Lab -1

* Aim :- Getting familiar with MATLAB and performing basic operations on image.

⇒ MATLAB BASICS

1. Create the following matrix A : $A = \begin{bmatrix} 43 & 21 & 22 & 11 \\ -5 & 6 & 34 & -21 \\ 12 & 17 & -18 & 42 \end{bmatrix}$

→ A = [43 21 22 11 ; -5 6 34 -21 ; 12 17 -18 42]

(a) Create vector Va that contains the element of 2nd row of A

→ Va = A(2,:)

(b) Create vector Vb that contains the elements of 3rd column of A

→ Vb = A(:,3)

(c) Create an eight element row vector named Vc that contains the element of the 1st & 3rd row of A

→ Vc = [A(1,:), A(3,:)]

(d) Create a six element row vector named Vd that contains the elements of the 2nd and 4th column of A

→ Vd = [A(:,2), A(:,4)]

Name	Value
Vd	[21, 11 ; 6, -21 ; 17, 42]
Vc	[43, 21, 22, 11, 12, 17, -18, 42]
Vb	[22 ; 34 ; -18]
Va	[-5, 6, 34, -21]
A	3×4 double

2. Create the following three matrices:

$$A = \begin{bmatrix} 5 & 2 & 4 \\ 2 & -5 & 8 \\ 1 & -3 & -7 \end{bmatrix} \quad B = \begin{bmatrix} 10 & 7 & 3 \\ -11 & 5 & 8 \\ 4 & -3 & -7 \end{bmatrix} \quad C = \begin{bmatrix} 6 & 9 & -4 \\ 10 & 5 & 8 \\ 2 & -3 & 7 \end{bmatrix}$$

```
A = [5 2 4; 2 -5 8; 1 -3 -7];
B = [10 7 3; -11 5 8; 4 -3 -7];
C = [6 9 -4; 10 5 8; 2 -3 7];
```

(a) Calculate A+B and B+A to show that addition of matrix is commutative.

→
```
SumAB = A + B;
SumBA = B + A;
isSumABsumBASame = (SumAB == SumBA);
```

(b) Calculate A + (B+C) and (A+B) + C to show that addition of matrices is associative.

-> D = A + (B+C);

E = (A+B) + C;

(c) Calculate 3(A+C) and 3A + 3C to show that, when matrices are multiplied by a scalar, the multiplication is distributive.

-> F = 3 * (A+B);

G = 3*A + 3*B;

(d) Calculate A*(B+C) and A*B + A*C to show that matrix multiplication is distributive.

-> H = A* (B+C)

I = A*B + A*C;

Name	Value
A	[5,2,4;2,-5,8;1,-3,7]
B	[10,7,3;-11,5,8;4,-3,-7]
C	[6,9,-4;10,5,8;2,-3,7]
D	[21,18,3;1,5,24;7,-9,-7]
E	[45,27,21;-27,0,48;15,-18,-42]
G	[45,27,21;-27,0,48;15,-18,-42]
H	[102,76,27;85,-66,-82;-23,28,...
I	[102,76,27;85,-66,-82;-23,28,...
isSumABSumBASame	3x3 logical
SumAB	[15,9,7;-9,0,16;5,-6,-14]

3. Create an Array A=[1 2 3 4 5 6] and using built in functions for array find
(a) length of A
(b) Average of the element of A
(c) maximum element of A
(d) minimum element of A
(e) Sum of all the elements of A.

→ A = [1, 2 3 4 5 6];
len = length(A);
avg = mean(A);
min = min(A);
max = max(A);
sum = sum(A);

Name	value
A	[1,2,3,4,5,6]
avg	3.5000
len	6
max	6
min	1
sum	21

4. Calculate:

$$\frac{3^7 \log 76}{7^3 + 546} + \sqrt[3]{910}$$

→ A = Power (3,7) * log2 (76);
B = Power (7,3) + 546;
C = A / B;
ans = C + nthroot (910, 3);

Name	Value
A	1.3664e+04
ans	25.0608
B	889
C	15.3703

5. Using the ones and zeros commands, create a 4x6 matrix in which the first two rows are 0's and the next two rows are 1's.

→ A = [zeros (2,6) ; Ones (2,6)];
inshow A ;
~~A~~

Name	Value
A	4x6 double.

⇒ Image Processing Toolbox in MATLAB.

1. Take your own photo (RGB Image) and create the following images and save them for future use

(a) GrayScale Image

→ A = imread("Himanshu.jpg")

subplot(4,3,1), imshow(A);

B = rgb2gray(A).

subplot(4,3,2); imshow(B);

(b) Black & white image

→ H = im2bw(A)

subplot(4,3,3); imshow(H);

(c) overexposed image

→ C = A + 100

subplot(4,3,4), imshow(C);

(D) underexposed image

→ D = A - 150

subplot(4,3,5), imshow(D);

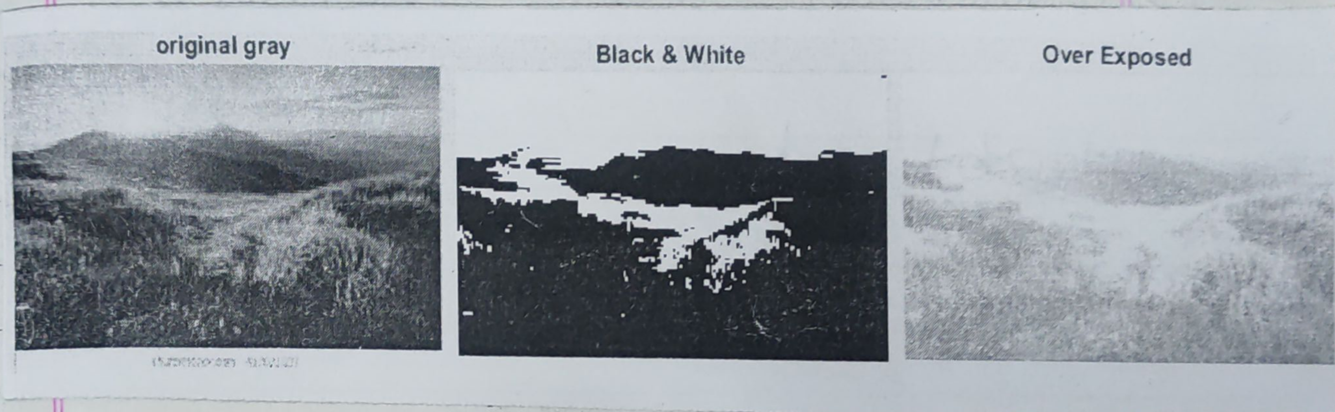
(e) Keep your face only and crop the rest of the part

→ F = imcrop(A)

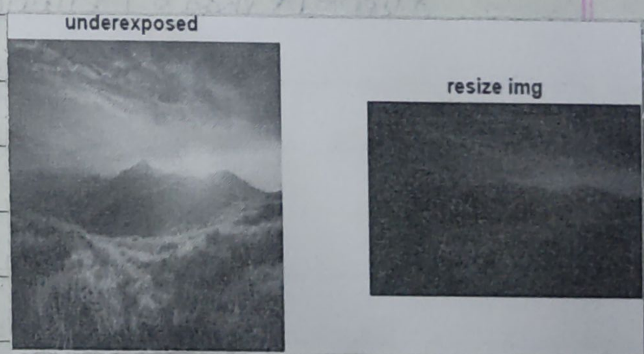
subplot(4,3,1), imshow(F);

5) resize the image to (256 × 256)

```
-> E = imresize(A, [256,256])
   subplot(4,3,6), imshow(E);
```



croped image



2 Take your own photo and process them for following results using loop controlling structures.

-> OriginalImage = imread ("Himanshu.jpg");
subplot (2,3,1); imshow (originalImage);
[rows, colums, ~] := size (originalImage);

(a) Flip your Image vertically
-> flippedImage = unit8 (zeros(rows, columns, 3));
for row = 1:rows
flippedRow = rows - row + 1;
flippedImage (flippedRow, :, :)
= originalImage (Row, :, :)
end
subplot (2,3,2);
imshow (flippedImage);

(b) Create the mirror Image.
-> mirrorImage = unit8 (zeros (colums, rows, 3));
for row = 1: rows
mirrorImage (row, :) = originalImage
(row, end: -1: 1, :);
end
subplot (2,3,3);
imshow (mirrorImage);

(c
->

(d
->

a

(c) Rotate Image by 90° degrees:

```
-> rotatedImage90 = unit8(zeros(columns, rows, 3));
   for row = 1: rows
       for colums = 1: columns
           rotatedRow = column;
           rotatedColumn = rows - row + 1;
           rotatedImage(rotatedRow, RotatedColumn, :)
               = originalImage(row, column, :);
       end
   end
   subplot(2,3,4);
   imshow(rotatedImage90);
```

(d) Rotate Image by 270 degrees

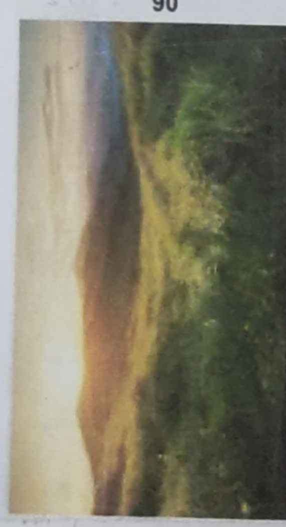
```
-> rotatedImage270 = unit8(zeros(columns, rows, 3));
   for row = 1: rows
       for column = 1: column;
           rotatedRow = column - column + 1;
           rotated column = row;
           rotatedImage(rotatedRow, rotatedColumn, :)
               = originalImage(row, column, :);
       end
   end
   Subplot(2,3,5);
   imshow(rotatedImage270);
```

original RGB

180

mirror

90



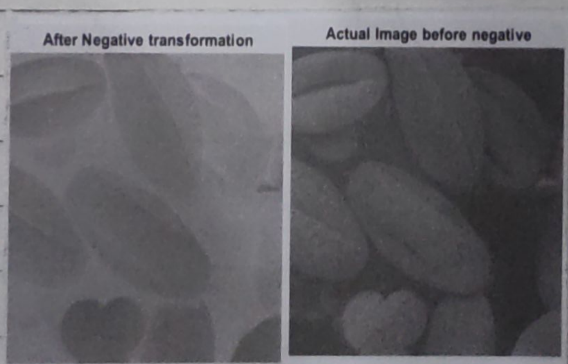
270

Lab2

Aim: Implement basic intensity transformation functions:
- Image Negatives
- Log Transformation
- Power-law (Gamma) Transformations
- Contrast Streching (Piecewise linear transformation)

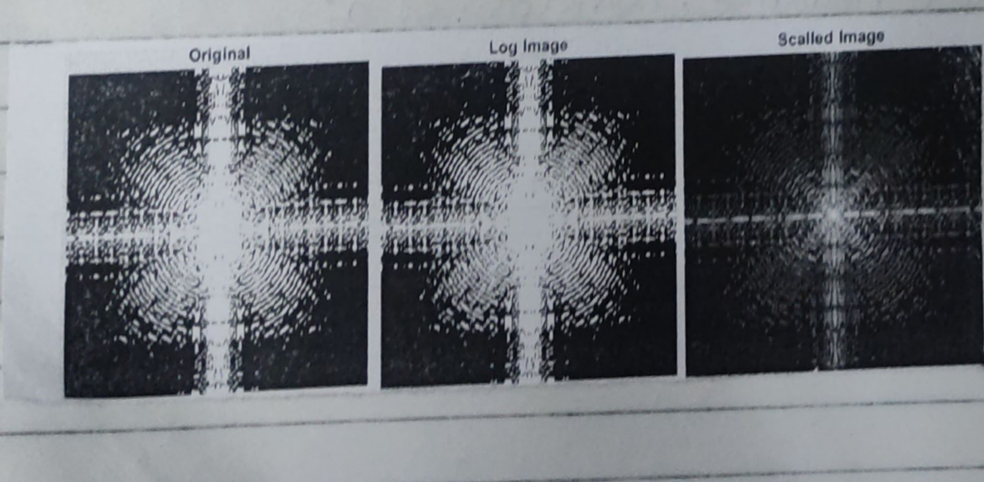
1) Take your own grayscale Photo and apply negative transformation.

```
→ img = imread('img.jpg');
gray-img = imresize(im2gray(img), [256,256]);

subplot(1,2,1);
imshow(gray-img);
title("original");
subplot(1,2,2);
neg-img = 255-gray-img;
imshow(neg-img);
title('negative transformed');
```



2) Consider image ex-log.tif. Enhance the image by applying log transformation.

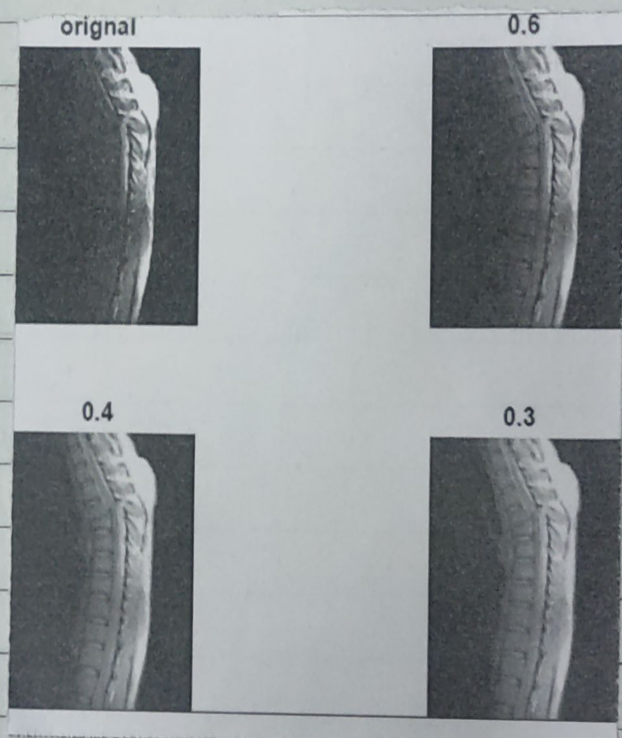
-> Img = imread("ex-log.tif');
subplot (2,2,1) ; imshow(Img);
title ('original Image');
a= im2double (Img);
subplot (2,2,2; imshow(a);
title (' ~~log transfer to~~ double');
c=1;
s= c* log (1+r); T = 255/(c*log(256));
subplot (2,2,3);
imshow(T); title('log transformed in double');
B= unit8(T*S);
subplot (2,2,4); imshow(B);
title('log transformed');



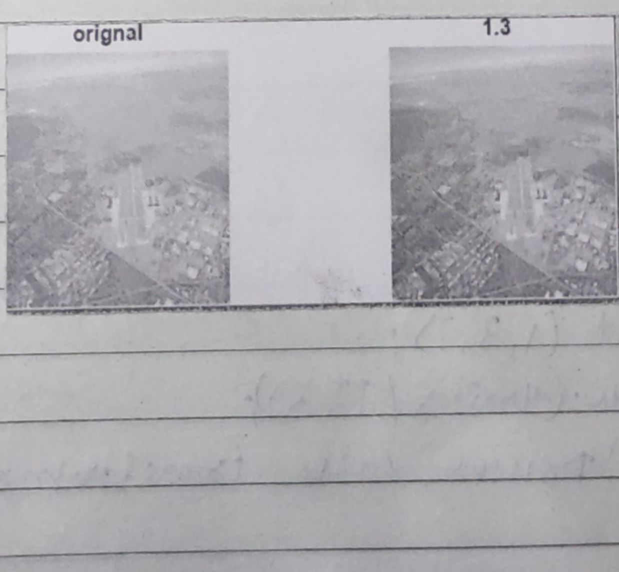
3) Consider images ex-power1.tif and ex-power2.tif and enhance them with power law transformation.

→
```
img = imread('ex-power.tif');
subplot (2,2,1);
imshow (img);
title(' original image');

c=1;
r= double (img);
gamma = 0.6;
S= c* (r^gamma);
T= 255/(c*(255^gamma));

subplot (2,2,2);
imshow(unit8(T*S));
title ('gamma=0.6');

gamma = 0.4;
S= c* (r^gamma);
T= 255/(c*(255.^gamma));
subplot (2,2,3);
imshow (unit8(T*S));
title(' Gamma=0.4');
```

```
gamma = 0.3;
S = c * (r^gamma);
T = 255 / (c * (255^gamma));
subplot (2,2,4);
imshow (unit8 (T * S));
title ('gamma = 0.3');
```

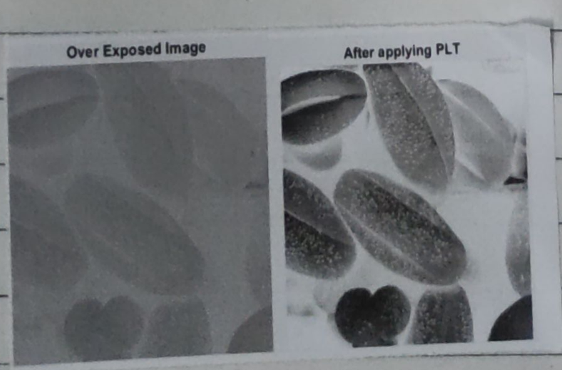


```
img =imread ('ex -power2-tif ');
c=1
r=double (img);
gamma =1.3;
s=c* (r^gamma);
T=255/ (c* (255^gamma));
subplot (2,2,1); imshow(img);
title('original image');
subplot(2,2,2);
imshow (unit8 (T* s));
title('gamma =1.3);
```



4. Consider your over exposed photo (that you generated for assignment 1) and enhance it by power law transformation. Specify the value of gamma which is suitable for this enhancement.

→
```
img = imread('img.png');
subplot(1,3,1);
imshow(img);
title('original image');
subplot(1,3,2);
imshow(img + 50);
title('over exposed image');

c=1;
r=double(img+50);
gamma=1.4;
s=c*(r^gamma);
T = 255/(c*(255^gamma));
subplot(1,3,3);
imshow(unit8(T*s));
title('Power law transformed');
```



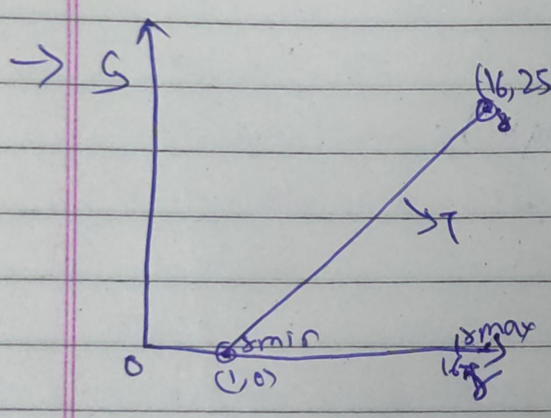
5. Consider your under exposed photo(that you generated for assignment1). and enhance it by power law transformation. Specify the value of gamma which is suitable for this enhancement.

```
-> img = imread('img.jpg');
   subplot(1,3,1); imshow(img);
   title('original image');
   subplot(1,3,2);
   imshow(img-50);
   title('under exposed image');

   c = 1;
   r = double(img-50);
   gamma = 0.6;
   S = c*(r^gamma);
   T = 255/(c*(255^gamma));
   subplot(1,3,3);
   imshow(unit8(T*S));
   title('Power law transformed');
```

6. Contrast Streching: A 3x3 8bits/pixel image given

7	12	8
16	9	6
10	15	1

Apply contrast strech to the image so that the new img has a dynamic range of [0,255]. Also show the output image. Sketch the transformation you used for contrast Streching.



$s = mr + c$

$m = \frac{255-0}{16-1} = \frac{255}{15} = 17$

$s = m(r - r_{min}) + s_{min}$

$= 17(r-1) + 0 = 17(r-1)$

r	s
7	$17(7-1) = 102$
12	$17(12-1) = 187$
8	$17(8-1) = 119$
16	$17(16-1) = 255$
9	$17(9-1) = 136$
6	$17(6-1) = 85$
10	$17(10-1) = 153$
15	$17(15-1) = 238$
1	$17(1-1) = 0$

New image:-

102	187	119
255	136	85
153	238	0

7. Do contrast streching for the image ex-contrast.tif. Obtain contrast streched image from low contrast image and apply threesholding.

→
```
img=imread('ex-contrast.tif');
subplot(1,3,1);
imshow(img);
title('original image');
rmin = min(min(img));
rmax = max(max(img));
smin = 0;
smax = 255;

s=((smax-smin)/(rmax-rmin))*(img-rmin) + smin;
subplot(1,3,2);
imshow(s);
title("enhanced image");

rmin=125;
rmax=125;
s=((smax-smin)/(rmax-rmin))*(img-rmin) + smin;
subplot(1,3,3);
imshow(s);
title('Threshold Image');
```


Original
strached image

Lab -3

Aim:- Perform the following tasks
- Calculate the brightness and contrast of image
- Perform AND, OR logical operation on image
- Perform image Rotation.
- Bit plane slicing & Reconstruction.

1. Calculate Brightness & contrast of an image

(i)
```
img := imread("home.jpg")
gray = rgb2gray (img);
gray = double (gray);
[h,w] = size (gray);
S = 0.0
for i = 1:h
    for j = 1:w
        S = S + gray(i,j)
    end
end.
disp(S)
S = S / (h*w)
b = mean2(gray)
disp(" Brightness" + b);
```

(ii)
```
cont = 0.0
for i = 1:height
    for j = 1: width
```

```
        cont = cont + (gray(i,j)-b)^2;
    end
end
cont = cont / (h*w);
cont = nthroot(cont, 2);
disp("contrast" + cont);
```

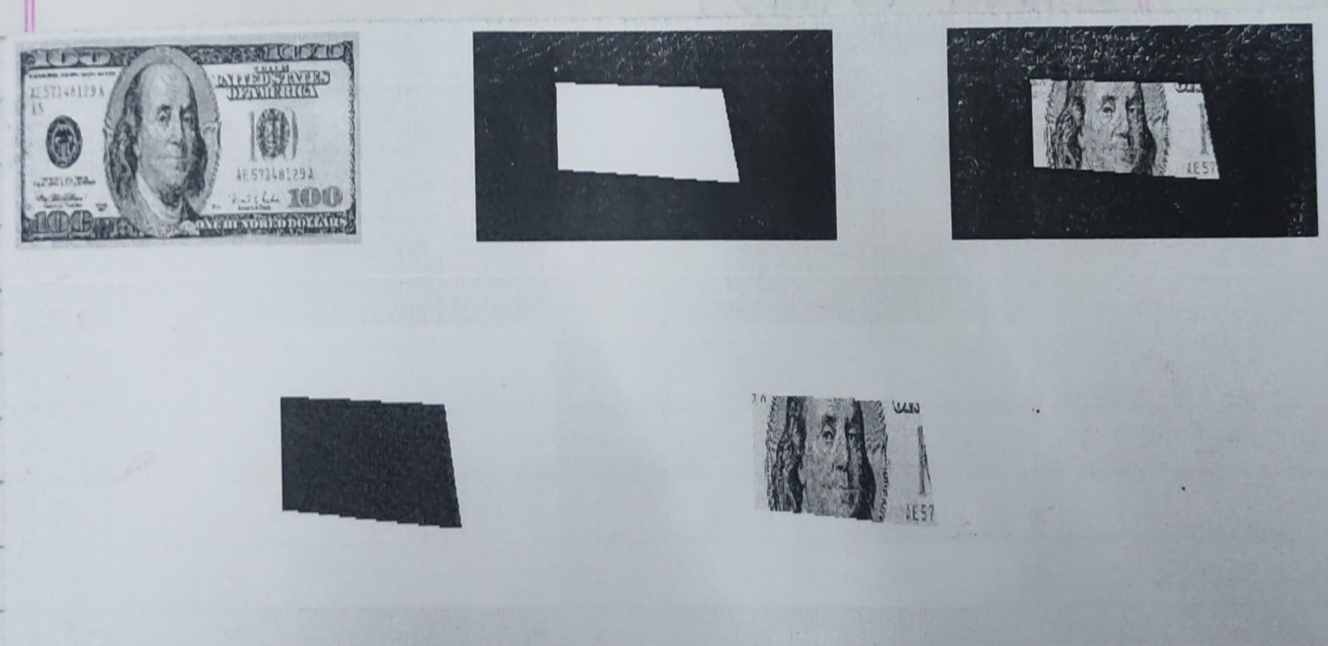
⇒ brightness :- 111.504
contrast :- 64.5811

2. ~~Part~~ Perform logical operation.

(i) AND

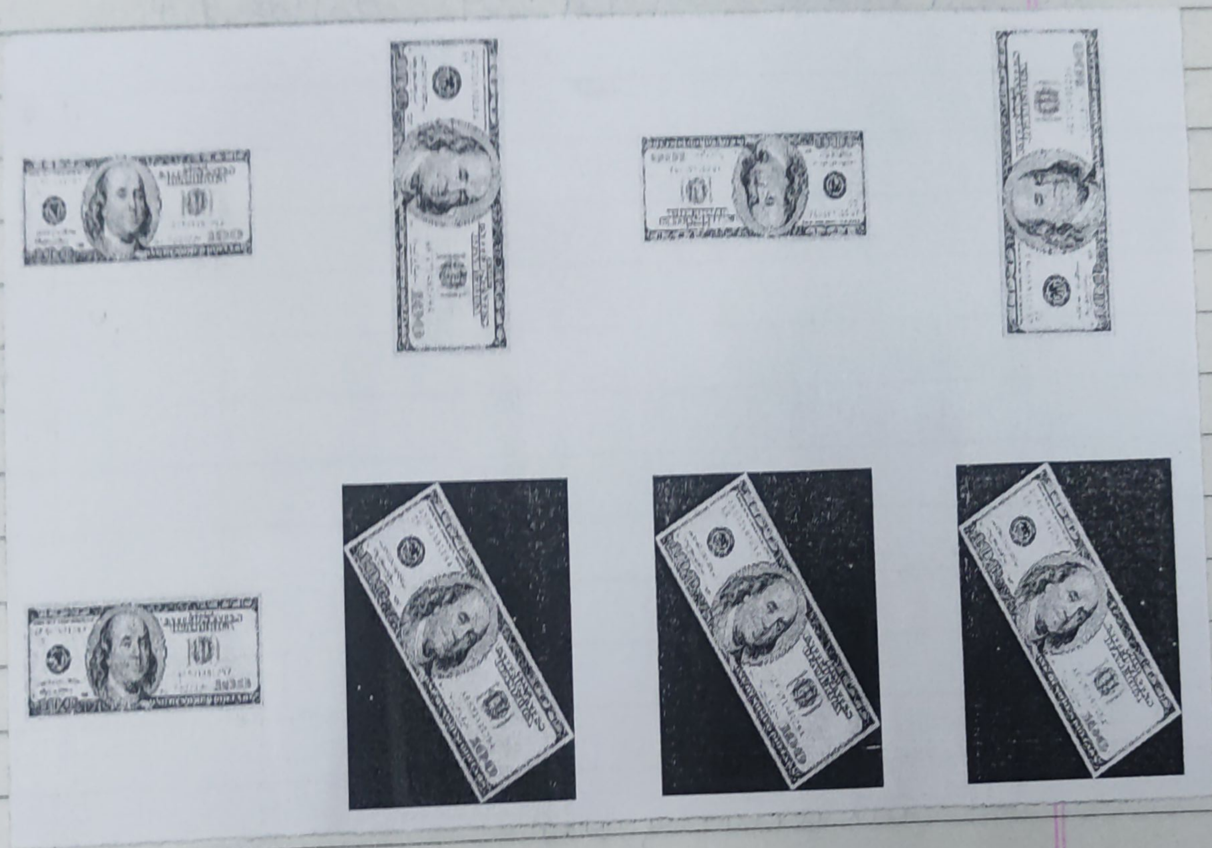
```
img = imread("dollar.tif");
img = im2gray(img);
subplot(2,2,1)
imshow(img)
title("original")
mask1 = unit8(roipoly(img)*255);

final1 = bitand(img, mask1)
subplot(2,2,2)
imshow(final1)
title("After and")
```

(ii)
```
mask2= 255-mask1;
final2= bitor (img, mask2)
subplat (2,2,3)
imshow (final2)
title("After or")
```



3. Image rotation

```
img := imread("hello.jpg")
subplot(2,2,1)
imshow(img)
title("original")
rot90 := rot90(img)
subplot (2,2,2)
imshow (rot 90)
title ("Rotated 90 degree);
rot 90-t2 = imrotate (img, 90)
subplot (2,2,3)
imshow(rot90-t2)
title ("Rotated 90 using imrotate");
rot45 = imrotate (img, 45)
subplot (2,2,4)
imshow (rot45)
title ("Rotated 45 using imrotate");
```



4. Bit plane slicing on dollar.tif.

i) Seprate out it's bit planes

```
Img = imread("dollar.tif")
img = im2gray (img)
subplot (2,2,1)
imshow (img)
title (original)

b1 = bitget (img, 1);
b2 = bitget (img, 2);
b3 = bitget (img, 3);
b4 = bitget (img, 4);
b5 = bitget (img, 5);
b6 = bitget (img, 6);
b7 = bitget (img, 7);
b8 = bitget (img, 8);
```

(ii) Reconstruct image using heigher order 2 bits (8,7)

```
h2_bit = unit8 (64 * b7 + 128 * b8);
subplot (2,2,2)
imshow (h2_bit)
title ("Image using heigher 2 bits").
```

(iii) Re construct Image using heigher order 4 bits (8,7,6,5).

```
h4_bit = unit(128*b8 + 64*b7 + 32*b6 + 16*b5)
subplot(2,2,3)
imshow(h4_bit)
title("Image using heigher order 4 bits")
```

(iv) (importante) Experiment with bit plane and derive your conclusion.

Importance of bit plance slicing is that it can be used to determine relative importance can be used in image compression also.

Lab-4

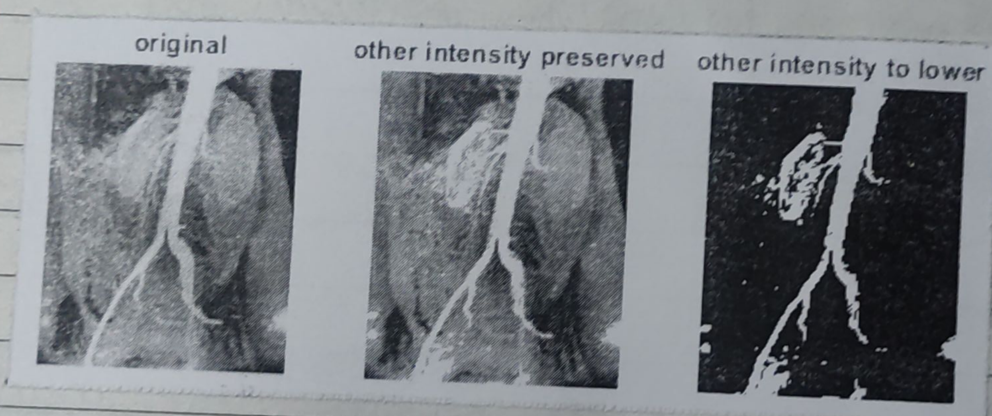
Aim: Introduction to image enhancement techniques:
- Intensity level slicing
- Histogram equalization
- Smoothing spatial filtering

1. Consider img kidney.tif and perform intensity level transformation.

a. Highlight given intensity range and keep all other intensities to as it is.

```
img = imread("kidney.tif");
[m,n] = size(img);
left = 150;
right = 230;
img1 = img;
img2 = img;

for i = 1:m
    for j = 1:n
        if img(i,j) >= left &&
            img(i,j) <= right

                img1(i,j) = 255;
                img2(i,j) = 255;
        else
            img2(i,j) = 0;
        end
    end
end
```

```
subplot (1,3,1);
imshow (img)
title ("original");
subplot (1,3,3);
imshow (img1);
title ("other intensity preserver");
subplot (1,3,3),
imshow (img2),
title ("other intensity to lower");
```



2. Can two visually different image, have same histogram.

-> yes, two visually different image, have same histogram.

histogram takes overall probability of intensity into account.

Hence, no matter location of pixel.
For this reason if two images can have same probability distribution but different location of pixels.
Example:

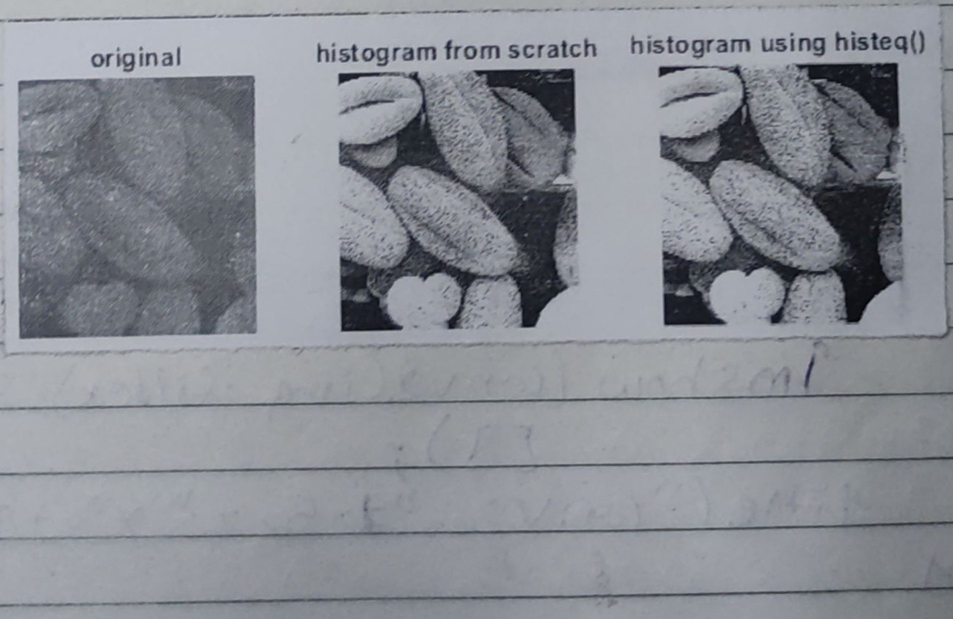
```
img1 = [zeros (2,4);
        ones (2,4)];
img2 = [ones (2,4);
        zeros(2,4)];
disp (isequal(imhist (img1), imhist (img2)));
```

Output = 1.

3. histogram equalization on excontrast.tif.
- create a function for histogram equalization.
- using histeq(img).

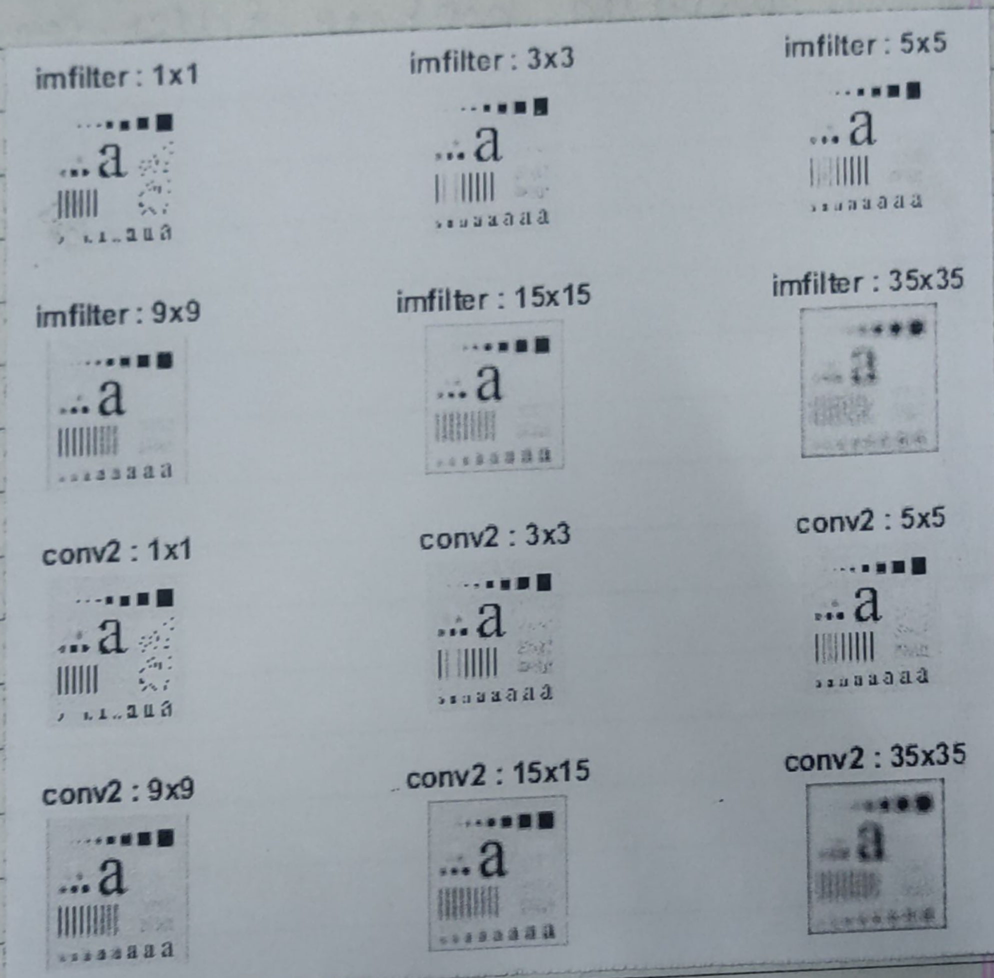
```
-> img = imread("ex_contrast.tif");
hist = imhist (img);
[m,n] = size (img);

probability = double (double(hist),
                      double (m * N));
cdf = probability;

for i = 2 : length(probability),
      cdF(i) = double ( cdF(i) + cdF (i-1));
end.
mapping = round (double (255.0 * cdF));
img_eq = img;

for i = 1 : m
      for j = 1 : n.
            Imgeq(i;j) = mapping (img(i,j) + 1);
      end.
end.
```

```
subplot(1,3,1);
imshow(img);
title("original");
subplot(1,3,2);
imshow(imgeq);
title("histogram from scratch");

imhist-eq = histeq(img);
subplot(1,3,3);
imshow(imghist-eq);
title("histogram using hist-eq");
```



4. Blur gray scale photo with box filter of 3×3, 5×5, 9×9, 15×15, 35×35

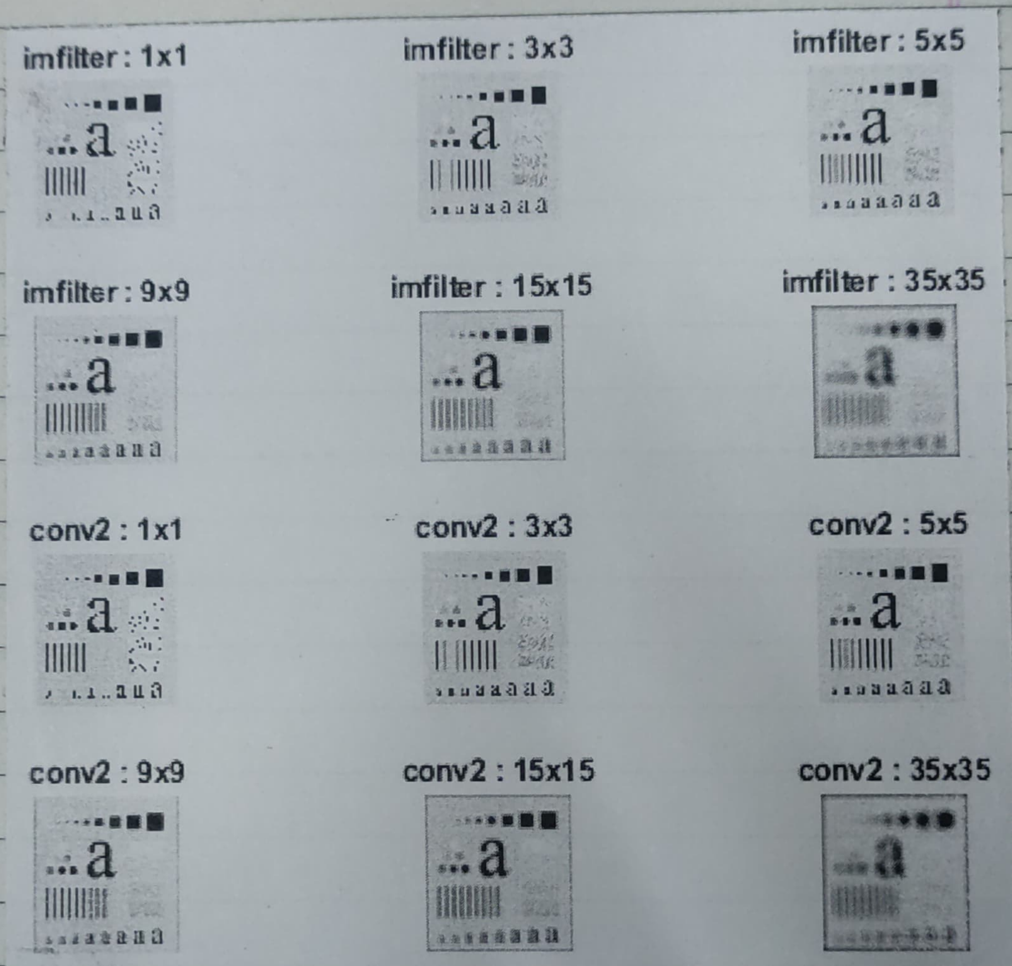
```
img = imread("smooth.tif");
filtersize = [1 3 5 9 15 35];

for i=1:6
    subplot(4,3,i)
    sz = filtersize(i);
    filter = ones(sz)/(sz*sz);
    imshow(imfilter(img, filter));
    title("imfilter:"+sz+"x"+sz);
end

for i=1:6
    subplot(4,3,i+6);
    sz=filtersize(i);
    filter =ones(sz)/(sz* sz);
    imshow(conv2(img, filter),'same',
            []);
    title("conv2:"+sz+"x"+sz);
end
```


imfilter : 1x1
imfilter : 3x3
imfilter : 5x5
imfilter : 9x9
imfilter : 15x15
imfilter : 35x35
conv2 : 1x1
conv2 : 3x3
conv2 : 5x5
conv2 : 9x9
conv2 : 15x15
conv2 : 35x35

5. Take any gray scale photo and blur it with weighted average filter. Compare the amount of blurring with standard box filter on same size.

→

```
stdfilter=ones(5)/25;
weighted filter=[ 1 2 1 2 1;
                  2 1 2 1 2;
                  1 2 1 2 1;
                  2 1 2 1 2;
                  1 2 1 2 1]/37.0;
Img=imread("smooth.tif");
subplot(1,3,1);
imshow(img);
title("original");
subplot(1,3,2);
imshow(Img, std filter);
title("std. Filter");

subplot(1,3,3);
imshow(imfilter(img, weightedFilter));
title("weighted filter");
```


imfilter : 1x1
imfilter : 3x3
imfilter : 5x5
imfilter : 9x9
imfilter : 15x15
imfilter : 35x35
conv2 : 1x1
conv2 : 3x3
conv2 : 5x5
conv2 : 9x9
conv2 : 15x15
conv2 : 35x35