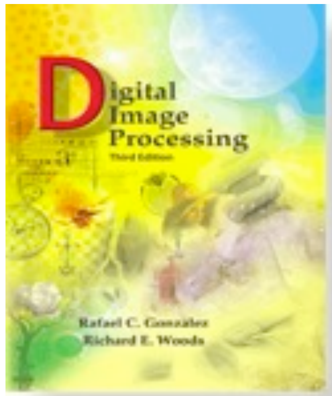


From last lecture:

A generic D_8 *linear* filter will have the following expression

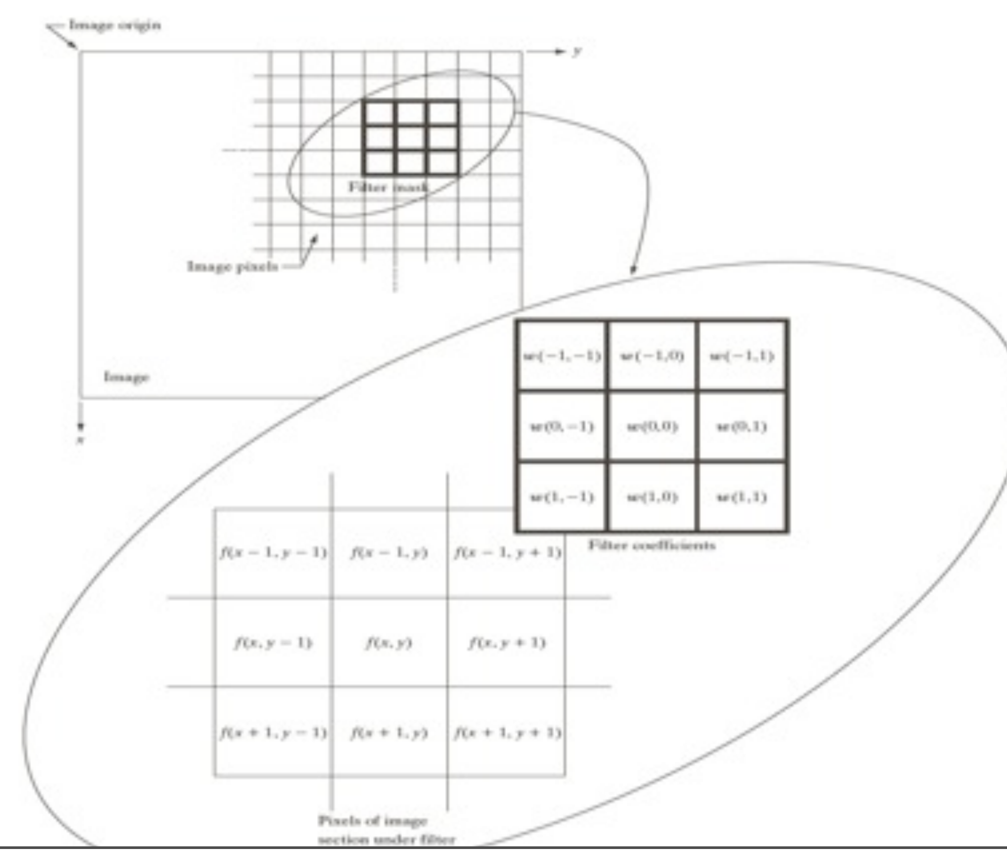
$$
\begin{aligned}
g(x,y) &= w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,0) + \cdots \\
&\quad + w(0,0)f(x,y) + \cdots + w(1,1)f(x+1,y+1)
\end{aligned}
$$

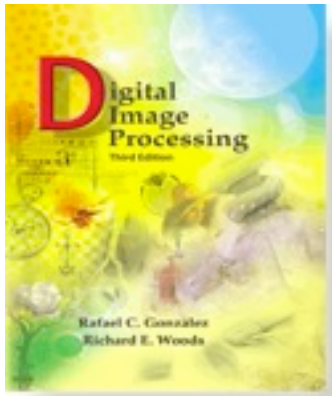
or, for a generic mask using mn pixels,

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

where

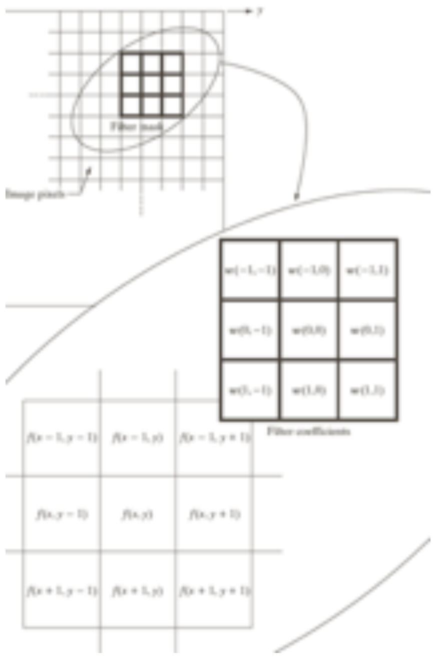
$$m = 2a+1, \qquad n = 2b+1$$





2 important filter concepts: Spatial correlation and convolution

Correlation: is the process of moving a filter mask over an image and computing the sum of the (pointwise) product.



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

Convolution: mechanically the same (except for the mask, rotated by 180 degrees).

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

What effect have correlations and convolution for a signal?

Correlation, 1D:

signal: 00100 → 0032100 (03210)

correlation filter: 123

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

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

Convolution, 1D:

signal: 00100 → 0012300

convolution filter: 123 (becomes 321 rotated 180) (01230)

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

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

Correlation:	given a unit input signal, it *reverses* the filter at the location of the input.	00100, 123 03210
Convolution:	given a unit input signal, it *copies* the filter at the location of the input.	00100, 123 01230

For a practical implementation, the input signal is padded with *m-1* zeros before and after the signal

00100   123 $m=3$

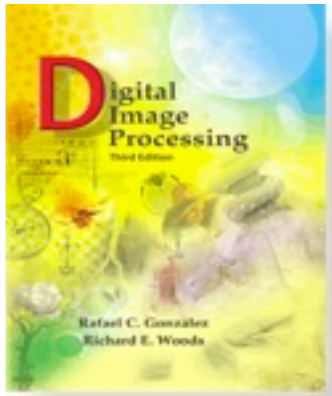
000010000

000010000

123

so that the last element of the filter overlaps with the first element of the signal.

The output signal is cropped by considering as a first entry the one corresponding to the position of the first center of the mask, and last center of the mask.



In Matlab: Convolution

```
CONV Convolution and polynomial multiplication.
    C = CONV(A, B) convolves vectors A and B.  The resulting
    vector is length LENGTH(A)+LENGTH(B)-1.
    If A and B are vectors of polynomial coefficients, convolving
    them is equivalent to multiplying the two polynomials.

    Class support for inputs A,B:
       float: double, single

    See also deconv, conv2, convn, filter and, in the signal
    Processing Toolbox, xcorr, convmtx.

    Overloaded methods:
       gf/conv

    Reference page in Help browser
       doc conv

>> conv([0 0 1 0 0], [1,2,3])

ans =

     0     0     1     2     3     0     0
```

Note that you might need to crop the result to the original size.

In Matlab: Correlation

Can be implemented:

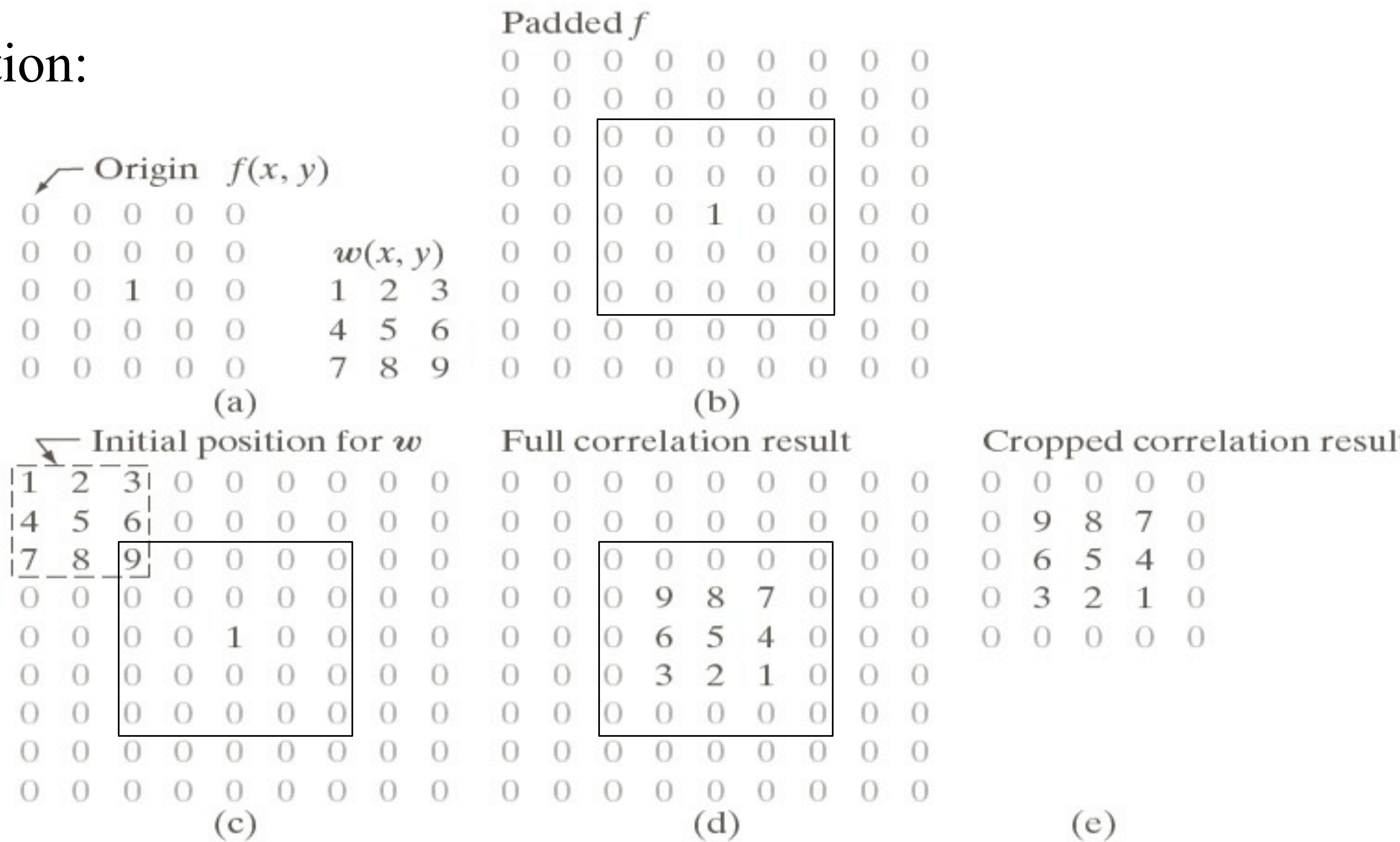
- using convolution (with the filter value reversed)
-using `xcorr`.

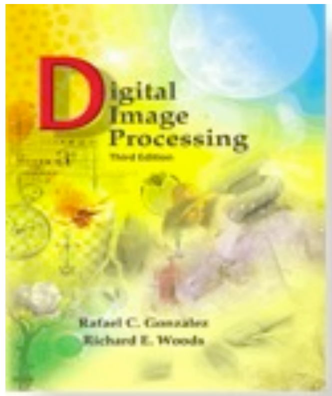
However, the xcorr command introduces extra padding, that has to be removed properly

```
xcorr([0 0 1 0 0], [1,2,3])

ans =

    0.0000    0.0000    0.0000   -0.0000    3.0000    2.0000    1.0000   -0.0000   -0.0000
```

For 2 dimensions (images): the procedure is exactly the same.

Correlation:





Convolution:

Rotated w

9	8	7	0	0	0	0	0	0
6	5	4	0	0	0	0	0	0
3	2	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	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	0	0	0	0	0	0	0

(f)

Full convolution result

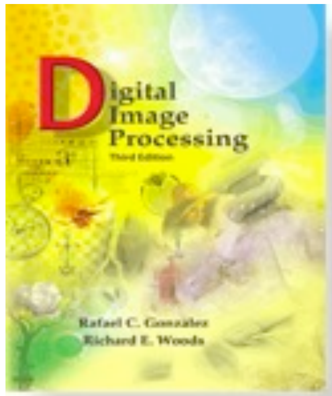
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	2	3	0	0	0
0	0	0	4	5	6	0	0	0
0	0	0	7	8	9	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(g)

Cropped convolution result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

(h)



Convolutions and correlations 2D in Matlab

Advice: avoid wrong cropping by using conv2 with rotated mask!

```
>> A = zeros(5); A(3,3) = 1; b=[1 2 3; 4 5 6; 7 8 9];
>> conv2(A,b,'same')

ans =

     0     0     0     0     0
     0     1     2     3     0
     0     4     5     6     0
     0     7     8     9     0
     0     0     0     0     0
```

note that we can specify the output to have the same dimension as the input matrix

```
xcorr2(A,b)

ans =

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     9     8     7     0     0
     0     0     6     5     4     0     0
     0     0     3     2     1     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

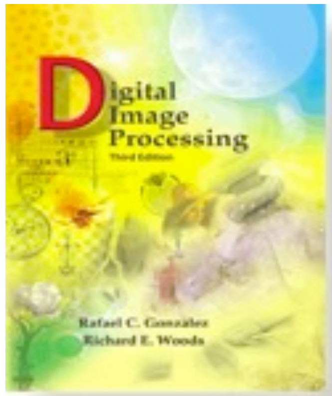
here, the output must be cropped to the right dimension

NB. If the mask is 180-symmetric, then correlation and convolution will produce the same effect.

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

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

In matlab, the `conv2` command works better for images than the `xcorr2` command.



Useful linear filters.

To define a filter:

must define the dimensions m, n, of the filter

must define the coefficients $w(s,t)$ of the filter

$w(1,1)$	$w(1,2)$	$\cdots$		
$w(2,1)$	$\cdots$			
$\cdots$				

Smoothing filters: used for blurring and noise reduction

Blurring removes small objects, useful prior f.ex. to (large) object extraction

Sharpening filters: used to highlight transitions in intensity

Smoothing <=> Integration/averaging

Sharpening <=> Differentiation

Smoothing linear filters:
(also called *averaging* or *lowpass* filters)

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Simple average mask

Idea: by averaging, we reduce the sharp transitions in intensity

As noise is also associated to random intensity transitions, noise will be reduced.

Edges have also sharp intensities, and the effect of smoothing will give an edge blurring.

In the simple averaging case, each pixel in the mask contributes equally to the result.

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

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

1	2	1
2	4	2
1	2	1

In some cases, it might be relevant that pixels contribute differently to the average.
For instance, the pixel in the centre of the mask might be assumed to contribute more to the averaging than others.

=> weighted average
(note: m,n odd)

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

The effect of averaging is that

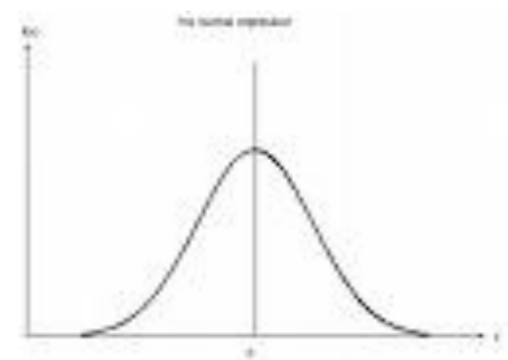
- small object blend with the background
- bigger objects tend to be blob-like

This makes easier to extract bigger object, evt. count them, remove random gaussian noise (small objects)

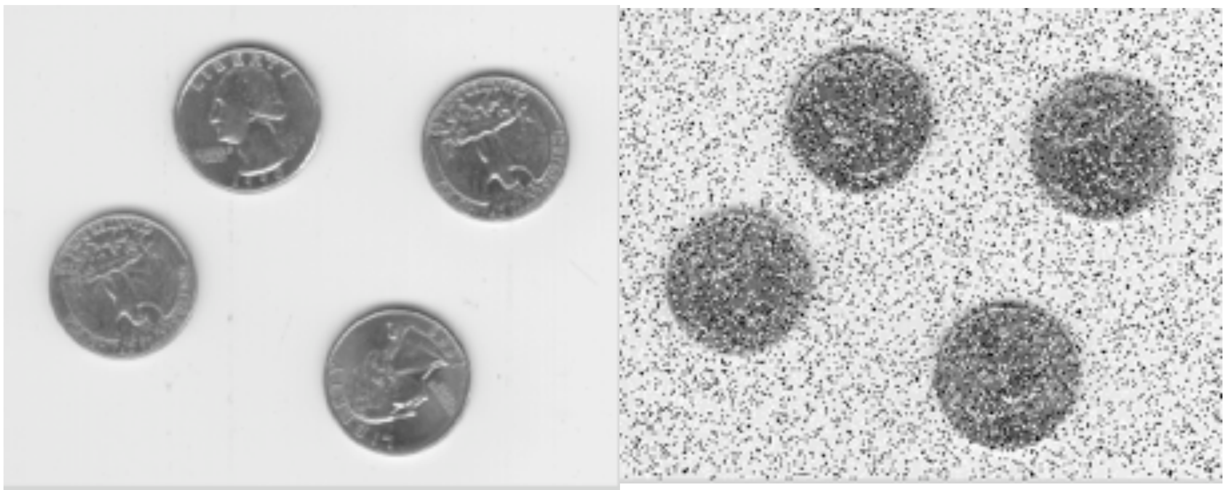
Random gaussian noise vs. salt & pepper noise

Gaussian noise: normally distributed around 0

All the pixels are affected by the same type of noise. The pixels do not lose completely the intensity information.



Salt & pepper noise: impulse noise, represented as 0 (black) or 1 (white) dot superposed to the image

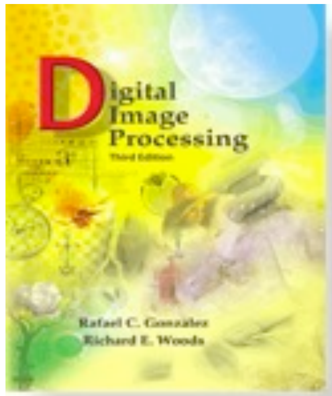


Not all the pixels are affected but those affected lose completely the intensity information.

Linear filters do not work so well for salt and pepper noise, because the resulting intensity value will be influenced of the black/white value

Nonlinear filters using order-statistics

Median filter (nonlinear)



Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

1. sort the values
2. find the median

Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

1. sort the values
2. find the median

22 23 55 25 100 123 20

20 22 23 25 55 100 123

Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

1. sort the values
2. find the median

22 23 55 25 100 123 20

20 22 23 25 55 100 123

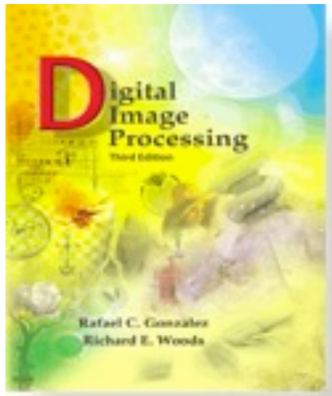
Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

1. sort the values
2. find the median

22 23 55 25 100 123 20
20 22 23 25 55 100 123

22 23 55 25 100 123 20 200
20 22 23 25 55 100 123 200



Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

1. sort the values
2. find the median

22 23 55 25 100 123 20
20 22 23 25 55 100 123

22 23 55 25 100 123 20 200
20 22 23 25 55 100 123 200

Median (50th percentile):

Def. Given a set of values, the median ξ is that value such that half of the values in the set are below ξ and half are over ξ.

1. sort the values
2. find the median

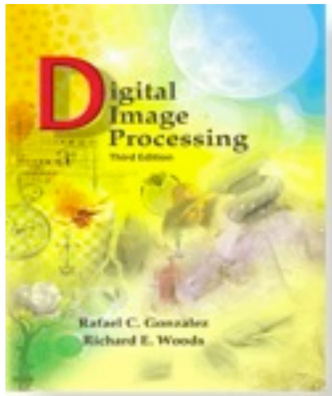
22 23 55 25 100 123 20

20 22 23 25 55 100 123

22 23 55 25 100 123 20 200

20 22 23 25 55 100 123 200

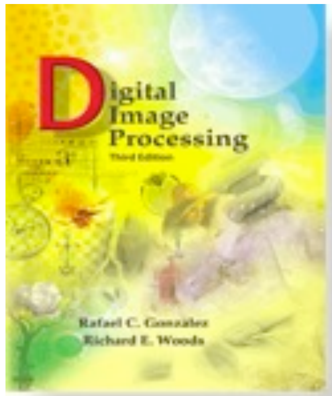
$$(25 + 55)/2 = 40 \text{ (median)}$$



How does the median filter work? Define the neighborhood *Sxy* and replace *f(x,y)* with the median in *Sxy*.

Max filter (100th percentile)

Min filter (0th percentile)

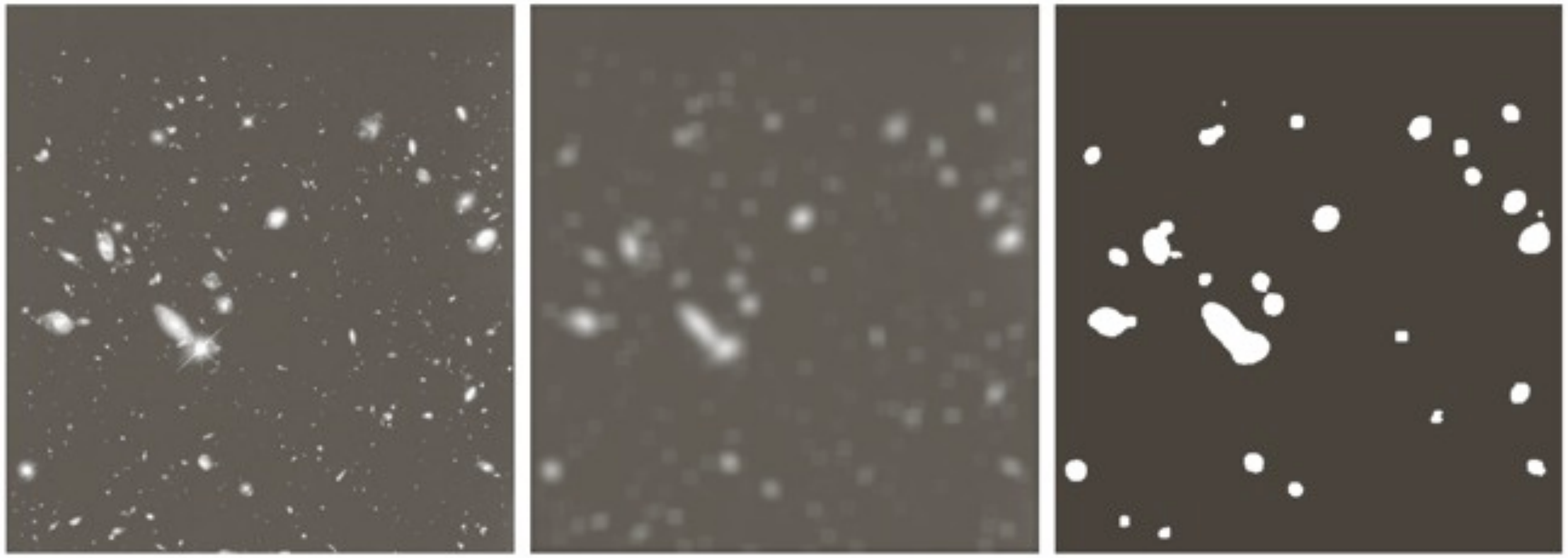


Chapter 3

Intensity Transformations & Spatial Filtering

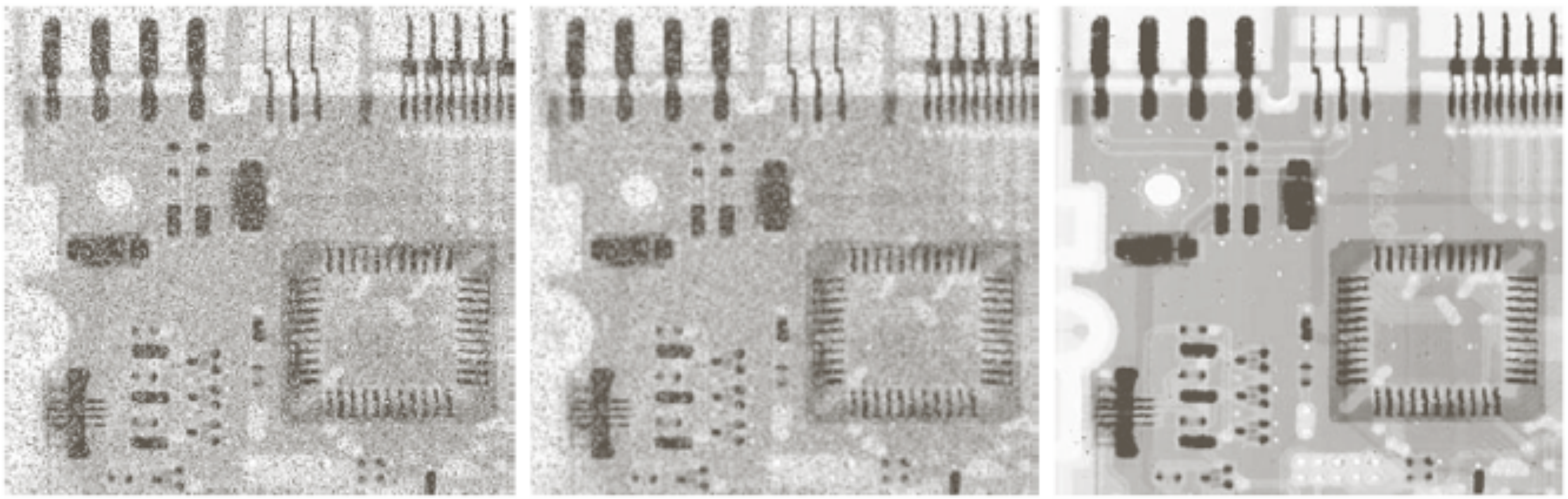
FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

a b
c d
e f



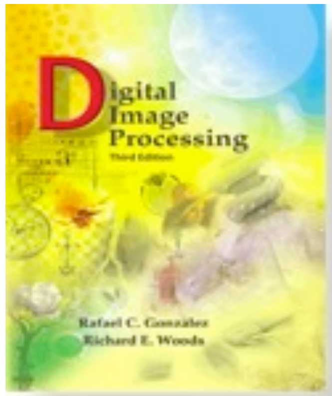
a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

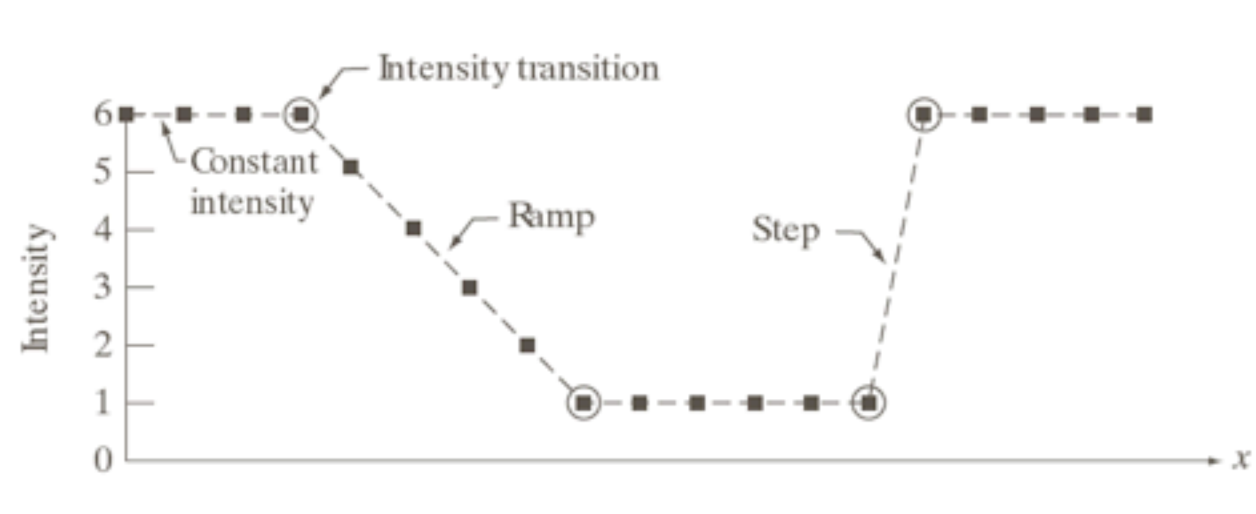


a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



Sharpening filters:



Enhance transitions in intensity.

Constant regions, ramps and steps.

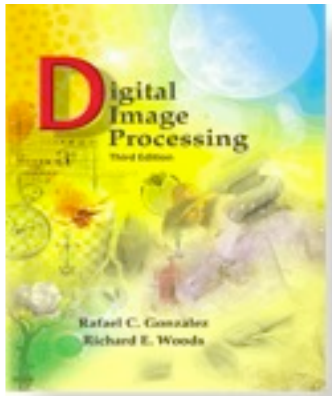
Ramp: joins 2 regions of constant intensity by several pixels
Step: joins 2 regions of constant intensity by 2 pixels.
Onset: the set of transition pixels

Derivative operator:

1. Must be 0 on constant regions
2. Is nonzero on ramps/ steps
3. Is nonzero on the onset of ramps/steps

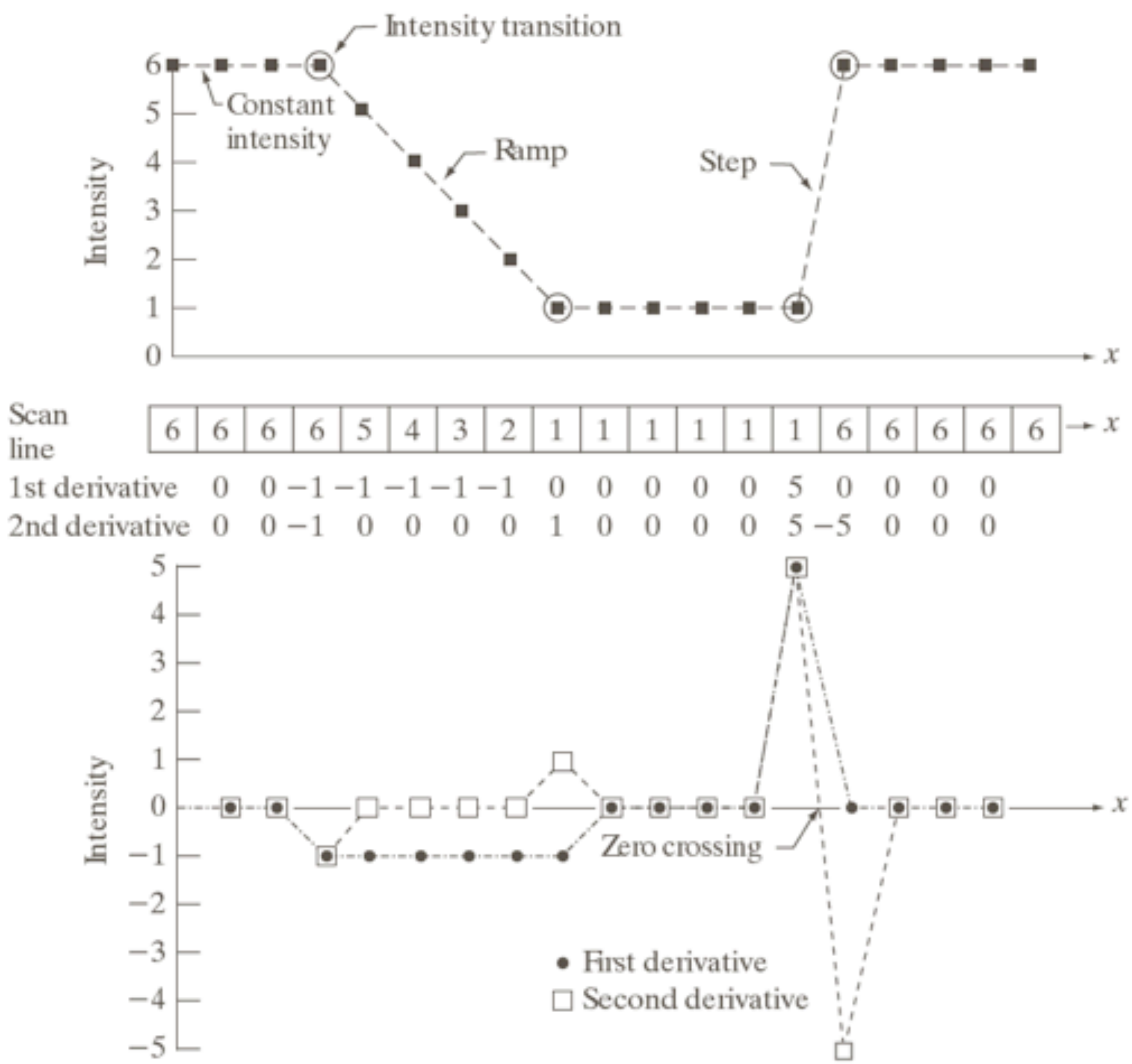
$$\frac{\partial f}{\partial x} \approx f(x+1) - f(x),$$

This approximation satisfies these assumptions



Second derivative operator:

1. must be 0 on constant regions
2. must be nonzero on the onset of steps/ramps
3. must be 0 along ramps with constant slope



a
b
c

FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

$$\frac{\partial^2 f}{\partial x^2} \approx f(x-1) - 2f(x) + f(x+1)$$

This discretization for the second derivative satisfies the requirements.

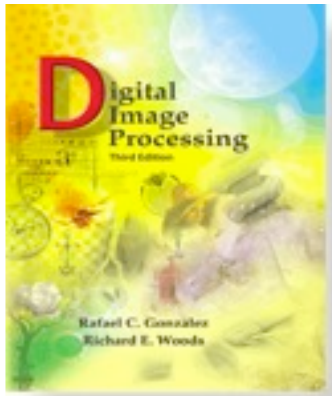
$$\frac{\partial^2 f}{\partial x^2} \approx f(x-1) - 2f(x) + f(x+1)$$

This discretization for the second derivative satisfies the requirements.

NB: Recall also the discretization of the gradient.

$$\frac{\partial f}{\partial x} \approx f(x+1) - f(x).$$

Since both the first and the second derivative must be 0 on constant regions, the weights of the corresponding mask must have zero sum.



Laplacian mask

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

$$\nabla^2 f \approx f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)$$

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

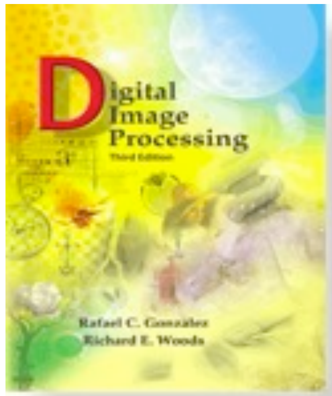
0	−1	0
−1	4	−1
0	−1	0

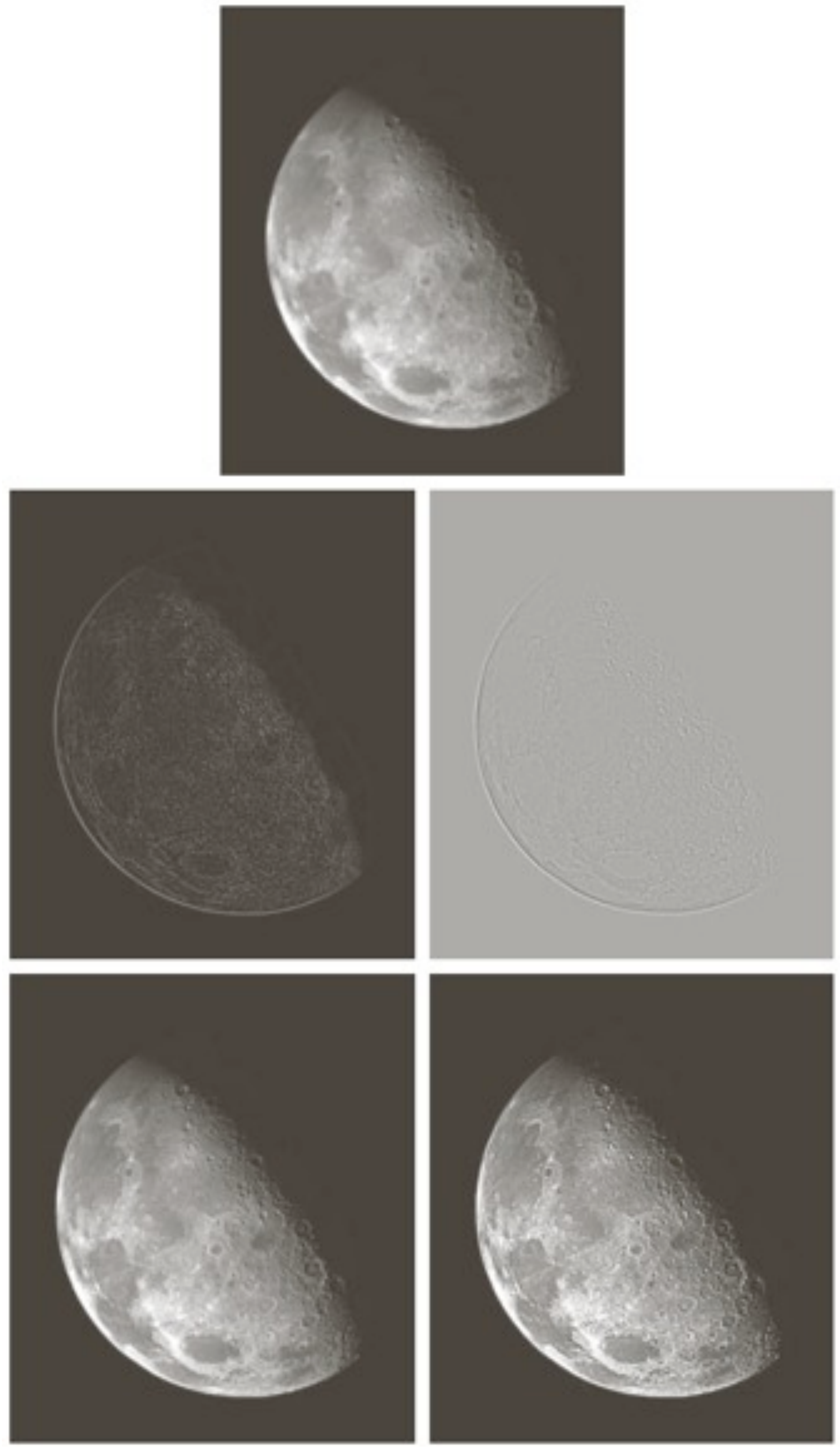
−1	−1	−1
−1	8	−1
−1	−1	−1

Using the Laplacian mask for sharpening:

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$$

Use c = <0 for schemes with negative central weight, c>0 for those with positive central weight.

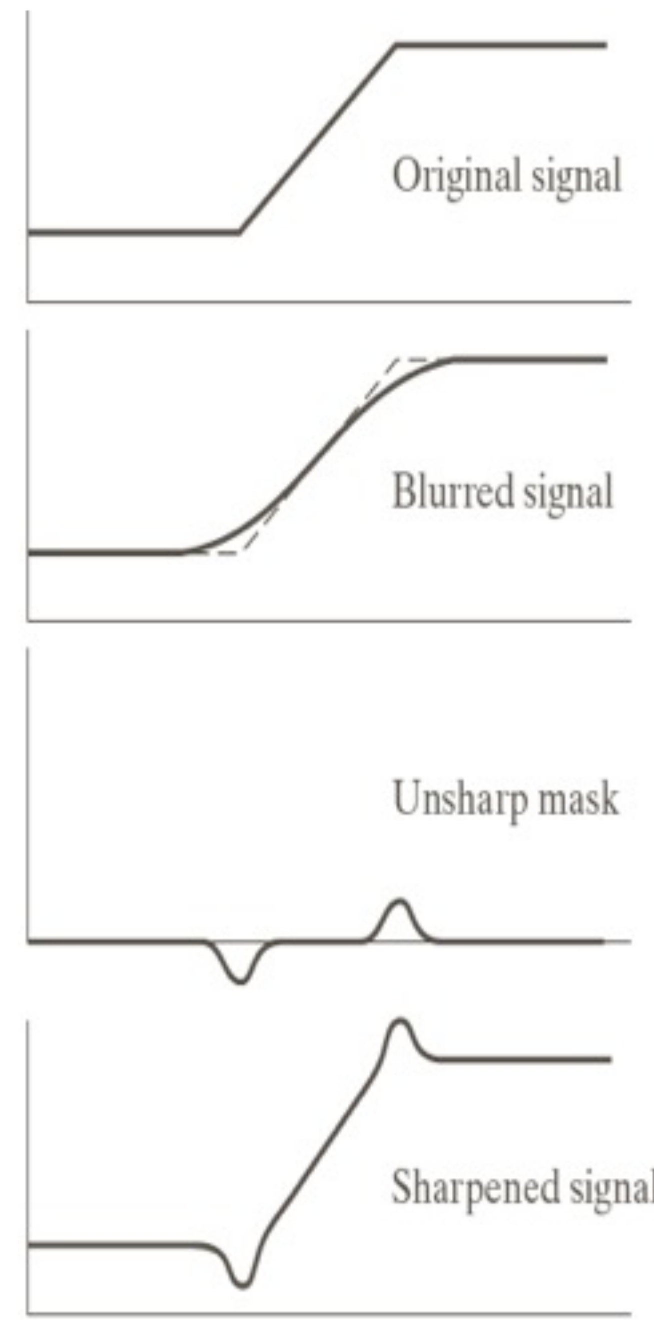




a
b c
d e

FIGURE 3.38 (a) Blurred image of the North Pole of the moon. (b) Laplacian without scaling. (c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a). (e) Result of using the mask in Fig. 3.37(b). (Original image courtesy of NASA.)

Unsharp masking and highboost filtering:



1. Blur original image

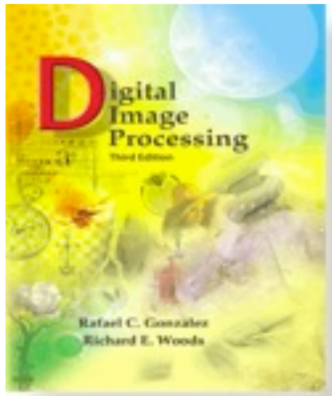
$$f(x,y) \rightarrow \bar{f}(x,y)$$

2. Subtract blurred from image to create a mask

$$g_{\text{mask}}(x,y) = f(x,y) - \bar{f}(x,y)$$

3. Add the mask to the original

$$g(x,y) = f + k * g_{\text{mask}}(x,y)$$



The global effect will be that of enhancing the edges.

Example: craters on Mars (image from NASA)