LAB 1

Fall 2010, BESE- 13 & 14

Fundamental Concepts

Objective

The aim of this introductory lab is to introduce you to the basic functions in the Matlab Image Processing Toolbox. By the end of today's lab, you should be able to read images from the disk display them, write them back to the disk and perform conversions between different image classes.

Submission Requirements

You are expected to complete the assigned tasks within the lab session and show them to the lab engineer/instructor. Some of these tasks are for practice purposes only while others (marked as '*Exercise*' or '*Question*') have to be answered in the form of a lab report that you need to prepare. Following guidelines will be helpful to you in carrying out the tasks and preparing the lab report.

Guidelines

- In the exercises, when you are asked to display an image, you have to put the image displayed in your project report. You may either save the image as 'jpeg' (File->Save As) and add it to the report or use the 'Print Screen' command on your keyboard to get a snapshot of the displayed image. This point will become clear to you once you actually carry out the assigned tasks**.**

- Name your reports using the following convention:
  - ***Lab#_Rank_YourFullName***

  - '#' replaces the lab number
  - '*Rank*' replaces Maj/Capt/TC/NC/PC
  - '*YourFullName*' replaces your complete name.

- You need to submit the report even if you have demonstrated the exercises to the lab engineer/instructor or shown them the lab report during the lab session.

Tasks for Today

1. Reading an Image

Images can be read into Matlab environment using the function ***imread***.

```
>> Img = imread('path\filename.ext')
```

You may use the following image[1] to test the function:

```
>> Img = imread('rice.png')
```

The following table shows the commonly used file formats supported by Matlab

Image Format	Description	Recognized Extensions
TIFF	Tagged Image File Format	tif, tiff
JPEG	Joint Photographic Experts Group	jpeg, jpg
BMP	Windows Bitmap	bmp
GIF	Graphics Interchange Format	gif
PNG	Portable Network Graphics	png

2. Image Size

Function ***size*** can be used to determine the dimensions of an image.

```
>> [M N] = size(Img)
```

OR

```
>> [M N B] = size(Img)
```

Where B gives the number of channels in the image. (3 for colored image and 1 for gray image)

The function ***whos*** gives additional information about the array. For example, *whos Img* gives:

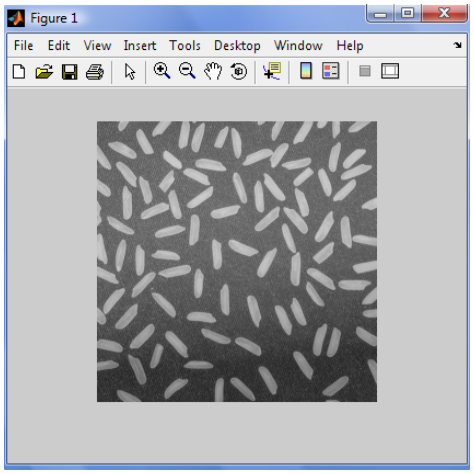
[1] A set of images that comes with Matlab can be accessed directly using the 'name.ext' without the need to specify the complete path. That is why we are able to write: imread('rice.png')

Name	Size	Bytes	Class
Img	256x256	65536	uint8

3. Displaying an Image

Images can be displayed using the function ***imshow***.

```
>> imshow(Img)
```



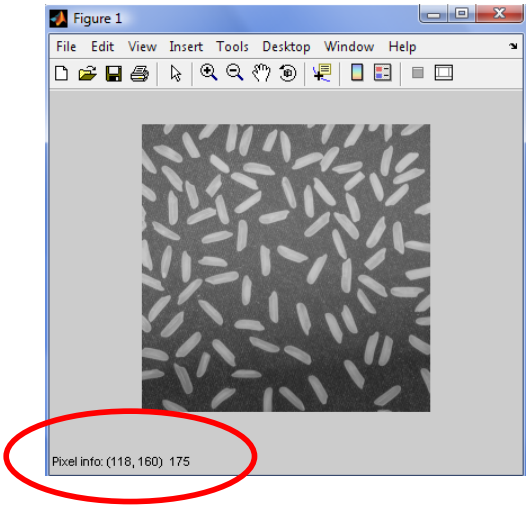
To display another image using imshow while keeping the first image, use:

```
>> imshow(Img), figure, imshow(Img2)
```

To see the pixel values in an image, use:

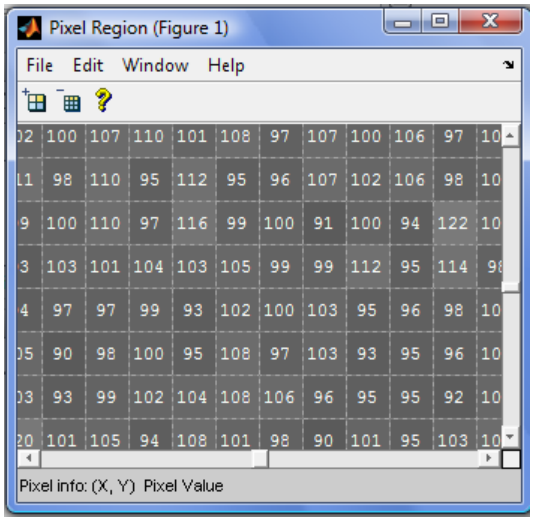
```
>> Figure, imshow(Img)
>> impixelinfo
```

Move the mouse over the image and you will get the pixel coordianates and the respective value at the bottom of the window as indicated in the following figure.



You may also use the command ***impixelregion*** to see the pixel values in an image interactively:

```
>> figure, imshow(Img)
>> impixelregion
```



Exercise 1

Load the image 'cameraman.tif', find its dimensions and number of channels and display it.

4. Writing Images

Images are written to disk using the function ***imwrite*** which has the following basic syntax:

```
>> imwrite (Img, 'filename')
```

```
>> imwrite (Img, 'testImage.tif')
```

The *imwrite* function can have additional parameters depending upon the file format selected. E.g.

```
>> imwrite (Img, 'filename.jpg', 'quality', q)
```

Where *q* is an integer between 0 and 100.

5. Getting Information about Images

The function ***imfinfo*** can be used to get information about a file stored on the disk.

```
>> imfinfo 'rice.png'
Filename: 'C:\Program Files\MATLAB\R2007b\toolbox\images\imdemos\rice.png'
FileModDate: '26-janv.-2003 05:03:06'
FileSize: 44607
Format: 'png'
FormatVersion: []
Width: 256
Height: 256
BitDepth: 8
ColorType: 'grayscale'
```

You may use this function to get the size of the file as:

```
>> K = imfinfo('rice.png')
>> ImageSize = K.FileSize.
```

Exercise 2

Load the image '*cameraman.tif*', and save it in '*jpeg*' format with a compression factor of 25.

Display the two images and compare them visually. What do you observe?

Using the function '*imfinfo*', find the size of files before and after compression and hence compute the compression ratio.

6. Data Classes

Some important data classes in Matlab are:

Name	Description
double	Double precision floating point numbers 8 bytes per element
uint8	Unsigned 8 bit integers [0 255]
uint16	Unsigned 16 bit integers [0 65535]
uint32	Unsigned 32 bit integers
int8	Signed 8 bit integers [-128 127]
int16	Signed 16 bit integers [-32768 32767]
int32	Signed 32 bit integers
single	Single precision floating point numbers 4 bytes per element
char	Characters (2 bytes)
logical	Values 0 or 1

Dealing with images, you will mostly encounter ***uint8*** and ***logical*** data types.

7. Image Types

The Image Processing Toolbox supports four types of images:

- Intensity images
- Binary images
- RGB images
- Indexed images

Intensity Images

Pixel values represent image intensities.

Class uint8 [0 255]

Class double [0 1]

Binary Images

Binary images are logical arrays of 0s and 1s. An array of type ***uint8*** having values 0 and 1 is NOT considered a binary image. An array can be converted to binary using:

```
>> A =
   5   0   1
   2   0   3

>> B = logical (A)
>> B =
   1   0   1
   1   0   1
```

The function ***islogical*** can be used to check if in array is logical or not. The other two types will be discussed at a later stage.

8. Conversion between Data Classes

The general syntax of conversion between data classes is:

```
>> B = data_class_name (A)
```

For example:

```
B= double(A) % Converts the type of A to double
```

> **Exercise 3**
>
> What happens if you convert a double having values out side the range [0 255] to an uint8?

9. Conversion between Image Types and Classes

The toolbox provides a number of functions to convert between image classes and types. Some of these are:

im2uint8, im2double, im2bw, … etc.

For example consider the following 2x2 image of type double which could result from some intermediate calculations:

```
f=[-0.5     0.5
   0.75     1.5 ]
```

Performing the conversion:

>> g = im2uint8(f) yields:

g = [0 128

191 255]

So this function sets:

All values less than 0 to 0.

All values greater than 1 to 255.

And multiplies all other values by 255.

The conversion of an arbitrary array of class double to an array of class double scaled to the range [0 1] can be done via function ***mat2gray***.

>> g= mat2gray(A)

The output values are in the range 0 (black) to 1 (white).

Finally, for conversion between an intensity image and a binary image, the function ***im2bw*** can be used:

>> g=im2bw(f,T)

Produces a binary image *g* from the intensity image *f* by thresholding (will be covered in detail in the lectures). The output image *g* has a value 0 for all pixels in the input image with a value less than T, and 1 for all other pixels. The value of T has to be in the range [0 1] regardless of the type of input image *f*. (If the input image is of type uint8, im2bw will first divide all pixels by 255 and then apply the threshold).

Exercise 4

Load the image '*rice.png*', binarize it using the function 'im2bw' with a threshold of 0.5 and display both original and binarized images.

++++++++++