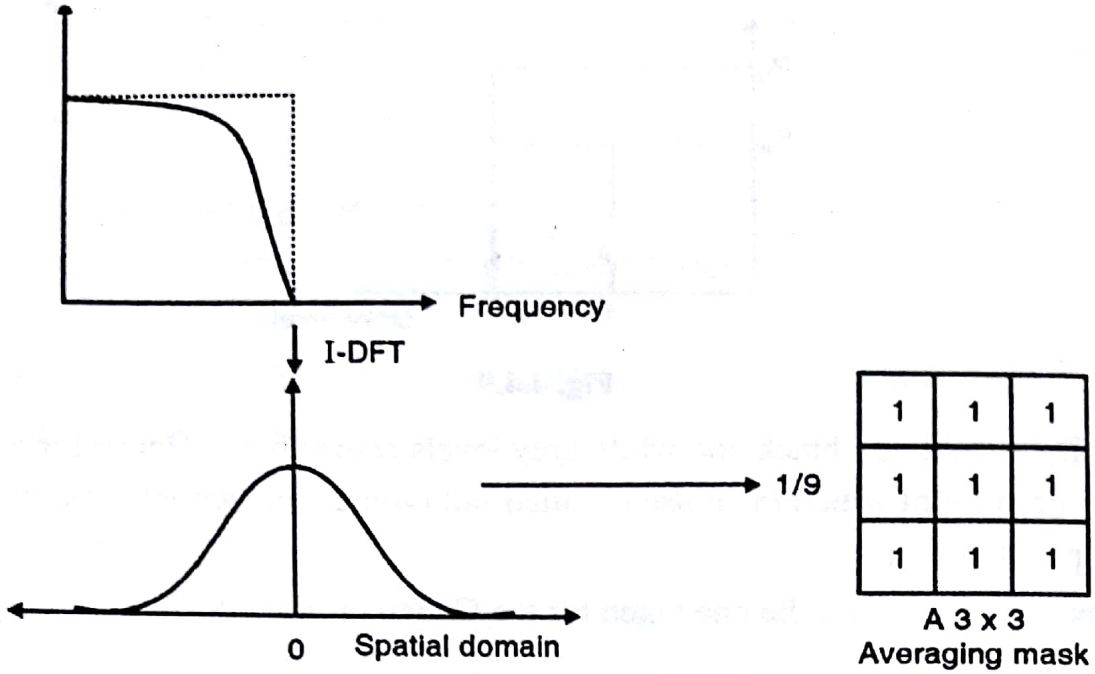


Fig. 4.4.11

From the spatial response we generate the mask that would give us the low pass filtering operation. One important thing to note from the spatial response is that all the coefficients are positive. The standard low pass averaging mask (3×3) is given above. As the name suggests, each element of the mask is the average value.

We could also use a 5×5 or a 7×7 mask as per our requirement.

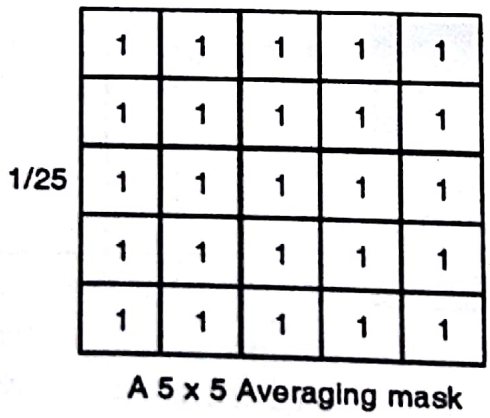


Fig. 4.4.12

Let us take an example of a 3×3 mask on a pseudo-image and see how it eliminates the edges. Consider a 8×8 size image. It is clear that the image has a single edge between 10 and 50. To get rid of this edge (high frequency) we use a 3×3 averaging mask.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

We place a 3×3 mask on this image. We start from the left hand top corner. We cannot work with the borders and hence are normally left as they are.

We then multiply each component of the image with the corresponding value of the mask. Since all the nine values of the image are 10, the average is also ten and the centre pixel (the underlined pixel) remains ten. We now shift the mask towards the right till we reach the end of the line and then move it downwards. Some of the mask positions are shown here.

Note : The resultant should be written in a new matrix (image).

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

The last one in the sequence being

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

The result of convolving the image with the 3×3 averaging mask is shown.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
23.3	23.3	23.3	23.3	23.3	23.3	23.3	23.3
36.6	36.6	36.6	36.6	36.6	36.6	36.6	36.6
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

As we notice, the low frequency regions have remained unchanged, but the sharp edge between 10 and 50 has become blurred. The transition has reduced. Now from 10, the grey level changes to 23.3 (23 after rounding off), from 23 it changes to 36.6 (36 after rounding off) and finally from 36 it changes to 50. Hence we can say that the sharp edge has become blurred. Check for yourself, what would happen if you took a bigger mask, say, 5×5. (You will have to take a bigger image to test your results).

These kinds of averaging filters are excellent when the image contains Gaussian noise. It achieves filtering by blurring the noise. Some of the other low pass averaging masks are shown.

$\frac{1}{6} \times$

0	1	0
1	2	1
0	1	0

$\frac{1}{10} \times$

1	1	1
1	2	1
1	1	1

The averaging filter removes the noise by blurring it till it is no longer seen. But in the process, it also blurs the edges. Bigger the averaging mask more is the blurring. There are times when the image contains salt and pepper noise. If we use an averaging filter to remove the same, it will blur the noise but it would also ruin the edges. Hence when we need to eliminate salt and pepper noise, we work with a non-linear filter known as the Median filter. They are also called order-statistic filters because their response is based on the ordering or ranking of the pixels contained within the mask.

In this case, we use a mask similar to the averaging filter except that the mask has no values. So it's like working directly with the 8 neighbours of the centre pixel.

The steps to perform median filtering are as follows :

(1) Assume a 3×3 empty mask.

(2) Place the empty mask at the left hand corner.

(3) Arrange the 9 pixels in ascending or descending order.

(4) Choose the median from these nine values.

(5) Place this median at the centre.

(6) Move the mask in a similar fashion to the averaging filter.

In median filtering, the grey level of the centre pixel is replaced by the median value of the neighbourhood. Always write the resultant in a new matrix (image). We shall explain median filtering with an example. In the image shown below let 250 be the salt and pepper noise. If we use an averaging filter, the noise would spread out and the edges would also end up getting blurred. If we only want to remove the noise without disturbing the edges, we use a median filter.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	250	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Just like the earlier case, we start by placing the mask at the left hand corner. We again ignore the borders.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	250	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	250	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	250	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Arranging the 9 pixels in ascending or descending order we get,

(10, 10, 10, 10, 10, 10, 10, 10, 250)

The median is the 5^{th} value since the mask is 3×3 mask. This value is placed at the center. We now move the mask to the new location and continue the procedure. Performing this operation on the entire image, we get the final image :

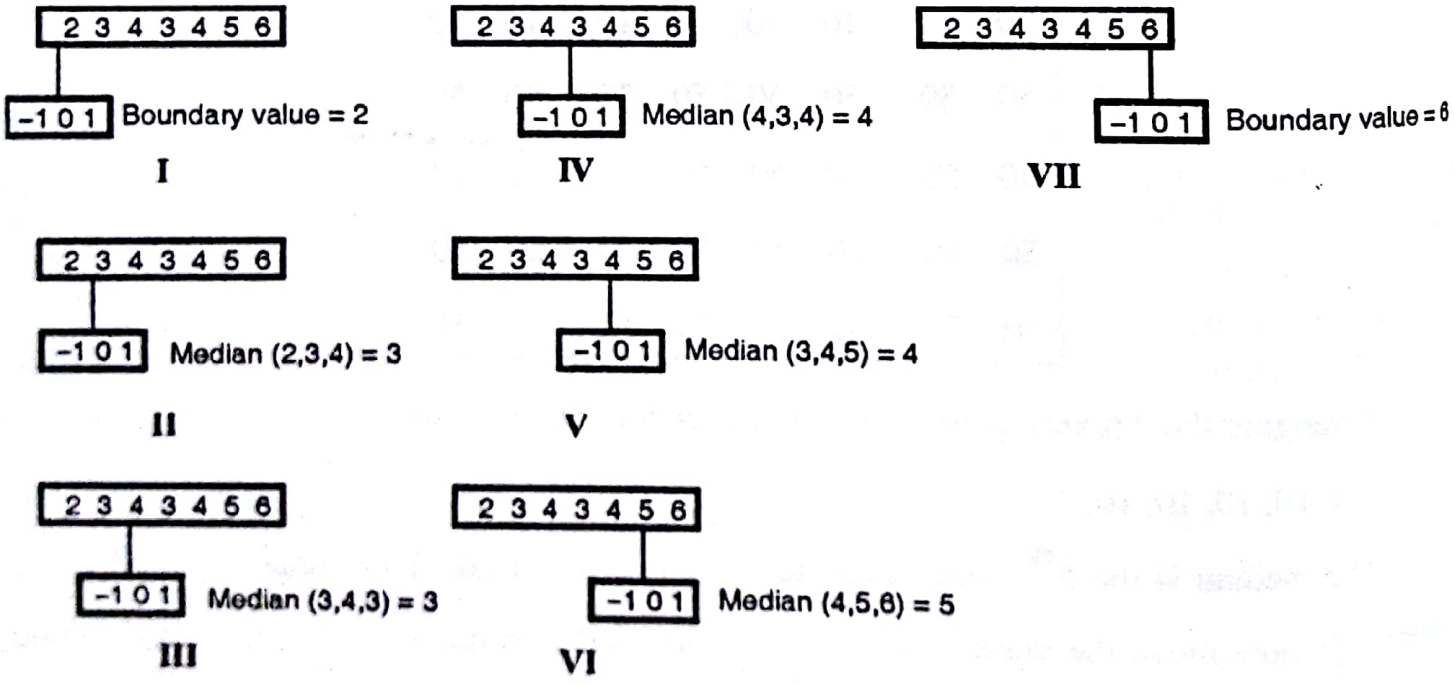
As can be seen, the salt and pepper noise gets eliminated without distorting the edges. Median filters give astonishing results even when the amount of noise is relatively large.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50

Ex. 4.4.1 : If x = {2 3 4 3 4 5 6} and w ={–1 0 1}, perform median filtering.

Soln. :

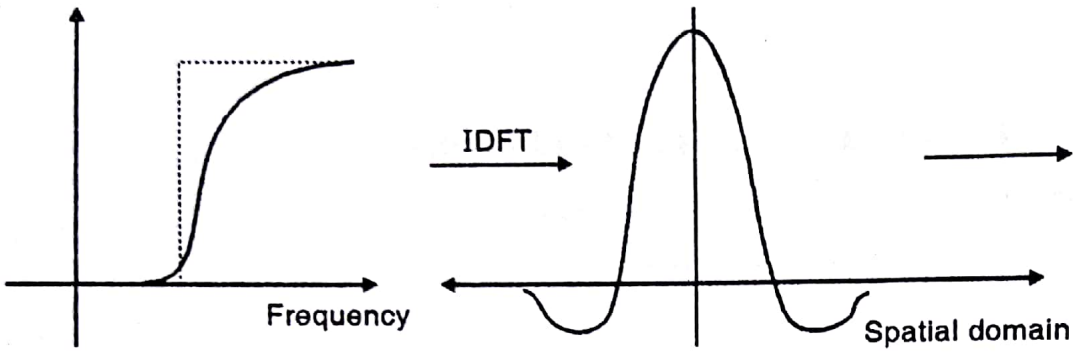
w = {–1, 0, 1} simply means that the size of the mask is 1 × 3 and the term 0 indicates the position from where the filtering starts.



Hence the final answer is y = {2 3 3 4 4 5 6}. The principal function of median filtering is to force points with distinct intensities to be more like their neighbours.

4.5 Highpass Filtering :

Highpass filtering eliminates the low frequency regions while retaining or enhancing the high frequency components. An image, which is high-passed, would have no background (as background are low frequency regions) and would have enhanced edges. Hence high pass filters are used to sharpen blurred images. Once we have understood as to how a mask moves over the entire image for the averaging filter, the same applies to the high pass filter as well. All that needs to be changed are the mask coefficients. The frequency response and the spatial response of a high pass filter are shown in Fig. 4.5.1. This will be proved in chapter of Image Enhancement in Frequency Domain.



−1	−1	−1
−1	8	−1
−1	−1	−1

A 3 x 3 high pass mask

Fig. 4.5.1

As seen from the spatial response, it is clear that the mask coefficients have to be such that they have a positive value at the centre and negative values at the periphery.

One of the high pass masks is shown in Fig. 4.5.1.

The important thing to note is that the sum of the coefficients of the high pass mask has to be equal to zero. This is because, when we place this mask over the low frequency regions, the result should be zero.

Let us take an example of a pseudo-image.

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100

By now we know what to expect when we do a high pass operation. The background, which is low frequency, gets eliminated while the edges get enhanced.

We move the mask in a similar fashion as in the low pass filter example. The output that we get is shown below. Note that the background has been eliminated and we get all zero values. This is because the sum of the mask coefficients is zero.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
–270	–270	–270	–270	–270	–270	–270	–270
270	270	270	270	270	270	270	270
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

There are two issues to be dealt with, when working with high pass masks.

(1) As we can see, there are negative values in the output image. Pixel values *cannot be negative*. They always start with zero. Hence we need to get rid of the negative values. If we consider the mod operation, do you think it would solve the problem ? NO. The reason for that is –270 is a value that is lower than zero, that is, it is supposed to be darker than the darkest value i.e. zero. If we now take the mod value i.e. MOD [–270], we get +270, which is definitely not darker than the zero value. What would happen is that all large negative values would be shown as bright spots. Hence taking the mod of negative values will distort the image grey levels. A simple way to get rid of this problem is to let all negative values be zero. Hence –270 would be written as 0.

(2) The values of the original image at the edge are very large and there is a tendency of them going out of range due to the centre weight of +8. We always encounter this problem in practice. To eliminate this problem, we use a mask with a scaling function.

Hence for practical purposes we use a mask shown below.

$\frac{1}{9}$

– 1	– 1	– 1
– 1	8	– 1
– 1	– 1	– 1

3×3

High pass mask

Note that 1/9 is simply a scaling function unlike the low pass mask in which the 1/9 term is a part of the averaging operation. In this case we could also work with 1/8 or 1/7 depending on the image. Some of the other high pass masks are also shown below.

0	–1	0
–1	4	–1
0	–1	0

–1	–2	–1
–2	12	–2
–1	–2	–1

The result using the first mask and putting negative values as zeros is

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
30	30	30	30	30	30	30	30
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MATLAB program for high pass filtering

```
%% High pass filtering %%
clear all
clc
aa=imread ('xray.tif');
a=double (aa);
[row col]=size(a);
w=[-1 -1 -1; -1 8 -1, -1 -1 -1]
for x=2:1:row-1
        for y=2:1:col-1
            a1(x,y)= w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)* ...
            a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)* ...
            a(x,y+1)+w(7)*a(x+1,y-1)+w(8)*a(x+1,y)+w(9)* ...
            a(x+1,y+1);
        end
end
figure(1)
imshow(uint8(a))
figure(2)
imshow(uint8(al))
```

(a) Original image

(b) High pass filtered image

Fig. 4.5.2

4.6 High-Boost Filtering :

As can be seen, the high pass filter gives great results. But there is one problem. It gets rid of the complete background. There are times, when we need to enhance the edges but also retain some of the background. To do that, we use a modified version of the high pass filter known as High-Boost filtering.

In High-Boost filtering, we pass some of the background along with the high frequency content.

We know that High pass = Original – Low pass

To pass some of the background, we multiply the original image with a multiplicative factor A. This gives us high boost filtering.

Hence,

$$\text{High Boost} = (A)\ \text{original} - \text{Low pass}$$
$$= (A-1)\ \text{Original} + \text{Original} - \text{Low pass}$$
$$\text{High Boost} = (A-1)\ \text{Original} + \text{High pass}$$

If A = 1, then

$$\text{High Boost} = \text{High pass}$$

If A > 1, then some of the original signal is added back to the high pass result. This process restores some of the background into the high passed image. This technique is also known as unsharp masking. The mask coefficients for high boost filtering are shown on the next page.

Here $X = 9A - 1$

Hence if $A = 1$, $X = 8$ which is a high pass mask.

$\frac{1}{9}$

–1	–1	–1
–1	X	–1
–1	–1	–1

3×3 High boost mask

If A = 1.1, X = 8.9.

We have a mask which is as shown

$\frac{1}{9}$

–1	–1	–1
–1	8.9	–1
–1	–1	–1

3×3 High boost mask

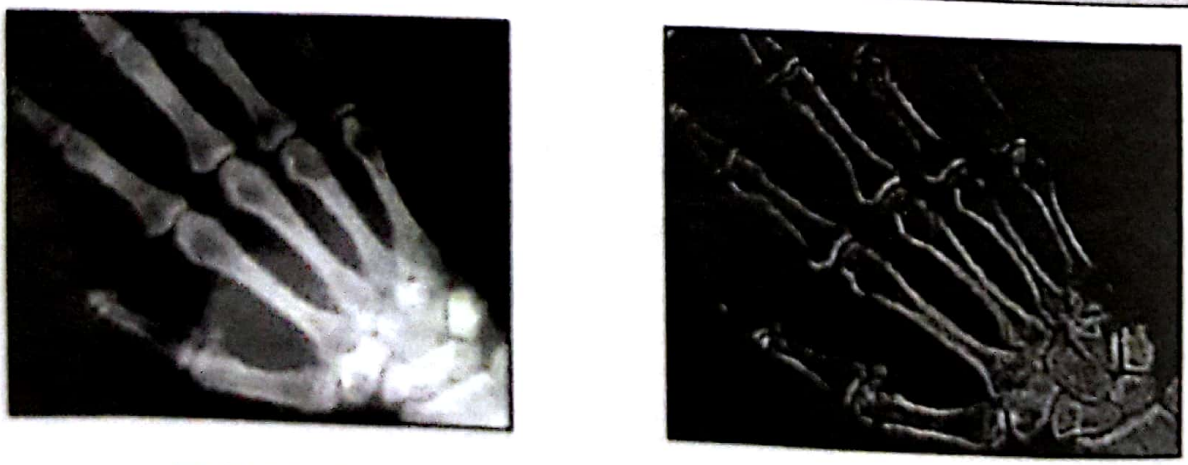
We can select different values of A and see the difference.

Use the same pseudo image as the one used in the high pass example to see the results. What you will notice is that places which had a zero in the high pass example will now have some positive values. Hence it does not eliminate the background completely. This technique is one of the basic tools that is used in the printing industry.

There are other neighbourhood techniques like Robert's filtering, Sobel's filtering and Prewitt's filtering, which are similar to the methods discussed above. They form a separate chapter called Segmentation and hence would be discussed later.

MATLAB program for high-boost filtering

```
%% High boost filtering
clear all
clc
aa=imread ('xray.tif');
a=double(aa);
[row col]=size (a);
w=[-1 -1 -1; -1 8.9 -1; -1 -1 -1]; %% High boost mask
for x=2:1:row-1
        for y=:1:col-1
            a1(x,y)= w(1)*a(x-1,y-1)+w(2)*a(x-1,y)+w(3)* ...
            a(x-1,y+1)+w(4)*a(x,y-1)+w(5)*a(x,y)+w(6)* ...
            a(x,y+1)+w(7)*a(x+1,y-1)+w(8)*a(x+1,y)+w(9)* ...
            a(x+1,y+1);
        end
end
figure(1)
imshow(uint8(a))
figure(2)
imshow(uint8(a1))
```



(a) Original (b) High boost

Fig. 4.6.1

The generalised MATLAB program for high pass filtering using any mask size is given below.