THE NEOTIA UNIVERSITY

Digital Image Processing Lab Manual

Experiment – 1

Object: To provides the thresholding an image and the evaluation of its histogram using histogram equalization and illustrates the relationship among the intensities (gray levels) of an image and its histogram.

Software: MATLAB

Theory :

Histogram is a graphical representation of the intensity distribution of an image. In simple terms, it represents the number of pixels for each intensity value considered.

Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

A color histogram of an image represents the number of pixels in each type of color component. Histogram equalization cannot be applied separately to the Red, Green and Blue components of the image as it leads to dramatic changes in the image's color balance. However, if the image is first converted to another color space, like HSL/HSV color space, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image.

Adaptive Histogram Equalization

Adaptive Histogram Equalization differs from ordinary histogram equalization in the respect that the adaptive method computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image. It is therefore suitable for improving the local contrast and enhancing the definitions of edges in each region of an image.

Contrastive Limited Adaptive Equalization

Contrast Limited AHE (CLAHE) differs from adaptive histogram equalization in its contrast limiting. In the case of CLAHE, the contrast limiting procedure is applied to each neighborhood from which a transformation function is derived. CLAHE was developed to prevent the over amplification of noise that adaptive histogram equalization can give rise to.

Program:

```
        clc;

        clear all;

        close all;

         imgetfile;

         u=imread(ans);

         o=rgb2gray(u);

         imshow(o);

       figure;
       imhist(o);
       i=histeq(o);
       figure;
       imshow(i);
       figure;
       imhist(i);
```

Result: In this experiment provides for thresholding an image and the evaluation of its histogram using Histogram equalization. This experiment illustrates the relationship among the intensities (gray levels) of an image and its histogram.

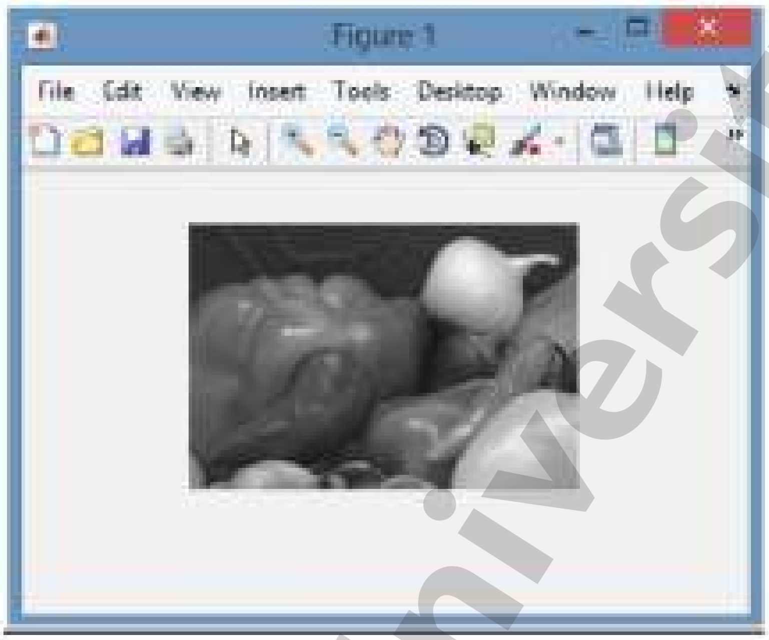


Fig 1.1: Gray Scale Image

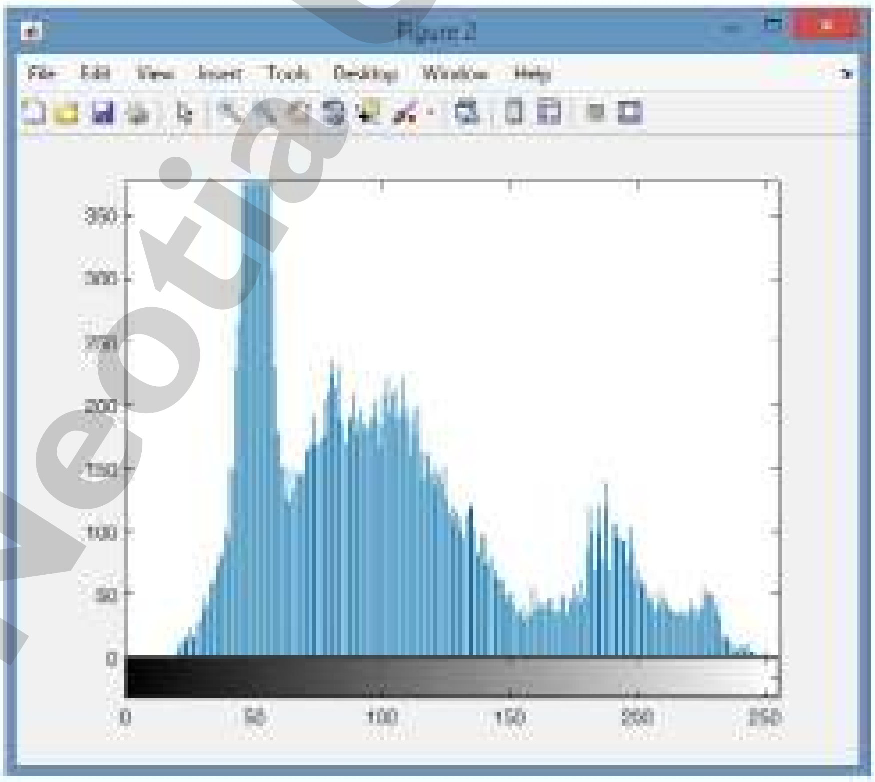


Fig 1.2: Histogram of Original Image

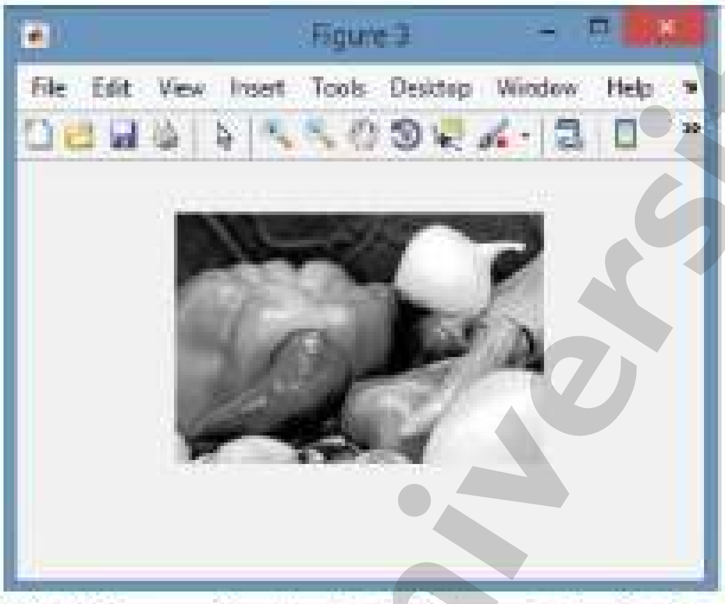


Fig 1.3: Image after Applying Histogram Equalization

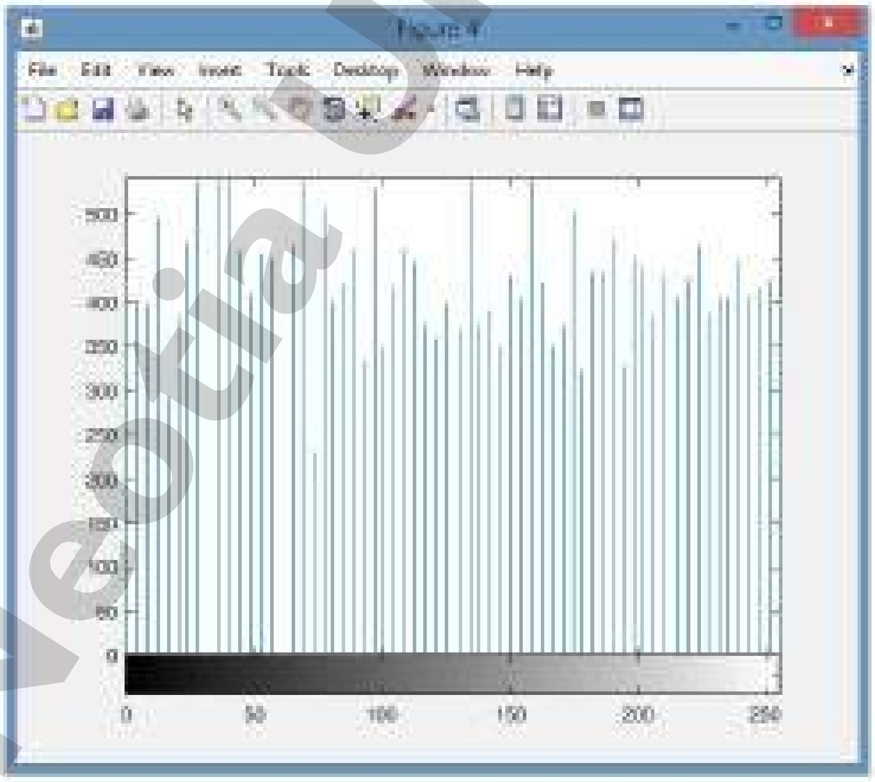


Fig 1.4: Histogram equalization

Experiment – 2

Object: To shows image rotation, scaling, and translation using Geometric transformations.

Software: MATLAB

Theory:

Perform generic geometric transformations using the imwarp workflow. Geometric transformations map pixel coordinates in the output image to coordinates in the input image. The mapping process then interpolates the value of output pixels from the input image. Use these functions to perform general 2-D, 3-D, and N-D geometric transformations. To perform a 2-D or 3-D geometric transformation, first create a geometric transformation object that stores information about the transformation. Then, pass the image to be transformed and the geometric transformation object to the imwarp function.

Functions

imwarp	Apply geometric transformation to image
affineOutputView	Create output view for warping images
fitgeotrans	Fit geometric transformation to control point pairs
findbounds	Find output bounds for spatial transformation
fliptform	Flip input and output roles of spatial transformation structure
makeresampler	Create resampling structure
maketform	Create spatial transformation structure (TFORM)
tformarray	Apply spatial transformation to N-D array
tformfwd	Apply forward spatial transformation
tforminv	Apply inverse spatial transformation

Program:

```
clc;

clear all;

close all;

 imgetfile;

u=imread(ans);
im_cr = imcrop(u,[75 68 130 112]);
imshow(u);
figure;
imshow(im_cr);
im_rt=imrotate(u,90);
figure;
imshow(u);
figure;
imshow(im_rt);
im_ts= imtranslate(u,[15, 25],'FillValues',255);
figure, pi=imshow(u);
pi.Parent.Visible='on';
figure, op=imshow(im_ts);
op.Parent.Visible='on';
```

Result: We have done the operation on digital image and shown image rotation, scaling, and translation using Geometric transformations.

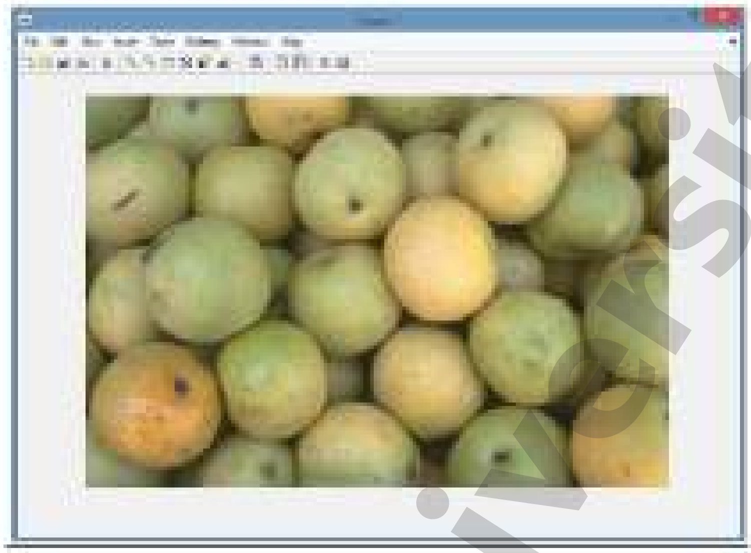


Fig 2.1: Original Image

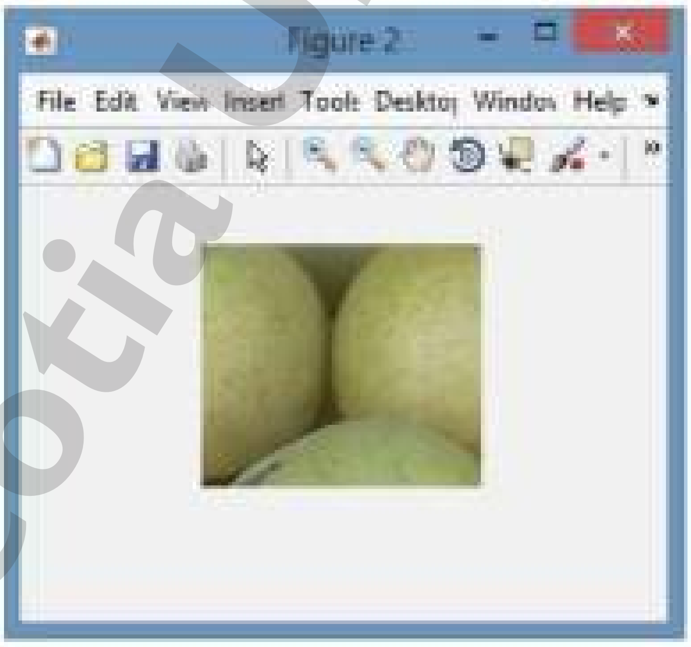


Fig 2.2: Image Crop

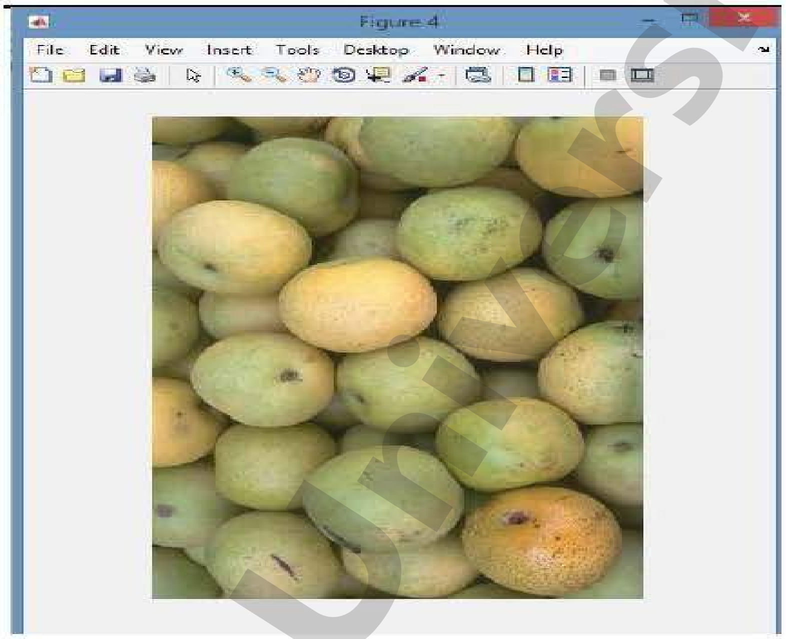


Fig 2.3: Original Image Rotate 90^0

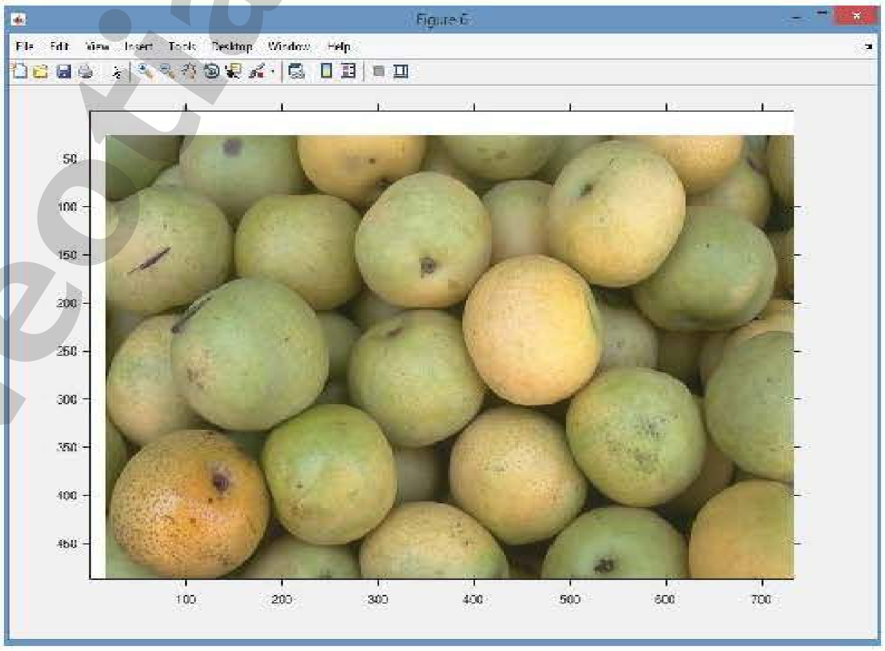


Fig 2.4: Image Scaling

Experiment – 3

Object: To perform the Two-dimensional Fourier transform operation in an image.

Software: MATLAB

Theory:

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

Program:

```
clc; clear all; close all;
imgetfile;
u=imread(ans);
oip=rgb2gray(u);
imshow(oip);
uiop=fft2(oip);
figure;
imshow(uiop);
uyiop=ifft2(uiop);
figure;
imshow(uint8(uyiop));
```

Result: Performed the Two-dimensional Fourier transform operation in an image.

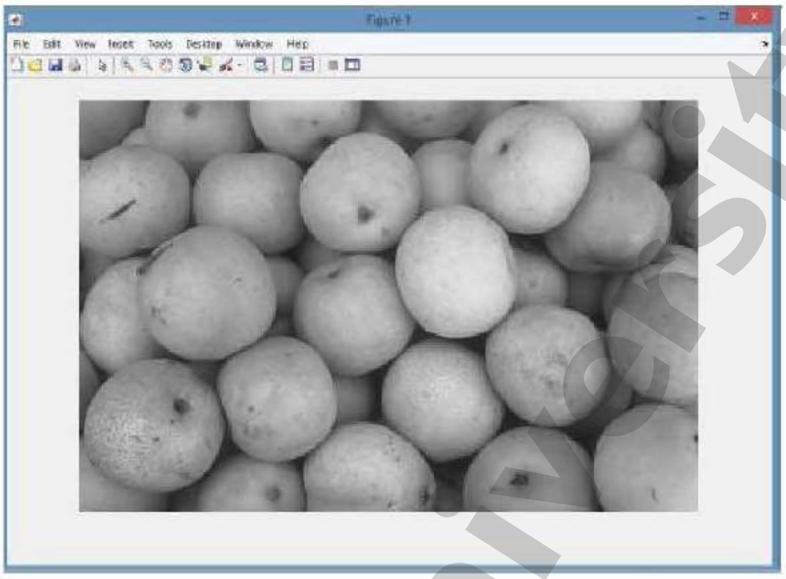


Fig 3.1: Gray Scale Image

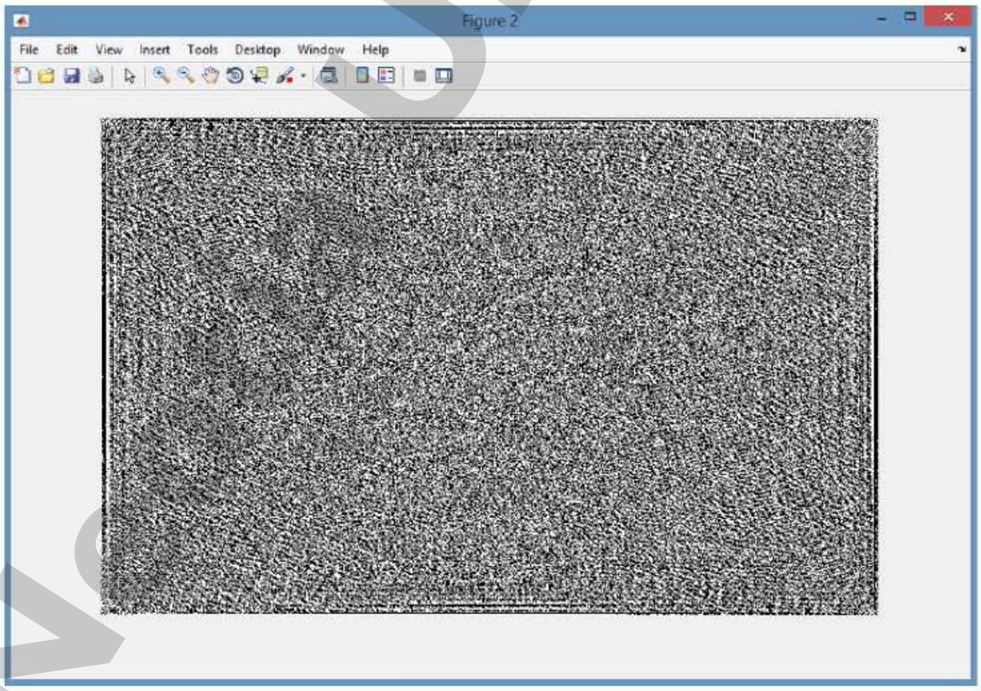


Fig 3.2: Encrypted Image after Applying Fourier Transform

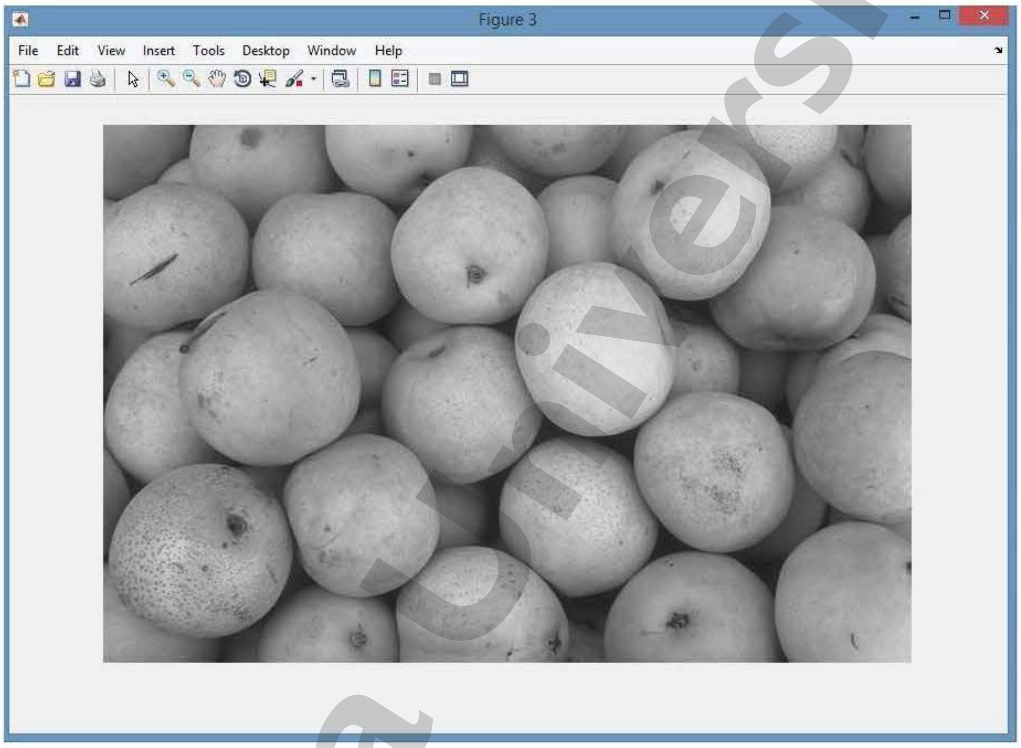


Fig 3.3: Decrypted image After Applying Inverse of Fourier Transform

Experiment – 4

Object: To perform the Linear filtering using convolution in an image.

Software: MATLAB

Theory:

Linear filtering of an image is accomplished through an operation called *convolution*. Convolution is a neighborhood operation in which each output pixel is the weighted sum of neighboring input pixels. The matrix of weights is called the convolution kernel, also known as the filter. A convolution kernel is a correlation kernel that has been rotated 180 degrees.
For example, suppose the image is

A = [17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9]

and the convolution kernel is

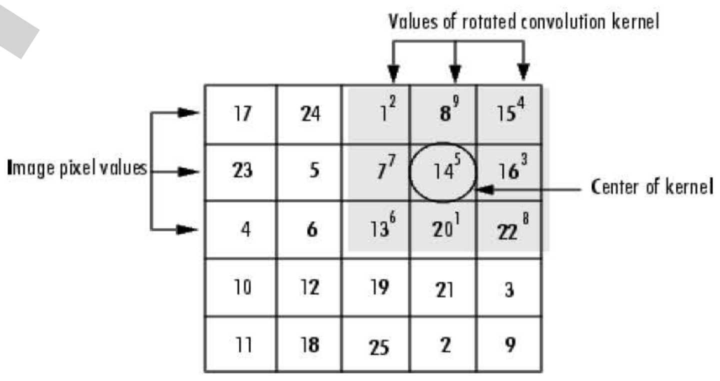
h = [8 1 6
3 5 7
4 9 2]

The following figure shows how to compute the (2,4) output pixel using these steps:

1. Rotate the convolution kernel 180 degrees about its center element.
2. Slide the center element of the convolution kernel so that it lies on top of the (2,4) element of A.
3. Multiply each weight in the rotated convolution kernel by the pixel of A underneath.
4. Sum the individual products from step 3.

Hence the (2,4) output pixel is

$$1\cdot 2+8\cdot 9+15\cdot 4+7\cdot 7+14\cdot 5+16\cdot 3+13\cdot 6+20\cdot 1+22\cdot 8=575$$



Program:

```
clc;
clear
all;
close
all;
imgetfile;
u=imread(ans);
imshow(u);
Hm        =        fspecial('motion',20,45);
MotionBlur                                =
imfilter(u,Hm,'replicate'); figure;
imshow(MotionBlur);
Hb                          =
fspecial('disk',10);
blurred     =     imfilter(u,Hb,'replicate');
figure;
imshow(blurred);
```

Result : We perform the Linear filtering using convolution in an image.

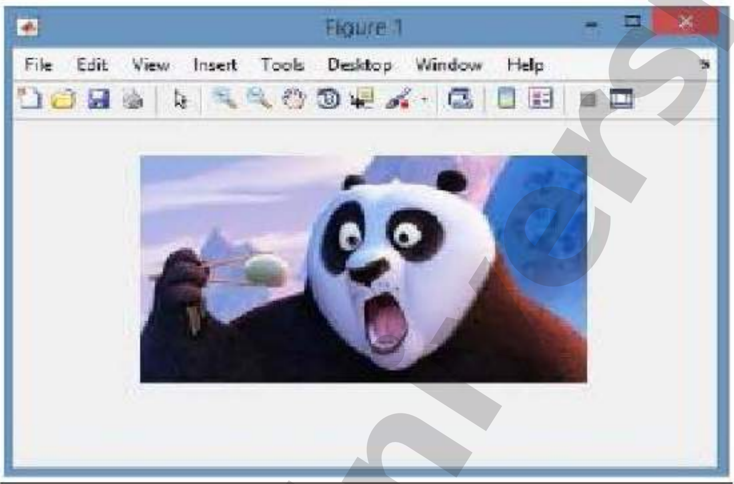


Figure 1
File Edit View Insert Tools Desktop Window Help

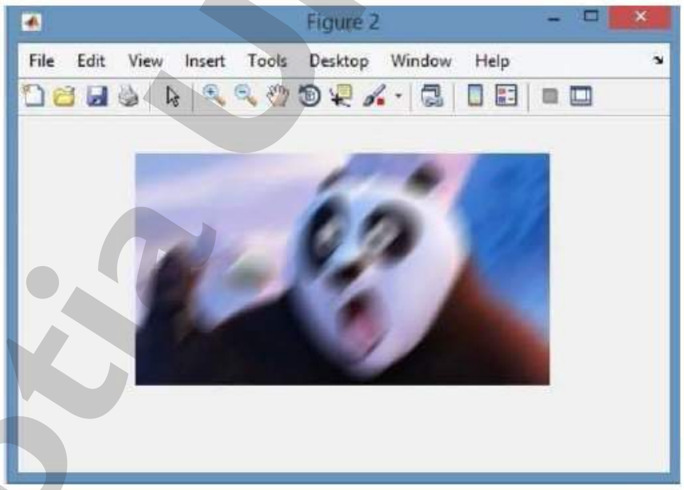


Figure 2
File Edit View Insert Tools Desktop Window Help

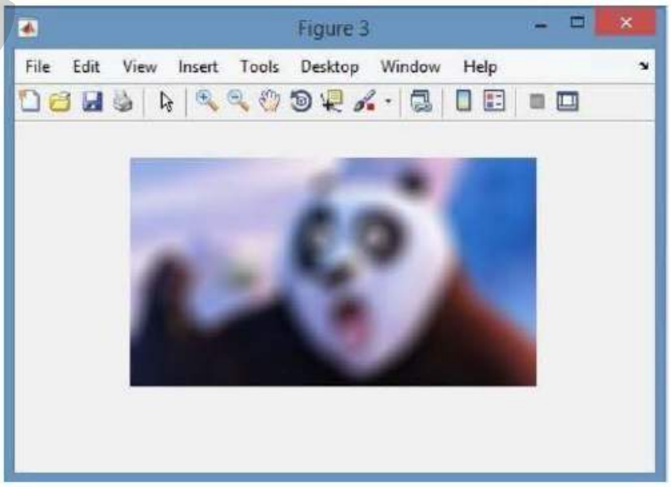


Figure 3
File Edit View Insert Tools Desktop Window Help

Experiment – 5

Object: Image Edge Detection Using Sobel Filtering and Canny Filtering.

Software: MATLAB

Theory:

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods.

Edge detection method, specified as one of the following.

Method	**Descripti on**
'Sobel'	Finds edges at those points where the gradient of the image I is maximum, using the Sobel approximation to the derivative.
'Prewitt'	Finds edges at those points where the gradient of I is maximum, using the Prewitt approximation to the derivative.
'Roberts'	Finds edges at those points where the gradient of I is maximum, using the Roberts approximation to the derivative.
'log'	Finds edges by looking for zero-crossings after filtering I with a Laplacian of Gaussian (LoG) filter.
'zerocross'	Finds edges by looking for zero-crossings after filtering I with a filter that you specify, h

'Canny'	Finds edges by looking for local maxima of the gradient of I. The edge function calculates the gradient using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges.
'approxcanny'	Finds edges using an approximate version of the Canny edge detection algorithm that provides faster execution time at the expense of less precise detection. Floating point images are expected to be normalized in the range [0 1].

Program:

```
clc;clear all;close all;

po=imgetfile;
I = imread(po);%select coin
 imshow(I)
BW1  = edge(I,'sobel');
BW2 = edge(I,'canny');
figure;
 imshowpair(BW1,BW2,'montage');
 title('Sobel Filter Canny Filter');
```

Result: we have perform the Image Edge Detection Using Sobel Filtering and Canny Filtering

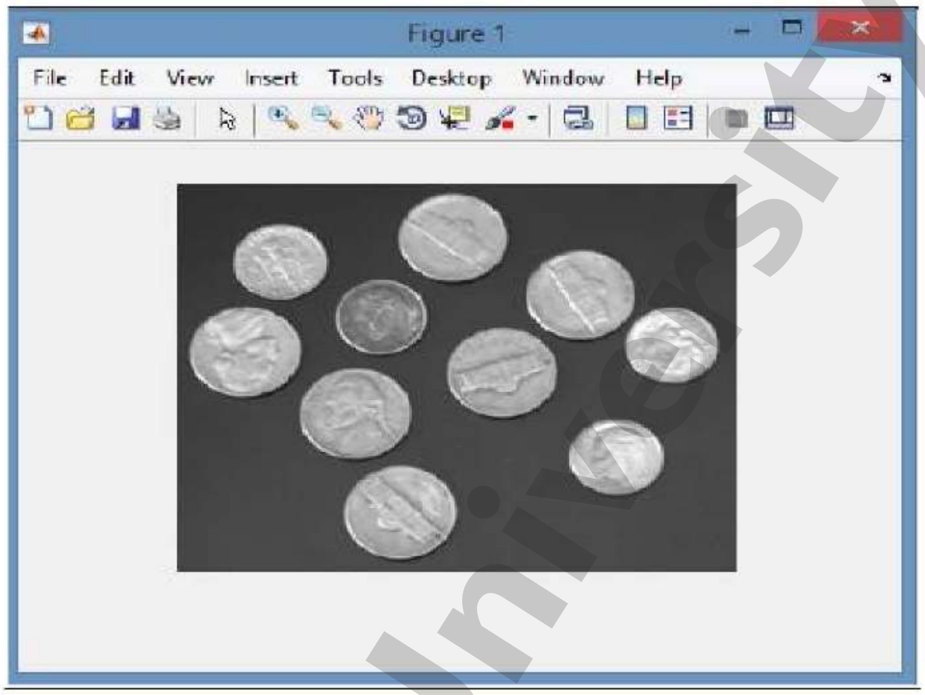


Figure 1
File Edit View Insert Tools Desktop Window Help

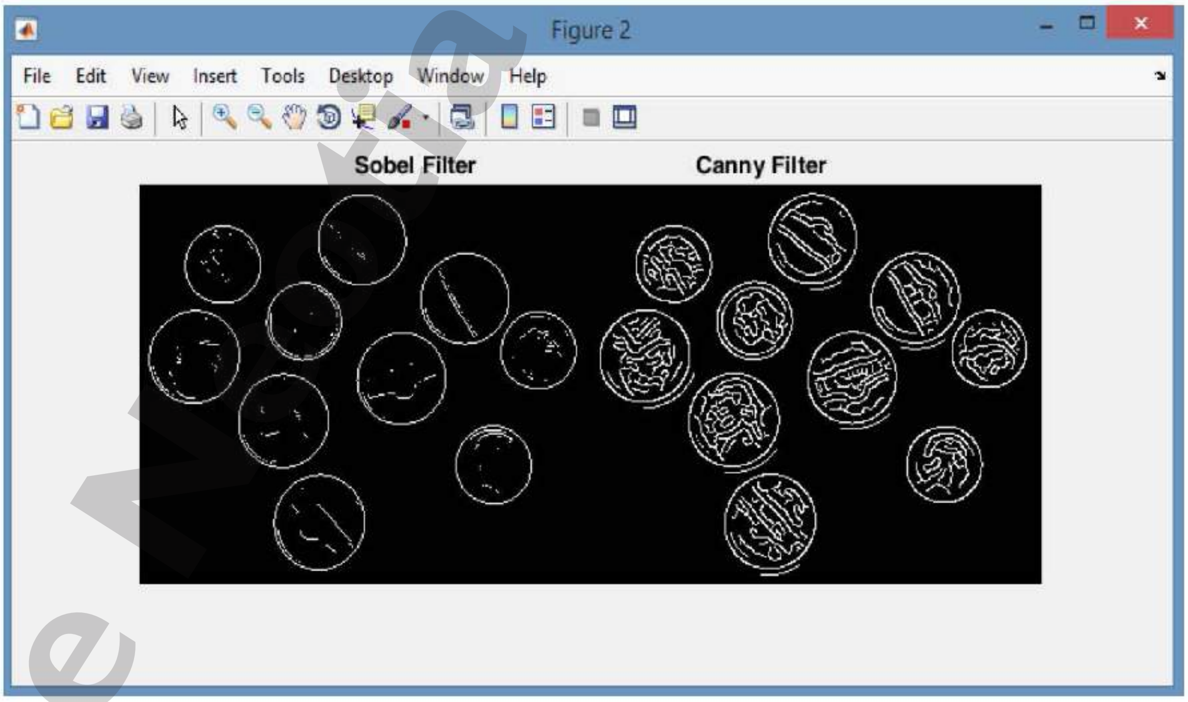


Figure 2
File Edit View Insert Tools Desktop Window Help
Sobel Filter
Canny Filter

Experiment – 6

Object: To perform the following operations in an image.

(a) erosion,

(b) dilation.

Software: MATLAB

Theory:

Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors.

The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion. This table lists the rules for both dilation and erosion.

Dilation and erosion are often used in combination to implement image processing operations. For example, the definition of a morphological opening of an image is an erosion followed by a dilation, using the same structuring element for both operations. We can combine dilation and erosion to remove small objects from an image and smooth the border of large objects.

Program:

(a) **erosion** clc; clear all; close all;

```
po=imgetfile;
I    =    imread(po);
originalBW = I;
se = strel('disk',11);
erodedBW        =        imerode(originalBW,se);
imshow(originalBW),
figure, imshow(erodedBW)
```

(b) dilation

```
clc; clear all; close all;
po=imgetfile;
I = imread(po);
se = strel('ball',5,5);
I2 = imdilate(I,se);
imshow(I), title('Original')
figure, imshow(I2),
title('Dilated')
```

Result: We have perform the erosion and dilation operations in an image.

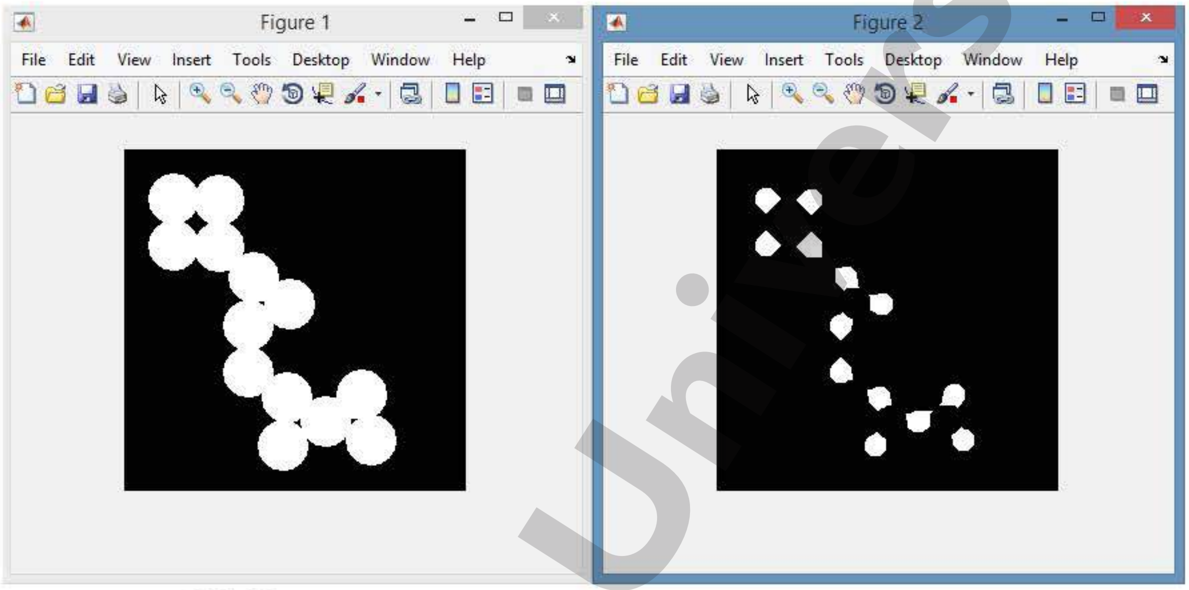


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
File Edit View Insert Tools Desktop Window Help

Original Image **Image After erosion operation**

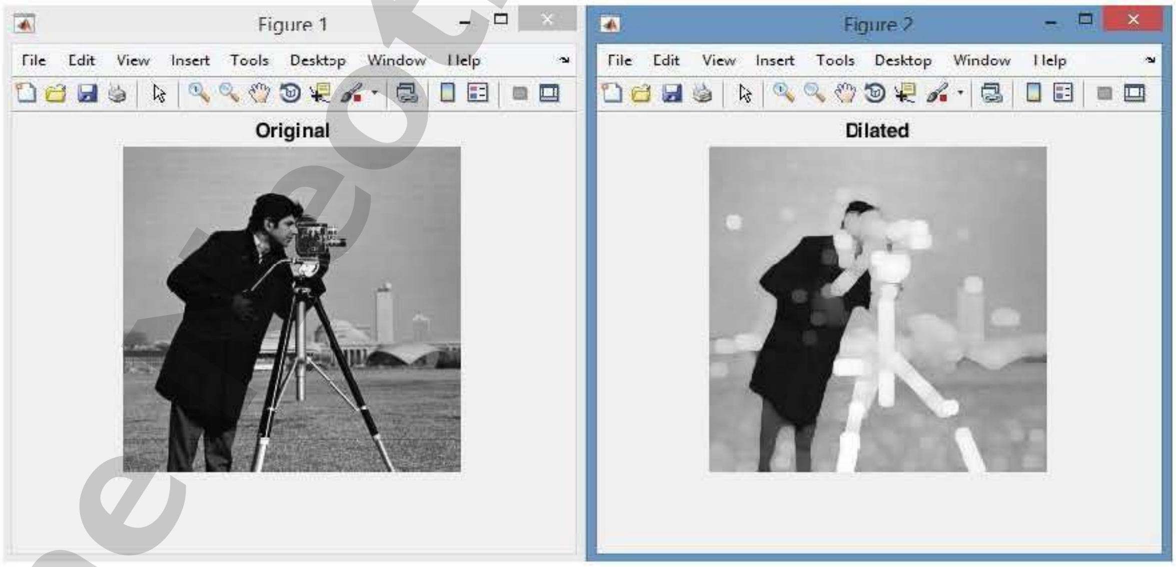


Figure 1
File Edit View Insert Tools Desktop Window Help
Original
Figure 2
File Edit View Insert Tools Desktop Window Help
Dilated

Experiment – 7

Object: To perform the following operations in an image.

(a) opening,

(b) closing.

Software: MATLAB

Theory:

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. According to Wikipedia, morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images. Morphological operations can also be applied to greyscale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest.

Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels. Some operations test whether the element "fits" within the neighborhood, while others test whether it "hits" or intersects the neighborhood:

Program:

(a) Opening

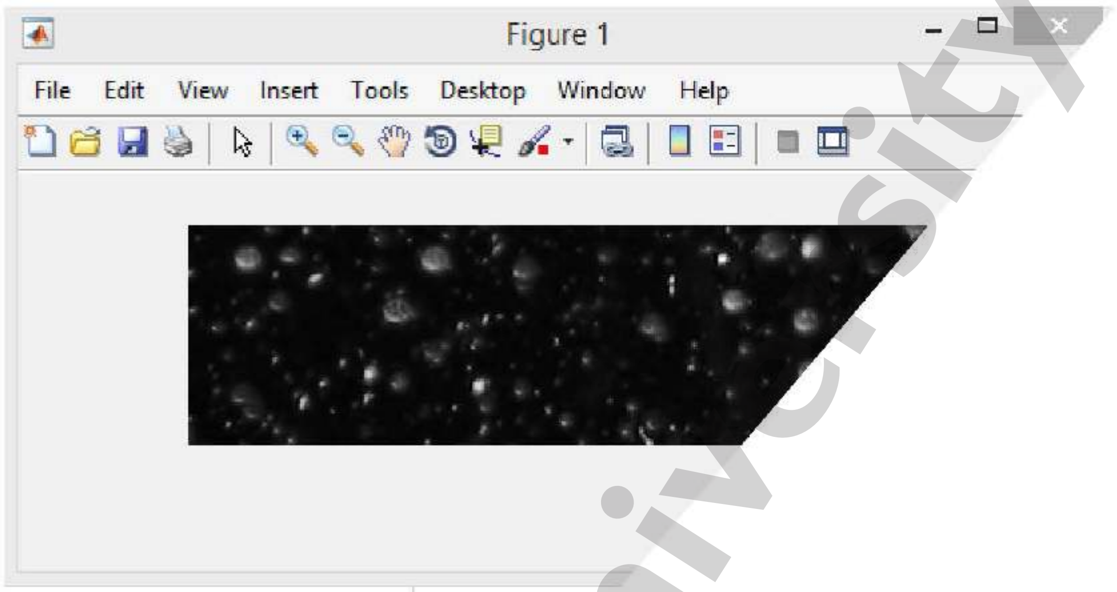
```
clc; clear all; close all;
po=imgetfile;
I = imread(po);

figure, imshow(I);

se = strel('disk',5);
afterOpening = imopen(I,se);
figure, imshow(afterOpening,[]);
```

(b) Closing

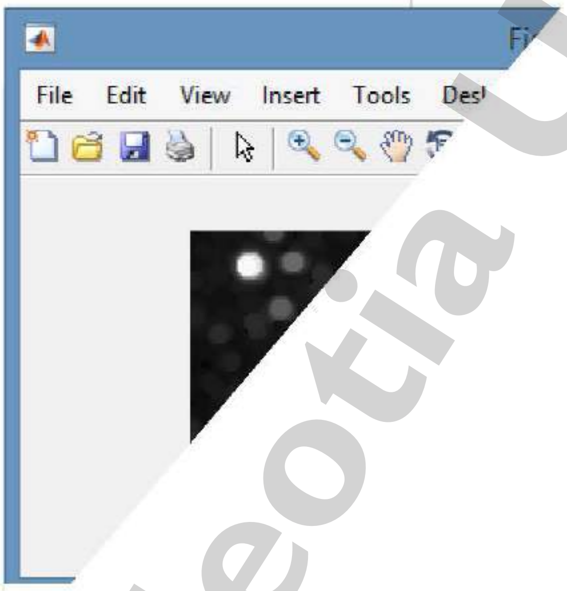
```
clc;clear all;close all;
```

```
po=imgetfile;
I = imread(po);
originalBW = I;
imshow(originalBW);
se = strel('disk',10);
closeBW = imclose(originalBW,se);
 figure, imshow(closeBW);
```

Result: We have perform the opening and closing operations in an image.



Original Image



EXPERIMEMT No-8

OBJECTIVE: Color Image Segmentation algorithm development

SOFTWARE REQURIED: MATLAB 7.7

THEORY:-

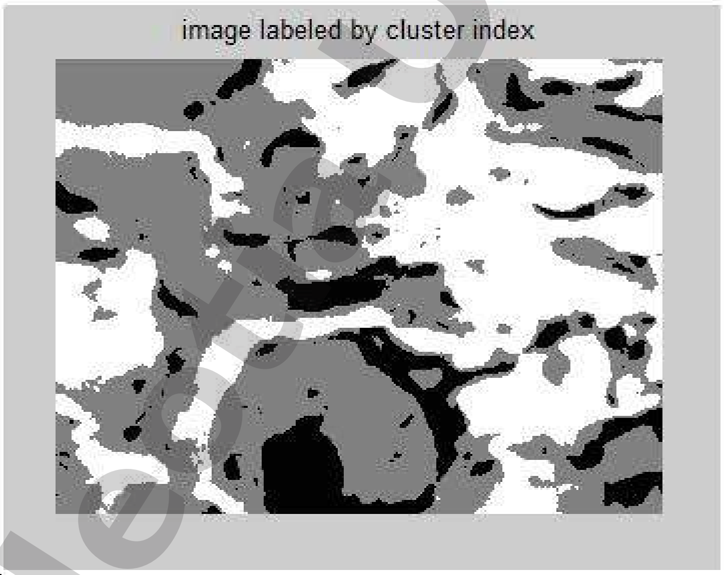
Image Segmentation:
Image segmentation is the division of an image into regions or categories, which correspond to different objects or parts of objects. Every pixel in an image is allocated to one of a number of these categories. A good segmentation is typically one in which:
pixels in the same category have similar grayscale of multivariate values and form a connected region, neighboring pixels which are in different categories have dissimilar values.
Segmentation is often the critical step in image analysis: the point at which we move from considering each pixel as a unit of observation to working with objects (or parts of objects) in the image, composed of many pixels. If segmentation is done well then all other stages in image analysis are made simpler.

A great variety of segmentation methods has been proposed in the past decades, and some categorization is necessary to present the methods properly here. A disjunct categorization does not seem to be possible though, because even two very different segmentation approaches may share properties that defy singular categorization. The categorization presented in this chapter is therefore rather a categorization regarding the emphasis of an approach than a strict division.
The following categories are used:
• Threshold based segmentation. Histogram thresholding and slicing techniques are used to segment the image. They may be applied directly to an image, but can also be combined with pre- and post-processing techniques.
• Edge based segmentation. With this technique, detected edges in an image are assumed to represent object boundaries, and used to identify these objects.
• Region based segmentation. Where an edge based technique may attempt to find the object boundaries and then locate the object itself by filling them in, a region based technique takes the opposite approach, by (e.g.) starting in the middle of an object and then "growing" outward until it meets the object boundaries.

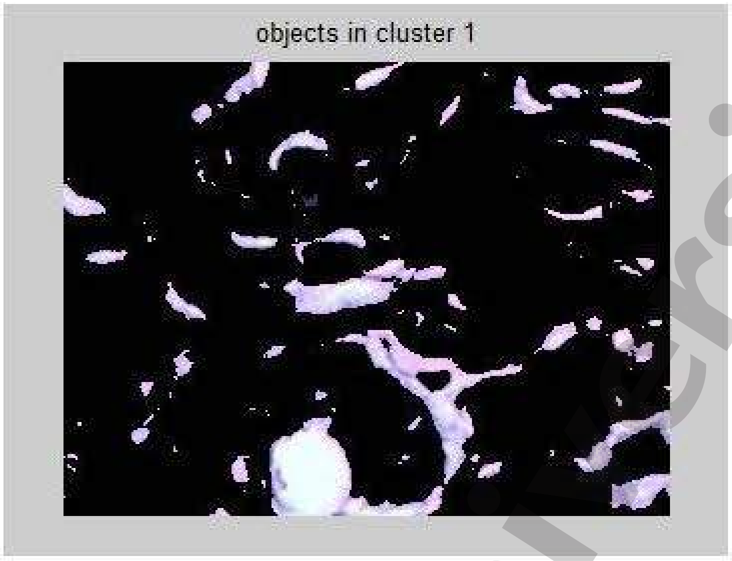
Matlab Code:

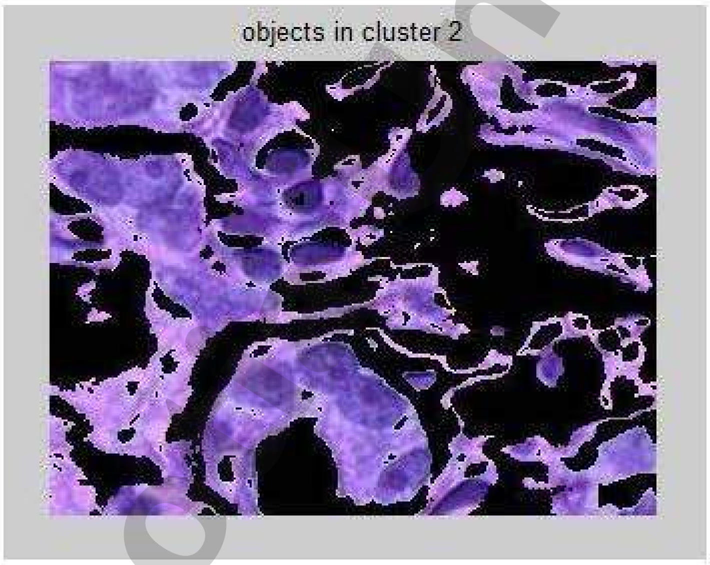
```
he = imread('hestain.png');
imshow(he), title('H&E image');
text(size(he,2),size(he,1)+15,...
    'Image courtesy of Alan Partin, Johns Hopkins University', ...
    'FontSize',7,'HorizontalAlignment','right');
 cform = makecform('srgb2lab');
lab_he = applycform(he,cform);
ab = double(lab_he(:,:,2:3));
```

```
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
 nColors = 3;
% repeat the clustering 3 times to avoid local minima
[cluster_idx, cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean', ...
                                 'Replicates',3);
pixel_labels = reshape(cluster_idx,nrows,ncols);
imshow(pixel_labels,[]), title('image labeled by cluster index');
segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1 1 3]);
 for k = 1:nColors
    color = he;
    color(rgb_label ~= k) = 0;
    segmented_images{k} = color;
end
 imshow(segmented_images{1}), title('objects in cluster 1');
imshow(segmented_images{2}), title('objects in cluster 2');
```



-


objects in cluster 1


objects in cluster 2

Experiment No-9

OBJECTIVE: Image filtering in spatial and frequency domain.

SOFTWARE REQURIED : MATLAB 7.5

Theory:-

SPATIAL DOMAIN

input image matrix
processing
output image matrix

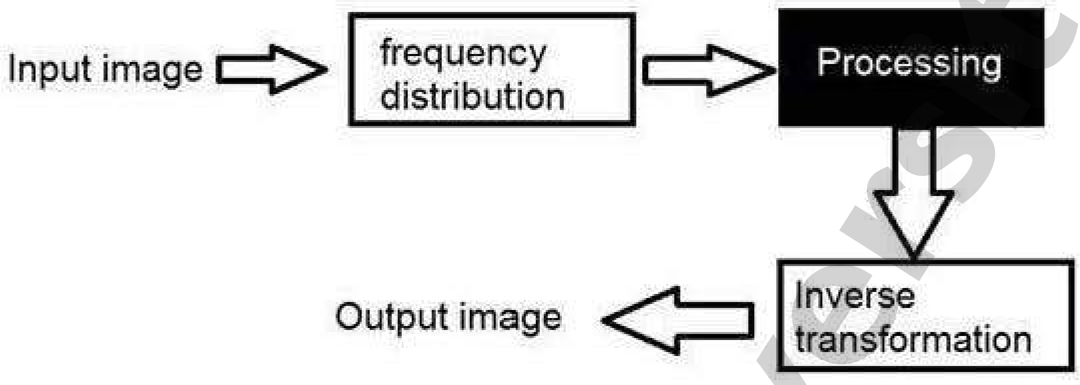
In simple spatial domain, we directly deal with the image matrix. Whereas in frequency domain, we deal an image like this.

- is manipulating or changing an image representing an object in space to enhance the image for a given application.
- Techniques are based on direct manipulation of pixels in an image
- Used for filtering basics, smoothing filters, sharpening filters, unsharp masking and laplacian

FREQUENCY DOMAIN

We first transform the image to its frequency distribution. Then our black box system perform what ever processing it has to performed , and the output of the black box in this case is not an image , but a transformation. After performing inverse transformation , it is converted into an image which is then viewed in spatial domain.

It can be pictorially viewed as



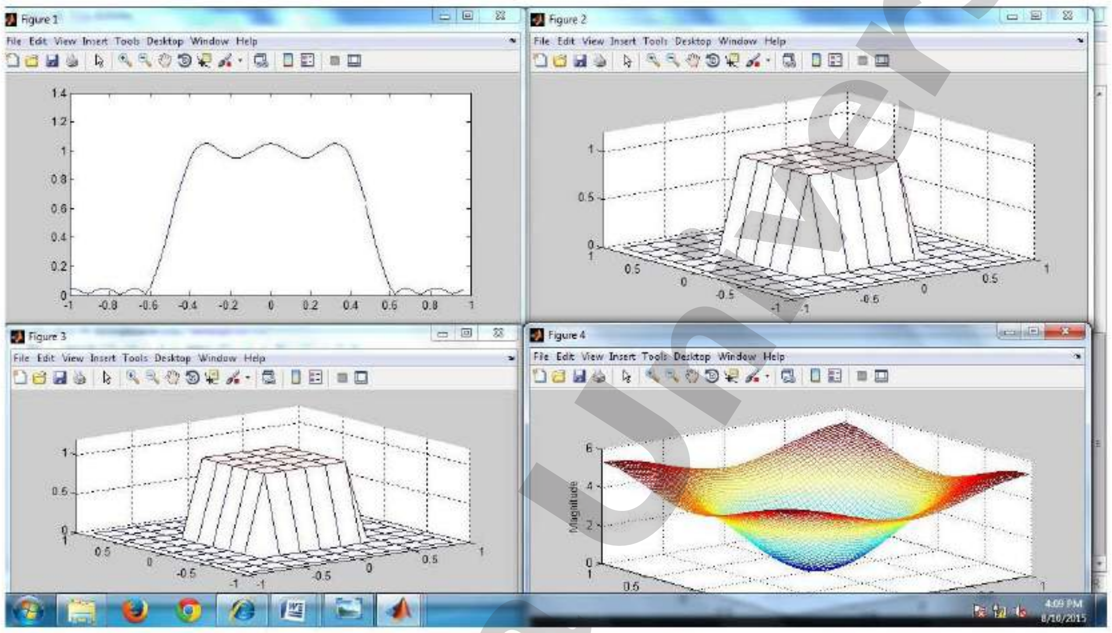
- Techniques are based on modifying the spectral transform of an image
- Transform the image to its frequency representation
- Perform image processing
- Compute inverse transform back to the spatial domain
- High frequencies correspond to pixel values that change rapidly across the image (e.g. text, texture, leaves, etc.)
- Strong low frequency components correspond to large scale features in the image (e.g. a single, homogenous object that dominates the image)

Program

```
b = remez(10,[0 0.4 0.6 1],[1 1 0 0])
h = ftrans2(b);
[H,w] = freqz(b,1,64,'whole');
colormap(jet(64))
plot(w/pi-1,fftshift(abs(H)))
figure, freqz2(h,[32 32])
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fsamp2(Hd);
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fwind1(Hd,hamming(11));
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
[f1,f2] = freqspace(25,'meshgrid');
Hd = zeros(25,25); d = sqrt(f1.^2 + f2.^2) < 0.5;
Hd(d) = 1;
mesh(f1,f2,Hd)
h =[0.1667    0.6667    0.1667
```

```
   0.6667   -3.3333    0.6667
   0.1667    0.6667    0.1667];
freqz2(h)
[H,f1,f2] = freqz2(h);
```

Result:



Experiment No-10

OBJECTIVE: Morphological operations in analyzing image structures.

SOFTWARE REQURIED : MATLAB 7.5

.

Theory:- ***The morphological transformations*** extract or modify the structure of the particles in an image. Such transformations can be used to prepare the particles for the quantitative analysis, for the analysis of the geometrical properties or for extracting the simplest modeling shapes and other operations. The morphological operations can also be used for expanding or reducing the particle dimensions, gap "filling" or closing inclusions, the averaging of the particle edges and others. The morphological transformations are separated in two main categories:

binary morphological functions, which are applied for binary images

gray-level morphological functions, which are applied for gray-level images

A binary image is an image which was segmented into an object region (which contains particles – typically the object pixels are coded by ones) and a background region (typically the background pixels are coded by zeros). The most simple segmentation process is by binary thresholding the gray-level images.

The basic morphological transformations include two types of processing: ***erosion*** and ***dilation***. The other types of transformations are obtained by combining these two operations.

Erosion

The ***erosion*** eliminates the isolated pixels from the background and erodes the boundaries of the object region, depending on the shape of the structuring element. For a given pixel *P0* we will consider the structuring element centered in *P0* and we will denote with *Pi* the neighboring pixels that will be taken into consideration (the ones corresponding to the coefficients of the structuring element having the value 1).

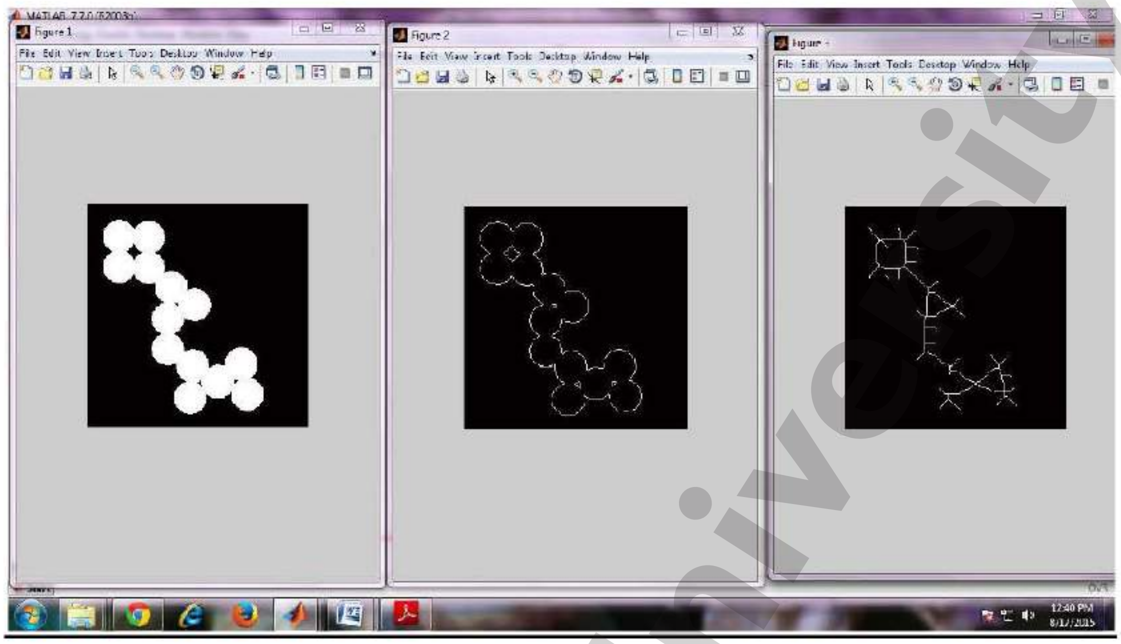
Dilation

The dilation process has the inverse effect of the erosion process, because the particle dilation is equivalent to the background erosion. This process eliminates the small and isolated gaps from the particles and enlarges the contour of the particles depending on the shape of the structuring element. For a given pixel *P0* we will consider the structuring element centered in *P0* and we will denote with *Pi* the neighboring pixels.

Program Code

```
BW = imread('circles.png');
imshow(BW);

BW2 = bwmorph(BW,'remove');
figure
imshow(BW2)

BW3 = bwmorph(BW,'skel',Inf);
figure
imshow(BW3)

BW1 = Array(imread('circles.png'));
figure
imshow(BW1)

BW2 = bwmorph(BW1,'remove');
figure
imshow(BW2)

BW3 = bwmorph(BW1,'skel',Inf);
figure
imshow(BW3)
```

Result:


MATLAB 7.7.0 (R2008b)
Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 2
File Edit View Insert Tools Desktop Window Help
Figure 3
File Edit View Insert Tools Desktop Window Help
12:40 PM
8/17/2015