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

Important Note: Submit all your codes to SUCourse as a single zip file. Deadline for submission to SUCourse is **until the end of the lab**.

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

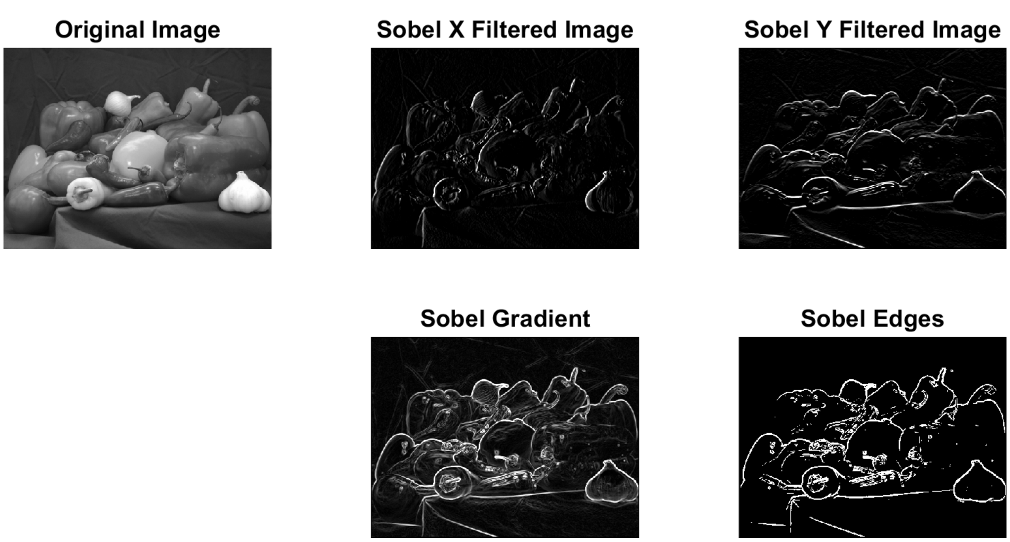
Y Filter

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

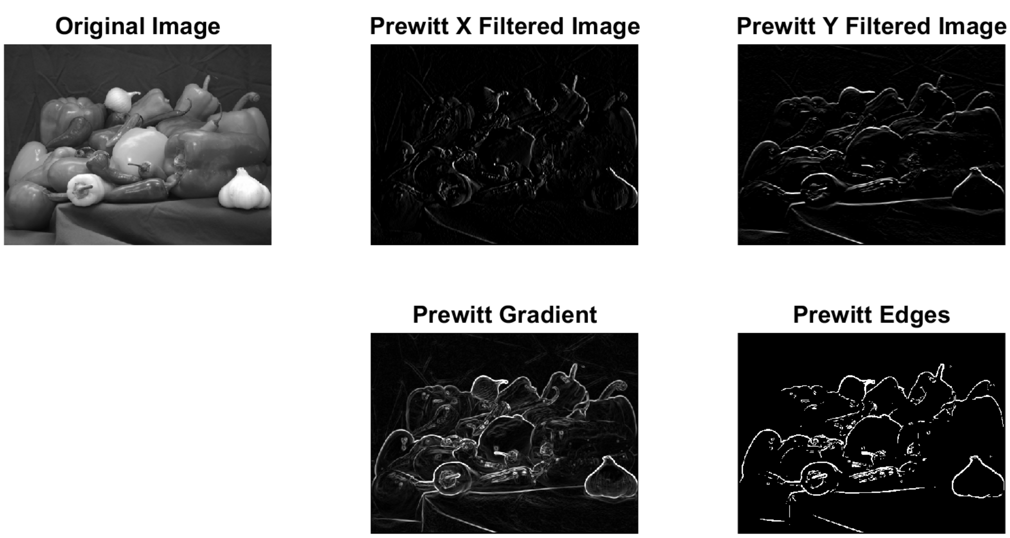
-1	-1	-1
0	0	0
1	1	1

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:



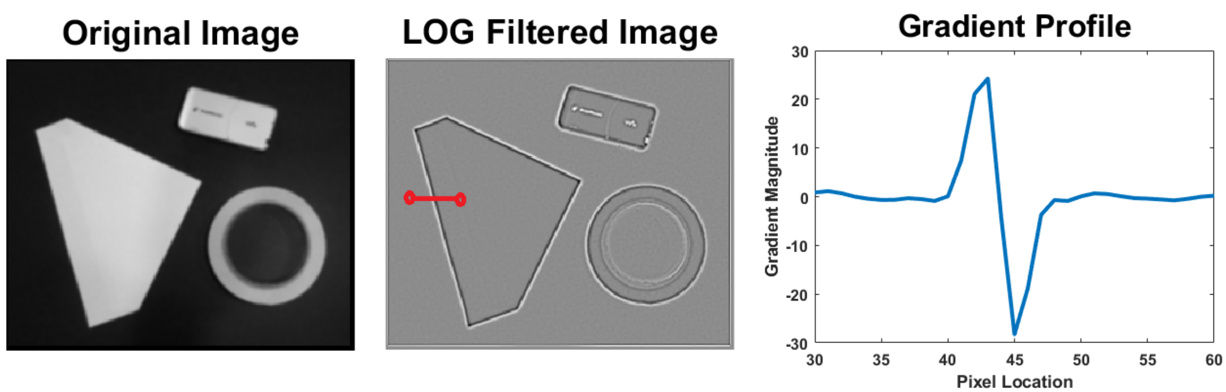
- **Laplacian of Gaussian:** As Laplace operator may detect edges as well as noise, it is desirable to smooth the image first by a Gaussian filter. Applying the Laplacian for a Gauss-filtered image can be done in one step of convolution with the following kernel

0	1	0
1	-4	1
0	1	0

Different from the Sobel and Prewitt based edge detection algorithms, zero crossing points represent the edge pixels. It should be noted that the determination of the zero crossing points is not a trivial task. Therefore, extract the gradient profile from a line segment of the LoG filtered image belonging to an edge region and investigate the zero crossing behavior as shown in the results below.

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

Your results should look as follows:



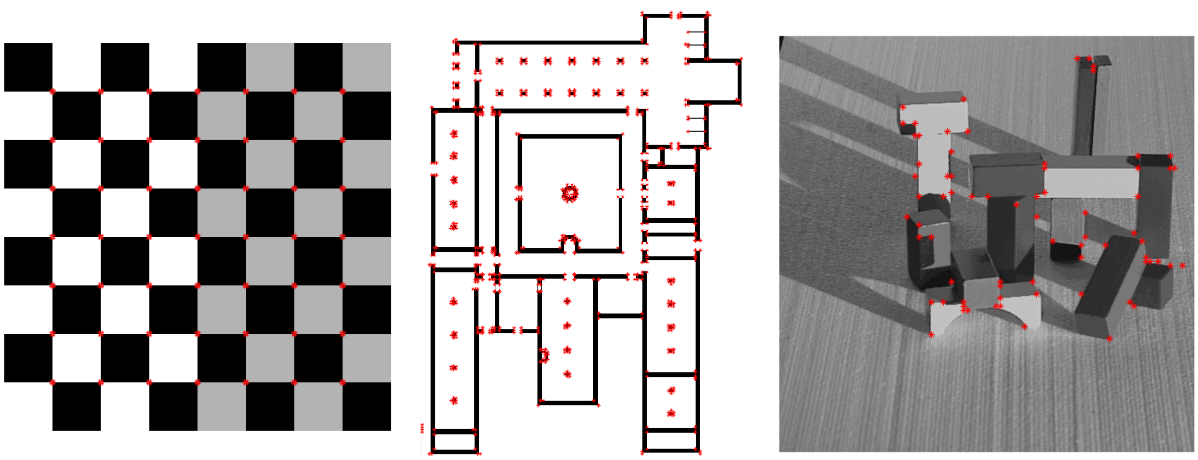
- **Corner Detection:** In order to detect the corners in a grayscale image by employing Kanade-Tomasi algorithm, you can apply the following steps:
  - Compute the gradients G_x and G_y (Hint: You can use `[Gx,Gy] = imgradientxy(I);`)
  - Create H matrix of each pixel in a window as follows

$$H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{2}$$

    where I_x and I_y are the image gradients of a window along x and y directions, respectively.
  - Compute the eigenvalues λ_1 and λ_2 of H.
  - If $\min(\lambda_1, \lambda_2)$ >Threshold, add the pixel coordinates to corner list.

Now write a function which takes an image and a threshold as inputs and utilize "**Kanade-Tomasi Corner Detection Algorithm**" to return the detected corner points. Your function name should be "lab3ktcorners.m".

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. Moreover, discuss the performance of Kanade-Tomasi corner detector on different images.

Deadline for post lab report submission to SUCourse: **30 October 2018, 13:00**.