SIES COLLEGE OF ARTS,SCIENCE & COMMERCE(AUTONOMOUS)
SION(W),MUMBAI-22

DEPARTMENT OF INFORMATION TECHNOLOGY

MSc(IT),SEMESTER II

Practical Journal

for

the subject
Digital Image Processing

Submitted by

Amit Kushwaha

FMIT2122011

for the academic year

2021-2022

SIES College of Arts, Science and Commerce (Autonomous), Sion (W), Mumbai – 400022

CERTIFICATE

This is to certify that Mr. / ~~Miss~~. **Amit Kushwaha** of M.Sc. [Information Technology] Semester 2, Seat No. **FMIT2122011** has successfully completed the practical journal for the subject of **Digital Image Processing** as a partial fulfilment of the degree M.Sc. [Information Technology] during the academic Year 2021-22.

Faculty-in-Charge
Dr. Satishkumar Sarjerao chavan

Examiner
Dr. Satishkumar Sarjerao chavan

Course Co-ordinator
Dr. Satishkumar Sarjerao chavan

College Seal

Date: 26/04/2022

INDEX

SR. NO.	PRACTICAL TITLE	SIGNATURE
1	Brightness Manipulation of an Image (dark, bright, plot histogram)	
2	Image Negative	
3	Thresholding for different values of Threshold	
4	Gray Level Slicing with & without background	
5	Contrast Stretching	
6	Histogram Plotting	
7	Histogram Equalization	
8	Spatial Filtering – blurring by average and weighted average (do it for noisy images)	
9	Spatial Filtering – blurring by median filter (do it for noisy images)	
10	Highpass filtering – Laplacian, Robert's cross gradient, Prewiit and Sobel	
11	Color image Processing – color model conversion and plane display	
12	Color image Processing – noise removal by average	
13	Color image Processing – noise removal Median	
14	Color image Processing – Histogram Equalization	
15	Color image Processing – Edge Detection	
16	Frequency domain Enhancement	

Practical No 1 – Brightness Manipulation of an Image

Aim: The brightness of an image will be increased or decreased without visual distortion

Objective:

1. To use the basic Scilab commands for image processing. 2.
To learn Conditional and unconditional loops in Scilab
3. To process the images as per the applications.

Learning Outcome:

1. Students will be able to read and display images.
2. Students will be able to access pixels and change it according to the application.
3. Students will be able to use conditional and unconditional loops for processing images.

Software Used: Scilab with SIVP toolbox

Theory:

The degree of the illuminance of the image is called brightness. The light reflected from an object defines the brightness of that object. If an object is placed in a dark region i.e. very less illumination, the reflected light will be very little. Therefore we can't see the object properly. Brightness in context with digital image processing is defined as the intensity level of the pixels of the image.

The binary image has 0 intensity means dark pixel and 1 intensity means white pixel. For gray scale image, 0 stands for black pixel and 255 is bright pixel. Therefore, brightness can be said as the average intensity value of all the pixels of an image.

To increase the brightness, we have to increase the intensity of each pixel. Image is a collection of pixel values. So, images are additive in nature. We can just add a constant to every pixel value to increase brightness. To decrease the brightness, we can just decrease a constant value from all the pixels of the image.

Algorithm:

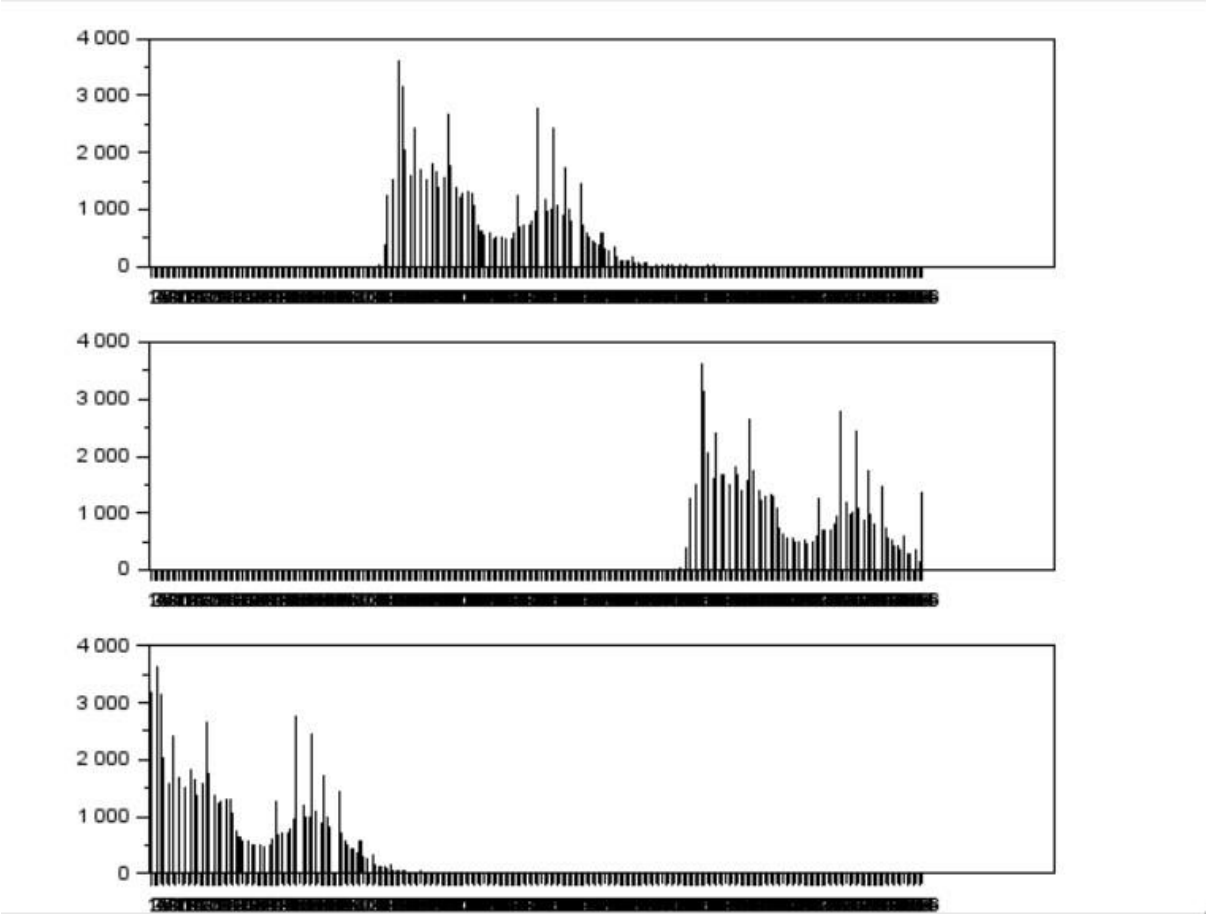
1. Read image.
2. Add a constant value in each pixel to increase brightness or subtract a constant value from each pixel to decrease the brightness of an image.
3. Display original, dark and bright images.

CODE:

```
clc; clear;
// read the image
a1=imread('pout.bmp');
a=double(a1);
b=double(a1);
b1=double(a1); [row
col]=size(a1);


// brightness modification
for x=1:row     for
y=1:col
temp=a(x,y)+100;
if temp>255
        temp1=255;
```

```
 else
        temp1=temp;
    end


 b(x,y)=temp1;  //brightness increase


    temp2=a(x,y)-80;
if temp2 <0
temp3=0;
    else
       temp3=temp2;
    end


    b1(x,y)=temp3; // brightness
decrease    end end


c=zeros(1,256);
c1=zeros(1,256);
c2=zeros(1,256); for
x=1:row    for
y=1:col
temp4=a(x,y)+1;
    c(temp4)=c(temp4)+1;


    temp5=b1(x,y)+1;
    c1(temp5)=c1(temp5)+1;


    temp6=b(x,y)+1;
c2(temp6)=c2(temp6)+1;
  end
end


t=[a1 uint8(b) uint8(b1)]


//display image
imshow(t);
subplot(3,1,1); bar(c,
0.01);
subplot(3,1,2);
bar(c2, 0.01);
subplot(3,1,3);
bar(c1, 0.01);
```

OUTPUT:



Conclusion:

Brightness of an image is increased or decreased by adding/subtracting a constant value in each pixel of an image.

Practical No 2 – Image Negative

Aim: Enhancement in spatial domain Part 1: Gray level transformation

Image Negative: Use of 'for' loop and commands like imread, rgb2gray, and imshow.

Objective:

4. To use the basic Scilab commands for image processing. 5.
To learn Conditional and unconditional loops in Scilab
6. To process the images as per the applications.

Learning Outcome:

4. Students will be able to read and display images.
5. Students will be able to access pixels and change it according to the application.
6. Students will be able to use conditional and unconditional loops for processing images.

Software Used: Scilab with SIVP toolbox

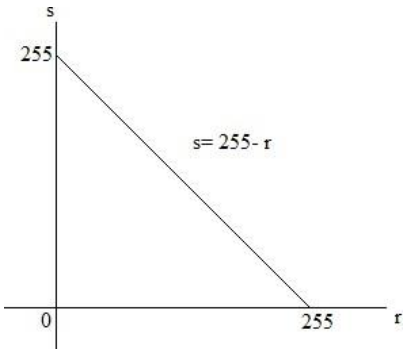
Theory:

Image Negative

A positive image is a normal image. Colour image uses RGB additive colour mixing. A negative image is an inversion of intensities in which light areas appear dark and viceversa. A negative colour image is colour-reversed with subtractive colour mixing. Film negatives usually have less contrast, but a wider dynamic range than the final printed positive images. The contrast typically increases when they are printed onto photographic paper. When negative film images are brought into the digital realm, their contrast may be adjusted at the time of scanning or more usually during subsequent post-processing.

Transfer Function: **Equation:**

s= (L-1) – r (1)



Where, s=pixels of transformed image L= total number of gray levels.

For 8 bit, $L=2^8$ =>256 r=pixels of original image **Algorithm:**

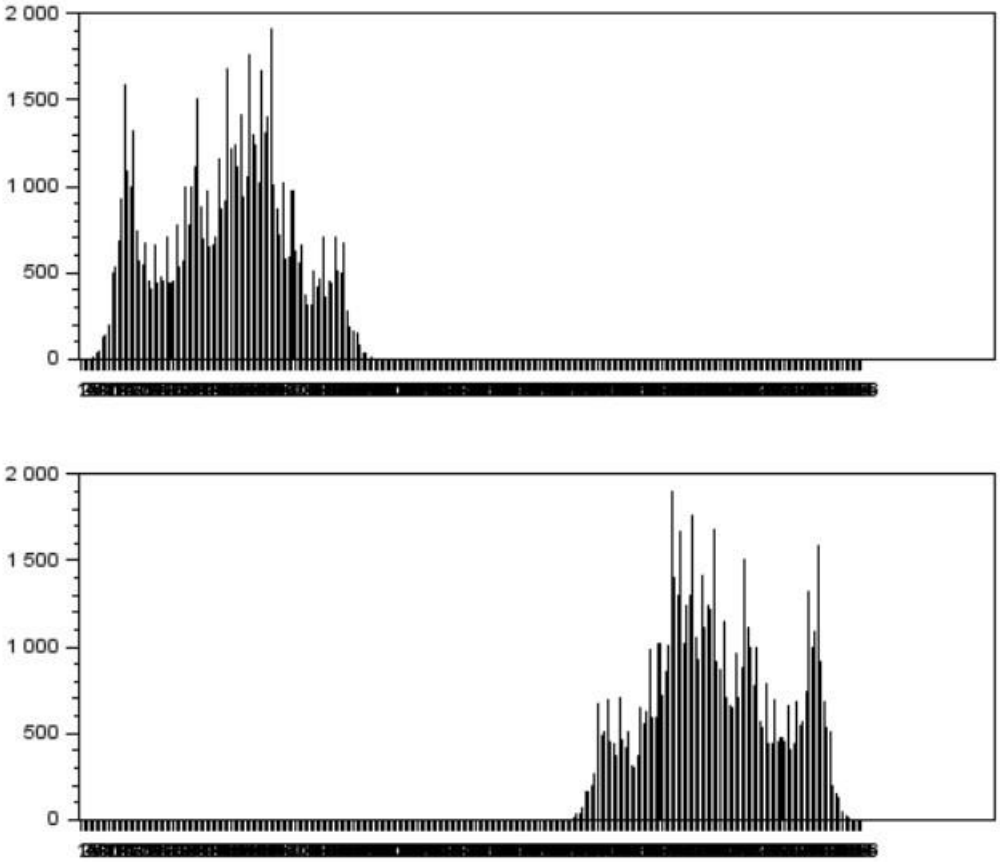
4. Read image.
5. Subtract each pixel from (L-1) .e.255
6. Display both the images.

CODE:

```
clc;
clear;
a1=imread('lena_dark.bmp');
a=rgb2gray(a1);
a=double(a); b=double(a);
//b=255-a b=a;
[row col]=size(a); for
x=1:row     for y=1:col
b(x,y)=255-a(x,y);
```

```
    end
end
d=[uint8(a) uint8(b)];
imshow(d);


c=zeros(1,256);
 c1=zeros(1,256); for
x=1:row     for
y=1:col
temp=a(x,y)+1;
       c(temp)=c(temp)+1;


       temp1=b(x,y)+1;
c1(temp1)=c1(temp1)+1;
   end end
subplot(2,1,1);
bar(c,0.01);
subplot(2,1,2);
bar(c1,0.01);
```

OUTPUT:



Conclusion:

Image negative presents image in complementary shades.

Practical No 3 – Thresholding for different values of Threshold

Aim: Enhancement in spatial domain Part 1: Gray level transformation Thresholding: Use of if-else loop

Objective:

1. To use the basic Scilab commands for image processing. 2.
To learn Conditional and unconditional loops in Scilab
3. To process the images as per the applications.

Learning Outcome:

1. Students will be able to read and display images.
2. Students will be able to access pixels and change it according to the application.
3. Students will be able to use conditional and unconditional loops for processing images.

Software Used: Scilab with SIVP toolbox

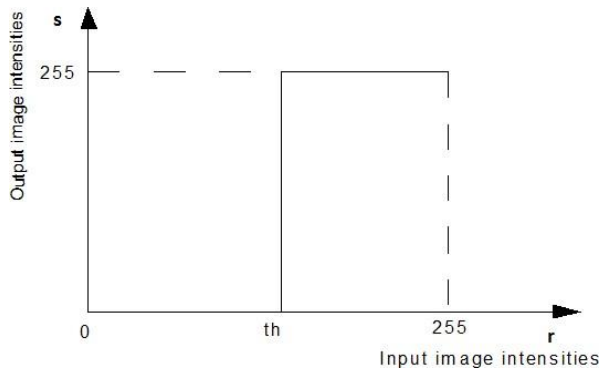
Theory:

Following are the image enhancement techniques to be implemented in this experiment:

A) Thresholding:

Thresholding is an important form of image segmentation and is a first step in the processing of images for many applications. It is a process of converting a gray scale input image to a bi-level image by using an optimal threshold. It replaces each pixel in an image with a black pixel if the image intensity is less than some fixed constant 'Th' or a white pixel if the image intensity is greater than that constant.

Transfer Function:



Equation:

$$s = \{1, \text{ if } r \geq th$$
$$0, otherwise$$

(2)

Where, s=pixels of transformed image
r=pixels of original image
th= Thresholding value

Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting gray scale images into binary images. Image thresholding is most effective in images with high levels of contrast.

Algorithm:

1) Read an image file.
2) Check if the image file is found in the folder where the code is expected to save.
3) Enter the threshold value.
4) If image pixel is less than threshold, new pixel will be 0 else new pixel will be 255.
5) Repeat step 3 and 4 for different threshold values.
6) Display thresholded images.

CODE:

```
clear;
clc;


//Read Image
a=imread('cat256.bmp');


[row col]=size(a);
//Negative Image
for x=1:row
for y=1:col
if a(x,y)<50
b(x,y)=0;
      else
b(x,y)=255;
      end
end
end


for x=1:row    for
y=1:col        if
a(x,y)<150
b1(x,y)=0;
else
b1(x,y)=255;
      end
end end


//b=255-a; //Display
t=[a uint8(b) uint8(b1)];
imshow(t);
```

OUTPUT:

Conclusion:

Object is separated from background using thresholding operation. As threshold value increases more back portion is visible in output image and vice versa.

Practical No 4 – Gray Level Slicing with & without background

Aim: Enhancement in spatial domain Part 1: Gray level transformation
Gray Level Slicing: Use of if-elseif-else loop and select-case loop.

Objective:

1. To use the basic Scilab commands for image processing. 2.
To learn Conditional and unconditional loops in Scilab
3. To process the images as per the applications.

Learning Outcome:

1. Students will be able to read and display images.
2. Students will be able to access pixels and change it according to the application.
3. Students will be able to use conditional and unconditional loops for processing images.

Software Used: Scilab with SIVP toolbox

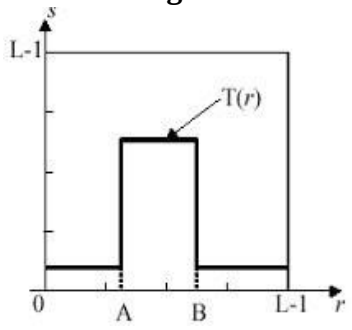
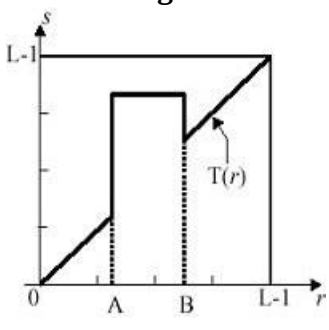
Theory:

Gray Level Slicing with and without background

Thresholding splits the grey level into 2 parts. At times, we need to highlight a specific range of grey values like for example enhancing the flaws in an X-ray or CT scan image. In such circumstances, a transformation known as grey level slicing is used. It looks similar to thresholding function except that here we select a band of grey level values.

There are two methods:

1. Gray level slicing without background: The background is completely lost.
2. Gray level slicing with background: In some applications, enhancing a band of grey levels and to retain the background is very much mandatory. Such technique of retaining the background is called as Grey-level slicing with background.

Gray Level Slicing without Background	**Gray Level Slicing with Background**	**Equation:** **Without background** s = {255 r1 ≤ r ≤ r2 0 Otherwise **With Background** s = {255 r1 ≤ r ≤ r2 r Otherwise
		

Algorithm:

1. Read image.
2. Take values of r1 and r2 from user.
3. Turn all pixels in between r1 and r2 equal to (L-1) i.e. 255
4. Rest of all pixels will be made 0 (without background) or kept as it is (with background).
5. Display both the images.

CODE:

```
clc; clear;
// read the image
a=imread('cat256.bmp');


[row col]=size(a);
//b=a;


r1=150; r2=180;
// Gray Level Slicing for x=1:row     for
y=1:col        if a(x,y)<r1        b(x,y)=0;
//without background        b1(x,y)=a(x,y);
//with background
       elseif a(x,y)>r2
b(x,y)=0;
b1(x,y)=a(x,y);
else        b(x,y)=255;
b1(x,y)=255;
       end
    end end
//b=255-
a;


//concatanation of input and output
t=[a uint8(b) uint8(b1)];


//display image
imshow(t);
```

OUTPUT:

Conclusion:

Gray level slicing helps in observing the location of pixels in an image for a range of gray values.

Practical No 5 – Contrast Stretching

Aim: Enhancement in Spatial Domain: Piecewise Linear

Transformation

Contrast Stretching

Objectives:

1. To improve the quality of image using contrast stretching.

Learning Outcome:

1. Students will be able to choose appropriate values of r1, r2, s1, and s2 for contrast stretching.

Software Used: Scilab with SIVP toolbox

Theory:

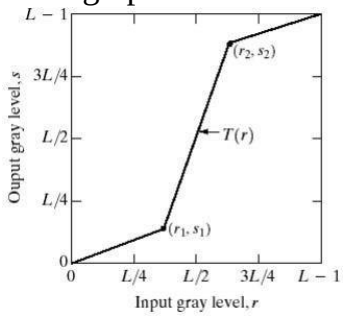
A) Contrast Stretching

The contrast of an image is a measure of its dynamic range, or the "spread" of its histogram. The dynamic range of an image is defined to be the entire range of intensity values contained within an image, or put a simpler way, the maximum pixel value minus the minimum pixel value. An 8-bit image at full contrast has a dynamic range of 28l=255, with anything less than that yielding a lower contrast image.

It differs from the more sophisticated histogram equalization in that it can only apply a linear scaling function to the image pixel values. As a result the `enhancement' is less harsh.

In image processing, contrast stretching is a process that changes the range of pixel intensity values. In more general fields of data processing, such as digital signal processing, it is referred to as dynamic range expansion.

Contrast Stretching is a method applies gamma correction (variable slopes) to different sections of the histogram as shown in the graph below



$$S = \begin{cases} \alpha * r; & r < r_1 \\ \beta * (r - r_1), & r_1 \leq r < r_2 \end{cases}$$

$\gamma * (r - r_2)$, $r > r_2$ Where,

S – Contrast stretched image pixels r1, r2 - Original image pixel values(threshold) α, β, γ are different slopes and α, γ < 1 , β >1

CODE:

```
clc; clear; // read the
image
a1=imread('pout.bmp');
a=double(a1);
b=double(a1); [row
col]=size(a1);
//b=a;


//pout r1=80;
r2=190;
s1=50;
s2=210;


// cat256
//r1=50;
//r2=180;
//s1=30;
//s2=200;


l=s1/r1; m=(s2-s1)/(r2-r1);
n=(255-s2)/(255-r2);


// Gray Level Slicing
 for x=1:row     for
y=1:col         if
a(x,y)<r1
       b(x,y)=l*a(x,y);          //r<r1
elseif a(x,y)>r2         b(x,y)=(n*(a(x,y)-
r2))+s2; // r<r2         else
       b(x,y)=(m*(a(x,y)-r1))+s1; // r<r2
 end     end end //b=255-a;


//concatanation of input and output
t=[a1 uint8(b)];


//display image
imshow(t);
```

OUTPUT:

Conclusion:

1. The quality of an image can be improved by using contrast stretching.

2. The proper selection of input and output levels in contrast stretching is intuitive and subjective. These values get changed if image under processing is changed.

Practical No 6 – Histogram Plotting

Aim: Enhancement in Spatial Domain: Piecewise Linear Transformation

A) Histogram

Objectives:

1. To improve the quality of image using histogram .
2. To understand the concept of histogram and its operation.

Learning Outcome:

1. Students will be able to plot histogram.
2. Students will use histogram algorithm to improve the quality of an image.

Software Used: Scilab with SIVP toolbox

Theory:

A) Histogram

A histogram is a graph. A graph that shows frequency of anything. Usually histogram have bars that represent frequency of occurring of data in the whole data set.

A Histogram has two axis the x axis and the y axis.

The x axis contains event whose frequency you have to count.

The y axis contains frequency.

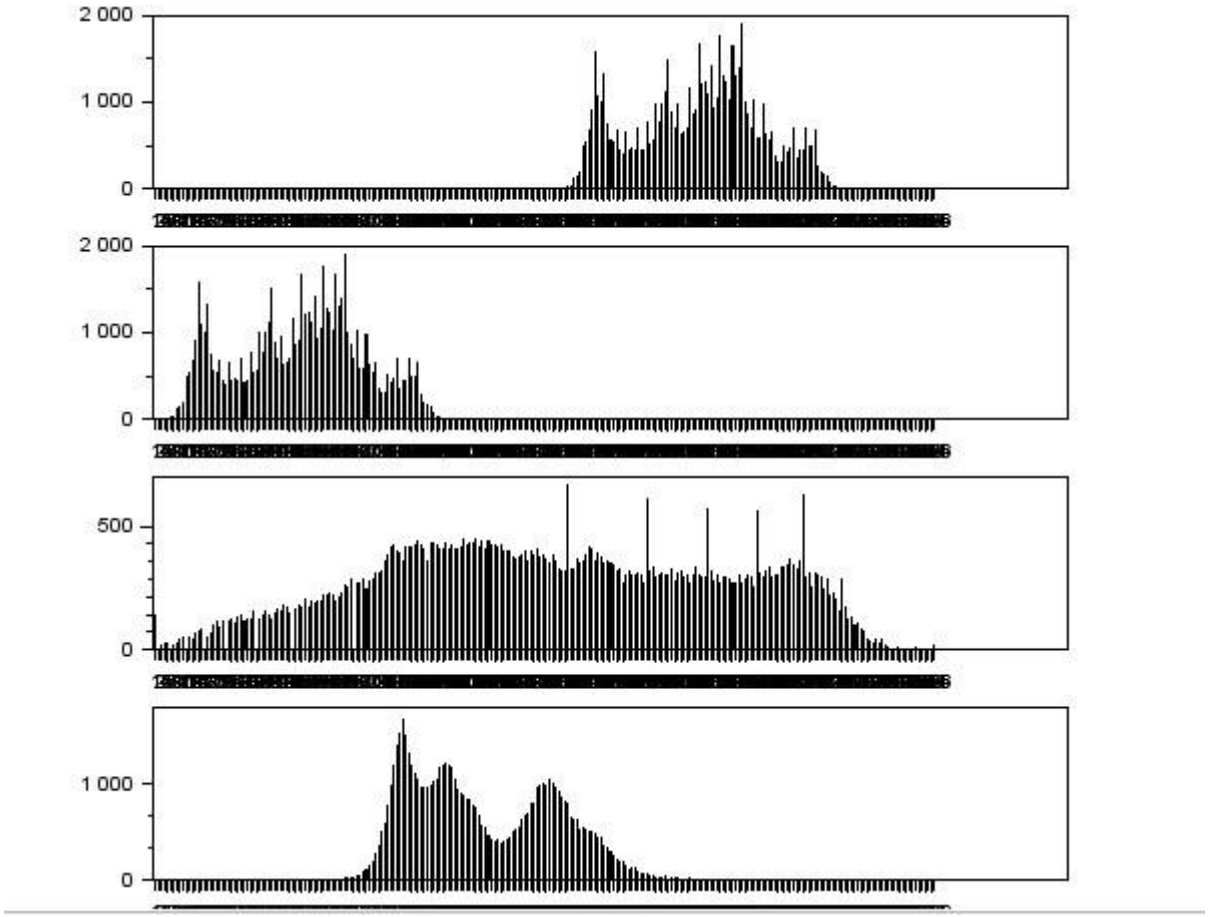
The different heights of bar shows different frequency of occurrence of data.

CODE:

```
clc;
clear;
a1=imread("lena.bmp");
b1=imread("lena_dark.bmp");
c1=imread("cat256.bmp");
d1=imread("out.bmp");
//c1=rgb2gray(c1)
d1=rgb2gray(d1)


b=double(b1);
a=b+130 c=double(c1);
d=double(d1);
[row col] = size(a);


count1 = zeros(1,256); count2
= zeros(1,256); count3 =
zeros(1,256); count4 =
zeros(1,256); for x=1:row
for y=1:col        temp1 =
a(x,y)+1;         temp2 =
b(x,y)+1;         temp3 =
c(x,y)+1;         temp4 =
d(x,y)+1;         count1(temp1)
= count1(temp1)+1;
count2(temp2) =
count2(temp2)+1;
count3(temp3) =
```

```
count3(temp3)+1;
count4(temp4) =
count4(temp4)+1;     end end


subplot(4,1,1)
bar(count1,0.01)
subplot(4,1,2)
bar(count2,0.01)
subplot(4,1,3)
bar(count3,0.01)
subplot(4,1,4)
bar(count4,0.01)


p= [uint8(a) uint8(b) ;uint8(c) uint8(d)]
imshow(p)
```

OUTPUT:



Conclusion:

1. The histogram of dark image and bright image is towards left of x-axis and right side of x-axis (dynamic range of pixels axis)
2. Low contrast image histogram is at the center of dynamic range of pixels and high contrast image histogram occupies entire dynamic range.

Practical No 7 – Histogram Equalization

Aim: Enhancement in Spatial Domain: Piecewise Linear Transformation

A) Histogram Equalization

Objectives:

1. To improve the quality of image using histogram equalization.
2. To understand the concept of histogram and its operation.

Learning Outcome:

1. Students will be able to plot histogram.
2. Students will use histogram algorithm to improve the quality of an image.

Software Used: Scilab with SIVP toolbox

Theory:

A) Histogram Equalization

It is a method of contrast adjustment using the image histogram. This method usually increases the local contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast without affecting the global contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. If the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

Implementation

Consider a discrete grayscale image, and let *ni* be the number of occurrences of gray level, *i*. The probability of an occurrence of a pixel of level *i* in the image is

$$p(x_i) = \frac{n_i}{n} \qquad i \in 0,1, \ldots \ldots \ldots, L-1$$

L being the total number of gray levels in the image, *n* being the total number of pixels in the image, and *p* being in fact the image's histogram, normalized to 0..1.

Let us also define *c* as the cumulative distribution function corresponding to *p*, defined by:

$$c(i) = \sum_{j=0}^{i} p(x_j)$$

c is the image's accumulated normalized histogram.

We would like to create a transformation of the form *y=T(x)* that will produce a level *y* for each level *x* in the original image, such that the cumulative probability function of *y* will be linearized across the value range. The transformation is defined by:

$$y_i = T(x_i) = c(i)$$

Notice that the T maps the levels into the domain of 0..1. In order to map the values back into their original domain, the following simple transformation needs to be applied on the result:

$$\acute{y}_i = y_i \,.\, (max - min) + min$$

The above describes histogram equalization on a greyscale image. However it can also be used on colour images by applying the same method separately to the Red, Green and Blue components of the RGB colour values of the image

General Working

The histogram equalization is operated on an image in three steps:

1) Histogram Formation

2) New Intensity Values calculation for each Intensity Levels 3) Replace the previous Intensity values with the new intensity values For the first step see the article on histogram.

In step 2, new intensity values are calculated for each intensity level by applying the following equation:

$$o_j = \sum^{j}_{k=0} N_k \times \frac{Imax}{M \times N}$$

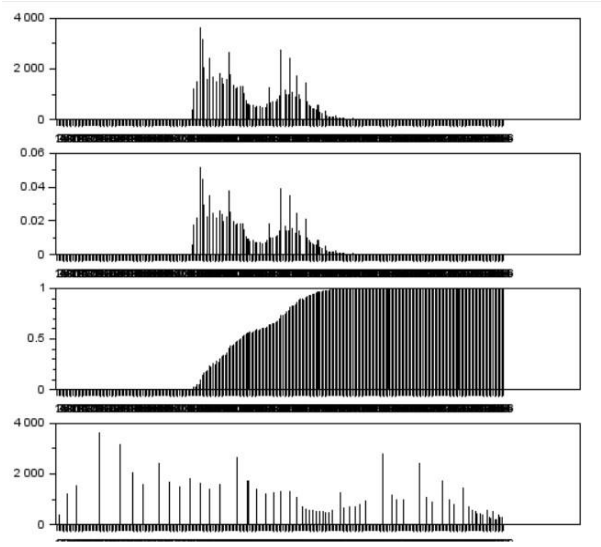
Where, *Imax* is maximum intensity level of a pixel. For example, if the image is in the grayscale, then the *Imax* is 255. M x N is height and width of an image. If the image is of size 256x256, then (MxN) is 65536. The expression is the bracket means that the no. of pixels having the intensity below the output intensity level or equal to it. For example, if we are calculating the output intensity level for 1 input intensity level, then the it means that the no. of pixels in the image having the intensity below or equal to 1 means 0 and 1. If we are calculating the output intensity level for 5 input intensity level, then the it means that the no. of pixels in the image having the intensity below or equal to 5 means 0 , 1 , 2 , 3 , 4 , 5. Thus, if we are calculating the output intensity level for 255 input intensity level, then the it means that the no. of pixels in the image having the intensity below or equal to 255 means 0
, 1 , 2 , 3 , ... , 255. That is how new intensity levels are calculated for the previous intensity levels.

The next step is to replace the previous intensity level with the new intensity level. This is accomplished by putting the value of *Oj* in the image for all the pixels, where *Oj* represents the new intensity value, whereas i represents the previous intensity level.

CODE:

```
clear; clc;

//Read Image a1=imread('pout.bmp');
a=double(a1); [row col]=size(a);
counter=zeros(1,256); //Negative
Image for x=1:row     for y=1:col
temp=a(x,y)+1;
counter(temp)=counter(temp)+1;
   end
end

subplot(4,1,1); bar(counter,
0.01);

N=row*col;

pdf=counter/N;

subplot(4,1,2);
bar(pdf, 0.01);

cdf(1)=pdf(1);

for i=2:256
cdf(i)=pdf(i)+cdf(i-1)
end

sk=cdf*255;  //L=2^8=256 ---> L-1=255

subplot(4,1,3); bar(cdf,
0.01);

for x=1:row     for
y=1:col
temp=a(x,y)+1;
temp1=sk(temp);
p(x,y)=temp1;
//end     end
end

counter1=zeros(1,256); for x=1:row
for y=1:col        temp=p(x,y)+1;
counter1(temp)=counter1(temp)+1;
   end
end

subplot(4,1,4); bar(counter1,
0.01);

t=[uint8(a) uint8(p)];
imshow(t);
```

OUTPUT:



Conclusion:

1. The quality of an image can be improved by using histogram equalization.
2. The histogram equalization is automatic process and gives better improvement as compared to contrast stretching.

Practical No 8 – Spatial Filtering – blurring by average and weighted average (do it for noisy images)

Aim: Spatial Filtering using

A) Average Filter

B) Weighted Average Filter

Objectives:

1. To blur the image using average filters.
2. To remove the noise (Gaussian, Salt & Pepper Noise) from given images and observe effectiveness of filters.

Learning Outcome:

1. Students will be able to choose appropriate filter to blur the image depending upon the application.
2. Students will be able to select proper filtering technique for removal of specific noise.

Software Used: Scilab with SIVP toolbox

Theory:

In image processing, to smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other finescale structures/rapid phenomena. In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal.

Mask Processing:

An image can be modified by applying a particular function to each pixel value. Neighbourhood processing be considered as an extension of this, where a function is applied to a neighbourhood of each pixel. The idea is to move a 'mask': a rectangle (usually with sides of odd length) or other shape over the given image. As we do this, we create a new image whose pixels have grey values calculated from the grey values under the mask, as shown in Figure 3.1. The combination of mask and function is called a filter. If the function by which the new grey value is calculated is a linear function of all the grey values in the mask, then the filter is called a linear filter

Mechanics of Spatial Filtering:

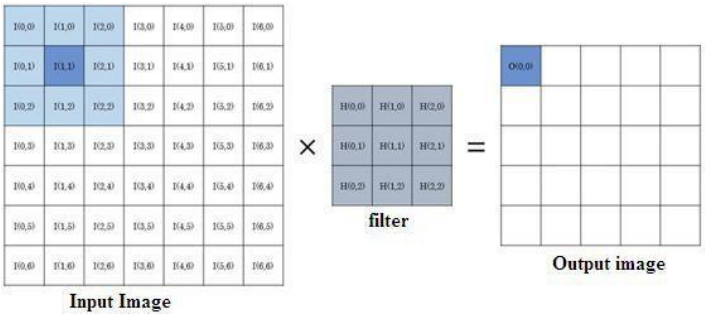


Fig. 3.1: Mechanics of spatial filtering

Three smoothing filters are used in this experiment:

A) Average Filter:

Average filtering is a method of reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images. The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbours, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighbourhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, although larger kernels (e.g. 5×5 squares) can be used for more severe smoothing.

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

CODE: Spatial Filtering (Average)

```
clear; clc;


//read image
a2=imread('cat256.bmp');
a1=imnoise(a2, 'salt & pepper');
a=double(a1); p=a1;
[row col]=size(a1);


mask=[1,1,1;1 1 1;1 1 1];  //Average
//filtering for
x=2:row-1
for y=2:col-1
temp=0;
for i=-1:1
        for j=-1:1
```

```
 temp=temp+(a(x+i,y+j)*mask(i+2,j+2));
temp1=round(temp/9);
p(x,y)=temp1;            end        end    end
end


t=[a1 uint8(p)]
imshow(t);
```

OUTPUT:

B) Weighted Average filter

In weighted average filter, we gave more weight to the centre pixel, due to which the contribution of centre becomes more than the rest of the values. Due to weighted average filtering, we can control the blurring of an image

$\frac{1}{12} \times$

0	2	0
2	4	2
0	2	0

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

CODE: weighted average

```
clear;
clc;


//read image
 a2=imread('cat256.bmp');
 a1=imnoise(a2, 'salt & pepper');
 a=double(a1); p=a1;
[row col]=size(a1);


mask=[1,2,1;2 4 2;1 2 1];  //Weighted Average
```

```
//filtering for
x=2:row-1     for
y=2:col-1
temp=0;         for
i=-1:1            for
j=-1:1
      temp=temp+(a(x+i,y+j)*mask(i+2,j+2));
temp1=round(temp/16);
      p(x,y)=temp1;
end
       end


  end
end


t=[a1 uint8(p)] imshow(t);
```

OUTPUT:

Conclusion:

1. Blurring is more in average filter/weighted average filters.
2. Average filters are better for Gaussian noise removal compared to median filter.

Practical No 9 – Spatial Filtering – blurring by median filter (do it for noisy images)

Aim: Spatial Filtering using
A) Median Filtering

Objectives:

3. To blur the image using average, weighted average and median filters.
4. To remove the noise (Gaussian, Salt & Pepper Noise) from given images and observe effectiveness of filters.

Learning Outcome:

3. Students will be able to choose appropriate filter to blur the image depending upon the application.
4. Students will be able to select proper filtering technique for removal of specific noise.

Software Used: Scilab with SIVP toolbox

Theory:

In image processing, to smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other finescale structures/rapid phenomena. In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal.

Mask Processing:

An image can be modified by applying a particular function to each pixel value. Neighbourhood processing be considered as an extension of this, where a function is applied to a neighbourhood of each pixel. The idea is to move a 'mask': a rectangle (usually with sides of odd length) or other shape over the given image. As we do this, we create a new image whose pixels have grey values calculated from the grey values under the mask, as shown in Figure 3.1. The combination of mask and function is called a filter. If the function by which the new grey value is calculated is a linear function of all the grey values in the mask, then the filter is called a linear filter.

Mechanics of Spatial Filtering:

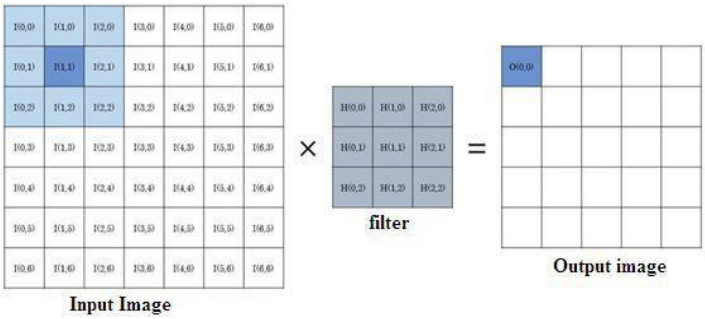


Fig. 3.1: Mechanics of spatial filtering

Three smoothing filters are used in this experiment:

A) Median Filter

Median filtering is order statistic based smoothing technique works similar to linear Gaussian filtering. Median filter considers neighbours pixels in block/window to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the *mean* of neighbouring pixel values, it replaces it with the *median* of those values. The median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the pixel being considered with the middle pixel value as shown in Fig 3.2. The smoothing techniques remove noise, but adversely affect the edges. Edges have critical importance to the visual appearance of images. Obviously, it is necessary to reduce the noise whilst preserve the edges.

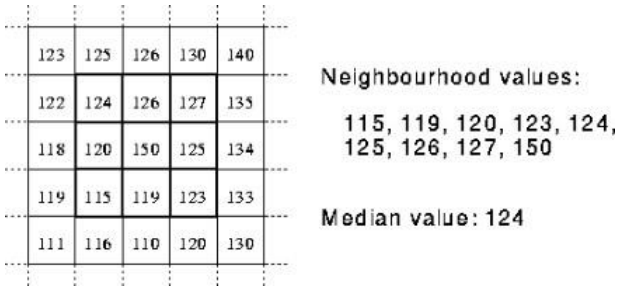


Fig. 3.2: Median Filtering process

For small to moderate levels of Gaussian noise, the median filter is demonstrably better than Gaussian blur at removing noise whilst preserving edges for a given fixed window size. However, its performance is much better for speckle noise and salt-andpepper noise (impulsive noise) compared to Gaussian blur for high levels of noise.

Algorithm:

1. Read image.
2. Take mask elements /window size for median filtering from user.
3. Apply 2D convolution.
4. Replace corresponding value of centre pixel.
5. Select normal, Gaussian noise and salt&pepper noise images.
6. Display output images with input image simultaneously.

CODE:

```
clear;
clc;


//read image
a2=imread('cat256.bmp');
a1=imnoise(a2, 'salt & pepper');
t1=imnoise(a2,'gaussian');
t2=double(t1); a=double(a1);
p1=double (a2); p2=double
(a2); p3=double (a2);
p4=double (a2);


[row col]=size(a1);


mask=[1,1,1;1 1 1;1 1 1];
//filtering for
 x=2:row-1
 for y=2:col-1
       temp=a(x-1,y-1)*mask(1,1)+a(x-1,y)*mask(1,2)+a(x-1,y+1)*mask(1,3)+ a(x,y-
1)*mask(2,1)+a(x,y)*mask(2,2)+a(x,y+1)*mask(2,3)+a(x+1,y-1)*mask(3,1)+a(x+1,y)*mask(3,2)+a(x+1,y+1)*mask(3,3);
p1(x,y)=round(temp/9);


       temp1=t2(x-1,y-1)*mask(1,1)+t2(x-1,y)*mask(1,2)+t2(x-1,y+1)*mask(1,3)+ t2(x,y-
1)*mask(2,1)+t2(x,y)*mask(2,2)+t2(x,y+1)*mask(2,3)+t2(x+1,y-1)*mask(3,1)+t2(x+1,y)*mask(3,2)+t2(x+1,y+1)*mask(3,3);
p3(x,y)=round(temp1/9);


       temp2=[a(x-1,y-1), a(x-1,y), a(x-1,y+1), a(x,y-1), a(x,y), a(x,y+1), a(x+1,y-1), a(x+1,y), a(x+1,y+1)];
temp3=gsort(temp2);
       p2(x,y)=temp3(5);


       temp4=[t2(x-1,y-1), t2(x-1,y), t2(x-1,y+1), t2(x,y-1), t2(x,y), t2(x,y+1), t2(x+1,y-1), t2(x+1,y), t2(x+1,y+1)];
temp3=gsort(temp4);
       p4(x,y)=temp4(5);


   end end


t=[a1 uint8(p1) uint8(p2); t1 uint8(p3) uint8(p4)]; imshow(t);
```

OUTPUT:

Conclusion:

1. Median filter is the most effective smoothing technique for salt & pepper noise than Gaussian noise.
2. Average filter is the best filter for removal of Gaussian noise.

Practical No 10 – Highpass filtering – Laplacian, Robert's cross gradient, Prewiit and Sobel

Aim: Edge detection using

A) Laplacian Operators
B) Prewitt Operators
C) Sobel Operators
D) Robert's Operators **Objectives:**

1. To apply various masks for line/edge detection in the image

Learning Outcome:

1. Students will be able to select appropriate mask to detect lines and edges in an image.

Software Used: Scilab with SIVP toolbox

Theory:

Image segmentation is an essential step in image analysis. Segmentation separates an image into its component parts or objects. The level to which the separation is carried depends on the problem being solved. Segmentation algorithms for images generally based on the discontinuity and similarity of image intensity values. Discontinuity approach is to partition an image based on abrupt changes in intensity and similarity is based on partitioning an image into regions that are similar according to a set of predefined criteria.

Point, Line and Edge detection:

Edge detection is a part of image segmentation. The effectiveness of many image processing also computer vision tasks depends on the perfection of detecting meaningful edges. It is one of the techniques for detecting intensity discontinuities in a digital image. The process of classifying and placing sharp discontinuities in an image is called the edge detection. The discontinuities are immediate changes in pixel concentration which distinguish boundaries of objects in a scene. Classical methods of edge detection engage convolving the image through an operator. Operators can be optimized to look for vertical, horizontal, or diagonal edges.

The sobel operator is very similar to Prewitt operator. Both are derivate masks and used for edge detection. They are used to detect two kinds of edges in an image namely Vertical direction and Horizontal direction.

Sobel Edge Detection

Roberts Cross Gradient

-1	0
0	+1

G_x

0	-1
+1	0

G_y

-1	-2	-1
0	0	0
+1	+2	+1

G_x

-1	0	1
-2	0	+2
-1	0	+1

G_y

Prewitt Edge Detection

-1	-1	-1
0	0	0
+1	+1	+1

G_x

-1	0	+1
-1	0	+1
-1	0	+1

G_y

Laplacian edge detection

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

This mask will prominent the horizontal edges in an image. It also works on the principle of above mask and calculates difference among the pixel intensities of a particular edge. As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge. Thus increasing the sudden change of intensities and making the edge more visible.

Laplacian Operator is a second order derivative operator which is used to detect edges in an image. The major difference between Laplacian and other operators like Prewitt & Sobel is that Laplacian detects inward or outward edges whilst other first order derivative masks detects edges in particular direction.

Algorithm:

1. Read image.
2. Select appropriate mask based on application.
3. Apply 2D convolution using mask on image.
4. Replace corresponding value of centre pixel. 5. Display output images for Gx, Gy and Edge detection.

CODE:

```
clc;
clear;
a1=imread("lena256color.bmp");
a=rgb2gray(a1)

[row col]= size(a);

//Laplacian
//m=[0,-1,0;-1,4,-1;0,-1,0]
//wm=[-1,-1,-1;-1,8,-1;-1,-1,-1]
//
////Sobel
//m=[-1,-2,-1;0,0,0;1,2,1]
//wm=[-1,0,1;-2,0,2;-1,0,1]
//
```

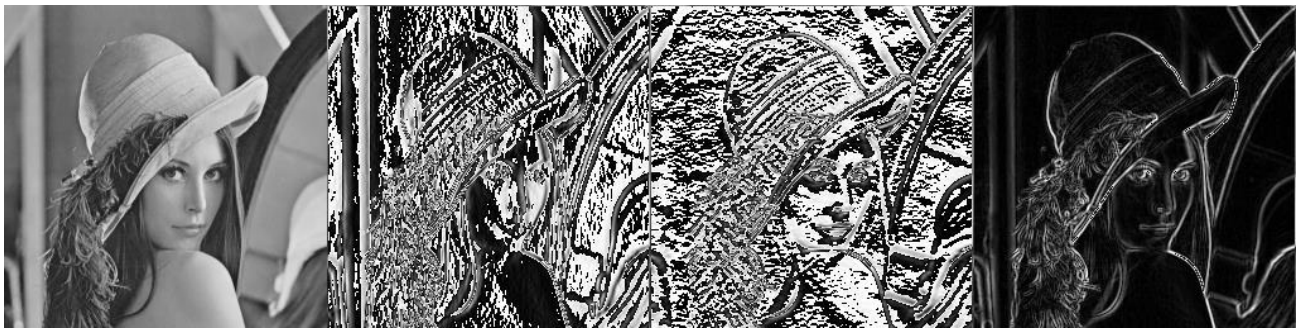
```
////prewit
//m=[-1,-1,-1;0,0,0;1,1,1]
//wm=[-1,0,1;-1,0,1;-1,0,1]
//
//Robert cross gradient
m=[0 0 0; 0 -1 0; 0 0 1]
wm=[0 0 0; 0 0 -1; 0 1 0]

a=double(a)
b=a
c=a
p=a
for x=2:row-1
   for y=2:col-1
      t1=a(x-1,y-1)*m(1,1)+a(x-1,y)*m(1,2)+a(x-1,y+1)*m(1,3)+a(x,y-
1)*m(2,1)+a(x,y)*m(2,2)+a(x,y+1)*m(2,3)+a(x+1,y-
1)*m(3,1)+a(x+1,y)*m(3,2)+a(x+1,y+1)*m(3,3);
      b(x,y)=t1
      t2=a(x-1,y-1)*wm(1,1)+a(x-1,y)*wm(1,2)+a(x-1,y+1)*wm(1,3)+a(x,y-
1)*wm(2,1)+a(x,y)*wm(2,2)+a(x,y+1)*wm(2,3)+a(x+1,y-
1)*wm(3,1)+a(x+1,y)*wm(3,2)+a(x+1,y+1)*wm(3,3);
      c(x,y)=t2
      t3=abs(t1)+abs(t2)

      p(x,y)=t3/2

   end
end

imshow([uint8(a) uint8(b) uint8(c) uint8(p)])
```

OUTPUT:

Laplacian

Sobel

Prewitt



Robert

Conclusion:

1. Direct edge detection is achieved using Laplacian mask.
2. Horizontal mask (Gx) results in horizontal edges become more visible and similarly all the vertical edges are more visible using vertical mask (Gy).
3. The Sobel masks gives out more edges as compared to Prewitt masks.

Practical No 11 – Color image Processing – color model conversion and plane display

Theory:

In image processing, to smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena. In smoothing, the data points of a signal are modified so individual points (presumably because of noise) are reduced, and points that are lower than the adjacent points are increased leading to a smoother signal.

Color model:

The color model aims to facilitate the specifications of colors in some standard way. Different types of color models are used in multiple fields like in hardware, in multiple applications of creating animation, etc.

1. In Digital Image Processing, the hardware-oriented models that are commonly used are the RGB model for printers and color monitors.
2. CMY(cyan, magenta, yellow) and CMYK(cyan, magenta, yellow, black) models are used for color printing.
3. HSI(hue, saturation, intensity) deals with colors as humans interpret.

A. RGB Model

In the RGB model, each color appears in its primary components of red, green and blue. This model is based on a Cartesian coordinate system as you can see in Figure 2.

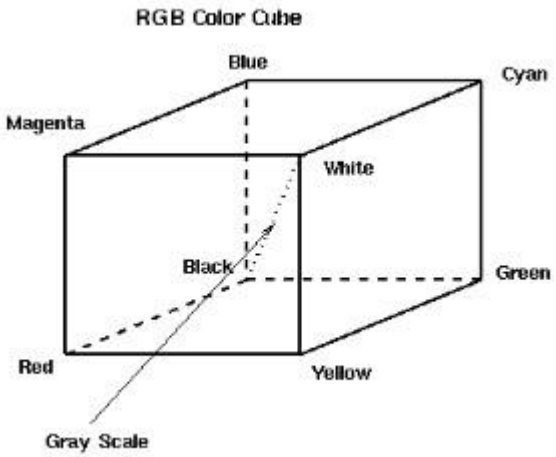


Figure 2: RGB color cube

B. YCbCr Color Model

The YCbCr color model is obtained by converting RGB to Linear Transformation. Y is nothing but the brightness or luminance. Cb and Cr are the chrominance or color components.

C. HSV

In color image processing, there are various models one of which is the hue, saturation, value (HSV) model. Using this model, an object with a certain color can be detected and to reduce the influence of light intensity from the outside. Tests were performed using six kinds of colors, ie brown, yellow, green, blue, black and white.

D. NTSC (YIQ)

YIQ is defined by the National Television System Committee (NTSC). Y describes the luminance, I and Q describes the chrominance. A more compact representation of the color. YUV plays similar role in PAL and SECAM.

CODE:

```
clc;
clear;

a=imread('lena.bmp');

R=a(:,:,1);
G=a(:,:,2);
B=a(:,:,3);

[row col]=size(R);

z=zeros(row, col);

//a(:,:,1)=z;
//a(:,:,2)=z;
//a(:,:,3)=z;

//imshow(a)

b=rgb2ycbcr(a); //imshow(b) c=rgb2hsv(a);
//imshow(c)
d=rgb2ntsc(a); //imshow(d)

Y1=d(:,:,1); //b-ycbcr   c-hsv
Y2=d(:,:,2); //b
Y3=d(:,:,3); //b

p=[Y1 Y2 Y3]; imshow(p)
```

OUTPUT:

ntsc

ycbcr

hsv

Practical No 12 & 13 – Color image Processing – noise removal by Average & Median

Theory:

A) Average Filter:

Average filtering is a method of reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images. The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbours, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighbourhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, although larger kernels (e.g. 5×5 squares) can be used for more severe smoothing.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

A) Median Filtering:

Median filtering is order statistic based smoothing technique works similar to linear Gaussian filtering. Median filter considers neighbours pixels in block/window to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the *mean* of neighbouring pixel values, it replaces it with the *median* of those values. The median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the pixel being considered with the middle pixel value as shown in Fig 3.2. The smoothing techniques remove noise, but adversely affect the edges. Edges have critical importance to the visual appearance of images. Obviously, it is necessary to reduce the noise whilst preserve the edges.

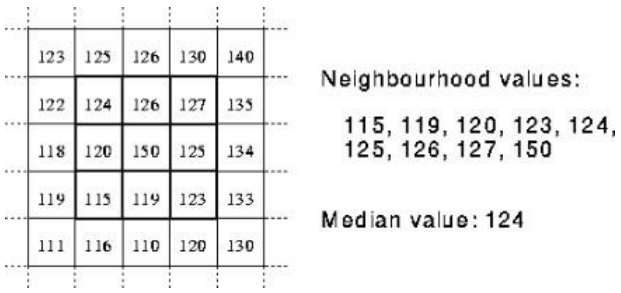


Fig. 3.2: Median Filtering process

For small to moderate levels of Gaussian noise, the median filter is demonstrably better than Gaussian blur at removing noise whilst preserving edges for a given fixed

window size. However, its performance is much better for speckle noise and salt-andpepper noise (impulsive noise) compared to Gaussian blur for high levels of noise.

CODE:

```
clc;
clear;

a2=imread('lena.bmp');

a1=rgb2ycbcr(a2); //imshow(b)
a3=a1; a4=a1;
//ntsc -- plane 1
//ycbcr - plane 2
//hsv - plane 3
p1=a1(:,:,1);
p2=a1(:,:,2);
p3=a1(:,:,3); //t=[p1
p2 p3];
//imshow(t);

b=imnoise(p2, 'salt &
pepper');
a1(:,:,2)=b; b1=ycbcr2rgb(a1);
imshow(b1);

[row col]=size(b);
a=double(round(b*255));

p=a; p1=a;

mask=[1 1 1;1 1 1;1 1 1];

for x=2:row-1
for y=2:col-1
      temp= a(x-1,y-1)*mask(1,1)+a(x-1,y)*mask(1,2)+a(x-1,y+1)*mask(1,3)+a(x,y-
1)*mask(2,1)+a(x,y)*mask(2,2)+a(x,y+1)*mask(2,3)+ a(x+1,y-1)*mask(3,1)+a(x+1,y)*mask(3,2)+a(x+1,y+1)*mask(3,3);
temp1=round(temp/9);
      p(x,y)=temp1/255;


      temp=[a(x-1,y-1), a(x-1,y),a(x-1,y+1),a(x,y-1),a(x,y),a(x,y+1), a(x+1,y-1),a(x+1,y),a(x+1,y+1)];
temp1=gsort(temp);
      p1(x,y)=temp1(5)/255;


    end
end

a3(:,:,2)=p; a4(:,:,2)=p1;
//out=ycbcr2rgb(a4);
out1=ycbcr2rgb(a3);

//out1=[b1 out]; //imshow(out);
//average
imshow(out1); //median
```

OUTPUT:

Average

Median

Practical No 14 – Color image Processing – Histogram Equalization

CODE:

```
clc;
clear;


a2=imread('lena.bmp'); //imshow(a2)
a1=rgb2ycbcr(a2); //imshow(b) a3=a1;
//ntsc -- plane 1
//ycbcr - plane 2
//hsv - plane 3


p1=a1(:,:,1); p2=a1(:,:,2);
p3=a1(:,:,3);


//b=imnoise(p2, 'salt & pepper');
//a1(:,:,2)=b;
//b1=ycbcr2rgb(a1);
//imshow(b1);


[row col]=size(p2);
a=double(round(p2*255));


p=a;


counter=zeros(1,256);
//Negative Image for
x=1:row     for
y=1:col
temp=a(x,y)+1;
       counter(temp)=counter(temp)+1;
end end


subplot(4,1,1);
bar(counter, 0.01);


N=row*col;


pdf=counter/N;


subplot(4,1,2);
bar(pdf, 0.01);


cdf(1)=pdf(1);


for i=2:256
    cdf(i)=pdf(i)+cdf(i-1)
end


sk=cdf*255;  //L=2^8=256 ---> L-1=255


subplot(4,1,3);
bar(cdf, 0.01);
```

```
for x=1:row     for
y=1:col
temp=a(x,y)+1;
temp1=sk(temp);
       p(x,y)=temp1;
      //end
   end
end


counter1=zeros(1,256);
for x=1:row     for
y=1:col
temp=p(x,y)+1;
       counter1(temp)=counter1(temp)+1;
end
end


subplot(4,1,4);
bar(counter1, 0.01);


a3(:,:,2)=p/255;


out=ycbcr2rgb(a3);


//out1=[b1 out];
imshow(out);
```

OUTPUT:

Practical No 15 – Color image Processing – Edge Detection

CODE:

```
clc;
clear;


a1=imread('lena.bmp');
a2=rgb2hsv(a1);


S=a2(:,:,1);
H=a2(:,:,2);
V=a2(:,:,3);
//t=[S H V];
//imshow(t)


[row col]=size(V); a=double(V*255);
p=a;


mask=[-1 -1 -1;-1 8 -1;-1 -1 -1];


for x=2:row-1
for y=2:col-1
      temp= a(x-1,y-1)*mask(1,1)+a(x-1,y)*mask(1,2)+a(x-1,y+1)*mask(1,3)+a(x,y-
1)*mask(2,1)+a(x,y)*mask(2,2)+a(x,y+1)*mask(2,3)+ a(x+1,y-1)*mask(3,1)+a(x+1,y)*mask(3,2)+a(x+1,y+1)*mask(3,3);
p(x,y)=temp1/255;
   end
end


a2(:,:,3)=p;
a3=hsv2rgb(a2)
; imshow(a3);
```

OUTPUT: