Lab 2: Introduction to Image Processing

Students are divided into group of two persons to do this lab assignment.

Please wrap the e-file of your report & MATLAB files in a compressed file and then e-submit this file via TA email address by the due day.

Thank you.

1. Goals

The goal of this lab is to introduce you to some basic concepts in image processing. You will learn how to read and display an image in MATLAB. You will perform simple edge detection on an image we give you as well as on an image of your own. Then, you will scale an image to create its thumbnail version.

This lab is also to prepare you for the higher-level class related to Image Processing that is EE440.

2. Lab resource

- PC with Matlab
- The image DailyShow.jpg in the JPEG format.

3. Background

Digital images consist of pixels (picture elements). When the pixels are placed close to each other, an image viewed on a computer display or printed on a paper appears to be continuous. The number of pixels per inch (ppi) varies with an application. Some monitors can only display 72 ppi. For publishing, 200–1200 ppi is often required. Laser printers are usually capable of 300-600 ppi.

The brightness and color information for each pixel is represented by a number in a two-dimensional array (matrix). The location in the matrix corresponds to the location of a pixel in the image. For example, $x(1,1)$ (usually) identifies the pixel located in the upper left corner, as shown in Figure 1.

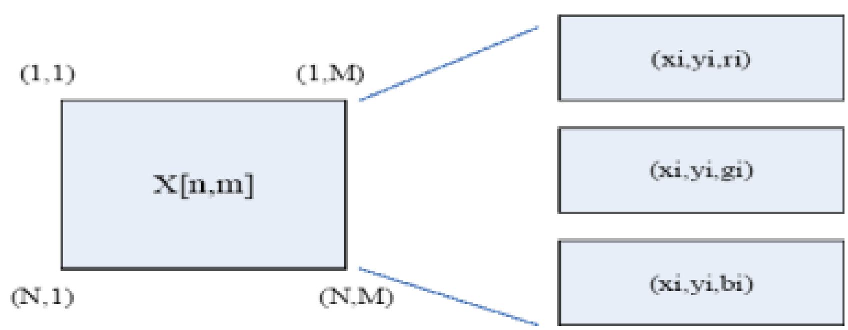

Figure 1. Pixel locations and file format; ri represents red pixel intensity at location (xi, yi) while gi and bi are green and blue pixel intensities, respectively

The pixel values in an 8-bit gray scale image can take any value from 0 to 255. Usually black is encoded by a value of 0 and white by a value of 255. A color image is stored in a three-dimensional array, where the first plane in the 3rd dimension represents the red pixel intensities, the second plane represents the green pixel intensities, and the third plane represents the blue pixel intensities. True color has 24 bits of resolution (8 bits for each of the red, green, and blue planes).

In this lab, you will use the Image Processing Toolbox. To learn more about the functions included in it, type **help images**.

4. Assignments

4.1. Edge Detection (25%)

Use the command **imread('DailyShow', 'jpeg')** to read the image DailyShow.jpg. To display your image, use the Matlab command **imshow()**. Since we will be working with gray scale images in this lab, your next step is to convert your input image to an 8-bit gray scale format using the **rgb2gray()** command. Display the resulting image. Check its size using the **size()** command. Use the title command to add a comment to your image. In your report, include the 8-bit grayscale image and specify its dimensions.

Next, you will perform edge detection on your image. (Edge detection is often a first step into later, more complicated image processing operations.) Two-dimensional convolution, appropriate for images, is implemented in MATLAB in the function **conv2().** Convolution can be used to implement edge detection.

Create the following Sobel vertical edge detection convolution kernel. This mask is designed to respond maximally to edges running vertically relative to the pixel grid. It is a two-dimensional matrix h1[n,m], in Matlab notation:

h1 = [-1 0 1; -2 0 2; -1 0 1]

Next create the following Sobel horizontal edge detection convolution kernel. This mask is designed to respond maximally to edges running horizontally relative to the pixel grid. It is a two-dimensional matrix h2[n,m], in Matlab notation:

h2 = [1 2 1; 0 0 0; -1 –2 -1]

Now, convolve your grayscale DailyShow image with the two edge detection kernels described as follows. Assume M1 is the result of convolving the grayscale DailyShow image with h1 (i.e. M1 is the row gradient of the grayscale DailyShow image), and M2 is the result of convolving the grayscale DailyShow image with h2 (i.e. M2 is the column gradient of the grayscale DailyShow image). Use Matlab to display the row gradient magnitude (|M1|), the column gradient magnitude (|M2|), and the overall gradient magnitude (i.e. (M1^2 + M2^2)^0.5).

Include these edge images (i.e. |M1|, |M2| and (M1^2+ M2^2)^0.5) in your report. Save your MATLAB function in an m-file. Include this file in your E-Submit.

You may notice lots of dark areas in these edge images. To save the toner (or cartridge) of your printer, you can try to figure out a way to reverse the grayscale of your edge images before printing them out. That is, you do a transformation to map the original darkest area to the brightest one, and the original brightest area (e.g. edge) to the darkest one.

4.2. Scaling (25%)

In this assignment, you will investigate scaling of images in the spatial domain. You will scale the image $x[n,m]$ in both the vertical and horizontal directions using the same scaling factor S. An example where you want to use such scaling would be creating thumbnail-sized pictures for a web page.

Write a MATLAB function that has an input argument for the scaling factor S. The function should read in the color DailyShow.jpg image, convert it to a gray scale, and then scale the image by the scaling factor to form a thumbnail image. You are going to scale the original image with scaling factors S = 2 and S = 5.

First, perform a simple scaling – keep one out of S^2 pixels. Since this is a 2D scaling, you can keep the center pixel in each square of S^2 pixels, when S is an odd number and one of the 4 center pixels when S is even.

Next, you are going to perform a more advanced scaling operation. Instead of keeping the center pixel in each square of S^2 pixels, keep the average of all of the pixels in this square.

<u>Display all four results in your report. Be sure to label each image with the scale factor used to create it. For S = 2 and S = 5 respectively, compare the two scaled versions of the original picture (i.e. picking one out of S^2 pixels versus picking the average of each block with S2 pixels). Which one is the better thumbnail image?</u>

4.3. Image Flipping (25%)

Guess how the following images look like when compared to the original image $x[n,m]$

(i) $x[N\text{-}n\text{+}1, m]$

(ii) $x[n, M\text{-}m\text{+}1]$

(iii) $x[N\text{-}n\text{+}1, M\text{-}m\text{+}1]$

where $1 \leq n \leq N$, $1 \leq m \leq M$

Use DailyShow.jpg as the original image (i.e. $x[n,m]$). Verify your guesses by displaying resulting images of (i)(ii)(iii) in your report.

Use ***help fliplr()*** and ***flipud()*** to see more information.

4.4. Image Expanding (25%)

When an image is scaled up to a larger size, there is a question of what to do with the new spaces in between the original pixels.

Suppose now we want to expand a 1D signal by a factor of 2, then you can think of two common ways:

(i) simply double each sample --- value replication; and

(ii) make the new samples half way between the previous samples --- 2 taps linear interpolation.

Now, write a MATLAB function that can expand the input image DailyShow.jpg with dimension NxM to a 2Nx2M image using “bilinear Interpolation” that is shown in Figure 2.

A	B
C	D

A	b	B	
a	e	d	
C	c	D	

$b = (A+B)/2$
$c = (C+D)/2$
$a = (A+C)/2$
$d = (B+D)/2$
$e = (A+B+C+D)/4$

Figure 2. Enlarging the image by a factor of 2 using bilinear interpolation

Display this 2Nx2M image in your report. Save your MATLAB function in an m-file and include this file in your E-Submit.

GOOD LUCK!!!

This lab is modified from Dr. A. Miguel at Seattle University.