Lab 3: Introduction to Image Processing

(Based on an early version by Dr. A. Miguel of Seattle University)

This lab will involve signal processing on images, which is a two-dimensional version of what we have been doing with audio signals. An image will be provided for initial experiments, but you will need to find an image of your own to work on as well for parts 3-5. Students should work on the programming and common image in teams, but the report and the other image that you work with should be your own. You can download an image from the web or off your cell phone or a digital camera. If the image is in color, convert it to gray scale as described in section 2.2.

1. Purpose

The purpose 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 convolutions to do edge detection and blurring using an image that we give you as well as on an image of your own. Then, you will scale an image to create its thumbnail version.

If you enjoy this lab, you may be interested in some of the higher-level undergraduate classes related to image processing: EE 440, CSE 455, and CSE 457.

2. Background

2.1 Digital Images

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 resolution of the image depends on pixel density, which is often measured as the number of pixels per inch (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. The pixel values in an 8-bit gray scale image can take any value from 0 to 255. Usually black corresponds to a value of 0 and white by a value of 255. A color image is often stored in a three-dimensional array, where the first plane represents the red pixel intensities, the second plane represents the green pixel intensities, and the third plane represents the blue pixel intensities. (The separation into these three colors underlies the mechanisms in electronic sensing and display, motivated by how humans perceive color.) True color has 24 bits of resolution (8 bits for each of the red, green, and blue planes).

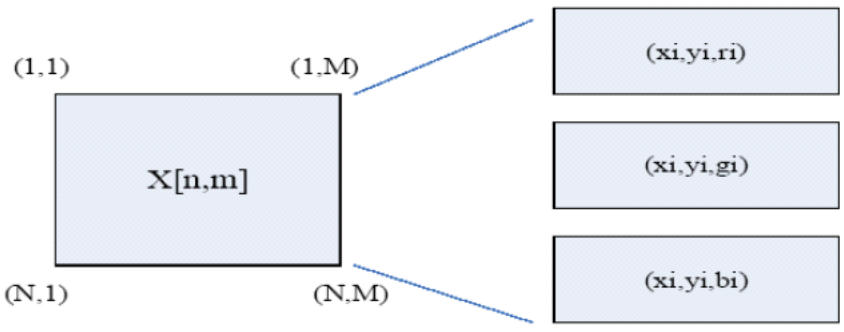


Figure 1. Pixel locations and file format; ri represents red pixel intensity at location (xi,yi) while gi and bi are green and blue pixel intensities, respectively.

2.2 Working with Images in MATLAB

In this lab, you will use some simple image processing commands. To develop your intuitions for how the different types of image processing affect different aspects of an image, you will work with three images, two that are provided (wanda_g.jpg and egg_g.jpg) and one of your own. Use the provided images for the in-lab check-off since we know what to expect for those. For your written report, use your own image (each team member should have a different image) and optionally include the other images to support a point you may want to make.

The main commands that you will use are:

X=imread(filename) – read an image file
Xg=Rgb2gray(X) – if your original image is in color, convert to grayscale
imshow(X) – display an image on the screen
imwrite(X,filename) – save an image to a file
Y=conv2(Xd,h,'same') – 2-dimensional convolution.

The extension of the image filename tells the read/write functions what format to use, or you can specify the format explicitly, as in

X=imread('wanda_g',jpeg)

The images provided are in jpeg format, but MATLAB can handle other formats if the image that you want to use for your individual report is in a different format.

For simplicity, we will be working with gray scale images in this lab, so if your image is in color you will need to convert each input image to an 8-bit gray scale format using the rgb2gray( ) command. Check its size using the size( ) command. Note that the color and gray scale images should differ in size, since color images have 3 values for each pixel.

To display an image, use the MATLAB command imshow( ). You can use the title command to add a comment to your image.

The read/write/show commands assume that your variables are uint8 type, but the conv2 command (and most other MATLAB commands) want variables that are double. You can convert between these as in: Xd=double(X) and use imshow(uint8(Y)).

The last argument in the conv2 command is optional. Just as convolving two finite length time signals typically gives you a signal that is longer, convolving an image with a 2D impulse response increases the size of the image. This will correspond to a small difference at the edges. The 'same' argument trims the edges and keeps all the images the same size for easy comparison.

3. Convolution with Images: Blurring and Edge Detection

Assignment 1

Convolution can be used to implement edge detection and smoothing operations for both time signals and images. For images, smoothing has the effect of blurring. Two-dimensional convolution, appropriate for images, is implemented in MATLAB via the function conv2( ).

Smoothing involves averaging over a neighborhood. For a time signal, a simple 3-sample moving window smoother has an impulse response of

$h[n]=(1/3)(u[n+1]-u[n-2])=(1/3)(\delta[n+1]+\delta[n]+\delta[n-1])$.

The equivalent impulse response for an image is two-dimensional. For example, the 2D version of the impulse response above is, e.g. in MATLAB notation:

```
hs = (1/9)*[1 1 1; 1 1 1; 1 1 1]
```

In your lab, use a 5x5 smoothing matrix. Convolve your grayscale images with hs. This should have the effect of blurring your images. The blurring effect of a 5x5 smoother is not large on these images. To get more blurring, you can try using a bigger matrix (just as a longer time window would have a greater smoothing effect on a time signal) or you can convolve with the smoothing matrix multiple times (which has the effect of increasing the window size). Save the resulting images for presentation to the TA and your written report.

Next, you will perform edge detection on your image. (Edge detection is often a first step used in more complicated image processing operations.) 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 $h_1[n,m]$, 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 $h_2[n,m]$, in MATLAB notation:

```
h2 = [1 2 1; 0 0 0; -1 –2 -1]
```

Now, convolve your grayscale images with the two edge detection kernels described as follows.

Assume M1 is the result of convolving the grayscale image with h1 (i.e. M1 is the row gradient of the grayscale image), and M2 is the result of convolving the grayscale image with h2 (i.e. M2 is the column gradient of the grayscale 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). Save the resulting images for presentation to the TA.

[Optional]

You may notice a lot of dark areas in these edge images, since the max value corresponds to white and so the edges are white. 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.

IN-LAB CHECK-OFF: On the computer, use MATLAB to display the following images and show them to the TA:

- The original gray-scale images
- The blurred images
- The magnitude images |M1| and |M2|
- The overall gradiant magnitude $(M1^2 + M2^2)^{1/2}$

Optionally, you can show the reverse grayscale version of the edge images.

4. Spatial Scaling

In this section 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.

Assignment 2

Since images have a 2-dimensional argument, there are two ways to reverse an image (mirror image or up/down). Guess how the following images look like when compared to the original image X[n,m]

(i) $X[N-n+1, m]$
(ii) $X[n, M-m+1]$
(iii) $X[N-n+1, M-m+1]$
where $1 \le n \le N,\ 1 \le m \le M$

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

IN-LAB CHECK-OFF: Display the resulting images of (i) (ii) (iii) using either the egg or wanda for your TA.

Assignment 3

Write a MATLAB function that has an input argument for the scaling factor S. The function should read in an image, and then scale the image by the scaling factor to form a thumbnail image. 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. Apply scaling to the images, and *notice whether any aspects are more sensitive to scaling than others or better suited to different methods of scaling*. Save the resulting images for presentation to the TA.

IN-LAB CHECK-OFF: Display the following versions of the class images for the TA, being sure to specify the method used to create each.

- Original gray scale image
- S=5 images, single pixel selection vs. average method for scaling
- A thumbnail no larger than 100x100

WRITTEN REPORT: Many images online are reduced in size, but if you use an image from a good quality camera (and many cell phones now have good cameras), chances are that it will be fairly high resolution. If so, you should reduce the image in size to no larger than 300x300 pixels. You can reduce the size by using spatial scaling as described above and/or selecting a subregion of the original matrix. If you made any changes to the original image, describe these in

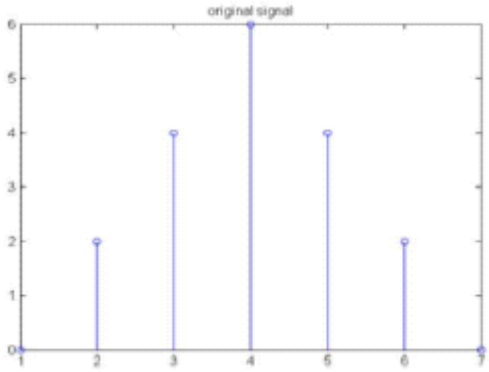
the report. If not, you can simply state that you used the original image as is. If you scale your image, use the technique you expect to work best, and comment on why you chose the one that you used with your image.

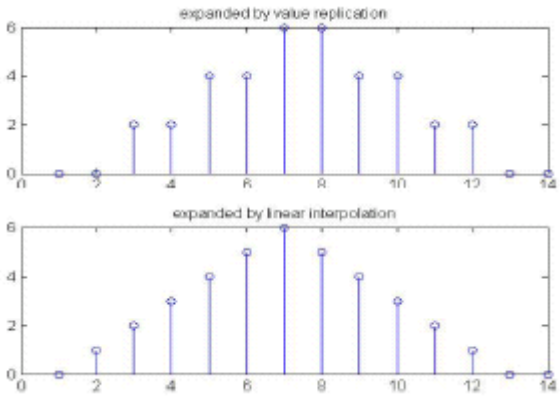
Assignment 4

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. Taking a 1-dimensional signal [0,2,4,6,4,2,0] for example, suppose now we want to expand it 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 tap linear interpolation.

The original signal and the up-scaled signals from the two methods are shown below.





Now, write a MATLAB function that can expand an image using the 2-dimensional version of repetition and interpolation for a factor of 3 expansion. You can directly use interp2( ) provided by MATLAB.

Apply the function to the S=5 results from Assignment 3 for both images. *Comment on whether any aspects of the image are more distorted from the original than others for different methods of scaling*.

IN-LAB CHECK-OFF: Display the following images for the TA.

- Original gray scale image
- Scaled image using sample tripling
- Scaled image using interpolation

WRITTEN REPORT: Repeat scaling with your image, using the technique you expect to work best. Comment on the trade-offs of the two techniques, and why you chose the one that you used with your image.

5. Working with Your Own Image

Assignment 5

Create a 3-image sequence by concatenating the scaled version of the original image plus two versions of your image using blurring and edge detection. You may change the size of the blurring window if you like.

6. Summary of Assignments and Deadlines

- In-lab group demonstrations (due in lab during the sixth week of class)
    - *Assignment 1:* Show the TA the original gray-scale, blurred and edge detection versions of the egg image.
    - *Assignment 2:* Display modified images
    - *Assignment 3:* Display thumbnail images; comment on whether one method for scaling is better than another
    - *Assignment 4:* Display expanded images; comment on whether one method of expansion is better than another
- Written report (Reminder: each person should use their own image.)
    - *Assignment 3:* Describe where you got your image and any modifications you needed to do to make it comply with the size requirements.
    - *Assignment 4:* Include images providing a side-by-side comparison of the starting and 3x expanded versions of your image. State which scaling methods you used and why. Comment on the tradeoffs of the different methods, including types of images that are more sensitive to scaling and/or methods of scaling.
    - *Assignment 5:* Describe details of what you implemented for the two processed images (a mathematical description, not Matlab details which are uploaded separately) and a discussion of observable effects on your image. The report should include the 3-image concatenation that you generated.
- Files to be turned in via Canvas (due the day before you start Lab 4)
    - Written report.
    - Electronic versions of your original image (full size) and the images you generated based on this image.
    - A zip file with all MATLAB code that you used in this lab.