Spatial filtering and convolution

Intensity Transformations by Spatial Filtering (reminder)

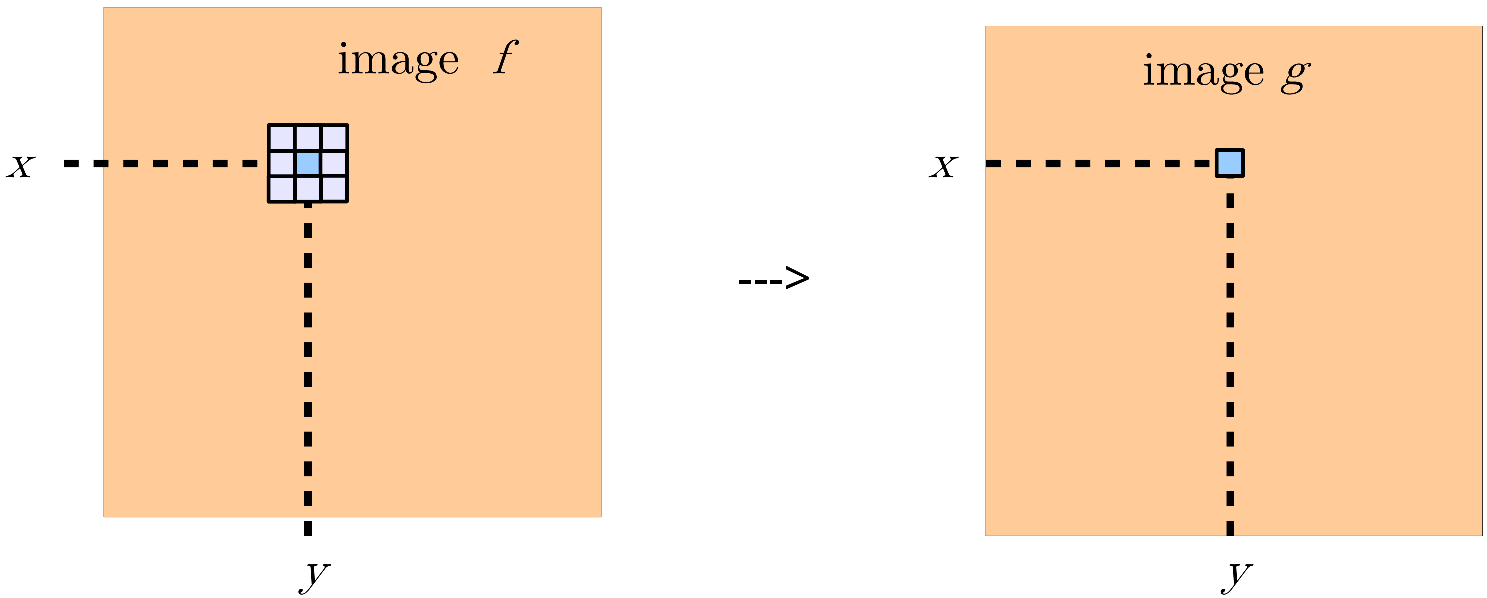
- Digital image: $f(x,y)$

where: $x = 0,\ldots,M-1$
and: $y = 0,\ldots,N-1$

- Want: $g(x,y) = T[f(x,y)]$

- T: Operator on intensities (*spatial* domain) of f.
- Operator acts on:
  - Single pixel at (x,y) ---> point-processing
  - Neighborhood around (x,y) ---> **spatial filtering**!

Spatial Filtering and convolution



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

Intensity Transformation by Spatial Filtering

Typical goal: Image enhancement by

- Reducing noise
- Sharpening edges

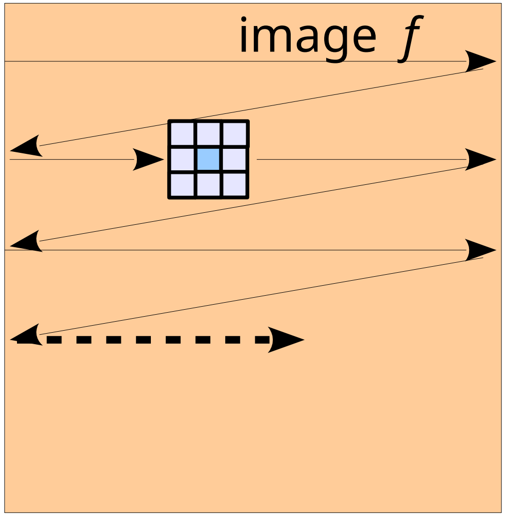
Filtering: applying a filter to the image

In spatial filtering:

1) We define a filter
2) We apply it to all the pixels

Defining a Filter:

- Neighborhood size, «window»
- *Operation* on the intensities in the neighborhood



Operation on each pixel can be seen as sliding the window over the image

(it can also be performed in parallel, with GPUs)

Two Types of Spatial Filtering

- Linear filtering (matrix/vectors operations)
- Non-linear filtering (example: thresholding, median filtering)

Linear filtering is done using convolution

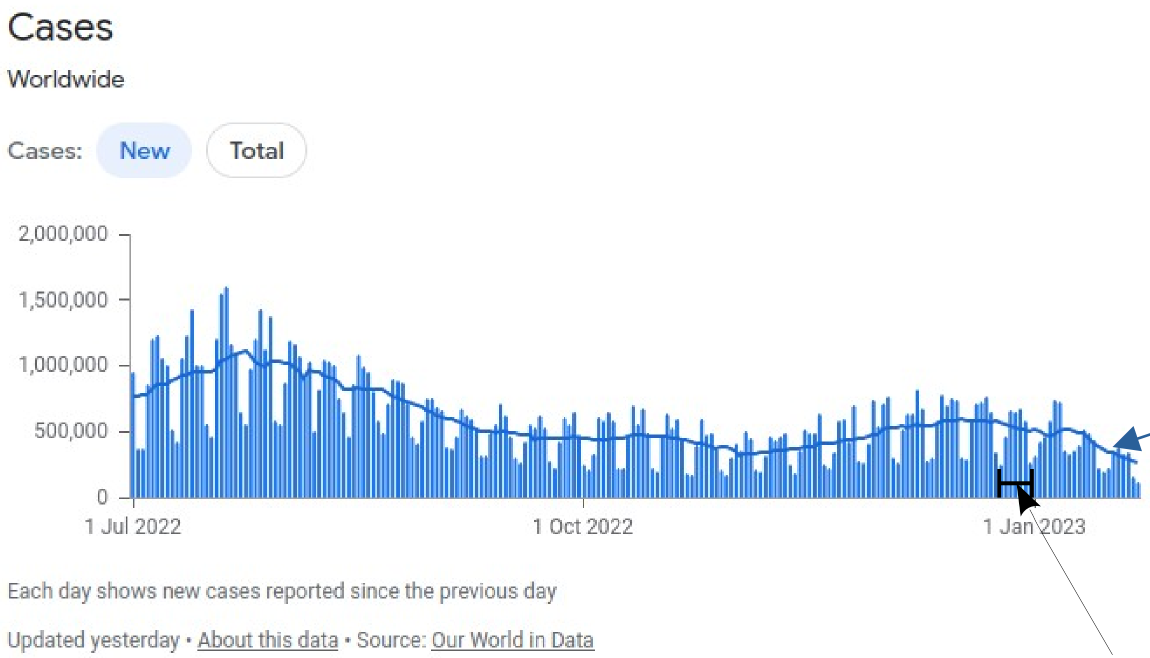
Linear filters

Linear filtering is done using the convolution operation. Let us see first what it is.

Convolution

Example: Moving average, convolution with a rectangular window

Covid



Average over 7 days

7 days window

Local mean value: $g(x) = \sum_{s=0}^{6} w(s) f(x-s)$

$$w(s) = 1/7$$

- *Convolution* («to roll together»)

Convolution of a filter with an *image* is common in digital image processing ---> linear spatial filtering

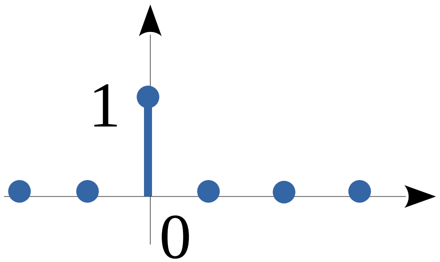
Convolution – definition and properties

$$(f*w)(x)=\sum_{k\in\mathbb{Z}} f(k)w(x-k)=\sum_{s\in\mathbb{Z}} f(x-s)w(s)$$

(By changing variable s=x-k)

Well defined f or w vanishing fast enough at infinity.

- commutative $(f*w)(x)=(w*f)(x)$
- Associative $(f*(w*g))(x)=((f*w)*g)(x)$
- Distributive $(f*(w+g))(x)=(f*w)(x)+(f*g)(x)$
- Neutral element: Kronecker delta $(f*\delta)(x)=f(x)$



2d convolution

For images we use the 2d Convolution:

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

Example (3 x 3) Mask

- Let a=1 and b=1

  $\mathbf{w}$ = [w(1,1), w(1,0), ... w(0,0), ... w(-1,-1)];

  $\mathbf{f}_{xy}$ = [f(x-1,y-1), f(x-1,y), ... f(x,y), ... f(x+1,y+1)];

  Then (convolution)

  g(x,y) = $\mathbf{w} \cdot \mathbf{f}_{xy}^{T}$; Inner-product!

- Remark: convolution and correleation of functions are very similar:

  Correlation $g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$
- convolution is correlation if w symmetric w(s,t)=w(-s,-t)

On images: example (3 x 3) Mask

- For each location (x,y) multiply overlapping elements and sum up

- *convolution*

$$g(x,y) = w(-1,-1)f(x+1,y+1) + w(-1,0)f(x+1,y) + \ldots + w(0,0)f(x,y) + \ldots$$

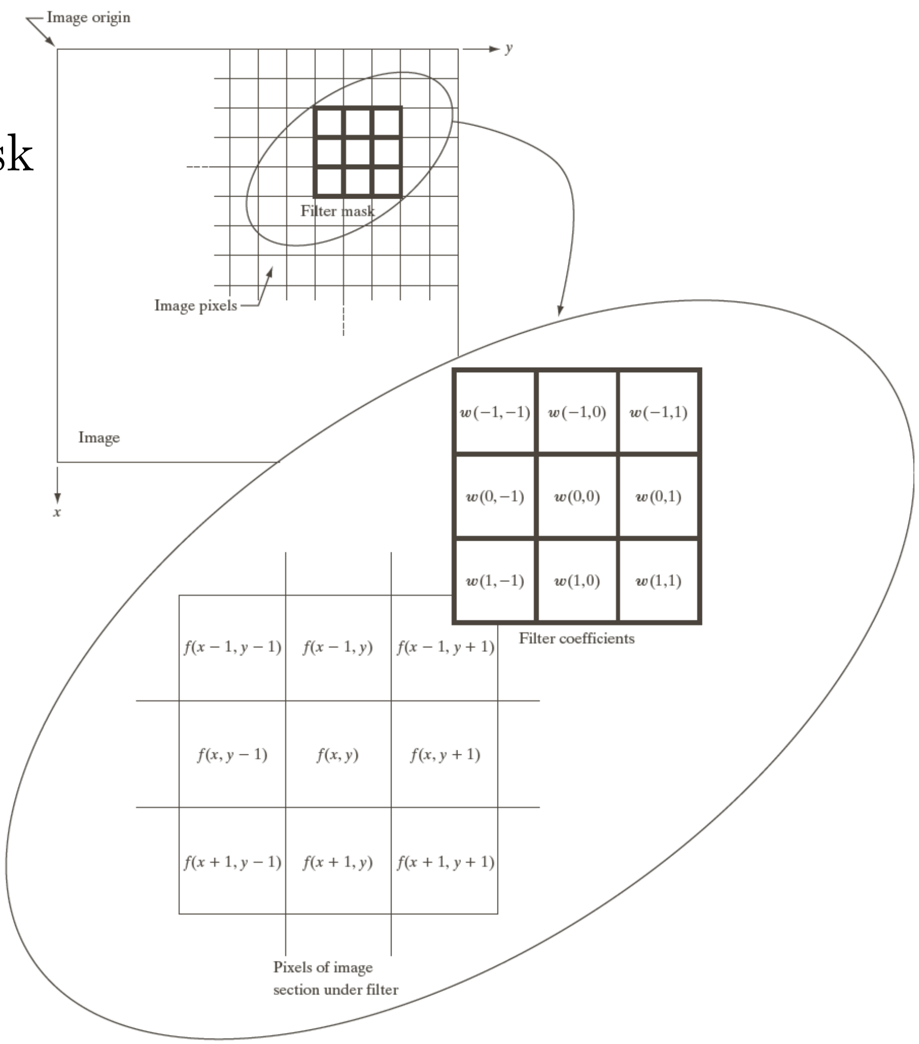
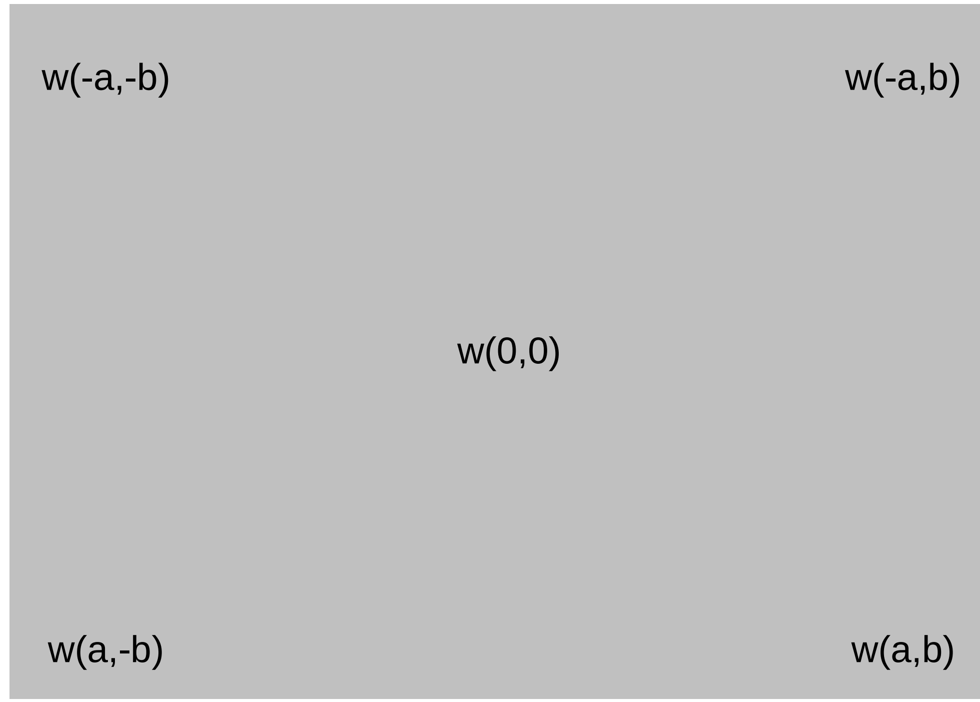


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.



Filter w:

- general (m x n) Mask
- Size:

m = 2a + 1

n = 2b + 1

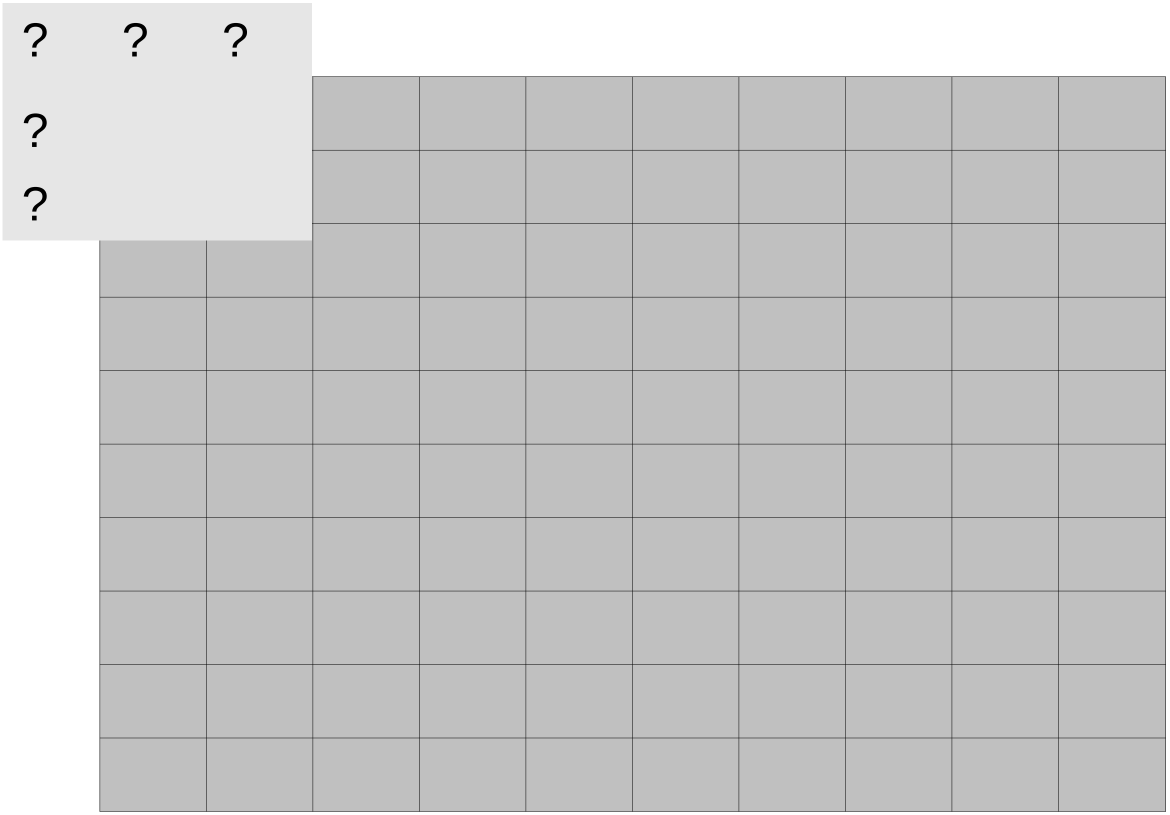
- The convolution has a double sum centered around the pixel at (x,y):

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

- Weights w can be fixed or learned

Problem with borders

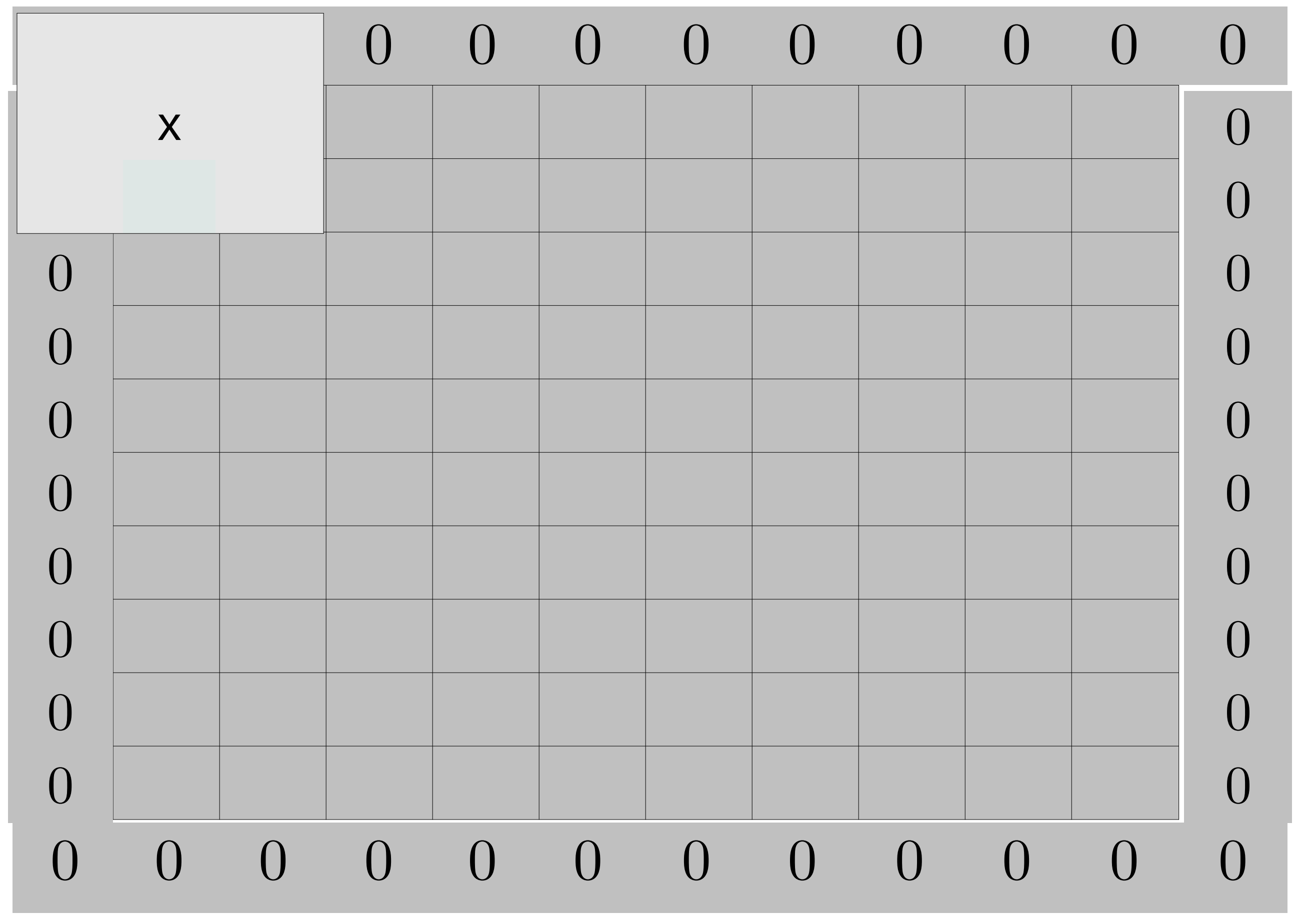
Filter centered around the pixel ? What happens at the border?



Solution:

Zero padding

Zero padding



Even better with 2 layers of zeros

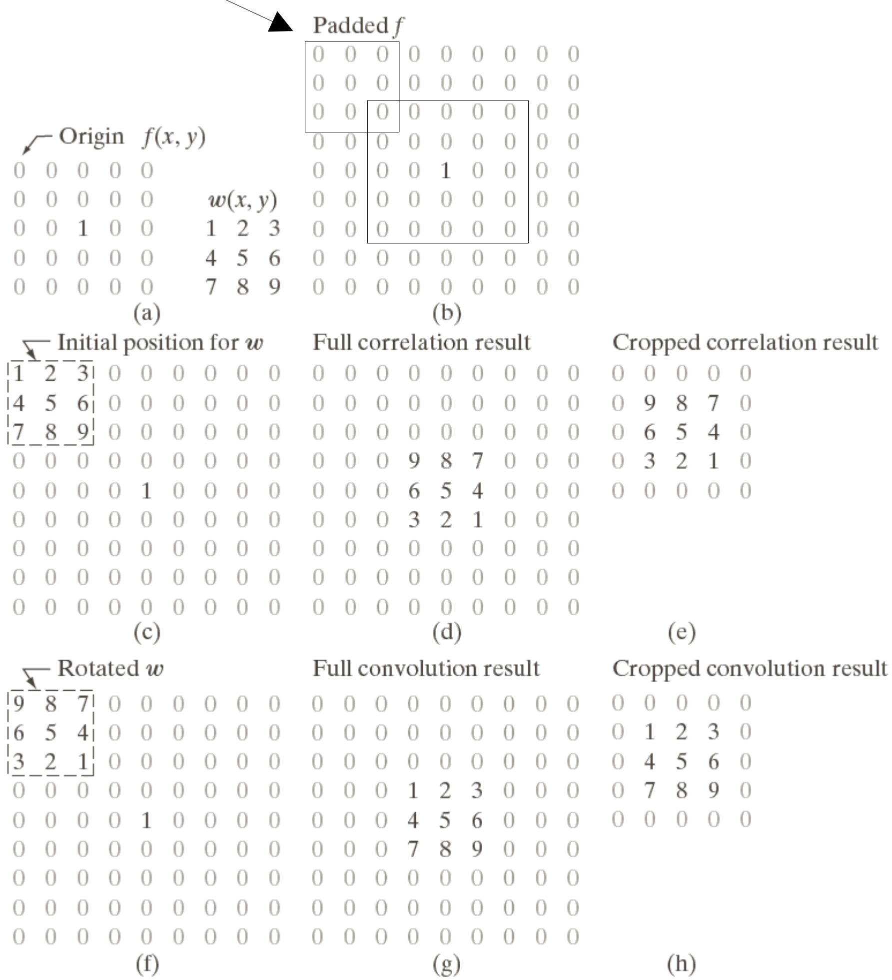


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

Averaging or smoothing Linear Filters

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Plain average — Weighted average

a b

FIGURE 3.32 Two 3×3 smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to 1 divided by the sum of the values of its coefficients, as is required to compute an average.

We will see a more precise definition of smoothing when we see the Fourier transform

Averaging or smoothing Linear Filters have a blurring effect

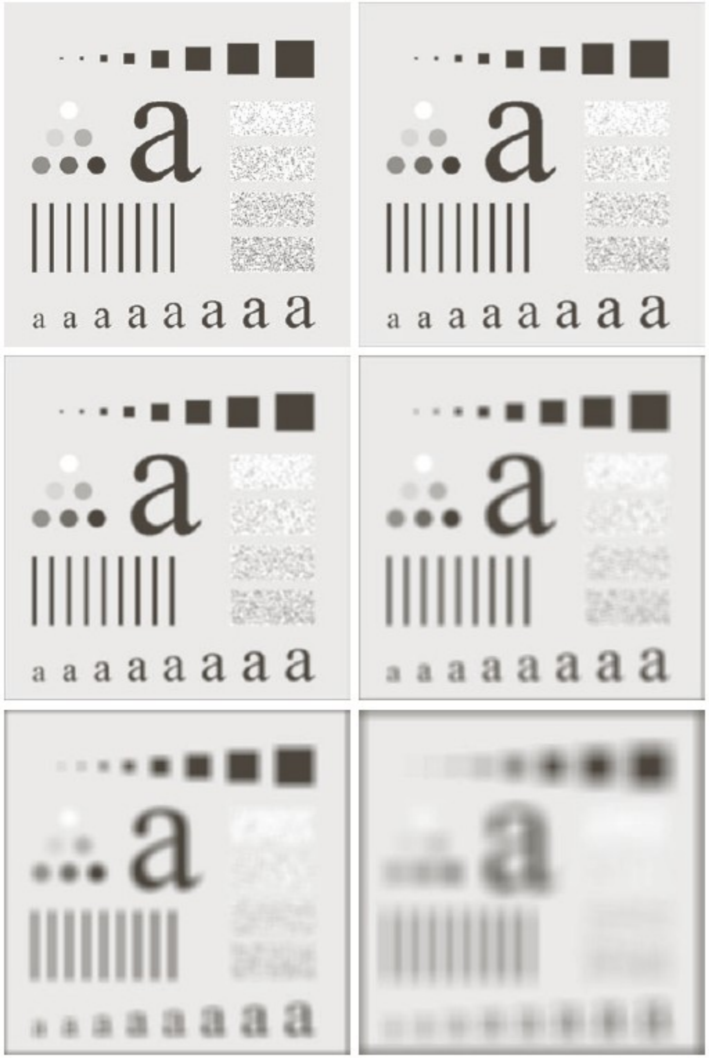
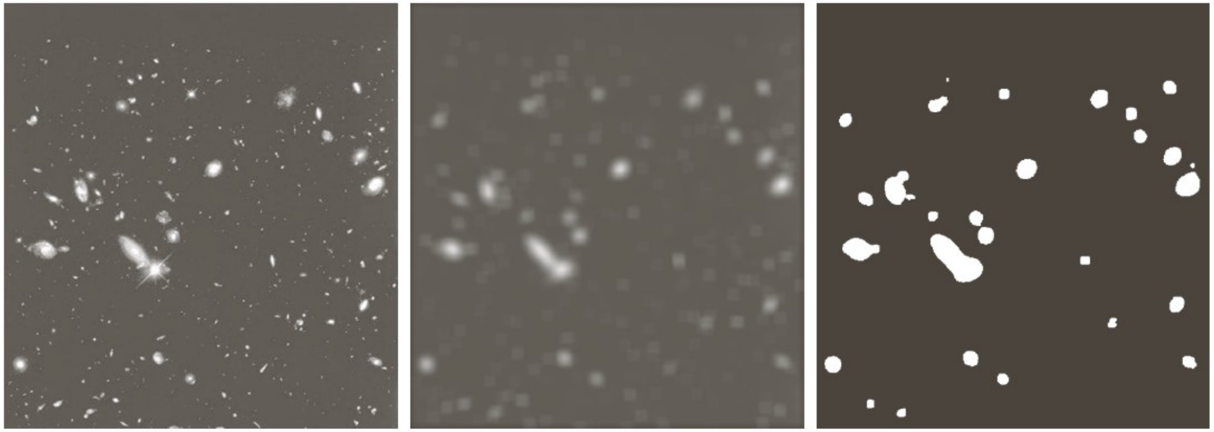


FIGURE 3.33 (a) Original image, of size 500×500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. 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

Smoothing Linear Filters

- Useful for filtering out small objects



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.)

Methods for Sharpening Edges

Unsharp Masking

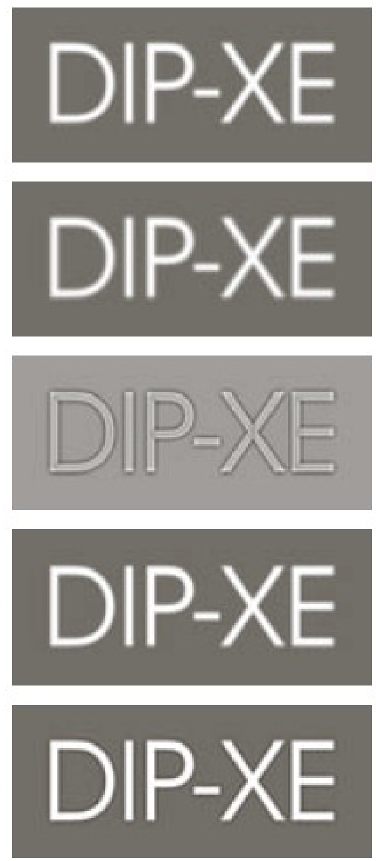
- Smooth/blur the image f: $\bar{f}(x,y)$
- Obtain difference: $g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$!!
- Add mask back: $g(x,y) = f(x,y) + k\, g_{mask}(x,y)$

- $k > 1$ ---> highboost filtering or oversharpening

https://www.adobe.com/creativecloud/photography/discover/unsharp-masking.html

$k = 1$
Unsharp masking

$k > 1$
Highboost filtering



a
b
c
d
e

FIGURE 3.40 (a) Original image. (b) Result of blurring with a Gaussian filter. (c) Unsharp mask. (d) Result of using unsharp masking. (e) Result of using highboost filtering.

Unsharp Masking Cont'd

Non-zero when there are variations

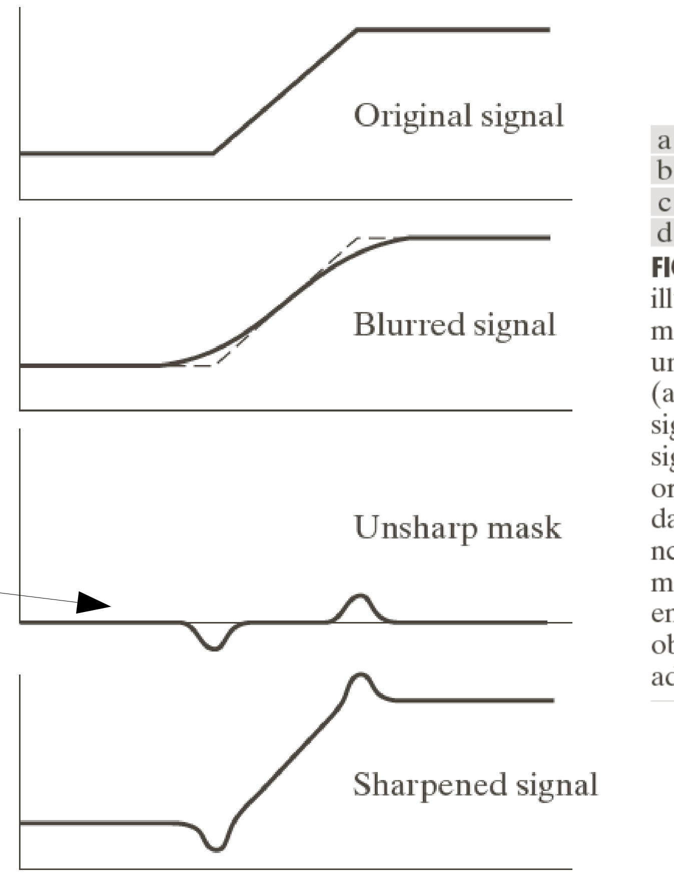
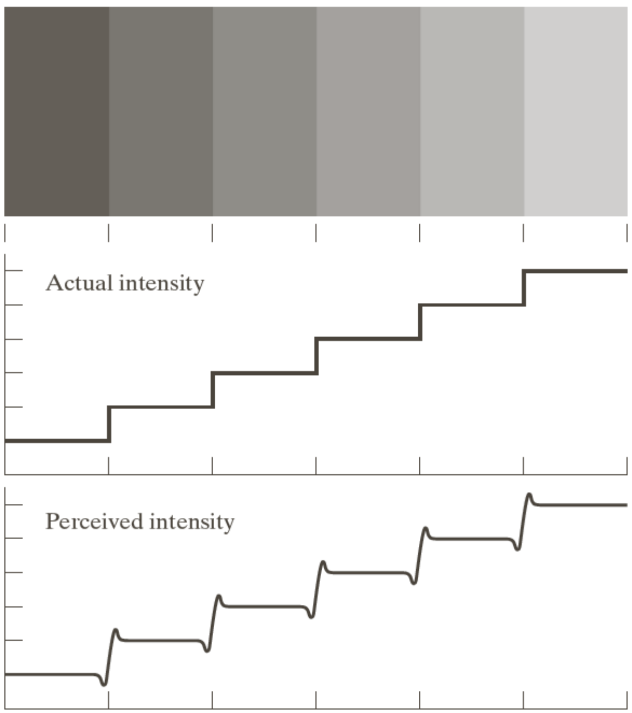
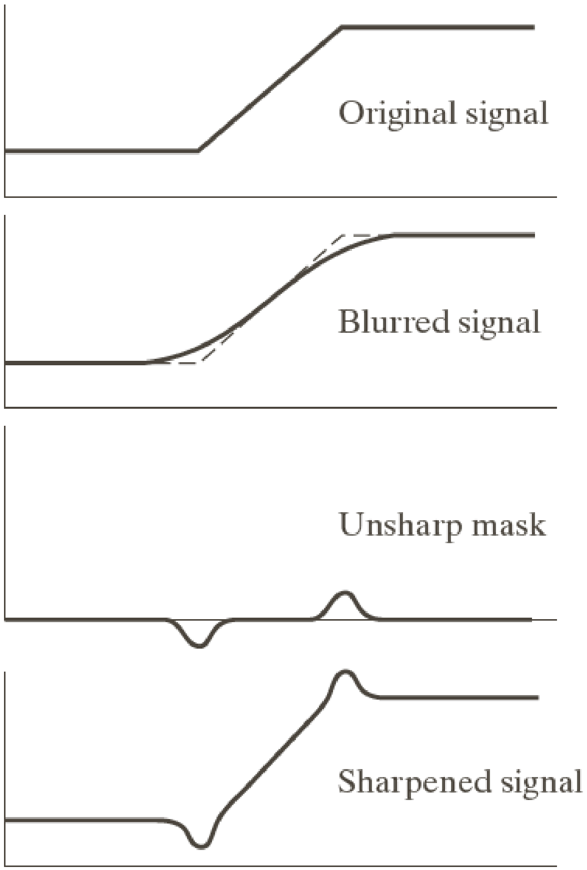


FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).



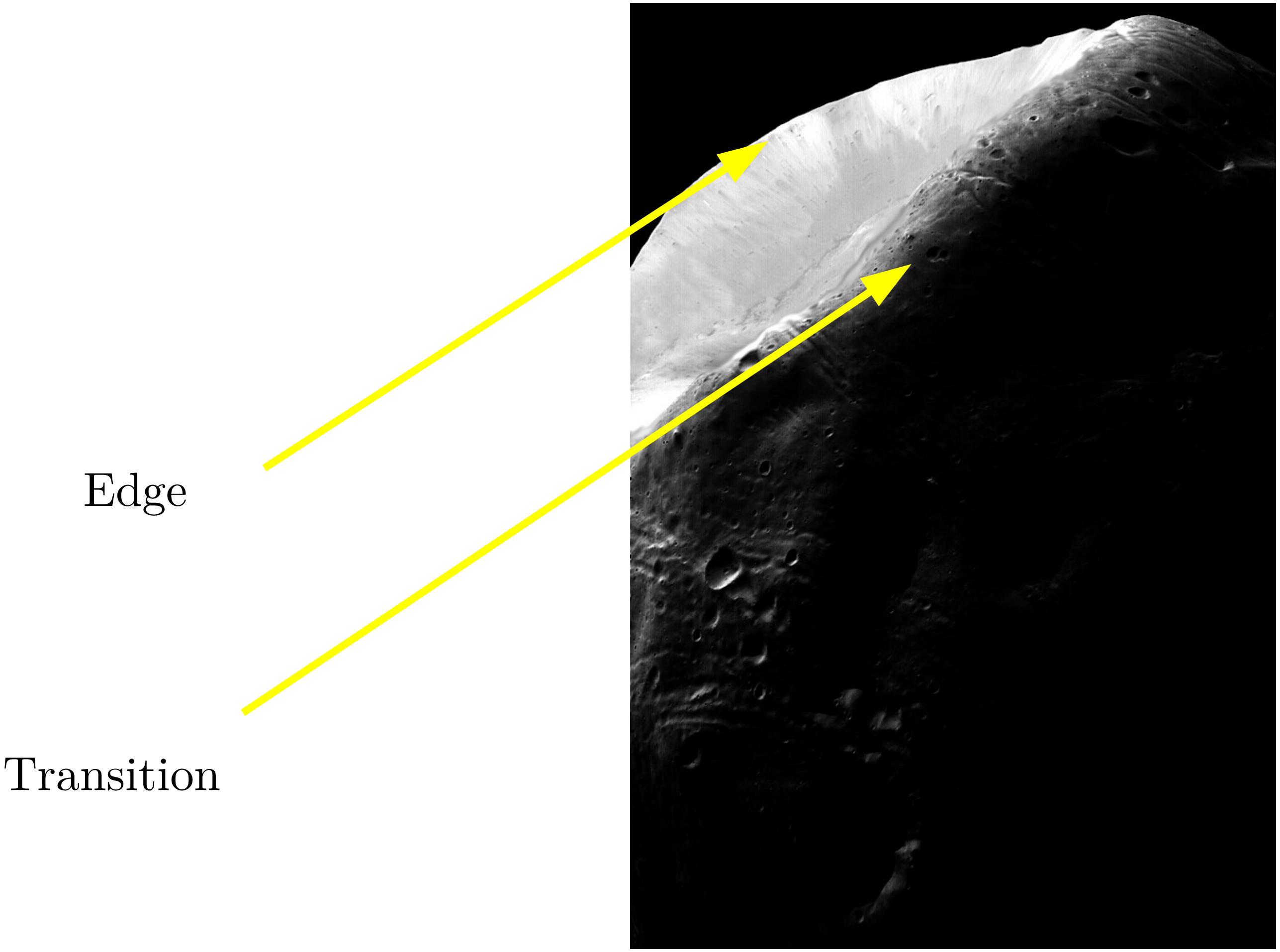


a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

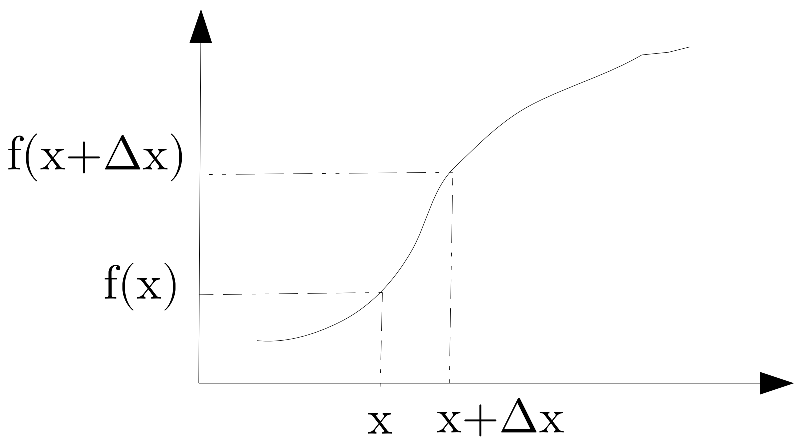
The Laplacian Filter

- Highlight transitions in intensity by utilizing derivatives
- 2D



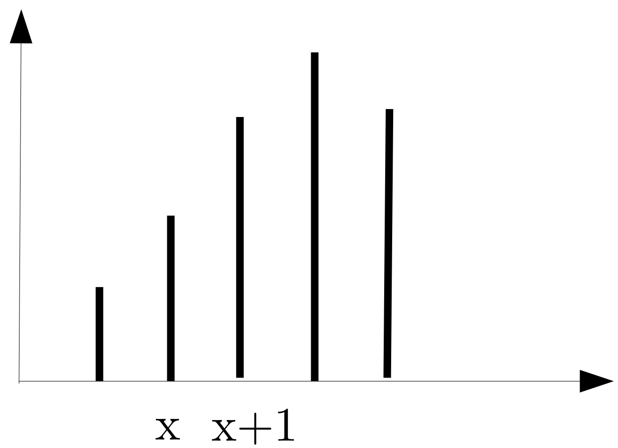
The Laplacian Filter

The gradient



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

The finite difference

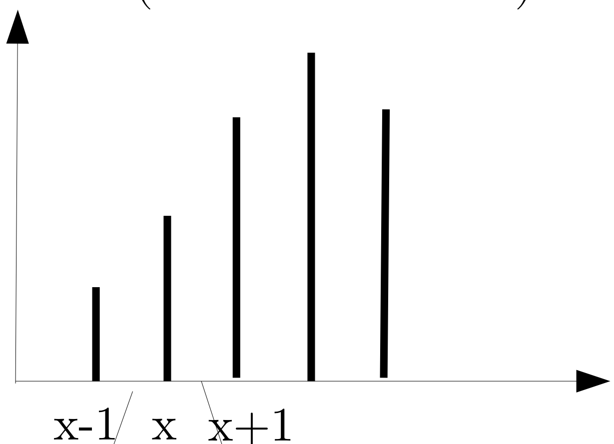


$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\nabla f(x) = f(x+1) - f(x)$$

The Laplacian Filter

The second derivative (discrete version)



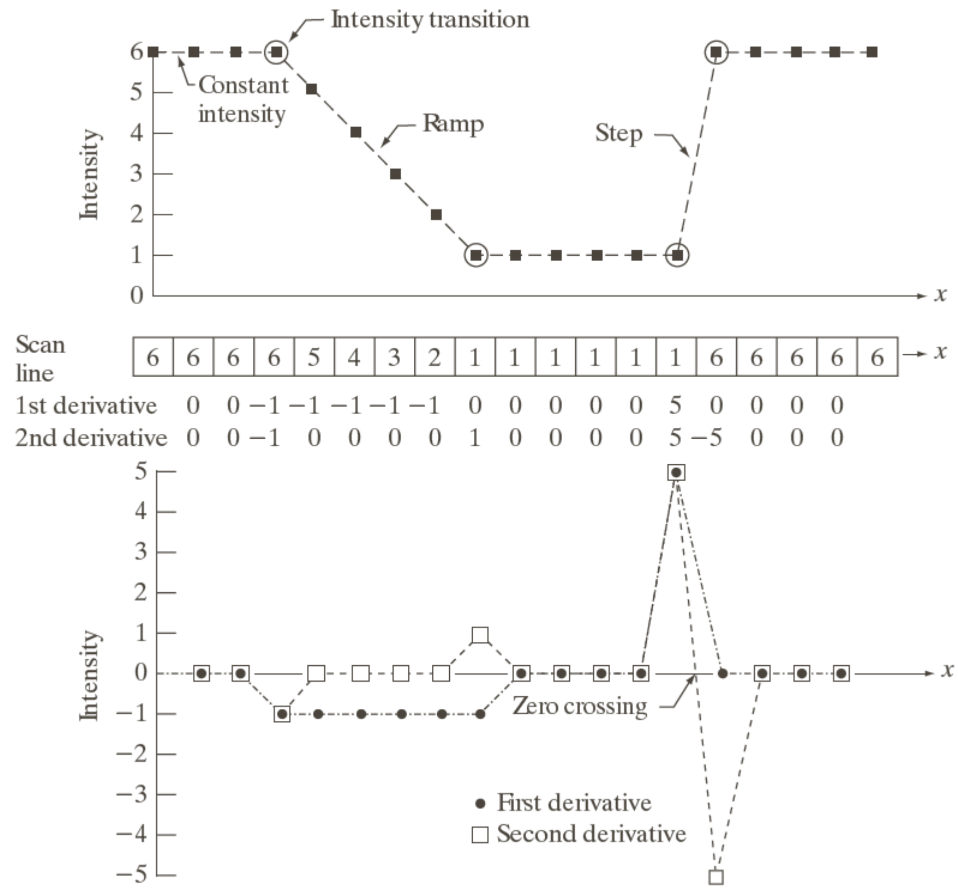
$f(x)-f(x-1)$ $f(x+1)-f(x)$ Variations of the function

Variation of the gradient

$$\nabla * \nabla f(x) = \Delta f(x) = (f(x+1)-f(x)) - (f(x)-f(x-1))$$

- The second derivative

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



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.

The Laplacian Filter

- 2^{nd} derivative operator for image

$$\Delta f(x,y) = \partial^2 f/\partial x^2 + \partial^2 f/\partial y^2$$

$$= f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$+ f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$= f(x+1,y) + f(x-1,y) + f(x,y+1)$$

$$+ f(x,y-1) - 4f(x,y)$$

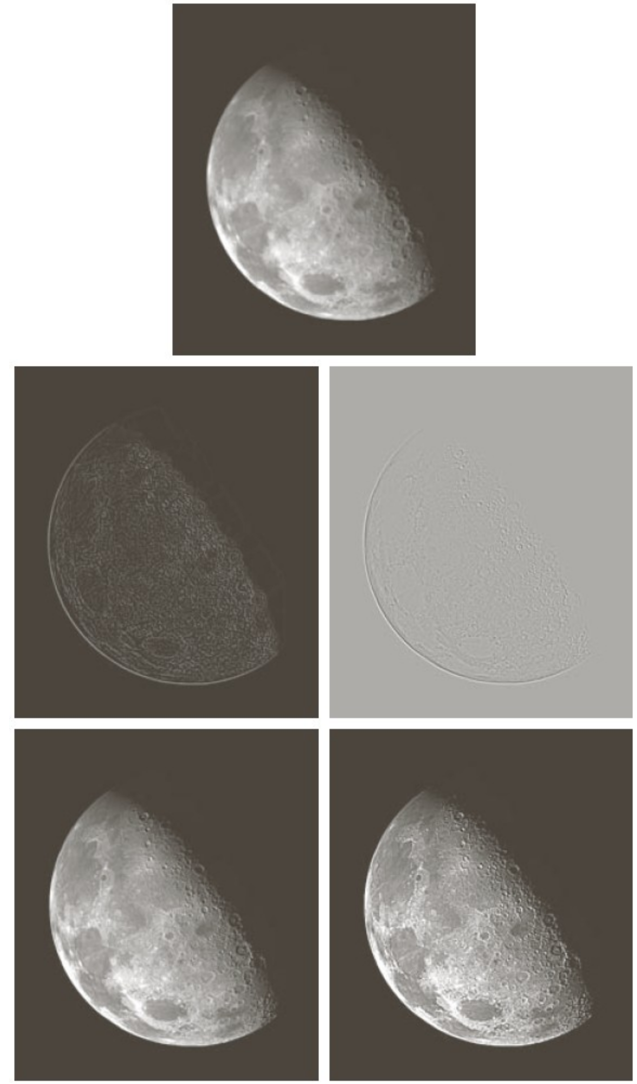
a	b
c	d

FIGURE 3.37 (a) Filter mask used to implement Eq. (3.6-6). (b) Mask used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other implementations of the Laplacian found frequently in practice.

0	1	0		1	1	1
1	−4	1		1	−8	1
0	1	0		1	1	1
0	−1	0		−1	−1	−1
−1	4	−1		−1	8	−1
0	−1	0		−1	−1	−1

The Laplacian Filter

$$g(x,y) = f(x,y) \pm \Delta f(x,y)$$



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.)

Scaled: converted to values in [0,255]

Nonlinear filters

(a few examples of them)

The Gradient Filter (Non-Linear)

- Gradient (vector)

$$\nabla f = \begin{pmatrix} g_x \\ g_y \end{pmatrix}$$

- Where $g_x = \partial f/\partial x$, $g_y = \partial f/\partial y$
  It gives 2 values per pixel

- We group them together: $M(x,y) = \mathrm{sqrt}(g_x^2 + g_y^2) \approx |g_x| + |g_y|$
  which makes the filter nonlinear

The Gradient Filter, second version

- How to approximate the gradient g_x using (3 x 3) filter?

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

.*

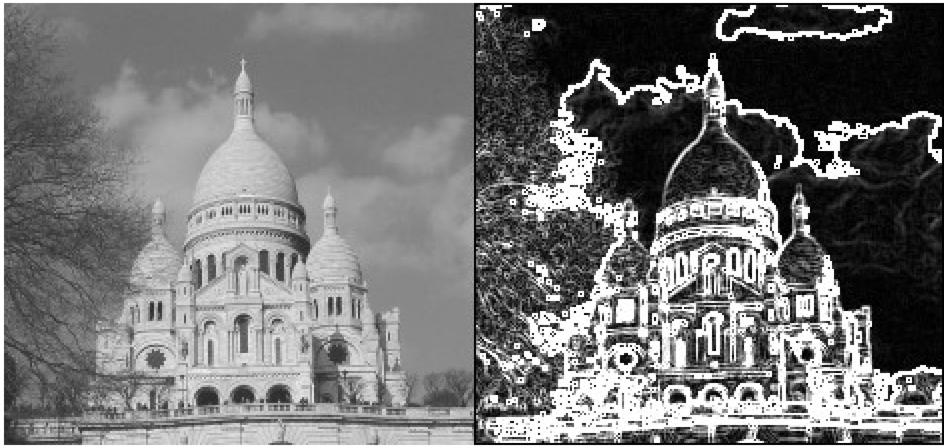
f(x-1,y-1)	f(x-1,y)	f(x-1,y+1)
f(x,y-1)	f(x,y)	f(x,y+1)
f(x+1,y-1)	f(x+1,y)	f(x+1,y+1)

More importance for the central pixel

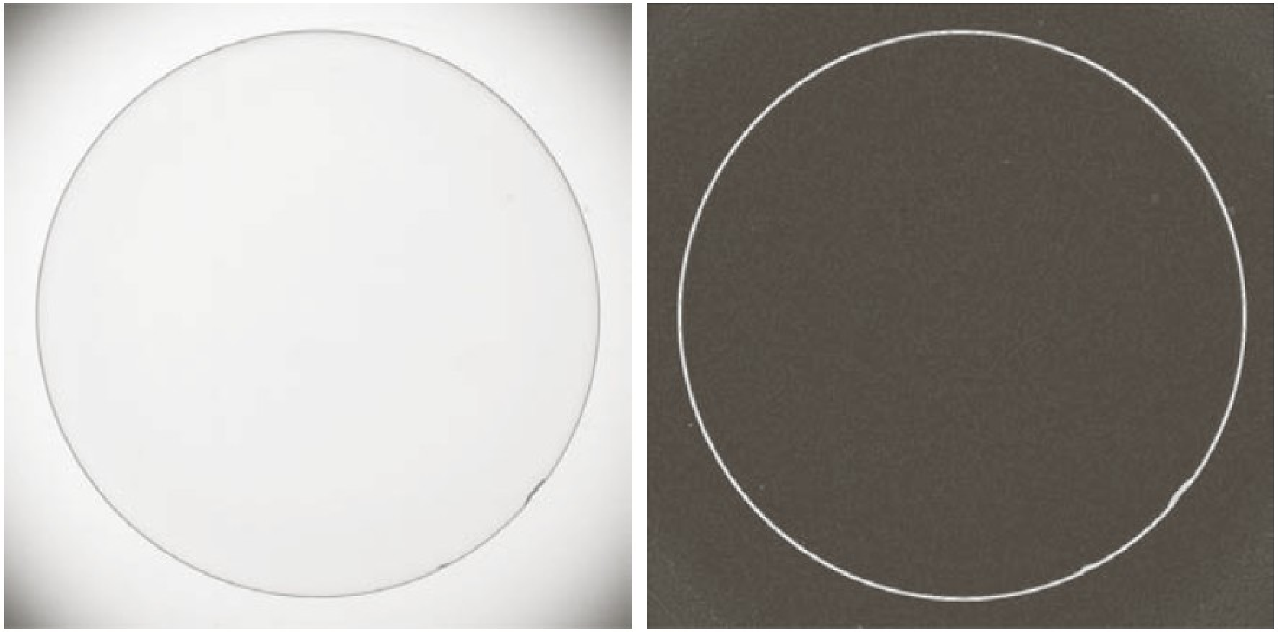
g_x = Weighted average «in front» - Weighted average «behind»

- Rotate 90 degrees for horizontal 'gradient' g_y.
- These filters are called **Sobel masks**

Sobel edge detection



From Wikipedia

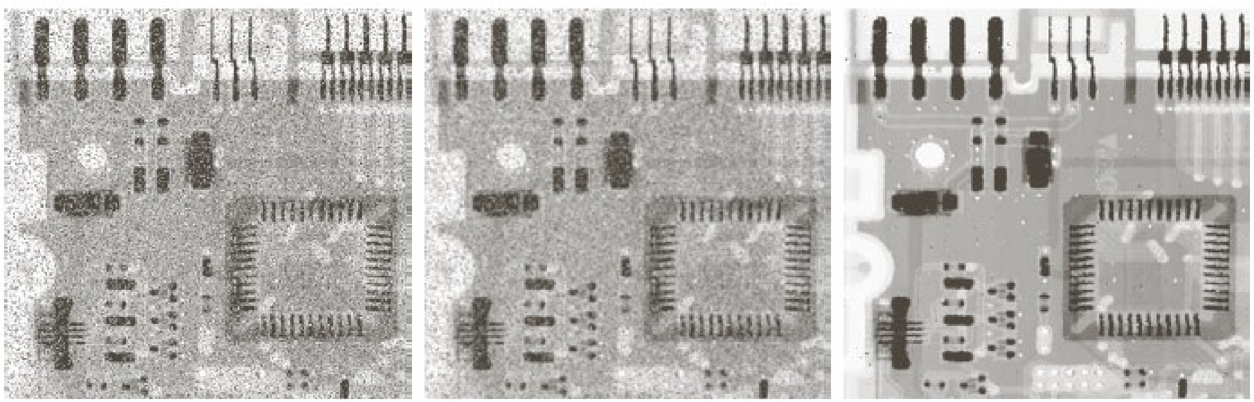


a b

FIGURE 3.42
(a) Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Pete Sites, Perceptics Corporation.)

Median filtering

- Denoising by Median Filtering (nonlinear filtering)
- Assign median of intensities in neighborhood around (x,y) in f to (x,y) in g
- Good for removing salt-and-pepper noise



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.)

2	3	2
4	50	5
5	2	3

Advantage of the median over the mean:
One pixel with a large deviation impact the mean but not the median,
Median is 3
Mean is 8.4