss it in the following page.

**—> Discuss how the positions of the zero-crossings depend on σ**

For the variation of position of zero-crossings caused by sigma, the zero crossings tend to move away from the center of shapes as sigma becomes greater (i.e. edges pixels move outwards).I performed a small "test" as follows to confirm my words.

First, I get the output image of the original image convolved with my Laplacian of Gaussian filter. Since this output image is dark in most places (which means most of its intensity values are below 0), I then added 0.7 to each of the pixels in the image. So in this case, when the intensity value of some pixels are close to 0.7, it means that they are around the locations where zero-crossings are supposed to be at. In this way, I got the following two images when sigma equals 1.65 and 1.9 respectively.

**sigma=1.65**




**sigma=1.9**


As we can see from above, the value of Y index indicates an outward movement of the zero-crossings as sigma gets greater.

**—> Compare the behavior of the zero-crossings for edges that are isolated versus edges that are close to "T junctions"**

**zero-crossings for isolated edge**

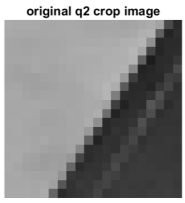
**zero-crossings for edges close to "T junction"**

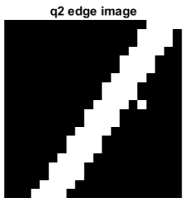
**same "T junction" in the original image**

Marr-Hildreth edge detector performs well in detecting isolated edges since the zero-crossings of them are clearly shown in the binary image. But it fails to detect some of the "T junctions" as above. The zero-crossings for part of the "T junction" are not presented in the binary image.

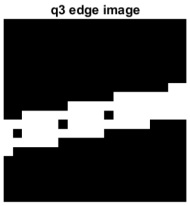
# Question 4

**—> The relationship between images and edge plots in question 2 and 3**



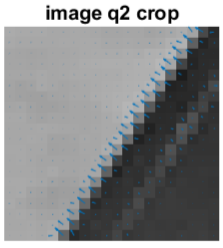



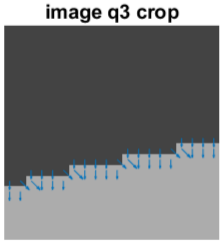

In question 2, we use a "first derivative" method to perform edge detection. We first calculate the local differences (used as gradient) in both x and y directions, and then calculating the gradient magnitude and using a threshold to plot edge pixels in a binary image. So an edge pixel in the edge plot corresponds to the pixel at the same location in the original image whose gradient magnitude is greater than some threshold.


In question 3, we use a "second derivative" method to perform edge detection. We first apply a two dimensional Laplacian of Gaussian to the image, which is the equivalent of taking the second derivative of the image. Then we loop through every pixel in the Laplacian of the image and look for sign changes. If there is a sign change and the slope across this sign change is greater than some threshold, mark this pixel as an edge.

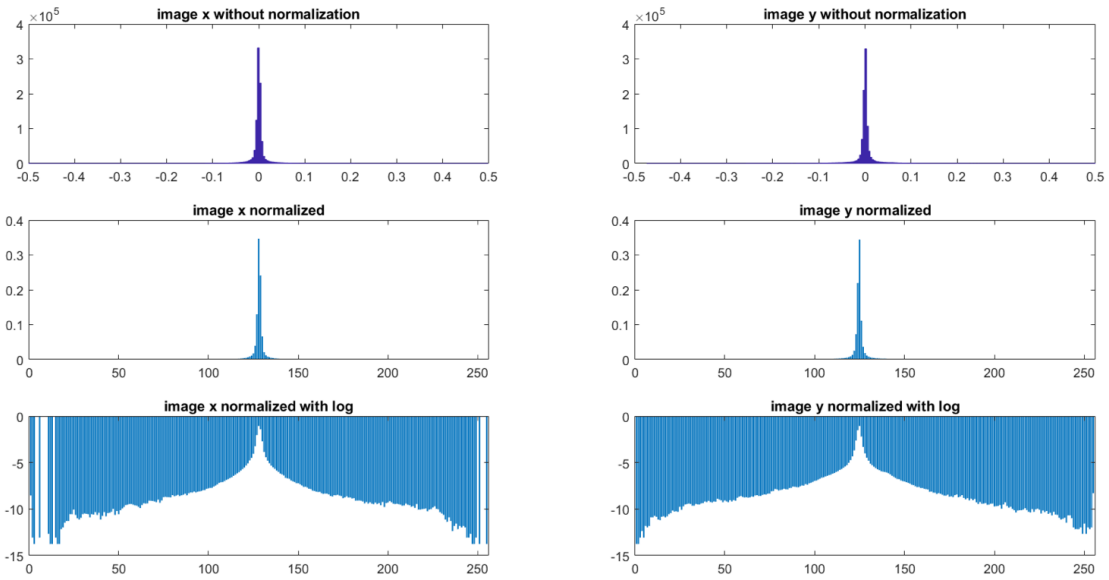
**—> Compare how image gradients vary at the intensity edges, at regions of flat intensity**




At the intensity edges, image gradients point at direction where the intensity values of pixels increase most significantly. This direction is perpendicular to the edge. The magnitude of image gradients (tell from the length of blue arrows in the left images) at the intensity edges is clearly greater than that at regions of flat intensity.

# Question 5




**Histograms of image (sigma=0.4)**

In this question, we first use hist() function to plot a frequency distribution of the partial derivatives of Gaussian filtered images from question 2. Next, we want to analyze the difference between Gaussian and the frequency histograms in both x and y direction (the two histograms on the top). Since it is hard to tell whether they are different from Gaussian, we first normalize them and then take logarithm. We need to focus on the tails of the two histograms at the bottom. The formula for Gaussian is:

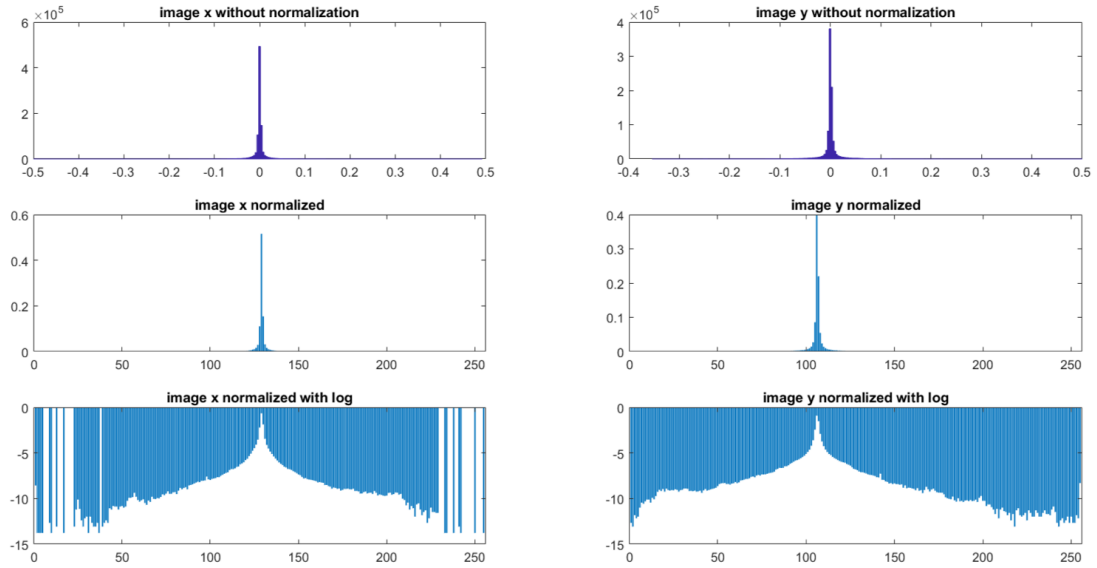
$$G(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

If we take logarithm of it, we will get a quadratic function whose plot is a negative parabola. Apparently, the plots we get above are not parabolic, especially for the tails where the frequency distributions we get are more flat than that of Gaussian. This means the frequency distributions of gradient in both directions are not Gaussian.

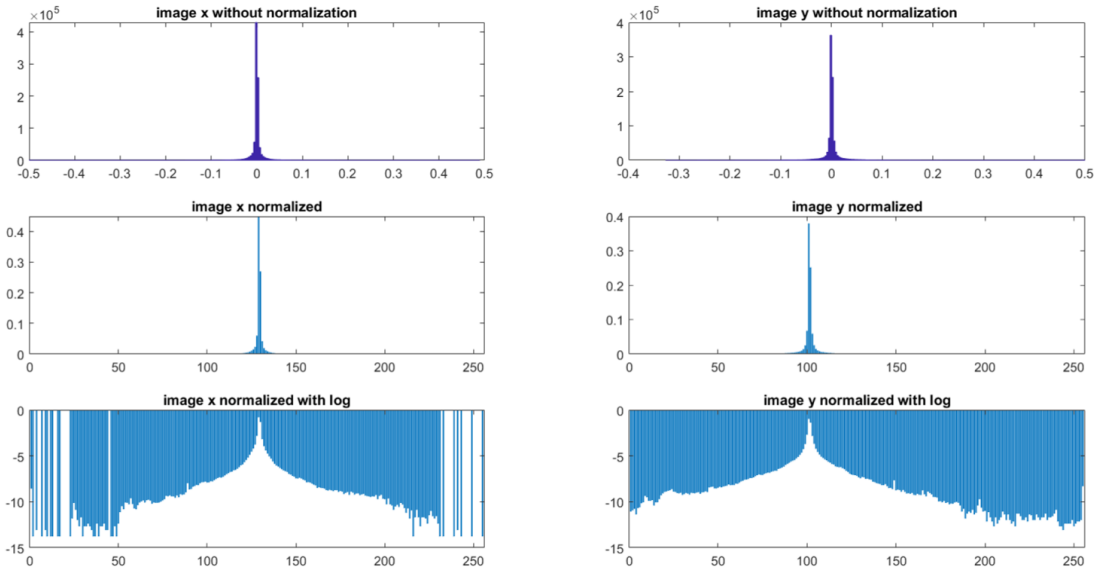
**—> Why does this difference occur?**

As we know, the intensities of neighboring pixels tend to be similar. Since nearby pixels tend to have similar intensities, the intensity difference of nearby pixels tends to be close to 0. Both "local difference filter" and "Gaussian filter" is used to reduce this difference to make it closer to 0.
But the "local difference filter" only reduce the intensity difference between pixels in one direction (x or y) of the central pixel, whereas the "Gaussian filter" reduce difference among pixels in a neighborhood (e.g. a 3x3 neighborhood) of the central pixel. So the intensities of pixels in an image filtered by Gaussian tend to be less variate (i.e more gathering around the mean intensity), whereas the intensities of pixels in an image filtered by "local difference filter" tend to distribute more evenly distributed (i.e. variate) along the axis, especially for the tail part since it only reduces intensity difference in one direction.

## —> Histogram analysis for different values of σ




**Histograms of image (sigma=0.8)**




**Histograms of image (sigma=1.6)**

As sigma grows, the image gets more smoothed, which means less intensity difference among the pixels in the image. As we can see from above, the tail parts of the logarithm histograms above have either less number of columns or the size of the columns get reduced. This indicates that more intensities of pixels in the image tend to be closer to the mean intensity when sigma gets greater.