**Module 4 Day3 Course Material**

**Image Processing**

**What Is the Image Processing Toolbox?**

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB® numeric computing environment.

**What Can You Do with the Image Processing Toolbox?**

The toolbox supports a wide range of image processing operations, including:

* Geometric operations
* Neighborhood and block operations
* Linear filtering and filter design
* Transforms
* Image analysis and enhancement
* Binary image operations
* Region of interest operations

Many of the toolbox functions are MATLAB M-files, which contain MATLAB code that implements specialized image processing algorithms. You can view the MATLAB code for these functions using the statement: type function\_name

You can extend the capabilities of the Image Processing Toolbox by writing your own M-files, or by using the toolbox in combination with other toolboxes, such as the Signal Processing Toolbox and the Wavelet Toolbox.

**Some Basic Topics**

Before beginning with this exercise, start MATLAB. You should already have installed the Image Processing Toolbox, which runs seamlessly from MATLAB.

For information about installing the toolbox, see the MATLAB Installation Guide for your platform.

**1. Read and Display an Image**

Clear the MATLAB workspace of any variables and close open figure windows.

clear, close all

To read an image use the imread command. Let’s read in a TIFF image named

pout.tif (which is one of the sample images that is supplied with the Image

Processing Toolbox), and store it in an array named I.

I=imread('pout.tif');

Now call imshow to display I.

imshow(I)



**2. Check the Image in Memory**

Enter the whos command to see how I is stored in memory.

Whos

MATLAB responds with

Name Size Bytes Class

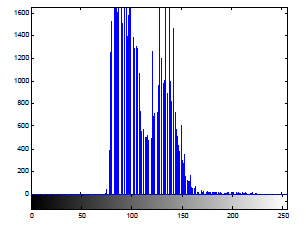
I 291x240 69840 uint8 array

Grand total is 69840 elements using 69840 bytes

**3. Perform Histogram Equalization**

As you can see, pout.tif is a somewhat low contrast image. To see the distribution of intensities in pout.tif in its current state, you can create a histogram by calling the imhist function. (Precede the call to imhist with the figure command so that the histogram does not overwrite the display of the image I in the current figure window.)

figure, imhist(I) % Display a histogram of I in a new figure



Notice how the intensity range is rather narrow. It does not cover the potential range of [0, 255], and is missing the high and low values that would result in good contrast. Now call histeq to spread the intensity values over the full range, thereby improving the contrast of I. Return the modified image in the variable I2.

I2 = histeq(I); % Equalize I and output in new array I2.

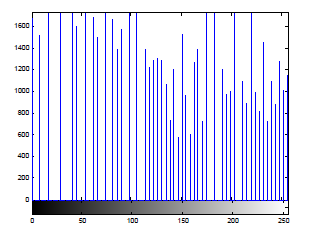
Display the new equalized image, I2, in a new figure window.

figure, imshow(I2) % Display the new equalized image I2.



Call imhist again, this time for I2.

figure, imhist(I2) % Show the histogram for the new image I2.



See how the pixel values now extend across the full range of possible values.

**4. Write the Image**

Write the newly adjusted image I2 back to disk. Let’s say you’d like to save it

as a PNG file. Use imwrite and specify a filename that includes the extension 'png'.

imwrite (I2, 'pout2.png');

**5. Check the Contents of the Newly Written File**

Now, use the imfinfo function to see what was written to disk. Be sure not to end the line with a semicolon so that MATLAB displays the results. Also, be sure to use the same path (if any) as you did for the call to imwrite, above.

imfinfo('pout2.png')

MATLAB responds with

ans =

Filename:'pout2.png'

FileModDate:'03-Jun-1999 15:50:25'

FileSize:36938

Format:'png'

FormatVersion:[]

Width:240

Height:291

BitDepth:8

ColorType:'grayscale'

**Functions by Category**

The tables below list all functions in the Image Processing Toolbox, plus a few functions in MATLAB that are especially useful for image processing. All of the functions listed have reference entries in this User’s Guide, with the following exceptions:

* Most MATLAB functions. To see the reference entries for most of the MATLAB functions listed here, see the MATLAB Function Reference. The MATLAB functions imread, imfinfo, and imwrite have entries in this reference because they are essential to image file I/O.
* The Image Processing Toolbox demo functions and slideshow functions. For information about any of these functions, see “Image Processing Demos” in the Preface.

