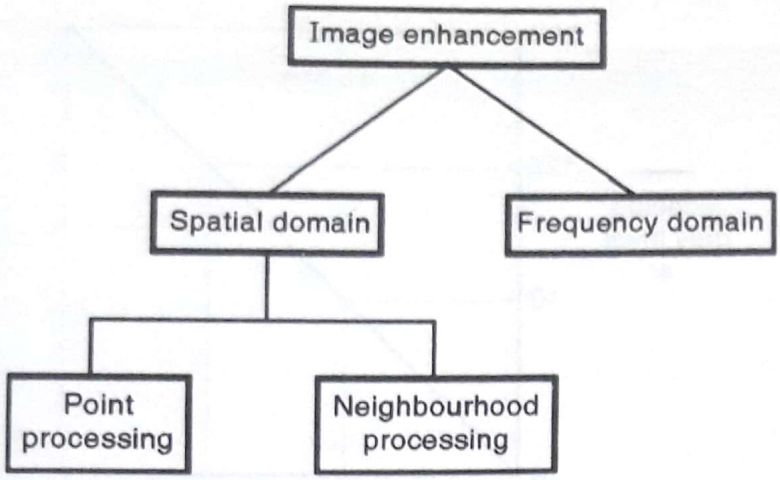


Fig. 4.2.2

4.3 Point Processing :

MU - May 2011, Dec. 2012, Nov. 2014, Dec. 2015

In point processing, we work with single pixels i.e. T is 1×1 operator. It means that the new value f(x, y) depends on the operator T and the present f(x, y). This statement will be clear as we start giving some examples.

Some of the common examples of point processing are

(1) Digital negative

(2) Contrast stretching

(3) Thresholding

(4) Grey level slicing

(5) Bit plane slicing

(6) Dynamic range compression

(7) Power law transformation

Before we proceed to explain the following examples, it is important to understand the identity transformation. The identity transformation is given in the Fig. 4.3.1.

In the Fig. 4.3.1, the solid line is the transformation T. The horizontal axis represents the grey scale of the input image (r) and the vertical axis represents the grey scale of the output image (s). It is called an identity transformation because it does not modify the new image at all !!. As seen, the grey level 10 is modified to 10, 125 to 125 and finally 255 to 255. In general $s_1 = r_1$. Fig. 4.3.1 will help us understand the point processing techniques better.

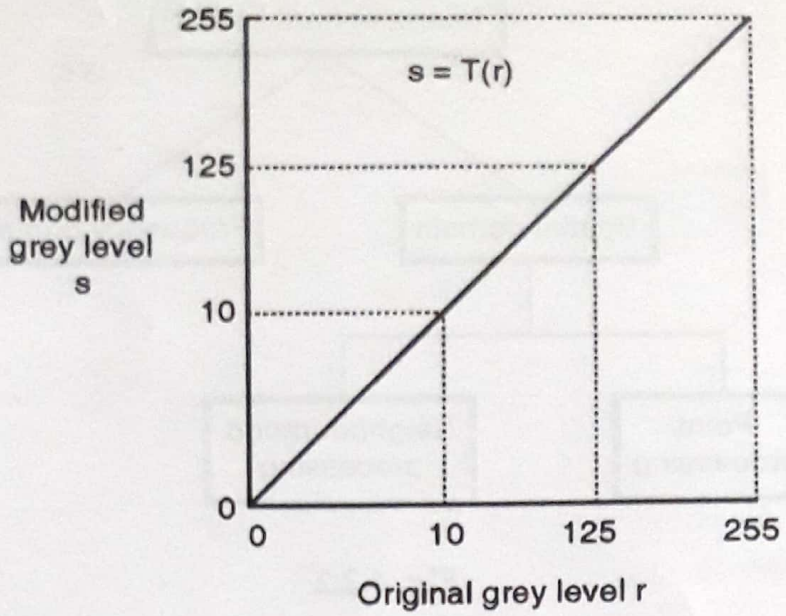


Fig. 4.3.1

(1) **Digital negative :** Digital negatives are useful in a lot of applications. A common example of digital negative is the displaying of an X-ray image. As the name suggests, negative simply means inverting the grey levels i.e. black in the original image will now look white and vice versa. The Fig. 4.3.2 is the digital negative transformation for a 8-bit image.

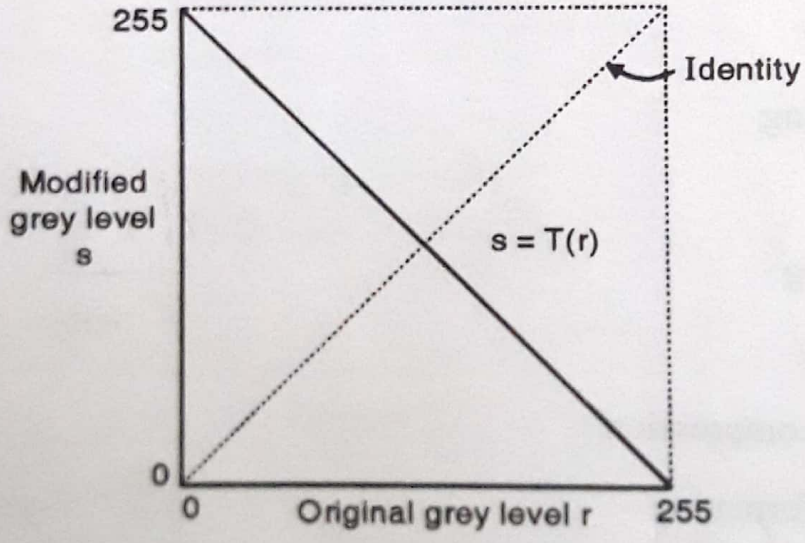


Fig. 4.3.2

The digital negative image can be obtained by using a simple transformation given by

$$s = 255 - r \; (r_{max} = 255)$$

Hence when

$$r = 0, s = 255 \text{ and when } r = 255, s = 0.$$

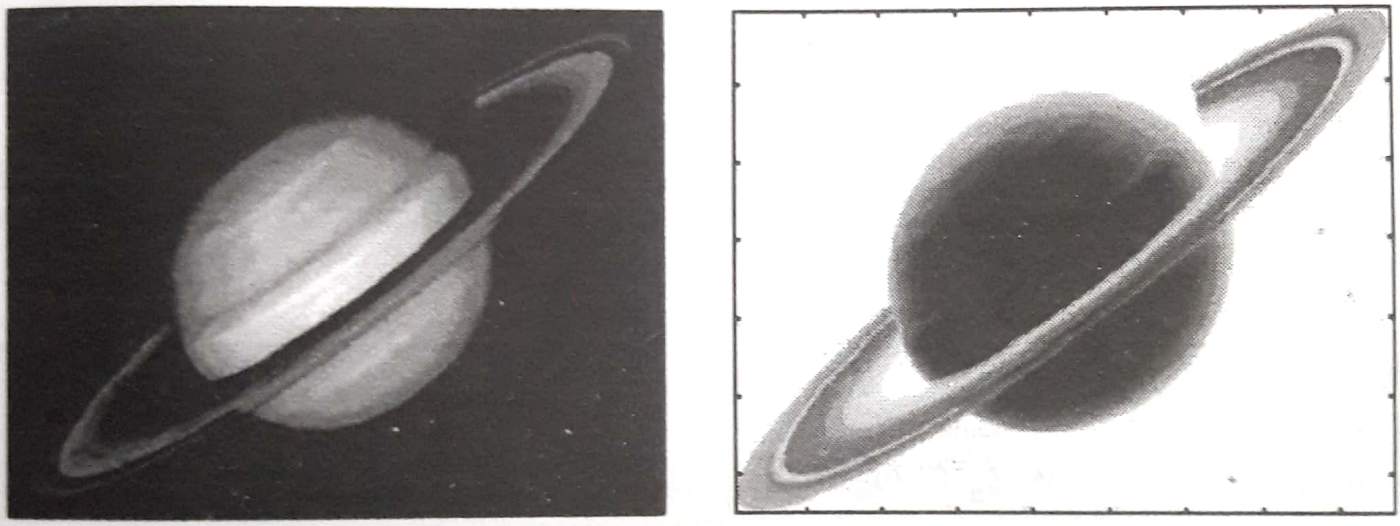
In general,

$$s = (L - 1) - r \qquad \ldots(4.3.1)$$

Here L is the number of grey levels. (256 in this case)

MATLAB program for finding the digital negative

```
%% MATLAB code to calculate negative %%
clear all
clc
aa=imread ('saturn.tif');
a=double (aa)
c=255; % for a 8-bit image %
b=c-a;
figure(1)
colormap(gray)
imagesc(a)
figure(2)
colormap(gray)
imagesc(b)
```



(a) Original image **(b) Digital negative**

Fig. 4.3.3

(2) Contrast stretching : Many times we obtain low contrast images due to poor illuminations or due to wrong setting of the lens aperture. The idea behind contrast stretching is to increase the contrast of the images by making the dark portions darker and the bright portions brighter.

Fig. 4.3.4 shows the transformation used to achieve contrast stretching. In the Fig. 4.3.4, the dotted line indicates the identity transformation and the solid line is the contrast stretching transformation. As is evident from the Fig. 4.3.4, we make the dark grey levels darker by assigning a slope of less than one and make the bright grey levels brighter by assigning a slope greater than one. One can assign different slopes depending on the input image and the application. As was mentioned, image enhancement is a subjective technique and hence there is no one set of slope values that would yield the desired result.

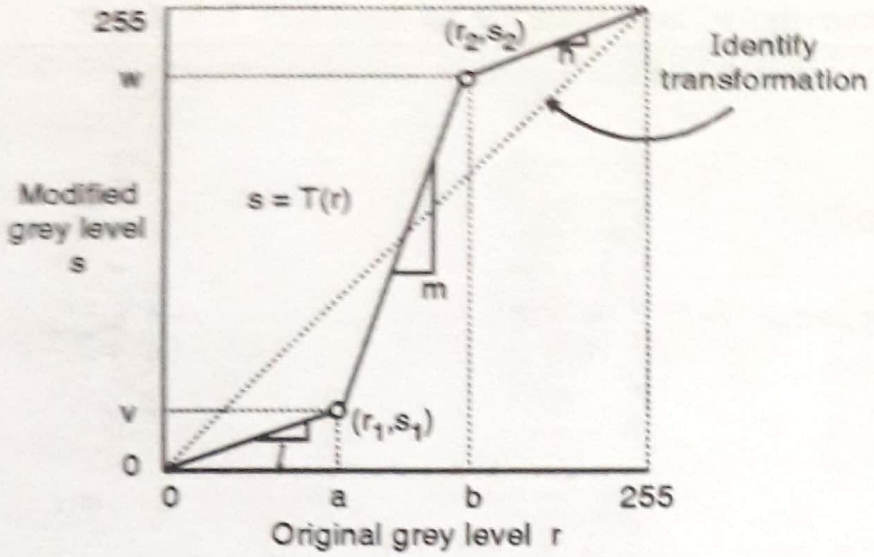


Fig. 4.3.4 : Original grey level r

The formulation of the contrast-stretching algorithm is given below.

$$s = \begin{cases} l \cdot r & 0 \le r < a \\ m \cdot (r-a) + v & a \le r < b \\ n \cdot (r-b) + w & b \le r < L-1 \end{cases} \quad \ldots(4.3.2)$$

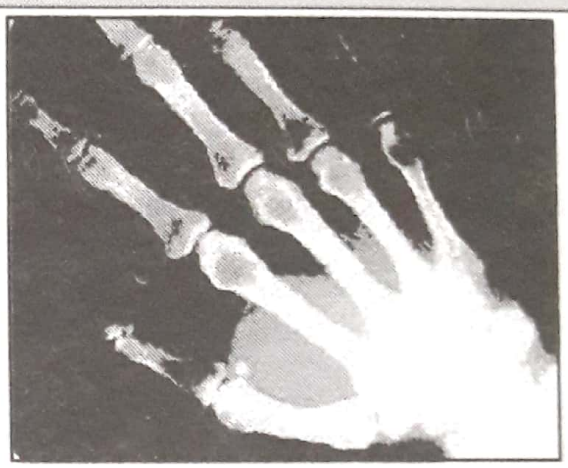
Where *l*, m and n are the slopes. It is clear from the figure that *l* and n are less than one while m is greater than one. The contrast stretching transformation increases the dynamic range of the modified image.

MATLAB program for contrast stretching

```
%% Contrast stretching of an image %%
% Slopes taken are 0.5, 2 and 0.5 %
clear all;
clc;
a=imread('xray1.tif')
a=double(a);
[row col]=size(a);
LT=input('Enter the lower threshold value:');
UT = input('Enter the upper threshold value:');
for x=1:1:row
    for y=1:1:col
     if a(x,y)<=LT
       b(x,y)=0.5*a(x,y);
     else if a(x,y)<=UT
            b(x,y)=2*(a(x,y)-LT)+0.5*LT;
     else b(x,y)=0.5*(a(x,y)-UT)+0.5*LT+2*(UT-LT);
      end
```

```
        end
    end
end
subplot(2,1,1)
imshow(uint8(a))
title('Original Image');
subplot(2,1,2)
imshow(uint8(b))
title('Image after Contrast Stretching')
```

(a) Original image



(b) Contrast stretched image

Fig. 4.3.5

(3) Thresholding : Extreme contrast stretching yields thresholding. If we observe the contrast stretching diagram closely we notice that if the first and the last slope are made zero, and the centre slope is increased, we would get a thresholding transformation i.e. if $r_1 = r_2$, $s_1 = 0$ and $s_2 = L - 1$, we get a thresholding function. It is shown in Fig. 4.3.6.

The formula for achieving thresholding is as follows.

$$\left.\begin{array}{ll} s = 0; & \text{if } r \le a \\ s = L - 1; & \text{if } r > a \end{array}\right\} \quad \ldots(4.3.3)$$

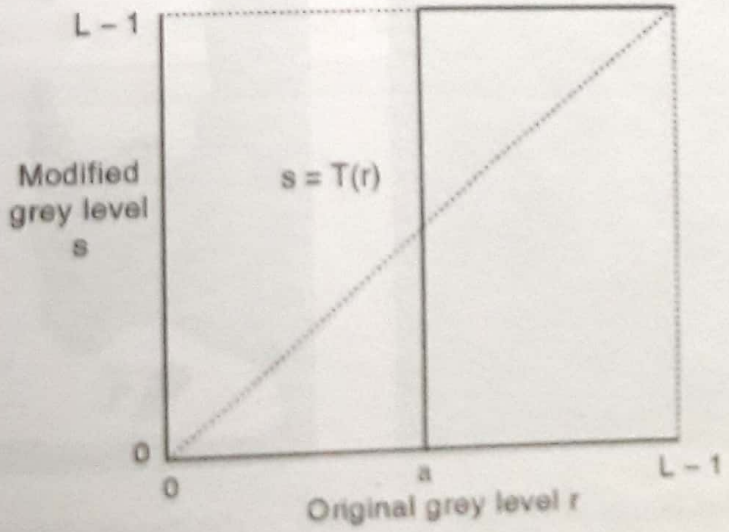


Fig. 4.3.6 : Original grey level r

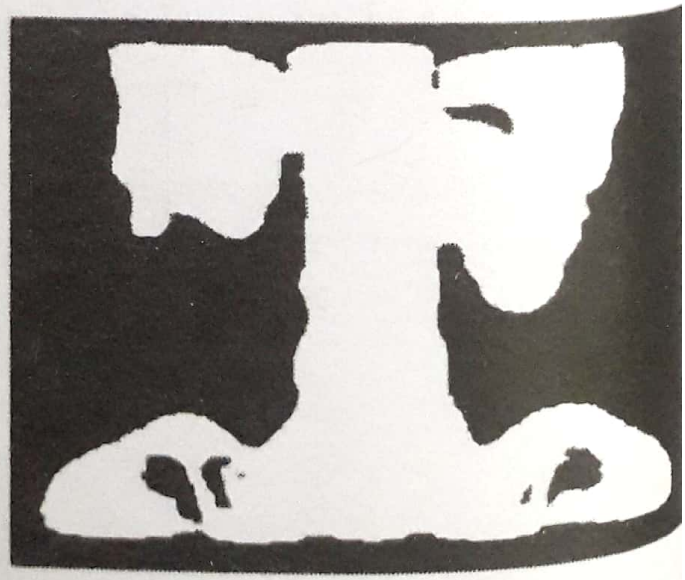
Where L is the number of grey levels.

As mentioned earlier, image enhancement being a subjective phenomena, the value of T will vary from image to image and from person to person. The objective is to identify the region that he or she is interested in. An important thing to note is that a thresholded image has the maximum contrast as it has only black and white grey values.

MATLAB program for thresholding

```
%%% Thresholding %%%
clear all
clc
p=imread('spine.tif');
a=p;
[row col]=size(a);
T=input('Enter value of Threshold :')
for i=1:1:row
        for j=1:1:col
            if(p(i,j)<T)
                a(i,j)=0;
            else
                a(i,j)=255;
        end
    end
end
figure(1),imshow(p),
figure(2),imshow(a)
```

(a) Original image of spine



(b) Image obtained using threshold

Fig. 4.3.7

(5) **Bit plane slicing :** In this technique, we find out the contribution made by each bit to the final image. As mentioned earlier, an image is defined as say a $256 \times 256 \times 8$ image. In this, 256×256 is the number of pixels present in the image and 8 is the number of bits required to represent each pixel. 8-bits simply means 28 or 256 grey levels.

Now each pixel will be represented by 8-bits. For example black is represented as 00000000 and white is represented as 11111111 and between them, 254 grey levels are accommodated. In bit plane slicing, we see the importance of each bit in the final image. This can be done as follows. Consider the LSB value of each pixel and plot the image using only the LSBs. Continue doing this for each bit till we come to the MSB. Note that we will get 8 different images and all the 8 images will be binary.

Ex. 4.3.1 : Given a 3×3 image, plot its bit planes.

1	2	0
4	3	2
7	5	2

3X3X3

Soln. :

Since 7 is the maximum grey level, we need only 3-bits to represent the grey levels. Hence we will have 3-bit planes. Converting the image to binary we get,

001	010	000
100	011	010
111	101	010

Binary image

1	0	0
0	1	0
1	1	0

LSB plane

0	1	0
0	1	1
1	0	1

Middle bit plane

0	0	0
1	0	0
1	1	0

MSB plane