LAB 2

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. Vector Indexing

An array of dimension *1* x *N* is called a row vector. The elements of such a vector are accessed using one-dimensional indexing.

```
>> V = [1 3 5 7 9]
>> V(2)
```

A row vector is converted to a column vector using the transpose operator (‘)

```
>> W=V'
```

To access blocks of elements by using (:)

```
>> V (1:3)
>> V(2:4)
>> V(3:end)    %end is the last element of vector
>> V(:)        %produces a column vector like v(1:end)
```

Exercise 1

Run and explain the following commands

```
>>    v(1:2:end)
>>    v(end:-2:1)
```

Function ***linspace*** is used to create row vector x of n elements linearly spaced b/w and including a and b.

```
>> x=linspace(a,b,n)
>> x=linspace(1,10,4)
```

A vector can even be used as an index into another vector

```
>> v([1 4 5])
```

2. Matrix Indexing

Matrices can be represented as a sequence of row vector enclosed by square brackets.

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

Creates a 3 x 3 matrix

Select elements in a matrix just as vector but now need two indices.

```
>> A=(2,3)
```

The colon operator is used in matrix indexing to select a two dimensional block of elements out of a matrix

```
>> C3=A(:,3)

>> R2=A(2,:)

>> T2=A(1:2,1:3)

>>B=A          %To create a B like A
```

Exercise 2

Run and explain the following commands

```
>>    B(:,3)=0
>>    A(end, end)
>>    A(end, end-2)
>>    A(2:end, end:-2:1)
```

A particular useful addressing approach using matrices for indexing is of the for A(D), where D is a logical array.

```
>> D=logical([1 0 0; 0 0 1; 0 0 0])

>> A(D)

Ans= 1

     6

>> V=T2(:)

>> s = sum(A(:))        %Same as >> sum(sum(A))
```

The image is a 1024 x 1024 intensity image, f, of class uint8. The image was flipped vertically using the statements

```
>> Img2=imread('cameraman.tif');

>> fp=Img2(end:-1:1,:)

>> imshow(fp)
```

The image is a section out of image

```
>> fc=Img2(57:268,57:268)
>>imshow(fc)
```

Similarly sub sampled image obtained using the statement

```
>> fs=Img2(1:2:end,1:2:end)
>>imshow(fs)
```

> **Exercise 3**
>
> Write the code to read an image and flip it left and right the picture.

3. Standard Arrays

```
>>zeros(M,N)     % generates 0s of class double
>>ones(M,N)      % generates 1s of class double
>>true(M,N)      % generates 1s logical matrix
>>false(M,N)     % generates 0s logical matrix
>>magic(M)       % generates square matrix having equal sum of all rows, columns
                  and diagonals
>>rand(M,N)      % generates a matrix whose entries are uniformaly distributed
                  random numbers in the interval [0 1]
```

4. M-Function Programming

M-Files in Matlab can be script that simply execute a series of statements or they can be function that can accept arguments and can produce one or more outputs. Function file has following components.

- The function definition line
- The function body

Function definition line has the following general form

Function [outputs] = name (inputs)

e.g. >> [s, p] = sumprod(f, g);

Function body is contains all the code that perform computation and assigns values to output argument.

s = f+g

p=f*g

Operators are grouped into three main categories Arithmetic, Relational and Logical.

Operator	***Name***	***Matlab function***
+	Array and matrix addition	plus(A,B)
-	Array and matrix subtraction	minus(A,B)
.*	Array multiplication	times(A,B)
*	Matrix multiplication	mtimes(A,B)
./	Array right division	rdivide(A,B)
.\	Array left division	ldivide(A,B)
/	Matrix right division	mrdivide(A,B)
\	Matrix left division	mldivide(A,B)
.^	Array power	power(A,B)
^	Matrix power	mpower(A,B)
‘	Vector and matrix transpose	transpose(A)
+	Unary plus	uplus(A)
-	Unary minus	uminus(A)

Logical Operator (AND, OR, NOT)

```
>> A=[1 2 0; 0 4 5]
>> B=[1 -2 3; 0 1 1]
>> A & B
>> A | B
>>~A
```

4. Flow Control

The ability to control the flow of operation based on set of predefined conditions is at the heart of all programming language.

Statement	***Description***
if	if together with else and elseif executes a group of statement based on a specified logical condition.

for	Executes a group of statement a fixed number of times
while	Executes a group of statements an indefinite number of times based on a specified logical condition
break	Terminates execution of a for or while loop
continue	Passes control to the next iteration of a for or while loop skipping any remaining statements in the body of the loop
switch	switch, together with case and otherwise, executes different groups of statements, depending on a specified value or string.
return	Causes execution to return to the invoking function

Exercise 4

Write a Function takes an image as an input, converts the image so that pixels only have 2 values 0 and 255. Return the converted image.

Exercise 5

Explore the imresize function.