All lab assignments must be done individually. You should submit your codes to SUCourse during the lab hours (In-Lab Code Submission) and you should submit your lab-reports within one week after the lab (Post-Lab Report).
Grading : In-lab Codes has a weight of 20% and Post-lab Reports has a weight of 80%. Your programs should be modular, bug-free, commented in MATLAB, self- explanatory. Also, your report should include necessary comments and discussions. Please note that if you miss the lab, you will get automatically zero (even if you submit post-lab report)

In this lab, you will implement edge detection algorithms based on Sobel, Prewitt, and Laplacian of Gaussian operators.

Important Note: You should complete the lab until the end of the lab hours and submit all your codes to SUCourse as a single zip file. Deadline for submission to SUCourse is **until the end of the lab,which is 18:30**.

Things to do:

Your functions must be as generic as possible, i.e., don't make any assumptions about the size, the type and the colors of the images. Your functions must convert the image to grayscale if it is colored and you must employ the row and column numbers of the images as variables.

- **Sobel Operator:** *Sobel* filtering is a discrete 2D derivative operation which can be applied with the following kernels

-1	0	1
-2	0	2
-1	0	1

X Filter

-1	-2	-1
0	0	0
1	2	1

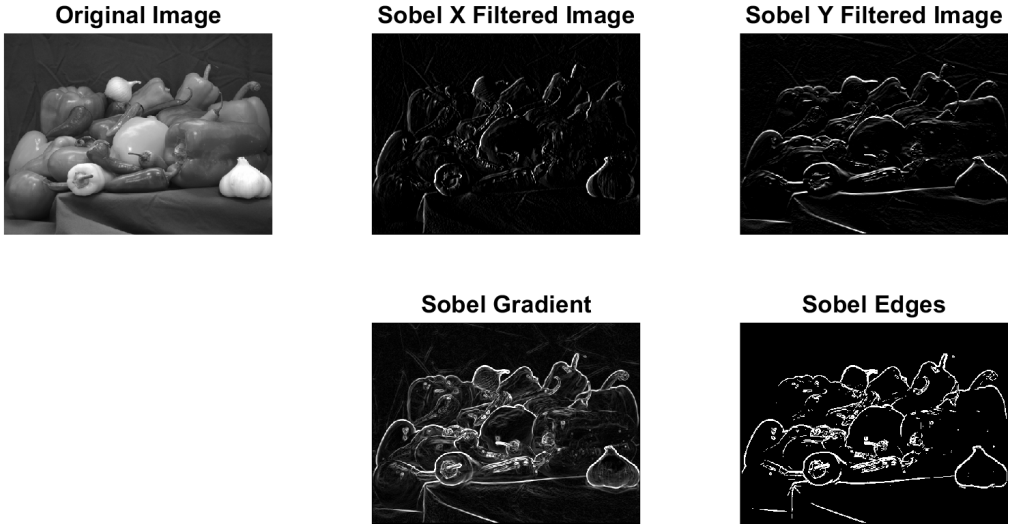
Y Filter

To detect edges in a grayscale image, the gradient image obtained by the horizontal and the vertical Sobel operators is binarized by a threshold. The gradient image is calculated as

$$G(p) = \sqrt{G_x(p)^2 + G_y(p)^2} \tag{1}$$

Now write a function which takes an image and a threshold value as inputs and utilize "**Sobel filters**" to return a binary image of detected edges. Your function name should be "lab3sobel.m".

Your results should look as follows:



- **Prewitt Operator:** *Prewitt* filtering is also a discrete 2D derivative operation which can be applied with the following kernels

-1	0	1
-1	0	1
-1	0	1

X Filter

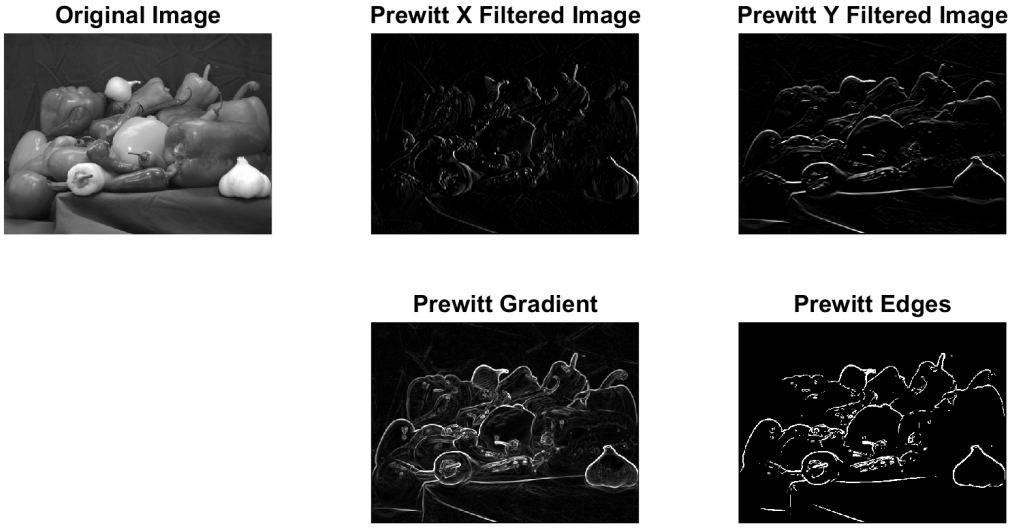
-1	-1	-1
0	0	0
1	1	1

Y Filter

To detect edges in a grayscale image by employing Prewitt operators, the same procedure is applied as Sobel based edge detection except the kernels are changed.

Now write a function which takes an image and a threshold value as inputs and utilize "**Prewitt filters**" to return a binary image of detected edges. Your function name should be "lab3prewitt.m".

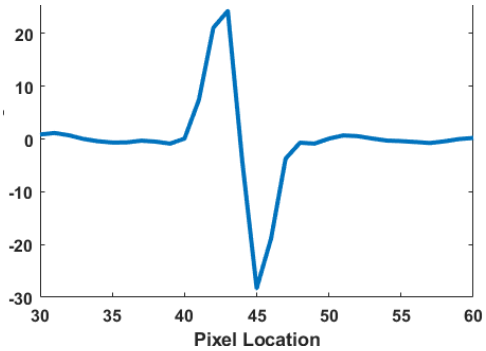
Your results should look as follows:



- **Laplacian of Gaussian:** As Laplace operator may detect edges as well as noise, it is desirable to smooth the image first by a Gaussian filter. Applying the Laplacian for a Gauss-filtered image can be done in one step of convolution with the following kernel

0	1	0
1	-4	1
0	1	0

Different from the Sobel and Prewitt based edge detection algorithms, the zero crossings Laplacian of Gaussian (LoG) filtered image represent the edge pixels. To understand how to determine the zero-crossings of the LoG Filtered image, consider LoG profile for a row data .



One can detect the zero-crossing of such profile by detecting sign change. Since there might be noise in the data, it is wiser to draw a line segment between the extremums of the LoG profile and declare a strong edge when its slope is bigger than a threshold. Also, we may have a non-zero pixel value at the zero crossing due to noise, therefore, another threshold is useful to determine the zero-crossings.

By considering a 3x3 sliding window operation,the zero-crossings after LoG filtering can determined by using the slope and the sign change of the current pixel and its neighbors as follows:

i) initialize all zero edge image E(i,j)=0;

ii) if the center pixel value in the sliding window $\geq$ *threshold* T_1*,*

if there is a sign change between the center pixel and its left-right or up-down neighbors

Then determine the slope of the line segment,

if the slope is bigger than $\geq$ *threshold* T_2*,*

Declare the center pixel as an edge point (i.e. E(i,j)=255 for the center pixel)

if the center pixel value in the sliding window <threshold T_1*,*

if there is a sign change between right-left or up-down neighbors of the center pixel,

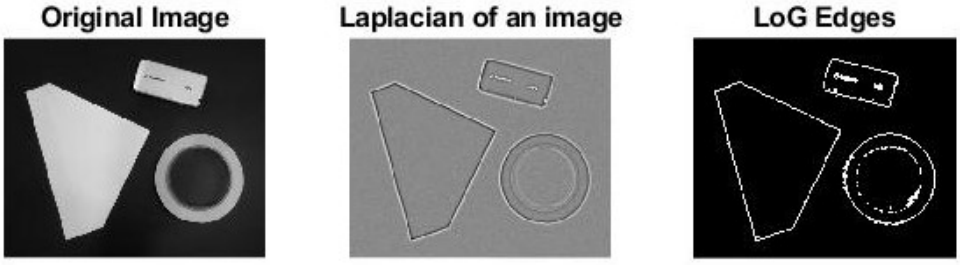
Then determine the slope of the line segment,

if the slope is bigger than$\geq$ *threshold* T_2*,*

Declare the center pixel as an edge pixel (i.e. E(i,j)=255 for the center pixel)

Now write a function which takes an image as input, thresholds T_1 and T_2 and utilize "**Laplacian of Gaussian**" to return the LoG filtered image. Your function name should be "lab3log.m".

By optimizing your thresholds, your results should look as follows:



Post Lab

Post lab reports must include brief explanations of the functions that you implemented in this lab. Provide resulting images by utilizing all these functions. Comment on your results. Discuss the differences of the edge detectors implemented in this lab in terms of their applicability for different scenarios.

Deadline for post lab report submission to SUCourse: **19 November 2020, 23:55**.