RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN

CD19P02 – FUNDAMENTALS OF IMAGE PROCESSING

LABORATORY RECORD

Name : . . . .

Year / Branch / Section : . . . .

Register No. : . . . .

Semester : . . . .

Academic Year : . . . .

CD19P02 – FUNDAMENTALS OF IMAGE PROCESSING

List of Experiments	
1.	Practice of important image processing commands – imread(), imwrite(), imshow(), plot() etc.
2.	Program to perform Arithmetic and logical operations
3.	Program to implement sets operations, local averaging using neighborhood processing.
4.	Program to implement Convolution operation.
5.	Program to implement Histogram Equalization.
6.	Program to implement Mean Filter.
7.	Program to implement Order Statistic Filters
8.	Program to remove various types of noise in an image
9.	Program to implement Sobel operator.

INDEX

EXP.No	DATE	NAME OF THE EXPERIMENT	SIGN
1		Practice of important image processing commands – imread(), imwrite(), imshow(), plot() etc.	
2a		Program to perform Arithmetic and logical operations	
2b		Program to perform logical operations	
3a		Program to implement sets operations using neighborhood processing.	
3b		Program to implement local averaging using neighborhood processing.	
4		Program to implement Convolution operation.	
5		Program to implement Histogram Equalization.	
6		Program to implement Mean Filter.	
7		Program to implement Order Statistic Filters	
8		Program to remove various types of noise in an image	
9		Program to implement Sobel operator.	
10			

INTRODUCTION TO MATLAB

MATLAB stands for MATrix LABoratory and the software is built up around vectors and matrices. It is a technical computing environment for high performance numeric computation and visualization. It integrates numerical analysis, matrix computation, signal processing and graphics in an easy-to-use environment, where problems and solutions are expressed just as they are written mathematically, without traditional programming. MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning. It enables us to solve many numerical problems in a fraction of the time that it would take to write a program and execute in a language such as FORTRAN, BASIC, or C. It also features a family of application specific solutions, called toolboxes. Areas in which toolboxes are available include signal processing, image processing, control systems design, dynamic systems simulation, systems identification, neural networks, wavelength communication and others. It can handle linear, non-linear, continuous-time, discretetime, multivariable and multirate systems.

How to start MATLAB

Choose the submenu "Programs" from the "Start" menu. From the "Programs" menu, open the "MATLAB" submenu. From the "MATLAB" submenu, choose "MATLAB".

Procedure

1. Open Matlab.
2. File New Script.
3. Type the program in untitled window
4. File Save type filename.m in Matlab workspace path.
5. Debug Run.
6. Output will be displayed at Figure dialog box.

Library Functions

clc:

Clear command window

Clears the command window and homes the cursor.

clear all:

Removes all variables from the workspace.

close all:

Closes all the open figure windows.

exp:

Y = exp(X) returns the exponential e x for each element in array X.

linespace:

y = linspace(x1,x2) returns a row vector of 100 evenly spaced points between x1 and x2.

rand:

X = rand returns a single uniformly distributed random number in the interval (0,1).

ones:

X = ones(n) returns an n-by-n matrix of ones.

zeros:

X = zeros(n) returns an n-by-n matrix of zeros.

plot:

plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

subplot:

subplot(m,n,p) divides the current figure into an m-by-n grid and creates an axes for a subplot in the position specified by p.

stem:

stem(Y) plots the data sequence, Y, as stems that extend from a baseline along the x-axis. The data values are indicated by circles terminating each stem.

title:

title(str) adds the title consisting of a string, str, at the top and in the center of the current axes.

xlabel:
xlabel(str) labels the x-axis of the current axes with the text specified by str.
ylabel:
ylabel(str) labels the y-axis of the current axes with the string, str.

A Summary of Matlab Commands Used

imread	Read image from graphics file
imwrite	Write image to graphics file
imfinfo	Information about graphics file
imshow	Display Image
Implay	Play movies, videos or image sequences
gray2ind	Convert grayscale to indexed image
ind2gray	Convert indexed image to grayscale image
mat2gray	Convert matrix to grayscale image
rgb2gray	Convert RGB image or colormap to grayscale
imbinarize	Binarize image by thresholding
adapthresh	Adaptive image threshold using local first-order statistics
otsuthresh	Global histogram threshold using Otsu's method
im2uint16	Convert image to 16-bit unsigned integers
im2uint8	Convert image to 8-bit unsigned integers
imcrop	Crop image
imresize	Resize image
imrotate	Rotate image
imadjust	Adjust image intensity values or colormap
imcontrast	Adjust Contrast tool
imsharpen	Sharpen image using unsharp masking
histeq	Enhance contrast using histogram equalization
adapthisteq	Contrast-limited adaptive histogram equalization (CLAHE)
imhistmatch	Adjust histogram of image to match N-bin histogram of reference image
imnoise	Add noise to image
imfilter	N-D filtering of multidimensional images
fspecial	Create predefined 2-D filter
weiner2	2-D adaptive noise-removal filtering
medfilt2	2-D median filtering
ordfilt2	2-D order-statistic filtering
imfill	Fill image regions and holes
imclose	Morphologically close image
imdilate	Dilate image
imerode	Erode image
imopen	Morphologically open image
imreconstruct	Morphological reconstruction
watershed	Watershed transform
dct2	2-D discrete cosine transform

hough	Hough transform
graydist	Gray-weighted distance transform of grayscale image
fft2	2-D fast Fourier transform
ifftshift	Inverse FFT shift
imcomplement	Complement image
immultiply	Multiply two images or multiply image by constant
imsubtract	Subtract one image from another or subtract constant from image
imdivide	Divide one image into another or divide image by constant
imadd	Add two images or add constant to image

Ex.No: 1 IMPLEMENTATION OF IMAGE PROCESSING COMMANDS

Date:

Aim :
To Perform important image processing commands using Matlab.

Software Used:

MATLAB

Program : (1A)

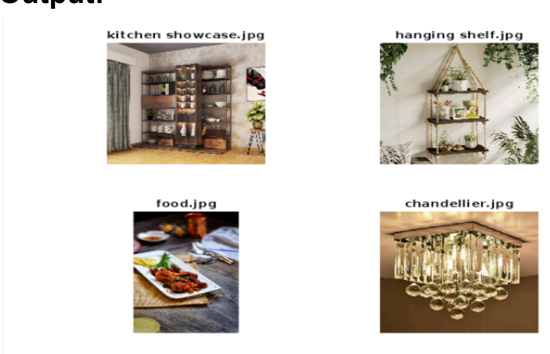
```
clear
close all
clc
I = imread('mountain1.jpg');
imshow(I);
```

Output:

Program:(1B)

```
subplot(2,2,1), imshow('kitchen showcase.jpg'),title('kitchen showcase.jpg');
subplot(2,2,2), imshow('hanging shelf.jpg'),title('hanging shelf.jpg');
subplot(2,2,3), imshow('food.jpg'),title('food.jpg');
subplot(2,2,4), imshow("chandellier.jpg"),title("chandellier.jpg");
impixelinfo;
imageinfo('kitchen showcase.jpg');
imageinfo('hanging shelf.jpg');
imageinfo('food.jpg');
imageinfo("chandellier.jpg");
```

Output:



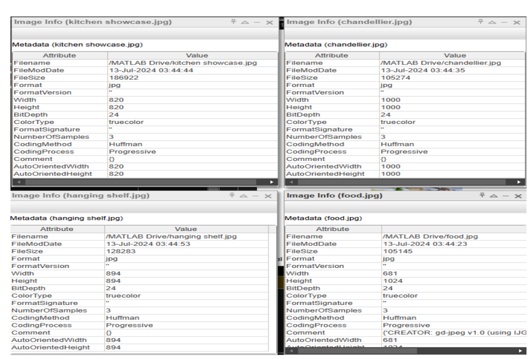


Image Info (kitchen showcase.jpg)

Metadata (kitchen showcase.jpg)

Attribute	Value
Filename	/MATLAB Drive/kitchen showcase.jpg
FileModDate	13-Jul-2024 03:44:44
FileSize	186922
Format	jpg
FormatVersion	''
Width	820
Height	820
BitDepth	24
ColorType	truecolor
FormatSignature	''
NumberOfSamples	3
CodingMethod	Huffman
CodingProcess	Progressive
Comment	{}
AutoOrientedWidth	820
AutoOrientedHeight	820

Image Info (chandellier.jpg)

Metadata (chandellier.jpg)

Attribute	Value
Filename	/MATLAB Drive/chandellier.jpg
FileModDate	13-Jul-2024 03:44:35
FileSize	105274
Format	jpg
FormatVersion	''
Width	1000
Height	1000
BitDepth	24
ColorType	truecolor
FormatSignature	''
NumberOfSamples	3
CodingMethod	Huffman
CodingProcess	Progressive
Comment	{}
AutoOrientedWidth	1000
AutoOrientedHeight	1000

Image Info (hanging shelf.jpg)

Metadata (hanging shelf.jpg)

Attribute	Value
Filename	/MATLAB Drive/hanging shelf.jpg
FileModDate	13-Jul-2024 03:44:53
FileSize	128283
Format	jpg
FormatVersion	''
Width	894
Height	894
BitDepth	24
ColorType	truecolor
FormatSignature	''
NumberOfSamples	3
CodingMethod	Huffman
CodingProcess	Progressive
Comment	{}
AutoOrientedWidth	894
AutoOrientedHeight	894

Image Info (food.jpg)

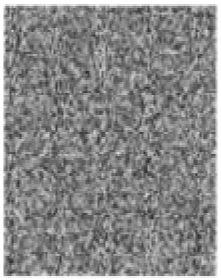
Metadata (food.jpg)

Attribute	
Filename	/MATLAB Drive/food.jpg
FileModDate	13-Jul-2024 03:44:23
FileSize	105145
Format	jpg
FormatVersion	''
Width	681
Height	1024
BitDepth	24
ColorType	truecolor
FormatSignature	''
NumberOfSamples	3
CodingMethod	Huffman
CodingProcess	Progressive
Comment	{'CREATOR: gd-jpeg v1.0 (using IJG
AutoOrientedWidth	681

Program : (1C)

```
clc;
clear all;
close all;
A = rand(150);
imwrite(A,'mountain.jpg');
imshow('mountain.jpg')
```

Output:



Program : (1D)

```
clc;
clear all;
close all;
load clown.mat
newmap = copper(81);
imwrite(X,newmap,'copperclown.png');
imshow('copperclown.png');
```

Output:

Result:

The important image commands have been displayed and studied

Ex.No: 2A **IMPLEMENTATION OF ARITHMETIC OPERATIONS**

Date:

Aim :

To implement arithmetic operations of an image using Matlab.

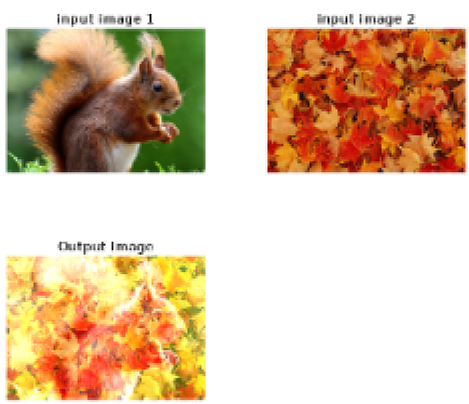
Software Used:

MATLAB

Program : (A)

```
clc;
close all;
clear all;
I = imread('Squirrel.jpg');
c = imread('autumn-leaves.jpg');
K = imadd(c, I);
figure;
subplot(2,2,1);imshow(I);title('input image 1');
subplot(2,2,2);imshow(c);title('input image 2');
subplot(2,2,3);imshow(K);title('Output image');
```

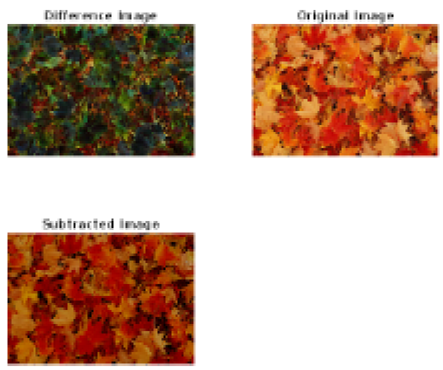
Output:



Program : (B)

```
close all;
clear;
I = imread("autumn-leaves.jpg");
background = imopen(I,strel('disk',15));
Ip = imsubtract(I,background);
Iq = imsubtract(I,50);
figure
subplot(2,2,1), imshow(Ip,[]), title('Difference Image')
subplot(2,2,2), imshow(I), title('Original Image');
subplot(2,2,3), imshow(Iq), title('Subtracted Image');
```

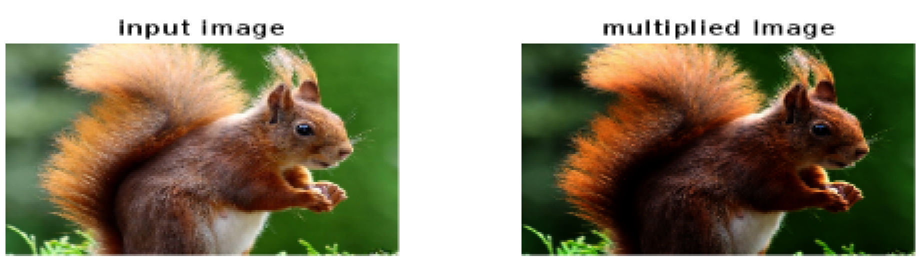
Output:



Program : (C)

```
clc;
close all;
clear all;
I = imread('Squirrel.jpg');
I16 = uint16(I);
J = immultiply(I16,I16);
subplot(1,2,1), imshow(I), title('input image');
subplot(1,2,2), imshow(J), title('multiplied Image');
```

Output:



Program : (D)

```
clc;
clear all;
close all;
I = imread('hanging shelf.jpg');
J = imdivide(I,2);
subplot(1,2,1), imshow(I), title('Input Image');
subplot(1,2,2), imshow(J), title('Output Image');
```

Output:

Result:

Thus the arithmetic operations of an image have been implemented using MATLAB.

Ex.No: 2B **IMPLEMENTATION OF LOGICAL OPERATIONS**

Date:

Aim :
To implement logical operations of an image using Matlab.

Software Used:
MATLAB

Program : (AND Operation)

```
imageSize = [200, 200];
i = zeros(imageSize);
rowStart = 100;
rowEnd = 150;
colStart = 50;
colEnd = 80;
i(rowStart:rowEnd, colStart:colEnd) = 1;
imageSize = [200, 200];
j = ones(imageSize);
resultImage = i & j;
subplot(1, 3, 1), imshow(i), title('Image 1');
subplot(1, 3, 2), imshow(j), title('Image 2');
subplot(1, 3, 3), imshow(resultImage), title('Output Image');
```

Output:

Program : (OR operation)

```
imageSize = [200, 200];
i = zeros(imageSize);
rowStart = 80;
rowEnd = 120;
colStart = 50;
colEnd = 120;
i(rowStart:rowEnd, colStart:colEnd) = 1;
imageSize = [200, 200];
j = ones(imageSize);
resultImage = i | j;
subplot(1, 3, 1), imshow(i), title('Image 1');
subplot(1, 3, 2), imshow(j), title('Image 2');
subplot(1, 3, 3), imshow(resultImage), title('Output Image');
```

Output:

Program : (NOT Operation)

```
imageSize = [200, 200];
i = zeros(imageSize);
rowStart = 90;
rowEnd = 110;
colStart = 50;
colEnd = 140;
i(rowStart:rowEnd, colStart:colEnd) = 1;
resultImage = ~i ;
subplot(2, 2, 1), imshow(i), title('Input Image ');
subplot(2, 2, 2), imshow(resultImage), title('Output Image');
```

Output:

Program : (XOR Operation)

```
imageSize = [200, 200];
i = zeros(imageSize);
rowStart = 20;
rowEnd = 100;
colStart = 40;
colEnd = 120;
i(rowStart:rowEnd, colStart:colEnd) = 1;
imageSize = [200, 200];
j = ones(imageSize);
resultImage = xor(i,j);
subplot(1, 3, 1), imshow(i), title('Image 1');
subplot(1, 3, 2), imshow(j), title('Image 2');
subplot(1, 3, 3), imshow(resultImage), title('Output Image');
```

Output:

Result:

Thus the logical operations of an image have been implemented using MATLAB.

Ex.No: 3A **IMPLEMENTATION OF SET OPERATIONS**

Date:

Aim :

To implement Set operations of an image using Matlab.

Software Used:

MATLAB

Program :

```
A = imread("blueberry.jpg");
B = imread("strawberry.jpg");
imageA = imresize(A, [200,200]);
imageB = imresize(B, [200,200]);
if ~isequal(size(imageA), size(imageB))
error("Input images must have the same dimensions");
end
unionImage = max(imageA, imageB);
intersectionImage = min(imageA, imageB);
complementImageA = 255 - imageA;
differenceImage = abs(imageA - imageB);
subplot(2, 3, 1);imshow(imageA);title('Image A');
subplot(2, 3, 2);imshow(imageB);title('Image B');
subplot(2, 3, 3);imshow(unionImage);title('Union (Max)');
subplot(2, 3, 4);imshow(intersectionImage);title('Intersection (Min)');
subplot(2, 3, 5);imshow(complementImageA);title('Complement of A');
subplot(2, 3, 6);imshow(differenceImage);title('Difference');
imwrite(unionImage, 'union_image.jpg');
imwrite(intersectionImage,'intersection_image.jpg');
imwrite(complementImageA,'complement_imageA.jpg');
imwrite(differenceImage,'difference_image.jpg');
disp('Set operation images saved.');
```

Output:

Result:

Thus, the set operations of an image have been implemented using MATLAB.

Ex.No: 3B **IMPLEMENTATION OF LOCAL AVERAGING USING NEIGHBORHOOD PROCESSING**

Date:

Aim :

To implement local averaging using neighborhood processing in an image using Matlab.

Software Used:

MATLAB

Program :

```
inputImage = imread('rose.jpg');
path
neighborhoodSize = 3;
filter = fspecial('average', neighborhoodSize);
averagedImage = imfilter(inputImage, filter);
subplot(1, 2, 1);
imshow(inputImage);
title('Original Image');
subplot(1, 2, 2);
imshow(averagedImage);
title('Averaged Image');
imwrite(averagedImage, 'averaged_image.jpg');
disp('Averaged image saved as &quot;averaged_image.jpg&quot');
```

Output:

Result:

Thus, the local averaging using neighborhoods processing of an image have been implemented using MATLAB.

Ex.No: 4 **IMPLEMENTATION OF CONVOLUTION OPERATION**

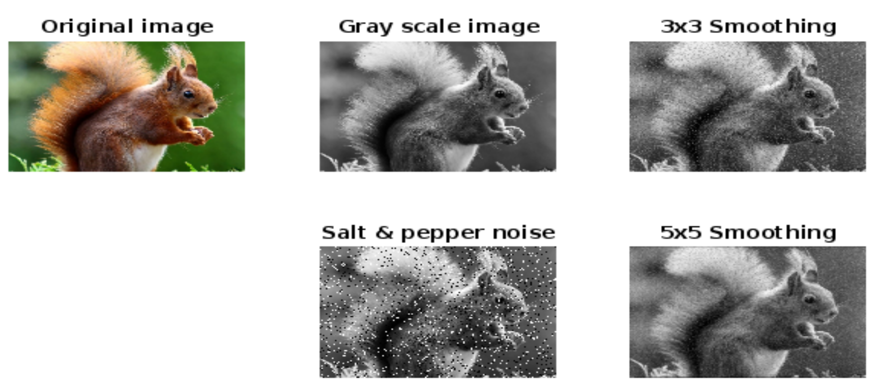
Date:

Aim :
To implement Convolution operation of an image using Matlab.

Software Used:
MATLAB

Program :
```
clc;
clear all;
close all;
a=imread('Squirrel.jpg');
subplot(3,3,1);imshow(a);title('Original image');
b=rgb2gray(a);
subplot(3,3,2);imshow(b);title('Gray scale image');
c=imnoise(b,'salt & pepper',0.1);
subplot(3,3,5);imshow(c);title('Salt & pepper noise');
h1=1/9*ones(3,3);
c1=conv2(c,h1,'same');
subplot(3,3,3);imshow(uint8(c1));title('3x3 Smoothing');
h2=1/25*ones(5,5);
c2=conv2(c,h2,'same');
subplot(3,3,6);imshow(uint8(c2));title('5x5 Smoothing');
```

Output:



Result:
Thus, the convolution operations of an image have been implemented using MATLAB.

Ex.No: 5 **IMPLEMENTATION OF HISTOGRAM EQUALIZATION**

Date:

Aim :

To implement Histogram equalization of an image using Matlab.

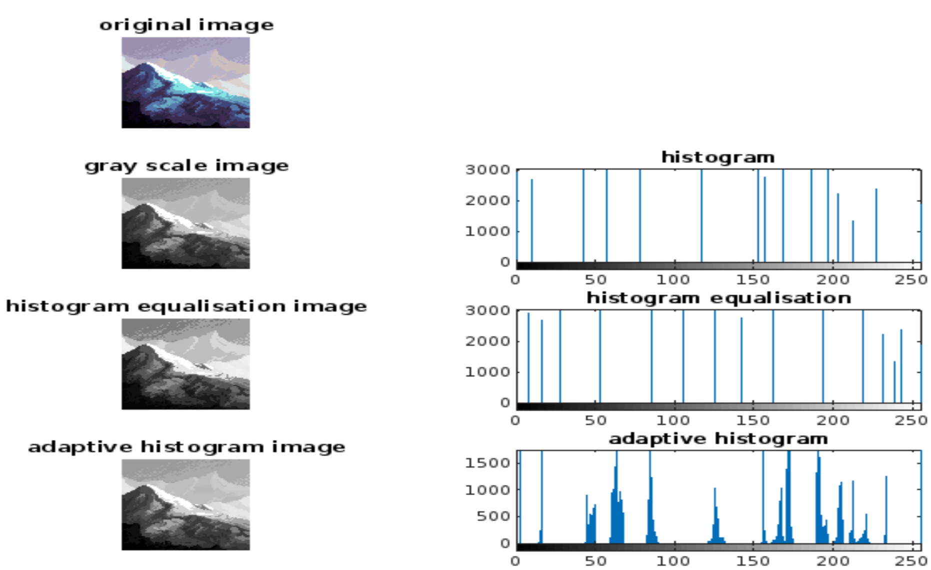
Software Used:

MATLAB

Program :

```
a= imread("blue-mountain.png");
subplot(4,2,1);imshow(a);title("original image");
b=rgb2gray(a);
subplot(4,2,3);imshow(b);title("gray scale image");
subplot(4,2,4);imhist(b);title("histogram");
subplot(4,2,5);c=histeq(b);
imshow(c);title("histogram equalisation image");
subplot(4,2,6);imhist(c);title("histogram equalisation");
subplot(4,2,7);f=adapthisteq(b);
imshow(f);title("adaptive histogram image");
subplot(4,2,8);imhist(f);title("adaptive histogram");
```

Output:



Result:

Thus, the Histogram equalization of an image has been implemented using MATLAB.

Ex.No: 5A **IMPLEMENTATION OF CORRELATION BETWEEN THE VISUAL QUANTITY OF AN IMAGE**

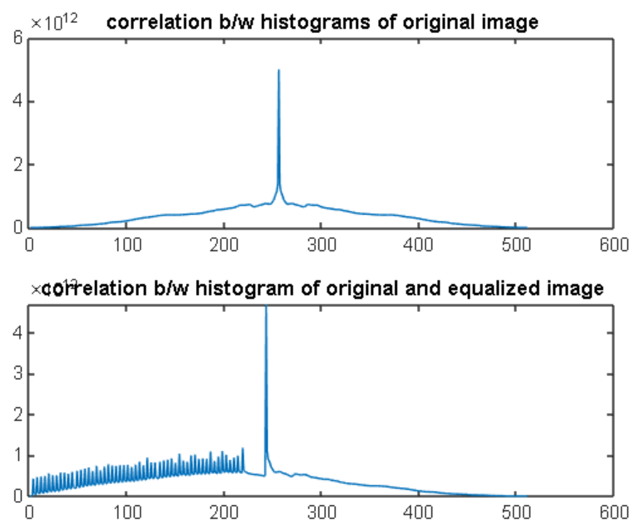
Date:

Aim :
To study the correlation between the visual quality of an image with its histogram.

Software Used:
MATLAB

Program :
```
img= imread ('color.jpg');
img=rgb2gray(img);
[ count , cells ]= imhist (img) ;
Iheq = histeq(img);
[count1,cells1 ]= imhist (Iheq);
corrbsameimg = corr2(img,Iheq)
disp(corrbsameimg);
x = xcorr ( count , count ) ;
x1 = xcorr ( count , count1 ) ;
subplot(2,1,1);
plot(x);
title('correlation b/w histograms of original image');
subplot(2,1,2);
plot(x1)
title('correlation b/w histogram of original and equalized image')
```

Output:



Result:
Thus, the correlation between visual quantity of an image have been implemented using MATLAB.

Ex.No: 6 **IMPLEMENTATION OF MEAN FILTER**

Date:

Aim :

To implement mean filter in an image reduce noise in digital images using Matlab.

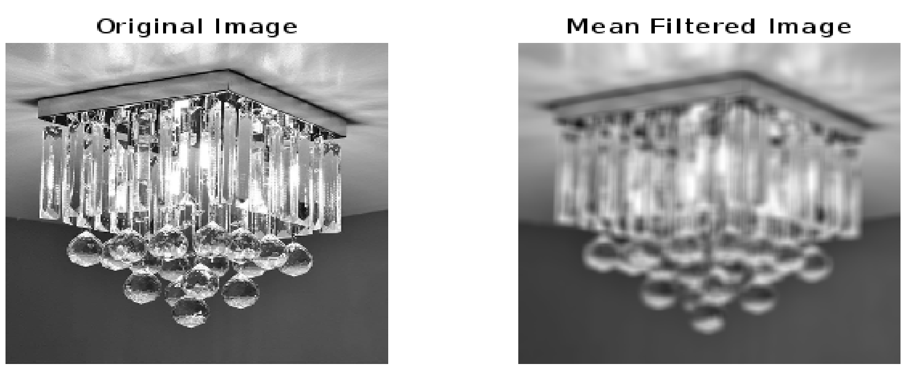
Software Used:

MATLAB

Program :

```
A = imread("chandellier.jpg");
B = rgb2gray(A)
filterSize = 25;
paddedImage = padarray(B, [filterSize, filterSize], "replicate");
outputImage = zeros(size(B));
for i = 1:size(B, 1)
 for j = 1:size(B, 2)
   neighborhood = paddedImage(i:i+filterSize-1, j:j+filterSize-1);
   meanValue = mean(neighborhood(:));
   outputImage(i, j) = meanValue;
 end
end
subplot(1, 2, 1);
imshow(B);
title("Original Image");
subplot(1, 2, 2);
imshow(uint8(outputImage));
title("Mean Filtered Image");
```

Output:



Result:

The noise in an image is reduced using a mean filter, and it has been implemented using MATLAB.

Ex.No: 7 **IMPLEMENTATION OF ORDER STATISTICS FILTERS**

Date:

Aim :

To implement Order Statistics filters in an image using Matlab.

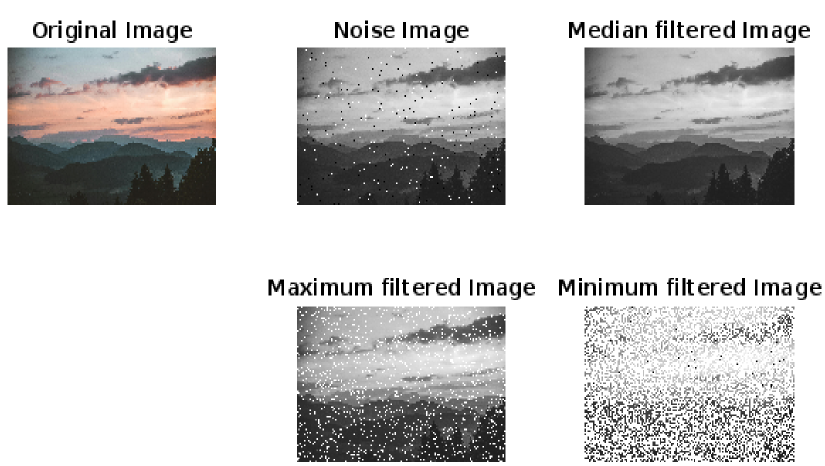
Software Used:

MATLAB

Program :

```
clc;
clear all;
close all;
b = imread("mont.jpeg");
subplot(3,3,1);imshow(b);title('Original Image');
a=rgb2gray(b);
a = im2double(a);
a = imnoise(a,'salt & pepper',0.02);
subplot(3,3,2);imshow(a);title('Noise Image');
I = medfilt2(a);
subplot(3,3,3);imshow(I);title('Median filtered Image');
x=rand(size(a));a(x(:)< 0.05)=0;
max_Img = ordfilt2(a,9,ones(3,3));
subplot(3,3,5);imshow(max_Img);title('Maximum filtered Image');
a(x(:)< 0.95)=255;min_Img = ordfilt2(a,1,ones(3,3));
subplot(3,3,6);imshow(min_Img);title('Minimum filtered Image');
```

Output:



Result:

The different Order Statistics filters in an image have been implemented using MATLAB.

Ex.No: 8 **REMOVE VARIOUS TYPES OF NOISE IN AN IMAGE**

Date:

Aim :
To Remove Various types of Noise in an Image an image using Matlab.

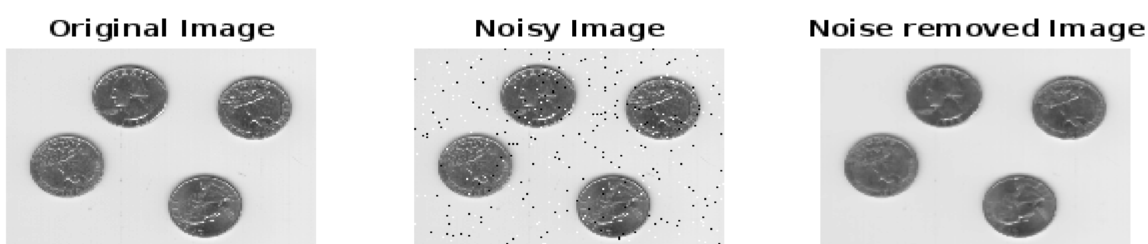
Software Used:

MATLAB

Program : (Salt and Pepper Noise)

```
I = imread("eight.tif");
J = imnoise(I,"salt & pepper",0.02);
subplot(2,3,1),imshow(I),title("Original Image");
subplot(2,3,2),imshow(J),title("Noisy Image");
Kmedian = medfilt2(J);
subplot(2,3,3),imshow(Kmedian),title("Noise removed Image");
```

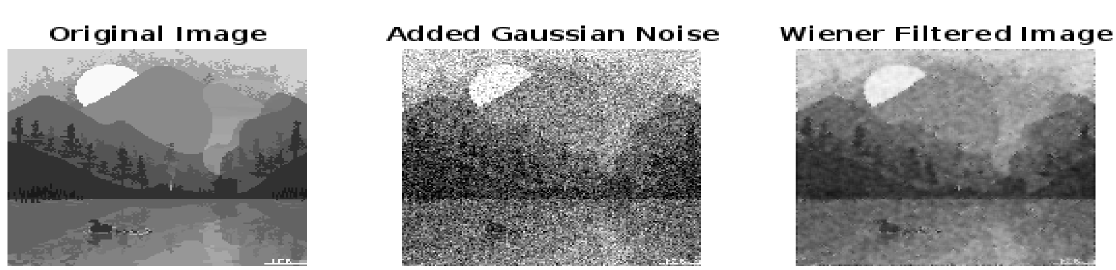
Output:



Program:(Gaussian Noise)

```
RGB = imread("red mountain.png");
I = im2gray(RGB);
J = imnoise(I,'gaussian',0,0.025);
K = wiener2(J,[5 5]);
subplot(2,3,1);
imshow(I)
title('Original Image');
subplot(2,3,2);
imshow(J)
title('Added Gaussian Noise');
subplot(2,3,3);
imshow(K);
title('Wiener Filtered Image');
```

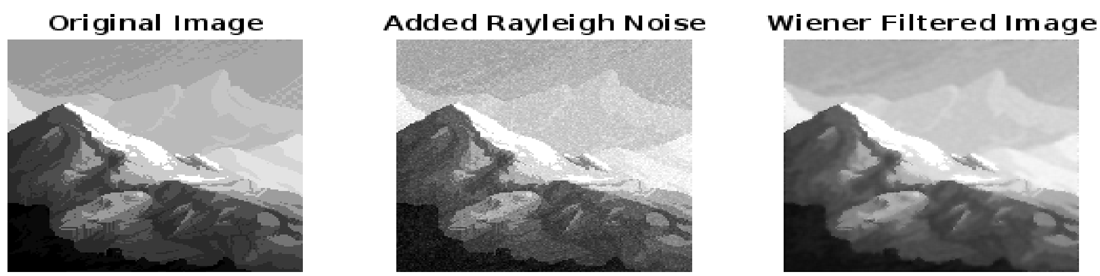
Output:



Program : (Rayleigh Noise)

```
RGB = imread('blue-mountain.png' );
I = im2gray(RGB);
rayleighNoise = raylrnd(0.05, size(I));
J = im2double(I) + rayleighNoise;
K = wiener2(J, [5 5]);
subplot(2,3,1);
imshow(I)
title('Original Image');
subplot(2,3,2);
imshow(J)
title('Added Rayleigh Noise');
subplot(2,3,3);
imshow(K);
title('Wiener Filtered Image');
```

Output:



Program:(Erlang Noise)

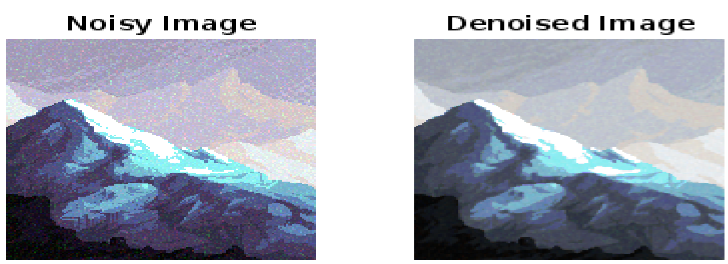
```
I = imread('red mountain.png');
scale = 10; shape= 5;
sizeSignal = size(I);
erlangNoise = scale*gamrnd(shape, 1, sizeSignal);
noisy = double(I) + erlangNoise;
noisy = min(max(noisy, 0), 255);
noisy = uint8(noisy);
denoised=medfilt3(noisy);
figure;
subplot(2, 3, 1);imshow(I);title('Input Image');
subplot(2, 3, 2);imshow(noisy);title('Noisy Image');
subplot(2, 3, 3);imshow(denoised);title('Denoised Image');
```

Output:

Program:(Exponential Noise)

```
I = imread('blue-mountain.png');
lambda = 0.1;
sizeSignal = size(I);
exponentialNoise = -log(1 - rand(sizeSignal)) / lambda;
noisy = double(I) + exponentialNoise;
noisy = min(max(noisy, 0), 255);
noisy = uint8(noisy);
denoised=medfilt3(noisy);
figure;
subplot(2, 3, 1);imshow(noisy);title('Noisy Image');
subplot(2, 3, 2);imshow(denoised);title('Denoised Image');
```

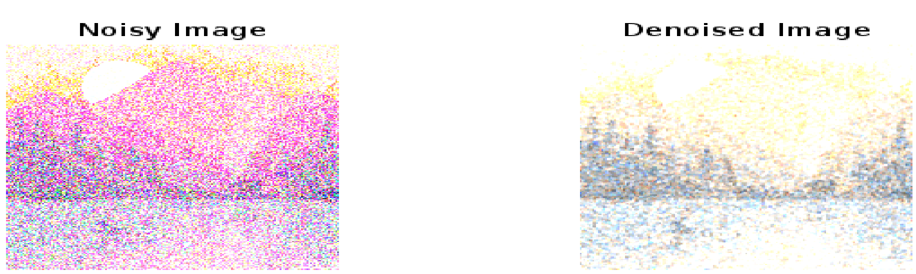
Output:



Program:(Uniform Noise)

```
I = imread('red mountain.png');
minValue = 0;
maxValue = 255;
sizeImage = size(I);
uniformNoise = (maxValue - minValue) * rand(sizeImage) + minValue;
noisy = double(I) + uniformNoise;
noisy = min(max(noisy, 0), 255);
noisy = uint8(noisy);
denoised=medfilt3(noisy);
figure;
subplot(2, 2, 1);imshow(noisy);title('Noisy Image');
subplot(2, 2, 2);imshow(denoised);title('Denoised Image');
```

Output:



Result:

Thus, the various types of noise in an image have been removed and implemented using MATLAB.

Ex No: 9 **IMPLEMENTATION OF SOBEL OPERATOR**

Date:

Aim :

To implement SOBEL operator in digital images for edge detection using Matlab.

Software Used:

MATLAB

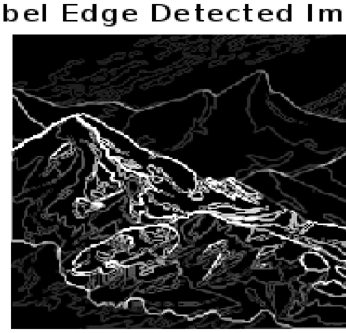
Program :

```
a = imread('blue-mountain.png');
b = rgb2gray(a);
gray_img = double(b);
h_kernel = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
v_kernel = [-1, -2, -1; 0, 0, 0; 1, 2, 1];
c = imfilter(gray_img, h_kernel);
d = imfilter(gray_img, v_kernel);
gradient_magnitude = sqrt(c.^2 + d.^2);
figure;
subplot(2, 2, 1);imshow(a);title('Original Image');
subplot(2, 2, 2);imshow(uint8(gradient_magnitude));
title('Sobel Edge Detected Image');
```

Output:

Original Image

Sobel Edge Detected Image



Result:

The SOBEL operator in digital images for edge detection has been implemented using MATLAB.

MINI - PROJECT

IMPLEMENTATION OF CAR PARKING INDICATOR USING MATLAB

Aim :
This project primarily aims to create a smart parking system with the help of MATLAB programming

Software Used:
MATLAB

Theory:
The goal of this experiment is to detect and count the number of cars in a parking lot image using image processing techniques. This involves converting the image to grayscale, enhancing features through edge detection, applying morphological operations to improve car separation, and finally, detecting and counting the individual cars based on their areas.

Steps Involved:

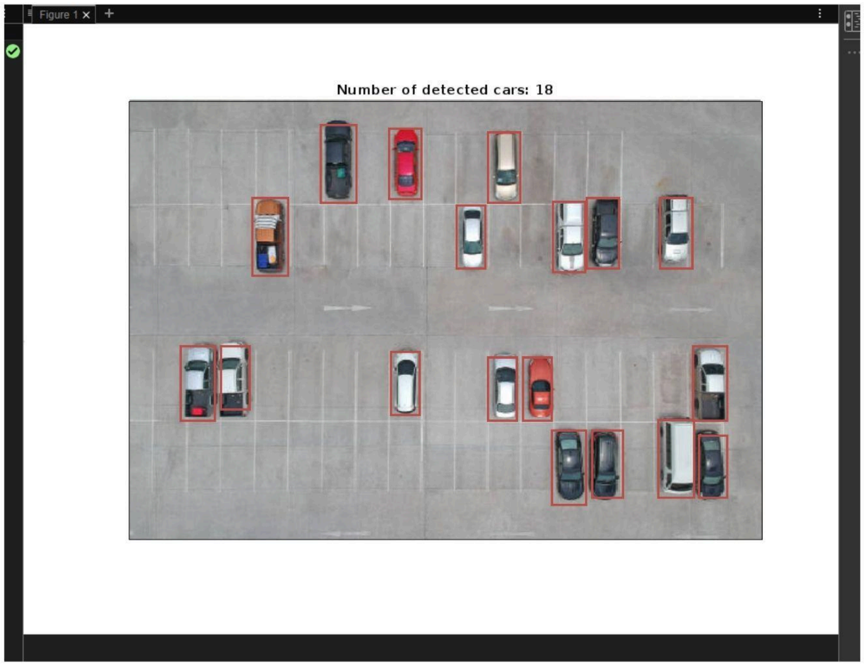
1. **Grayscale Conversion**: If the input image is in color (RGB), it is first converted to grayscale to simplify processing. Grayscale images contain intensity values, making it easier to perform subsequent analysis like edge detection.
2. **Noise Reduction**: A Gaussian filter is applied to the grayscale image to smooth the image and reduce noise. This is important to eliminate small, irrelevant details that could interfere with the car detection.
3. **Edge Detection**: The Canny edge detector is used to identify edges in the image. This step highlights the boundaries of objects (cars), making it easier to distinguish the cars from the background.
4. **Morphological Operations**: Dilation and erosion are used to enhance the detected edges. Dilation connects nearby edges, helping to identify adjacent cars, while erosion removes small noise that might falsely appear as a car.
5. **Component Labeling**: The `bwlabel` function is used to label connected components in the processed binary image. Each connected component corresponds to a potential car.
6. **Bounding Box and Area Filtering**: For each labeled component, the area is calculated. Components with areas within a specified range (e.g., 90 to 1500 pixels) are considered valid cars. A bounding box is drawn around each valid component to highlight the detected cars.
7. **Car Counting**: The number of cars is determined by counting the number of valid components. The number of available parking spaces is calculated by subtracting the detected car count from the total available spaces.

Program :

```
img = imread('car parking.jpg');
if size(img, 3) == 3
   grayImg = rgb2gray(img);
else
   grayImg = img;
end
filteredImg = imgaussfilt(grayImg, 2);
edges = edge(filteredImg, 'Canny', [0.15, 0.3]);
se = strel('disk', 5);
dilatedImg = imdilate(edges, se);
cleanedImg = imerode(dilatedImg, se);
[labeledImg, numLabels] = bwlabel(closedImg);
minAreaThreshold = 90;
maxAreaThreshold = 1500;
carCount = 0;
if size(img, 3) == 1
   resultImg = cat(3, img, img, img);
else
   resultImg = img;
end
for label = 1:numLabels
   component = (labeledImg == label);
   area = sum(component(:));

   if area >= minAreaThreshold && area <= maxAreaThreshold
       carCount = carCount + 1;
       stats = regionprops(component, 'BoundingBox');
       bbox = stats.BoundingBox;
       resultImg = insertShape(resultImg, 'Rectangle', bbox, 'Color', 'red',
'LineWidth', 2);
   end
end
figure;
imshow(resultImg);
title(['Number of detected cars: ', num2str(carCount)]);
totalSpaces = 76;
availableSpaces = totalSpaces - carCount;
disp(['Available parking spaces: ', num2str(availableSpaces)]);
```

Output:



Result:

The ‘`Available parking spaces: 58`’, is displayed in the Command Window. The car detection program was tested on a variety of parking lot images to verify its accuracy and effectiveness. The results from the program were compared to manual counts of the cars and visual inspection of the bounding boxes.