Laboratory Journal

of

DIGITAL IMAGE PROCESSING

For completion of term work of 7th semester

curriculum program

Bachelor of Technology

In

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION

ENGINEERING

Dr. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY

Lonere-402 103, Tal. Mangaon, Dist. Raigad (MS)

INDIA

Digital Image Processing

List of Practicals

1. To read and display an image, to find size and information about the image.
2. Conversion of image file format.
3. To convert RGB image to gray scale, indexed and binary image.
4. To find average intensity and energy of image.
5. To convert RGB image to HSV image.
6. To implement negative of Binary, Intensity, RGB image using matlab function and c.
7. To generate chessboard.
8. Perform algebraic operations on two images.
9. Noise filteration from images.
10. Piecewise Linear Transformation Functions.
11. To perform histogram equalization.
12. Blurring and Deblurring of an Image.

EXPERIMENT NO.:- 01

Title	: To read and display an image, to find size and information about the image.
Class	: Final year B.Tech (Extc)
Batch	:
Roll No.	:
Date	**:**

Experiment No.-01

Aim: To read and display an image

To find size and information about the image.

Software: MATLAB 2009b.

Theory:

An image is 2-D light intensity function. It can be considered as matrix where row,column indices specify a point in the image and element value identifies gray level value at that point.

An RGB image has three channels: red, green, and blue. RGB channels roughly follow the color receptors in the human eye and are used in computer displays and image scanners.

A grayscale image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have equal intensity in RGB space and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

Functions used:

1) Clc : It clears all input and output from command window display giving you clean screen.

2) Clear all : It clears the workspace window.

3) Close all : Deletes all figures.

4) Imread : Reads image from graphics file, it reads a grayscale or color image from a file specified by string file name.
Syntax:-a=imread (filename with format)

5) Imshow : Displays the image like colour and grayscale image.
Syntax:-imshow (‘nature.gif’).

6) Disp **:** Display the data assigned to a variable.
Syntax:-disp(s).

7) Size **:** Display the size of image/matrix.
Syntax:-size(a).

8)imfinfo : It shows the all information about image.It also gives the file size height, width, format, resolution, mask, colortype etc.
Syntax-imfinfo(name of image).

MATLAB CODE:

```
clc;
close all;
clear all;
a=imread('cameraman.tif');
subplot(121);
imshow(a);
s=size(a);
info=imfinfo('cameraman.tif');
disp('information of cameraman.tif');
disp(info);

b=imread('peppers.png');
subplot(122);
imshow(b);
s1=size(a);
info1=imfinfo('peppers.png');
disp('information of peppers.png');
disp(info1);
```

OUTPUT:-

```
information of cameraman.tif
Filename: [1x67 char]
FileModDate: '04-Dec-2000 13:57:54'
FileSize: 65240
Format: 'tif'
FormatVersion: []
Width: 256
Height: 256
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: [77 77 42 0]
ByteOrder: 'little-endian'
NewSubFileType: 0
BitsPerSample: 8
Compression: 'PackBits'
PhotometricInterpretation: 'BlackIsZero'
StripOffsets: [8x1 double]
SamplesPerPixel: 1
```

RowsPerStrip: 32
StripByteCounts: [8x1 double]
XResolution: 72
YResolution: 72
ResolutionUnit: 'None'
Colormap: []
PlanarConfiguration: 'Chunky'
TileWidth: []
TileLength: []
TileOffsets: []
TileByteCounts: []
Orientation: 1
FillOrder: 1
GrayResponseUnit: 0.0100
MaxSampleValue: 255
MinSampleValue: 0
Thresholding: 1
Offset: 64872
ImageDescription: [1x112 char]

information of peppers.png
Filename: [1x65 char]
FileModDate: '16-Dec-2002 06:10:58'
FileSize: 287677
Format: 'png'
FormatVersion: []
Width: 512
Height: 384
BitDepth: 24
ColorType: 'truecolor'
FormatSignature: [137 80 78 71 13 10 26 10]
Colormap: []
Histogram: []
InterlaceType: 'none'
Transparency: 'none'
SimpleTransparencyData: []
BackgroundColor: []
RenderingIntent: []
Chromaticities: []
Gamma: []
XResolution: []

YResolution: []
ResolutionUnit: []
XOffset: []
YOffset: []
OffsetUnit: []
SignificantBits: []
ImageModTime: '16 Jul 2002 16:46:41 +0000'
Title: []
Author: []
Description: 'Zesty peppers'
Copyright: 'Copyright The MathWorks, Inc.'
CreationTime: []
Software: []
Disclaimer: []
Warning: []
Source: []
Comment: []
OtherText: []

Conclusion: Hence, we have studied to read and display image and also to find size and information about the image.

EXPERIMENT NO:-

Title : Conversion of image file format.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

EXPERIMENT NO. -

Aim: To read and display image and to change file format in

(a) .JPG (b) .BMP (c) .PNG (d) .TIF

Software: MATLAB R2009a

Theory:

Image file formats are standardized means of organizing and storing digital images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color.

Image file sizeexpressed as the number of bytes, increases with the number of pixels composing an image, and the colour depth of the pixels. The greater the number of rows and columns, the greater the image resolution, and the larger the file. Also, each pixel of an image increases in size when its colour depth increases, an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter known as true-color.

Including proprietary types, there are hundreds of image file types. The JPEG, GIF, Exif (Exchangeable image file format), TIFF (Tagged Image File Format) PNG (Portable Network Graphics),BMP formats are most often used to display images on the Internet. These graphic formats are listed and briefly described below, separated into the two main families of graphics: raster and vector.

This table shows the file formats that you can import and export from the MATLAB application.

File Format	Discription	Exten-sion	Returns
TIFF	Tagged Image File Format	.tif or .tiff	True color, grayscale, or indexed image(s)
JPEG	Joint Photographic Experts Group	.jpg or .jpeg	True color or grayscale image
GIF	Grphics Interchange format	.gif	Indexed image
BMP	Windows Bitmap	.bmp	True color or indexed image
PNG	Portable Network Graphics	.png	True color, grayscale, or indexed image
XWD	X-windows Dump	.xwd	Indexed image

BMP

The BMP file format (Windows bitmap) handles graphics files within the Microsoft Windows OS. Typically, BMP files are uncompressed, hence they are large; the advantage is their simplicity and wide acceptance in Windows programs.

TIFF:

The Tagged Image File Format or TIFF is one of the most comprehensive image format. It can store multiple images per file.It allows Binary, Grayscale, True color or Indexed images.

JPEG:-

The compression method used by GIF and PNG are lossless: the Original information can be recovered completely. The JPEG algorithm uses lossy compression, in which not all original data can be recovered. Such a methods result in much higher compression rate, and JPEG images are in general much smaller than GIF and PNG images.

PNG:

The PNG format has used to overcome disadvantages of GIF format. Specifically PNG does not rely on any patented algorithm, and it supports more images than GIF. PNG supports grayscale, true color and indexed images.

Conversion of image file format:

To convert the image file format, read the image using and change the file format of image in imwrite command, specifying appropriate format. To illustrate, this example uses the imread to read the image in TIFF format in workspace. Then writes the TIFF image to a file using PNG format.

Functions used:

1) **imwrite**:- Writes the image to the specified location by the file name in format specified.

Syntax: b= imwrite(a,filename)

MATLAB Code:

```
clc;
close all;
clear all;

a=imread('cameraman.tif');
b=a;

imwrite(b,'cameraman.jpg');
info1=imfinfo('cameraman.jpg');

imwrite(b,'cameraman.jpeg');
info2=imfinfo('cameraman.jpeg');

imwrite(b,'cameraman.bmp');
info3=imfinfo('cameraman.bmp');
```

OUTPUT:

```
>> info1

info1 =

Filename: 'cameraman.jpg'
FileModDate: '24-Sep-2014 09:04:14'
FileSize: 10717
Format: 'jpg'
FormatVersion: ''
Width: 256
Height: 256
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: ''
NumberOfSamples: 1
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: {}

>> info2

info2 =

Filename: 'cameraman.jpeg'
FileModDate: '24-Sep-2014 09:04:15'
FileSize: 10717
Format: 'jpg'
FormatVersion: ''
```

Width: 256
Height: 256
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: ''
NumberOfSamples: 1
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: {}

>> info3

info3 =

Filename: 'cameraman.bmp'
FileModDate: '24-Sep-2014 09:04:16'
FileSize: 66614
Format: 'bmp'
FormatVersion: 'Version 3 (Microsoft Windows 3.x)'
Width: 256
Height: 256
BitDepth: 8
ColorType: 'indexed'
FormatSignature: 'BM'
NumColormapEntries: 256
Colormap: [256x3 double]
RedMask: []
GreenMask: []
BlueMask: []
ImageDataOffset: 1078
BitmapHeaderSize: 40
NumPlanes: 1
CompressionType: 'none'
BitmapSize: 65536
HorzResolution: 0
VertResolution: 0
NumColorsUsed: 256
NumImportantColors: 0

Conclusion: Hence we have studied to change the file format of an image.

EXPERIMENT NO:

Title : To convert RGB image to gray scale, indexed and binary image.

Class : Final year B.Tech (Extc)

Batch :

Roll No. :

Date :

EXPERIMENT NO.-

Aim: To convert RGB image to gray scale, indexed and binary image.

To study conversion of image.

To find histogram of images.

Software: MATLAB 2009b.

Theory:

Image is a pictorial representation or the collection of intensity level values connected in array of m x n matrix. Image data is random in form. Thus, it is necessary to use statistical technique to analyse and process image.

Histogram:

Histograms are the basis for numerous spatial domain processing techniques. The histogram of a digital image with intensity levels in the range [0,L-1] is a discrete function

$$\mathrm{h}(r_k) = n_k$$

Where, $r_k$is the *k*th intensity value and n_k is the number of pixels in the image with intensity r_k. A normalized histogram is given by

$\mathrm{p}(r_k) = \frac{n_k}{\mathrm{MN}}$for *k*=0,1,2,…….L-1

Where, M and N are the row and column dimensions of the image.

The sum of all components of a normalized histogram is equal to 1. Histograms are simple to calculate in software and also lend themselves to economic hardware implementations, thus making them a popular tool for real-time image processing.

Gray Scale Images:are images which carry only the intensity information. These images are commonly named as a Black and white. Their intensity levels range from 0 to 255: 0 being the darkest whereas the 255 the brightest pixel.

RGB Images: are images which carry the Red, Green and Blue components. Each component add up and constitute the value of the pixel.

Indexed Images:are images encoded between values 0 to 1 due to many reasons including tosave memory and file storage.

Matlab have built in functions to convert from one format to another.

Image importing:

To import an image from any supported graphics image file format, in any of the supported bit depths, use the imread function. This example reads a truecolor image into the MATLAB workspace as the variable RGB.

```
RGB = imread('football.jpg');
```

If the image file format uses 8-bit pixels, imread stores the data in the workspace as a uint8 array.

Size of Image:

Image data can be either indexed or true color. An indexed image stores colors as an array of indices into the figure color map. A true color image does not use a color map; instead, the color values for each pixel are stored directly as RGB triplets. In MATLAB graphics, the data property of a true color image object is a three-dimensional (m-by-n-by-3) array. This array consists of three m-by-n matrices (representing the red, green, and blue color planes) concatenated along the third dimension.

Functions used:

1) rgb2gry : Converts RGB image to grayscale image by eliminating image hue and saturation while retaining luminance.
Syntax:-rgb2gray(‘imagename’).

2) Imhist : Displays histogram of image data.
Syntax:-imhist(A)
A=imread(‘nature.jpg’)
B=imhist(A)
Imshow(B)

3) im2bw : Converts image to binary image based on threshold.
Syntax:- A=imread(‘nature.jpg’)
B=im2bw(A)
Imshow(B)

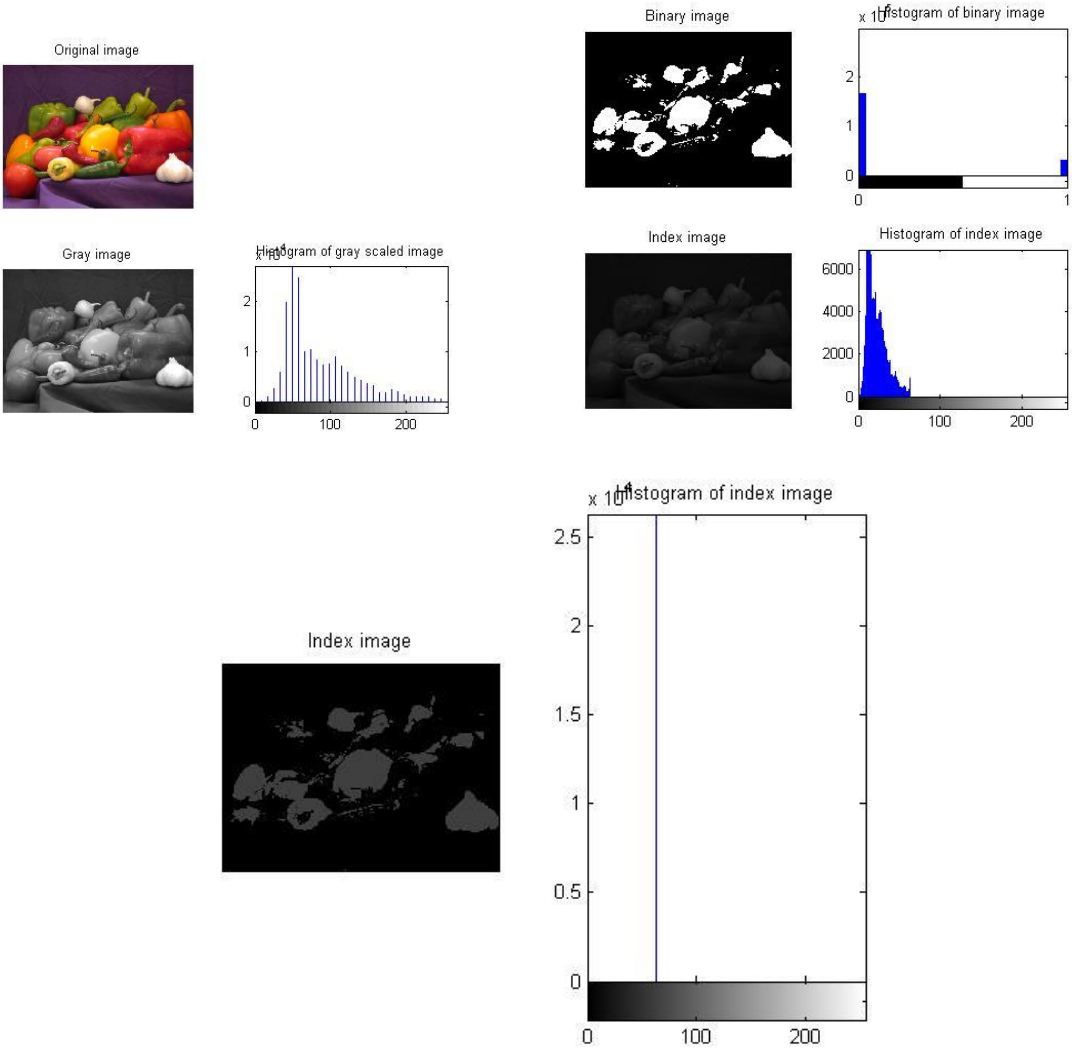
4) gray2id : Converts the grayscale image to indexed image.
Syntax:-[X,map]=gray2ind(I,n)
A=imread(‘nature.jpg’)
[X,map]=gray2ind(A,16)
Imshow(x,map)

5) rgb2ind : Converts RGB image to indexed image.
Syntax: A=imread(‘nature.jpg)
[X,map]=rgb2ind(A,16)
Imshow(x,map)

MATLAB Code:

```
clc;
clear all;
close all;

a = imread('peppers.png');
figure
subplot(221);
imshow(a);
title('Original image');

% rgb image is converted into gray scale image
J=rgb2gray(a);
subplot(223);
imshow(J);
title('Gray image');
subplot(224);
imhist(J,32);
title('Histogram of gray scaled image');

% rgb image is converted into binary image
figure
BW=im2bw(a);
subplot(221);
imshow(BW);
title('Binary image');
subplot(222);
imhist(BW);
title('Histogram of binary image');

% Convert grayscale or binary image to indexed image
e=gray2ind(J,64);
subplot(223);
imshow(e);
title('Index image');
subplot(224);
imhist(e);
title('Histogram of index image');

% Convert grayscale or binary image to indexed image
figure
```

```
e=gray2ind(BW,64);
subplot(221);
imshow(e);
title('Index image');
subplot(222);
imhist(e);
title('Histogram of index image');
```



Conclusion:

Hence, we have studied to read and display image. Also we have learnt the conversion of RGB image to black and white, gray and indexed image and also the plotting of histogram for each.

EXPERIMENT NO:-

Title : To find average intensity and energy of image.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

EXPERIMENT NO.-

Aim: To find average intensity and energy of image.

Software: MATLAB 2009b.

Theory:

Average intensity is the intensity of monochrome image of function at the co-ordinates(x,y) is the gray level of image at thet point where

$0<f(x,y)<\infty$

Thus, average intensity gives the average value of intensity of image.

It is given by

$$g(x,y)=\Sigma Gi(x,y)/N \qquad 1\leq i\leq N$$

where,

g(x,y)=average intensity

Gi(x,y)=intensity of pixel.

N=no. of pixel.

Average intensity gives the idea whether the image is dark or bright.

Energy of image is given by,

$$E=\Sigma(Gi(x,y))^2 \qquad 1\leq i\leq N$$

Where,

E=energy of image.

Gi=intensity of individual pixel.

Functions used:

1) **UINT32:** converts the elements of array image into unsigned 32 bit integer.
   Syntax- img=uint32(img).

MATLAB Code:

```
clc;
clear al;
close all;

% energy & average intensity of gray scale image
img=imread('cameraman.tif');
s=size(img);
subplot(121);
imshow(img);
title('original image');

%img=double(img);
img=uint32(img);
sum=0;
for i=1:s(1)
```

```
    for j=1:s(2)
        sum = sum + img(i,j);
    end
end
np=s(1)*s(2);
avg=sum/np;
disp('average intensity is ');
disp(avg);

sum=0;
for i=1:s(1)
    for j=1:s(2)
        sum = sum + (img(i,j)*img(i,j));
    end
end
disp('energy is ');
disp(sum);

% energy & average intensity of colour image
img=imread('peppers.png');
s=size(img);
subplot(122);
imshow(img);
title('original image');

%img=double(img);
img=uint32(img);
sum=0;
for i=1:s(1)
    for j=1:s(2)
        sum = sum + img(i,j);
    end
end
np=s(1)*s(2);
avg=sum/np;
disp('average intensity is ');
disp(avg);

sum=0;
for i=1:s(1)
    for j=1:s(2)
        sum=sum+ (img(i,j)*img(i,j));
    end
end
```

```
disp('energy is ');
disp(sum);
```

OUTPUT:

```
average intensity is
        119

energy is
  1178464030

average intensity is
        121

energy is
  3738717666
```

Conclusion: Thus, we have studied average intensity and energy of image calculation.

EXPERIMENT NO.:-

Title : To convert RGB image to HSV image.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

EXPERIMENT NO.-

Aim: To convert RGB image into HSV.

Software: MATLAB 2009b

Theory:

An RGB image has three channels: red, green, and blue. RGB channels roughly follow the color receptors in the human eye and are used in computer displays and image scanners.

The HSV model separates Hue (color) Saturation (amount of color) and Value (intensity). It is much more useful for manipulation than the RGB space and also for segmenting images by color.HSV is the most common cylindrical co-ordinate representations of points in an RGB. In each cylinder, the angle around the central vertical axis corresponds to hue, the distance from the axis corresponds to saturation.

Functions used:

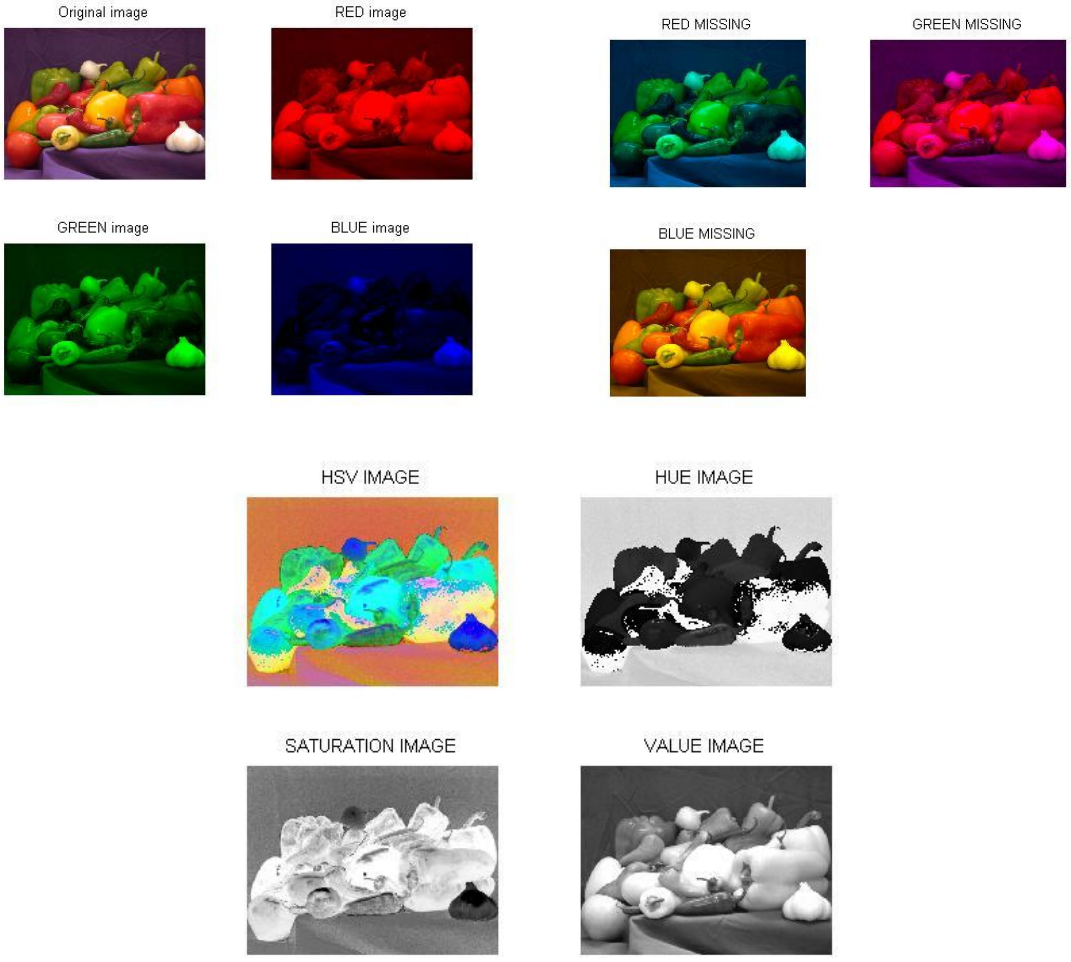
1) **RGB2HSV**: converts rgb image into hsv.
   Syntax- HSV=rgb2hsv(a).

MATLAB Code:

```
clc;
close all;
clear all;

a=imread('peppers.png');
figure
subplot(2,2,1);
imshow(a);
title('Original image');

R=a;
R(:,:,2)=0;
R(:,:,3)=0;
subplot(2,2,2);
imshow(R);
title('RED image');

G=a;
G(:,:,3)=0;
G(:,:,1)=0;
subplot(2,2,3);
imshow(G);
```

```
title('GREEN image');

B=a;
B(:,:,1)=0;
B(:,:,2)=0;
subplot(2,2,4);
imshow(B);
title('BLUE image');

R=a;
R(:,:,1)=0;
figure
subplot(2,2,1);
imshow(R);
title('RED MISSING');

G=a;
G(:,:,2)=0;
subplot(2,2,2);
imshow(G);
title('GREEN MISSING');

B=a;
B(:,:,3)=0;
subplot(2,2,3);
imshow(B);
title('BLUE MISSING');

HSV=rgb2hsv(a);
figure
subplot(2,2,1)
imshow(HSV);
title('HSV IMAGE');

H=HSV(:,:,1);
subplot(2,2,2)
imshow(H);
title('HUE IMAGE');

S=HSV(:,:,2);
subplot(2,2,3)
imshow(S);
title('SATURATION IMAGE');
```

```
V=HSV(:,:,3);
subplot(2,2,4)
imshow(V);
title('VALUE IMAGE');
```



Conclusion: Thus we have studied conversion of color image into hsv.

EXPERIMENT NO.:-

Title : To implement negative of Binary, Intensity, RGB image using matlab function and c.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

EXPERIMENT NO.-

Aim: To implement negative of Binary, Intensity, RGB image using matlab function and c.

Software: MATLAB R2009a

Theory:

The negative of an image with gray level in the range [0,-1] is obtained by using negative transformation which is given by expression s=L-1-r.

Reversing the intensity levels of an image in this manner produces equivalent of a photographic negative. This type of processing is practically suited for enhancing white or grey level detail embedded in dark region of image externally to which block areas are dominant in size. It is used for displaying medical image.

Functions used:

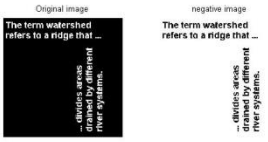
1. **imcomplement :** In the complement of a binary image, zeros become ones and ones become zeros; black and white are reversed. In the complement of an intensity or RGB image, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images).

Syntax- IM2 = imcomplement(IM)

MATLAB Code:

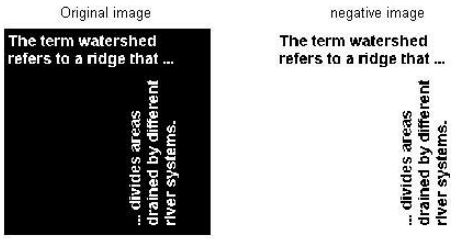
```
%negative of Binary image using inbuilt functions
clc;
close all;
clear all;

bw = imread('text.png');
subplot(1,2,1);
imshow(bw);
title('Original image');

bw2 = imcomplement(bw);
subplot(1,2,2);
imshow(bw2);
title('negative image');
```

OUTPUT:



```
%negative of Binary image using c
clc;
clear all;
close all;
figure
a=imread('text.png');
subplot(221);
imshow(a);
title('original image');
s=size(a);
for i=1:s(1)
   for j=1:s(2)
      y(i,j)=1.0-a(i,j);
   end;
end;
subplot(222);
imshow(y);
```

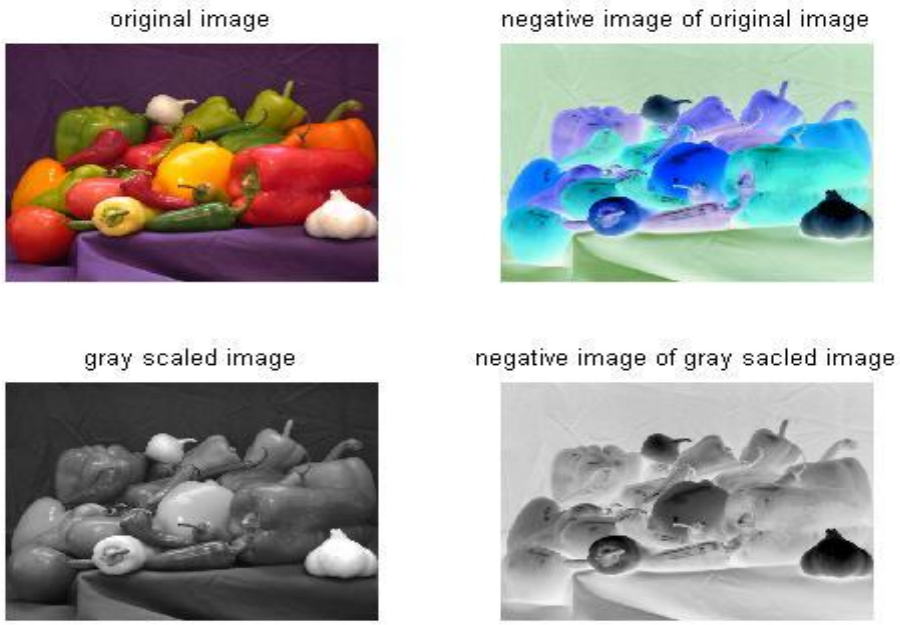
OUTPUT:




```
%negative of RGB image using inbuilt functions
clc;
close all;
clear all;
x1=imread('peppers.png');
subplot(2,2,1);
imshow(x1);
title('original image');

c1=imcomplement(x1);
subplot(2,2,2);
imshow(c1);
```

```
title('negative image of original image');

x=rgb2gray(x1);
subplot(2,2,3);
imshow(x);
title('gray scaled image');

c=imcomplement(x);
subplot(2,2,3);
imshow(c);
title('negative image of graysacled image');
```

OUTPUT:



Conclusion:

Thus we have implemented the negative of binary image, intensity image and RGB image.

EXPERIMENT NO.:-

Title : To generate chessboard.

Class : Final year B.Tech (Extc)

Batch :

Roll No. :

Date :

EXPERIMENT NO. -

Aim: To generate chessboard.

Software: MATLAB R2009a

Theory: A chessboard is the type of checkerboard used in the board gamechess, and consists of 64 squares (eight rows and eight columns) arranged in two alternating colors (light and dark). The colors are called "black" and "white" (or "light" and "dark"), although the actual colors are usually dark green and buff for boards used in competition, and often natural shades of light and dark woods for home boards. Materials vary widely; while wooden boards are generally used in high-level games, vinyl and cardboard are common for low-level and informal play. Decorative glass and marble boards are available but not usually accepted for sanctioned games.

Functions used:

1. zeros: zeros(m, n,...,classname) or zeros([m,n,...],classname) is an m-by-n-by-... array of zeros of data type classname. classname is a string specifying the data type of the output.
   Syntax- B = zeros(n).
2. logical: returns an array that can be used for logical indexing or logical tests.
   syntax- K = logical(A)
3. imresize: returns image B that is scale times the size of A. The input image A can be a grayscale, RGB, or binary image. If scale is between 0 and 1.0, B is smaller than A. If scale is greater than 1.0, B is larger than A.
   syntax- B = imresize(A, scale).

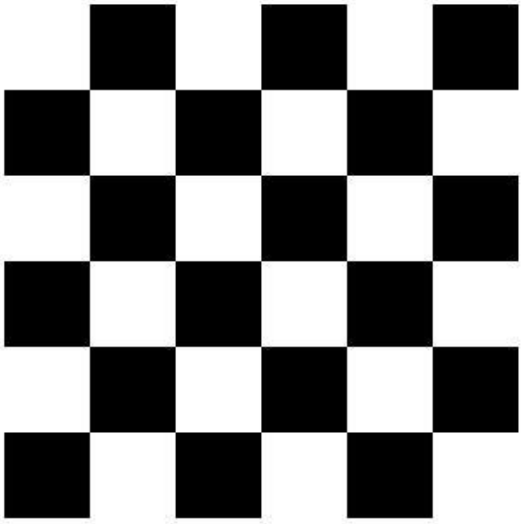
MATLAB Code:

```
clc;
close all;
clear all;

m=input('Enter no of rows=m= ');
n=input('Enter no of columns=n= ');
if(m==n)
   z=zeros(m,n);
   s=size(z);
   r=1;
   p=zeros(1,n);
   for i=1:n
      p(r)=r;
      r=r+2;
   end
   t=2;
   q=zeros(1,n);
   for i=1:n
      q(t)=t;
      t=t+2;
   end
   for i=1:s(1)
      for j=1:s(1)
         if(p(j)==j&&~q(i))
            z(i,j)=1;
         end
         if(q(i)==i&&~p(j))
             z(i,j)=1;
         end
      end
   end
disp(z)
l=logical(z);
b=imresize(l,50,'nearest');
imshow(b);
else
   input('Error')
end
```

OUTPUT:

Enter no of rows=m= 6
Enter no of columns=n= 6

1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	1	0	1



Conclusion: Hence we have studied to generate chessboard.

EXPERIMENT NO:-

Title : Perform arithmatic operations on two images.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

Experiment No.-

Aim: To perform arithmetic operations on two images.

Software: MATLAB R2009a

Theory: In mathematics, an arithmeticoperation is any one of the operationsaddition, subtraction, multiplication, division, raising to an integer power, and taking roots (fractional power). Algebraic operations are performed on an algebraic variable, term or expression,[1] and work in the same way as arithmetic operations.

Functions Used:

1. **imadd:** Syntax- Z = imadd(X,Y)
   Z = imadd(X,Y) adds each element in array X with the corresponding element in array Y and returns the sum in the corresponding element of the output array Z.
2. **imsubtract:** Syntax- Z = imsubtract(X,Y).
   Z = imsubtract(X,Y) subtracts each element in array Y from the corresponding element in array X and returns the difference in the corresponding element of the output array Z.
3. **immultiply:** Syntax- Z = immultiply(X,Y)
   Z = immultiply(X,Y) multiplies each element in array X by the corresponding element in array Y and returns the product in the corresponding element of the output array Z.
4. **imdivide**: Syntax-Z = imdivide(X,Y)
   Z = imdivide(X,Y) divides each element in the array X by the corresponding element in array Y and returns the result in the corresponding element of the output array Z.

MATLAB Code:

Add two images together:

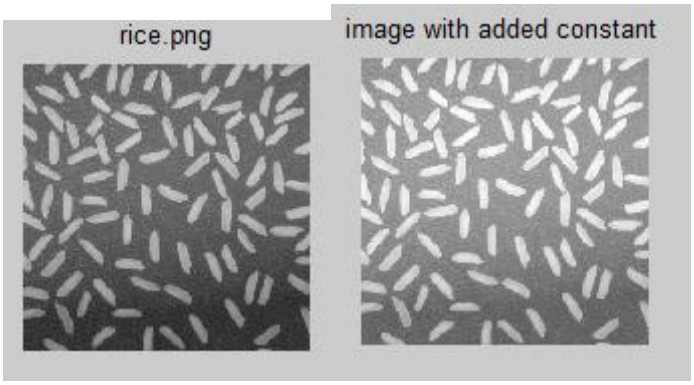
```
I = imread('rice.png');
I=imresize(I,[256 256]);
J = imread('cameraman.tif');
J=imresize(J,[256 256]);
K = imadd(I,J);
figure
subplot(2,2,1);
imshow(I);
title('rice.png')
subplot(2,2,2);
imshow(J);
title('cameraman.tif')
subplot(2,2,3);
imshow(K);
title('Addition of two images');
 M=imsubtract(K,I);
subplot(2,2,4);
```

```
imshow(M);
```

Add a constant to an image:

```
L = imread('rice.png');
Iplus50 = imadd(L,50);
figure
subplot(2,2,1);
imshow(L);
title('rice.png')
subplot(2,2,2);
imshow(Iplus50);
title('image with added constant')
```

OUTPUT:-



MATLAB CODE:-

Estimate and subtract the background of the rice image:

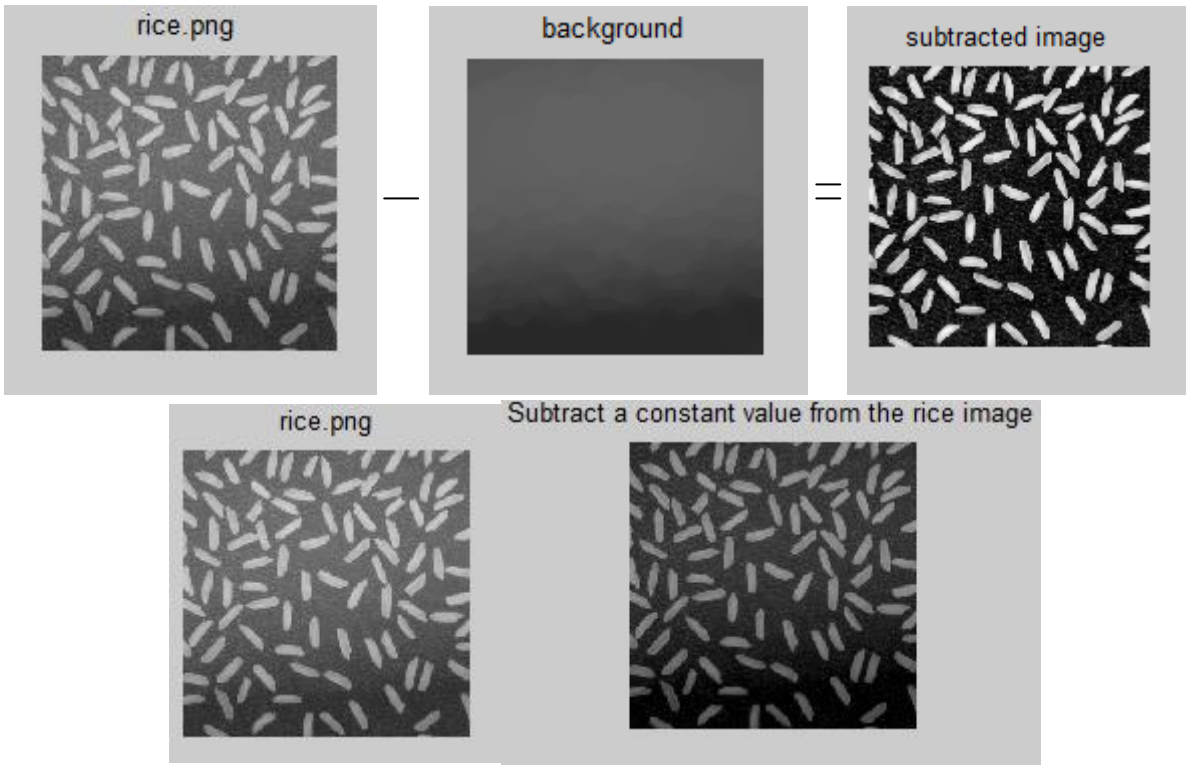
```
I = imread('rice.png');
background = imopen(I,strel('disk',15));
Ip = imsubtract(I,background);
figure
subplot(2,2,1);
imshow(I);
title('rice.png');
subplot(2,2,2);
imshow(background);
```

```
title('background');
subplot(2,2,3);
imshow(Ip,[]);
title('subtracted image');
```

Subtract a constant value from the rice image:

```
I1 = imread('rice.png');
Iq = imsubtract(I,50);

figure
subplot(2,2,1);
imshow(I1);
title('rice.png');
subplot(2,2,2);
imshow(Iq);
title('Subtract a constant value from the rice image');
```

OUTPUT:-



MATLAB CODE:-

Multiply two uint8 images with the result stored in a uint16 image:

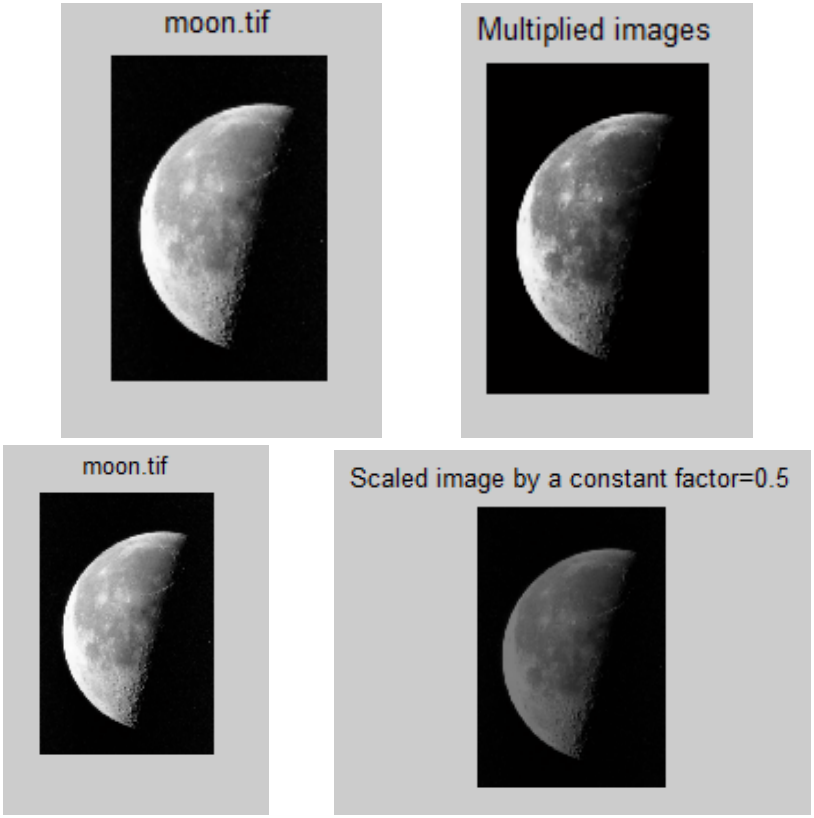
```
I = imread('moon.tif');
I16 = uint16(I);
J = immultiply(I16,I16);
figure
subplot(2,2,1);
```

```
imshow(I);
title('moon.tif')
subplot(2,2,2);
imshow(J);
title('Multiplied images')
```

Scale an image by a constant factor:

```
I1 = imread('moon.tif');
J1 = immultiply(I1,0.5);
figure
subplot(2,2,1);
imshow(I1);
title('moon.tif')
subplot(2,2,2);
imshow(J1);
title('Scaled image by a constant factor=0.5')
```

OUTPUT:-



MATLAB CODE:-

Estimate and divide out the background of the rice image:

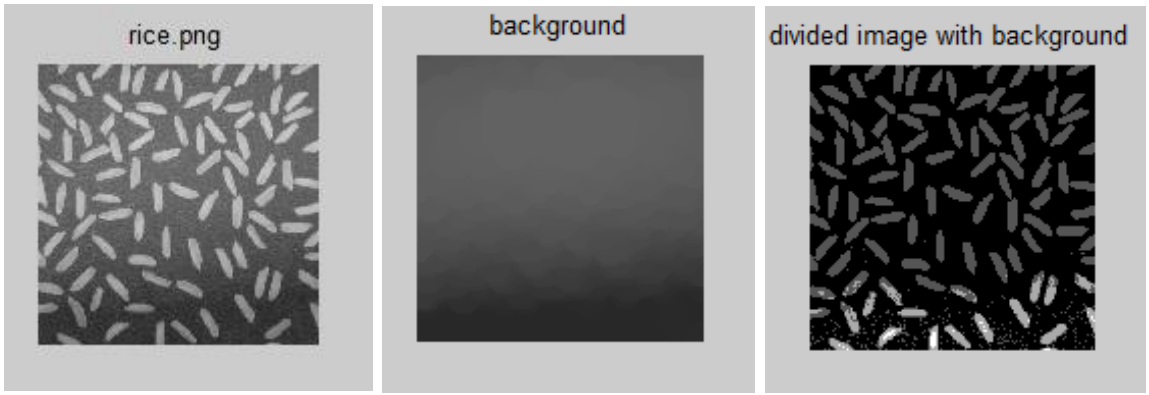
```
I = imread('rice.png');
background = imopen(I,strel('disk',15));
Ip = imdivide(I,background);
```

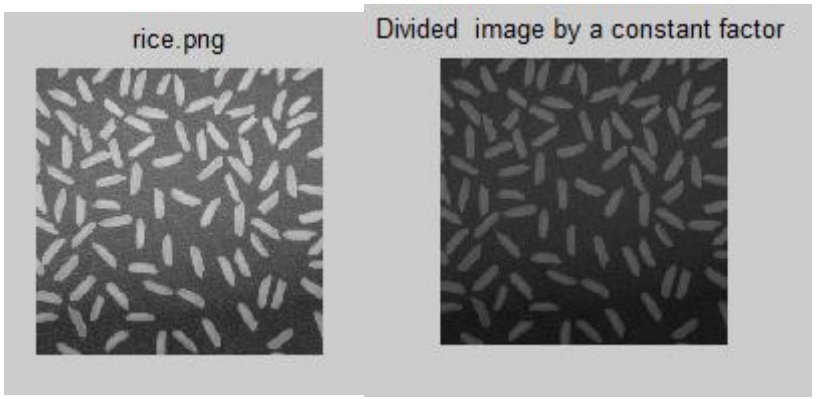
```
figure
subplot(2,2,1);
imshow(I);
title('rice.png');
subplot(2,2,2);
imshow(background);
title('background');
subplot(2,2,3);
imshow(Ip,[]);
title('divided image with background');
```

Divide an image by a constant factor:

```
I1 = imread('rice.png');
J = imdivide(I,2);
figure
subplot(2,2,1);
imshow(I1);
title('rice.png');
subplot(2,2,2);
imshow(J);
title('Divided  image by a constant factor');
```

OUTPUT:-





Conclusion: Hence we performed different algebraic operations on two images.

EXPERIMENT NO:-

Title : Addition & removal of noise using different filters.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

Experiment No.-

Aim: To add and remove noise using different type of filters.

Software: MATLAB R2009a.

Theory:

Noise reduction is the process of removing noise from a signal. All recording devices, both analog or digital, have traits which make them susceptible to noise. Noise can be random or white noise with no coherence, or coherent noise introduced by the device's mechanism or processing algorithms.

In the case of photographic film and magnetic tape, noise (both visible and audible) is introduced due to the grain structure of the medium. In photographic film, the size of the grains in the film determines the film's sensitivity, more sensitive film having larger sized grains. In magnetic tape, the larger the grains of the magnetic particles (usually ferric oxide or magnetite), the more prone the medium is to noise.

Functions Used:

1. **imnoise:** Add noise to image
   Syntax- J = imnoise(I,type).
2. **wiener2:** 2-D adaptive noise-removal filtering.
   Syntax- J = wiener2(I, [m n], noise).
3. **medfilt2:** 2-D median filtering
   Syntax- B = medfilt2(A, [m n]).

MATLAB Code:

```
clc;
close all;
clear all;

I=imread('cameraman.tif');
subplot(221)
imshow(I)
title('original image')

%TO GENERATE GAUSSIAN NOISE IN THE IMAGE....
J1=imnoise(I,'gaussian',0,0.01);      %M=0,v=0.01 default values
subplot(222);
imshow(J1);
title('noisy image(GAUSSIAN NOISE)')
K1=wiener2(J1,[5,5]);                 %To filter the gaussian noise..
subplot(223);
imshow(K1);
title('filterd image')
```

```
%TO GENERATE POISSON NOISE IN THE IMAGE....
figure
J2=imnoise(I,'poisson');
subplot(121);
imshow(J2);
title('noisy image(POISSON NOISE)')
K2=wiener2(J2,[5,5]);                    %To filter the poisson noise..
subplot(122);
imshow(K2);
title('filtered image')

%TO GENERATE SALT AND PEPPER NOISE IN THE IMAGE....
figure
J3=imnoise(I,'salt& pepper',0.02);
subplot(121);
imshow(J3);
title('noisy image(SALT AND PEPPER NOISE)')
K3=medfilt2(J3);              %To filter the salt & pepper noise
subplot(122);
imshow(K3);
title('filtered image')

%TO GENERATE SPECKLE NOISE IN THE IMAGE....
figure
J4=imnoise(I,'speckle',0.04);              %default value for v=0.04
subplot(121);
imshow(J4);
title('noisy image(SPECKLE NOISE)')
K4=medfilt2(J4);                         % To filter speckle noise  \
subplot(122);
imshow(K4);
title('filtered image')
```

OUTPUT:

original image

noisy image(GAUSSIAN NOISE)

filterd image

noisy image(POISSON NOISE)

filtered image

noisy image(SALT AND PEPPER NOISE)

filtered image

noisy image(SPECKLE NOISE)

filtered image

Conclusion: Hence we have studied filtration of image.

EXPERIMENT NO:-

Title : Piecewise linear transformation functions.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

Experiment No.-

Aim: To perform Piecewise linear transformation functions.

Software: MATLAB R2009a.

Theory:

An image lacks contrast when there are no sharp differences between black and white. Brightness refers to the overall lightness or darkness of an image.

To change the contrast or brightness of an image, the Adjust Contrast tool performs *contrast stretching*. In this process, pixel values below a specified value are displayed as black, pixel values above a specified value are displayed as white, and pixel values in between these two values are displayed as shades of gray. The result is a linear mapping of a subset of pixel values to the entire range of grays, from black to white, producing an image of higher contrast.

The principle of piecewise linear function is that of piecewise function can be arbitrarily complex.

Contrast Stretching:

It is one of simplest piecewise linear function. Low contrast images are result from poor illumination lack of dynamic range in imaging sensor or even being processed.

The function of point (r1,s1) &(r2,s2) control shape of transformation function.

1. r1=s1, r2=s2 linear transformation. No change in gray level.
2. r2=s2 & s1=0 &s2=L-1 transformation becomes thresholding function and binary image.

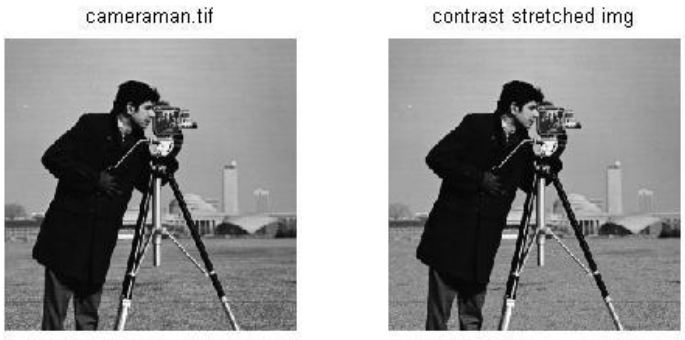
MATLAB Code:

Contrast Stretching:-

```
clc;
close all;
clear all;
img=imread('cameraman.tif');
subplot(121)
imshow(img)
title('cameraman.tif')
s=size(img);
for x=1:s(1)
   for y=1:s(2)
      if(img(x,y)>=0 &&img(x,y)<64)
img(x,y)=img(x,y);
elseif(img(x,y)>64 &&img(x,y)<=128)
img(x,y)= 1.2*img(x,y);
elseif (img(x,y)>128)
img(x,y)=img(x,y);
      end
   end
end
```

```
subplot(122)
imshow(img);
title('contrast stretched img');
```

OUTPUT:



Thresholding:-

```
clc;
close all;
clear all;
img=imread('cameraman.tif');
subplot(121)
imshow(img)
title('cameraman.tif')
s=size(img);
for x=1:s(1)
   for y=1:s(2)
      if(img(x,y)<128)
img(x,y)=0;
elseif(img(x,y)<=128)
img(x,y)=255;
      end
    end
end
subplot(122)
imshow(img);
title('thesholdedimg');
```

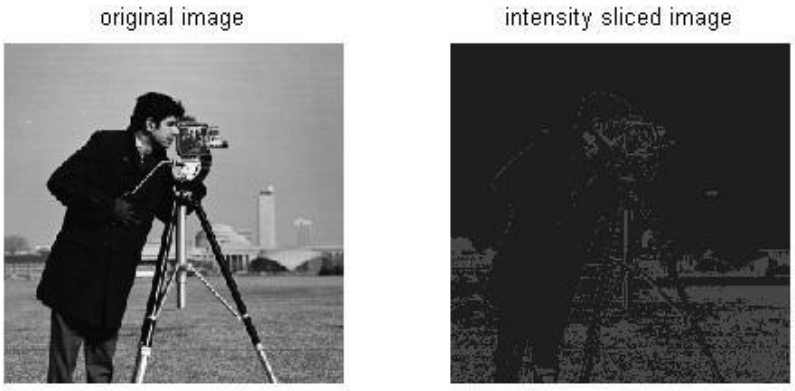
OUTPUT:-

Intensity Slicing:-

Approach-1

```
clc;
clear all;
close all;
img=imread('cameraman.tif');
s=size(img);
subplot(1,2,1);
imshow(img);
title('original image')
for x=1:s(1)
   for y=1:s(2)
      if(img(x,y)>=0 &&img(x,y)<64)
img(x,y)=30;
elseif(img(x,y)>64 &&img(x,y)<=128)
img(x,y)= 70;
elseif (img(x,y)>128)
img(x,y)=30;
      end
   end
end
subplot(1,2,2);
imshow(img);
title('intensity sliced image')
```

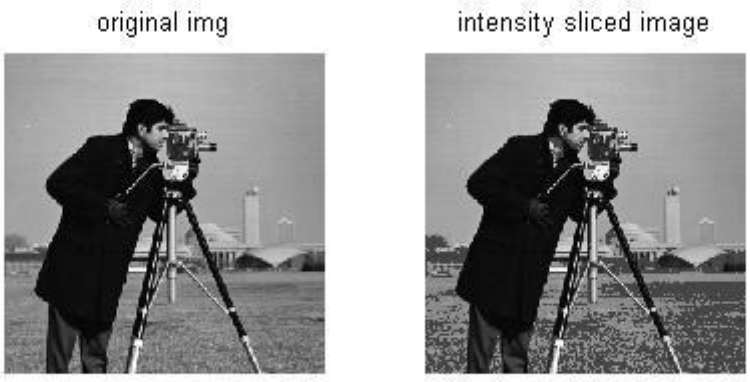
OUTPUT:



Approach-2

```
clc;
clear all;
close all
img=imread('cameraman.tif');
s=size(img);
subplot(1,2,1);
imshow(img);
title('original img')
for x=1:s(1)
   for   y=1:s(2)
      if(img(x,y)>=0 &&img(x,y)<64)
img(x,y)=img(x,y);
elseif(img(x,y)>64 &&img(x,y)<=128)
img(x,y)= 70;
elseif (img(x,y)>128)
img(x,y)=img(x,y);
       end
   end
end
subplot(1,2,2);
imshow(img);
title('intensity sliced image');
```

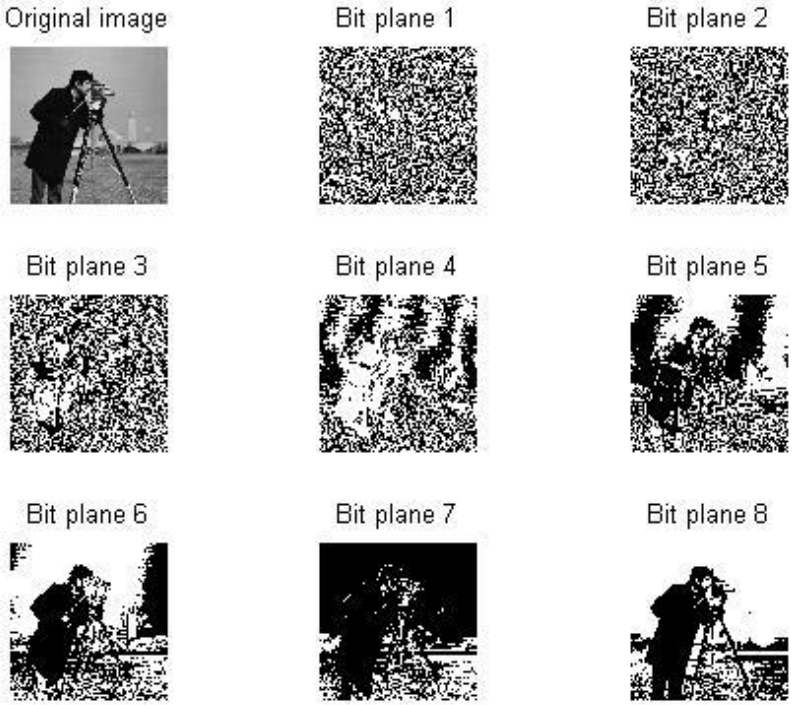
OUTPUT:



Bit-plane slicing:-

```
A=imread('cameraman.tif');
subplot(3,3,1);imshow(A);title('Original image');
B=bitget(A,1)
subplot(3,3,2); imshow(logical(B));title('Bit plane 1');

B=bitget(A,2); subplot(3,3,3); imshow(logical(B));title('Bit plane 2');

B=bitget(A,3); subplot(3,3,4); imshow(logical(B));title('Bit plane 3');

B=bitget(A,4); subplot(3,3,5); imshow(logical(B));title('Bit plane 4');

B=bitget(A,5); subplot(3,3,6); imshow(logical(B));title('Bit plane 5');

B=bitget(A,6); subplot(3,3,7); imshow(logical(B));title('Bit plane 6');

B=bitget(A,7); subplot(3,3,8); imshow(logical(B));title('Bit plane 7');

B=bitget(A,8); subplot(3,3,9); imshow(logical(B));title('Bit plane 8');
```

OUTPUT:



Conclusion: Thus, we have studied piecewise linear transformation function.

EXPERIMENT NO.:-

Title : To perform histogram equalization.

Class : Final year B.Tech (Extc)

Batch :

Roll No. :

Date :

Experiment No.-

Aim: To write an image and perform histogram equalization.

Software: MATLAB 2009b

Theory:

In image writing we just change the format of the image from the original. Histogram equalization is a method in image processing of contrast adjustment using the image histogram. This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. So in theory, if the histogram equalization function is known, then the original histogram can be recovered.

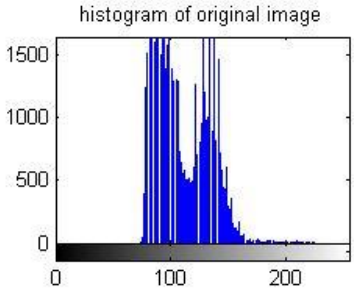
Functions used:

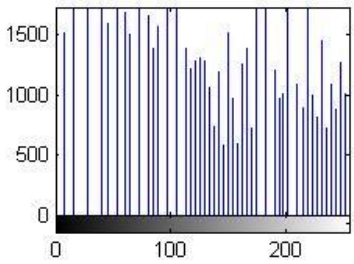
1. **histeq**: histeq enhances the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified histogram.
   Syntax-J = histeq(I, hgram)

MATLAB Code:

```
clc;
close all;
clear all;
a=imread('pout.tif');
subplot(221)
imshow(a)
title('original image')
s=size(a);
subplot(222)
imhist(a)
title('histogram of original image')
J=histeq(a);
subplot(223)
imshow(J)
title('equalized image')
subplot(224)
imhist(J)
title('histogram equalized image')
```

OUTPUT:

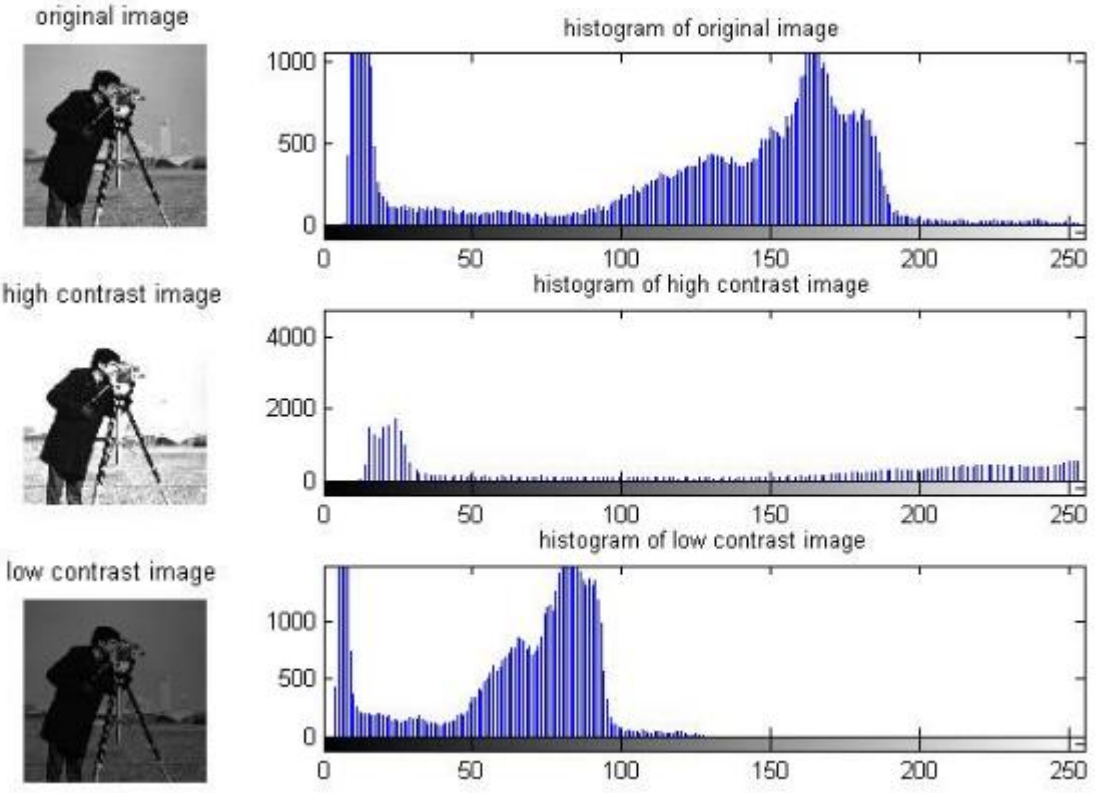


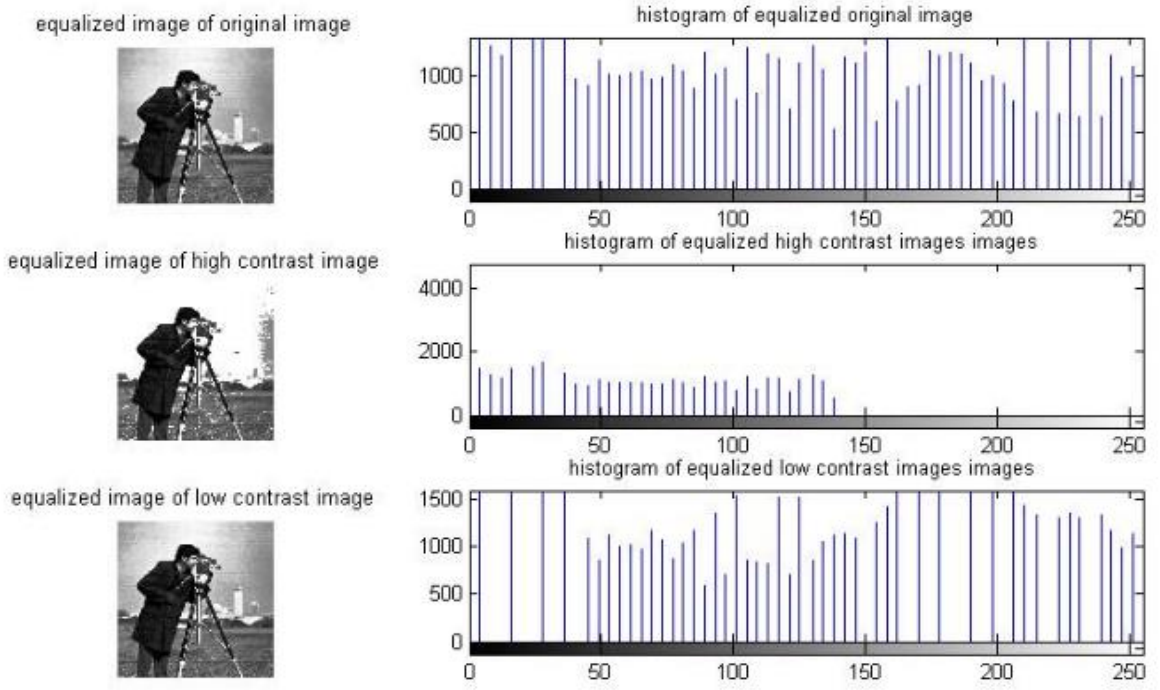



```
clc;
close all;
clear all;
a=imread('cameraman.tif');
subplot(421);
imshow(a)
title('original image')
subplot(422)
imhist(a);
title('histogram of original image')
s=size(a);
for i=1:s(1)
   for j=1:s(2)
      b(i,j)=1.7*a(i,j);
   end
end
subplot(423)
imshow(b)
title('high contrast image')
subplot(424)
imhist(b)
title('histogram of high contrast image')
for i=1:s(1)
   for j=1:s(2)
      c(i,j)=0.5*a(i,j);
   end
end
subplot(425)
imshow(c)
title('low contrast image')
```

```
subplot(426)
imhist(c)
title('histogram of low contrast image')

figure
J = histeq(a);
subplot(421)
imshow(J)
title('equalized image of original image')
subplot(422)
imhist(J)
title('histogram of equalized original image')
J1 = histeq(b);
subplot(423)
imshow(J1)
title('equalized image of high contrast image')
subplot(424)
imhist(J1)
title('histogram of equalized high contrast images images ')
J2 = histeq(c);
subplot(425)
imshow(J2)
title('equalized image of low contrast image')
subplot(426)
imhist(J2)
title('histogram of equalized low contrast images images ')
```

OUTPUT:

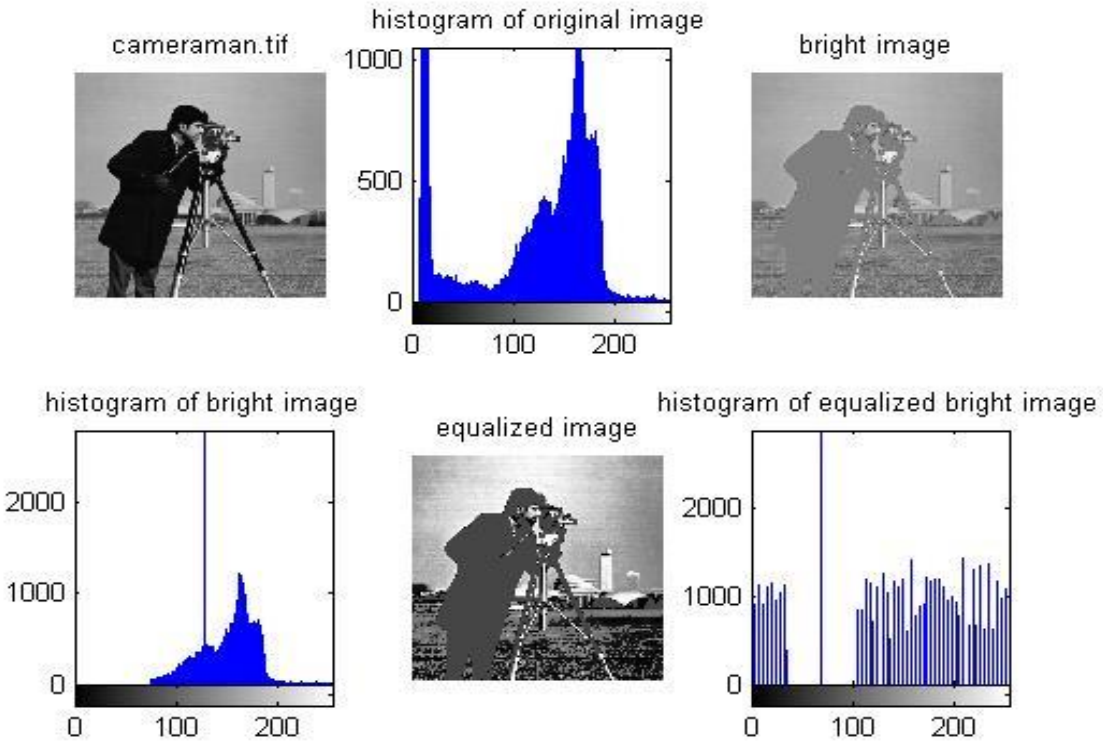





```
clc;
close all;
clear all;
img=imread('cameraman.tif');
subplot(231)
imshow(img)
title('cameraman.tif')
subplot(232)
imhist(img);
title('histogram of original image');
s=size(img);
for x=1:s(1)
   for y=1:s(2)
      if(img(x,y)<=75)
         img(x,y)=128;
      else
         img(x,y)=img(x,y);
      end
   end
end
subplot(233)
imshow(img);
title('bright image');
subplot(234)
```

```
imhist(img);
title('histogram of bright image');
subplot(235);
img1=histeq(img);
imshow(img1)
title('equalized image');
subplot(236)
imhist(img1)
title('histogram of equalized bright image');
```

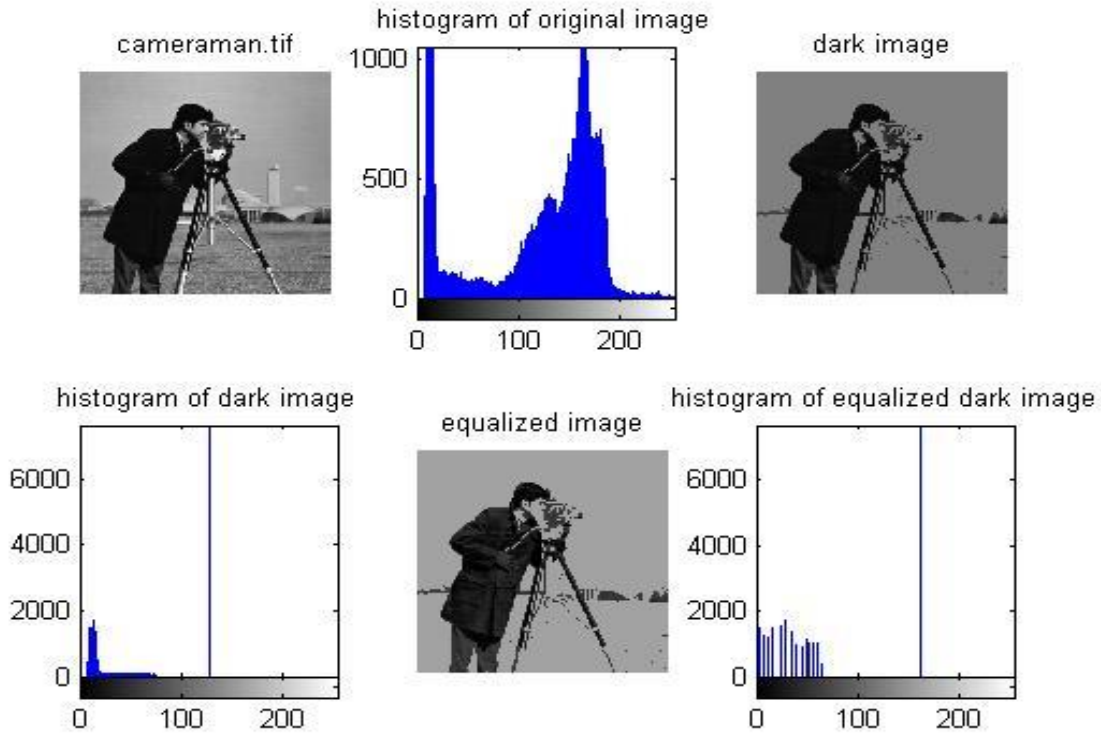
OUTPUT:




```
clc;
close all;
clear all;
img=imread('cameraman.tif');
subplot(231)
imshow(img)
title('cameraman.tif')
subplot(232)
imhist(img);
title('histogram of original image');
s=size(img);
for x=1:s(1)
   for y=1:s(2)
```

```
        if(img(x,y)<=75)
           img(x,y)=img(x,y);
        else
           img(x,y)=128;
        end
    end
end
subplot(233)
imshow(img);
title('dark image');
subplot(234)
imhist(img);
title('histogram of dark image');
subplot(235);
img1=histeq(img);
imshow(img1)
title('equalized image');
subplot(236)
imhist(img1)
title('histogram of equalized dark image');
```

OUTPUT:



Conclusion:

Thus we have studied writing of image and histogram equalization.

EXPERIMENT NO.:-

Title : Blurring and Deblurring of an image.
Class : Final year B.Tech (Extc)
Batch :
Roll No. :
Date :

Experiment No.

Aim: Blurring and Deblurring of an image.

Software: MATLAB 2009b

Theory:

The use of spatial masks for image processing usually is called spatial filtering. High frequency components characterize edges and other sharp details in an image. So the net effect of low pass filtering is image filtering. The band pass filtering removes selected frequency regions between low and high frequency. These filters are used for image restoration and for image restoration and are seldom of interest in image enhancement.

Smoothing filters are used for blurring and for noise reduction. Blurring is used in pre-processing steps such as removal of small details from an image. Noise reduction is accomplished by blurring with a linear filter and also by non-linear filtering.

The main object of sharpening is to highlight fine details that has been blurred either in error or as a natural effect of a particular method of image acquisition. Use of sharpening vary and include application ranging from electronic printing inspection and autonomous target detection in smart weapons.

Functions used:

1. **fspecial(type):** creates a 2D filter of specified type.
   Syntax-h = fspecial(type)
   h = fspecial(type, parameters)
2. **imfilter:** Each element of the output B is computed using double-precision floating point. If A is an integer or logical array, then output elements that exceed the range of the integer type are truncated, and fractional values are rounded.
   Syntax-B = imfilter(A, H)
   B = imfilter(A, H, option1, option2,...)

3. **deconvwnr:** deconvolves image I using the Wiener filter algorithm, returning deblurred image J. Image I can be an N-dimensional array. PSF is the point-spread function with which I was convolved. NSR is the noise-to-signal power ratio of the additive noise. NSRcan be a scalar or an array of the same size as
   Specifying 0 for the NSR is equivalent tocreating an ideal inverse filter.
   Syntax-J = deconvwnr(I,PSF,NSR)
   J = deconvwnr(I,PSF,NCORR,ICORR)

MATLAB Code:

```
clc;
clear all;
close all;
x=imread('cameraman.tif');
subplot(2,2,1);
imshow(x);
title('original image');
LEN=10;
THETA=10;
PSF = fspecial('motion',LEN,THETA);
Blurred = imfilter(x,PSF,'circular','conv');
subplot(2,2,2);
imshow(Blurred);
title('Blurred Image');
deblurr=deconvwnr(Blurred,PSF,0.0005);
subplot(2,2,3);
imshow(deblurr);
title('Restored, True PSF');
```

Output:

Conclusion: Thus, we have studied blurring and deblurring of image.