|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name:      Id No:** | | | | |
| **Pre-lab Session work (5M)** | **In-Lab Session work (15M)** | **Post Lab session work (5M)** | **Viva**  **(5M)** | **Total Marks**  **30M** |
|  |  |  |  |  |
| **Remarks if any:** | | | | |
| **Date: Signature of the Instructor Marks awarded** | | | | |

**Medical Imaging and Image Processing-Lab (BM3705)**

**Lab1: Image View, Display and Exploration**

Viewing, display and storing images are the fundamentals to image processing. The Image Processing toolbox provides a number of image processing apps to view and explore images. Using the Image viewer app, you can view pixel information, pan and zoom, adjust contrast, and measure distances etc. The Image Processing Toolbox supports a number of image display techniques. For example, the function **imshow** displays any supported image type with a single function call. Other functions handle more specialized display needs. This lab further explores the basic display techniques for each image type supported by the toolbox, as well as how to set the toolbox preferences for the imshow function. It also discusses special display techniques, such as multiple image display.

**Note:** If you are new to MATLAB, you should first read *Getting Started with MATLAB*.

**Objectives:**

To

1. understand various types of images such as intensity image, binary image and storage class.
2. reading and displaying various types of images.
3. check the image memory and contents of images
4. writing / storing the image data.
5. converting data types of images.
6. generate images
7. understand imtool, an open image viewer

Pre-Lab work: All the students are instructed to go through the given reference text books before proceeding to the examples and exercises.

* 1. **Reading an Image:**

Clear the MATLAB workspace of any variables, close open figure windows and clear the command window.

**clear, close all; clc;**

To read an image use the **imread** command. Let’s read in a TIFF image named cameraman.tif (which is one of the sample images that is supplied with the Image Processing Toolbox), and store it in an array named im.

**im = imread(cameraman.tif);**

* 1. Display an image: Now call imshow to display im.

**imshow(im)**

1.3 Check the Image in Memory

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

**whos**

**Name Size Bytes Class**

**im 291x240 69840 uint8 array**

**Grand total is 69840 elements using 69840 bytes**

|  |
| --- |
| **What is happening** |
| **Step 1:** The imread function recognized cameraman.tif as a valid TIFF file and stored it in the variable im. (For the list of graphics formats supported, see imread in the “Function Reference” chapter.).  **Step 2:** The function imshow display graphics images in MATLAB. In general, it is preferable to use imshow for displaying images because it handles the image-related MATLAB properties for you. (The MATLAB function image is for low-level programming tasks.)  Note that if cameraman.tif were an *indexed* image, the appropriate syntax for imread would be,  **[X, map] = imread(‘cameraman.tif');**  (For more information on the supported image types, see “Image Types in the Toolbox”)  **Step 3:** You called the whos command to see how cameraman.tif had been stored into the MATLAB workspace. As you saw, cameraman.tif is stored as a 256-by-256 array. Since cameraman.tif was an 8-bit image, it gets stored in memory as an uint8 array. MATLAB can store images in memory as uint8, uint16, or double arrays. (See “Reading a Graphics Image” for an explanation of when the different storage classes are used.). |

**1.4 Writing Image Data: Imwrite, w**rite image to graphics file

**Syntax**

**imwrite(A, filename, fmt)**

**Example:**

**a=imread(‘cameraman.tif’);**

**imwrite(a, gray(256), 'b.bmp');**

**imshow('b.bmp')% imshow is used to display image**

**imwrite (im, 'cameraman.png');**

|  |
| --- |
| **What is happening** |
| **Step 4:** MATLAB recognized the file extension of 'png' as valid and wrote the image to disk. It wrote it as an 8-bit image by default because it was stored as a uint8 intensity image in memory. If I2 had been an image array of type RGB and class uint8, it would have been written to disk as a 24-bit image. If you want to set the bit depth of your output image, use the BitDepth parameter with imwrite. This example writes a 4-bit PNG file.  **imwrite(im, 'cameraman.png', 'BitDepth', '4');**  Note that all output formats do not support the same set of output bit depths. For example, the toolbox does not support writing 1-bit BMP images. See imwrite in the “Reference” chapter for the list of valid bit depths for each format. See also “Writing a Graphics Image” for a tutorial discussion on writing images using the Image Processing Toolbox. |

**1.5 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'**

**Note** The value in the FileModDate field for your file will be different from what is shown above. It will show the date and time that you used imwrite to create your image. Note also that we truncated the number of field names and values returned by this call.

|  |
| --- |
| **What is happening** |
| **Step 5:** When you called imfinfo, MATLAB displayed all of the header fields for the PNG file format that are supported by the toolbox. You can modify many of these fields by using additional parameters in your call to imwrite. The additional parameters that are available for each file format are listed in tables in the reference entry for imwrite. (See “Querying a Graphics File” for more information about using imfinfo.). |

**1.6: How to get number of rows and columns of image**

Function **size** gives the rows and columns dimension of image

**[r,c]=size(a)**

**r =**

**291**

**c =**

**240**

**1.7 Accessing the Pixel data:** There is a one-to-one correspondence between pixel coordinates and the coordinates MATLAB® uses for matrix subscripting. This correspondence makes the relationship between an image's data matrix and the way the image is displayed easy to understand. For example, the data for the pixel in the fifth row, second column is stored in the matrix element (5, 2). You use normal MATLAB matrix subscripting to access values of individual pixels. For example, the MATLAB code A(2, 15) returns the value of the pixel at row 2, column 15 of the image A.

**1.8 Imtool:** Open Image Viewer app.

The Image Viewer presents an integrated environment for displaying images and performing common image processing tasks. The Image Viewer provides all the image display capabilities of [imshow](file:///C:\\Program%20Files\\MATLAB\\R2018a\\help\\images\\ref\\imshow.html), which optimizes figure, axes, and image object property settings for image display. The Image Viewer also provides access to several tools for navigating and exploring images, such as the Pixel Region tool, Image Information tool, and the Adjust Contrast tool.

**Example:**

**clear; close all; clc;**

**% imtool('board.tif');**

**im = imread('board.tif');**

**% im = imread('cameraman.tif'); % Display a grayscale image.**

**% h = imtool(I,[0 80]); %Display a grayscale image, adjusting the display range.**

**% close(h)**

**imtool(im);**

**im\_Max = max(max(im(:,:,1)))**

**im\_Min = min(min(im(:,:,1)))**

**imtool(im, [im\_Min im\_Max]);**

**imtool(im, [0 120]);**

**1.9 Read and Write 1-Bit Binary Images**

Example: This example shows how to read and write 1-bit binary images.

Check the bit depth of the graphics file containing a binary image, text.png. Note that the file stores the binary image in 1-bit format.

**info = imfinfo('text.png');**

**info.BitDepth**

**ans = 1**

Read the binary image from the file into the workspace. When you read a binary image stored in 1-bit format, imread represents the data in the workspace as a logical array.

**BW = imread('text.png');**

**whos**

**Name Size Bytes Class Attributes**

**BW 256x256 65536 logical**

**ans 1x1 8 double**

**info 1x1 4518 struct**

Write the binary image to a file in 1-bit format. If the file format supports it, imwrite exports a binary image as a 1-bit image, by default. To verify this, use imfinfo to get information about the newly created file and check the BitDepth field. When writing binary files, imwrite sets the ColorType field to grayscale.

**imwrite(BW,'test.tif');**

**info = imfinfo('test.tif');**

**info.BitDepth**

**ans = 1**

**Lab1-Exercise Questions**

**Exercise1:** (a)Read an image of **cameraman.tif**., and display in both unit8 and double. Comment on the results.

(b) Repeat the above of any image of your choice.

(c) Use imtool and display with various range of gray values

**Exercise2:** Consider the following Matlab program:

**clear, close all; clc;**

**im = imread(‘cameraman.tif’);**

**imshow(im)**

1. Identify whether the ‘**im**’ in the Matlab program is intensity or binary image.

(b)Mentionthe intensity range values of the above identified image for the following

classes

(i) uint8 (ii) double (iii) uint16

(c) What results are obtained for the following commands. Discuss in terms of intensity

ranges.

(i) **imshow(I,[low high])** (ii) **imshow(im, [ ])**

**Exercise3:** Compare the function imagesc with the other display functions image and imshow with an example. What should you do to obtain the same results with all the above three functions?

**Exercise4:** Consider the following Matlab program:

**clear; close all; clc;**

**im = im2double(imread('cameraman.tif'));**

**im1 = uint8(255\*im);**

**im2 = im2uint8(im);**

1. Explain, what signifies the outputs im1 and im2
2. Display the outputs im, im1 and im2 using display commands imshow, imagesc and image.
3. Store the images in png, jpg and .mat files

**Exercise5:** Discuss how can you read, display and store a binary image. Explain with an example in Matlab.

**Exercise6:** Consider the following files attached. Read these files and display. Find their memory size.

(i) img1.mat (ii) image2.mat

Further save these files in jpg and png formats.

References:

1. Digital Image Processing using Matlab by Gonzalez
2. Mathswork Image Processing Toolbox documentation