CSE 473 Computer Vision and Image Processing

Programming Assignment # 1:
1D, 2D Convolution on Images and Histogram Equalization

Instructor Radhakrishna Dasari

Due Date: June 29, 2016

Winnie Liang
wliang6@buffalo.edu
50038993

1D and 2D Convolution on Images

Original Image of Lena Söderberg

Sobel filter is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasizing edges. It computes gradient in x and y directions.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \qquad G_{mag} = \sqrt{G_x^2 + G_y^2}$$

(a) Perform 2D convolution on grayscale Image lena_gray.jpg with filters specified above to obtain gradient images Gx and Gy. Display the three images Gx, Gy and the gradient magnitude image Gmag. (20%)

2D Convolution Gx	2D Convolution Gy	2D Convolution Gradient
		

Convolution is done by shifting the Sobel Kernel across the original image, one pixel at a time. At each pixel, the kernel is applied to the pixel and its neighbors, then summed to produce a new value. Detecting edges using Sobel Operator involves two separate kernels to calculate the x and y gradients in the image.

Here are the steps for my implementation in performing 2D convolution using Sobel filters:

1) Iterate through each pixel of the original image.
2) Apply the gradient kernel Gx in the x direction.
3) Apply the gradient kernel Gy in the y direction.
4) Find the gradient magnitude G using the $\sqrt{Gx^2 + Gy^2}$ equation.
5) Normalize the gradient magnitude to fit the range 0-255.
6) Display Gx, Gy, G results using matplotlib.

A timer is used to compute the time for the convolution process.

Winnie Liang

The filter kernels of Sobel filter are linearly separable:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

(b) Perform 1D convolution on grayscale Image lena_gray.png with 1D-filters specified above to obtain gradient images Gx and Gy. Verify the result after 1D convolution is same as the one obtained from 2D convolution from (a) (20%)

1D Convolution Gx	1D Convolution Gy	1D Convolution Gradient
		

After displaying the images above, the 1D convolution results can be verified to be the same as the 2D convolution results.
Here are the steps for my implementation in performing 1D convolution using Sobel filters:

1) Iterate through each pixel of the original image lena_gray.png.
2) Apply the gradient separable kernel pairs to get Gx in the x direction.
3) Apply the gradient separable kernel pairs to get Gy in the y direction.
4) Find the gradient magnitude G using the $\sqrt{Gx^2 + Gy^2}$ equation.
5) Normalize the gradient magnitude to fit the range 0-255.
6) Display Gx, Gy, G results using matplotlib.

A timer is used to compute the time for the convolution process as well.

(c) Given an MxN Image and a PxQ filter, compute the computational complexity of performing 2D convolution vs using separable filters with 1D convolution (10%)

2D Convolution computational complexity:
0.006763999999999992 seconds
1D Convolution computational complexity:
0.013694000000000095 seconds
The time complexity for performing 2D convolution is faster than the time complexity of using separable filters with 1D convolution. These times are computed from the timer implemented in the code. The time.clock() function returns the current processor time in seconds. By calling the clock at a start time and end time, the total runtime is achieved by subtracting the two.

Histogram Equalization

Histogram equalization is a technique for adjusting image intensities to enhance contrast. Capture or pick an Image of your choice, convert it to grayscale and perform histogram equalization using the algorithm below (excerpt from Milan & Sonka):

Plot the histogram and cumulative histogram of the original image, Transformation function in step 4 and histogram of image obtained after step 5. Also show both original image and enhanced image side by side for comparison in your report

Here are the steps for my implementation in performing Histogram Equalization:

1) Convert input image into grayscale.
2) For an NxM image of G gray-levels, create an array H of length G initialized with 0 values.
3) Compute histogram from the grayscale image by scanning every pixel and increment.
4) Compute cumulative histogram.
5) Formulate lookup table.
6) Remap pixel intensities of original image
7) Compute equalized histogram.
8) Display histograms and the equalized image result.

Original chosen image of a rose

Original → Grayscale → Equalized Images

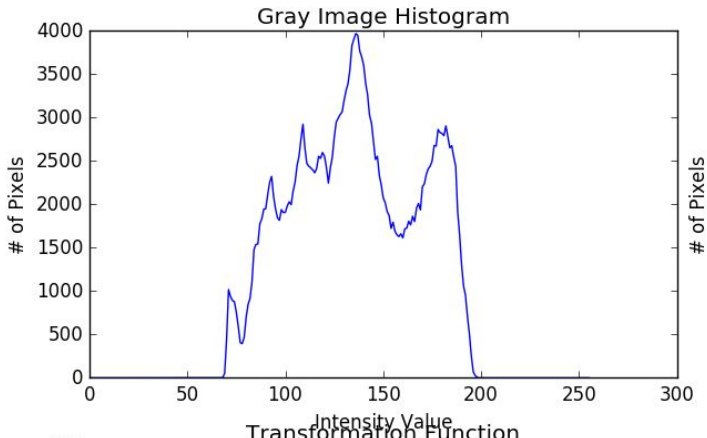
Original Image

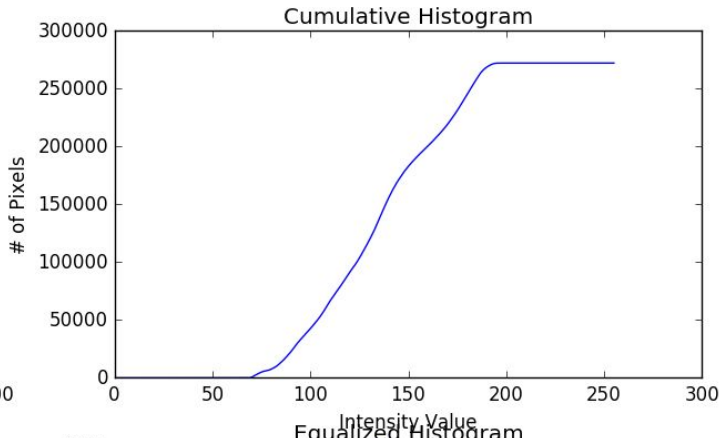
Grayscale Image

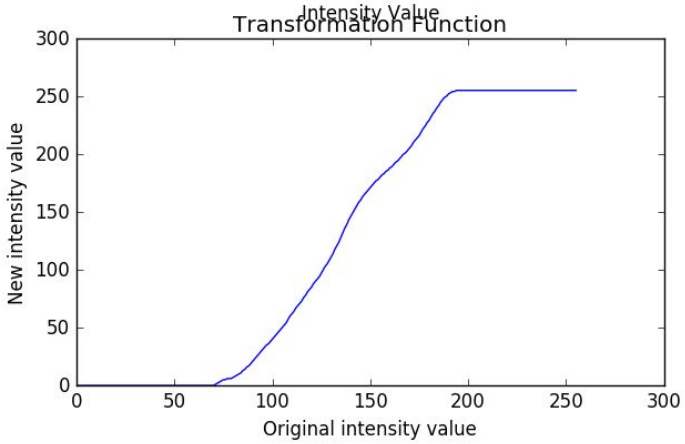
Equalized / Enhanced Image

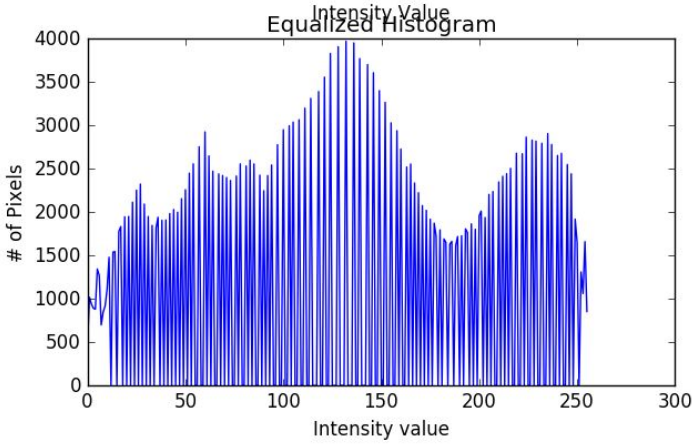
Winnie Liang

Corresponding Histograms to rose.png image









Winnie Liang