INDEX

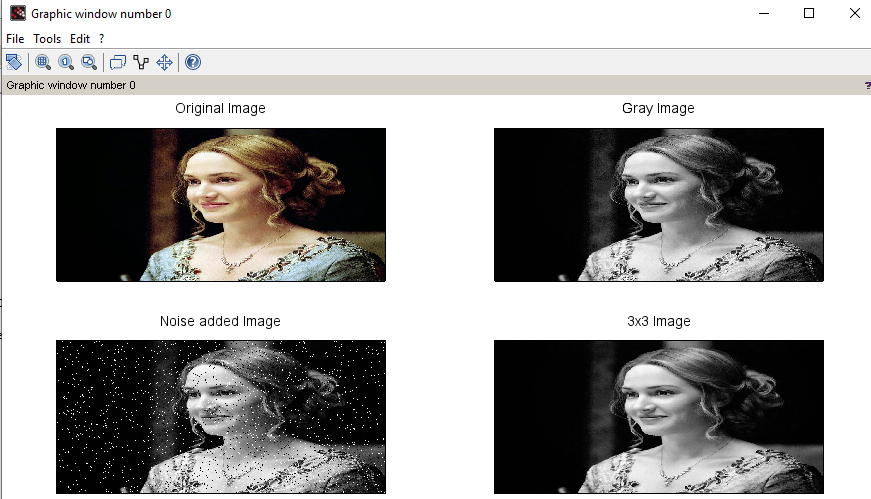
|  |  |  |  |
| --- | --- | --- | --- |
| MODULE NO. | PG. NO. | TOPIC | REMARK |
| 1 | 04 | Median Filter in MATLAB to remove Salt & Pepper noise. |  |
|  |  |  |
| 2 | 06 | MATLAB program for Deblur Images Using a Wiener  Filter |
|  |  |  |
| 3 | 07 | MATLAB program for Image Negation. |
|  |  |  |
| 4 | 08 | Edge Detection using Sobel, Prewitt and Roberts Operators. |
|  |  |  |
| 5 | 09 | MATLAB program for morphological operations on binary images. |
|  |  |  |
| 6 | 11 | Image Smoothening and Sharpening |
|  |  |  |
| 7 | 12 | MATLAB program for Scaling & Rotation Scaling (Resize). |
| 8 | 13 | MATLAB program for edge detection, gray level Thresholding in Image Segmentation. |
| 9 | 15 | Write a program on Discrete Cosine Transform. |

# PRACTICAL NO: -1

AIM: Median Filter in MATLAB to remove Salt & Pepper noise.

|  |
| --- |
| clc;  clear all  I = imread('C:\Users\admin\Pictures\desktop.jpg');  K = rgb2gray(I);  J = imnoise(K, 'salt & pepper', 0.05);  [m, n] = size(J);  *//d = zeros(m, n); % Initialize the output image*  for i = 2: m-1  for j = 2: n-1  d(i, j) = median([J(i-1, j+1), J(i, j+1), J(i+1, j+1); J(i-1, j), J(i, j), J(i+1, j); J(i-1, j), J(i, j-1), J(i+1, j-1)]);  end  end  subplot(3, 2, 1);  imshow(I);  title('Original Image');  subplot(3, 2, 2);  imshow(K);  title('Gray Image');  subplot(3, 2, 3);  imshow(J);  title('Noise added Image');  subplot(3, 2, 4);  imshow(d);  title('3x3 Image'); |

OUTPUT:



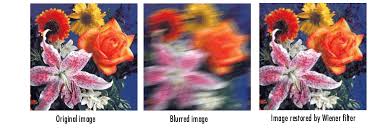
PRACTICAL NO: -2

AIM: MATLAB program for Deblur Images Using a Wiener Filter.

Code:

|  |
| --- |
| clc;  clear all;  Ioriginal = imread('C:\Users\admin\Pictures\MT 15 2 Final.jpg');  subplot(1,3,1);  imshow(Ioriginal);  title('Original Image');  PSF = fspecial('motion',21,11);  Idouble = im2double(Ioriginal);  blurred = imfilter(Idouble, PSF, 'conv', 'circular');  subplot(1,3,2);  imshow(blurred);  title('Blurred Image');  wnr1 = deconvwnr(blurred, PSF);  subplot(1,3,3);  imshow(wnr1);  title('Restored Blurred Image'); |

OUTPUT: -



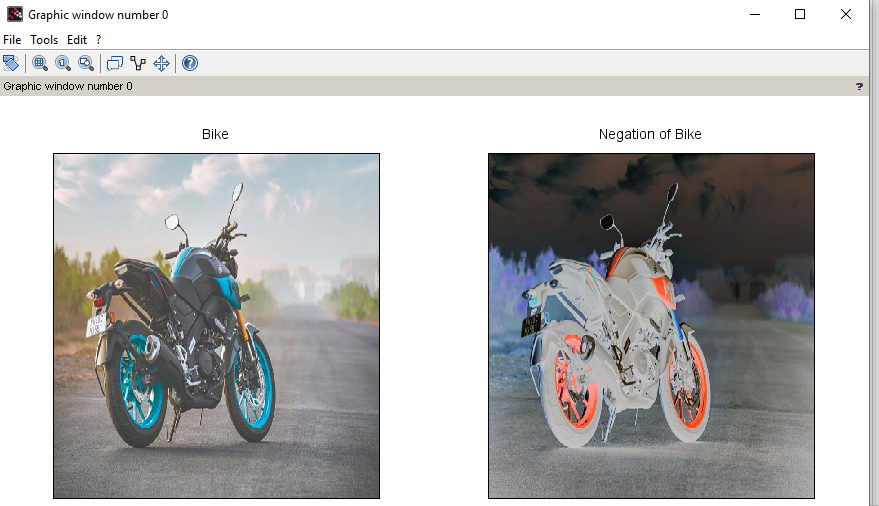
PRACTICAL NO: -3

AIM: MATLAB program for Image Negation.

Code:

|  |
| --- |
| clc; clear all;  a = imread('C:\Users\admin\Pictures\MT 15 2 Final.jpg')  subplot(1,2,1);  imshow(a);  title('Bike');  b = 255-a;  subplot(1,2,2);  imshow(b);  title('Negation of Bike'); |

Output:-

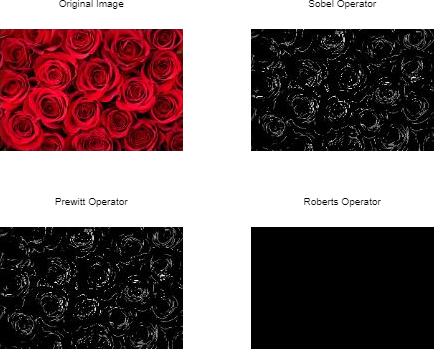
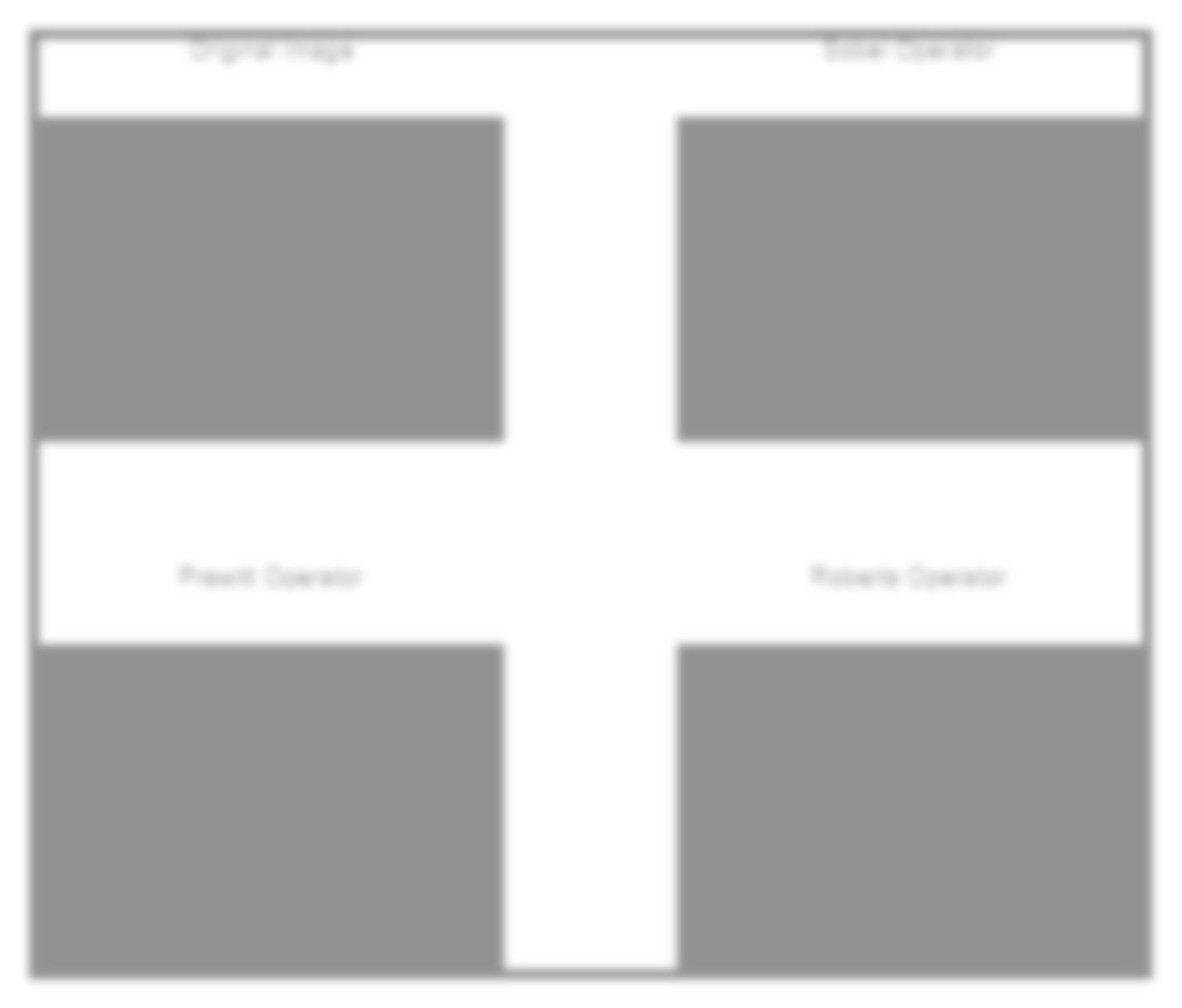


PRACTICAL NO: -4

AIM: Edge Detection using Sobel, Prewitt and Roberts Operators. Code : edge.m

|  |
| --- |
| clc;  clear all;  close all;  a = imread('rose.jpg');  b = rgb2gray(a);  subplot(2,2,1);  imshow(a);  title('Original Image');  c1 = edge(b,'sobel');  subplot(2,2,2);  imshow(c1);  title('Sobel Operator');  c2 = edge(b,'prewitt');  subplot(2,2,3); imshow(c2);  title('Prewitt Operator');  c3 = edge(b,'roberts');  subplot(2,2,4);  imshow(c3);  title('Roberts Operator'); |

Outpu



PRACTICAL NO: -5

AIM: MATLAB program for morphological operations on binary images.

Code: # Importing the image

|  |
| --- |
| clc;  clear all;  I = imread("C:\Users\admin\Pictures\Rose.jpeg");  subplot(2, 3, 1),  imshow(I);  title("Original image");  *//% Dilated Image*  se = strel("line",7,7);  dilate = imdilate(I, se);  subplot(2,3,2),  imshow(dilate);  title("Dilated image");  *// Eroded image*  erode = imerode(I, se);  subplot(2, 3, 3),  imshow(erode);  title("Eroded image");  *//Opened image*  open = imopen(I, se);  subplot(2, 3, 4),  imshow(open);  title("Opened image");  *// Closed image*  close = imclose(I, se);  subplot(2, 3, 5),  imshow(close);  title("Closed image"); |

OUTPUT: -



# 

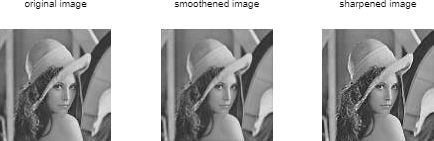
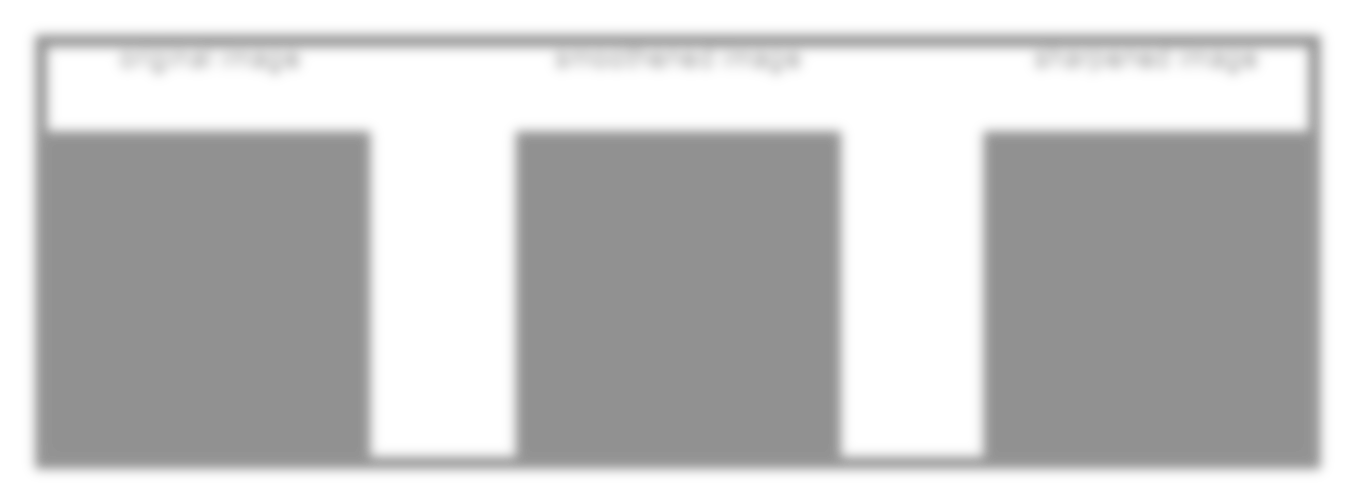
PRACTICAL NO: -6

# Aim:- Image Smoothening and Sharpening

Code: Smoothening.m clc;

|  |
| --- |
| clc;  clear all;  a=imread('C:\Users\admin\Pictures\desktop.jpg');  subplot(1,3,1);  imshow(a);  title('original image');  h = fspecial('gaussian');  b = imfilter(a,h);  subplot(1,3,2);  imshow(b);  title('smoothened image');  c = imsharpen(a);  subplot(1,3,3);  imshow(c);  title('sharpened image'); |

# Output:-



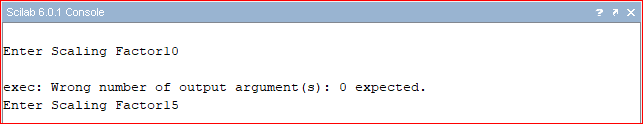
# PRACTICAL NO: -7

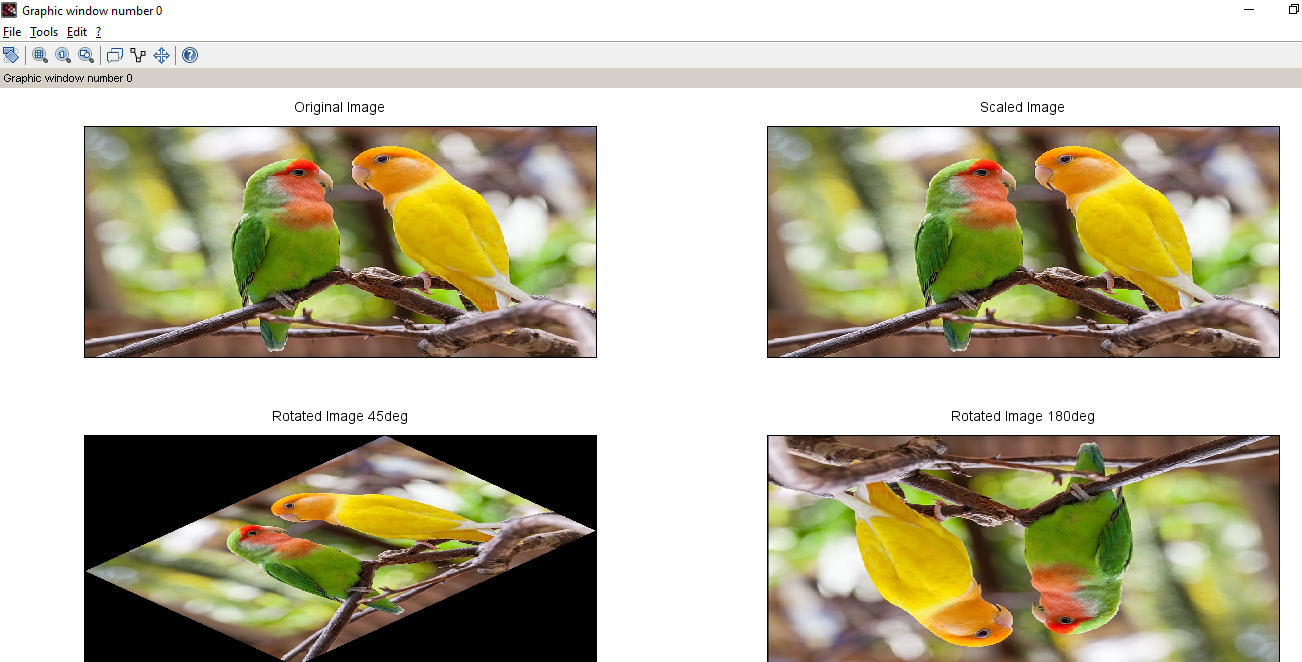
AIM: MATLAB program for Scaling & Rotation Scaling (Resize).

Code:

|  |
| --- |
| clc;  clear all;  I = imread("C:\Users\admin\Pictures\birds.jpg");  subplot (2,2,1);  imshow(I);  title('Original Image');  s= input ('Enter Scaling Factor');  j=imresize (I,10);  subplot (2,2,2);  imshow(j);  title('Scaled Image');  K= imrotate (I,45);  subplot (2,2,3);  imshow (K);  title('Rotated Image 45deg');  R =imrotate (I,180);  subplot (2,2,4);  imshow (R);  title('Rotated Image 180deg'); |

OUTPUT:





# PRACTICAL NO: -8

AIM: MATLAB program for edge detection, gray level Thresholding in Image Segmentation.

Code: -

% Following MATLAB function will take a grayscale

% or an RGB image as input and will return a

% binary image as output

|  |
| --- |
| function [binary] = convert2binary(img)       [x, y, z]=size(img);       % if Read Image is an RGB Image then convert       % it to a Gray Scale Image For an RGB image       % the value of z will be 3 and for a Grayscale       % Image the value of z will be 1      if z==3           img=rgb2gray(img);      end      % change the class of image      % array from 'unit8' to 'double'      img=double(img);      % Calculate sum of all the gray level      % pixel's value of the GrayScale Image      sum=0;      for i=1:x           for j=1:y          sum=sum+img(i, j);       end       end      % Calculate Threshold value by dividing the      % calculated sum by total number of pixels      % total number of pixels = rows\*columns (i.e x\*y)      threshold=sum/(x\*y);      % Create a image array having same number      % of rows and column as Original image      % with all elements as 0 (Zero).      binary=zeros(x, y);      % iterate over all the pixels of Grayscale      % Image and Assign 1 to binary(i, j), if gray      % level value is >=  threshold value      % else assign 0 to binary(i, j)      for i=1:x       for j=1:y          if img(i, j) >= threshold                  binary(i, j) = 1;          else              binary(i, j)=0;          end       end      end  end  % driver function  % Read the target Image  img=imread('apple.png');  % Call convert2binary() function to convert  % Image to binary using thresholding  binary\_image=convert2binary(img);  % Display result  imshow(binary\_image); |

OUTPUT: -

Practical no: 9

Aim: Write A program on Discrete cosine Transform

Code:

|  |
| --- |
| import java.util.\*;  class GFG  {  public static int n = 8,m = 8;  public static double pi = 3.142857;  static strictfp void dctTransform(int matrix[][])  {  int i, j, k, l;  double ci, cj, dct1, sum;  for (i = 0; i < m; i++)  {  for (j = 0; j < n; j++)  {  if (i == 0)  ci = 1 / Math.sqrt(m);  else  ci = Math.sqrt(2) / Math.sqrt(m);  if (j == 0)  cj = 1 / Math.sqrt(n);  else  cj = Math.sqrt(2) / Math.sqrt(n);  sum = 0;  for (k = 0; k < m; k++)  {  for (l = 0; l < n; l++)  {  Math.cos((2 \* l + 1) \* j \* pi / (2 \* n));  }  dct1 = matrix[k][l] \*Math.cos((2 \* k + 1) \* i \* pi / (2 \* m)) \*  sum = sum + dct1;  }  dct[i][j] = ci \* cj \* sum;  }  }  for (i = 0; i < m; i++)  {  for (j = 0; j < n; j++)  System.out.printf("%f\t", dct[i][j]);  System.out.println();  }  }  public static void main (String[] args)  {  int matrix[][] = { { 255, 255, 255, 255, 255, 255, 255, 255 },  {255, 255, 255, 255, 255, 255, 255, 255},  {255, 255, 255, 255, 255, 255, 255, 255},  {255, 255, 255, 255, 255, 255, 255, 255},  {255, 255, 255, 255, 255, 255, 255, 255},  {255, 255, 255, 255, 255, 255, 255, 255},  {255, 255, 255, 255, 255, 255, 255, 255},  {255, 255, 255, 255, 255, 255, 255, 255}};  }  }  dctTransform(matrix); |

Output;-

