**COMSATS University Islamabad, Lahore Campus**
**Department of Electrical and Computer Engineering**

# CPE415 –Digital Image Processing

## Lab Manual for SPRING 2020& Onwards

Lab Resource Person

Assad Ali

Theory Resource Person

Moazzam Ali Sahi

Supervised By

Dr. Ikram Ullah Khosa

**Name:** _____ **Registration Number:***CIIT/ - - /LHR*

**Program:** _____ **Batch:** _____

**Semester** _____

## Revision History

| Sr.No. | Update | Date | Performed by |
|---|---|---|---|
| 1 | Lab Manual Preparation | Aug-15 | Engr M. Hassan Aslam |
| 2 | Lab Manual Review | Aug-15 | Dr Ikram Ullah Khosa |
| 3 | Layout modifications | Aug-15 | Engr M. Hassan Aslam |
| 4 | Lab Manual Review | Sep-15 | Dr Ikram Ullah Khosa |
| 5 | Layout modifications | Sep-15 | Engr M. Hassan Aslam |
| 6 | Review and Update | Aug-16 | Engr Rizwan Asif Engr M. Hassan Aslam |
| 7 | Lab Manual Modification according to OBE Requirements. | Feb-18 | Engr Assad Ali |
| 8 | Lab Manual Review | Feb-18 | Dr Ikram Ullah Khosa |
| 9 | Lab Manual Modification | Feb-2020 | Moazzam Ali Sahi |
| 10 | Lab Manual Update | Sep-2021 | Moazzam Ali Sahi |
| 11 | Lab Manual Update | Feb-2022 | Moazzam Ali Sahi and Assad Ali |

# Preface

First, I thank ALLAH Almighty, who gave the strength to complete this lab manual and for His limitless blessings and mercy on me.

Image processing has found applications in many areas from astrophysics, neuroscience, medical imaging to computer graphics. The students, thus need to be introduced to this field, keeping in view the emerging need of more experts in this domain.

This Lab would develop a good understanding about Digital Image Processing as it occurs in various domains. Various Image Transformations would be elaborated. It would help in developing expertise to model, analyze and process Images. Upon successful completion of this lab, students would be able to understand basics of Image Enhancement in spatial and frequency domain, Image restoration, and image segmentation.

The course emphasis on the following topics of Digital Image Processing:

- Introduction: Image presentation and Image processing devices
- Image Fundamentals: Visual perception, Sampling and quantization
- Image Enhancement: Histogram-modification techniques, Smoothing and sharpening.
- Image filtering: Spatial domain filtering, frequency domain filtering (Fourier transform, Homomorphic filtering etc)
- Colour Image Processing
- Image Restoration: Algebraic approach, inverse filtering, and Geometric transformations
- Image Segmentation:

Lastly, I would like to avail this opportunity to express my gratitude to all those who have been a source of guidance, support and advice throughout this lab manual writing. I hope and pray that you will find this manual useful to enhance your skills and expertise in the field of computer networking.

Enjoy 'learning by doing'!!!!!

***M. Hassan Aslam*** ***[August 2015]***
Lecturer,
Department of Electrical Engineering,
COMSATS IIT, Lahore.

## Books

### Textbooks

- Digital Image Processing/3E, R.C. Gonzales, R.E. Woods, Addison-Wesley, 2009.

### Reference Books

- Digital Image Processing, Kenneth R. Castleman, Prentice Hall, 1996
- Digital Image Processing and Applications, I. Pitas, John Wiley, 2000
- Digital Image Processing/3E, William K. Pratt, John Wiley, 2001

### Reference Books for Lab Manual

- Digital Image processing using MATLAB, R.C. Gonzales, R.E. Woods, Addison-Wesley

## Learning Outcomes

#### Theory CLOs

1. **Apply**the concepts of the digital image processing in spatial and frequency domain to understand and analyze image transformation and enhancement. (PLO1-C3)
2. **Analyze**digital images using spatial filtering for image restoration, segmentation and representation. (PLO2-C4)

#### Lab CLOs

3. **Follow**the concepts of digital image processing to reproduce image enhancement, restoration and segmentation using software tools. (PLO5-P3)

## CLOs – PLOs Mapping

| PLO / CLO | PLO1 | PLO2 | PLO5 | PLO10 | Cognitive Domain | Affective Domain | Psychomotor Domain |
|---|---|---|---|---|---|---|---|
| CLO1 | x | | | | C3 | | |
| CLO2 | | x | | | C4 | | |
| Lab CLO1 | | | x | | | | P3 |

## Lab CLOs – Lab Experiment Mapping

| Lab / CLO | Lab 1 | Lab 2 | Lab 3 | Lab 4 | Lab 5 | Lab 6 | Lab 7 | Lab 8 | Lab 9 | Lab 10 | Lab 11 | Lab 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Lab CLO1 | P2 | P2 | P2 | P3 | P3 | P3 | P2 | P3 | P2 | P3 | P3 | P3 |

## Grading Policy

| | |
|---|---|
| Lab Assignments:<br>i. Lab Assignment 1 Marks (Lab marks from Labs 1-3)<br>ii. Lab Assignment 2 Marks (Lab marks from Labs 4-6)<br>iii. Lab Assignment 3 Marks (Lab marks from Labs 7-9)<br>iv. Lab Assignment 4 Marks (Lab marks from Labs 10-12) | 25% |
| Lab Mid Term = 0.5*(Lab Mid Term exam) + 0.5*(average of lab evaluation of Lab 1-6) | 25% |
| Lab Terminal = 0.5*(Lab Terminal exam) +0.375*(average of lab evaluation of Lab 7-12) + 0.125*(average of lab evaluation of Lab 1-6) | 50% |

The minimum pass marks for both lab and theory shall be 50%. Students obtaining less than 50% marks (in either theory or lab, or both) shall be deemed to have failed in the course. The final marks would be computed with 75% weight to theory and 25% to lab final marks.

## Software Recourses

MATLAB

## Lab Instructions

- All labs comprise two parts: Pre-Lab and In-Lab Exercises
- The students should complete and demonstrate each lab task separately for stepwise evaluation (please ensure that course instructor/lab engineer has signed each step after ascertaining its functional verification)
- Only those tasks that completed during the allocated lab time will be credited to the students. Students are however encouraged to practice on their own in spare time for enhancing their skills

## Lab Report Instructions

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

- Lab objectives
- MATLAB codes
- Results (graphs/tables) duly commented and discussed
- Critical analysis/Discussion
- Conclusion

## Safety Instructions

### Laboratory Safety Rules

All students must read and understand the information in this document with regard to laboratory safety and emergency procedures prior to the first laboratory session. **Your personal laboratory safety depends mostly on YOU**. Effort has been made to address situations that may pose a hazard in the lab but the information and instructions provided cannot be considered all-inclusive.

The danger of injury or death from electrical shock, fire, or explosion is present while conducting experiments in this laboratory. To work safely, it is important that you understand the prudent practices necessary to minimize the risks and what to do if there is an accident.

Avoid contact with conductors in energized electrical circuits. Do not touch someone who is being shocked while still in contact with the electrical conductor or you may also be electrocuted. Instead, press the Emergency Disconnect (red button located near the door to the laboratory). This shuts off all power, except the lights.

Make sure your hands are dry. The resistance of dry, unbroken skin is relatively high and thus reduces the risk of shock. Skin that is broken, wet, or damp with sweat has a low resistance. When working with an energized circuit, work with only your right hand, keeping your left hand away from all conductive material. This reduces the likelihood of an accident that results in current passing through your heart.

Be cautious of rings, watches, and necklaces. Skin beneath a ring or watch is damp, lowering the skin resistance. Shoes covering the feet are much safer than sandals.

If the victim isn't breathing, find someone certified in CPR. Be quick! Some of the staff in the Department Office are certified in CPR. If the victim is unconscious or needs an ambulance, contact the Department Office for help or call 1122. If able, the victim should go to the Student Health Services for examination and treatment.

Transistors and other components can become extremely hot and cause severe burns if touched. If resistors or other components on your proto-board catch fire, turn off the power supply and notify the instructor. If electronic instruments catch fire, press the Emergency Disconnect (red button). These small electrical fires extinguish quickly after the power is shut off. Avoid using fire extinguishers on electronic instruments

# Table of Contents

# LAB # 1 :To display different image types and their details using MATLAB.

## Objectives

- To verify the matrix operations using MATLAB
- To display several image types using MATLAB
- To display image details using MATLAB

**Part 1:**

1. How to use MATLAB interface
2. How to use MATLAB Help
3. How to handle Matrices and Vectors in MATLAB

**Part 2:**

4. How to read, show, access and write an image in MATLAB

## Pre-Lab Theory

## 1.1 MATLABTutorial

MATLAB is a commercial "Matrix Laboratory" package which operates as an interactive programming environment. MATLAB program and script files always have filenames ending with ".m"; the programming language is exceptionally straightforward since almost every data object is assumed to be an array. Graphical output is available to supplement numerical results.

## 1.2 How to use **MATLAB help**

Online help is available from the MATLAB prompt (a double arrow), both generally (listing all available commands).In the text that follows, any line that starts with two greater than signs (>>) is used to denote the MATLAB command line (also known as prompt). This is where you enter your commands.

```
>> help
```

a long list of help topics follows, for specific commands.

```
>> help fft
```

a help message on the fft function follows.

## 1.3 Introduction to Vectors in MATLAB

This is the basic introduction to MATLAB. Creation of vectors is included with a few basic operations.

### 1.3.1 Defining a Vector

MATLAB is a software package that makes it easier for you to enter matrices and vectors, and manipulate them. The interface follows a language that is designed to look a lot like the notation use in linear algebra.

Almost all of MATLAB's basic commands revolve around the use of vectors. A vector is defined by placing a sequence of numbers within square braces:

```
>> v = [3 1]

v =

     3     1
```

This creates a row vector which has the label "v". The first entry in the vector is a 3 and the second entry is a 1. Note that MATLAB printed out a copy of the vector after you hit the enter key. If you do not want to print out the result put a semi-colon at the end of the line:

```
>> v = [3 1];

>>
```

If you want to view the vector just type its label:

```
>> v

v =

     3     1
```

Notice, though, that this always creates a row vector. If you want to create a column vector you need to take the transpose of a row vector. The transpose is defined using an apostrophe (''' ")

```
>> v = [3 1 7 -21 5 6]'

v =

     3
     1
     7
   -21
     5
     6
```

A common task is to create a large vector with numbers that fit a repetitive pattern. MATLAB can define a set of numbers with a common increment using colons. For example, to define a vector whose first entry is 1, the second entry is 2, the third is three, up to 8 you enter the following:

```
>> v = [1:8]

v =

     1     2     3     4     5     6     7     8
```

If you wish to use an increment other than one that you have to define the start number, the value of the increment, and the last number. For example, to define a vector that starts with 2 and ends in 4 with steps of .25 you enter the following:

```
>> v = [2:.25:4]

v =

  Columns 1 through 7

    2.0000      2.2500      2.5000      2.7500      3.0000      3.2500
3.5000

  Columns 8 through 9

    3.7500    4.0000
```

Accessing elements within a vector

You can view individual entries in this vector. For example, to view the first entry just type in the following:

```
>> v(1)

ans =

     2
```

This command prints out entry 1 in the vector. Also notice that a new variable called ans has been created. Any time you perform an action that does not include an assignment MATLAB will put the label ans on the result.

## 1.4 Introduction to Matrices in MATLAB

A basic introduction to defining and manipulating matrices is given here. It is assumed that you know the basics on how to define and manipulate **vectors** using MATLAB.

### 1.4.1 Defining Matrices

Defining a matrix is similar to defining a **vector**. To define a matrix, you can treat it like a column of row vectors (note that the spaces are required!).

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

A =

     1     2     3
     3     4     5
     6     7     8
```

You can also treat it like a row of column vectors.

```
>> B = [ [1 2 3]' [2 4 7]' [3 5 8]']

B =

     1     2     3
     2     4     5
     3     7     8
```

If you have been putting in variables through this and the tutorial on **vectors**, then you probably have a lot of variables defined. If you lose track of what variables you have defined, the `whos` command will let you know all of the variables you have in your work space.

```
>>whos
  Name        Size                    Bytes  Class

  A           3x3                        72  double array
  B           3x3                        72  double array
  v           1x5                        40  double array
```

Grand total is 23 elements using 184 bytes

You can work with different parts of a matrix, just as you can with vectors. Again, you have to be careful to make sure that the operation is legal.

```
>> A(1:2,3:4)
???  Index exceeds matrix dimensions.

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

ans =

     2     3
     4     5

>> A(1:2,2:3)'

ans =

     2     4
     3     5
```

Finally, sometimes you would like to clear all of your data and start over. You do this with the "clear" command. Be careful though, it does not ask you for a second opinion and its results are.

```
>> clear
```

## 1.5 Loops

With loop control statements, you can repeatedly execute a block of code, looping back through the block while keeping track of each iteration with an incrementing index variable.

### 1.5.1 For Loop

The `for` loop executes a statement or group of statements a predetermined number of times.

**Syntax**

```
for index = start:increment:end
    statements
end
```

The default increment is 1. You can specify any increment, including a negative one. For positive indices, execution terminates when the value of the index exceeds the end value; for negative increments, it terminates when the index is less than the end value.

**Example**

```
for n = 2:6
    x(n) = 2 * x(n - 1);
end
```

this loop executes five times.You can nest multiple for loops, as follows

```
for m = 1:5
    for n = 1:100
        A(m, n) = 1/(m + n - 1);
    end
end
```

### 1.5.2 While Loop

The while loop executes a statement or group of statements repeatedly as long as the controlling expression is true (1).

**Syntax**

```
while expression
    statements
end
```

If the expression evaluates to a matrix, all its elements must be 1 for execution to continue. To reduce a matrix to a scalar value, use the all and any functions.

For example, this while loop finds the first integer n for which n! (n factorial) is a 100-digit number.

```
n = 1;
while prod(1:n) < 1e100
    n = n + 1;
end
```

## 1.6 Break

The break statement terminates the execution of a `for` loop or `while` loop. When a break statement is encountered, execution continues with the next statement outside of the loop. In nested loops, break exits from the innermost loop only.

## 1.7 Continue

The continue statement passes control to the next iteration of the `for` or `while` loop in which it appears, skipping any remaining statements in the body of the loop. In nested loops, continue passes control to the next iteration of the `for` or `while` loop enclosing it.

## 1.8 Plotting

For Linear 2-D plot

**Syntax**

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,...)
```

**Description**

`plot(Y)` plots the columns of Y versus their index if Y is a real number. If Y is complex, `plot(Y)` is equivalent to `plot(real(Y),imag(Y))`. In all other uses of plot, the imaginary component is ignored.

`plot(X1,Y1,...)` plots all lines defined by Xn versus Yn pairs. If only Xn or Yn is a matrix, the vector is plotted versus the rows or columns of the matrix, depending on whether the vector's row or column dimension matches the matrix.

`plot(X1,Y1,LineSpec,...)` plots all lines defined by the Xn,Yn,LineSpec triples, where `LineSpec` is a line specification that determines line type, marker symbol, and color of the plotted lines.

For Titles, Axis Labels, and Annotations, following commands can be used

```
xlabel('Text')
ylabel('Text')
title('Text')
text(...)
```

## 1.9 Subplot

**Syntax**

```
subplot(m,n,p)
```

## 1.10 Condition Control

This group of control statements enables you to select at run-time which block of code is executed. To make this selection based on whether a condition is true or false, use the `if` statement (which may include `else` or `elseif`). To select from a number of possible options depending on the value of an expression, use the switch and case statements.

### 1.10.1 If Statement

`if` evaluates a logical expression and executes a group of statements based on the value of the expression. In its simplest form, its syntax is

```
 if logical_expression
    statements
end
```

`If` the logical expression is true (1), MATLAB executes all the statements between the if and end lines. It resumes execution at the line following the end statement. If the condition is false (0), MATLAB skips all the statements between the `if` and end lines and resumes execution at the line following the end statement. You can nest any number of `if` statements.

## Digital Image Processing Fundamentals

The basic data structure in MATLAB is the array, an ordered set of real or complex elements. This object is naturally suited to the representation of images, real-valued ordered sets of color or intensity data. MATLAB stores most images as two-dimensional arrays (i.e., matrices), in which each element of the matrix corresponds to a single pixel in the displayed image. (Pixel is derived from picture element and usually denotes a single dot on a computer display.)
For example, an image composed of 200 rows and 300 columns of different colored dots would be stored in MATLAB as a 200-by-300 matrix. Some images, such as RGB, require a three-dimensional array, where the first plane in the third dimension represents the red pixel intensities, the second plane represents the green pixel intensities, and the third plane represents the blue pixel intensities. This convention makes working with images in MATLAB similar to working with any other type of matrix data and makes the full power of MATLAB available for image processing applications.

## In-Lab Tasks

## 1.1 Reading an Image

To import an image from any supported graphics image file format, in any of the supported bit depths, use the `imread` function.

**Syntax**

```
A = imread(filename,fmt)
```

**Description**

```
A = imread(filename,fmt)
```

reads a greyscale or color image from the file specified by the string filename, where the string `fmt` specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname of the location on your system.
Example

```
A = imread('chestxray.jpg');
```

This reads the image from the JPEG file chestxrayinto image array A. Note theuse of single quotes

( ' ) to delimit the string file name. The semicolon at theend of a statement is used by MATLAB for suppressing output. If a semicolonis not included, MATLAB displays on the screen the results of the operation(s)specified in that line. When, as in the preceding command line, no path information is includedin file name, `imread` reads the file from the Current Directory and, if that fails, it tries to find the file in the MATLAB search path. Thesimplest way to read an image from a specified directory is to include a full orrelative path to that directory in file name.

```
A = imread('D:\my images\chestxray.jpg');
```

## 1.2 Display an Image

To display image, use the `imshow` function.

**Syntax**
`imshow(X)`

**Description**
`imshow(A)` displays the image stored in array A.

**Example**
`imshow(A);`

Where A is an image array.
Now, Use the syntax

```
imshow(A,[low high]);
```

It displays as black all values less than or equal to low, and as white all valuesgreater than or equal to high. The values in between are displayed as intermediateintensity values.
Finally, the syntax

```
imshow(A,[]);
```

It sets variable low to the minimum value of array A and high to its maximum value. This form of `imshow` is useful for displaying images that have a lowdynamic range or that have positive and negative values.

## 1.3 Writing Image Data

To write image to graphics file`imwrite` is used

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

**Example**
`imwrite(A,'Test.tif');`

Function `imwrite` writes the image as a TIFF file because it recognizes the ".tif"extension in the filename.Alternatively, the desired format can be specified explicitly with a third input argument.

This syntax is useful when the desired file does not use one oftherecognized file extensions. For example, the following command writes f toa TIFF file called “Test”.

```
imwrite(A,'Test','tif');
```

## 1.4 How to get no. of rows and columns of image

Functionsize gives the rows and columns dimension of image

```
[r,c]=size(A)
r = no of Rows
c = no of columns
```

## 1.5 Sine and cosine waves plotting

Write a MATLAB code to plot sine and cosine waves with proper titles, labels and annotations, also use different markers and colour for both waves.

## 1.6 Fundamentals of image processing

Write a MATLAB code that perform the following operations

1. Ask the user to enter the name of the image file
2. Read the image file
3. Store the file with a different format and name, the name and format should also be chosen by the user.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# LAB # 2 :To display conversion of image types and scaling using MATLAB.

## Objectives

- To convert digital images in several types using MATLAB
- To display binary image using different intensity thresholds using MATLAB

.

## Pre-Lab Theory

Images are most commonly stored in MATLAB using the logical, **uint8**, **uint16** and **double** data types. You can perform many standard MATLAB array manipulations on uint8 and uint16 image data, including basic arithmetic operations, such as addition, subtraction, and multiplication. Indexing, including logical indexing Reshaping, reordering, and concatenating Reading from and writing to MAT-files Using relational operators Certain MATLAB functions, including the find, all, any, conv2, convn, fft2, fftn, and sum functions, accept uint8 or uint16 data but return data in double-precision format. Storage Classes in the Toolbox. By default, MATLAB stores most data in arrays of class double. The data in these arrays is stored as double-precision (64-bit) floating-point numbers. All MATLAB functions work with these arrays. For image processing, however, this data representation is not always ideal. The number of pixels in an image can be very large; for example, a 1000-by-1000 image has a million pixels. Since each pixel is represented by at least one array element, this image would require about 8 megabytes of memory. To reduce memory requirements, MATLAB supports storing image data in arrays as 8-bit or 16-bit unsigned integers, class uint8 and uint16. These arrays require one eighth or one fourth as much memory as double arrays.

## 2.1 Types of Images

1. Indexed images
2. Intensity images
3. Binary images
4. RGB images

### 2.1.1 Indexed Image

An indexed image consists of a data matrix, X, and a colormap matrix, map. The data matrix can be of class uint8, uint16, or double. The colormap matrix is an m-by-3 array of class double containing floating-point values in the range [0,1]. Each row of map specifies the red, green, and blue components of a single color. An indexed image uses direct mapping of pixel values to colormap values. The color of each image pixel is determined by using the corresponding value of X as an index into map. The value 1 points to the first row in map, the value 2 points to the second row, and so on.

A colormap is often stored with an indexed image and is automatically loaded with the image when you use the imread function. However, you are not limited to using the default colormap--you can use any colormap that you choose. The figure below illustrates the structure of an indexed image. The pixels in the image are represented by integers, which are pointers (indices) to color values stored in the colormap.

The following figure depicts an indexed image.

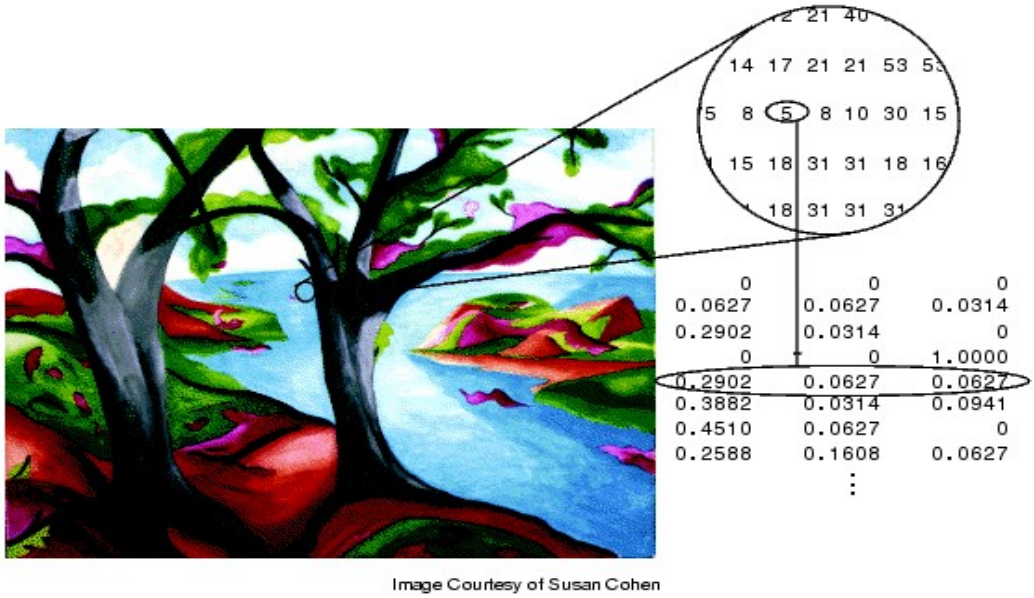


*Figure 2-1Indexed Image*

### 2.1.2 Intensity Image

An intensity image is a data matrix, I, whose values represent intensities within some range. MATLAB stores an intensity image as a single matrix, with each element of the matrix corresponding to one image pixel. The matrix can be of class double, uint8, or uint16.While intensity images are rarely saved with a colormap, MATLAB uses a colormap to display them. The elements in the intensity matrix represent various intensities, or gray levels, where the intensity 0 usually represents black and the intensity 1, 255, or 65535 usually represents full intensity, or white.
The figure below depicts an intensity image of class double.

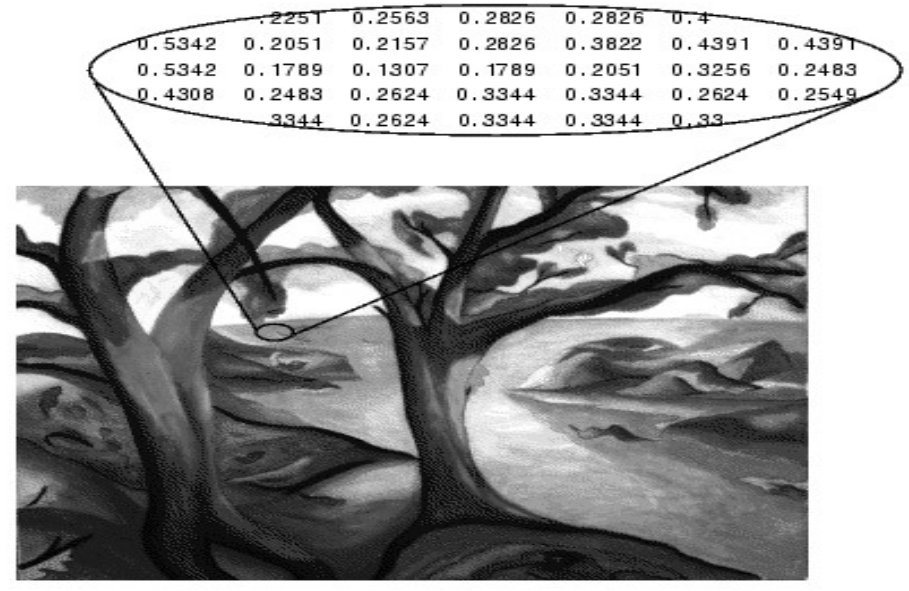


*Figure 2-2 Intensity Image*

### 2.1.3 Binary Image

In a binary image, each pixel assumes one of only two discrete values. Essentially, these two values correspond to on and off. A binary image is stored as a logical array of 0's (off pixels) and 1's (on pixels). The figure below depicts a binary image.

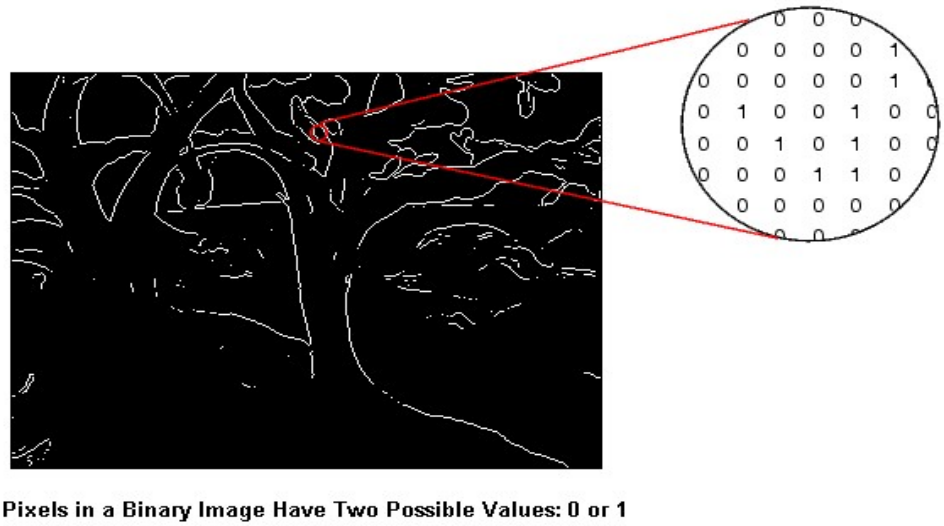


*Figure 2-3 Binary Image*

### 2.1.4 RGB Image

An RGB image, sometimes referred to as a true-color image, is stored in MATLAB as an m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel. RGB images do not use a palette. The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location. Graphics file formats store RGB images as 24-bit images, where the red, green, and blue components are 8 bits each. This yields a potential of 16 million colors. The precision with which a real-life image can be replicated has led to the commonly used term true-color image.

An RGB array can be of class double, uint8, or uint16. In an RGB array of class double, each color component is a value between 0 and 1. A pixel whose color components are (0,0,0) is displayed as black, and a pixel whose color components are (1,1,1) is displayed as white. The three color components for each pixel are stored along the third dimension of the data array.

For example, the red, green, and blue color components of the pixel (10,5) are stored in RGB (10,5,1), RGB (10,5,2), and RGB (10,5,3), respectively. The following figure depicts an RGB image of class double.

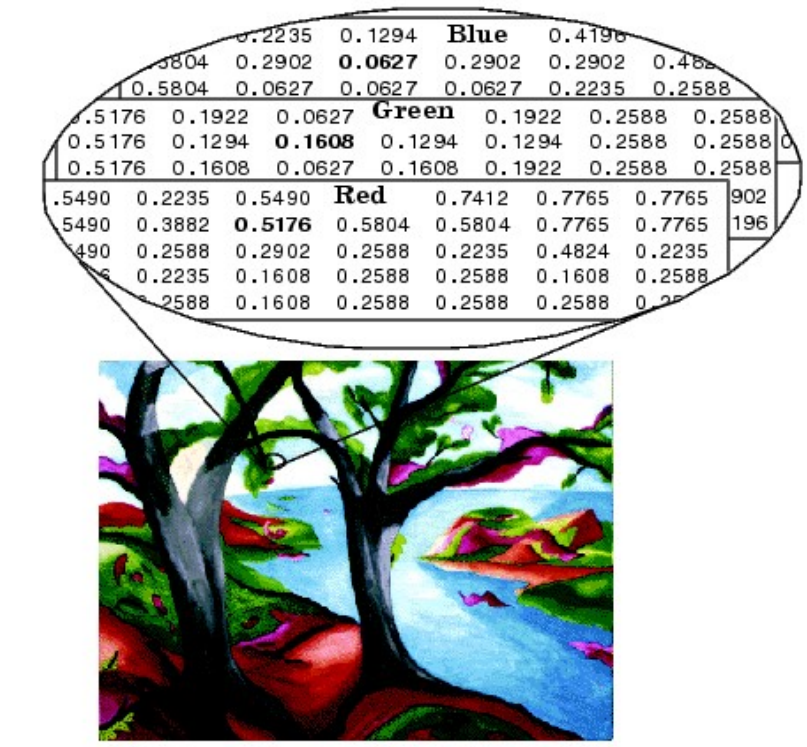


*Figure 2-4 RGB Image*

## In-Lab Tasks

### 2.1 Conversion between different formats

When you store an image, you should store it as a uint8 image since this requires far less memory than double. When you are processing an image (that is performing mathematical operations on an image) you should convert it into a double. Converting back and forth between these classes is easy.

```
I=im2double(I);
```

It converts an image named I from uint8 to double.

```
I=im2uint8(I);
```

It converts an image named I from double to uint8.

There are some other commands to convert the format, use the following commands and discuss the outputs.

1. dither();
2. grayslice();
3. gray2ind();
4. ind2gray();
5. rgb2ind();
6. ind2rgb();
7. rgb2gray();

## 2.2 Thresholding or gray to binary conversion

Threshold is simple concept of setting range of certain value to be a value. The basic purpose of thresholding in image processing is to adjust the pixel value of an image to certain value. Write a MATLAB code to set the pixel value to 0 if the original pixel value is below or equal to 128. Otherwise, if the original pixel value is above 128, the new pixel value will be 255.

```
a=imread('Image_name');
imshow(a);
title('Original Image');

%Write your code here

figure;
imshow(a);
title('Updated Image');
```

## 2.3 Generate mirror image

There are many ways to generate mirror image, in this task we will generate it using nested loops
Try implementing the following steps to generate mirror image

1. Read the image
2. Start the outer loop from 1 to the total rows of the image
3. Start the inner loop from 1 to the total columns of the image
4. In inner loop, write the code to swap the values as shown below

| | | | | |
|---|---|---|---|---|
| $a_{11}$ | $a_{12}$ | $a_{13}$ | $a_{14}$ | $a_{15}$ |
| $a_{21}$ | $a_{22}$ | $a_{23}$ | $a_{24}$ | $a_{25}$ |
| $a_{31}$ | $a_{32}$ | $a_{33}$ | $a_{34}$ | $a_{35}$ |
| $a_{41}$ | $a_{42}$ | $a_{43}$ | $a_{44}$ | $a_{45}$ |
| $a_{51}$ | $a_{52}$ | $a_{53}$ | $a_{54}$ | $a_{55}$ |

*Table 1 Input Image*

| | | | | |
|---|---|---|---|---|
| $a_{15}$ | $a_{14}$ | $a_{13}$ | $a_{12}$ | $a_{11}$ |
| $a_{25}$ | $a_{24}$ | $a_{23}$ | $a_{22}$ | $a_{21}$ |
| $a_{35}$ | $a_{34}$ | $a_{33}$ | $a_{32}$ | $a_{31}$ |
| $a_{45}$ | $a_{44}$ | $a_{43}$ | $a_{42}$ | $a_{41}$ |
| $a_{55}$ | $a_{54}$ | $a_{53}$ | $a_{52}$ | $a_{51}$ |

*Table 2 Mirror Image*

## 2.4 How to generate flipped image

Write a MATLAB code that perform the following tasks without using built-in commands

1. Reads a colored test image provided by the lab instructor during the lab.
2. Convert it into gray scale image
3. Generates the flipped image of original image.
4. Store the flipped image with the name provided by the user.

To display conversion of image types and scaling using MATLAB.

| | | | | |
|---|---|---|---|---|
| $a_{11}$ | $a_{12}$ | $a_{13}$ | $a_{14}$ | $a_{15}$ |
| $a_{21}$ | $a_{22}$ | $a_{23}$ | $a_{24}$ | $a_{25}$ |
| $a_{31}$ | $a_{32}$ | $a_{33}$ | $a_{34}$ | $a_{35}$ |
| $a_{41}$ | $a_{42}$ | $a_{43}$ | $a_{44}$ | $a_{45}$ |
| $a_{51}$ | $a_{52}$ | $a_{53}$ | $a_{54}$ | $a_{55}$ |

*Table 1 Input Image*

| | | | | |
|---|---|---|---|---|
| $a_{51}$ | $a_{52}$ | $a_{53}$ | $a_{54}$ | $a_{55}$ |
| $a_{41}$ | $a_{42}$ | $a_{43}$ | $a_{44}$ | $a_{45}$ |
| $a_{31}$ | $a_{32}$ | $a_{33}$ | $a_{34}$ | $a_{35}$ |
| $a_{21}$ | $a_{22}$ | $a_{23}$ | $a_{24}$ | $a_{25}$ |
| $a_{11}$ | $a_{12}$ | $a_{13}$ | $a_{14}$ | $a_{15}$ |

*Table 3 Flipped Image*

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# LAB # 3 : To show the result of arithmetic and logical operations on images using MATLAB

- To display digital images by performing arithmetic operations using MATLAB.
- To display digital images by performing logical operations using MATLAB

## 3.1 Arithmetic Operations on Images

### 3.1.1 Image Addition and Subtraction

If there are two images $I_1$ and $I_2$ then addition of image can be given by:

$$I(x,y) = I_1(x,y) + I_2(x,y)$$

Where $I(x,y)$ is resultant image due to addition of two images $x$ and $y$ are coordinates of image.Image addition is pixel to pixel. Value of pixel should not cross maximum allowed value that is 255for 8-bit grey scale image. When it exceeds value 255, it should be clipped to 255. To increase over all brightness of the image, we can add some constant value depending on brightness required. In example program we will add value 50 to the image and compare brightness of original and modified image.

$$I(x,y) = I_1(x,y) - I_2(x,y)$$

To decrease brightness of image, we can subtract constant value. In example program, value 100 is subtracted for every pixel. Care should be taken that pixel value should not less than 0 (minus value).Subtract operation can be used to obtain complement (negative) image in which every pixel is subtracted from maximum value i.e. 255 for 8-bit greyscale image.

### 3.1.2Image Multiplication and Division

Multiplication operation can be used to mask the image for obtaining region of interest. Black and white mask (binary image) containing 1s and 0s is multiplied with image to get resultant image. To obtain binary image function `im2bw()` can be used. Multiplication operation can also be used to increase brightness of the image.

Division operation results into floating point numbers so data type required is floating pointnumbers. It can be converted into integer data type while storing the image. Division can be usedto decrease the brightness of the image. It can also use to detect change in image.

## 3.2 Logical Operations on Images

Bitwise logical operations can be performed between pixels of one or more than oneimage.

#### 3.2.1.1 AND/NAND logic operation

AND/NAND Logical operations can be used for following applications:

a. Compute intersection of the images
b. Design of filter masks

c. Slicing of gray scale images

#### 3.2.1.2 OR/NOR logic operation

OR/NOR logical operations can be used for following applications:

a. Merging of two images

#### 3.2.1.3 XOR/XNOR logic operation

XOR/XNOR operations can be used for following applications:

a. To detect change in gray level in the image
b. Check similarity of two images

#### 3.2.1.4 NOT logic operation

NOT operation is used for:

a. To obtain negative image
b. Making some features clear

## Write a MATLAB code that perform the following tasks

1. Read any grayscale image.
2. Perform NOT logical operation.
3. Perform arithmetic operations (Add and Subtract).
4. Store the results as an image file and give your critical analysis about the results.

## Brightness changes using arithmetic operations

Write a MATLAB code which can increaseor decrease the brightness of the image according to the user input.

## Logical Operations

Write a MATLAB code, which perform the following tasks:

1. Read the image provided by the lab instructor.
2. Apply different logical operations (AND, NAND, NOR and XOR).

## Watermarking

Write a MATLAB code, which perform the following tasks:

3. Read the image provided by the lab instructor.
4. Watermark the COMSATS university logo on the provided image.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 4 To follow intensity transformation techniques for digital image enhancement using MATLAB.

## 4.1 Objectives

- To display different intensity transformed versions of digital images using MATLAB.

## 4.2 Pre-Lab Theory

Image enhancement is a very basic image processing task that defines us to have a better subjective judgement over the images. And Image Enhancement in spatial domain (that is, performing operations directly on pixel values) is the very simplistic approach. Enhanced images provide better contrast of the details that images contain. Image enhancement is applied in every field where images are ought to be understood and analyzed. For example, Medical Image Analysis, Analysis of images from satellites, etc.

Image enhancement simply means, transforming an image $f$ into image $g$ using $T$. Where $T$ is the transformation. The values of pixels in images $f$ and $g$ are denoted by $r$ and $s$, respectively. As said, the pixel values *r* and *s* are related by the expression,

$$s = T(r)$$

where $T$ is a transformation that maps a pixel value $r$ into a pixel value $s$. The results of this transformation are mapped into the grey sclale range as we are dealing here only with grey scale digital images. So, the results are mapped back into the range $[0, L-1]$, where $L = 2k, k$ being the number of bits in the image being considered. So, for instance, for an 8-bit image the range of pixel values will be [0, 255].

There are three basic types of functions (transformations) that are used frequently in image enhancement. They are,

- Linear.
- Logarithmic.
- Power-Law.

The transformation map plot shown below depicts various curves that fall into the above three types of enhancement techniques.

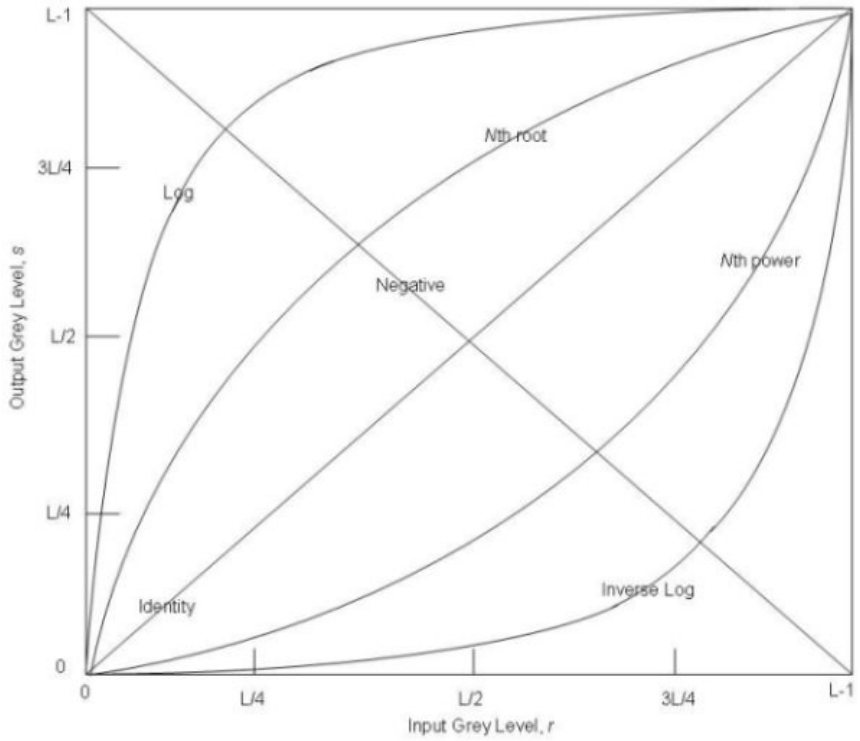


*Figure 4-1 Trnasformations graph*

The Identity and Negative curves fall under the category of linear functions. Identity curve simply indicates that input image is equal to the output image. The Log and Inverse-Log curves fall under the category of Logarithmic functions and *n*th root and *n*th power transformations fall under the category of Power-Law functions.

### 4.2.1 Image Negation

The negative of an image with grey levels in the $[0, L-1]$ is obtained by the negative transformation shown in figure above, which is given by the expression,

$$s = L - 1 - r$$

This expression results in reversing of the grey level intensities of the image thereby producing a negative like image. The output of this function can be directly mapped into the grey scale look-up table consisting values from 0 to $L-1$.

### 4.2.2 Log Transformations

The log transformation curve shown in above figure, is given by the expression,

$$s = c\,log(1 + r)$$

Where $c$ is a constant and it is assumed that $r \geq 0$. The shape of the log curve in fig. A tells that this transformation maps a narrow range of low-level grey scale intensities into a wider range of output values. And similarly maps the wide range of high-level grey scale intensities into a narrow range of high-level output values. The opposite of this applies for inverse-log transform. This transform is used to expand values of dark pixels and compress values of bright pixels.

### 4.2.3 Power-Law Transformations

The *n*th power and *n*th root curves shown in fig. A can be given by the expression,

$$s = cr^{\gamma}$$

This transformation function is also called as *gamma* correction. For various values of $\gamma$ different levels of enhancements can be obtained. This technique is quite commonly called as *Gamma Correction*. If you notice, different display monitors display images at different intensities and clarity. That means, every monitor has built-in gamma correction in it with certain gamma ranges and so a good monitor automatically corrects all the images displayed on it for the best contrast to give user the best experience.

The difference between the log-transformation function and the power-law functions is that using the power-law function a family of possible transformation curves can be obtained just by varying the $\lambda$.

These are the three basic image enhancement functions for grey scale images that can be applied easily for any type of image for better contrast and highlighting. Using the image negation formula given above, it is not necessary for the results to be mapped into the grey scale range [0, $L-1$]. Output of $L-1-r$ automatically falls in the range of [0, $L-1$]. But for the Log and Power-Law transformations resulting values are often quite distinctive, depending upon control parameters like λ and logarithmic scales. So the results of these values should be mapped back to the grey scale range to get a meaningful output image. For example, Log function $s = c\,log(1+r)$ results in 0 and 2.41 for $r$ varying between 0 and 255, keeping $c = 1$. So, the range [0, 2.41] should be mapped to $[0, L-1]$ for getting a meaningful image.

## Pre-Lab Task

Write a MATLAB code to implement the linear transformation on any test image and also write a brief analysis in the light of Figure 4-1, later show the results to lab instructor for verification.

## 4.3 In-Lab Task

### 3.1 Log and antilog Transformation

The following program apply log transformation on the image passed to it.

```
f=imread('Image_name');
g=rgb2gray(f);
c=input('Enter the constant value, c = ');
[M,N]=size(g);
        for x = 1:M
            for y = 1:N
                m=double(g(x,y));
                z(x,y)=c.*log10(1+m);
            end
        end
imshow(f);
figure;
imshow(z);
```

### 4.3.1 Power Law Transformation

The following program apply Power Law transformation on the image passed to it.

```
image=imread('Image_name');
imshow(image);
image_double=im2double(image);
[r c]=size(image_double);
c=input(' Enter the constant value, c = ');
gamma=input('Enter the gamma value, gamma =');
for i=1:r
    for j=1:c
imout(i,j)=c*power(image_double(i,j),gamma);
    end
end
figure;
imshow(imout);
```

### 4.3.2 Linear and negative Transformation

Write a MATLAB code to implement the flowing tasks

1. Read an image provided by the lab instructor.
2. Apply linear transformation.
3. Apply negative transformation.
4. Compare the results.

### 4.3.3 Write a MATLAB code to implement the following transformations on the image provided by the lab instructor and write a brief observation related to each transformation.

1. Linear Transformations
2. Logarithmic transformations
3. Power transformation

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 5 To follow histogram equalization technique for digital image enhancement using MATLAB

## 5.1 Objectives

- To display image histogram using MATLAB.
- To manipulate image histogram for equalization using MATLAB.
- To manipulate image for desired histogram matching using MATLAB

## 5.2 Pre-Lab Theory

Intensitytransformation functions based on information extracted from image, intensity histograms play a central role in image processing, in areas such as enhancement, compression, segmentation, and description. Histogram is a bar-graph used to profile the occurrence of each gray level in the image. Mathematically it is represented as

$$h(r_k) = n_k$$

Where $r_k$ is k$^{th}$grey level and $n_k$ is number of pixel having that Grey level. Histogram can tell us whether image was scanned properly or not. It gives us idea about tonal distribution in the image. Histogram equalization can be applied to improve appearance of the image. Histogram also tells us about objects in the image. Object in an image have similar gray levels so histogram helps us to select threshold value for object detection. Histogram can also be used for image segmentation.

Assume for a moment that intensity levels are continuous quantities normalizedto the range [0, 1] and let $P_r$ denote the probability density function(PDF) of the intensity levels in a given image, where the subscript is used fordifferentiating between the PDFs of the input and output images. Suppose thatwe perform the following transformation on the input levels to obtain output(processed) intensity levels "$S$", where w is the dummy variable for integration.

$$s = T_r = \int_0^r P_r(w)dw$$

The preceding transformation generates an image whose intensitylevels are equally likely, and, in addition, cover the entire range [0, 1]. Thenet result of this intensity-level equalization process is an image with increaseddynamic range, which will tend to have higher contrast. Note that the transformationfunction is really nothing more than the cumulative distribution function(CDF).

Histogram equalization produces a transformation function that is adaptive, inthe sense that it is based on the histogram of a given image. However, once thetransformation function for an image has been computed, it does not changeunless the histogram of the image changes. As noted in the previous section,histogram equalization achieves enhancement by spreading the levels of theinput image over a wider range of the intensity scale. We show in this sectionthat this does not always lead to a successful result. In particular, it is useful insome applications to be able to specify the shape of the histogram that we wishthe processed image to have. The method used to generate an image that has aspecified histogram is called histogram matching or histogram specification.Histogram matching is a method in image processing of colour adjustment of two images using the image histograms.

## 5.3 Pre-Lab Task

Histograms are generally used to find the frequency of intensity levels of an image to perform further image enhancing operations. To display a histogram of image data, following MATLAB command is used.

**Syntax**

```
imhist(I,n)
imhist(X,map)
[counts,x] = imhist(...)
```

Now you are required to the use the `imhist()` command and perform histogram formation of any test image, Also briefly explain that what type of information can be extracted from the output.

Read about Histogram, how it is formed, histogram equalization and histogram matching before coming for lab.

## 5.4 In Lab Tasks

### 5.4.1 Histogram formation

Write a MATLAB code that show the histogram of the image passed to it, without using the build in command `imhist()`.

### 5.4.2 Histogram Equalization

Write a MATLAB code that apply histogram equalization on the image passed to it, without using the built-in command.

### 5.4.3 Histogram matching

Write a MATLAB code that perform the histogram matching of the image passed to it.

### 5.4.4 Histogram Equalization

Write a MATLAB code to implement the histogram equalization on an image provided by the lab instructor, also give your analysis about the image before and after histogram equalization.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 6 To follow linear and non linear spatial filtering techniques for digital image enhancement using MATLAB

## 6.1 Objectives

- To observe the operation of smoothing filters on noisy images using MATLAB
- To observe the operation of sharpening filters on noisy images using MATLAB
- To observe the operation of order statistics filters on noisy images using MATLAB

## 6.2 Pre-Lab Theory

Some neighborhood operations work with the valuesof the image pixels in the neighborhood and the corresponding values of asubimage that has the same dimensions as the neighborhood.The subimage iscalled a *filter, mask, kernel, template, or window,* with the first three terms beingthemost prevalent terminology.The values in a filter subimage are referred toas coefficients, rather than pixels
The idea is to move a *mask*: a rectangle (usually with sides of odd length) or other shape over the given image. As we do this, we create a new image whose pixels have grey values calculated from the grey values under the mask, as shown in Figure 6.1.

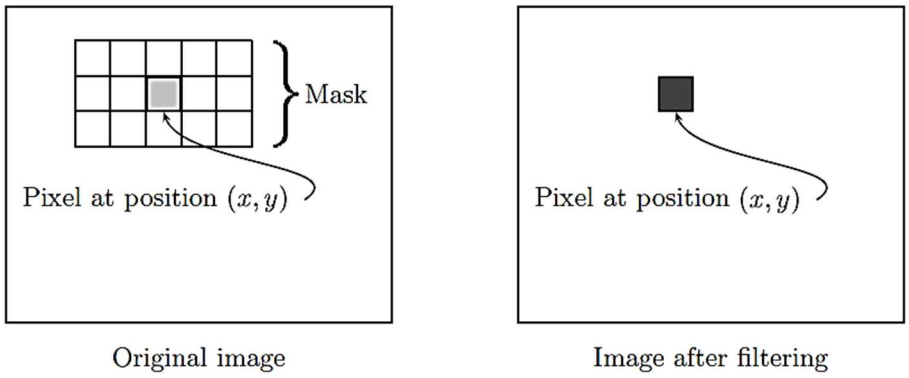


*Figure 6-1 Using a spatial mask on an image*

The combination of mask and function is called a filter. If the function by which the new grey value is calculated is a linear function of allthe grey values in the mask, then the filter is called a linear filter.
We can implement a linear filter by multiplying all elements in the mask by correspondingelements in the neighborhoodand adding up all these products. Suppose we have a 3x5 maskas illustrated in figure 6.1. Suppose that the mask values are given by:

| | | | | |
|---|---|---|---|---|
| $m(-1,-2)$ | $m(-1,-1)$ | $m(-1,0)$ | $m(-1,1)$ | $m(-1,2)$ |
| $m(0,-2)$ | $m(0,-1)$ | $m(0,0)$ | $m(0,1)$ | $m(0,2)$ |
| $m(1,-2)$ | $m(1,-1)$ | $m(1,0)$ | $m(1,1)$ | $m(1,2)$ |

and that corresponding pixel values are

| | | | | |
|---|---|---|---|---|
| $p(i-1,j-2)$ | $p(i-1,j-1)$ | $p(i-1,j)$ | $p(i-1,j+1)$ | $p(i-1,j+2)$ |
| $p(i,j-2)$ | $p(i,j-1)$ | $p(i,j)$ | $p(i,j+1)$ | $p(i,j+2)$ |
| $p(i+1,j-2)$ | $p(i+1,j-1)$ | $p(i+1,j)$ | $p(i+1,j+1)$ | $p(i+1,j+2)$ |

We now multiply and add:

$$\sum_{s=-1}^{1}\sum_{t=-2}^{2} m(s,t)p(i+s,j+t).$$

We see that spatial filtering requires three steps:

1) position the mask over the current pixel,
2) form all products of filter elements with the corresponding elements of the neighborhood,
3) add up all the products.

This must be repeated for every pixel in the image
We may describe this operation as follows:

| | | | |
|---|---|---|---|
| | | | |
| $a$ | $b$ | $c$ | |
| $d$ | $e$ | $f$ | |
| $g$ | $h$ | $i$ | |
| | | | |

$$\longrightarrow \frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

where ***e***is grey value of the current pixel in the original image, and the average is the grey value of the corresponding pixel in the new image.
It is convenient to describe a linear filter simply in terms of the coefficients of all the grey values of pixels within the mask. This can be written as a matrix.The averaging filter, for example, could have its output written as

$$\frac{1}{9}a+\frac{1}{9}b+\frac{1}{9}c+\frac{1}{9}d+\frac{1}{9}e+\frac{1}{9}f+\frac{1}{9}g+\frac{1}{9}h+\frac{1}{9}i$$

and so we can describe this filter by the matrix

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

## 6.2.1 Smoothing Filters

### 6.2.1.1 Mean Filter

Mean filtering is also named as Smoothing, Averaging or Box filtering**.** Mean filtering is a simple, intuitive and easy to implement method of smoothing images, i.e. reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images.

$$\left(\frac{1}{9}\right)\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \text{ or } (1/16)\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

## 6.2.2 Sharpening Filters

The principal objective of sharpening is to highlight fine detail in an image orto enhance detail that has been blurred, either in error or as a natural effect ofa particular method of image acquisition. Uses of image sharpening vary and includeapplications ranging from electronic printing and medical imaging to industrialinspection and autonomous guidance in military systems.

### 6.2.2.1 Laplacian Filter

Laplacian filter is also known as Laplacian of Gaussian (LoG). The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection (see zero crossing edge detectors). The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian smoothing filter in order to reduce its sensitivity to noise, and hence the two variants will be described together here. The operator normally takes a single gray-level image as input and produces another gray-level image as output.The Laplacian $L(x,y)$ of an image with pixel intensity values $I(x,y)$ is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | −4 | 1 | 1 | −8 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | −1 | 0 | −1 | −1 | −1 |
| −1 | 4 | −1 | −1 | 8 | −1 |
| 0 | −1 | 0 | −1 | −1 | −1 |

*Figure 6-2Filter mask used to implement the digital Laplacian*

### 6.2.3 Order-Statistics Filters

Order-statistics filters are nonlinear spatial filters whose response is based onordering (ranking) the pixels contained in the image area encompassed bythe filter, and then replacing the value of thecentrepixel with the value determinedby the ranking result.

#### 6.2.3.1 Median Filter

Median filtering is also known as rank filtering. The median filter is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image.Like the mean filter, the median filter considers each pixel in the image in turn and looks at its nearby neighbours to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the mean of neighbouring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighbourhood under consideration contains an even number of pixels, the average of the two middle pixel values is used).

## 6.3

## 6.4 Pre-Lab Task

## 3.1 Average filtering

The following program implement the average filtering on the image passed to it.

a.

```
img = imread('Image_name');
imgd = im2double(img);
f = ones(3,3)/9;
img1 = filter2(f, imgd);
subplot(121);
imshow(img);
subplot(122);
imshow(img1);
```

b.

```
img = imread(Image_name');
imgd = im2double(img);   % imgd in [0,1]
imgd = imnoise(imgd,'salt & pepper',0.02);
f = ones(3,3)/9;
img1 = filter2(f, imgd);
subplot(121);
imshow(imgd);
subplot(122);
imshow(img1);
```

### 6.4.1 Median filtering

The following program apply median filtering on the image passed to it.

```
I = imread('Image_name');
J = imnoise(I,'salt& pepper',0.02);
K = medfilt2(J);
subplot(121);
imshow(J);
subplot(122);
imshow(K);
```

| Value | Description |
|---|---|
| `Average` | Averaging filter |
| `Disk` | Circular averaging filter (pillbox) |
| `Gaussian` | Gaussian lowpass filter |
| `Laplacian` | Laplacian of Gaussian filter |
| `Motion` | Approximates the linear motion of a camera |
| `Prewitt` | Prewitt horizontal edge-emphasizing filter |
| `Sobel` | Sobel horizontal edge-emphasizing filter |

## 6.5 In-Lab Tasks

### 3.2 Average Filter

Write a MATLAB code to implement the average filter according to the details and mathematical form mentioned above.

### 3.3 Weighted average filtering

Write a MATLAB code to implement the weighted averaging filter on an image provided by the lab instructor without using the built-in commands, also give your analysis about the image before and after applying the filter.

### 6.5.1 Laplacian filtering

Write a MATLAB code to implement the Laplacian filtering on an image provided by the lab instructor without using the built-in commands, then add the original image in the filtered image and give your analysis about the resultant image.

To follow linear and non linear spatial filtering techniques for digital image enhancement using MATLAB

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 7 To display digital image in frequency domain using MATLAB

## 7.1 Objectives

- To manipulate digital image between spatial and frequency domain using MATLAB
- To display magnitude and angle of digital image in frequency domain using MATLAB

## 7.2 Pre-Lab Theory

The general idea is that the image ($f(x,y)$ of size M x N) will be represented in the frequency domain ($F(u,v)$). The equation for the two-dimensional discrete Fourier transform (DFT) is:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

The concept behind the Fourier transform is that any waveform that can be constructed using a sum of sine and cosine waves of different frequencies. The exponential in the above formula can be expanded into sines and cosines with the variables *u* and *v* determining these frequencies.The inverse of the above discrete Fourier transform is given by the following equation:

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

Thus, if we have $F(u,v)$, we can obtain the corresponding image ($f(x,y)$) using the inverse, discrete Fourier transform.

MATLAB has three functions to compute the DFT:

1. `fft` -for one dimension (useful for audio)
2. `fft2` -for two dimensions (useful for images)
3. `fftn` -for n dimensions

MATLAB has three functions that compute the inverse DFT:

1. `ifft`
2. `ifft2`
3. `ifftn`

The following convolution theorem shows an interesting relationship between the spatial domain and frequency domain.

$$f(x,y) * h(x,y) \leftrightarrow H(u,v)F(u,v)$$

and, conversely,

$$f(x,y)h(x,y) \leftrightarrow H(u,v) * G(u,v)$$

The symbol "*" indicates convolution of the two functions. The important thing to extract out of this is that the multiplication of two Fourier transforms corresponds to the convolution of the associated functions in the spatial domain.

## 7.3 Pre-Lab Task

Write a MATLAB code capable of finding the Fourier transform of the image ("`cameraman.tif`"). Required output should be identical to Figure 7-1.

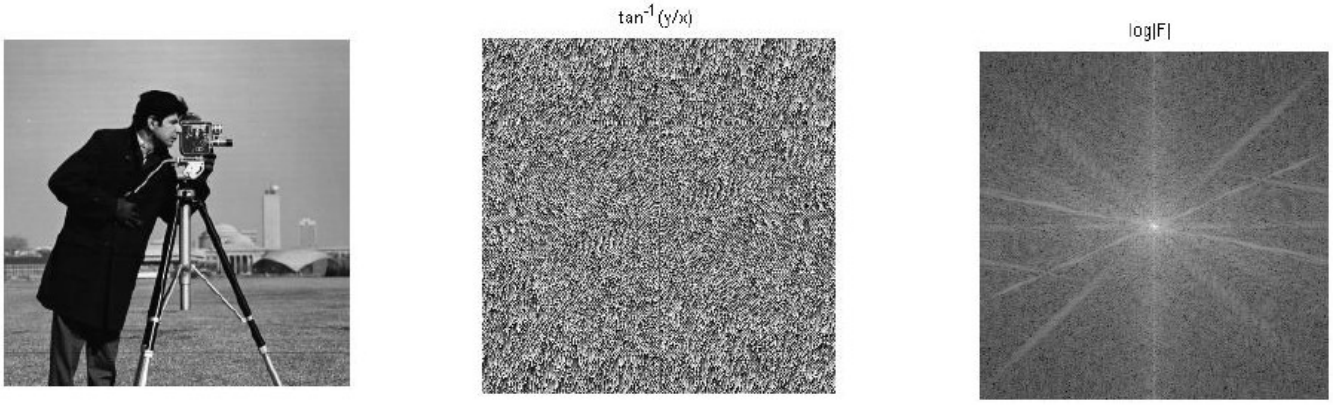


*Figure 7-1 Frequency domain analysis*

## 7.4 In-Lab Task

### 7.4.1 Inverse Fourier transform

Write a MATLAB code to find inverse Fourier transform of an image.

### 7.4.2 Frequency domain analysis

Write a MATLAB code to perform the following task using mathematical formula

1. Read a grayscale image provided by the lab instructor.
2. Plot magnitude response of the image.
3. Plot phase response of the image.
4. Write a brief analysis of the magnitude and phase response of the provided image.

### 7.4.3 Fourier Transform of Image

1. Write a MATLAB code to perform the following task using mathematical formula
    a. Read any grayscale image
    b. Apply Fourier transform
    c. Show the results
    d. Apply inverse Fourier transform
    e. Show the result

Note: Image after inverse Fourier transform should be entirely identical with the input image.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 8 To follow frequency domain filtering techniques for digital image enhancement using MATLAB

## 8.1 Objectives

- To display image filtering by applying low-pass filter in frequency domain using MATLAB
- To display image filtering by applying high-pass filter in frequency domain using MATLAB

## 8.2 Pre-Lab Theory

### 8.2.1 Lowpass and HighPass Frequency Domain Filters

Based on the property that multiplying the FFT of two functions from the spatial domain produces the convolution of those functions, you can use Fourier transforms as a fast convolution on large images. As a note, on small images, it is faster to work in the spatial domain. However, you can also create filters directly in the frequency domain. There are two commonly discussed filters in the frequency domain.

1. Lowpass filters, sometimes known as smoothing filters
2. Highpass filters, sometimes known as sharpening filters

### 8.2.2 Lowpass Frequency Domain Filters

Lowpass filters:

1. create a blurred (or smoothed) image
2. attenuate the high frequencies and leave the low frequencies of the Fourier transform relatively unchanged

Three main lowpass filters are discussed in *Digital Image Processing Using MATLAB*

1. Ideal lowpass filter (ILPF)
2. Butterworth lowpass filter (BLPF)
3. Gaussian lowpass filter (GLPF)

### 8.2.3 Highpass Frequency Domain Filters

Highpass filters:

1. sharpen (or shows the edges of) an image
2. attenuate the low frequencies and leave the high frequencies of the Fourier transform relatively unchanged

The highpass filter ($H_{hp}$) is often represented by its relationship to the lowpass filter ($H_{lp}$):

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

Because highpass filters can be created in relationship to low pass filters.

## 8.3 Pre-Lab Task

Read chapter no 3 of the book "Digital Image processing using MATLAB, R.C. Gonzales, R.E. Woods, Addison-Wesley" and try to understand the relationship between filters and images.

## 8.4 In-Lab Task

### 3.1 Gaussian Filter

Write a MATLAB program toimplement the Gaussian filters on the image passed to it.

### 8.4.1 Butterworth Filter

Write a MATLAB code to implement the Butterworth low and high pass filter on an image provided by the lab instructor, also give your analysis about the image before and after applying the filters.

### 8.4.2 Ideal Filter

Write a MATLAB code to implement the Ideallow and high pass filter on an image provided by the lab instructor, also give your analysis about the image before and after applying the filters.

### 8.4.3 High Pass Filter

Write a MATLAB code to implement the Ideal, Butterworth and Gaussian high pass filter on an image provided by the lab instructor and find that which filter most sharpens the edges of the in the provided image, support your answer with proper justifications.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 9 To show digital images in different color spaces using MATLAB

## 9.1 Objectives

- To display individual channels color image using MATLAB
- To display a color image conversion between different color spaces using MATLAB

## 9.2 Pre-Lab Theory

An image in MATLAB is a matrix $m \times n$ containing colors to be displayed. The colors have to be defined in a color map, which is another matrix. A color map matrix may have any number of rows, but it must have exactly three columns. Each row is interpreted as a color, with the first element specifying the intensity of red light, the second green, and the third blue (that's why it's called an RGB image or matrix). Color intensity can be specified on the interval 0.0 to 1.0 in case of double. Similarly, the range of values is [O, 255] or [O, 65535]for RGB images of class `unit8` or `unit16`, respectively. The number of bits used to represent the pixel values of the component images determines the bit depth of an RGB image. For example, if each component image is an 8-bit image, the corresponding RGB image is said to be 24 bits deep.

## 9.3 Pre-Lab Task

Write a MATLAB code, capable of separating the RGB layers of any color image. You are also required to show the three layers separately.

### 9.3.1 Basic color mapping

```
colors1 = [
0 0 0   % First element = black
0 0 1   % blue
0 1 0   % green
0 1 1   % cyan
1 0 0   % red
1 0 1   % purple
1 1 0   % yellow
1 1 1]; % Last element = white
% We prepare the matrix that contains the colors to be displayed
% The list refers to the number of the elements in the color list
w = [1 2 3 4 5 6 7 8];
% We use the 'colormap' function to define the actual
% palette in our workspace
colormap(colors1)
% We use the 'image' instruction to display the matrix
image(w)
axis off
```

### 9.3.2 Separating layers of RGB

```
I = imread('Image_name');
r = I(:, :, 1);
g = I(:, :, 2);
b = I(:, :, 3);
K = cat(3, r, g, b);
figure
subplot(121);
imshow(I);
subplot(122);
imshow(K);
figure
subplot(311);
imshow(r);
subplot(312);
imshow(g);
subplot(313);
imshow(b);
```

### 9.3.3 Brightness changes in RGB Image

```
I = imread('Image_name');
r = I(:, :, 1);
g = I(:, :, 2);
b = I(:, :, 3);
r_new=r*2;
g_new=g*2;
b_new=b*2
K = cat(3, r_new, g_new, b_new);
figure
subplot(321);
imshow(r);
subplot(322);
imshow(r_new);
subplot(323);
imshow(g);
subplot(324);
imshow(g_new);
subplot(325);
imshow(b);
subplot(326);
imshow(b_new);
figure
subplot(121);
```

```
imshow(I);
subplot(122);
imshow(K);
```

### 9.3.4 Conversion between different color schemes: RGB to HSI

```
image=imread('Image_name');
imshow(image);
title('ORIGINAL IMAGE')
image=im2double(image);
r=image(:,:,1);
g=image(:,:,2);
b=image(:,:,3);
num=0.5 * ((r-g)+(r-b));
den=sqrt((r-g).^2 +(r-b).*(g-b));
theta=acos(num./(den+eps));
H=theta;
H(b>g)=2*pi -H(b>g);
H=H/(2*pi);
num=min(min(r,g),b);
den=r+g+b;
den(den==0)=eps;
S=1-3.* num./den;
H(S==0)=0;
I=(r+g+b)/3;
hsiimage=cat(3,H,S,I);
figure;
imshow(hsiimage);
title('HSI IMAGE')
```

## 9.4 In-Lab Task

### 3.1 Conversion between different color schemes: HIS to RGB and RGB to CMY

Write a MATLAB code to perform the following task without using the built-in commands

1. Convert the HSI image to RGB.
2. Convert the RGB image to CMY.

## 9.5

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 10 To follow edge detection techniques for boundary detection using MATLAB.

## 10.1 Objectives

- to manipulate digital image by applying different gradient operators using MATLAB.
- to manipulate digital image by applying Laplacian operator using MATLAB.

## 10.2 Pre-Lab Theory

There are many operators which can be used for edge detection, some of them are as fallows.

1. Sobel operator
2. Roberts operator
3. Prewitt operator
4. Laplacian of Gaussian method
5. Zero-cross method

To use these operator, edge() function is used, syntax of `edge()` function is.

**Syntax**

```
BW = edge(I);
BW = edge(I,'sobel');
BW = edge(I,'prewitt');
BW = edge(I,'roberts');
```

`BW = edge(I)` takes a grayscale or a binary image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

By default, edge uses the Sobel method to detect edges but the following provides a complete list of all the edge-finding methods supported by this function:

1. The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
2. The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
3. The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
4. The Laplacian of Gaussian method finds edges by looking for zero crossings after filtering I with a Laplacian of Gaussian filter.
5. The zero-cross method finds edges by looking for zero crossings after filtering I with a filter you specify.
6. The Canny method finds edges by looking for local maxima of the gradient of I. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

The parameters you can supply differ depending on the method you specify. If you do not specify a method, edge uses the Sobel method.

## 10.3 In-Lab Tasks

### 10.3.1 Edge detection using Sobel, Roberts and Prewitts filters

Write a MATLAB code that detects edges in a n image using Sobel, Roberts and Prewitts filters without using built-in commands and compare the result of each filter with built-in commands Filtering using built-in commands is given below

```
i = imread('Image_name');
I = rgb2gray(i);
BW1 = edge(I,'prewitt');
BW2= edge(I,'sobel');
BW3= edge(I,'roberts');
subplot (2,2,1);
imshow(I);
title('original');
subplot(2,2,2);
imshow(BW1);
title('Prewitt');
subplot(2,2,3);
imshow(BW2);
title('Sobel');
subplot(2,2,4);
imshow(BW3);
title('Roberts');
```

### 10.3.2 Edge detection using canny filters

Write a MATLAB code to perform edge detection on an image provided by the lab instructor using canny filter.

### 13.3 Canny versus Sobel Filter

Write a MATLAB code to perform the edge detection on an image provided by the lab instructor, using canny and sobel method. You are also required to compare the result of edge detection using both methods and identify which method shows the best result for the provided image.

### 10.3.3 Vertical and Horizontal edge detection

Write a MATLAB code to perform the vertical and horizontal edge detection on an image provided by the lab instructor and then add the results of vertical and horizontal edge detection. You are also required to compare the result ofvertical and horizontal edge detection with normal edge detection and give your analysis that which edge detection gives better result.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 11 To follow the morphological techniques for segmentation in digital images using MATLAB

## 11.1 Objectives

- to manipulate digital image by applying different morphological operations using MATLAB.

## 11.2 Pre-Lab Theory

In imageprocessing, we use mathematical morphology to extract image components which are useful in representation and description of region shape such as Boundaries, Skeletons, Convex hull, Thinning, Pruning etc. There are two Fundamental morphological operations.

1. Dilation
2. Erosion

### 11.2.1 Dilation

Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels onobject boundaries. The value of the output pixel is the maximum value of all the pixels in theinput pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixelis set to 1.

### 11.2.2 Erosion

Erosion removes pixels to the boundaries of objects in an image, while erosion removes pixels onobject boundaries. The value of the output pixel is the minimum value of all the pixels in theinput pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to0.

Opening and closing operations can be done by combination of erosion and dilation in differentsequence.

## 11.3 Pre-Lab Task

Read chapter no 10 of the book "Digital Image processing using MATLAB, R.C. Gonzales, R.E. Woods, Addison-Wesley" and try to understand the concept and usage of morphological operation on images.

## 3.1 Dilation

```
bw = imread('text.png');
se=[0 1 0; 1 1 1; 0 1 0];
bw_out=imdilate(bw,se);
subplot(1,2,1);
imshow(bw);
subplot(1,2,2);
imshow(bw_out);
```

### 11.3.1 Erosion

```
bw = imread('text.png');
se=ones(6,1);
bw_out=imerode(bw,se);
subplot(1,2,1);
imshow(bw);
subplot(1,2,2);
imshow(bw_out);
```

## 11.4 In-Lab Task

### 3.1 Morphological operation example

Write a MATLAB code, capable of removing the salt and pepper noise of animage provided by the lab instructor, using different morphological operations.

### 11.4.1 Closing and Opening using Erosion and Dilation

Write a MATLAB code to implement the closing and opening operation using erosion and dilation on an image provided by the lab instructor.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____

# 12 To follow segmentation techniques for object detection in digital images using MATLAB.

## 12.1 Objectives

- To manipulate digital image by segmenting it into several parts using MATLAB.

## 12.2 Pre-Lab Theory:

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse.Segmentation subdivides an image into its constituent regions or objects.The level to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects of interest have been isolated. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of the products with the objective of determining the presence or absence of specific anomalies, such as missingcomponents or broken connection paths. There is no reason to carry segmentation past the level of detail required to identify those elements.

## 12.3 Pre-Lab Task

Read chapter no 11 of the book “Digital Image processing using MATLAB, R.C. Gonzales, R.E. Woods, Addison-Wesley” and try to understand the relationship between filters and images.

### 12.3.1 Image Segmentation: Region Based

```
I=imread('Image_name');
level = graythresh(I);
newI=im2bw(I,level);
subplot(121);
imshow(I);
subplot(122)
imshow(newI);
newI=double(newI);
newI=imfill(newI)
[cc,num]=bwlabel(newI);
for n=1:num
figure;
imshow(cc==n);
end
```

### 12.3.2 Image Segmentation: Boundary Based

```
image = imread('jump.jpg');     % read image
[height, width, planes] = size(image);
rgb = reshape(image, height, width * planes);
imagesc(rgb);                   % visualize RGB planes
```

```
r = image(:, :, 1);                 % red channel
g = image(:, :, 2);                 % green channel
b = image(:, :, 3);                 % blue channel
threshold = 100;                    % threshold value
imagesc(b < threshold);             % display the binarized image
% apply the blueness calculation
blueness = double(b) - max(double(r), double(g));
figure
imagesc(blueness);                  % visualize RGB planes
%colorbar on                         % display colorbar
% apply thresholding to segment the foreground
mask = blueness < 45;
figure
imagesc(mask);
% create a label image, where all pixels having the same value
labels = bwlabel(mask);
% get the label at point (200, 200)
id = labels(200, 200);
% get the mask containing only the desired object
man = (labels == id);
figure
imagesc(man);
imwrite(man, 'man.jpg');
```

## 12.4 In-Lab Task

### 3.1 Image Segmentation

Write a MATLAB code to perform image segmentation on an image provided by the lab instructor.

## Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned taskswithout any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:** _____ **Date:** _____