CHAPTER 8

Histogram Modelling

Syllabus

Histogram Processing, Histogram equalization

8.1 Introduction

- Histogram of images provide a global description of the appearance of an image. The information obtained from histograms is enormous and hence histogram modelling; though a spatial domain technique is introduced as a separate chapter.
- By definition, histogram of an image represents the relative frequency of occurrence of the various grey levels in an image. Histogram of an image can be plotted in two ways.
- In the first method, the x-axis has the grey levels and the y-axis has the number of pixels in each grey level, while in the second method, the x-axis represents the grey levels, while the y-axis represents the probability of the occurrence of that grey level.

Method 1

Grey level	Number of pixels (n_k)
0	40
1	20
2	10
3	15
4	10
5	3
6	2

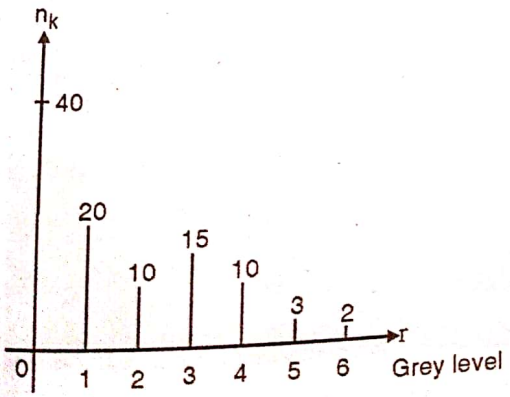


Fig. 8.1.1

Method 2

In the second method, instead of plotting the number of pixels directly, we plot their probability of occurrence i.e.,

$$p(r_k) = \frac{n_k}{n}$$

Where, $r_k \rightarrow k^{th}$ grey level

$n_k \rightarrow$ Number of pixels in the k^{th} grey level

$n \rightarrow$ Total number of pixels in an image

Grey level	Number of pixels (n_k)	$p(r_k) = \frac{n_k}{n}$
0	40	0.4
1	20	0.2
2	10	0.1
3	15	0.15
4	10	0.1
5	3	0.03
6	2	0.02
	n = 100	

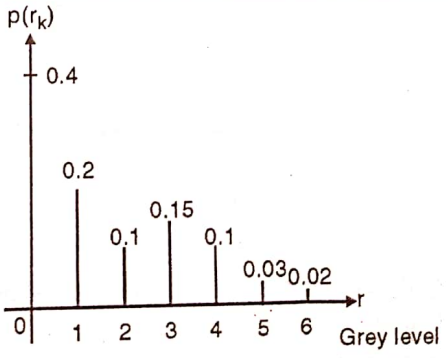


Fig. 8.1.2

- This is known as a normalized histogram. The advantage of the second method is that the maximum value to be plotted will always be 1. Generally black is considered as grey level 0 and white as the maximum.

- Just by looking at the histogram of the image, a great deal of information can be obtained. Some of the typical histograms are shown in Fig. 8.1.3.

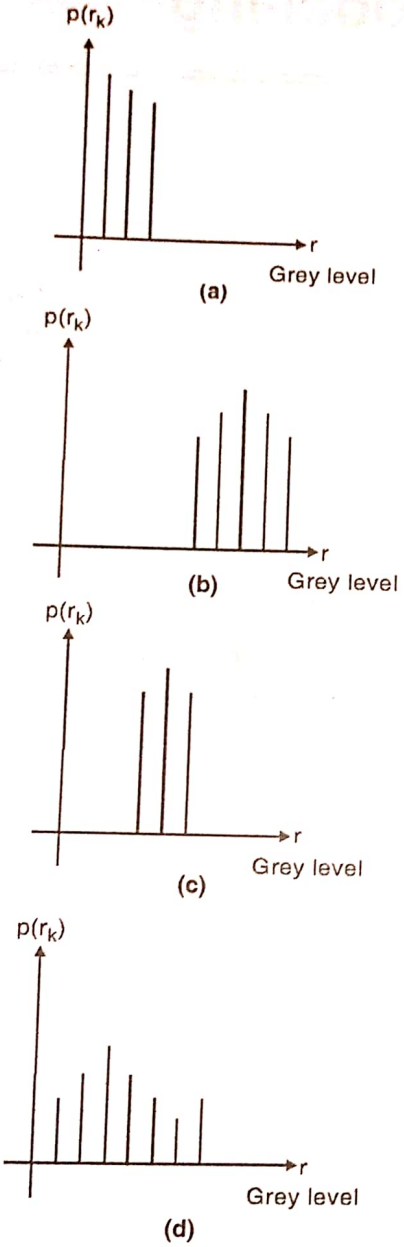


Fig. 8.1.3

- In Fig. 8.1.3(a) all the pixels belong to the lower grey levels 0, 1, ... and hence we can be sure that the image, represented by this histogram, is a dark image. Similarly, Fig. 8.1.3(b) is the histogram of a bright image. Fig. 8.1.3(c) is a low contrast image since only a small range of grey levels are present. Fig. 8.1.3(d) is a high contrast image of all the 4 histograms shown, the last histogram represents the best image.

- Our aim in this chapter would be to transform the first three histograms into the fourth histogram. In other words, we try to increase the dynamic range of the image. Though MATLAB has a inbuilt function to plot the histogram (hist) we shall write our own code.

MATLAB code for plotting the histogram of an image.

```
%% Program for histogram %%
clear all
clc
a=imread('test.tif');
a=double(a);
[row col]=size(a);
h=zeros(1,300);
for n=1:1:row
for m=1:1:col
if a(n,m)==0
%% To ensure that the values of a are not zero
            a(n,m)=1;
        end
    end
end
for n=1:1:row
    for m=1:1:col
        t=a(n,m);
%% Takes the value of the pixel ex. 12
        h(t)=h(t)+1;
%% Incrementing each counter h(12)
    end
end
%% PLOTTING %%
figure (1)
imshow(uint8(a))
figure(2)
bar(h)
```

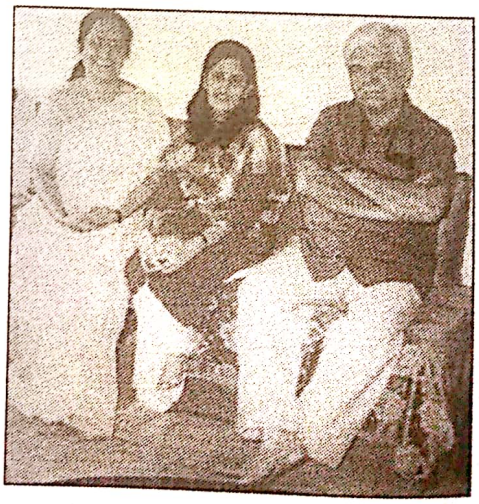


Fig. 8.1.4(a) : Original image

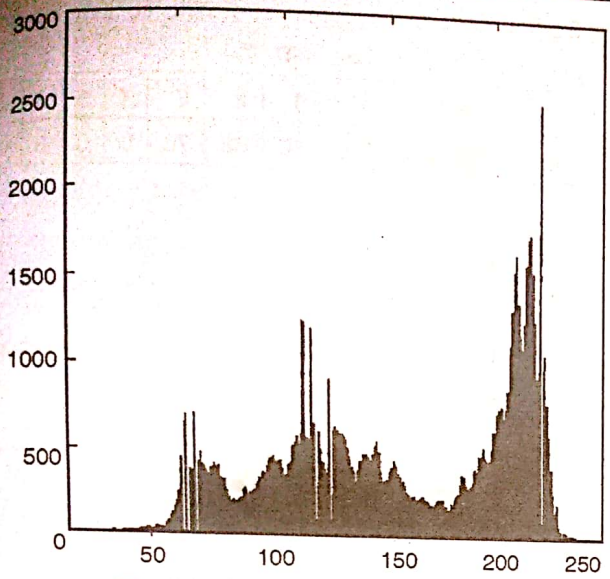


Fig. 8.1.4(b) : Histogram of image

8.1.1 Mean and Standard Deviation of Histogram

Histogram of an image provides a global description of the appearance of an image. By definition, histogram of an image represents the relative frequency of occurrence of the various grey levels in an image.

1. **Mean (μ)** : The Mean is a measure of central tendency. Mean is calculated by multiplying and adding the columns of the frequency table and diving it by the total number of elements present in the image. The mean value best reflects the typical score of a data set. Mean is also known as the Expected value.

   Consider the given frequency table.

Grey level	1	2	3	4
Number of pixels	5	12	10	5

The mean in this case will be = $((1 \times 5) + (2 \times 12) + (3 \times 10) + (4 \times 5))/32 = 2.46$

2. **Standard deviation (σ)** : In image processing, standard deviation is a measure that is used to measure the amount of variation of grey levels from the mean. A standard deviation close to 0 indicates that the data points tend to be very close to the mean of the image, while a high standard deviation indicates that the data points are spread out over a wider range of grey level values.

8.2 Linear Stretching

- One way to increase the dynamic range is by using a technique known as *histogram stretching*.

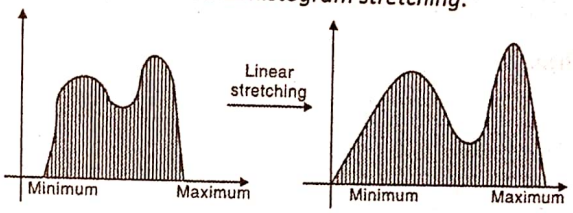


Fig. 8.2.1

- In this method, we do not alter the basic shape of the histogram, but we spread it so as to cover the entire dynamic range. We do this by using a straight line equation having slope $(s_{max} - s_{min}) / (r_{max} - r_{min})$.

Where, s_{max} → Maximum grey level of output image
s_{min} → Minimum grey level of output image
r_{max} → Maximum grey level of input image
r_{min} → Minimum grey level of input image

$$s = T(r) = \frac{s_{max} - s_{min}}{r_{max} - r_{min}} (r - r_{min}) + s_{min} \quad ...(8.2.1)$$

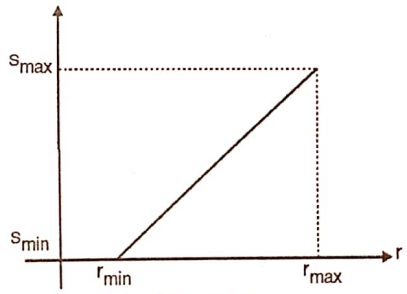


Fig. 8.2.2

- This transformation function shifts and stretches the grey level range of the input image to occupy the entire dynamic range (s_{min}, s_{max}).

8.2.1 Solved Examples on Linear Stretching

Ex. 8.2.1 : Perform histogram stretching so that the new image has a dynamic range of [0, 7].

Grey level	0	1	2	3	4	5	6	7
Number of pixels	0	0	50	60	50	20	10	0

Soln. :

Here, $r_{min} = 2$ $r_{max} = 6$

$s_{min} = 0$ $s_{max} = 7$

$$\therefore \quad s = \frac{7}{4}(r-2) + 0$$

When, $r = 2$, $s = 0$

$r = 3$, $s = 1.75 \approx 2$

$r = 4$, $s = 3.5 \approx 4$

$r = 5$, $s = 5.2 \approx 5$

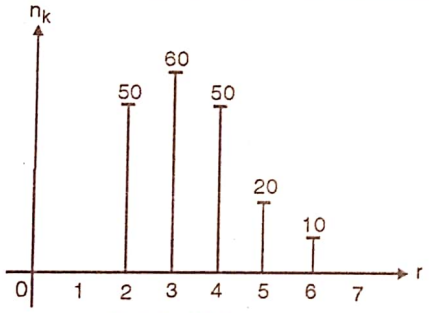
$r = 6$, $s = 7$

∴ We have,

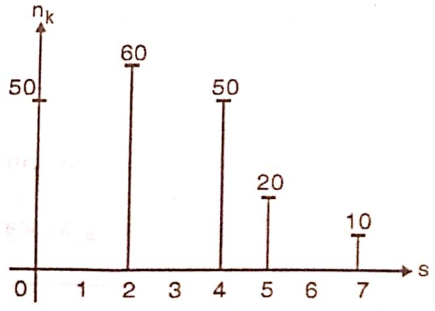
r = 2	s = 0
r = 3	s = 2
r = 4	s = 4
r = 5	s = 5
r = 6	s = 7

∴ Modified histogram is,

Grey level	0	1	2	3	4	5	6	7
Number of pixels	50	0	60	0	50	20	0	10



Original histogram



Histogram after linear stretching

Fig. P. 8.2.1

We see that the shape of the histogram is retained but it is spread out or stretched.

Let us take another example of histogram linear stretching.

Ex. 8.2.2 : Perform histogram stretching so that the new image has a dynamic range of [0, 7].

Grey level	0	1	2	3	4	5	6	7
Number of pixels	100	90	85	70	0	0	0	0

Soln. :

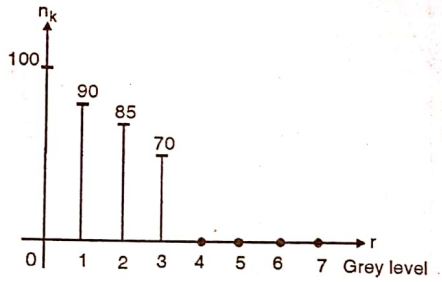


Fig. P. 8.2.2

This histogram of the image is shown. As can be concluded, it is a dark image.

Here, $r_{min} = 0$, $r_{max} = 3$,

$s_{min} = 0$, $s_{max} = 7$

$$s = \frac{s_{max} - s_{min}}{r_{max} - r_{min}} (r - r_{min}) + s_{min}$$

$$\therefore \quad s = \frac{7}{3}(r-0) + 0$$

When, $r = 0, s = 0$

$$r = 1, s = \frac{7}{3}(1-0) = 2.3 \approx 2$$

$$r = 2, s = \frac{7}{3}(2-0) = 4.67 \approx 5$$

$$r = 3, s = \frac{7}{3}(3-0) = 7$$

∴ We have,

r = 0	s = 0
r = 1	s = 2
r = 2	s = 5
r = 3	s = 7

Hence the histogram tends to spread out as shown in Fig. P. 8.2.2 (a).

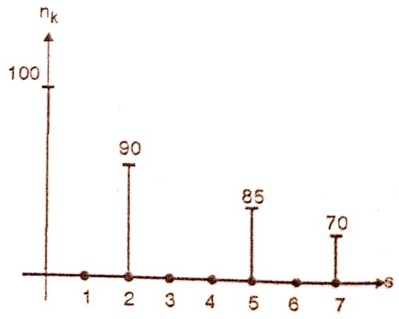


Fig. P. 8.2.2(a)

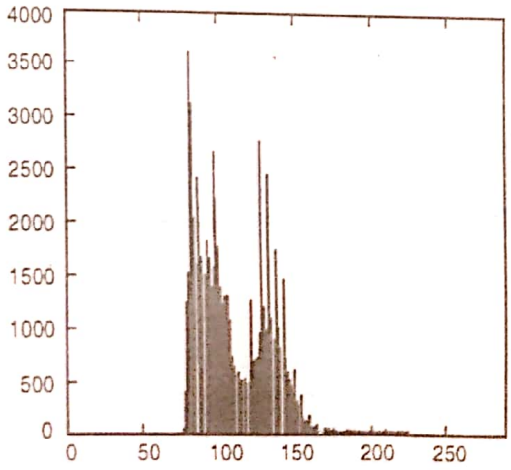
MATLAB code for histogram stretching %%

```
%% Histogram stretching %%
clear all
clc
a=imread('pout.tif');
a=double(a);
[row col]=size(a);
a=a+1;
w=min(min(a));
constant=255/(max(max(a))-min(min(a)));
cmin=0;
for x1=1:1:row
    for y1=1:1:col
        c(x1,y1)=constant*(a(x1,y1)-w)+cmin;
  end
 end

%% Plotting histogram of the original image %%
h=zeros(1,300);
for n=1:1:row
for m=1:1:col
t=a(n,m);  %% Takes the value of the pixel for example 12
h(t)=h(t)+1; %% Incrementing the corresponding counter h(12)
end
end
figure(1), bar(h)
    % Plotting histogram of the modified image %
c=c+1;
h=zeros(1,400);
for n=1:1:row
  for m=1:1:col
    t=round(c(n,m));
    h(t)=h(t)+1;
   end
end
figure(2),bar(h)
figure(3),imshow(uint8(a))
figure(4),imshow(uint8(c))
```

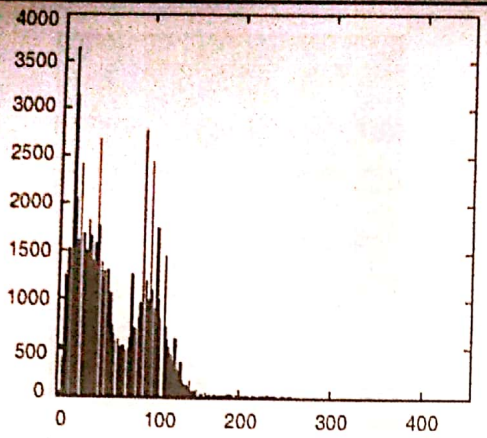
(a) Original image

(b) Image after histogram stretching



(c) Original histogram

Fig. 8.2.3 : Cont...



(d) Linear stretched histogram

Fig. 8.2.3

8.3 Histogram Equalization

- Linear stretching is a good technique but as mentioned earlier, the shape remains the same.
- There are many applications, wherein we need a flat histogram. This cannot be achieved by histogram stretching.
- We now introduce a new technique known as histogram equalization. A perfect image is one which has equal number of pixels in all its grey levels. Hence our objective is not only to spread the dynamic range, but also to have equal pixels in all the grey levels.
- This technique is known as histogram equalization. We now search for a transformation that would transform a bad histogram to a flat histogram

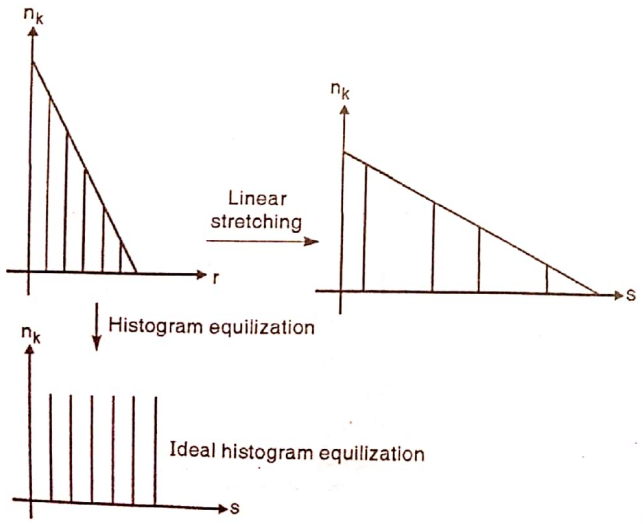


Fig. 8.3.1

We know, $s = T(r)$

What is this T which produces equal values in each grey level ?

- The transformation that we are looking for must satisfy the following two conditions :

(a) T(r) must be single valued and monotonically increasing in the interval $0 \le r \le 1$ and

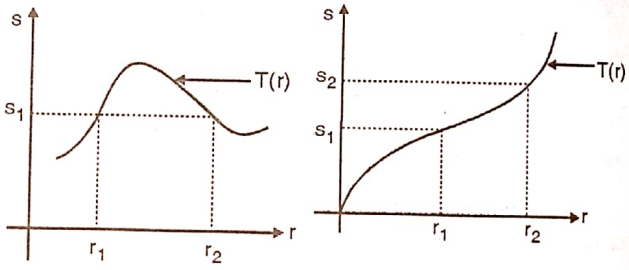
(b) $0 \le T(r) \le 1$ for $0 \le r \le 1$ i.e., $0 \le s \le 1$ for $0 \le r \le 1$

Here the range of r is taken as [0, 1]. This is called the normalized range. This range is taken for simplicity. So instead of r being in the range [0, 255] we take [0, 1]

Let us see what these two conditions :

(1) If T (r) is not single valued. Refer Fig. 8.3.2(a). Here r_1 and r_2 are mapped as a single grey level s_1 i.e., two different grey levels look the same in the modified image. This is a big drawback, hence the transformation has to be single valued. This preserves the order from black to white.

A grey level transformation that is both single valued and monotonically increasing is shown in Fig. 8.3.2(b).



(a) (b)

Fig. 8.3.2

(2) If condition (b) is not satisfied, then the mapping will not be consistent with the allowed range of pixel values. In other words, s will go beyond the valid range.

Since the transformation is single valued and monotonically increasing, the inverse transformation exists.

$\therefore r = T^{-1}(s) ; 0 \le s \le 1$

- The grey levels for continuous variables can be characterized by their probability density functions $p_r(r)$ and $p_s(s)$.
- From probability theory we know that if $p_r(r)$ and T(r) are known and if $T^{-1}(s)$ satisfies condition (a), then the probability density of the transformed grey level is

$$p_s(s) = \left[p_r(r)\frac{dr}{ds}\right]_{r=T^{-1}(s)} \quad \ldots(8.3.1)$$

i.e., the probability density of the transformed image is equal to the probability density of the original image multiplied by the inverse slope of the transformation.

– We now need to find a transformation which would give us a flat histogram. Let us consider the Cumulative Density Function (CDF). Cumulative density function is obtained by simply adding up all the Probability Density Functions (PDF).

$$s = T(r) = \int_0^r p_r(r)\,dr; \quad 0 \le r \le 1 \quad \ldots(8.3.2)$$

Differentiating with respect to r, we get

$$\frac{ds}{dr} = p_r(r) \quad \ldots(8.3.3)$$

Using Equation (8.3.3) in Equation (8.3.1)

$$p_s(s) = [1] = 1; \quad 0 \le s \le 1$$

i.e., $p_s(s) = 1$

This is nothing but a uniform density function !!

Let us see what we have got here.

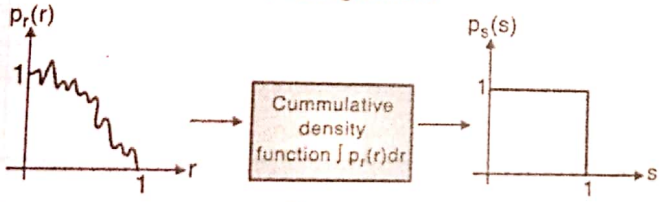


Fig. 8.3.3

A bad histogram becomes a flat histogram when we find the cumulative density function!! Let us cement this concept by taking an examples.

8.3.1 Solved Examples on Histogram Equalization

Ex. 8.3.1 For the given histogram. Perform histogram equalization.

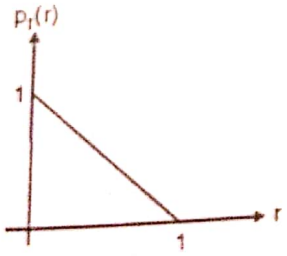


Fig. P. 8.3.1

Note : $p_r(r)$ is the PDF and hence cannot be greater than 1.

Soln. :

From the diagram, we know that the above histogram represents a dark image.

$$p_r(r) = \begin{cases} -r+1 & ; 0 \le r \le 1 \\ 0 & ; \text{otherwise} \end{cases}$$

We shall use the CDF that we just discovered.

$$s = T(r) = \int_0^r p_r(r)dr = \int_0^r (-r+1)\,dr$$

$$s = \frac{-r^2}{2} + r$$

$$2s = -r^2 + 2r$$

$$2s - 1 = -r^2 + 2r - 1$$

$$(1-2s) = (r^2 - 2r + 1)$$

$$(1-2s) = (r-1)^2$$

$$(r-1) = \pm\sqrt{1-2s}$$

$$r = 1 \pm \sqrt{1-2s} \qquad \because 0 \le r \le 1$$

$$\therefore r = 1 - \sqrt{1-2s}$$

We know,

$$p_s(s) = \left[p_r(r)\frac{dr}{ds}\right] = \left[(-r+1)\frac{dr}{ds}\right]$$

Substituting the value of r,

$$p_s(s) = (-1 + \sqrt{1-2s} + 1)\frac{d}{ds}\left\{1 - (1-2s)^{1/2}\right\}$$

$$p_s(s) = \sqrt{1-2s} \cdot \frac{d}{ds}\left\{-(1-2s)^{1/2}\right\}$$

$$\left(\because \frac{d}{ds}(1) = 0\right)$$

$$= \sqrt{1-2s} \cdot \left\{\frac{-1}{2}(1-2s)^{-1/2}(-2)\right\}$$

$$= \sqrt{1-2s} \cdot \frac{1}{\sqrt{1-2s}}$$

$\therefore p_s(s) = 1;$

Hence $p_s(s) = 1; \quad 0 \le r \le 1.$

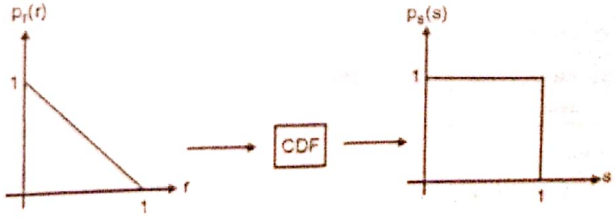


Fig. P. 8.3.1(a)

We see that $p_r(r)$ which represented a dark histogram has been transformed to an equalized histogram.

Ex. 8.3.2 : Though $p_r(r)$ cannot be greater than 1, you could just consider it as another problem and solve it

$$p_r(s) = \begin{cases} -2r + 2 & ; 0 \le r \le 1 \\ 0 & ; \text{otherwise} \end{cases}$$

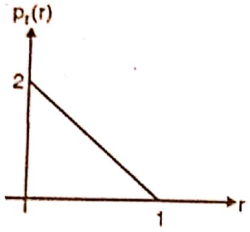


Fig. P. 8.3.2

Soln. :

Try out this problem yourself and see what you get.

We see that CDF gives us a flat response no matter what the shape of the original histogram is.

So far, we have developed the technique of histogram equalization using continuous domain mathematics. For image processing, we need to work in the discrete domain.

Hence if n is the total number of pixels in an image, then the PDF is

$$p_r(r_k) = \frac{n_k}{n}$$

We know, $$s = T(r) = \int_0^r p_r(r)$$

In the discrete domain,

$$s_k = T(r_k) = \sum_0^r p_r(r).$$ This is the C. D. F

Let us take up an example to see how this works in the discrete domain. Some additional steps need to be taken into account while working in the discrete domain.

Ex. 8.3.3 : Equalize the given histogram

Grey level	0	1	2	3	4	5	6	7
Number of pixels	790	1023	850	656	329	245	122	81

Soln. :

L = 8 (Number of grey levels)

We first plot the original histogram

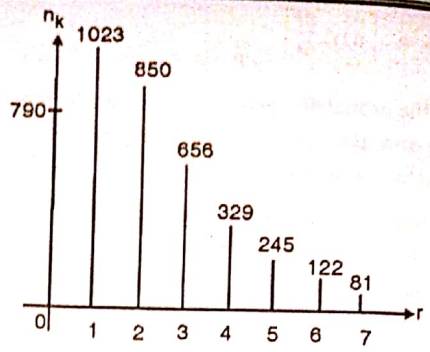


Fig. P. 8.3.3 : Original dark histogram

Grey level	n_k	PDF $p_r(r_k) = \frac{n_k}{n}$	CDF $s_k = \sum p_r(r_k)$	$(L-1) \times s_k$ i.e. $7 \times s_k$	Rounding off
0	790	0.19	0.19	1.33	1
1	1023	0.25	0.44	3.08	3
2	850	0.21	0.65	4.55	5
3	656	0.16	0.81	5.67	6
4	329	0.08	0.89	6.23	6
5	245	0.06	0.95	6.65	7
6	122	0.03	0.98	6.86	7
7	81	0.02	1	7	7
	N = 4096				

We take 1st, 2nd and the last column

Old grey level	Equalized grey level	New grey level
0	790	→ 1
1	1023	→ 3
2	850	→ 5
3	656	→ 6
4	329	→ 6
5	245	→ 7
6	122	→ 7
7	81	→ 7

We notice that the new grey levels have pixels only at 1, 3, 5, 6, 7. There are no pixels in grey levels 0, 2 and 4.

Equalized grey level	Number of pixels
0	0
1	790
2	0
3	1023
4	0
5	850
6	656 + 329 = 985
7	245 + 122 + 81 = 448

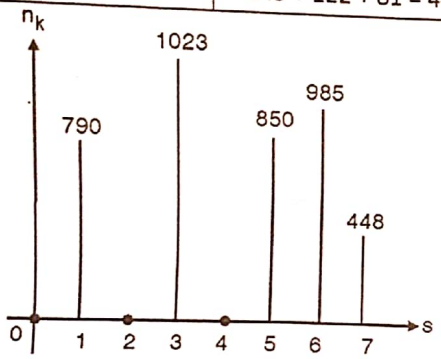


Fig. P. 8.3.3(a) : Equalized histogram

So here we are. A dark histogram becomes an evenly spaced histogram. But wait a minute, we had started out to get a flat histogram as in the continuous case but the histogram obtained is not flat; why?

Discrete domain is an approximation of the continuous domain i.e., values between 0 and 1, 1 and 2 and so on are not known. Due to this reason, perfectly flat results are never obtained.

Ex. 8.3.4

Grey level	n_k
0	100
1	90
2	50
3	20
4	0
5	0
6	0
7	0

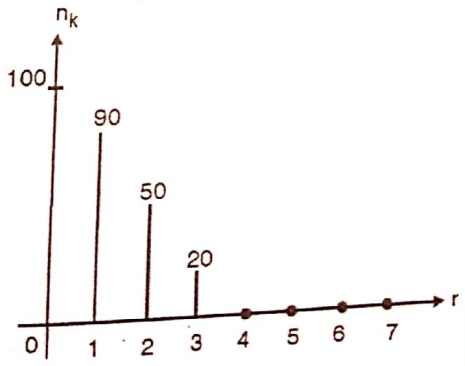


Fig. P. 8.3.4

Equalize the above histogram (Here L = 8)

Soln. :

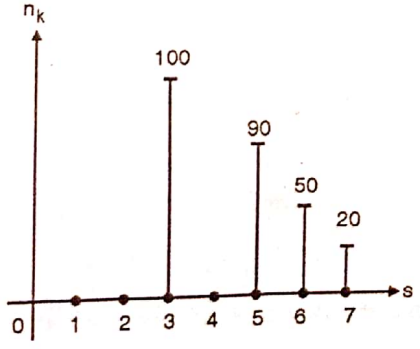
Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$s_k = \sum p_r(r_k)$	$s_k \times (L-1)$ i.e. $s_k \times 7$	Rounding off
0	100	0.384	0.384	2.688	3
1	90	0.346	0.73	5.11	5
2	50	0.1923	0.9223	6.456	6
3	20	0.0769	1	7	7
4	0	0	1	7	7
5	0	0	1	7	7
6	0	0	1	7	7
7	0	0	1	7	7
	n = 260				

We take the 1st, 2nd and the last column.

Old grey levels	n_k	New grey levels	New n_k	Modified grey levels
0	100 →	3 →	100	0 → 0
1	90 →	5 →	90	1 → 0
2	50 →	6 →	50	2 → 0
3	20	7		3 → 100
4	0	7		4 → 0
5	0	7 (20 + 0 + 0 + 0 + 0 = 20)	20	5 → 90
6	0	7		6 → 50
7	0	7		7 → 20

The original histogram showed that the image had only 0-3 grey levels. After histogram equalization we see that the dynamic range has increased. Hence histogram equalization does increase the dynamic range.

Next question that is usually asked is what happens when we equalize an already equalized histogram. How many times can we equalize an image ?



Equalized histogram

Fig. P. 8.3.4(a)

i.e. Dark histogram → Equalize → Flat histogram → Equalize again ?

We shall see what happens by considering an example.

Ex. 8.3.5 : Given a histogram, see what happens when we equalize it twice.

Grey level	0	1	2	3
n_k	70	20	7	3

Soln. :

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$s_k = \sum p_r(r_k)$	$L = 4$ $s_k \times (L-1)$	Rounding off	New n_k
0	70	0.7	0.7	2.1	2 →	70
1	20	0.2	0.9	2.7	3	
2	7	0.07	0.97	2.91	3 →	20 + 7 + 3
3	3	0.03	1	3	3	= 30
L = 4		100				

∴ The modified grey levels are

Grey level	0	1	2	3
n_k	0	0	70	30

We now equalize this again.

Grey level	n_k	$p_r(r_k)$	s_k	$L = 4$ $s_k \times (L-1)$	Rounded off	Modified grey level
0	0	0	0	0	0	
1	0	0	0	0	0	
2	70	0.7	0.7	2.1	2 →	70
3	30	0.3	1	3	3 →	30
	100					

∴ we get,

Grey level	0	1	2	3
n_k	0	0	70	30

Which is the same as what we had got after the first equalization.

Hence, equalizing again gives us the same result i.e., equalizing an already equalized histogram causes no change in the histogram. Go ahead, try out your own examples. They will all give similar results.

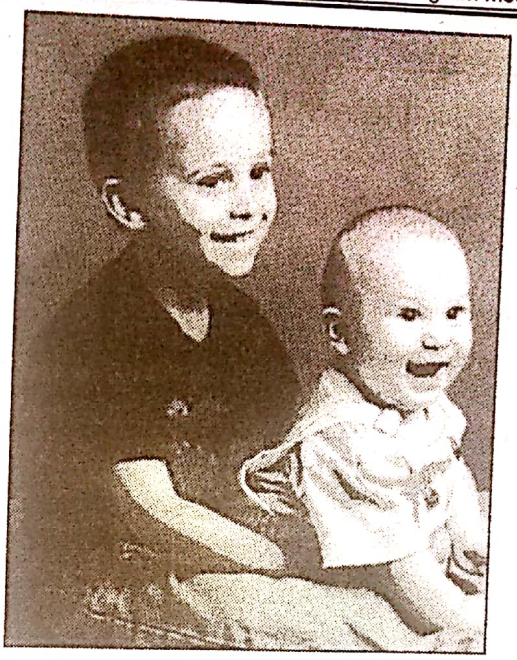
Histogram equalization is guaranteed to yield exact results only in the continuous case. As the number of levels decrease, the error between the specified and resulting histogram increase and hence we do not get a flat histogram.

MATLAB code for histogram equalization

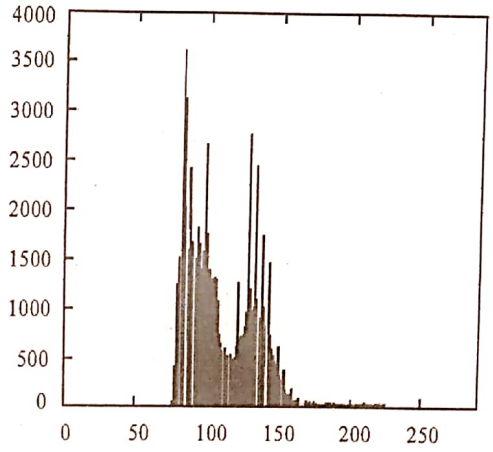
```
%% Histogram equalization %%
clear all
clc
a=imread('kids2.tif');
% You can try the program with 'pout.tif'%
a=double(a);
big=max((max(a)));
%% Gives maximum value in the image
[row col]=size(a);
C=row*col;   %% Gives the total number of pixels required
h=zeros(1,300);
%% Defining two arrays (h and z) to store the histogram values
z=zeros(1,300);
for n=1:1:row
        for m=1:1:col
            if a(n,m)==0
          %% To ensure that the values of 'a' are not zero
    a(n,m)=1;
        end
 end
end
%% Plotting the histogram of the original image %%
for n=1:1:row
        for m=1:1:col
            t=a(n,m);
%% Takes the value of the pixel example 12 %%
        h(t)=h(t)+1;
%% Incrementing the corresponding counter h(12)%%
        end
end
pdf=h/C;              %% Probability function %%
cdf(1)=pdf(1);
%% So that we don't have the condition x(0) in the
                        for loop below %
for x=2:1:big
cdf(x)=pdf(x)+cdf(x-1);  %% Calculating the cdf %%
```

```
end
new=round(cdf*big);        %% New is the new gray level
new=new+1;        %% if new = 0, it is not to be taken in
the loop
for p=1:1:row
for q=1:1:col
            temp=a(p,q);
            b(p,q)=new(temp);
%% b has the equalized image %%
%% Plotting the equalised histogram %%
    t=b(p,q);
%% Takes the value of the pixel example 12 %%
            z(t)=z(t)+1; %% Incrementing the corresponding
counter z(12)%%
    end
end
b=b-1;
%% Since we had initially incremented the value of n %%
%% Plotting %%
figure (1), imshow(uint8(a))
figure (2), bar(h)
figure (3), imshow(uint8(b))
figure (4), bar(z)
```

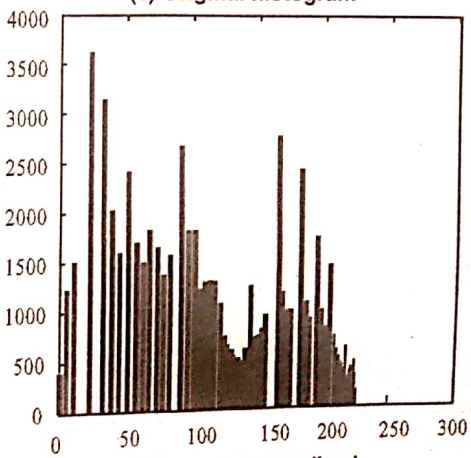
Fig. P. 8.3.5(a) : Original image



(b) Image after histogram equalization



(c) Original histogram



(d) Histogram equalized

Fig. P. 8.3.5

8.4 Histogram Specification

- From the earlier method (Histogram equalization) we note that histogram equalization is automatic. It is not interactive. i.e., it always gives one result - an approximation to an uniform histogram. It is at times desirable to have an interactive method in which certain grey level are highlighted.
- It should be noted that if we modify the grey level of an image that has a uniform PDF, $p_s(s)$, using the inverse transformation $r = T^{-1}(s)$, we get back the original histogram $p_r(r)$. Exploiting this knowledge, we can obtain any shape of the grey level histogram by processing the given image in the following way.
- Suppose $p_r(r)$ and $p_s(s)$ represent grey level PDFs of input and output images and r and s are their respective grey levels. Suppose k represents grey level of some intermediate image result i.e.,

$$k = T_1(r) = \int_0^r p_r(r)\,dr \quad \text{and} \quad k = T_2(r) = \int_0^s p_s(s)\,ds$$

- In both these cases $p_k(k)$ is uniform. Hence the transformation $s = T_2^{-1}(T_1(r))$ achieves the desired result. The procedure of histogram specification can be computed as follows :
  (1) Equalize the levels of the original image.
  (2) Specify the desired density function and obtain the transformation function.
  (3) Apply the inverse transformation function.

8.4.1 Solved Example Histogram

Ex. 8.4.1 : Given histograms (a) and (b), modify histogram a as given by histogram (b).

Histogram (a)

Grey	0	1	2	3	4	5	6	7
Number of pixels	790	1023	850	656	329	245	122	81

Histogram (b)

Grey	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	614	819	1230	819	614

Soln. :

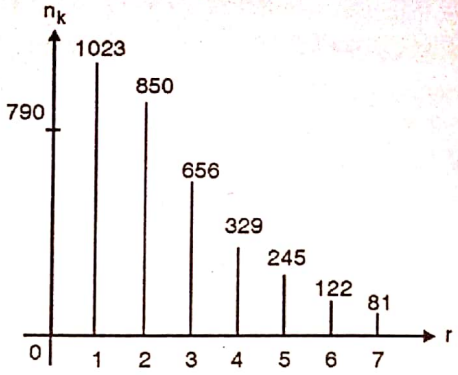


Fig. P. 8.4.1(a)

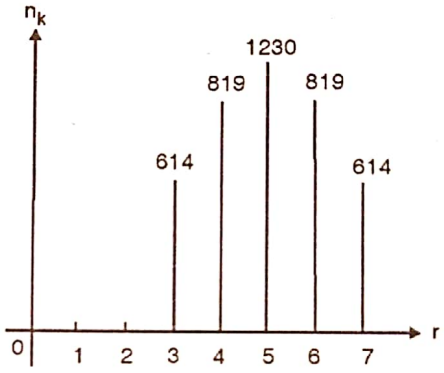


Fig. P. 8.4.1(b)

Now equalize (a) as well as (b)

Histogram (a) : L = 8

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	CDF = $\sum p_r(r_k)$	CDF × (L − 1)	Rounding off		New grey level
0	790	0.19	0.19	1.33	1	→	790
1	1023	0.25	0.44	3.08	3	→	1023
2	850	0.21	0.65	4.55	5	→	850
3	656	0.16	0.81	5.67	6	→	656 +
4	329	0.08	0.89	6.23	6		329 =
							985
5	245	0.06	0.95	6.25	7	→	
6	122	0.03	0.98	6.86	7	→	245 +
							122 +
							81
7	81	0.02	1	7	7	→	= 448
	4096						

Note that there are 5 distinct levels after histogram equilization of histogram (a)

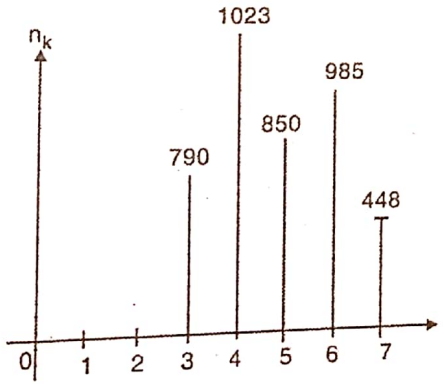
Grey level	0	1	2	3	4	5	6	7
Number of pixels	0	790	0	1023	0	850	985	448

Now equalize (b) : L = 8

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	CDF	CDF × (L – 1)	New grey level
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	614	0.149	0.149	1.05	1
4	819	0.20	0.35	2.45	3
5	1230	0.30	0.65	4.55	5
6	819	0.20	0.85	5.97	6
7	614	0.15	1	7	7
	4096				

To obtain histogram specification, we apply the inverse transform comparing equalized (a) and (b).

1 → 3
3 → 4
5 → 5
6 → 6
7 → 7



Resultant histogram

Fig. P. 8.4.1(c)

Hence the final result is as shown

Grey	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	790	1023	850	985	448

Hence we see that histogram (a) has been equalized and mapped in such a way that it resembles the histogram (b). This is histogram specification.

8.5 Solved Examples on Histogram Modelling

Ex. 8.5.1 : Two images can have the same histogram (Justify/ Contradict with reason). **MU - Dec 2017, 5 Marks**

Soln. :

Histograms give us information about the numbers of grey levels present in the image, it does not give us any spatial information. This means, histograms do not tell us where the grey levels lie in an image. If all pixels in an image are shuffled, there will be no change in the histogram. For example, the given two images are completely different but their histograms will be equal as the total number of black pixels and white pixels are the same in both the images. Hence two images can have the same histogram.

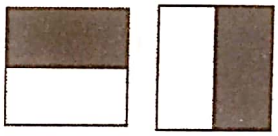


Fig. P. 8.5.1

Ex. 8.5.2 : What effect would setting to zero the lower order bit planes have on the histogram of an image in general ?

Soln. : The number of pixels having different grey level values would decrease, thus causing the number of components in the histogram to decrease. Since the number of pixels is constant, this would cause the height of the remaining histogram peaks to increase. Typically, less variability in grey level values will reduce the contrast.

Let us check this out with an example.

Given below is a 4 × 4 image having grey levels 0-18.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Histogram of the original image is as shown in Fig. P. 8.5.2.

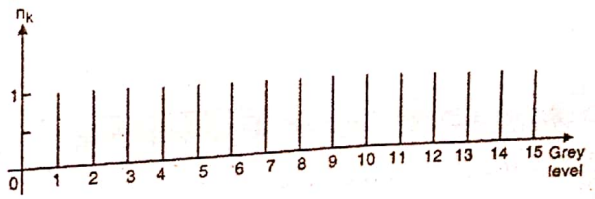


Fig. P. 8.5.2

We convert the original image to a binary image. Since the maximum grey level is 15, we need 4-bits for its representation.

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

Suppose we make the last two bits equal to zero, we get

0000	0000	0000	0000
0100	0100	0100	0100
1000	1000	1000	1000
1100	1100	1100	1100

Converting back to the grey level format we get,

0	0	0	0
4	4	4	4
8	8	8	8
12	12	12	12

∴ its histogram is as shown in Fig. P. 8.5.2(a).

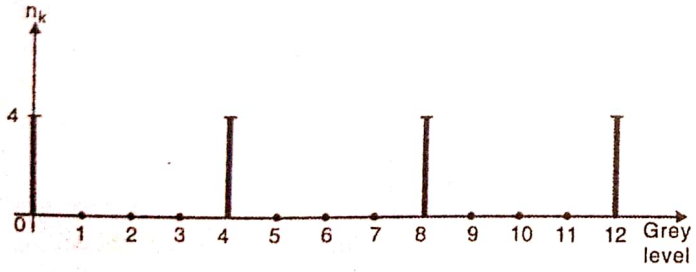


Fig. P. 8.5.2(a)

Comparing this histogram with the original histogram, we see that the variability is reduced, i.e. number of grey levels are reduced and the height of these grey levels has increased.

Ex. 8.5.3 : What would be the effect on the histogram if we set to zero, the higher order bit planes.

Soln. :

Proceeding in a similar manner as in Ex. 8.5.2 and working with the same image, we get

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

(a)

Converting to binary

0000	0001	0010	0011
0100	0101	0110	0111
1000	1001	1010	1011
1100	1101	1110	1111

(b)

Converting the upper two bits to zero

0000	0000	0000	0000
0100	0100	0100	0100
1000	1000	1000	1000
1100	1100	1100	1100

(c)

Converting back to grey levels we get

0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

(d)

Fig. P.8.5.3

Plotting the histogram as shown in Fig. P. 8.5.3(e).

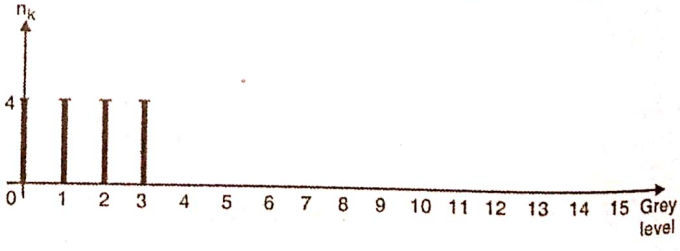


Fig. P. 8.5.3 (e)

In this case too, the grey levels reduce. But one important thing that happens is that the image becomes much darker.

Ex. 8.5.4 : A digital image with 8 quantization levels is given below :

Perform histogram equalization.

$f(x,y) = |x - y|$ for x = 0 to 7 and y = 0 to 7

MU - Dec. 2018, 10 Marks

Soln. :

In this we form an image by taking the modulus of the difference of the spatial coordinates. The image that is formed using the equation $f(x,y) = |x - y|$ is shown below :

$f(x, y) =$

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	1	2	3	4	5	6
2	2	1	0	1	2	3	4	5
3	3	2	1	0	1	2	3	4
4	4	3	2	1	0	1	2	3
5	5	4	3	2	1	0	1	2
6	6	5	4	3	2	1	0	1
7	7	6	5	4	3	2	1	0

We generate the frequency table / histogram of the original image.

Grey levels	0	1	2	3	4	5	6	7
Number of pitch	8	14	12	10	8	6	4	2

The original histogram is shown in Fig. P.8.5.4.

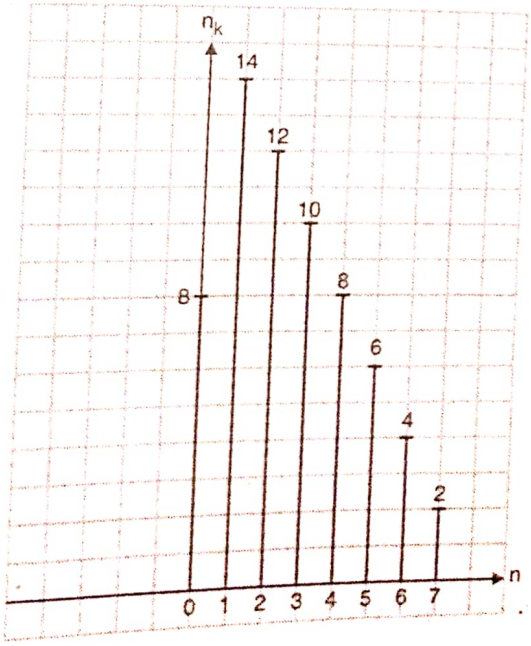


Fig. P. 8.5.4

We now, generate a table to perform histogram equalization

Grey level	n_k	$p_r(r_k) = \frac{n_k}{n}$	$S_k = \Sigma p_r(r_k)$	$S_k \times (L-1)$ $S_k \times 7$	Rounding off
0	8	0.125	0.125	0.875	1
1	14	0.218	0.343	2.401	2
2	12	0.187	0.53	3.71	4
3	10	0.156	0.686	4.802	5
4	8	0.125	0.811	5.677	6
5	6	0.093	0.904	6.328	6
6	4	0.062	0.966	6.762	7
7	2	0.031	1	7	7
	$\sum n_k = 64$				

We consider the 1st, 2nd and last column

Greg level	n_k	Rounding off	New n_k
0	8	1→	8
1	14	2→	14
2	12	4→	12
3	10	5→	10
4	8	6 } 8 + 6→	14
5	6	6 }	
6	4	7 } 4 + 2 →	6
7	2	7 }	

Hence the modified frequency table is

Grey levels	0	1	2	3	4	5	6	7
n_k	0	8	14	0	12	10	14	6

The equalized histogram is shown in Fig. P. 8.5.4(a).

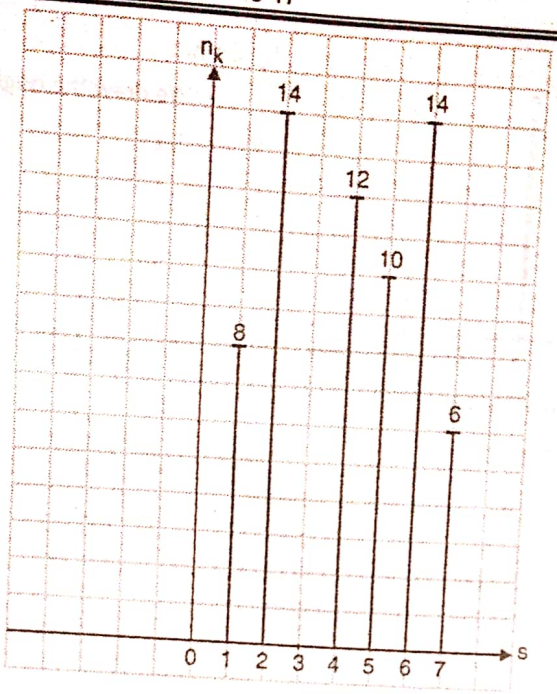


Fig. P. 8.5.4(a)

Ex. 8.5.5 : *Given below is the slope transformation and the image. Draw the frequency tables for original and new image. Assume* $l_{min} = 0$ *and* $l_{max} = 7$.

2	3	4	2
5	5	2	4
3	6	3	5
5	3	5	5

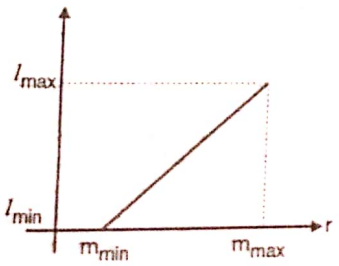


Fig. P. 8.5.5

Soln. : This is a sum of histogram linear stretching. By frequency tables, we mean histogram. From the image, it is clear that $m_{min} = 2$, $m_{max} = 6$, $l_{min} = 0$ and $l_{max} = 7$.

Using the formula

$$l = \frac{l_{max} - l_{min}}{m_{max} - m_{min}} (m - m_{min}) + l_{min}$$

$$\therefore \quad l = \frac{7-0}{6-2} (m-2) + 0$$

$$\therefore \quad l = \frac{7}{4} (m-2)$$

From the above formulation

when, $m = 2, \quad l = 0$

$m = 3, \quad l = 1.75 \approx 2$

$m = 4, \quad l = 3.5 \approx 4$

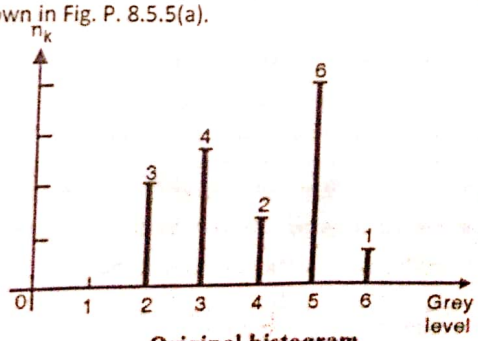
$m = 5, \quad l = 5.25 \approx 5$

$m = 6, \quad l = 7$

$\therefore$ The modified image is

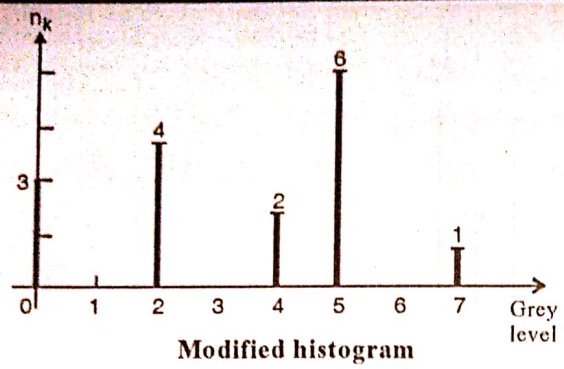
0	2	4	0
5	5	0	4
2	7	2	5
5	2	5	5

Histogram of the original and modified images are shown in Fig. P. 8.5.5(a).



Original histogram

Fig. P. 8.5.5(a)



Modified histogram

Fig. P. 8.5.5(b)

Ex. 8.5.6 : A popular procedure for image enhancement combines high frequency emphasis and histogram equalization to achieve edge sharpening and contrast enhancement. Does it matter which of the two processes are applied first ?

Soln. :

High frequency emphasis is the same as a high boost filter. Let us see what we get when we apply a high frequency emphasis filter.

$$f(x,y) \longrightarrow \boxed{\begin{matrix} h(x,y) \\ \text{High boost} \end{matrix}} \longrightarrow f(x,y) * h(x,y) = g(x,y)$$

i.e., the output g(x, y) is a convolution of the input image and the impulse response of the high boost filter.

If we now perform histogram equalization we get

$$s = T[g(x, y)]$$

$$s = T[f(x, y) * h(x, y)] \quad \ldots(1)$$

If we decide to apply histogram equalization prior to high boost filter, we get

$$T[f(x,y)] \longrightarrow \boxed{\begin{matrix} h(x,y) \\ \text{High boost} \end{matrix}} \longrightarrow T[f(x,y)] * h(x,y)$$

$\ldots(2)$

We see that Equation (1) ≠ Equation (2).

∴ The order of the operation does matter i.e., the final result varies if we interchange the two operations.

Ex. 8.5.7 : The grey level distribution of an image is shown in the table below. Perform histogram equalization and plot the original and equalized histograms.

Grey level	0	1	2	3	4	5	6	7
Frequency of occurrence	0	50	100	200	400	200	50	0

MU - Dec. 2013, 10 Marks

Soln. :

We draw the original histogram of the image.

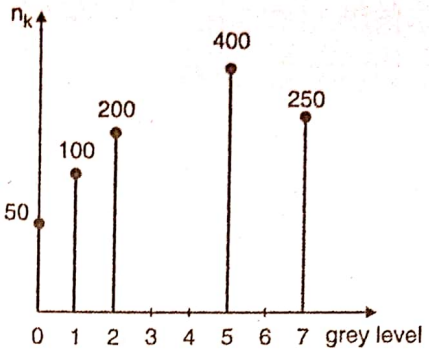


Fig. P. 8.5.7

Here L = 8 We now perform histogram equalization.

Grey level	n_k	PDF $p_r(k) = \frac{n_k}{\sum n}$	CDF i.e., $s_k = \sum p_r(k)$	$s_k \times (L-1)$ i.e., $s_k \times 7$	Round off
0	0	0	0	0	0
1	50	0.05	0.05	0.35	0
2	100	0.1	0.15	1.05	1
3	200	0.2	0.35	2.45	2
4	400	0.4	0.75	5.25	5
5	200	0.2	0.95	6.65	7
6	50	0.05	1	7	7
7	0	0	1	7	7
	$\sum n = 1000$				

Consider the 1st, 2nd and last column

Grey level	n_k	New grey level	
0	0	0	0 + 50 = 50
1	50	0	
2	100	1	100
3	200	2	200
4	400	5	400
5	200	7	200 + 50 = 250
6	50	7	
7	0	7	

Hence the equalized frequency table is,

Grey level	0	1	2	3	4	5	6	7
n_k	50	100	200	0	0	400	0	250

The equalized histogram is shown in Fig. P. 8.5.7(a)

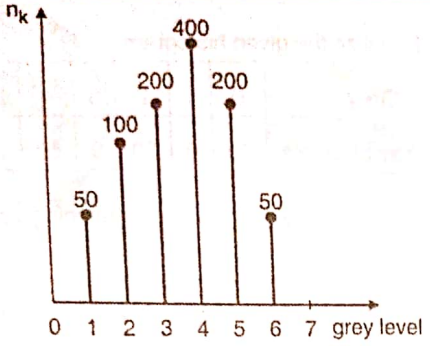


Fig. P. 8.5.7(a)

Ex. 8.5.8 : A 64 × 64 image, represented by 3 bit / pixel has the following grey level distribution :

Gray level	0	1	2	3	4	5	6	7
No. of Pixel	128	75	280	416	635	1058	820	684

Perform histogram equalization and give new distribution of gray level. Show plots of original and equalized image.

MU - May 2011, Dec. 2012, 10 Marks

Soln. :

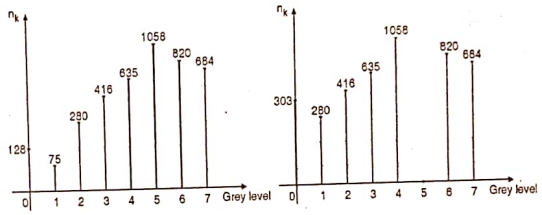
We generate the histogram table

Grey level	n_k	PDF $p_r(r_k) = \frac{n_k}{n}$	CDF $s_k = \Sigma p_r(r_k)$	$(L-1) \times s_k$	Rounding off
0	128	0.031	0.031	0.217	0
1	75	0.018	0.0496	0.347	0
2	280	0.068	0.1179	0.825	1
3	416	0.101	0.2195	1.536	2
4	635	0.155	0.3745	2.622	3
5	1058	0.258	0.6328	4.430	4
6	820	0.201	0.8330	5.831	6
7	684	0.1670	1	7.00	7
	n = 4096				

We take the 1st, 2nd and the last column

Old grey level	n_k	New grey level	Modified grey level
0	128	0 → 128 + 0 → 175	0 → 303
1	75		1 → 280
2	280	1 → 280	2 → 416
3	416	2 → 416	3 → 635
4	635	3 → 635	4 → 1058
5	1058	4 → 1058	5 → 0
6	820	6 → 820	6 → 820
7	684	7 → 684	7 → 684

We now draw the original as well as the equalized histogram.



(a) Original histogram (b) Equalized histogram

Fig. P. 8.5.8

Summary

In this chapter, the concept of histograms is introduced. Though histogram processing is a spatial domain technique, it is presented here as a separate chapter because of its wide spread applications. Histograms give us an idea about the quality of an image. One of the most common defects found in recorded images is poor contrast. This degradation may be caused by inadequate lighting, aperture size and/or shutter speed. Procedures of modifying the histogram to get a high contrast image are discussed. Emphasis has been laid on the two important techniques: linear stretching and histogram equalization.

Review Questions

Q. 1 Plot the graph of a cumulative density function (CDF).

Q. 2 Explain how histogram can aid in the process of thresholding and contrast stretching.

Q. 3 Equalize the given histogram.

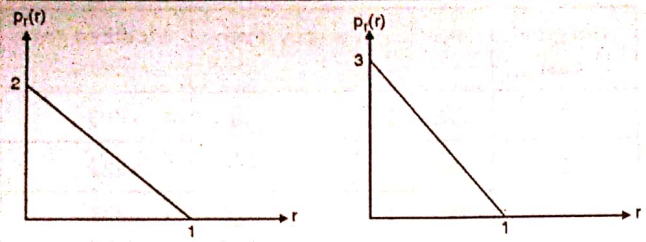


Fig. Q. 3

Note : $p_r(r)$ cannot be greater than 1 but solve this as any other mathematical problem.

Q. 4 Explain why a discrete histogram technique does not, in general, yield a flat histogram.

Q. 5 Suppose that a image is subjected to a histogram equalization. Show that a second pass of histogram equalization will produce exactly the same result as the first pass.

Q. 6 What are the two conditions that a transformation has to satisfy ?

Q. 7 Write down an algorithm to plot the histogram of an image.

Q. 8 Equalize the given histogram.

Grey	0	1	2	3	4	5	6	7
Number of pixels	50	0	50	0	50	0	50	0

Q. 9 Compare between contrast stretching and histogram equalization.

Q. 10 Suppose we have a dark image which needs to be compressed and also equalized. Which operation would we use first ? Will the results be the same if the order of operations is reserved ?

Q. 11 Equalize the given image.

0	1	2
3	4	5
6	7	8

Q. 12 Given the frequency table,

Grey	0	1	2	3	4	5	6	7
Number of pixels	0	0	a	b	c	d	e	0

Perform linear stretching.

□□□

CHAPTER 9

Segmentation

Syllabus :

Segmentation based on Discontinuities (point, Line, Edge), Image Edge detection using Robert, Sobel, Previtt masks, Image Edge detection using Laplacian Mask. Connectivity

9.1 Introduction

May 2016, Dec. 2016, Dec. 2018, 2 Marks

Q. What is image segmentation

(May 2016, Dec. 2016, Dec. 2018, 2 Marks)

- Segmentation, as the name suggests, subdivides (segments) an image into its constituent regions or objects. Unlike image enhancement, were our primary concern was to improve the subjective quality of images for display, in segmentation, we address some aspects of analyzing the content of an image.
- This simply means we endeavour to find out what is in the picture. Segmentation forms a section of computer vision i.e., we use segmentation, when we want the computer to make decisions.
- Segmentation is used when we need to automate a particular activity. The ultimate aim in an automated system is to extract important features from image data, from which description, interpretation, or understanding of the scene can be provided by the machine.
- Segmentation is not required when the images have to be shown to a human being. This is because a human visual system has an inherent quality to segment the image shown to it.

Let us take two simple examples where image segmentation is used.

(1) Automated blood cell counting :

- We all know blood is made up of RBC's, WBC's, platelets etc.
- In manual machines, the technician looks at the slide and based on various parameters classifies them as RBC's, WBC's etc. As stated earlier, the human visual system automatically segments the image as RBC's, WBC's... etc.
- In an automated blood cell counting machine, this classification has to be made by the machine (computer). The computer scans the slide and based on certain algorithms classifies them as RBC's, WBC's....etc.
- We hence needed to do image segmentation to divide the image into various parts.

(2) Finger print matching in forensic studies

- In this, there exists a data base of finger prints of people who have been convicted of crime at some point in time.
- Now if there has been a crime at a place and some finger prints have been found, images are taken, and these images are run through an algorithm to see if this particular finger print matches with the one in the data base.
- Here too, since the system is automated we need to use segmentation to divided the image into various parts.
- We hence say that image segmentation is used for computer decision making (machine vision) and not for human interpretation and the main purpose of segmentation is to partition the image.
- Image segmentation algorithms are generally based on one of the two basic properties i.e. image segmentation can be achieved in any one of the two ways viz.
  (1) Segmentation based on discontinuities in intensity.
  (2) Segmentation based on similarities in intensity.

In the first method, the approach is to partition an image based on abrupt changes in intensity, such as edges, while in the second method, we partition an image into regions that are similar according to a set of predefined criteria.

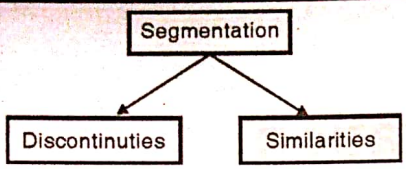


Fig. 9.1.1

- The syllabus only contains Segmentation based on discontinuities

9.2 Image Segmentation based on Discontinuities

When we look at an image, we immediately observe 3 basic types of discontinuities in the grey level viz. points, lines and edges. The easiest way is to use spatial masks which have properties to detect these discontinuities.

9.2.1 Point Detection

- Remember one thing, points, lines and edges are all high frequency components and hence we need masks which are basically high pass. Hence the masks that we present here for point, line and edge detection have the properties of a high mask mainly, the sum of the coefficients of the mask has to be equal to zero in all the three cases.
- Detecting points is fairly simple. We use the standard high pass mask for it. And so that this mask detects only points and not lines, we set a threshold value i.e., we say a point has been detected at the location on which the mask is centered only if

−1	−1	−1
−1	8	−1
−1	−1	−1

$$|R| \geq T \qquad \ldots(9.2.1)$$

Where R is derived from the standard convolution formula

$$R = w_1 z_1 + w_2 z_2 + \ldots\ldots\ldots + w_9 z_9$$

$$R = \sum_{i=1}^{9} w_i z_i$$

- We take |R| because we want to detect both the kinds of points i.e. white points on a black background as well as black points on a white background.

  T is a non negative threshold which is defined by the user.

9.2.2 Line Detection

- Detection of lines can be done using the masks shown below. In an image, lines can be in any direction and detecting these lines would need different masks.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

- We notice that all these masks have a sum equal to zero, and hence all of them are high pass masks. The first mask would respond strongly to lines that are oriented horizontally. The second mask would respond to lines at an angle of + 45°, the third mask would respond strongly to vertical lines and the last mask would respond strongly to lines at an angle of − 45°.

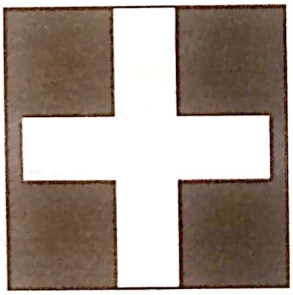


Fig. 9.2.1

Let us take a simple example of a 8 × 8 pseudo image.

0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	10	10	10	10	10	10	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0
0	0	0	10	0	0	0	0

- It is quite evident that in this image we have two lines, one horizontal and the second vertical.
- Let us see what happens when we move the first (Horizontal) mask over the image. By now you should be quite comfortable with moving masks on the entire image. We leave the borders and hence start from the encircled pixel.

The result that we obtain is as shown

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	−20	−20	−20	−20	−30	−20	0
0	+40	+40	+40	+40	+60	+40	0
0	−20	−20	−20	−20	−30	−20	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Since line detection masks are high pass masks, we encounter negative values.
- These values are made zero as was discussed in chapter of Image Enhancement in Spatial Domain. Hence the final image obtained as follows,

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	40	40	40	40	60	40	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- We see that the horizontal mask detects lines only in the horizontal direction and cleans up the other lines (vertical line in this case).
- In a similar fashion we can show that vertical, + 45°, and − 45° masks respond strongly to lines in their respective directions. It would be a good idea to try this out yourself.

9.2.3 Edge Detection

- More than isolated points and lines, it is the detection of edges that form an important part of image segmentation. An edge can be defined as a set of connected pixels that form a boundary between two disjoint regions.
- A typical example of an edge is shown in Fig. 9.2.2.

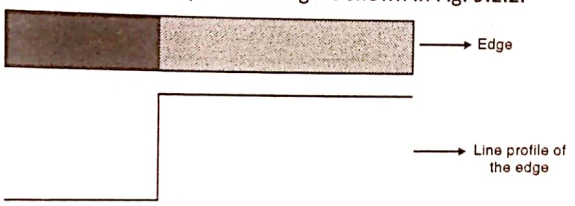


Fig. 9.2.2

- Here the set of pixels that separate the black region from the grey region is called an *edge*. The line profile of such an edge is shown along with edge.
- The image shown has a step edge. Such step edges occur only in artificially generated images such as test patterns. Images obtained from digitization of optical images of real scenes do not possess step edges. Recollect what happens when we digitize an image.
- We first sample the image using the Nyquist condition and then quantize it. Real world signals are not band-limited i.e. $F(u, v) \neq 0$ outside an interval and maximum frequency present in it is infinite. Hence the Nyquist criteria is not met. We therefore pass the original signal through an anti-aliasing filter (which is a low pass filter) to remove the very high frequencies.

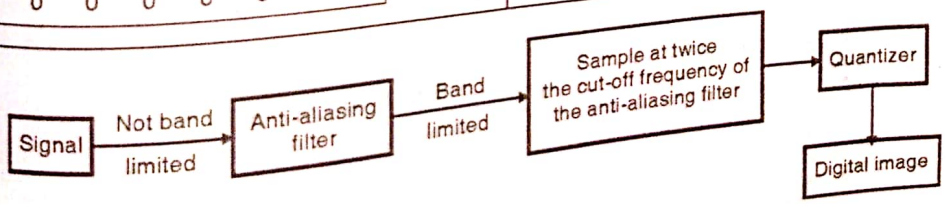


Fig. 9.2.3

- It is this anti-aliasing filter which reduces the slope of the edge. Due to this, real world images when captured through a camera will not have step edges. In practice, edges are modelled as having a ramp like profile.

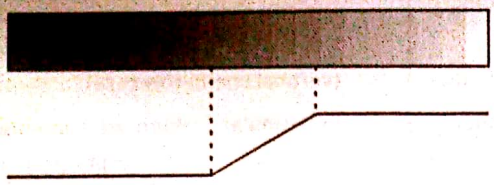


Fig. 9.2.4

Here the slope of the ramp is inversely proportional to the degree of blurring.

Given below are some of the two dimensional, discrete domain edge models.

a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b

Vertical step edge

a	a	a	b	b	b	b	b	b	b
a	a	a	a	b	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	a	b	b	b	b
a	a	a	a	a	a	a	b	b	b

Diagonal step edge

a	a	a	a	a	a	a	a	a	a
a	a	a	a	a	a	a	a	a	a
a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b
a	a	a	a	a	b	b	b	b	b

Corner step edge

a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b

Vertical ramp edge

a	a	c	b	b	b	b	b	b	b
a	a	a	c	b	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	a	c	b	b	b	b
a	a	a	a	a	a	c	b	b	b

Diagonal ramp edge Single pixel transition

a	a	a	a	a	a	a	a	a	a
a	a	a	a	c	c	c	c	c	c
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b

Corner ramp edge

a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b

Vertical ramp edge

a	a	d	e	b	b	b	b	b	b
a	a	a	d	e	b	b	b	b	b
a	a	a	a	d	e	b	b	b	b
a	a	a	a	a	d	e	b	b	b
a	a	a	a	a	a	d	e	b	b

Diagonal ramp edge Smoothed transition

a	a	a	a	a	a	a	a	a	a
a	a	a	a	d	c	c	c	c	c
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b
a	a	a	a	c	b	b	b	b	b

Corner ramp edge

$$c = \frac{a+b}{2}, \quad d = \frac{3a+b}{4}, \quad e = \frac{a+3b}{4}, \quad b > a$$

Fig. 9.2.5

9.3 Detection of Edges

How do these edges come in an image ?

- Variation of scene features, usually brightness, give rise to edges. In other words, edges are representations of the discontinuities of the scene intensity function.
- There could be various reasons such as type of material, surface texture, lighting conditions, etc., which play an important role in forming these discontinuities.
- Our aim in this chapter is to detect these discontinuities as edges. The process of edge detection is carried out by the derivative approach.
- The basic idea of the derivative approach is the computation of the local derivative operator.

Consider the image shown in Fig. 9.3.1.

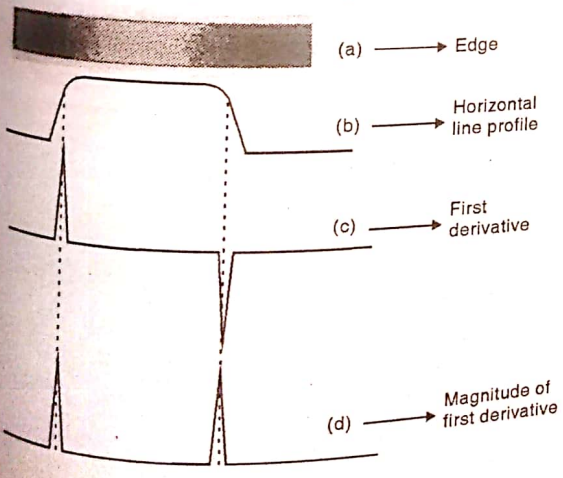


Fig. 9.3.1

Fig. 9.3.1 shows that the first derivative of the grey line profile is positive at the leading edge of a transition, negative at the trailing edge and zero in areas of constant grey levels.

- If we now want to plot the derivative image, we need to take the magnitude of Fig. 9.3.1(c). This gives us Fig. 9.3.1(d). As can be seen, we get non-negative values only at the two edges of the image. We can thus conclude that computing the derivative helps us in detecting the edges.
- Although this discussions has been limited to the 1-D horizontal profile, a similar argument applies to an edge of any orientation in an image.
- The first derivative at any point in an image is obtained by using the magnitude of the gradient at that point.

9.3.1 Computing the Gradient

Let us spend some time understanding the basics of the gradient. This discussion will help us understand edge detection better.

Case 1 : Output is a function of one variable
(**Example :** 1-dimensional signal) y = f(x).

We know that the derivative of y w.r.t. x is

$$\frac{dy}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} = f'(x)$$

The slope is $\frac{dy}{dx}$

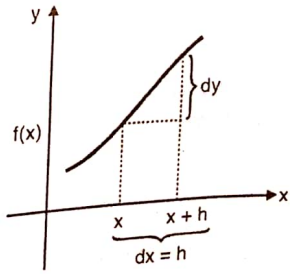


Fig. 9.3.2

Case 2 : Output is a function of two variables (2-dimensional example), f(x, y). In the two variable case, x and y are the variables. Both these variables change to say x + h and y + k. To find the gradient, we work with each separately i.e., we take partial derivatives.

Let us first keep y fixed and alter only x.

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

Here $\frac{\partial f}{\partial x}$ is the rate of change of f w.r.t. x keeping y fixed.

Similarly, keeping x fixed and changing y, we get

$$\frac{\partial f}{\partial y} = \lim_{k \to 0} \frac{f(x, y+k) - f(x, y)}{k}$$

$\frac{\partial f}{\partial y}$ is the rate of change of f w.r.t. y keeping x constant.

Hence the final gradient is

$$\nabla F = \hat{i} \frac{\partial f}{\partial x} + \hat{j} \frac{\partial f}{\partial y} \quad ...(9.3.1)$$

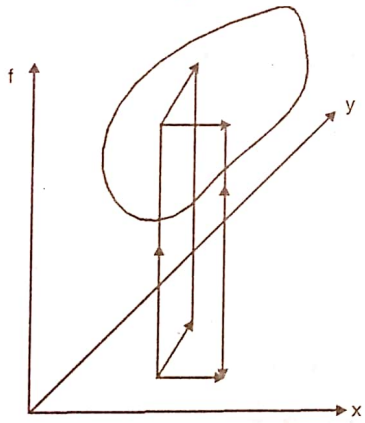


Fig. 9.3.3

Case 3 : Output is a function of three variables (3-dimensional, e.g.. Density at different points in the atmosphere or pressure at different points) (Not useful in image processing)

$$P = f(x, y, z)$$

∴ the three partial derivatives are

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y, z) - f(x, y, z)}{h}$$

$$\frac{\partial f}{\partial y} = \lim_{k \to 0} \frac{f(x, y+k, z) - f(x, y, z)}{k}$$

$$\frac{\partial f}{\partial z} = \lim_{l \to 0} \frac{f(x, y, z+l) - f(x, y, z)}{l}$$

$$\therefore \text{ gradient } \nabla F = \hat{i} \frac{\partial f}{\partial x} + \hat{j} \frac{\partial f}{\partial y} + \hat{k} \frac{\partial f}{\partial z} \quad ...(9.3.2)$$

Case 3 will not be used in this book

Now, magnitude of a vector is given by ,

$$A^2 = A_x^2 + A_y^2$$

$$|A| = \sqrt{A_x^2 + A_y^2}$$

Similarly, magnitude of a 2-D gradient is,

$$|\nabla F| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2} \quad ...(9.3.3)$$

Here $\frac{\partial f}{\partial x}$ = Rate of change of 'f' w.r.t the x-direction

$\frac{\partial f}{\partial y}$ = Rate of change of 'f' w.r.t the y-direction

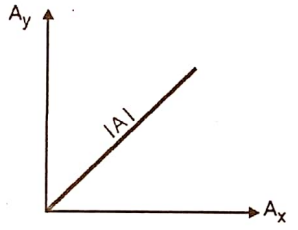
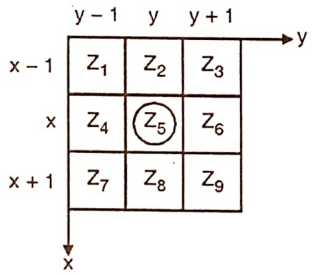


Fig. 9.3.4

9.4 Finding Gradients using Masks

Consider a 3 × 3 neighbourhood with Z_5 as the origin.



We know, $\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$

$$\frac{\partial f}{\partial y} = \lim_{k \to 0} \frac{f(x, y+k) - f(x, y)}{k}$$

In the discrete domain, h = k = 1

$$\therefore \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

and $\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y)$

From the 3 × 3 neighbourhood we have,

$$\frac{\partial f}{\partial x} = Z_8 - Z_5 \quad \text{and} \quad \frac{\partial f}{\partial y} = Z_6 - Z_5$$

Since, $|\nabla F| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$

$\therefore |\nabla F| = [(Z_8 - Z_5)^2 + (Z_6 - Z_5)^2]^{1/2}$

So as to make it easier for implementation,

$$|\nabla F| = |Z_8 - Z_5| + |Z_6 - Z_5|$$

i.e. $|\nabla F| = |Z_5 - Z_8| + |Z_5 - Z_6|$ ...(9.4.1)

This is the first order difference gradient. This can be implemented using two masks.

$|Z_5 - Z_8| =$

1	0
−1	0

→ mask 1

and $|Z_5 - Z_6| =$

1	−1
0	0

→ mask 2

This is known as the Ordinary operator.

Steps to compute the gradient of an image are as follows

(1) Convolve the original image with mask 1. This gives us the gradient along the x-direction.

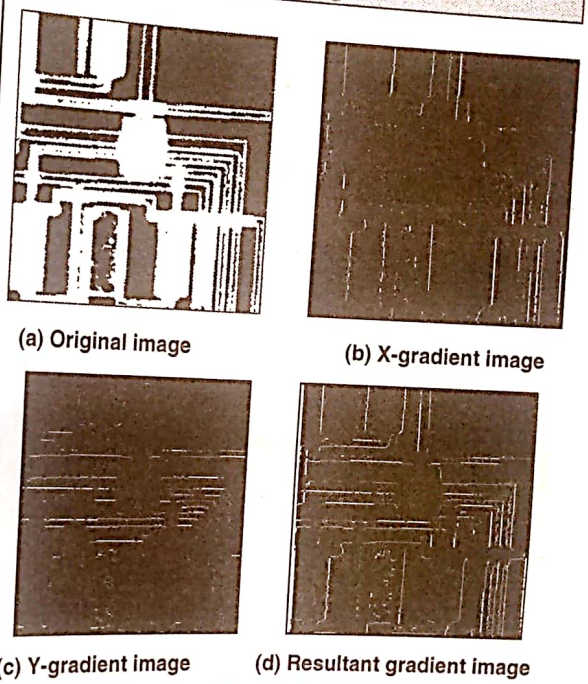
(2) Convolve the original image with mask 2. This gives us the gradient along the y-direction.

(3) Add the results of 1 and 2.

The advantage of this prolonged method is that, we can see the effect of each mask separately.

MATLAB code for ordinary operator

```
%% Ordinary operator %%
clear all
clc
aa = imread('test.jpg');
a = double(aa);
[row col] = size(a);
w1=[1 0;-1 0];
w2=[1 -1; 0 0];
for x=2:1:row-1;
for y=2:1:col-1;
a1(x,y)=w1(1)*a(x,y)+w1(2)*a(x,y+1)+w1(3)*a(x+1,y)
+w1(4)*a(x+1,y+1);
a2(x,y)=w2(1)*a(x,y)+w2(2)*a(x,y+1)+w2(3)*a(x+1,y)
+w2(4)*a(x+1,y+1);
end
a3=a1+a2;   %% To find the resultant gradient image
figure(1)
imshow(uint8(a1)) %% The x-gradient image, you might need normalization
figure(2)
imshow(uint8(a2)) %% The y-gradient image, you might need normalization
figure(3)
imshow(uint8(a3)) %% Final image
```



(a) Original image (b) X-gradient image

(c) Y-gradient image (d) Resultant gradient image

Fig. 9.4.1

There is another direct method of implementing the ordinary operator practically. It gives results that are almost similar.

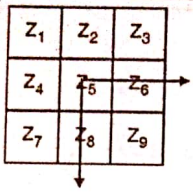
(1) Add masks 1 and 2

1	0	+	1	−1	=	2	−1
−1	0		0	0		−1	0

(2) Convolve the original image with this resultant mask to get the gradient image.

9.4.1 Roberts Mask

Robert L.G in 1965 published a paper in which he stated that better results could be obtained if cross differences were taken instead of the straight difference.

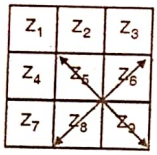


Ordinary masks

$$|\nabla F| = |Z_5 - Z_8| + |Z_5 - Z_6|$$

Roberts masks

$$|\nabla F| = |Z_5 - Z_9| + |Z_6 - Z_8| \quad \ldots(9.4.2)$$



∴ Roberts masks are

1	0
0	−1

0	1
−1	0

The results of the Roberts mask are shown along with the program.

The sum of the two Roberts masks is

1	1
−1	−1

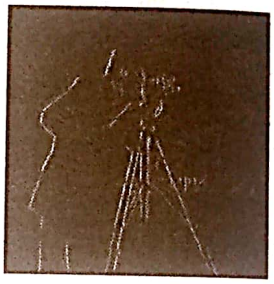
MATLAB code for Roberts operator

```
% Roberts operator %%
clear all
clc
aa=imread('cameraman.jpg');
a=double(aa);
[row col]=size(a);
w1= [1 0; 0 -1];
w2= [0 1; -1, 0];
for x=2:1:row-1;
for y=2:1:col-1;
a1(x,y)=w1(1)*a(x,y)+w1(2)*a(x,y+1)+w1(3)*a(x+1,y)
+w1(4)*a(x+1,y+1);
a2(x,y)=w2(1)*a(x,y)+w2(2)*a(x,y+1)+w2(3)*a(x+1,y)
+w2(4)*a(x+1,y+1);
        end
end
a3=a1+a2;      %% Final Gradient Image %%
figure(1)
imshow(uint8(a1)) %% The x-gradient image, you might
need normalization
figure(2)
imshow(uint8(a2)) %% The y-gradient image, you might
need normalization
figure(3)
imshow(uint8(a3)) %% Final image %%
```

(a) Original image

(b) X-gradient image

(c) Y-gradient image



(d) Resultant image

Fig. 9.4.2

9.4.2 Prewitts and Sobel Operators

Roberts mask, as is evident, is an even sized mask (2 × 2). Masks of even sizes are awkward to implement. Note that the differential along the diagonals of a 2 × 2 mask is used and the edge value after the convolution corresponds to the central point $\left(r-\frac{1}{2}, c-\frac{1}{2}\right)$.

This problem can be avoided using 3 × 3 masks.

Prewitts Operator

Prewitt J.M.S in his paper "Object Enhancement and Extraction" in 1970 came up with a 3 × 3 mask. The Prewitt operator, as is now called, while approximating the first derivative, assigns similar weights to all the neighbours of the candidate pixel whose edge strength is being calculated.

$$\nabla F \approx \underbrace{|(Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3)|}_{\text{x-gradient}} + \underbrace{|(Z_3 + Z_6 + Z_9) - (Z_1 + Z_4 + Z_7)|}_{\text{y-gradient}} \quad \ldots(9.4.3)$$

From this equation, the masks that we obtain are

These masks are known as the Prewitts masks.

-1	-1	-1
0	0	0
1	1	1

F_x

-1	0	1
-1	0	1
-1	0	1

F_y

Sobel Operator

Duda R.O and Hart P.E in 1973 published a paper "Pattern Classification and Scene Analysis" in which they used a new operator known as the Sobel Operator.

In the Sobel Operator higher weights are assigned to the pixels close to the candidate pixels.

$$\nabla F = |(Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)| + |(Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)|$$

x-gradient y-gradient

...(9.4.4)

From this equation, the masks that we obtain are

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

F_x

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

F_y

Implementation of Prewitt and Sobel Operators

(1) Convolve original image with the Fx mask to get the x-gradient image.

(2) Convolve original image with Fy mask to get the y-gradient image.

(3) Add the results of step (1) and (2)

As mentioned earlier, the advantage of using three steps is that we can see the results of the Fx mask and the Fy mask separately.

A shorter way of doing this which would give almost similar results is

(i) Add the Fx and Fy masks first.

(ii) Convolve the new mask with the original image.

Prewitt mask

-1	-1	-1
0	0	0
1	1	1

F_x +

-1	0	1
-1	0	1
-1	0	1

F_y =

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

$F_x + F_y$

Sobel mask

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

F_x +

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

F_y =

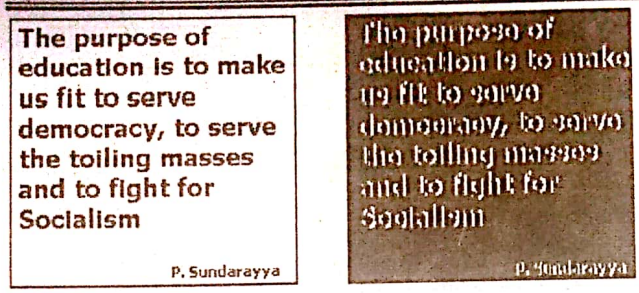
-2	-2	0
-2	0	2
0	2	2

$F_x + F_y$

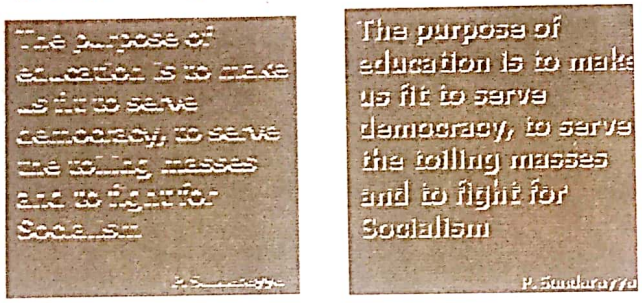
MATLAB code for Prewitts operator

```
%% Prewitts operator %%
clear all
clc
aa=imread('test.tif');
a=double(aa);
[row col]=size(a);
w2=[-1  0  1;-1  0  1;-1  0  1];
w1=[-1  -1  -1; 0  0  0; 1  1  1];

for x=2:1:row-1;
for y=2:1:col-1;
a1(x,y)=w1(1)*a(x-1,y-1)+w1(2)*a(x-1,y)
+w1(3)*a(x-1,y+1)+w1(4)* ...
a(x,y-1)+w1(5)*a(x,y)+w1(6)*a(x,y+1)
+w1(7)*a(x+1,y-1)+w1(8)* ...
a(x+1,y)+w1(9)*a(x+1,y+1);
a2(x,y)=w2(1)*a(x-1,y-1)+w2(2)*a(x-1,y)
+w2(3)*a(x-1,y+1)+w2(4)* ...
a(x,y-1)+w2(5)*a(x,y)+w2(6)*a(x,y+1)
+w2(7)*a(x+1,y-1)+w2(8)* ...
a(x+1,y)+w2(9)*a(x+1,y+1);
    end
end
a3=a1+a2;        %% The final gradient value %%
figure(1)
imshow(uint8(a1))%% The x-gradient image,
Normalisation might be required
figure(2)
imshow(uint8(a2))%% The y-gradient image %%
figure(3)
imshow(uint8(a3))%% Final gradient image,
Normalisation might be required
```



(a) Original image (b) X-gradient image

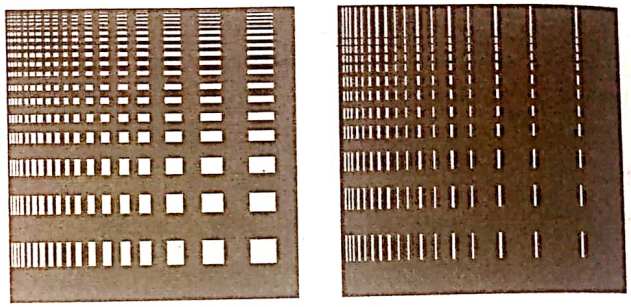


(c) Y-gradient image (d) Resultant gradient image

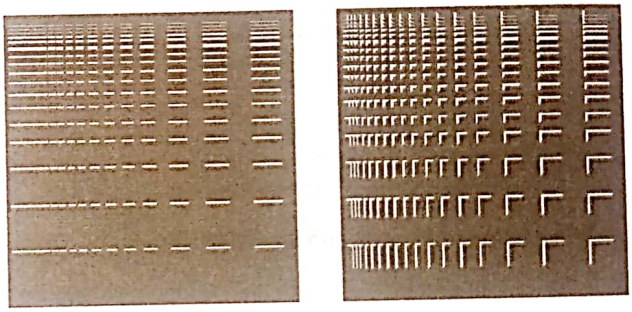
Fig. 9.4.3

MATLAB code for Sobel operator

```
%% Sobel operator %%
clear all
clc
aa=imread('warne.tif');
a=double(aa);
[row col]=size(a);
w1=[-1  -2  -1; 0 0  0;      1   2   1 ];
w2=[-1   0   1; -2 0  2;     -1  0   1 ];
for x=2:1:row-1;
for y=2:1:col-1;
a1(x,y)=w1(1)*a(x-1,y-1)+w1(2)*a(x-1,y)
+w1(3)*a(x-1,y+1)+w1(4)* ...
a(x,y-1)+w1(5)*a(x,y)+w1(6)*a(x,y+1)
+w1(7)*a(x+1,y-1)+w1(8)* ...
a(x+1,y)+w1(9)*a(x+1,y+1);
a2(x,y)=w2(1)*a(x-1,y-1)+w2(2)*a(x-1,y)
+w2(3)*a(x-1,y+1)+w2(4)* ...
a(x,y-1)+w2(5)*a(x,y)+w2(6)*a(x,y+1)
+w2(7)*a(x+1,y-1)+w2(8)*
...a(x+1,y)+w2(9)*a(x+1,y+1);
        end
end
    a3=a1+a2;      %% Final gradient values %%
figure(1)
imshow(uint8(a1)) %% The x-gradient image,
Normalisation might be required
figure(2)
imshow(uint8(a2)) %% The y-gradient image,
Normalisation might be required
figure(3)
imshow(uint8(a3)) %% Final gradient image,
Normalisation might be required
```



(a) Original image (b) X-gradient image



(c) Y-gradient image (d) Resultant gradient image

Fig. 9.4.4

Both results can be obtained using absolute values.

Have you noticed one thing that is common to all the edge detection masks that we have studied so far ?

Let us draw all the masks together.

1	0
−1	0

1	−1
0	0

1	0
0	−1

0	1
−1	0

Ordinary operator Robert's operator

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Prewitt mask Sobel mask

Sum of the coefficients of each of these masks is zero !!

- This is a very important property to note. Edges are abrupt discontinuities in the gray levels and hence are high frequency regions. Since the sum of the coefficients of all these masks is zero, they eliminate all the low frequency components of the image i.e., when these masks are placed on low frequency regions, the output is zero. Hence these masks give edges without any low frequency regions in the final output image.
- Derivative filters (Gradient filters), as the name suggests, calculate the gradient of the image.
- Since noise is also high frequency, the derivative of the noise terms will always be large values and hence derivative filters are very sensitive to noise.
- There is a huge advantage in using Prewitt and Sobel masks for edge detection. Both these operators provide a smoothing effect along with providing differentiation.
- Hence Prewitt and Sobel operators perform well even when the image is noisy because they smoothen the noise !!

How do they achieve this ?

Let us consider the Prewitt operator F_x and F_y

-1	-1	-1
0	0	0
1	1	1

and

-1	0	1
-1	0	1
-1	0	1

These operators can be factored into the successive application of two simpler operators.

i.e.

-1	-1	-1
0	0	0
1	1	1

$=$

-1
0
1

$\times$ [1 1 1]

and

-1	0	1
-1	0	0
-1	0	1

$=$

1
1
1

$\times$ [-1 0 1]

In this [1 1 1] is a low pass or a smoothing operator, while [-1 0 1] is a high pass operator.

- Hence the Prewitt operator performs uniform smoothing in one direction with edge detection in the perpendicular direction.
- In a similar manner, we can split up the Sobel operator into smaller and simpler operators.

i.e

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

$=$

-1
0
1

$\times$ [1 2 1]

HP LP

and

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

$=$

1
2
1

$\times$ [-1 0 1]

LP HP

- One can verify this by comparing the results of a Prewitt/ Sobel filter with that of standard highpass filter on an image corrupted by Gaussian noise. (Use **imnoise** command to corrupt the image).

9.4.3 Compass Operators

It is seen that edges in the horizontal as well as in the vertical direction are enhanced when Prewitt's or Sobel's operator is used. There are applications, in which we need edges in all the directions. A simple method would be to rotate the Prewitts or Sobel's mask in all the possible directions.

Considering a Prewitts operator

-1	-1	-1
0	0	0
1	1	1

We move this mask in the anticlockwise direction to get all other masks

-1	-1	-1
0	0	0
1	1	1

-1	-1	0
-1	0	1
0	1	1

-1	0	1
-1	0	1
-1	0	1

0	1	1
-1	0	1
-1	-1	0

1	1	1
0	0	0
-1	-1	-1

1	1	0
1	0	-1
0	-1	-1

1	0	-1
1	0	-1
1	0	-1

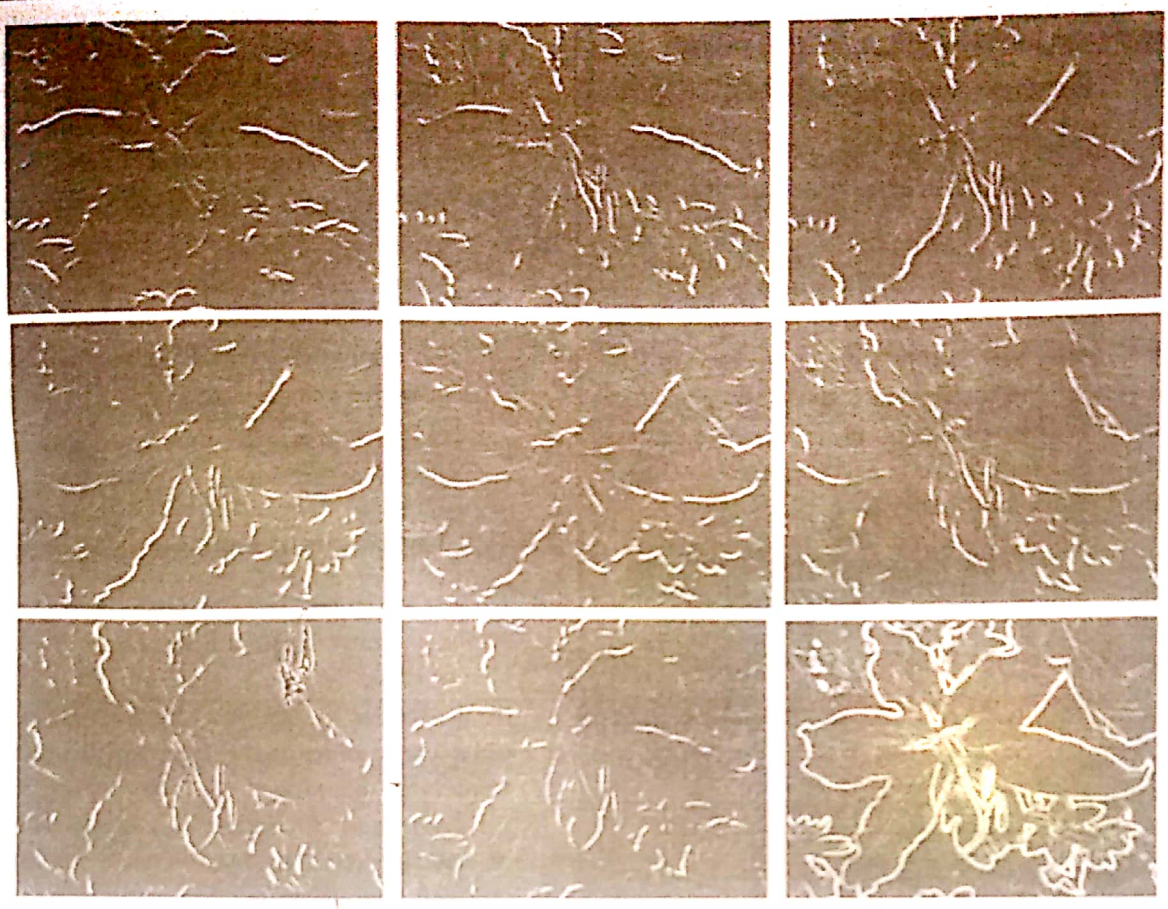
0	-1	-1
1	0	-1
1	1	0

-1	-1	-1
0	0	0
1	1	1

This operator is known as the compass operator and is very useful for detecting weak edges.

Compass operators can also be implemented using the Sobel operator.

```
%% Program of compass operator %%
clear all, clc
aa=imread('lily.tif');
I=double(aa);
s=size(I);
a=[-1 -1 -1; 0 0 0; 1 1 1]'
b=[a(1,2) a(1,3) a(2,3); a(1,1) a(2,2) a(3,3); a(2,1) a(3,1)
a(3,2)]
c=[b(1,2) b(1,3) b(2,3); b(1,1) b(2,2) b(3,3); b(2,1) b(3,1)
b(3,2)]
d=[c(1,2) c(1,3) c(2,3); c(1,1) c(2,2) c(3,3); c(2,1) c(3,1)
c(3,2)]
e=[d(1,2) d(1,3) d(2,3); d(1,1) d(2,2) d(3,3); d(2,1) d(3,1)
d(3,2)]
f=[e(1,2) e(1,3) e(2,3); e(1,1) e(2,2) e(3,3); e(2,1) e(3,1)
e(3,2)]
g=[f(1,2) f(1,3) f(2,3); f(1,1) f(2,2) f(3,3); f(2,1) f(3,1)
f(3,2)]
h=[g(1,2) g(1,3) g(2,3); g(1,1) g(2,2) g(3,3); g(2,1) g(3,1)
g(3,2)]
for x=2:1:s(1)-1;
for y=2:1:s(2)-1;
A(x,y)=[a(1)*I(x-1,y-1)+a(2)*I(x-1,y)
+a(3)*I(x-1,y+1)+a(4)*I(x,y-1)+ ...
a(5)*I(x,y)+a(6)*I(x,y+1)+a(7)*I(x+1,y-1)
+a(8)*I(x+1,y)+a(9)*I(x+1,y+1)];
B(x,y)=[b(1)*I(x-1,y-1)+b(2)*I(x-1,y)
+b(3)*I(x-1,y+1)+b(4)*I(x,y-1) ...
+b(5)*I(x,y)+b(6)*I(x,y+1)+b(7)*I(x+1,y-1)
b(8)*I(x+1,y)+b(9)*I(x+1,y+1)];
C(x,y)=[c(1)*I(x-1,y-1)+c(2)*I(x-1,y)
+c(3)*I(x-1,y+1)+c(4)*I(x,y-1) ...
+c(5)*I(x,y)+c(6)*I(x,y+1)+c(7)*I(x+1,y-1)
+c(8)*I(x+1,y)+c(9)*I(x+1,y+1)];
D(x,y)=[d(1)*I(x-1,y-1)+d(2)*I(x-1,y)
+d(3)*I(x-1,y+1)+d(4)*I(x,y-1)+ ...
d(5)*I(x,y)+d(6)*I(x,y+1)+d(7)*I(x+1,y-1)
+d(8)*I(x+1,y)+d(9)*I(x+1,y+1)];
E(x,y)=[e(1)*I(x-1,y-1)+e(2)*I(x-1,y)
+e(3)*I(x-1,y+1)+e(4)*I(x,y-1) ...
+e(5)*I(x,y)+e(6)*I(x,y+1)+e(7)*I(x+1,y-1)
+e(8)*I(x+1, y)+e(9)*I(x+1,y+1)];
F(x,y)=[f(1)*I(x-1,y-1)+f(2)*I(x-1,y)
+f(3)*I(x-1,y+1)+f(4)*I(x,y-1) ...
+f(5)*I(x,y)+f(6)*I(x,y+1)+f(7)*I(x+1,y-1)
+f(8)*I(x+1,y)+f(9)*I(x+1,y+1)];
G(x,y)=[g(1)*I(x-1,y-1)+g(2)*I(x-1,y)
+g(3)*I(x-1,y+1)+g(4)*I(x,y-1) ...
+g(5)*I(x,y)+g(6)*I(x,y+1)+g(7)*I(x+1,y-1)
+g(8)*I(x+1,y)+g(9)*I(x+1, y+1)];
H(x,y)=[h(1)*I(x-1,y-1)+h(2)*I(x-1,y)
+h(3)*I(x-1,y+1)+h(4)*I(x,y-1) ...
+h(5)*I(x,y)+h(6)*I(x,y+1)+h(7)*I(x+1,y-
1)+h(8)*I(x+1,y)+h(9)*I(x+1, y+1)];
        end
end
W=max(max(max(max(max(max(A,B),C),D),E),F),G),
H); %% To calculate maximum
%% Plotting %%
figure(1),imshow(uint8(A)),figure(2),imshow(uint8(B))
figure(3),imshow(uint8(C)),figure(4),imshow(uint8(D))
figure(5),imshow(uint8(E)),figure(6),imshow(uint8(F))
figure(7),imshow(uint8(G)),figure(8),imshow(uint8(H))
figure(9),imshow(uint8(W)),figure(10),imshow(uint8(I))
%% Normalisation may be required %%
```



Resultant image

Fig. 9.4.5 : Compass operated images

9.5 Image Segmentation using the Second Derivative - the Laplacian

- We have already seen the effects of using the first derivative of the image. It was shown that the first derivative does enhance the edges of the image. We now try to find out the effects of computing the second derivative on the edges.

We know,

$$\nabla F = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}$$

$$\therefore \quad \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y) \quad \text{and}$$

$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y)$$

The second derivative is given by

$$\nabla^2 F = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Where $\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$ and

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 F = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\therefore \quad |\nabla^2 F| = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)] \quad ...(9.5.1)$$

Considering the 3 × 3 neighbourhood

	y − 1	y	y + 1
x − 1	Z_1	Z_2	Z_3
x	Z_4	Z_5	Z_6
x + 1	Z_7	Z_8	Z_9

This equation in the discrete form reduces to,

$$|\nabla^2 F| = [Z_2 + Z_4 + Z_6 + Z_8 - 4Z_5]$$

- This equation can be implemented using a mask i.e.

0	1	0
1	-4	1
0	1	0

- This is known as the Laplacian operator. Some books reverse the signs of the coefficients of the mask i.e.,

0	−1	0
−1	4	−1
0	−1	0

- There might be cases when, we would need to add higher weights at the center pixel (>4). While doing this we must make sure that the sum of the coefficients of the mask is zero since edges are high frequency components. Hence if we increase the weights at the centre, we would also have to change the values of the coefficients at the borders.

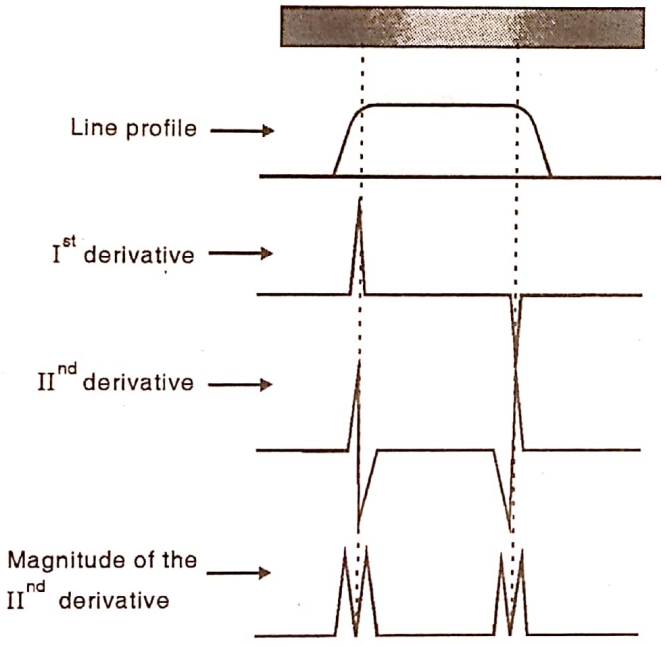


Fig. 9.5.1

- One important thing to note about the Laplacian mask is that unlike the Sobel and Prewitt operator, they are isotropic filters. i.e., their response is independent of the direction of the discontinuties in the image. In other words, isotropic filters are rotation invarient. i.e., rotating the image and then applying the mask gives the same result as applying the mask to the image first and then rotating the result.
- Now that we have the second derivative Laplacian mask with us, can we use it directly on the image ? the answer is No. The Laplacian is not generally used in its original form as an edge detector for mainly two reasons.

(1) Since the Laplacian is a second derivative filter, it is very sensitive to noise (much more than the first derivative). Hence in an image, if there is any noise present, the Laplacian gives very large values and ruins the entire image.

(2) The magnitude of the Laplacian produces double edges, which is an undesireable effect. Let us explain what we mean by double edges. Consider a strip of image shown in Fig. 9.5.1. It is seen that the magnitude of the 2nd derivative produces two peaks for a single edge.

The zero crossing property of the Laplacian is used to detect edges.

9.5.1 Laplacian of Gaussian Operator

- As stated earlier, the Laplacian mask evokes very strong response to stray noise pixels. Hence if some kind of noise cleaning is done prior to the application of the Laplacian operator, better results could be obtained. In practice the Laplacian is preceded by a smoothing operation (Gaussian smoothing). The resultant algorithm is commonly known as a Laplacian of Gaussian (LoG) or Marr-Hildreth operator. (Marr D.C. and Hildreth presented a paper "Theory of Edge Detection" in 1980 which explains in this concept)

Let us explain the mathematics of this.

- What we just said was that, we first smoothen the image using a Gaussian function (Remember low pass Gaussian filter?) and then take the Laplacian (second derivative of the image). A simple method to implement this practically, is to combine the two i.e. combine the action of a gaussian function and a second derivative function.

Consider the gaussian function

$$h(x, y) = e^{-\left(\frac{x^2 + y^2}{2\sigma^2}\right)} \quad \ldots(9.5.2)$$

where σ is the standard deviation and x and y are the two variables (spatial coordinates). It is σ that determines the degree of blurring.

$$\text{Let } x^2 + y^2 = r^2$$

$$\therefore \quad h(r) = e^{-\left(\frac{r^2}{2\sigma^2}\right)}$$

- The Laplacian of h i.e., the second derivative of h with respect to r is now calculated. We first calculate the first derivative w.r.t. r

$$\frac{\partial}{\partial r} h(r) = \frac{\partial}{\partial r} e^{-r^2/2\sigma^2} = e^{-r^2/2\sigma^2} \cdot \frac{\partial}{\partial r}\left(\frac{-r^2}{2\sigma^2}\right)$$

$$\frac{\partial}{\partial r} h(r) = e^{-r^2/2\sigma^2} \cdot \left(\frac{-2r}{2\sigma^2}\right) = e^{-r^2/2\sigma^2} \cdot \left(\frac{-r}{\sigma^2}\right)$$

Now taking the second derivative, we get

$$\frac{\partial^2}{\partial r^2} h(r) = \frac{\partial}{\partial r}\left(\frac{-r}{\sigma^2} e^{-r^2/2\sigma^2}\right) = \frac{-1}{\sigma^2}\frac{\partial}{\partial r}\left(re^{-r^2/2\sigma^2}\right)$$

$$\frac{\partial^2}{\partial r^2} h(r) = \frac{-1}{\sigma^2}\left(re^{-r^2/2\sigma^2}\left(\frac{-r}{\sigma^2}\right) + e^{-r^2/2\sigma^2}\right)$$

$$= \frac{1}{\sigma^2}\left(\frac{r^2 e^{-r^2/2\sigma^2}}{\sigma^2} - e^{-r^2/2\sigma^2}\right)$$

$$= \frac{1}{\sigma^2} e^{-r^2/2\sigma^2}\left(\frac{r^2}{\sigma^2} - 1\right)$$

$$\frac{\partial^2}{\partial r^2} h(r) = \frac{r^2 - \sigma^2}{\sigma^4} e^{-r^2/2\sigma^2}$$

$$\therefore \nabla^2 h = \left(\frac{r^2 - \sigma^2}{\sigma^4}\right) e^{-r^2/2\sigma^2} \quad \ldots(9.5.3)$$

- This is the Laplacian of a Gaussian (LoG). $\nabla^2 h$ can be easily plotted using MATLAB. Due to the shape of the LoG, it commonly referred to as the Mexican Hat function.

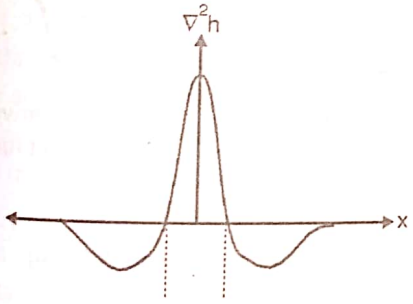


Fig. 9.5.2

- It can be seen that $\nabla^2 h$ has a large positive value at the centre and negative values at the peripheries.

We can approximate this by using a 5 × 5 mask as shown.

- Remember, this mask is not unique. We could use any values that approximate the shape of the LoG.
- One thing that needs to be kept in mind is that since our intention is to enhance the edges, the sum of coefficients of the mask should be zero.

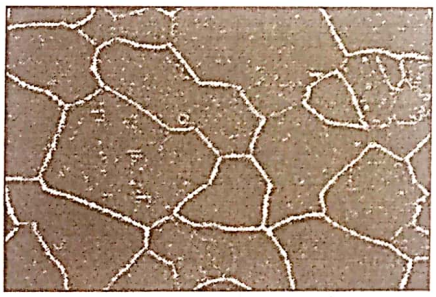
0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

MATLAB code for obtaining Laplacian of a Gaussian image

```
clear all
clc
a=imread('alumgrns.tif');
a=double(a);
   [row col]=size(a);
   lap=[0 1 0;1 -4 1;0 1 0];   %% Laplacian mask
   log=[0 0 -1 0 0;0 -1 -2 -1 0;-1 -2 16 -2 -1;0 -1 -2 -1
0;0 0 -1 0 0];%% LOG
   for x=3:1:row-2;
   for y=3:1:col-2;
%% Calculating only the Laplacian %%
c(x,y)=[lap(1)*a(x-1,y-1)+lap(2)*a(x-1,y)
+lap(3)*a(x-1,y+1)+lap(4)*a(x,y-1)+...
   lap(5)*a(x,y)+lap(6)*a(x,y+1)
+lap(7)*(x+1,y-1)+lop(8)*a(x+1,y)+...
   lap(9)*a(x+1,y+1)];
%% Calculating the Laplacian of a Gaussian %%
dee(x,y)=log(1)*a(x-2,y-2)+log(2)*a(x-2,y-1)
+log(3)*a(x-2,y)+log(4)*a(x-2,y+1)+...
   log(5)*a(x-2,y+2)+log(6)*a(x-1,y-2)
+log(7)*a(x-1,y-1)+log(8)*a(x-1,y)+...
   log(9)*a(x-1,y+1)+log(10)*a(x-1,y+2)
+log(11)*a(x,y 2)+log(12)*a(x,y-1)+...
   log(13)*a(x,y)+log(14)*a(x,y+1)+log(15)*a(x,y+2)
+log(16)*a(x+1,y-2)+...
   log(17)*a(x+1,y-1)+log(18)*a(x+1,y)
+log(19)*a(x+1, y+1)+log(20)*a(x+1,y+2)+...
   log(21)*a(x+2,y-2)+log(22)*a(x+2,y-
1)+log(23)*a(x+2,y)+log(24)*a(x+2,y+1)+...
   log(25)*a(x+2,y+2);
   end
end
%% Thresholding the output of the LoG operator %
for x=1:1:row-2
   for y=1:1:col-2
      if dee(x,y)<40
         deep(x,y)=0;
      else
         deep(x,y)=255
   end
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%ZERO CROSSINGS%%%%%%%%%%%%%
for x=2:1:row-2
   for y=2:1:col-2
      a1=deep(x,y);
      a2=deep(x-1,y-1);
```

```
        if a1==0 && a2==0
        deepa(x,y)=0;
        deepa(x-1,y-1)=0;
    else if a1==255 && a2==255
            deepa(x,y)=0;
            deepa(x-1,y-1)=0;
        else
            deepa(x,y)=255;
            deepa(x-1,y-1)=255;
            end
            end
        end
        end
end
figure(1),imshow(uint8(a))        %% Original image
figure(2),imshow(uint8(c))        %% Only Laplacian
figure(3),imshow(uint8(dee))      %% LoG
figure(4),imshow(uint8(deep))     %% LoG+Threshold
figure(5),imshow(uint8(deepa))    %% Zero crossings
```

(a) Original image



(b) Image using LoG operator

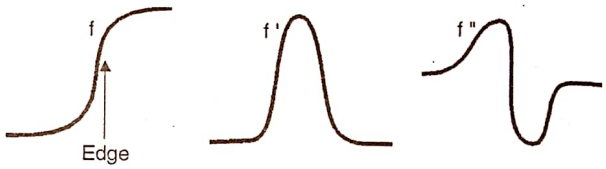
(c) Zero crossing image

Fig. 9.5.3

9.5.2 Canny Edge Detector

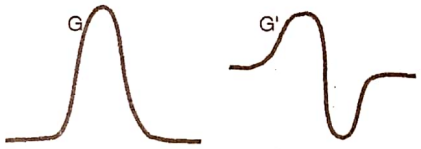
- This is the last of the edge detectors that we would study here. Canny operator (Named after J. K. Canny) is a first derivative edge detector coupled with noise cleaning. Canny has, over the years, become one of the most popular derivative operators. Like in the LoG, a Gaussian function is used to smoothen the noise.
- In the Canny edge detector, we first smoothen the image using a Gaussian low pass filter and then take the first derivative.

Consider the Fig. 9.5.4,



(a) (b) (c)

(a) A ramp edge, (b) First derivative of ramp edge, (c) Second derivative of ramp edge



(d) Gaussian function, **(e) First derivative of Gaussian function**

Fig. 9.5.4

Compare Fig. 9.5.4(c) and Fig. 9.5.4(e). Notice the similarity between the shape of the first derivative of the Gaussian and the second derivative of the ramp edge.

Hence the derivative of the bell shape of the Gaussian function approximates the second derivative or zero crossing operator (LoG).

9.6 Connectivity

MU : Dec. 2016, Dec. 2017

Q. Write short note on : 4, 8 and m-connectivity.
(Dec. 2016, Dec. 2017, 5 Marks)

- Consider a pixel p(x, y). This pixel p has two horizontal and two vertical neighbours which share a surface with p(x, y).
- This set of four pixels is known as the 4-neighbourhood of pixel p(x, y), where each pixel is said to be at a unit distance from p(x, y). This set of pixels is denoted as $N_4(p)$. In a similar manner, apart from these 4-neighbours, there are another four diagonal pixels which touch p(x, y) at the corners $N_D(p)$.

	p(x – 1, y)	
p(x, y – 1)	p(x, y)	p(x, y + 1)
	p(x + 1, y)	

$N_4(p)$

- This set of eight pixels (The 4-neighbours as well as the diagonal pixels) is called the 8-neighbourhood of pixel p(x, y) and is denoted as $N_8(p)$

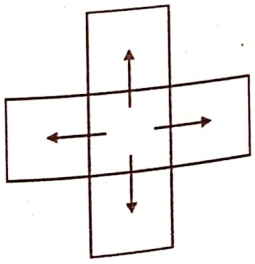
$\therefore N_8(p) = N_4(p) + N_D(p)$.

- Connectivity is an important concept that is used in segmentation. When we started off with this chapter, it was stated that segmentation is used for machine vision.

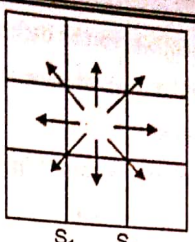
p(x –1, y – 1)		p(x –1, y + 1)
	p(x, y)	
p(x + 1, y – 1)		p(x + 1, y + 1)

$N_D(p)$

- In segmentation, the machine (computer) scans the image to form different regions. This scanning of the image by the machine is based on the connectivity that the user or the code specifies. To establish if two pixels are connected, it must be determined whether they are neighbours (4-neighbours or 8-neighbours) and if their grey levels satisfy a specified criteria.
- Two regions that touch only at a corner can be considered to be a single region or two distinct regions : How they are considered depends on the definition of connectivity used ?
- Two pixels p and q with some common criteria are said to be 4-connected if they share a side. i.e., two pixels p and q with some common criteria are 4-connected if q is in the set $N_4(p)$.



- Two pixels p and q with some common criteria are said to be 8-connected if they share either a side or a corner. i.e., two pixels p and q with some common criteria are 8-connected if q is in the set $N_8(p)$.



S_1 S_2

0	0
1 (p)	0
0	1 (q)

- Let us take an example. Given the criteria that p = q = 1 for the region to be connected, check whether the two region S_1 and S_2 are connected.
- Even though p and q are both equal to one, these two regions will not be shown connected, if we use a 4-connectivity algorithm. This is because, in the 4-connectivity algorithm, we only check the 4-neighbours and see if any of these four neighbours has a pixel with the same value 1, in this case
- If we use an 8-connectivity algorithm, S_1 and S_2 would be shown as a single connected region. Apart from 4-connectivity and 8-connectivity, we also have m-adjacent or m-connectivity (mixed connectivity). It is used to eliminate the double paths that arise when we use 8-connectivity.

Let us take an example.

Consider the image shown.

0	2	2
0	2	0
0	0	2

Here the condition used is that pixels belong to a region if their value is 2.

When we use 8-connectivity here, we get

0	2	2
0	2	0
0	0	2

We notice that the pixel at the right hand upper corner gets connected twice due to 8-connectivity. The ambiguity is removed by using m-connectivity.

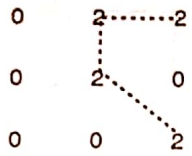
Hence m-connectivity can be defined as follows.

Two pixels p and q are said to be m-connected if

(i) q is in $N_4(p)$ or

(ii) q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ is empty

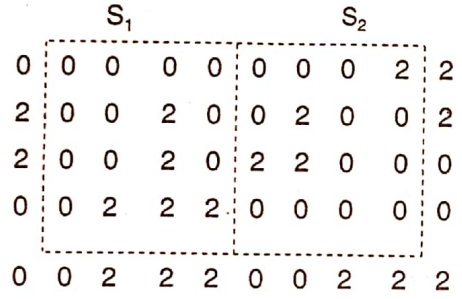
Using this definition we get



Any connectivity can be used as long as one is consistent. Often 8-connectivity yields results that lie closer to one's intuition.

9.6.1 Solved Example on Connectivity

Ex. 9.6.1 : Consider two image subsets S_1 and S_2

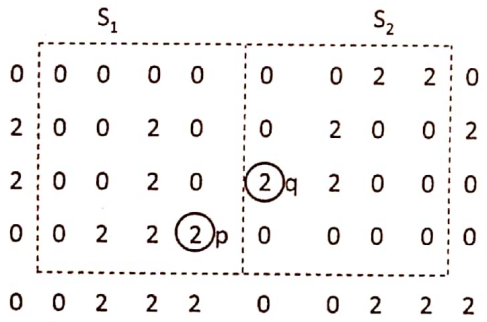


For V = {2}, determine whether S_1 and S_2 are

(a) 4-connected (b) 8-connected (c) m-connected

Soln. :

V = {2} simply means that two pixels p and q are connected if their values are equal to 2.



(a) S_1 and S_2 are not 4-connected because q is not in the set of $N_4(p)$.

(b) S_1 and S_2 are eight connected because q is in the set of $N_8(p)$.

(c) S_1 and S_2 are m-connected because

(i) q is in the set $N_D(p)$ and

(ii) the set $N_4(p) \cap N_4(q)$ is empty

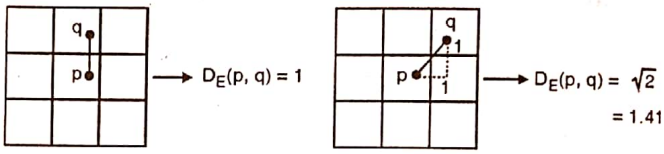
9.7 Distance Transform

Though not directly related to segmentation, Distance transform is introduced here as it is related to connectivity. Distance transforms are required in image processing projects where measurements have to be made. The distance transform provides a measure of the separation of points in an image.

(1) Euclidean Distance : Euclidean distance is the straight line distance between two pixels. If p and q are the two pixels with coordinates (x_1, y_1) and (x_2, y_2), then

$$D_E = [(x_1 - x_2)^2 + (y_1 - y_2)^2]^{1/2}$$

Diagrammatically it can be shown as



∴ for a 3 × 3 region, we have the Euclidian distance as

(2) City Block Distance (D_4 Distance) : For the same points $p(x_1, y_1)$ and $q(x_2, y_2)$, the city block distance is defined as

$$D_{CITY}(p, q) = D_4(p, q) = |x_1 - x_2| + |y_1 - y_2|$$

The city block distance measures the path between the pixels based on a 4-connected neighbourhood. Pixels whose edges touch are 1 unit apart and pixels touching diagonally are 2 units apart.

2	1	2
1	0	1
2	1	2

(3) Chess Board Distance (D_8 Distance) : For pixels $p(x_1, y_1)$ and $q(x_2, y_2)$, the chess board distance is defined as

$$D_{CHESS}(p, q) = D_8(p, q) = \max(|x_1 - x_2|, |y_1 - y_2|)$$

The chessboard distance measures the path between the pixels based on a 8-connected neighbourhood. Pixels whose edges or corners touch are 1 unit apart.

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

(d) D_m Distance (m-adjacency Distance) : This distance measure is based on m-adjacency. Pixels p and q are m-adjacent if

(a) q is in $N_4(p)$ or

(b) q is in $N_D(p)$ and $N_4(p) \cap N_4(q)$ is empty.

9.7.1 Solved Examples on Distance Transform

Ex. 9.7.1 : Measure the D_m distance between p and q for the following images.

Soln. :

0	0	1 q
0	1	0
1 p	0	0

→ The path from p to q is shown by dotted line
$D_m = 2$

0	0	1 q
1	1	0
1 p	0	0

→ Since we consider m-connectivity, the path that needs to be taken is shown by dotted line
$\therefore D_m = 3$

0	1 r	1 q
1	1	0
1 p	0	0

→ The path taken from p to q under m-connectivity is shown
$\therefore D_m = 4$

Ex. 9.7.2 : Given below is a 5 × 5 image. Find out D_4, D_8.

Soln. :

	0	1	2	3	4
0	3	2	4	3	(1q)
1	0	4	4	3	2
2	2	2	2	0	2
3	2	2	1	1	1
4	(1p)	0	1	0	3

(a)
$p(x_1, y_1) = p(4, 0)$
$q(x_2, y_2) = q(0, 4)$
$D_4(p, q) = |x_1 - x_2| + |y_1 - y_2|$
$= |4 - 0| + |0 - 4|$
$= 4 + 4 = 8$

(b)
$D_8(p, q) = \max(|x_1 - x_2|, |y_1 - y_2|)$
$= \max(|4 - 0|, |0 - 4|)$
$= \max(4, 4) = 4$

Ex. 9.7.3 : Consider the image segment shown

3	1	2	1q
2	2	0	2
1	2	1	1
1p	0	1	2

for V = {0, 1}, compute the length of the shortest m-path between p and q.

Soln. : The path taken is shown below :

3	1	2	1
2	2	0	2
1	2	1	1
1	0	1	2

Hence the shortest m-distance is equal to 5.

9.8 Additional Solved Examples

Ex. 9.8.1 : Develop an algorithm for converting a one-pixel thick 8-path to a 4-path.

Soln. : This can be done by defining all possible neighbourhood shapes to go from a diagonal segment to a corresponding 4-connected segment, as shown in Fig. P. 9.8.1.

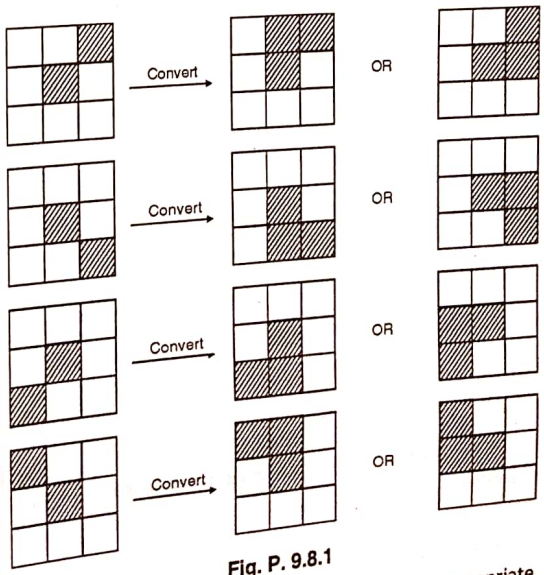


Fig. P. 9.8.1

The algorithm then simply looks for the appropriate match everytime a diagonal segment is encountered.

Ex. 9.8.2 : A binary image contains straight lines oriented horizontally, vertically, at 45° and at – 45°. Give a set of 3 × 3 masks that can be used to detect 1 pixel long breaks in these lines. Assume that the grey level of the lines is 1 and that the grey level of the background is 0.

Soln. :

The 4 masks that we could use are

0	0	0
1	–2	1
0	0	0

Horizontal

0	1	0
0	–2	0
0	1	0

Vertical

0	0	1
0	–2	0
1	0	0

+45°

1	0	0
0	–2	0
0	0	1

–45°

Note that the sum of the coefficients along the required direction is zero.

Hence each mask would yield a value of 0 when centered on a pixel of an unbroken 3-pixel segment oriented in the direction of the required mask. The response of the mask would be +2 when the mask is centered on a one pixel gap in a 3-pixel segment oriented in the direction of the required mask.

Ex. 9.8.3 : Show that the average value of the Laplacian operator $\nabla^2 h$ is zero.

Soln. :

We have seen that

$$\nabla^2 h(r) = +\left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}}$$

What we need to prove is

$$\int_{-\infty}^{\infty} \left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} dr = 0$$

$$\frac{1}{\sigma^4}\int_{-\infty}^{\infty} r^2 e^{-\frac{r^2}{2\sigma^2}} dr - \frac{1}{\sigma^2}\int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr = 0$$

We know from the definition of the Guassian density that

$$\frac{1}{\sqrt{2\pi\sigma^2}}\int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr = 1$$

We also know that the variance of a Guassian random variable is

$$\text{Variance}(r) = \sigma^2 = \int_{-\infty}^{\infty} r^2 e^{-\frac{r^2}{2\sigma^2}} dr$$

∴ We get

$$\int_{-\infty}^{\infty} \left[\frac{r^2-\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} dr = \frac{\sqrt{2\pi\sigma^2}}{\sigma^4}\sigma^2 - \frac{\sqrt{2\pi\sigma^2}}{\sigma^2} = 0$$

Ex. 9.8.4 : Show that the Laplacian mask is a Isotropic filter. (Hint : use a 3 × 3 mask)

Soln. :

By Isotropic filters we mean, filters that are rotation invariant. It means that rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result.

A 3 × 3 Laplacian mask is given

0	–1	0
–1	4	–1
0	–1	0

Consider a 3 × 3 image

1	2	1
2	4	2
4	8	4

Case 1: Image → Convolve with Laplacian mask → Rotate

Applying the mask we get,

0	2	0
–1	2	–1
6	20	6

Rotate →

6	20	6
–1	2	–1
0	2	0

Case 2 : Image → Rotate → Convolve with Laplacian mask

Rotating image we get,

4	8	4
2	4	2
1	2	1

Apply mask →

6	20	6
–1	2	–1
0	2	0

We see that (1) = (2)

Hence we conclude that the Laplacian mask is an Isotropic filter (Invariant to rotation).

Note : Convolution was performed after zero padding the image.

Ex. 9.8.5 : Show that the Laplacian operation defined by $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ is istotropic

Soln. :

We use the following equations relating coordinates after axis rotation by angle θ.

$$x = x' \cos\theta - y' \sin\theta$$
$$y = x' \sin\theta + y' \cos\theta$$

where (x, y) are the unrotated and (x′, y′) are the rotated coordinates.

Laplacian operation for the unrotated coordinates are,

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

and the Laplacian operation for the rotated coordinates are,

$$\nabla^2 f = \frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2}$$

Let us start with

$$\frac{\partial f}{\partial x'} = \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y} \cdot \frac{\partial y}{\partial x'} = \frac{\partial f}{\partial x} \cos\theta + \frac{\partial f}{\partial y} \sin\theta$$

We take the partial derivative of this expression again with respect to x′

$$\frac{\partial^2 f}{\partial x'^2} = \frac{\partial^2 f}{\partial x^2} \cos^2\theta + \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) \sin\theta \cos\theta + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) \cos\theta \sin\theta + \frac{\partial^2 f}{\partial y^2} \sin^2\theta \quad \ldots(1)$$

We now find $\frac{\partial f}{\partial y'}$

$$\frac{\partial f}{\partial y'} = \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y} \cdot \frac{\partial y}{\partial y'} = -\frac{\partial f}{\partial x} \sin\theta + \frac{\partial f}{\partial y} \cos\theta$$

Taking the partial derivatives of this expression with respect to y′ gives us

$$\frac{\partial^2 f}{\partial y'^2} = \frac{\partial^2 f}{\partial x^2} \sin^2\theta - \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right) \cos\theta \sin\theta - \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right) \sin\theta \cos\theta + \frac{\partial^2 f}{\partial y^2} \cos^2\theta$$

Adding the two equations we get

$$\frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

This proves that the Laplacian operation is independent of rotation.

Ex. 9.8.6 : Show that subtracting the Laplacian from an image is proportional to unsharp masking.

Soln. :

Let f(x, y) be the original image.

The Laplacian mask is given by

0	1	0
1	−4	1
0	1	0

$$\therefore f(x, y) - \nabla^2 f(x, y) = f(x, y) - [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)]$$
$$= 6 f(x, y) - [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) + f(x, y)]$$
$$= 5\left[1.2 f(x, y) - \frac{1}{5}[f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) + f(x, y)]\right]$$

$$f(x, y) - \nabla^2 f(x, y) = 5\{1.2 f(x, y) - \bar{f}(x, y)\}$$

Here $\bar{f}(x, y)$ denotes the average of f(x, y).

Treating the constants as factors of proportionality we get.

$$f(x, y) - \nabla^2 f(x, y) \approx f(x, y) - \bar{f}(x, y)$$

The RHS is the equation for unsharp masking. Hence subtracting the Laplacian from an image is equivalent to unsharp masking.

Ex. 9.8.7 : Segment the following image into two regions based on the edge-orientations derived through a gradient operator clearly depicting the boundary.

27	27	27	27	27
25	25	25	25	25
23	23	30	33	36
21	21	33	36	39
19	19	36	39	42

Soln. : The easiest way to segment this image into two regions would be use a thresholding function with T ≥ 28. But since we have been asked to do this using gradient operators, let us use a Prewitts compass operator.

We use the following Prewitts masks and convolve each one of them with the image.

We finally calculate the maximum.

$M_1 =$

−1	−1	−1
0	0	0
1	1	1

$M_2 =$

−1	−1	0
−1	0	1
0	1	1

$M_3 =$

−1	0	1
−1	0	1
−1	0	1

$M_4 =$

0	1	1
−1	0	1
−1	−1	0

$M_5 =$

1	1	1
0	0	0
−1	−1	−1

$M_6 =$

1	1	0
1	0	−1
0	−1	−1

$M_7 =$

1	0	−1
1	0	−1
1	0	−1

$M_8 =$

0	−1	−1
1	0	−1
1	1	0

Let the original image be f (x, y).

While convolving we ignore the corner pixels.

$A = f * M_1 =$

0	0	0	0
0	−5	5	18
0	0	15	33
0	−2	8	18

$B = f * M_2 =$

0	0	0	0
0	−8	−1	9
0	−15	−6	13
0	−27	−23	0

$C = f * M_3 =$

0	0	0	0
0	−7	−10	−6
0	−19	−25	−12
0	−36	−45	−18

$D = f * M_4 =$

0	0	0	0
0	1	−9	−15
0	−11	−29	−31
0	−21	−37	−24

$E = f * M_5 =$

0	0	0	0
0	5	−5	−18
0	0	−15	−33
0	2	−8	−18

$F = f * M_6 =$

0	0	0	0
0	8	1	−9
0	15	6	−13
0	27	23	0

$G = f * M_7 =$

0	0	0	0
0	7	10	6
0	19	25	12
0	36	45	18

$H = f * M_8 =$

0	0	0	0
0	−1	9	15
0	11	29	31
0	21	37	24

We now take the maximum from each of the pixel locations.

Hence we have

$W =$

0	0	0	0
0	8	10	18
0	19	29	33
0	36	45	24

We zero pad one row and one column to get the final image.

Final =

0	0	0	0	0
0	0	0	0	0
0	0	8	10	18
0	0	19	29	33
0	0	36	45	24

Hence we have two regions, one being the zero region (say a) and the other being the non-zero region (say b).

a	a	a	a	a
a	a	a	a	a
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

Summary

Segmentation is an essential step in almost all image analysis systems. It is imperative to understand the applications of image segmentation. Segmentation partitions an image into meaningful regions having certain characteristics unique to that region. In this, the output is an abstract representation of the input image. There exists no general segmentation algorithm which will work satisfactorily for all images. Users have to choose a particular segmentation algorithm which would be suitable for the problem in hand. Segmentation based on dissimilarities as well as segmentation based on similarities have been discussed in detail. MATLAB codes have been provided for gradient operators such as Prewitts and Sobel. Special emphasis has been given to Hough transforms which forms an important topic of edge linking.

Review Questions

Q. 1 Explain the term Image Segmentation.

Q. 2 Explain, segmentation based on discontinuties and segmentation based on similarities.

Q. 3 Explain edge detection in detail.

Q. 4 Show that the Laplacian operator is invariant to rotation.

Q. 5 Explain the need of a LoG operator.

Q. 6 Explain the following edge extraction operators :

(a) Sobel (b) Prewitt

(c) Roberts (d) Laplacian

Use these masks on an image of your choice.

Q. 7 A binary image contains straight lines oriented horizontally, vertically, at + 45° and at – 45°. Give a set of 3 × 3 masks that can be used to detect 1-pixel long breaks in these lines. Assume that the grey level of the lines is 1 and that of the background is 0.

Q. 8 Explain the concept of compass operators.

Q. 9 Explain whether poorly illuminated image can be easily segmented.

Q. 10 Write in detail on connectivity of pixels.

Q. 11 What is the distance transform ?

Q. 12 Explain :

(a) Euclidean distance

(b) City block distance

(c) Chess board distance

(d) D_m distance.

❑❑❑