DIP

Prac 1:

a) 2D linear convolution

```
clc;
x = [2,5,6; 4,5,6; 5,2,4];
h = [1; 1; 1];
y = conv2(x,h);
disp(y,'Linear conv')
```

```
2.   5.   6.
6.   10.  12.
11.  12.  16.
9.   7.   10.
5.   2.   4.
```

b) circular convolution between two 2D matrices

```
clc;
x=[5,6; 2,8];
h=[4,6; 7,8];
X = fft2(x);
H = fft2(h);
Y = X.*H;
y = fft2(Y)
disp(y,'circular linear convolutn').
```

```
536.   504
556.   504.
```

Scanned By Scanner Go

Prac 2:
circular convalution expressed as linear convolution plus alias.

```
clc;
x = [3,4; 4,5];
h = [5,6; 8,9];
y = conv2(x,h);
y1 = [y(:,1) + y(:,3); y(:,2)];
y2 = [y1(1,:) + y1(3,:); y1(2,:)]
disp(y,'linear convolution')
disp(y2,'Circular convolutn exp.....')
```

15. 38. 24.
44. 108. 66.
32. 76. 45.

116. 110.

Prac 3:
a) Linear Cross-correlation of 2D Matrix.

```
clc;
x = [5,1; 2,4];
h1 = [1,4; 2,4];
h2 = h1(:,3:-1:1);
h = h2(3:-1:1,:);
y = conv2(x,h)
disp(y,'cross-correlatn')
```

20. 14. 2.
20. 29. 9.
5. 18. 4.

Scanned By Scanner Go

b) Lin. circular correlation between two signals.

```
clc;
x = [4,6 ; 2,4];
h = [5,4 ; 4,1];
h = h(:, $ : -1 : 1);
h = h($ : -1 : 1, 7:);
X = fft2(x);
H = fft2(h);
Y = X .* H;
y = fft2(Y);
disp(Y, 'cic. . . . . ')
```

```
224.   192.
256.   224.
```

c) linear auto cross correlation.

```
clc;
x1 = [1,1 ; 1,1];
x2 = x1(:, $ : -1 : 1);
x2 = x2($ : -1 : 1, :);
x = conv2d(x1, x2)
disp(x, 'Linear auto')
```

```
1.  2.  1.
2.  4.  2.
1.  2.  1.
```

Scanned By Scanner Go

Prac 4:
DFT of 4x4 gray scale image.

```
clc;
clear;
m = [2,1,2,1; 2,1,2,1; 1,2,1,1;
     2,1,2,1];
w = fft(m)
disp(w, '2D dft');
```

23. +0.i 0. -i 5. +0.i 0. +i

Prac 5:
a) discrete cosine transform.

```
clc;
N = 4
f = zeros(1,N);
for k = 1:N
    for n = 1:N
        if (k-1) == 0
            c(k,n) = inv(sqrt(N));
        else
            c(k,n) = sqrt(2) * inv(sqrt(N)) * cos
                     (%pi * (2*(n-1)+1) * (k-1)/(2*N));
        end
        disp(c(k,n));
    end
end
```

Scanned By Scanner Go

Prac 6:

a) brightness enhancement of an image.

```
clc;
clear;
a = imread("lena.jpg")
a = rgb2gray(a)
b = double(a) + 20;   // -20 (for brightness suspension).
b = uint8(b)
figure(1)
imshow(uint8(a))
title("original image")
figure(2)
imshow(uint8(b))
~~ti~~title("enhanced image");
```

c) ~~cords~~ contrast manipulation:

```
clc;
clear;
a = imread("baby.jpg");
b = a*0.4;
c = a*2;
subplot(1,3,1), imshow(a), title('original image');
subplot(1,3,2), imshow(b), title('Decreasin contrast
        (1,3,3), imshow(c), title('Increase in contra');
```

Scanned By Scanner Go

d) image negative.

```
clc;
close;
a = imread('label.jpg');
k = 255 - double(a);
k = uint8(k);
imshow(a);
title("original image');
imshow(k);
title('Negative image');
```

Prac 7: threshold operation.

same as enhancement.

Scanned By Scanner Go

b) gray level slicing without background

```
clc;
x = imread("img.jpg");
x = rgb2gray(x);
y = double(x);
[m,n] = size(y);
L = max(max(x));
a = round(L/2);
b = L;
for i = 1:m
for j = 1:m
if(y(i,j) >= a & y(i,j) <= b)
z(i,j) = L;
else
z(i,j) = 0;
end
end
end
z = uint8(z);
figure(1)
imshow(uint8(x));
title('og image');
fig(2)
imshow(uint8(z));
title('Gray level ....');
```

Scanned By Scanner Go

Prac 8 : Image Segmentation:

a) gaussian.

```
clc;
close;
sigma = input ('Enter value of sigma:')
i = -10:.1:10;
j = -10:.1:10;
r = sqrt (i.*i+j.*j);
Y = (1/(sigma^2))*(((r.*r)/sigma^2)-1)
        *exp(-r.*r/2*sigma^2);
plot (i, Y)
legend (sprintf('The sigma value is %.g', sigma))
title('guasian')
```

out: Enter the value of sigma : 5.

b) Shape of dog:

```
clc;
clear;
Sigma 1 = input ('value of Sigma 1:')
sigma 2 = input ('value of Sigma 2:')
i = -10:.1:10;
j = -10:.1:10;
r = sqrt (i.*j+j.*j);
Y1 = (1/(sigma 1^2))*(((r*r)/sigma1^2)-1)*
      exp(-r*r/2*sigma1^2);
Y2 = (1      "       "   );
Y = Y1 - Y2
plot(i, Y)
title ('shape of dog')
```

out: sigma 1 : 10 sigma 2 : 5

Scanned By Scanner Go

Prac 9: Image compression.

```
clc;
clear;
close;
x=[69, 75, 80, 70; 72, 75, 82, 68;
   84, 72, 62, 65; 66, 68, 72, 80];
disp(x,'original Block is x = ')
[m1 n1] = size(x);
blk = input ('Enter the block size : ');
for i = 1 : blk : m1
for j = 1 : blk : n1
    y = x(i:i+(blk-1), j:j+(blk-1));
    m = mean (mean(y));
    disp(m,'mean value is m = ');
    sig = std2(y);
    disp(sig,'standard deviation of block is =')
    b = y > m;
    disp (b,'Binary allocation matrix is B = ')
    k = sum (sum(b));
    disp(k,'num of ones = ')
      if (k = blk^2) & (k = 0)
        mL = m - sig * sqrt (k/((blk^2) - k));
        disp(m1,'The value of a = ')
        mU = m + sig * sqrt (((blk^2) - k)/k);
        disp(mU,'The value of b = ')
        x(i:i+(blk-1), j:j+(blk-1)) =
            b*mU + (1-b)*mL;
    end
end
end
disp (round(x),'Reconstructed Block is x = ')
```

Scanned By Scanner Go

Prac 10 :

a) Binary Image procesing.

```
clc;
clear;
close;
a = imread ('img.jpg');
StructureElement = Create Structure Elemen
                   ('square', 3);
a1 = DilateImage (a, StructureElement);
b1 = ErodeImage (a, s ' '  );
figure (1)
ShowImage (a, 'org image');
figure(2)
ShowImage (a1, 'Dilated Image');
figure(3) title(
ShowImage (b1, 'Eroded Image');
title( ).
```

Scanned By Scanner Go