**AIM:** To write a code to read an image, convert color to grayscale and grayscale to binary image.

**CODE:**

a= imread('1.jpg');% To read image from Computer

b= rgb2gray(a);% To convert color to grayscale

c= im2bw(b);% To convert grayscale to binary image

imshow (c)

**AIM:** To write a Matlab code for obtaining Low pass filter on an image.

**Theory**: Low pass Filter in image processing is used for most smoothing methods. An image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels. This low pass filter image looks a lot blurrier. Often images can be noisy no matter how good the camera is, it always adds an amount of ‘snow’ into the image. The statistical nature of light itself also contributes noise to the image. Noise always changes rapidly from pixel to pixel because each pixel generates its own independent noise. So the low pas filters affects the noise more than it does the image. By suppressing the noise, gradual changes can be seen that were invisible before. Therefore, a low pass filter can be used to bring out faint details that were smothered by noise. Filtering can be visualised by drawing a “convolution kernel”. A kernel is a small grid showing how pixel’s filtered value depends on its neighbors. To perform a low-pass filter by simply averaging adjacent pixels, the following kernel is used.

**CODE:**

clear all;

close all;

clc

m=[1 1 1; 1 1 1; 1 1 1]/9;

a=imread('golden.jpg');

p=rgb2gray(a);

figure; imshow(p);

f=imnoise(p,'salt & pepper',.2);

title('original image');

for i=2:size(f,1)-1

for j=2:size(f,2)-1

s=0;

for l=-1:1

for k=-1:1

s=s+f(i+l,j+k)\*m(l+2,k+2);

end

end

newIm(i,j)=s;

end

end

figure

imshow(f)

title('salt and pepper image');

figure

imshow(newIm)

title('LPF image');

figure

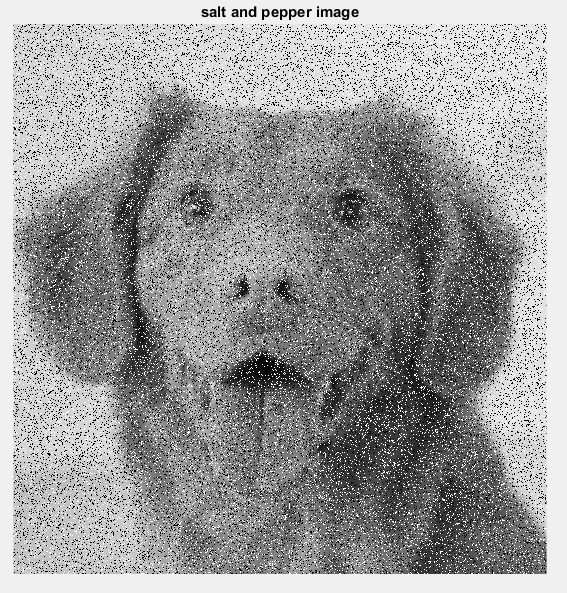
g=medfilt2(f);

imshow(g)

title ('median filter')

**OUTPUT:**









**Conclusion:** We can see that the low pass filter has retain the low frequency information within an image while reducing the high frequency information.

**AIM:** Write a Matlab code for obtaining High Pass Filter on an image.

**THEORY:** High Pass Filter in image processing is used to make an image appear sharper. An image is sharpened when contrast is enhanced between adjoining areas with little variation in brightness or darkness. High pass filtering can also cause small, faint details to be greatly exaggerated. An over-processed image will look grainy and unnatural, and point sources will have dark donuts around them. So, while high-pass filtering can often improve an image by sharpening detail, overdoing it can actually degrade the image quality significantly. The following kernel is used in high pass filter to increase brightness of the center pixel relative to neighboring pixel.

**CODE:**

clear all;

close all;

clc

m=[-1 -1 -1; -1 8 -1; -1 -1 1];

a=imread('car.jpg');

p=rgb2gray(a);

figure; imshow(p)

title('original image');

f=imnoise(p,'salt & pepper',.2);

for i=2:size(f,1)-1

for j=2:size(f,2)-1

s=0;

for l=-1:1

for k=-1:1

s=s+p(i+l,j+k)\*m(l+2,k+2);

end

end

newIm(i,j)=s;

end

end

figure

imshow(f)

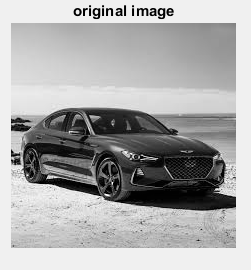
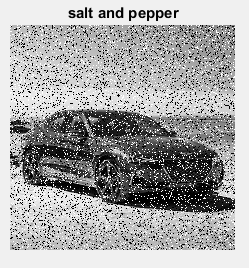
title('salt and pepper');

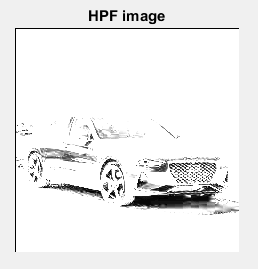
figure

imshow(newIm)

title('HPF image');

**OUTPUT**

** **



**CONCLUSION:** Here we notice that High Pass Filter tends to retain the high frequency information within an image while reducing the low frequency information.

**AIM:** Write a Matlab code for bit plane slicing.

**THEORY:** Bit plane slicing is a fundamental technique of an image processing in which the image is sliced into different planes(each layer contains sequences of only binary digits 0 or 1). It ranges from plane 1 which contains the least significant bit (LSB) to the last plane N which contains the most significant bit (MSB) where the number of layers depends on the bit depth of the image. The bit depth means how many bits need to represent the pixel’s intensity. For example, if the image is grayscale then the bit depth is 8 bit and it will be seperated into 8 layers, or into 24 layers if the image is colored i.e bit depth is 24 bit. The first bit in the binary number is the least significant bit (LSB) which is not very efficient and its value is very small and will not effects on the pixel’s value. While the last bit is the most significant bit (MSB) which is efficient and contains the important weight of the pixel’s value.

**CODE:**

clc;

clear all;

close all;

x=imread('car.jpg');

y=size(x,1);

p=zeros(y,y,8);

for l=1:8

for m=1:y

for n=1:y

if mod(fix(x(m,n)/2^(l-1)),2)==1

p(m,n,l)=1;

else p(m,n,l)=0;

end

end

end

end

subplot(3,3,1); imshow(x,[]);

Bit1=p(:,:,1);

subplot(3,3,2); imshow(Bit1,[]);

Bit2=p(:,:,2);

subplot(3,3,3); imshow(Bit2,[]);

Bit3=p(:,:,3);

subplot(3,3,4); imshow(Bit3,[]);

Bit4=p(:,:,4);

subplot(3,3,5); imshow(Bit4,[]);

Bit5=p(:,:,5);

subplot(3,3,6); imshow(Bit5,[]);

Bit6=p(:,:,6);

subplot(3,3,7); imshow(Bit6,[]);

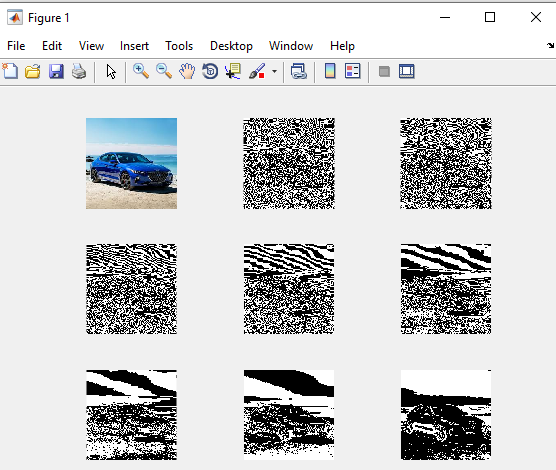
Bit7=p(:,:,7);

subplot(3,3,8); imshow(Bit7,[]);

Bit8=p(:,:,8);

subplot(3,3,9); imshow(Bit8,[]);

**OUTPUT**



**CONCLUSION:** Separating a digital image into its bit plane is useful for analyzing the relative importance played by each bit of an image, implying, it determines the adequacy of numbers of bits used to quantize each pixel, useful for image compression.

**AIM:** Write a Matlab code for converting the colored image to binary image.

**THEORY**: Binary images are those images whose pixel has only two possible intensity values and they are normally displayed as black and white. The usually way to generate a binary image is by thresholding. Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit that is 0 and 1.

**CODE:**

clc;

clear all;

close all;

i=imread('car.jpg');

v=rgb2gray(i);

a=im2bw(v,0.5);

imshow(i);

title('colour image');

figure

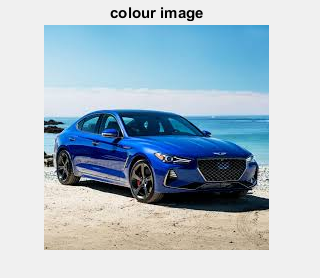
imshow(v);

title('gray image');

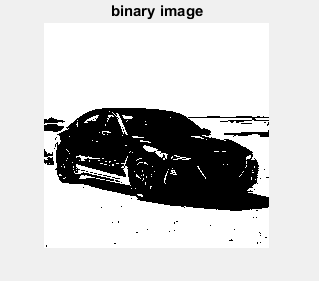
figure, imshow(a)

title('binary image');

**OUTPUT**







**CONCLUSION:** Selecting the correct threshold value is important while generating binary image. For the above given image the threshold value is given as 0.5. If we keep on increasing the threshold value the image seems to get darker.

**AIM:** To write a Matlab code for converting an image from RGB to Gray image.

**THEORY**: RGB stands for Red, Green and Blue. RGB refers to three hues of light that can be mixed together to create different colors. Combining red, green and blue light is the standard method of producing color images on screen, such as TVs, computer monitors ans smart phone**.** Gray scale image is one in which the value of each pixel is a single sample representing only amount of light, that is, it carries only intensity information. Therefore,Gray scale images are preferred over RGB or the colored image just to simplify mathematics. It is relatively easier to deal with the single color channel than multiple color channels.

**CODE:**

clc;

clear all;

close all;

i=imread('car.jpg');

v=rgb2gray(i);

imshow(i);

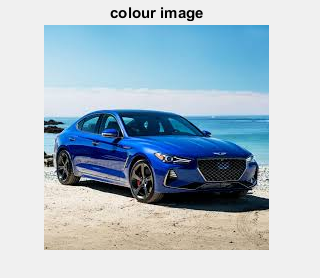
title('colour image');

figure

imshow(v);

title('gray image');

**OUTPUT:**





**CONCLUSION**: The rgb2gray function converts RGB images to grayscaleimage by eliminating the hue and saturation information while retaining the luminance.

**AIM:** Write a Matlab Code to flipped and image.

**THEORY:** Image rotation is a common image processing routine used to rotate images at any desired angle. The rotation operator performs a geometric transform which maps the position of a picture element in an input image onto a position in an output image by rotating it through a user specified angle about the origin. In most implementations, output locations which are outside the boundary of the image are ignored. This helps in image reversal, flipping and obtaining an intended view of the image. Image rotation has applications in matching, alignment and other image based algorithm.

**CODE:**

clc;

clear all;

close all;

i=imread('house.jpg');

v=rgb2gray(i);

imshow(i);

title('colour image');

figure

imshow(v);

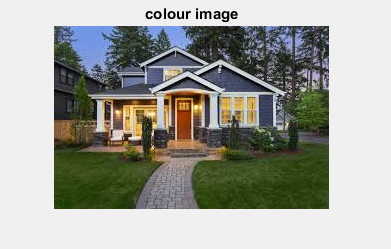
title('gray image');

C=flipdim(i, 1);

figure, imshow(C)

title('flipped image');

**OUTPUT**

****

****

****

**CONCLUSION:** The above code is to show how we can flip the image as per our desired angle. The output is the same as the input however with a rotation of 180 degree angle rotation from the original image.

**AIM:** Write a MATLAB code for adjusting intensity values of an image.

**THEORY:** Image enhancement techniques are used to improve an image, where improve is sometimes defined objectively ( example increase the signal -to-noise ration) and sometimes subjectively (example make certain features easier to see by modifying the colors or intensity). Intensity adjustment is an image enhancement technique that maps an image’s intensity values to a new range. We can adjust the intensity values in an image using the imadjust function, where we can specify the range of intensity values in the output image.

**CODE:**

clc;

clear all;

close all;

i=imread('camera.jpg');

v=rgb2gray(i);

imshow(i);

title('colour image');

figure

imshow(v);

title('gray image');

k = imadjust(v,[0 0.6],[0.5 1]);

figure, imshow(k);

title('adjust image');

**OUTPUT**



**CONCLUSION:** We can see that after decreasing the contrast of an image by narrowing the range of the data the adjusted image appears washed out making it much easier to see the details in the image.

**AIM:** To write a MATLAB code for Histogram Equalization.

**THEORY:** Histogram equalization is used for image enhancement and it is a graphical representation of any data. To get a high quality image or to get a high contrast image the histogram value should be distributed on the entire plane or it should have a flat profile. It is used for various image processing applications like manipulating the contrast value and also brightness of the digital image. Since the human eye is sensitive to contrast rather than absolute pixel intensities, we would perceive less information from an image with poor intensity distributions than from the same image with better intensity distributions. Images with skewed distributions can be helped with histogram equalization. Histogram equalization is a point process that redistributes the image's intensity distributions in order to obtain a uniform histogram for the image.

**CODE:**

clc;

clear all;

close all;

i=imread('car.jpg');

v=rgb2gray(i);

imshow(i);

title('colour image');

figure

imshow(v);

title('gray image');

k = imadjust(i,[], [0.2 0.5 ]);

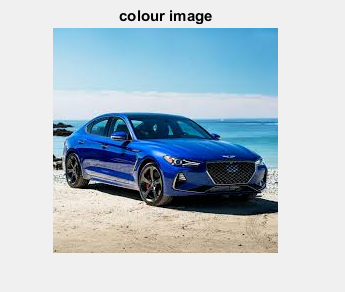
figure, imshow(k);

title('adjuast image');

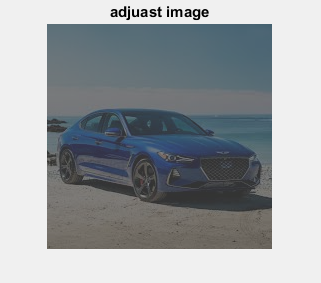
m = histeq(k);

figure, imshow (m);

**OUTPUT**

****

****

****

**CONCLUSION:** We can adjust the contrast of an image using histogram equalization, using the histeq function. Therefore, the above example shows the transformation in the intensity values. By default, the histogram equalization function, histeq, tries to match a flat histogram with 64 bins, but we can specify a different histogram instead as in this example.

**AIM:** Write a MATLAB code to compute laplacian filter in an image.

**THEORY:**A Laplacian filter is an edge detector used to compute the second derivative of an image, measuring the rate at which the first derivative change. Laplacian kernels usually contain negative values in a cross pattern, centered within the array. The corners are either zero or positive values. The center value can be either negative or positive. Because these kernel is approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often gaussian smoothed before applying the laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

**CODE:**

clear all;

close all;

clc;

m=[0 -1 0; -1 4 -1; 0 -1 0];

a=imread('car.jpg');

p=rgb2gray(a);

figure; imshow(p)

f=imnoise(p,'salt & pepper',.2);

for i=2:size(f,1)-1

for j=2:size(f,2)-1

s=0;

for l=-1:1

for k=-1:1

s=s+f(i+l,j+k)\*m(l+2,k+2);

end

end

newIm(i,j)=s;

end

end

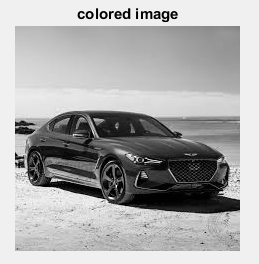
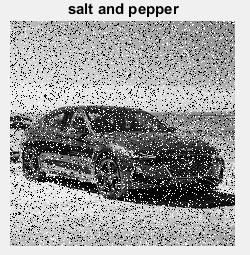
figure

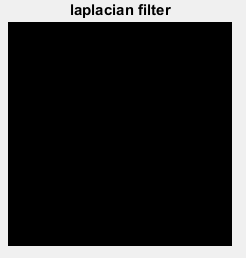
imshow(f)

figure

imshow(newIm)

**OUTPUT**

** **

****

**CONCLUSION**: The code shown above creates the three images for laplacian filter, each displaying in separate windows.

**AIM:** Write a code on different types of edge detection.

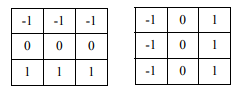
**THEORY**: Edge is a basic feature of an image. Edges define the boundaries between regions in an image and basic step for segmentation and object recognition. Edge detection is a process of identifying edges, the sharp change in image pixel intensity. Three types of edge detection are there. Namely, robert edge detection, sobel edge detection and prewitt edge detection

Robert edge detection:Roberts operator, approximate the gradient of an image through discrete differentiation. Robert mask is 2×2 matrix. Robert mask convolve with the entire image using horizontal and vertical Robert masks to give edge detected image in x direction and y direction respectively.

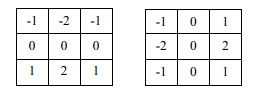
****

****

Prewitt mask: Prewitt mask is a discrete differentiation operator. This operator uses two 3×3 masks for calculating approximate derivative values in horizontal and vertical directions.

****

Sobel mask: Sobel mask is same as that of the Prewitt mask .There is only one difference, Sobel operator has ‘2’ and ‘-2’ values in center of first, third column of horizontal mask and first, third rows of vertical mask. This gives more weightage to the pixel values around the edge region, hence increases the edge intensity.

****

**CODE:**

clear all

close all

clc

m=[1 1 1;1 1 1;1 1 1]/9;

a=imread('jyoti.jpg');

p=rgb2gray(a);

figure, imshow(p)

BW1 = edge(p,'sobel');

figure,imshow(BW1);

title('using sobel')

BW1 = edge(p,'prewitt');

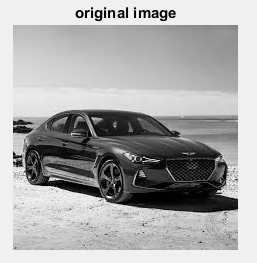
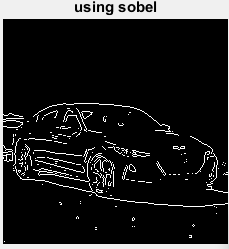
figure, imshow(BW1);

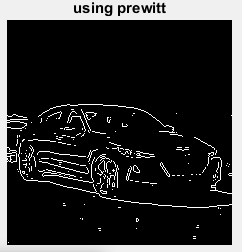
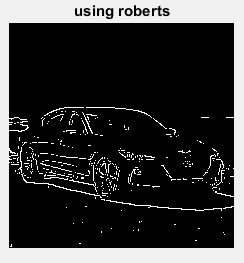
title('using prewitt')

BW1 = edge(p,'roberts');

figure,imshow(BW1);

title('using roberts')

**CONCLUSION:** We have successfully done the experiment on sobel, prewitt and roberts edge detection.

**AIM:** Write a code to resize the given image.

**THEORY:** Image resizing is necessary when you need to increase or decrease the total number of pixels. Image scaling can be interpreted as a form of image resampling or image reconstruction from the view of the [Nyquist sampling theorem](https://en.wikipedia.org/wiki/Nyquist_theorem). According to the theorem, downsampling to a smaller image from a higher-resolution original can only be carried out after applying a suitable 2D [anti-aliasing filter](https://en.wikipedia.org/wiki/Anti-aliasing_filter) to prevent aliasing artifacts. The image is reduced to the information that can be carried by the smaller image.

**CODE:**

clc;

clear all;

close all;

i=imread('house.jpg');

v=rgb2gray(i);

imshow(i);

title('colour image');

figure

imshow(v);

title('gray image');

C=flipdim(i, 1);

figure, imshow(C)

title('flipped image');

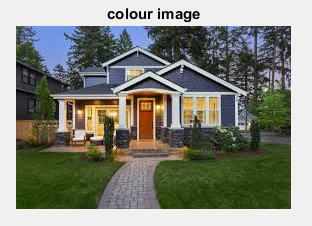
d=imresize(v,[100 100]);

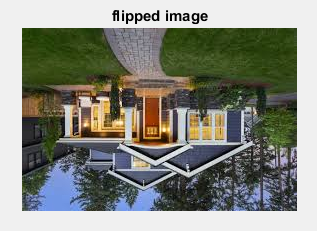
figure, imshow (d)

title('resized iamge');

imwrite(d,'resized\_image.jpg');

**OUTPUT**





**CONCLUSION:** Therefore we have sucessfully resize the image as our desired size.

**AIM:** Write a code for Historgram Equailzation without using the inbuild function.

**CODE:**

close all;

clear all;

clc

I=imread('download.jpg');% Image is loaded in I

pixels=size(I,1)\*size(I,2);% Total number of pixels in the Image

imshow(I);% Display the image stored in I

title('Original Image');

figure

%Probability Distribution Function

Hp=uint8(zeros(size(I,1),size(I,2)));%Image is returned as a numeric array of same size as the input image I

f=zeros(256,1);% Defined Variable for Frequency and preallocated with 256 zero value to the row

pf=zeros(256,1);% Defined Variable for Probability and preallocated with 256 zero value to the row

pc=zeros(256,1);% Defined Variable for Cumulative Probability and preallocated with 256 zero value to the row

c=zeros(256,1);% Defined Variable for Cumulative and preallocated with 256 zero value to the row

out=zeros(256,1);% Defined Variable for Rounding output and preallocated with 256 zero value to the row

%For Probability

for i=1:size(I,1)% Creation of loop for rows of the Image

for j=1:size(I,2)% Creation of loop for columns of the Image

A=I(i,j);%Defined Variable for representation of the number the Grey level from 0 to 255

f(A+1)=f(A+1)+1;% nos. of pixel incremented, Since zero indexing not allowed in Matlab, the grey level 0 to 255 is mapped from 1 to 256

pf(A+1)=f(A+1)/pixels;% Finding the Probability formula

end

end

%Cumulative Distribution Function

sum=0;% Defined Variable Sum with value zero

for i=1:size(pf)% creation of loop

sum=sum+f(i);% Formula for calculating Cumulaitve frequency

c(i)=sum;

pc(i)=c(i)/pixels;% Formula for Cumulative Probability

out(i)=round(pc(i)\*255);% Formula for calauclting the Value of L and Rounding

end

%Histogram equalization

for i=1:size(I,1)% Creation of loop for Rows in the Image

for j=1:size(I,2)% Creation of loop for Columns in the Image

Hp(i,j)=out(I(i,j)+1);% Replace new pixel values with the original values

end

end

imshow(Hp);

title('Histogram equalization without inbuild code');

figure

%Converting RGB image to grayscale

b=rgb2gray(I);

%Using build-in function

J = histeq(b) ;

imshow(J) ;

title('Histogram equalization using in built code');