State University of New York at Buffalo

CSE 473/573 Fall 2015 Homework Set #2

Date: Monday October 26, 2015; Due: Wednesday November 04, 2015 at 3:00PM

Homework Summary

Language :	**Python**
Modules :	**OpenCV, MathPlotLib, Numpy**

Note: I have implemented custom sobel filter (filters.py) , which work as intended but some corrections need to be done in the pixel mapping part. (Check: custom_sobel_1D.py and custom_sobel_2D.py and filter.py – for the actual implementation of sobel filter module).

Problem 1: (1D and 2D convolution of images)

We were to do 2D and 1D convolutions using the sobel filter, which is a separable filter I.e can be split as a product of two 1D matrices or kernel. Sobel filter kernel come in two parts for 2D convolution eg: Gx ad Gy as per the problem statement, each being a product of kernel with the image.

Since

Gx = kernel(x) * Image and similarly
Gy = kernel(y) * Image

Also to be noted, kernel(x) is kernel(y) rotated by 90 degrees , which is the special feature about the sobel filter.

Implementation

(a) Steps:

I. Read Image (lena_gray.png) from disk.
II. Store the kernels as per the homework statement.
III. Start the timer for the convolution of Gx,Gy and resulting into G.
IV. Compute the convolutions and hence computing Gx,Gy and G.
V. Stop the timer and make a note for answering (c) of the problem statement.
VI. Generating subplots and plotting Gx,Gy and G accordingly.

Output: (Source: HW1a.py)
Time Elapsed : 6.47 msec(s)

2D Convolution using Sobel Filter:

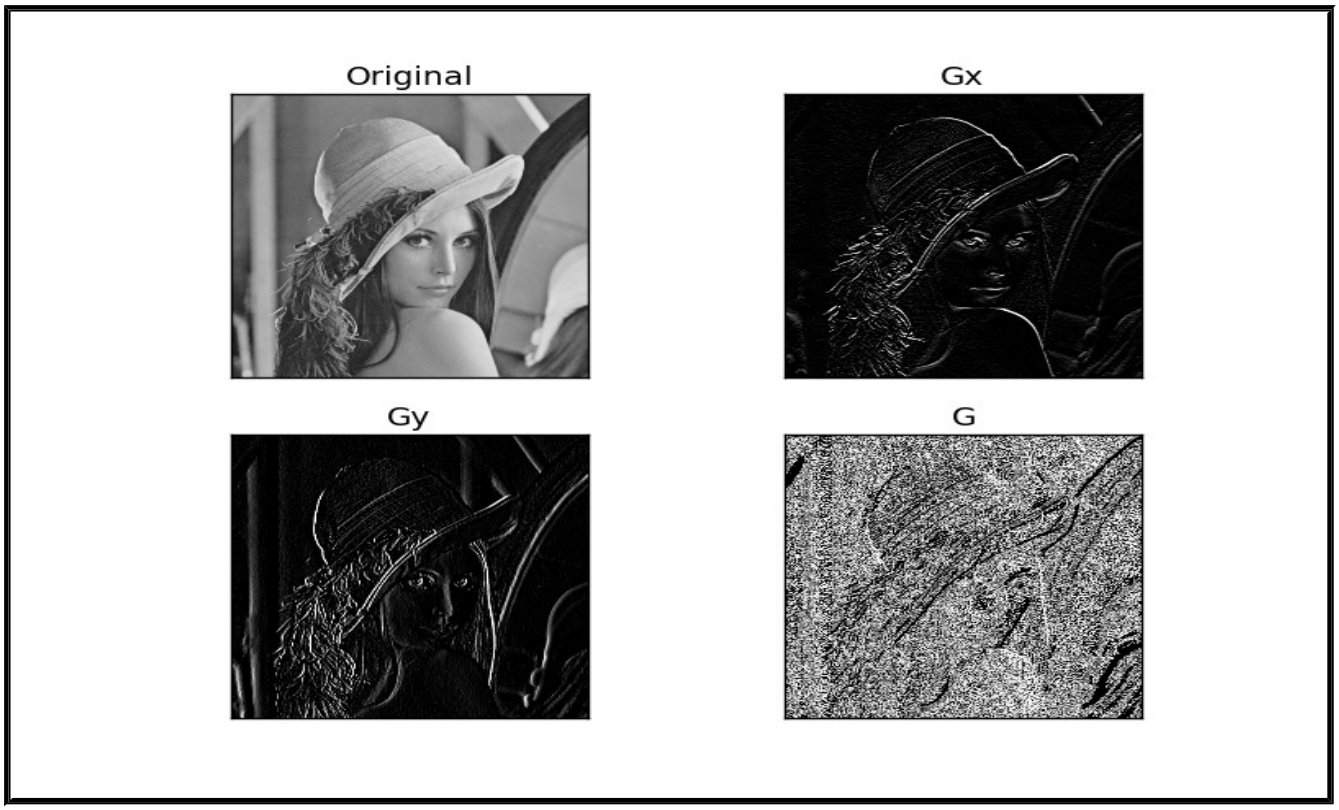
(b) Steps:

I. Read Image (lena_gray.png) from disk.
II. Store the separable kernel pairs as per the homework statement.
III. Start the timer for the convolution of Gx,Gy and resulting into G.
IV. Computing Gx using cv2.sepFilter2D library, with input kx1, ky1, similarly Gy.
V. Compute G as G = np.sqrt(np.square(Gx) + np.square(Gy)).
VI. Stop the timer and make a note for answering (c) of the problem statement.
VII. Generating subplots and plotting Gx,Gy and G accordingly.

The result was same as the part (a) , since the 2 1D kernels worked similar to using a 2D kernel but required some extra time when compared to using 1D kernel.

Output: (Source: HW1b.py)
Time Elapsed : 17.50 msec(s)

1D Convolution using Separable Sobel Filters:



(c)

I. Execution time using 2D kernel : 6.47 msec(s)
II. Execution time using 2D kernel : 17.50 msec(s)

Problem 2: (Histogram Equalization)

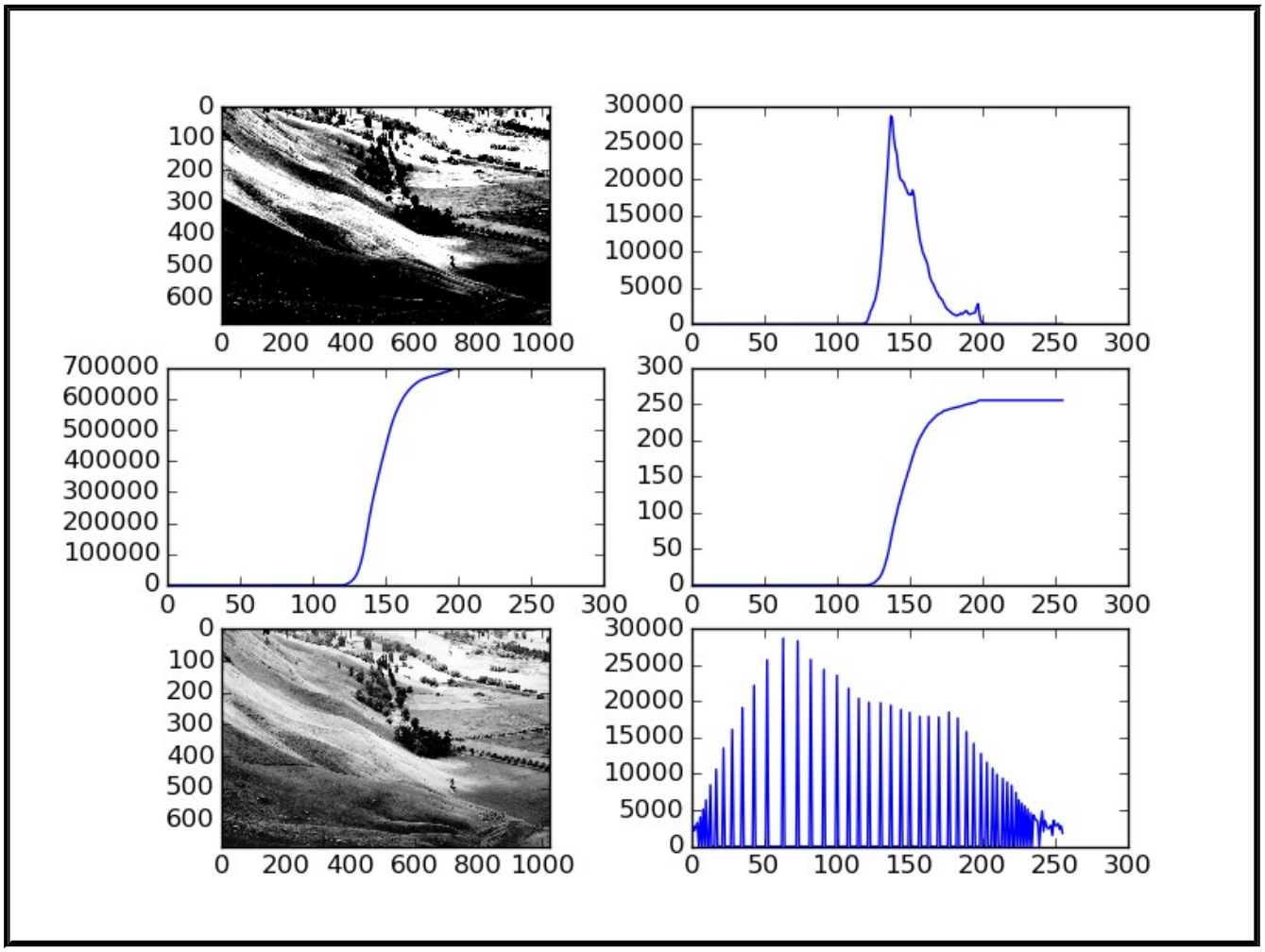
Histogram is a standard technique to improve the contrast of a scene, by computing the histogram of the image pixels and equalizing the image histogram by using the transformation function , which being a part of the histogram equalization as specified in the problem statement.

Implementation

(a) Steps:

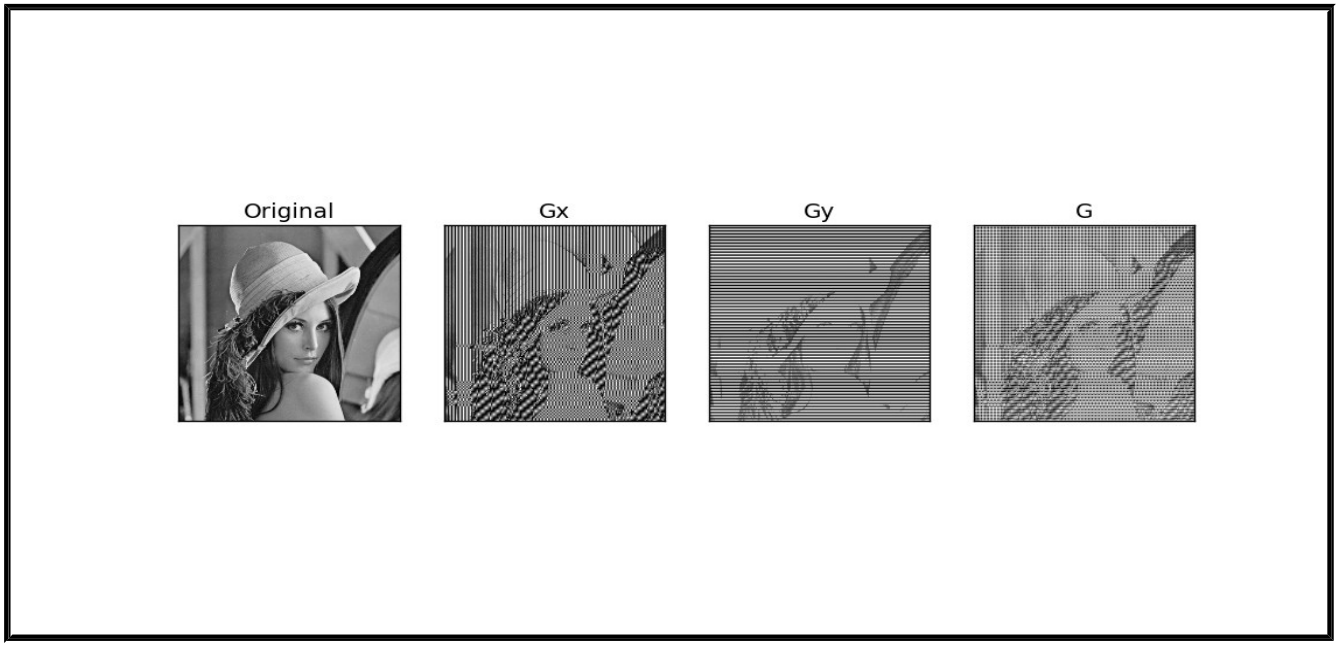
I. Read Image (host.png) from disk in grayscale mode.
II. Create a copy of the original image used during transformation.
III. Initialize the histogram bins of the size of 256 , since we work on grayscale.
IV. Iterate through image and incrementing pixel counts by 1. (Alg-Step2)
V. Compute the cumulative histogram using original image histogram. (Alg-Step3)
VI. Generate the transformation histogram as per (Alg-Step4)
VII. Using the transformation histogram as lookup, update image pixels
IX. The resultant image becomes the histogram equalized image.
X. Use pyplot to plot the various phases of the histogram equalization

Output: (Source: HW2.py)

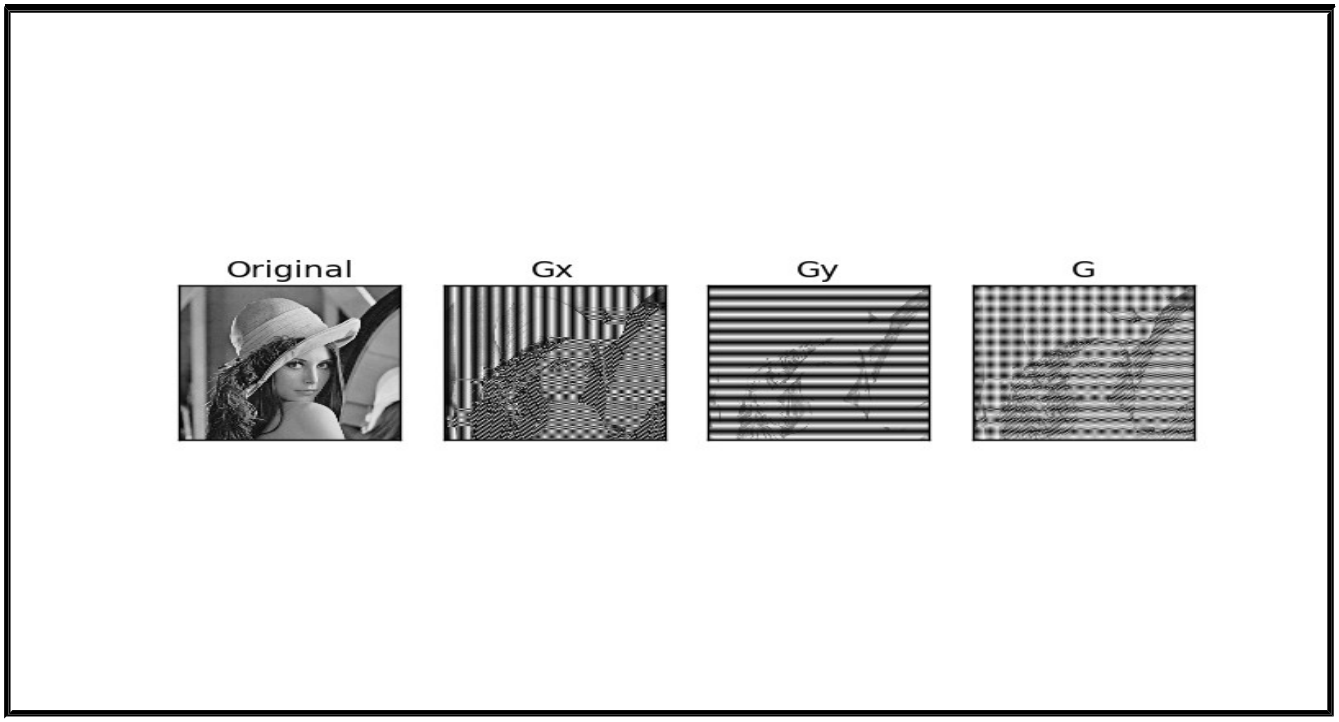


Images from the custom sobel filter modules:

2D Convolution Example using Custom Sobel Filter



1D Convolution Example using Custom Sobel Filter



END OF SUMMARY