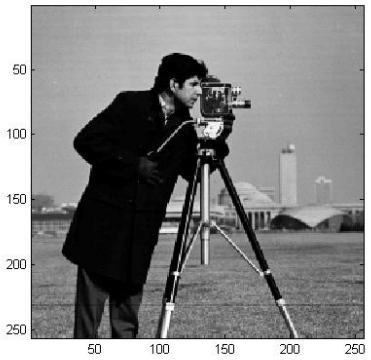
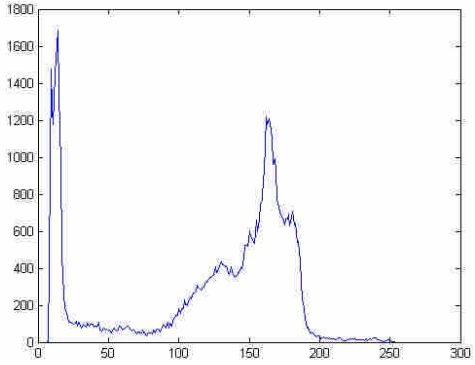
# Example

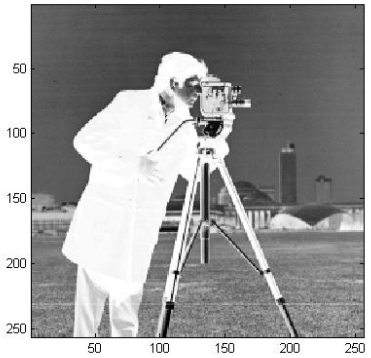
```
A=imread('cameraman.tif');
A=double(A);
for i=1:8
B(:,:,i)=uint8(A>=2^(8-i));
A=A-2^(8-i).*double(B(:,:,i));
end
figure
for i=1:8
subplot(3,3,i)
imagesc(B(:,:,i))
end
colormap(gray)
subplot(3,3,9)
A=imread('cameraman.tif');
A=double(A);
imagesc(A)
```



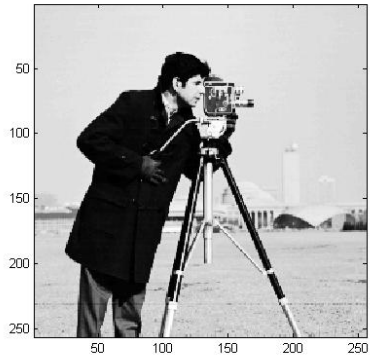
A (Original image)



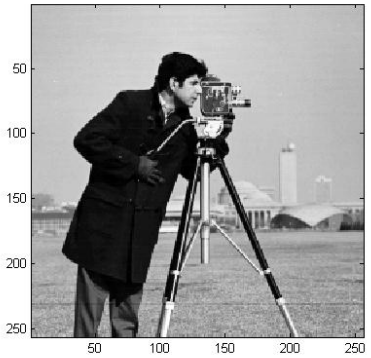
plot(x,y) A



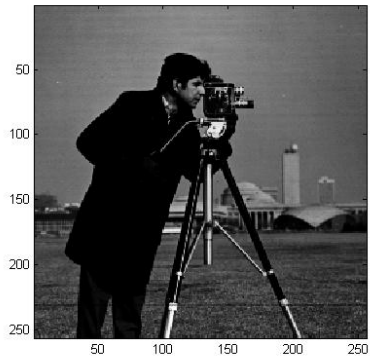
B (negative)



C



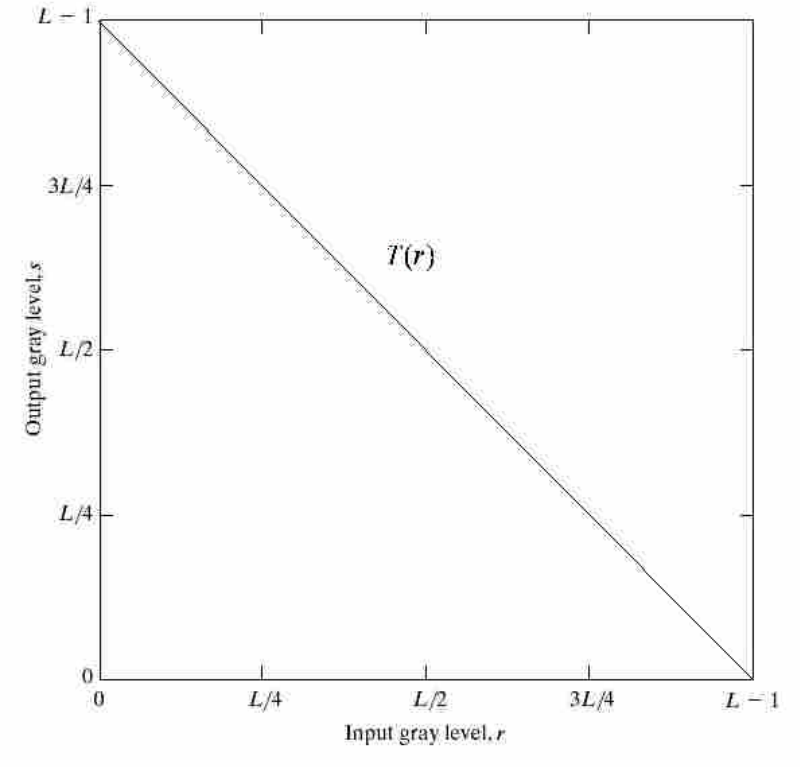
D



E

```
A=imread('cameraman.tif');
imagesc(A)
colormap(gray)
A=double(A);
x=0:255;
[y,x]=hist(A(:),x);
figure
plot(x,y)
B=255-double(A);
figure
imagesc(B)
colormap(gray)
C=A*2.*(A<=100)+(200+55/155
*(A-100)).*(A>100);
figure
imagesc(C)
colormap(gray)
D=sqrt(A*255);
imagesc(D)
E=A.^2/255;
imagesc(E)
```

# Image negatives




The negative of an image with gray levels in the range [0,L-1] is obtained by the following expression

$$s = L - 1 - r.$$

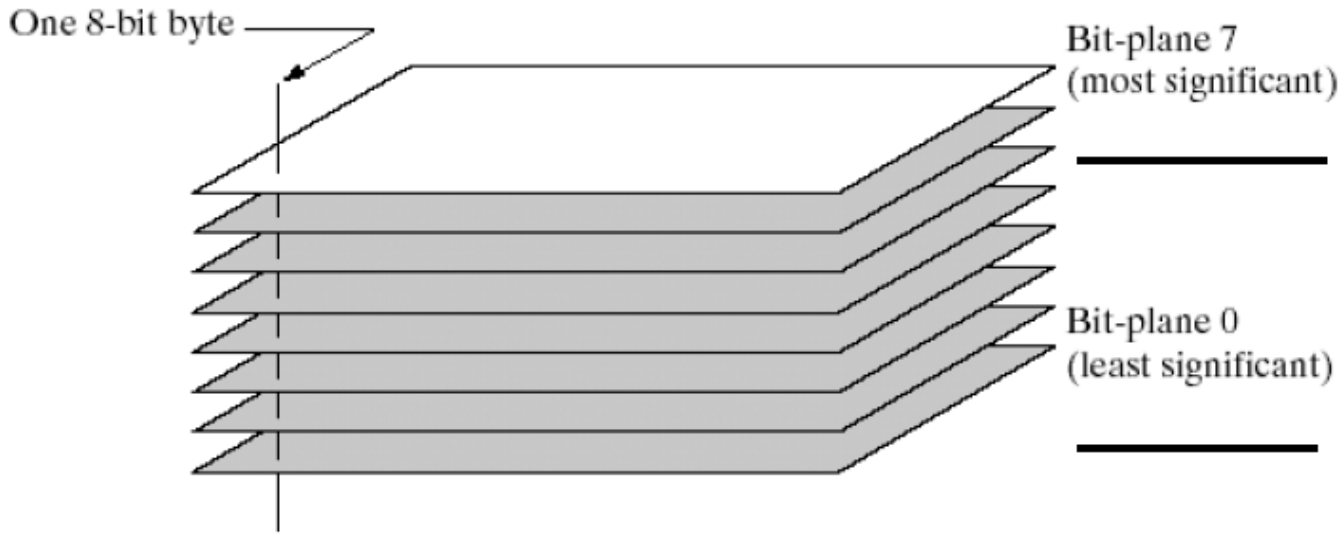
Transformation function

Original Image

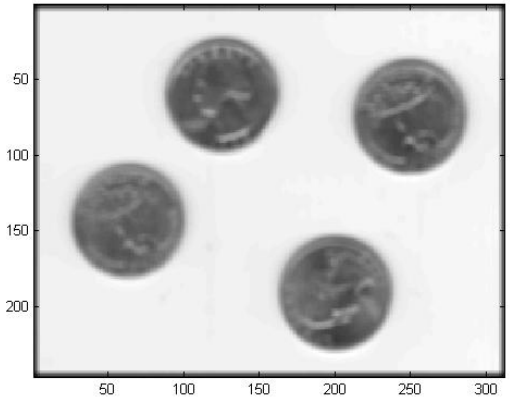
Negative Transformation

# Bit-plane slicing

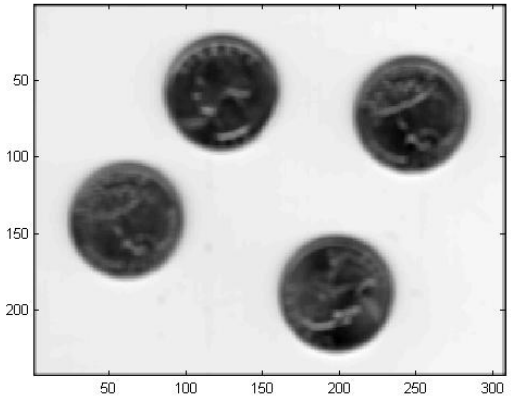
- Bit-plane slicing :highlighting the contributions of specific bits



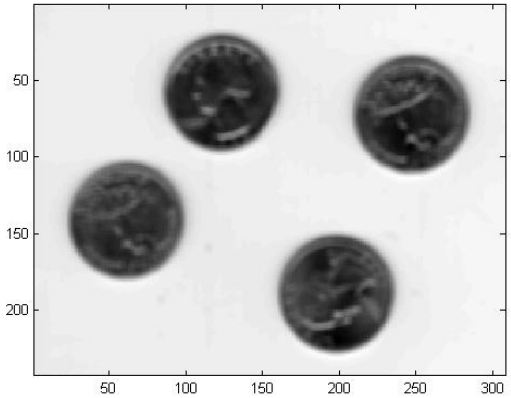
A (Original image)



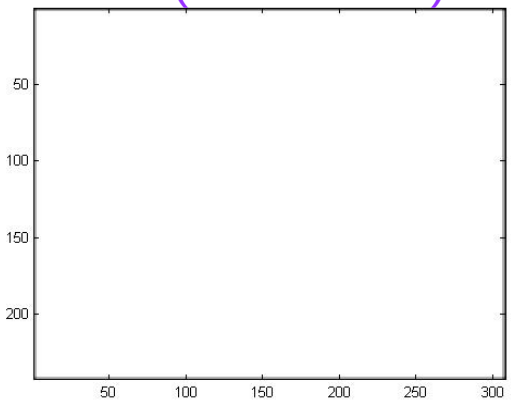
B (convolution)



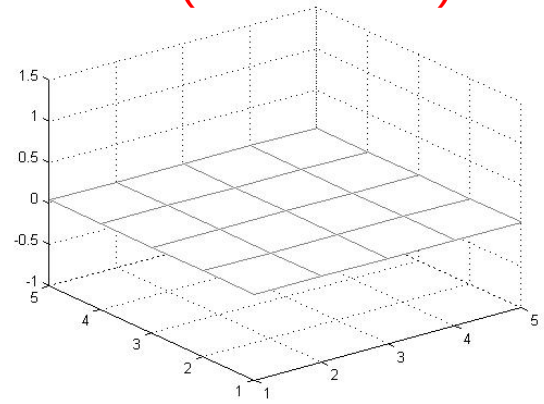
B (convolution)



B (convolution)

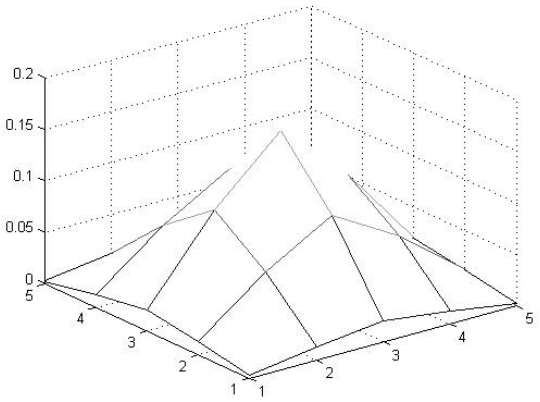


C1

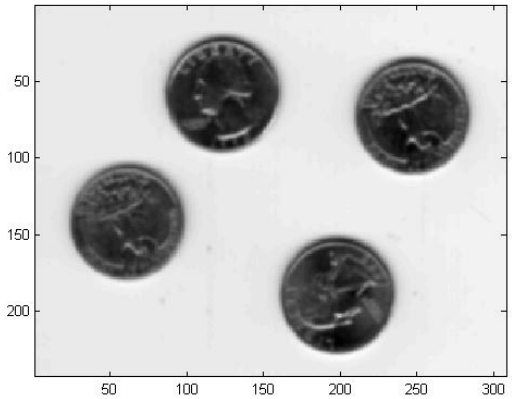


M

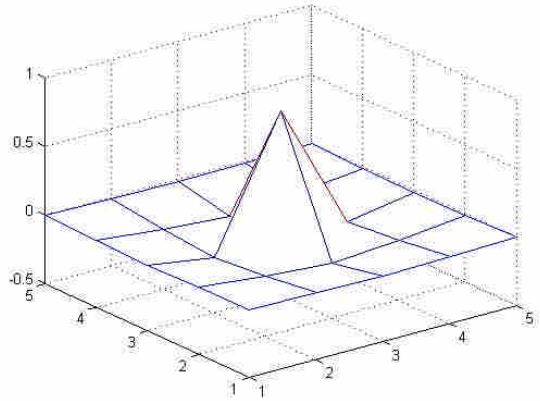
```
A=imread('eight.tif');
A=double(A);
imagesc(A)
colormap(gray)
M=ones(5)/25;
B=conv2(A,M);
figure
imagesc(B)
colormap(gray)
B=conv2(A,M,'same');
imagesc(B)
colormap(gray)
C=ones(size(B));
B=conv2(A,M,'same')./
conv2(C,M,'same');
imagesc(B)
colormap(gray)
C1=conv2(C,M,'same');
figure
colormap(gray)
imagesc(C1*255)
mesh(M)
```



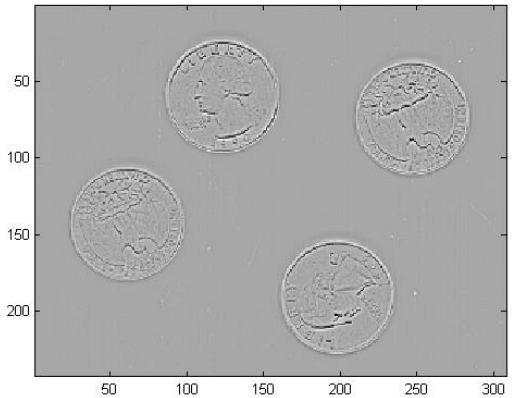
M1



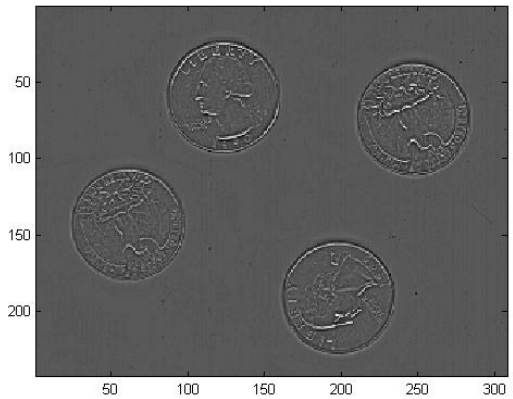
B (lowpass)



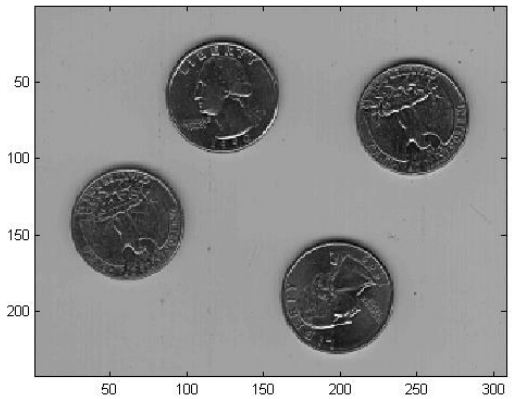
M2



B (highpass)



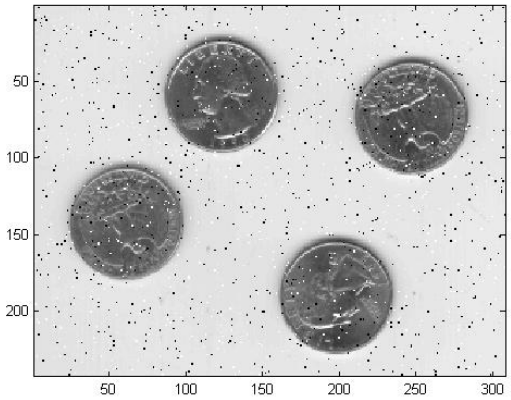
C (highpass=1-lowpass)



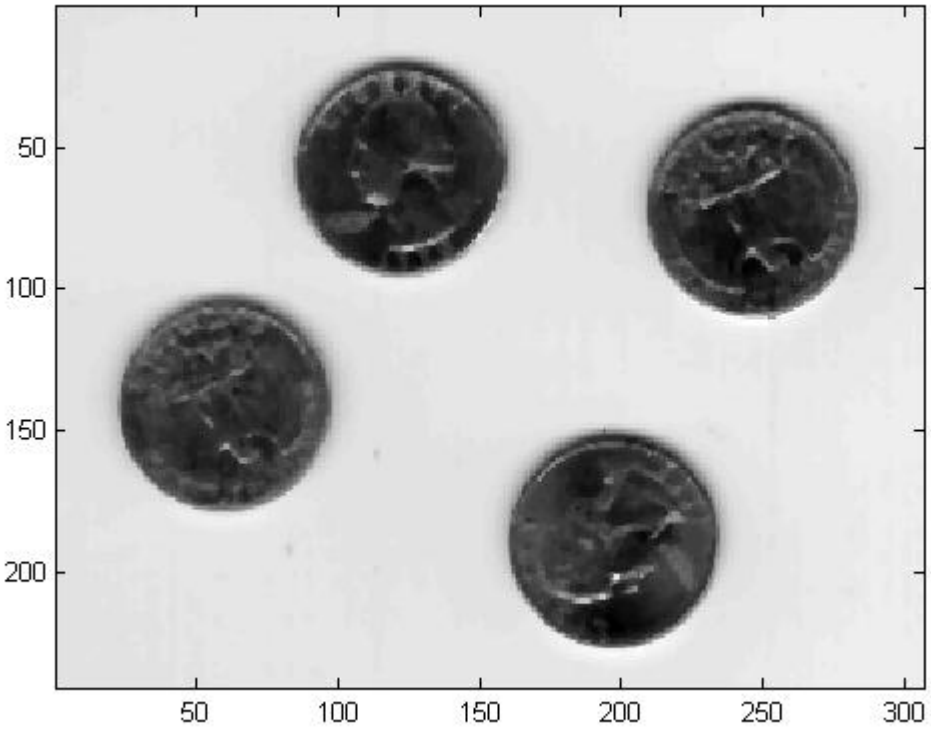
C+A (highboost)

```
f=[.05 .25 .4 .25 .05];
M1=f'*f;
mesh(M1)
B=conv2(A,M1,'same')./
conv2(C,M1,'same');
figure
imagesc(B)
colormap(gray)
D=zeros(5);
D(3,3)=1;
M2=D-M1;
mesh(M2)
B=conv2(A,M2,'same')./
conv2(C,M2,'same');
imagesc(B)
B=conv2(A,M1,'same')./
conv2(C,M1,'same');
C=A-B;
imagesc(C)
imagesc(C+A)
```


I (Original image)



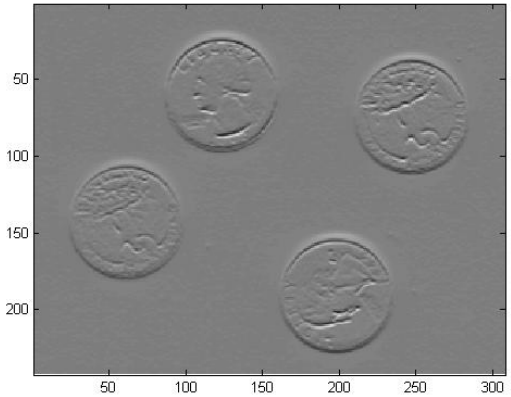

J (salt & pepper noise)




G (median filter)

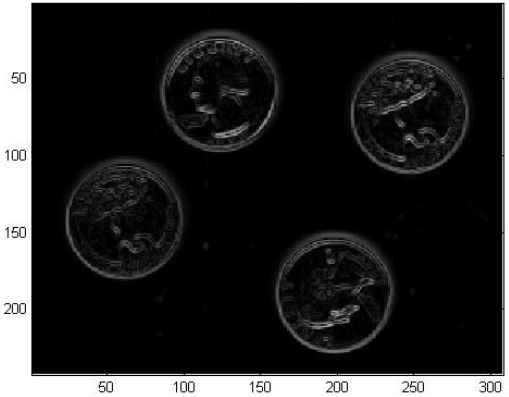
```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper', 0.02);
imagesc(I), figure, imagesc(J)
J=double(J);
for i=1:240
for j=1:306
temp=J(i:i+2,j:j+2);
temp1=sort(temp(:));
G(i,j)=temp1(5);
end,end
figure
colormap(gray)
imagesc(G)
```


Original image




A1(Sobel1)


A2(Sobel2)




A1+A2

```
sobel1=[-1 -2 -1;0 0 0;1 2 1];
sobel2=sobel1‘;
A1=conv2(A,sobel1,'same');
A2=conv2(A,sobel2,'same');
figure
imagesc(A1)
colormap(gray)
figure
imagesc(A2)
colormap(gray)
imagesc(sqrt((A1).^2+(A2).^2))
```

# Basics of Spatial Filter

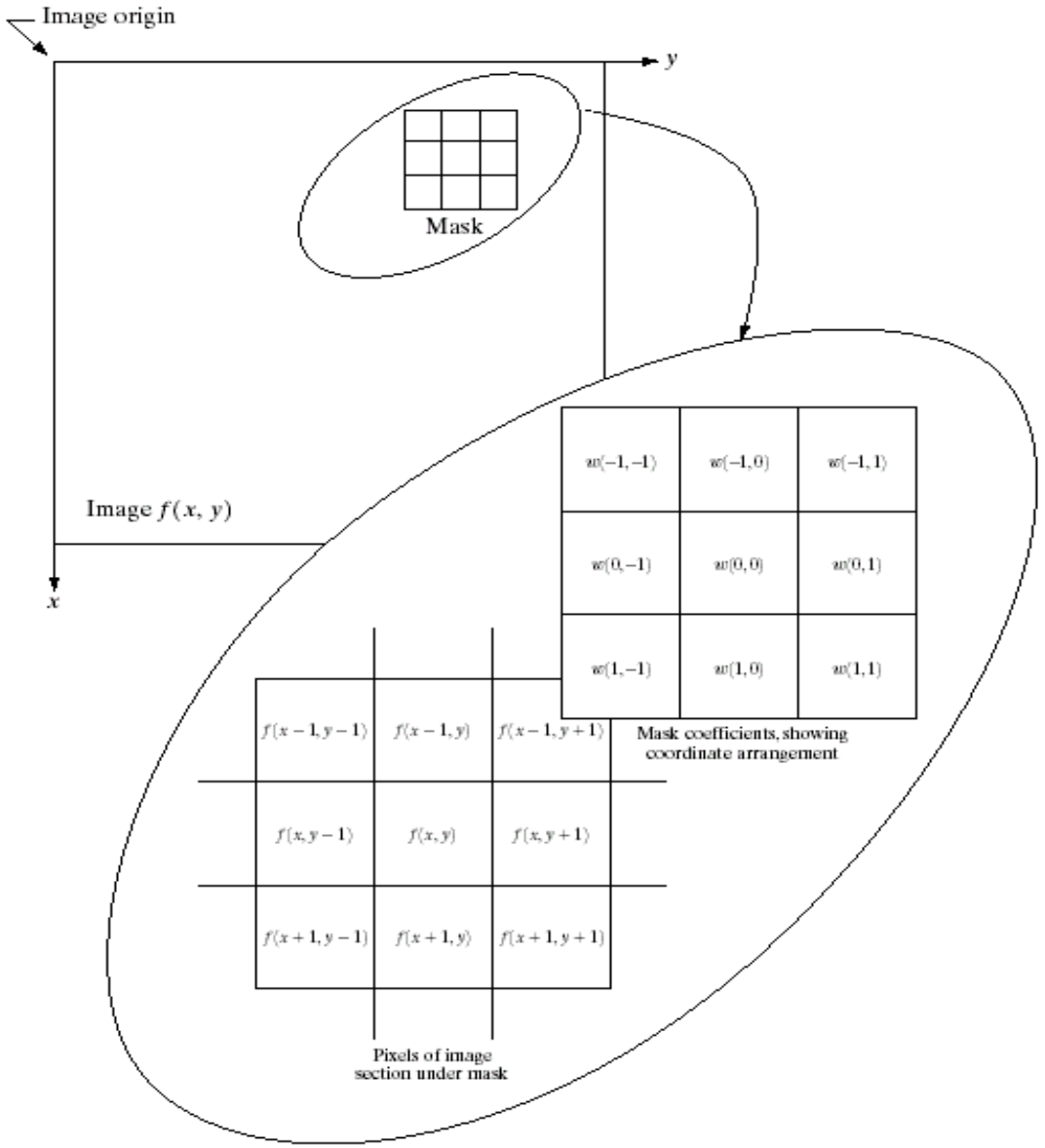



**FIGURE 3.32** The mechanics of spatial filtering. The magnified drawing shows a $3 \times 3$ mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

# Basics of Spatial Filter

- In general, linear filtering of an image f of size M*N with a filter mask of size m*n is given by the expression:

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)$$

*a=(m-1)/2* and *b=(n-1)/2*,*m* x *n* (odd numbers)

# Basics of Spatial Filter

- The basic approach is to sum products between the mask coefficients and the intensities of the pixels under the mask at a specific location in the image:

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

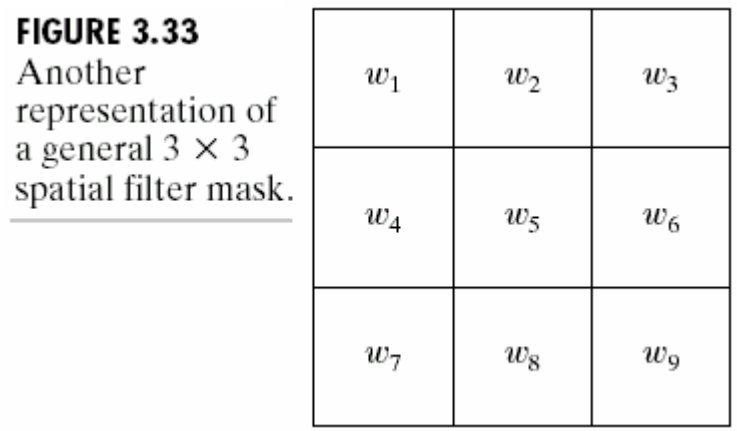



**FIGURE 3.33** Another representation of a general 3 × 3 spatial filter mask.

(for a 3 x 3 filter)