DIGITAL IMAGE PROCESSING LAB

L A B F I L E

GEETANJALI INSTITUTE OF TECHNICAL STUDIES

(Affiliated to Rajasthan Technical University, Kota, Rajasthan)

(Session 2020-21)

NAME :

ROLL NUMBER :

SECTION :

VISION & MISSION OF INSTITUTE

Vision of Geetanjali Institute of Technical studies

To achieve excellence in technical and management education through quality teaching and innovation.

Mission of Geetanjali Institute of Technical studies

M1: To provide an excellent learning environment to produce socially responsible and productive technical professionals.

M2: To set up the state-of-the-art facilities for quality education and innovation.

M3: To impart knowledge & Skills leading to shaping a budding manager as a quality executive.

M4: To encourage for life-long learning and team-based problem solving through learning environment.

VISION & MISSION OF DEPARTMENT

Department Vision

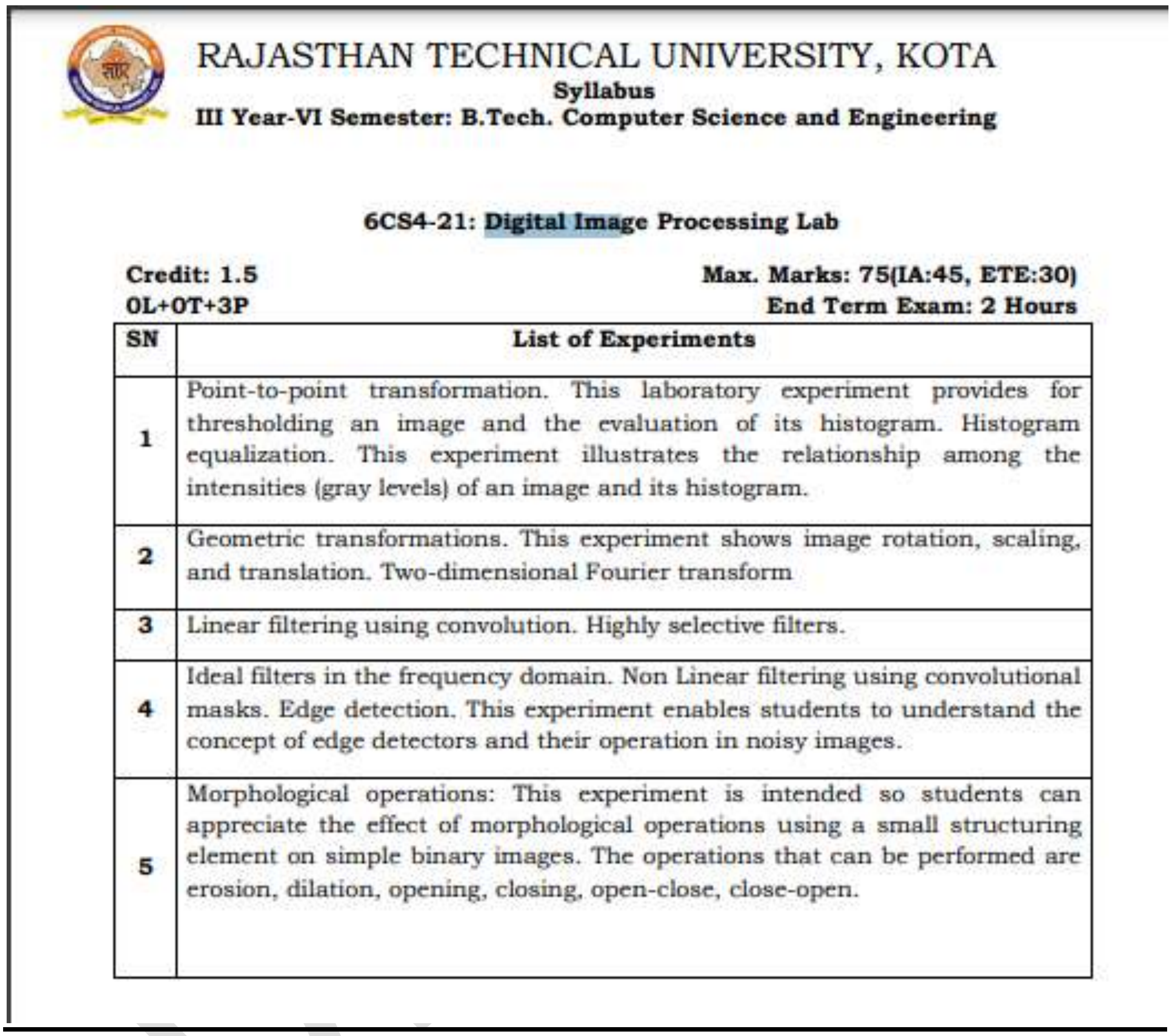
To nurture the students to become employable graduates who can provide solutions to the societal issues through ICT.

Department Mission

M1: *To focus on practical approach towards learning and exposing the students on the latest ICT technologies.*

M2: *To foster logical thinking among the students to solve real-time problems using innovative approaches.*

M3: To provide state-of-the-art resources that contributes to inculcate ethical and life-long learning environment.



RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

III Year-VI Semester: B.Tech. Computer Science and Engineering

6CS4-21: Digital Image Processing Lab

Credit: 1.5 | **Max. Marks: 75(IA:45, ETE:30)**

0L+0T+3P | **End Term Exam: 2 Hours**

SN	**List of Experiments**
1	Point-to-point transformation. This laboratory experiment provides for thresholding an image and the evaluation of its histogram. Histogram equalization. This experiment illustrates the relationship among the intensities (gray levels) of an image and its histogram.
2	Geometric transformations. This experiment shows image rotation, scaling, and translation. Two-dimensional Fourier transform
3	Linear filtering using convolution. Highly selective filters.
4	Ideal filters in the frequency domain. Non Linear filtering using convolutional masks. Edge detection. This experiment enables students to understand the concept of edge detectors and their operation in noisy images.
5	Morphological operations: This experiment is intended so students can appreciate the effect of morphological operations using a small structuring element on simple binary images. The operations that can be performed are erosion, dilation, opening, closing, open-close, close-open.

Index

EXPERIMENT No. 1

AIM: Reading and Displaying of different formats

a) Reading Images

Images are read into MATLAB environment using function imread, whose basic syntax is

imread('filename')

Here, filename is a string containing the complete name of the image file (including any applicable extension). For example, the statement

```
>>f=imread('face.jpg');
```

reads the image from the JPEG file face into image array f.

Note :

1. The use of single quotes (') to delimit the string filename.
2. The semicolon at the end of a statement is used by MATLAB for *suppressing* output. If a semicolon is not included, MATLAB displays on the screen the results of the operation(s) specified in that line.
3. The prompt symbol (>>) designates the beginning of a command line, as it appears in the MATLAB Command Window.

When, as in the preceding command line, no path information is included in filename, imread reads the file from the Current Directory and, if that fails, it tries to find the file in the MATLAB search path. The simplest way to read an image from a specified directory is to include a full or relative path to that directory in filename. For example,

```
>> f = imread('D:\myimages\face.jpg');
```

reads the image from a directory called myimages in the D: drive, whereas

```
>> f = imread('.\myimages\face.jpg');
```

reads the image from the myimages subdirectory of then current working directory. The MATLAB Desktop displays the path to the Current Directory on the toolbar, which provides an easy way to change it.

Table 2.1 lists some of the most popular image/graphics formats supported by imread

Format Name	**Description**	**Recognized Extensions**
BMP	Window Bitmap	.bmp
CUR	Window Cursor Resources	.cur
FITS	Flexible Image Transport System	.fts,.fits
GIF	Graphics Interchange Format	.gif
HDF	Hierarchical Data Format	.hdf
ICO	Window Icon Resources	.ico
JPEG	Joint Photographic Experts Group	.jpg,.jpeg
JPEG2000	Joint Photographic Experts Group	.jp2,.jpf,.jpx
PBM	Portable Bitmap	.pbm
PGM	Portable Bitmap	.pgm
PNG	Portable Network Graphics	.png
PNM	Portable Any Map	.pnm
RAS	Sun Raster	.ras
TIFF	Tagged Image File Format	.tif, .tiff
XWD	X Window Dump	.xwd

Typing size at the prompt gives the row and column dimensions of an image:

```
>>size(f)

ans =

241   181     3
```

More generally, for an array A having an arbitrary number of dimensions, a statement of the form

$$[D_1, D_2, \ldots, D_K] = \text{size}(A)$$

returns the sizes of the first K dimensions of A. This function is particularly useful in programming to determine automatically the size of a 2-D image:

```
>> [M, N] = size(f);
```

This syntax returns the number of rows (M) and columns (N) in the image. Similarly, the command

```
>> M = size(f, 1);
```

gives the size of f along its first dimension, which is defined by MATLAB as the vertical dimension. That is, this command gives the number of rows in f. The second dimension of an array is in the horizontal direction, so the statement size(f, 2) gives the number of columns in f. A *singleton dimension* is any dimension, dim, for which size(A, dim) = 1.

The whos function displays additional information about an array. For instance, the statement

```
>>whos f
```

gives

```
Name      Size            Bytes  Class    Attributes

f       241x181x3         130863  uint8
```

The Workspace Browser in the MATLAB Desktop displays similar information. The uint8 entry shown refers to one of several MATLAB data classes. A semicolon at the end of a whos line has no effect, so normally one is not used.

b) <u>Displaying Images</u>

Images are displayed on the MATLAB desktop using function imshow, which has the basic syntax:

```
imshow(f)
```

wheref is an image array. Using the syntax

```
imshow(f, [low high])
```

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

```
imshow(f, [ ])
```

sets variable low to the minimum value of array f and high to its maximum value. This form of imshowis useful for displaying images that have a low dynamic range or that have positive and negative values.

Example: Reading and Displaying images

The following statements read from disk an image called face.jpg, extract information about the image, and display it using imshow:

```
>> f = imread(face.jpg);
>>whos f
Name       Size             Bytes  Class    Attributes

f       241x181x3         130863  uint8

>>imshow(f)
```

A semicolon at the end of an imshowline has no effect, so normally one is not used. Figure shows what the output looks like on the screen.

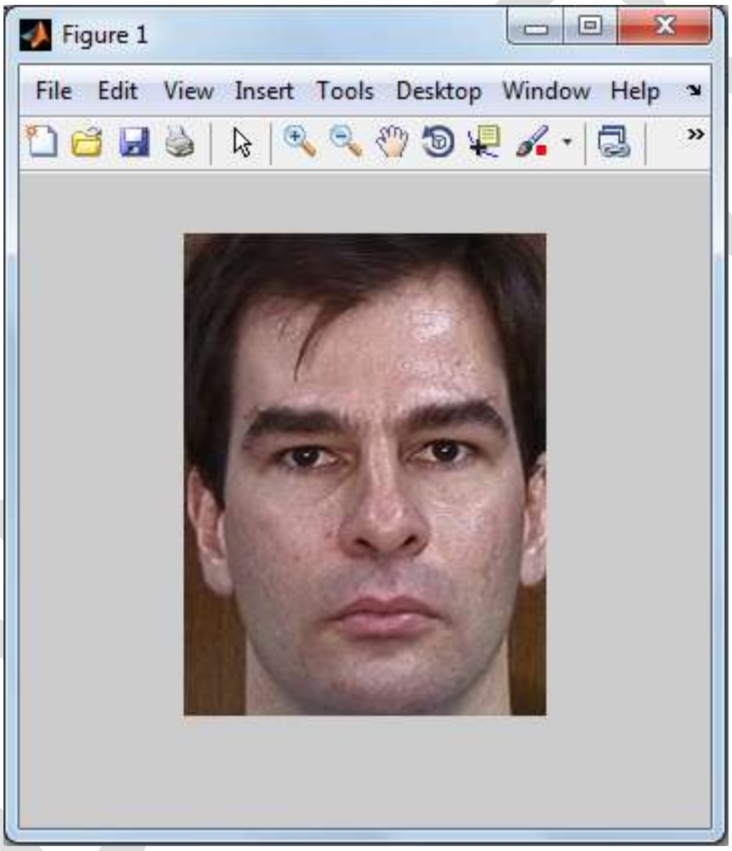


Fig: Screen capture showing how an image appears on the MATLAB desktop. Note the figure number on the top, left of the window.

The figure number appears on the top, left of the window. Note the various pull-down menus and utility buttons. They are used for processes such as scaling, saving, and exporting the contents of the display window. In particular, the **Edit** menu has functions for editing and formatting the contents before they are printed or saved to disk.

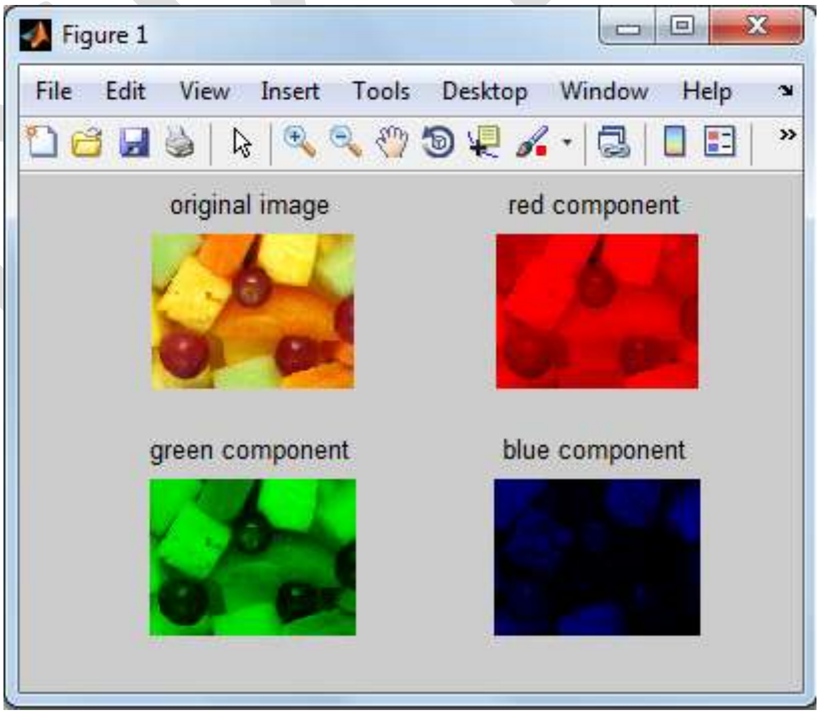
EXPERIMENT No. 2

AIM: Read an RGB image and extract the three colour components, red, green and blue

MATLAB CODE

```
clc;
close all;
clear all;
RGB=imread('Mixed-Fruit.bmp');
 R=RGB;
G=RGB;
B=RGB;
R(:,:,2)=0;
R(:,:,3)=0;
G(:,:,1)=0;
G(:,:,3)=0;
B(:,:,1)=0;
B(:,:,2)=0;
subplot(2,2,1), imshow(RGB), title('original image')
subplot(2,2,2), imshow(R), title('red component')
subplot(2,2,3), imshow(G), title('green component')
subplot(2,2,4), imshow(B), title('blue component')
```

OUTPUT:



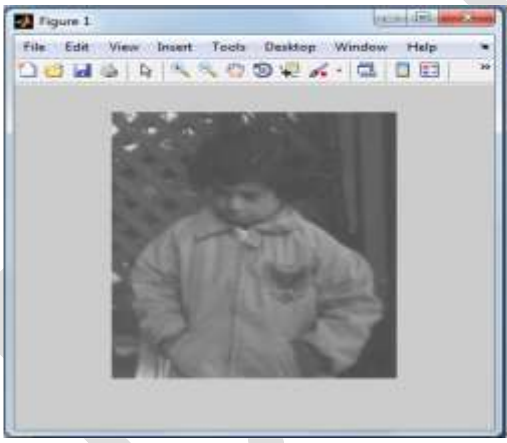
EXPERIMENT No. 3

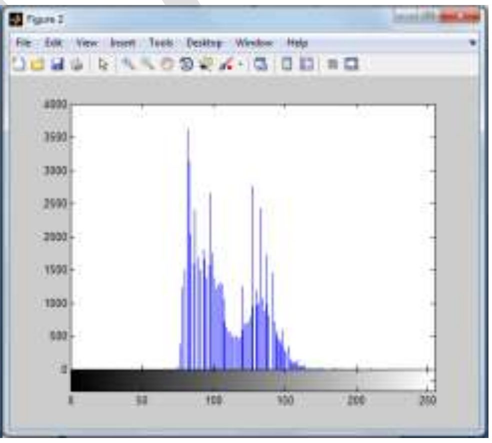
AIM: Compute the image histogram and image equalisation

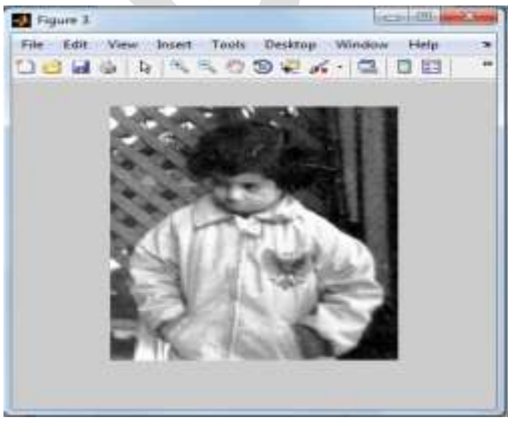
MATLAB CODE

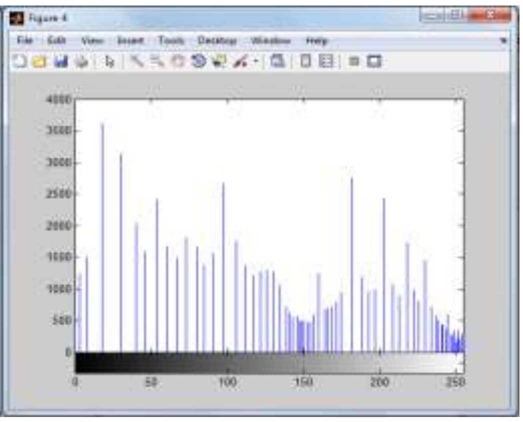
```
clc;
close all;
clear all;
f = imread('pout.tif');
imshow(f); % Figure1
figure, imhist(f) % Figure2
ylim('auto')
g = histeq(f, 256);
figure, imshow(g) % Figure3
figure, imhist(g) % Figure4
ylim('auto')
```

OUTPUT









EXPERIMENT No. 4

AIM: Morphological operations in analyzing image structures

A) Perform the dilation and eroded process

MATLAB CODE

```
clc;
close all;
clear all;
a=imread('digitalmonogram.jpg');
b=[1 1 1;1 1 1;1 1 1];
a1=imdilate(a,b);
a2=imerode(a,b);
imshow(a),title('original image'); %FIGURE1
figure,imshow(a1),title('Dilated image'); %FIGURE2
figure,imshow(a2),title('erode omage'); %FIGURE3
```

OUTPUT

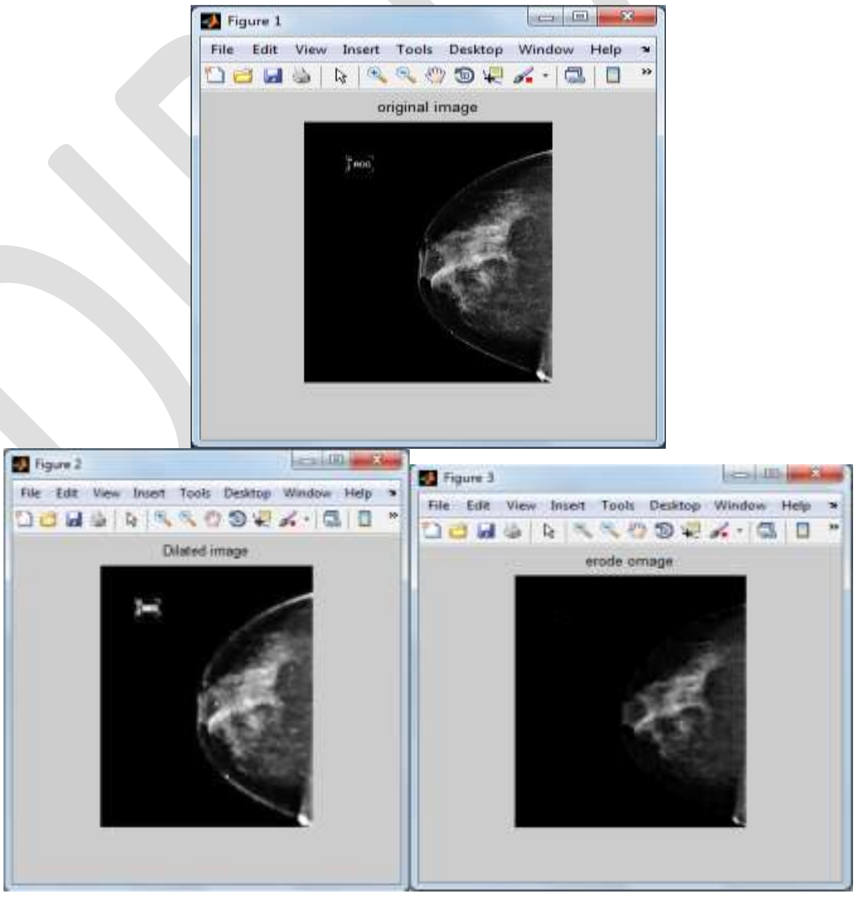


Figure 1
File Edit View Insert Tools Desktop Window Help
original image
Figure 2
File Edit View Insert Tools Desktop Window Help
Dilated image
Figure 3
File Edit View Insert Tools Desktop Window Help
erode omage

Perform an opening and closing operation

MATLAB CODE

```
clc;
close all;
clear all;
a=imread('digitalmonogram.jpg');
b=[1 1 1;1 1 1;1 1 1];
a1=imopen(a,b);
a2=imclose(a,b);
imshow(a),title('original image');%figure1
figure,imshow(a1),title('opened image'); %figure2
figure,imshow(a2),title('closed omage'); %figure3
```

OUTPUT

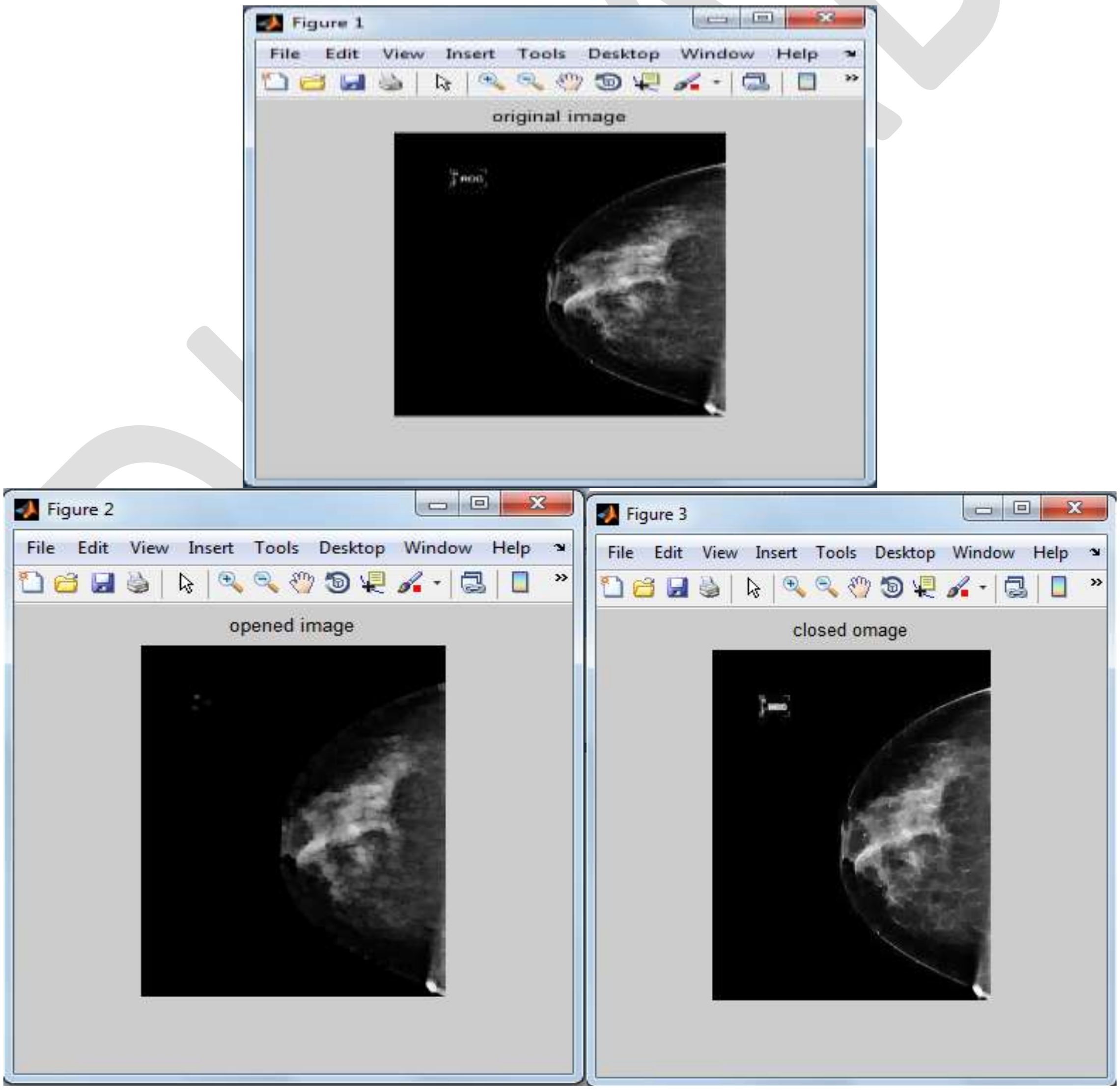


Figure 1
File Edit View Insert Tools Desktop Window Help
original image
Figure 2
File Edit View Insert Tools Desktop Window Help
opened image
Figure 3
File Edit View Insert Tools Desktop Window Help
closed omage

EXPERIMENT No. 5

AIM: Image filtering in spatial and frequency domain

Image processing techniques supports linear and non linear spatial filters

a) Linear Spatial Filters

The toolbox supports a number of predefined 2D linear spatial filters obtained by using function fspecial which generates a filter mask w, using the syntax

w=fspecial('type',parameters)

where 'type' specifies the filter type and parameters define the specified filter. The spatial filter supported by fspecial are summarized in Table.

Type	Syntax & Parameters
average	fspecial('average',[r,c]). A rectangular averaging filter of size r X c. The default is 3 X 3. A single number instaed of [r,c] specifies a square filter
disk	fspecial('disk',r). A circular averaging filter with radius r.
gaussian	fspecial('gaussian',[r,c],sig). A gaussian low pass filter of size r X c. A single number instaed of [r,c] specifies a square filter
laplacian	fspecial('laplacian', alpha). A 3 x 3 laplacian filter whose shape is specified by alpha a number in the range [0,1]
Log	fspecial('log',[r,c],sig). Laplacian of gaussian filter of size r X c. A single number instaed of [r,c] specifies a square filter
motion	fspecial('motion',len,theta). Outputs a filter that when convolved with an image, approximates linear motion of len pixels. The direction of motion is theta measured in degrees counterclockwise from horizontal.
prewitt	fspecial(''prewitt'). Output a 3 X 3 Prewitt mask, wv, that approximates a vertical gradient. A mask for the horizantal gradient is obtained by transposing the result: wh=wv'

sobel	fspecial('sobel').Output a 3 X 3 sobel mask, sv, that approximates a vertical gradient. A mask for the horizantal gradient is obtained by transposing the result: sh=sv'
unsharp	fspecial('unsharp', alpha).Output a 3 X 3 unsharp filter. Parameter alpha controls the shape; it must be greater than 0 and less than or equal to 1; the default is 0.2.

b) Non Linear Spatial Filters

A commonly used tool for generating spatial filters in image processing techniques is function ordfilt2, which generates order statistic filters (also called rank filters). These are nonlinear spatial filters whose response is based on ordering the pixels contained in an image neighbourhood and then replacing the value of the centre pixel in the neighbourhood with the value determined by the ranking result. Non linear filters generated by ordfilt2.

The syntax of function ordfilt2 is

```
g=ordfilt2(f, order, domain)
```

This function creates the output image g by replacing each element of f by the order-th element in the sorted set of neighbours specified by the nonzero elements in domain.

EXPERIMENT No. 6

AIM: Study of Image Compression.

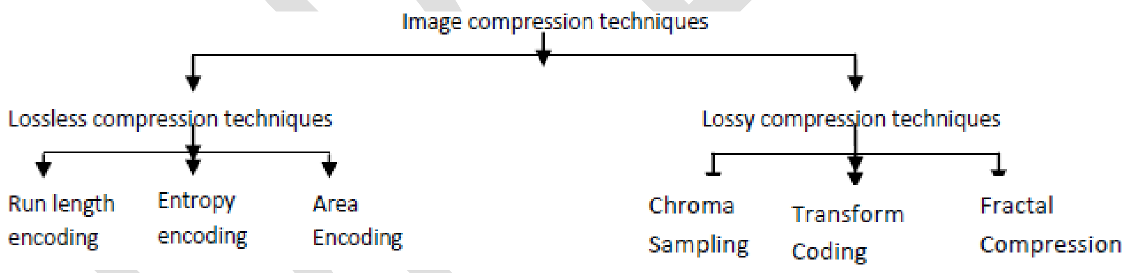
Theory: Image compression is the process of encoding or converting an image file in such a way that it consumes less space than the original file. It is a type of compression technique that reduces the size of an image file without affecting or degrading its quality to a greater extent.

Image compression is typically performed through an image/data compression algorithm or codec. Typically such codecs/algorithms apply different techniques to reduce the image size, such as by:

- Specifying all similarly colored pixels by the color name, code and the number of pixels. This way one pixel can correspond to hundreds or thousands of pixels.
- The image is created and represented using mathematical wavelets.
- Splitting the image into several parts, each identifiable using a fractal.

COMPRESSION TECHNIQUES

The compression techniques used in image processing are for various applications are:



a) LOSSLESS COMPRESSION TECHNIQUES

As the name indicates the original image can be perfectly recovered using the lossless compression techniques. They are also known as entropy coding, noiseless compression etc. They will not introduce any noises to the image and they are using statistics or decomposition techniques to reduce the redundancy. As mentioned earlier they are preferred for medical imaging, technical drawing etc. The following are some of the methods which are used for lossless compression.

1. Run length encoding
2. Entropy encoding
3. Area coding.

Run length encoding

This method is a simple lossless compression method. It is mainly used for sequential data. This is most useful on data that contains repetitive information. This method will replace identical symbols. These identical symbols are known as runs. They are replaced by shorter symbols. This technique supported by most bitmap file formats, such as TIFF, BMP, and PCX.

RLE is suited for compressing any type of data regardless of its information content, but the content of the data will affect the compression ratio achieved by RLE. Although most RLE algorithms cannot achieve the high compression ratios of the more advanced compression methods, RLE is both easy to implement and quick to execute, making it a good alternative to either using a complex compression algorithm or leaving your image data uncompressed. There are a number of variants of run-length encoding.

Image data is normally run-length encoded in a sequential process that treats the image data as a 1D stream, rather than as a 2D map of data. In sequential processing, a bitmap is encoded starting at the upper left corner and proceeding from left to right across each scan line (the X axis) to the bottom right corner of the bitmap.

But alternative RLE schemes can also be written to encode data down the length of a bitmap (the Y axis) along the columns), to encode a bitmap into 2D tiles, or even to encode pixels on a diagonal in a zig-zag fashion Odd RLE variants such as this last one might be used in highly specialized applications but are usually quite rare

Entropy encoding

Entropy encoding is another lossless compression technique. It works independent of the specific characteristics of medium. Besides using it as a compression technique it can be also used to measure the similarity in data streams. This method works as follows. It will create a unique prefix code and assign this code to unique symbol in the input. Unlike RLE, entropy encoders works by compressing data by replacing the fixed length output with a prefix code word.

This is of varying size after creating the prefix code. This will be similar to the negative logarithm of probability. There are many entropy coding methods. The most common techniques are Huffman coding and arithmetic coding. Huffman coding was developed by David A. Huffman. It will use a variable-length code table for encoding a source symbol (such as a character in a file) where it has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. The prefix code used in this technique is known as prefix-free codes. This technique is similar to block encoding technique and it is optimal for symbol by symbol encoding. But when symbol by symbol restriction is dropped it will not be optimal.

Constant Area Encoding

This method is an enhanced form of run length encoding method. There is some significant advantage of using this technique over other lossless methods .In constant area coding special code words are used to identify large areas of contiguous 1's and 0's.Here the image is segmented in to blocks and then the segments are classified as blocks which only contains black or white pixels or blocks with mixed intensity.

Another variant of constant area coding is to use an iterative approach in which the binary image is decomposed into successively smaller and smaller block .A hierarchical tree is built from these blocks. The section stops when the block reaches certain predefined size or when all pixels of the block have the same value. The nodes of this tree are then coded. For compressing white text a simpler approach is used. This is known as white block skipping. In this blocks containing solid white areas are coded to 0 and all other areas are coded to 1.They are followed by bit pattern.

LOSSY COMPRESSION TECHNIQUES

Lossy schemes provide much higher compression ratios than lossless schemes. By this scheme, the decompressed image is not identical to the original image, but reasonably close to it. But this scheme is widely used. Lossy methods are especially suitable for natural images such as photographs in applications where minor loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces imperceptible differences may be called visually lossless. The following methods are used in lossy compression

1. Chroma sub sampling
2. Transform coding
3. Fractal Compression

Chroma sub sampling

This takes advantage of the fact that the human eye perceives spatial changes of brightness more sharply than those of color, by averaging or dropping some of the chrominance information in the image. It works by taking advantage of the human visual system's lower acuity for color differences than for luminance.

It is mainly used in video encoding, jpeg encoding etc. Chroma sub sampling is a method that stores color information at lower resolution than intensity information. The overwhelming majority of graphics programs perform 2x2 chroma sub sampling, which breaks the image into 2x2 pixel blocks and only stores the average color information for each 2x2 pixel group.

Transform coding

It is a type of compression for natural data like photographic images .It will result a low quality output of original image. It is a core technique recommended by jpeg. Transform coding is used to convert spatial image pixel values to transform coefficient values. Since this is a linear process and no information is lost, the number of coefficients produced is equal to the number of pixels transformed. Many types of transforms have been tried for picture coding, including for example Fourier, Karhonen-Loeve, Walsh-Hadamard, lapped orthogonal, discrete cosine (DCT), and recently, wavelets.

Fractal Compression It is one of the lossy compression technique used in digital images .As the name indicates it mainly based on the fractals. This approach is good for natural images and textures. It works on the fact that parts of an image often resemble other parts of the same image. This method converts these parts into mathematical data. These data are called "fractal codes" Which are used to recreate the encoded image.