Digital Image Processing

Dr. Mubashir Ahmad (Ph.D.)

SPATIAL CORRELATION AND CONVOLUTION

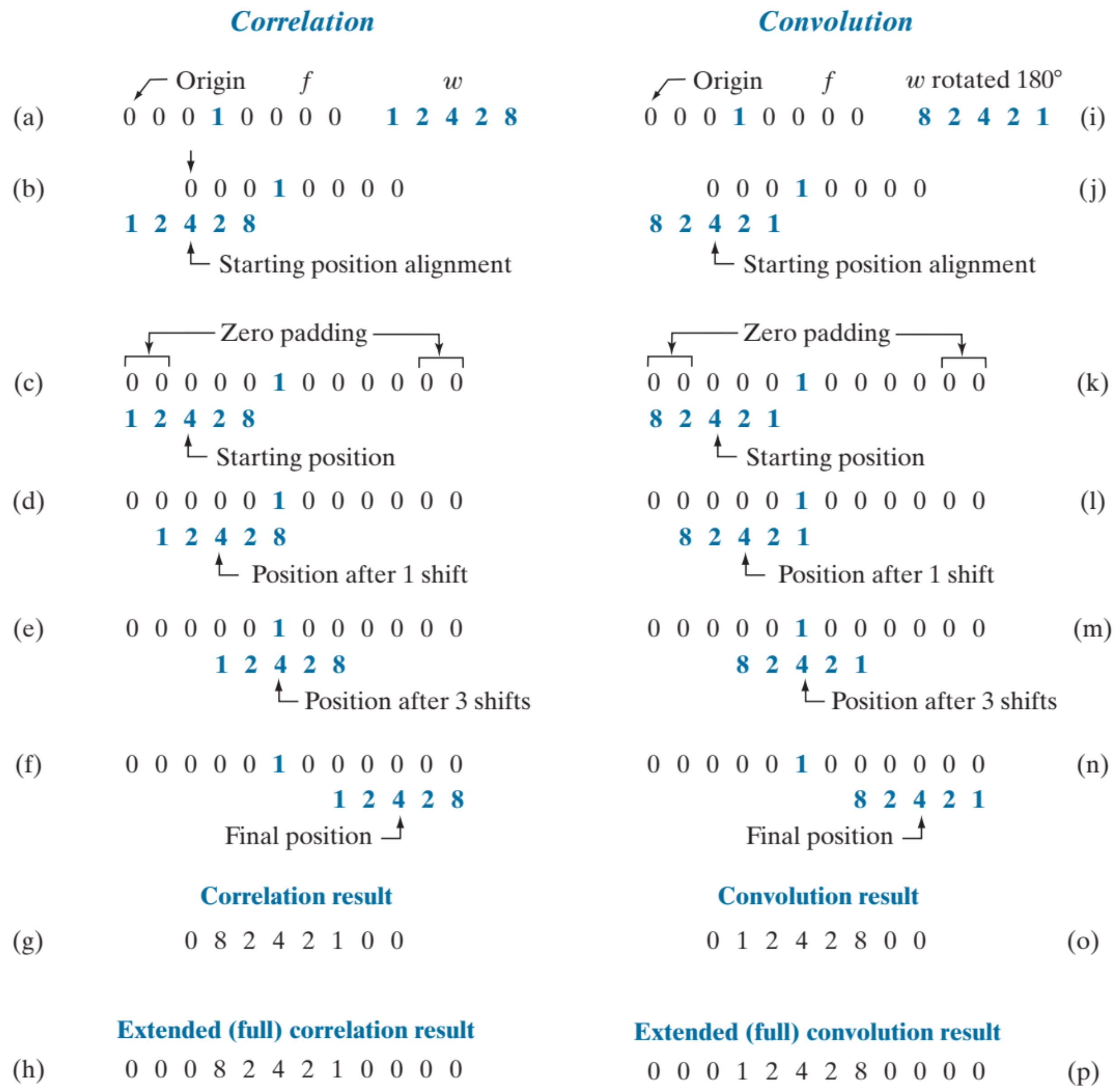
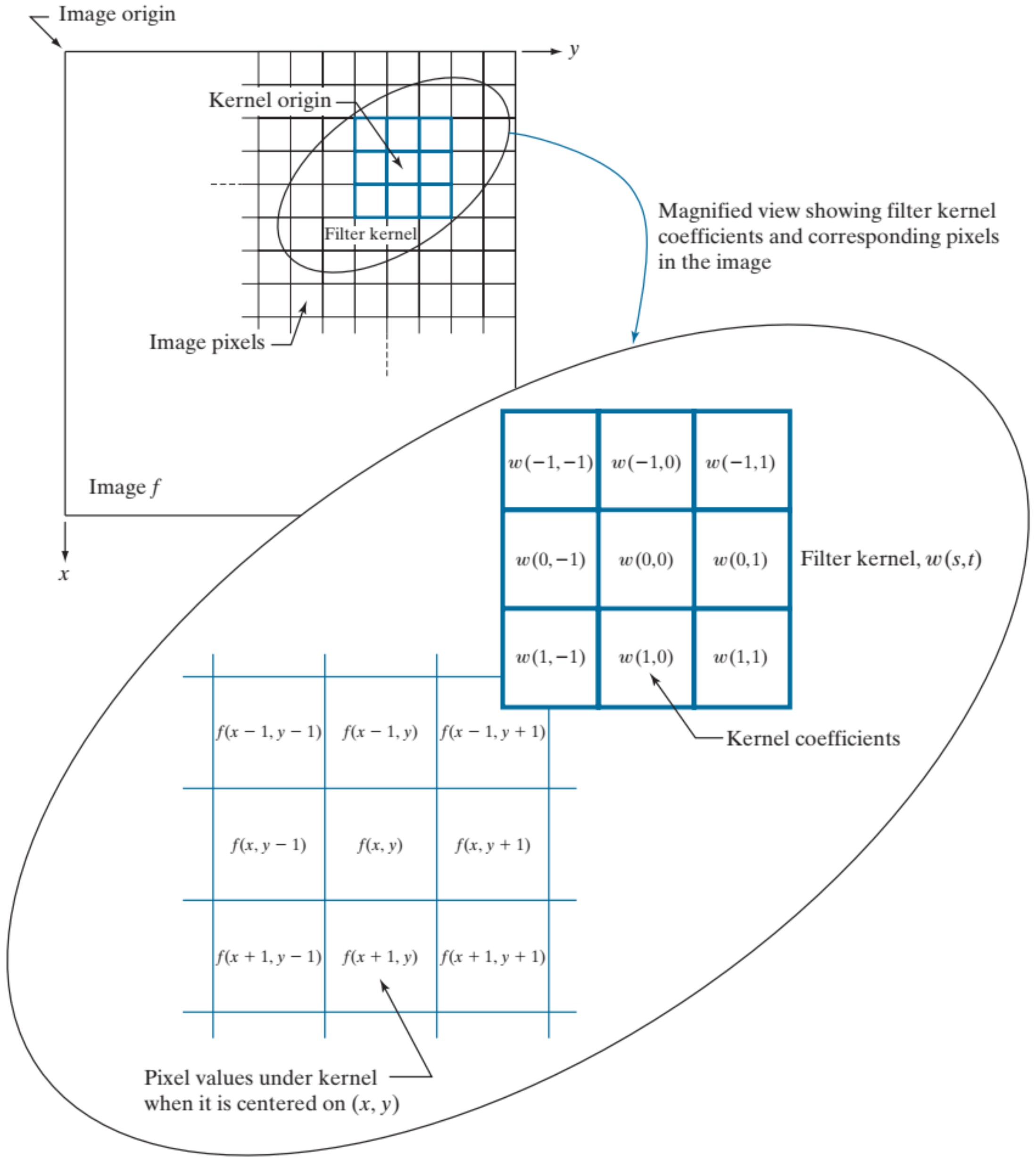

Correlation
Convolution
Origin
f
w
w rotated 180°
(a) 0 0 0 1 0 0 0 0 1 2 4 2 8
(i) 0 0 0 1 0 0 0 0 8 2 4 2 1
(b) 0 0 0 1 0 0 0 0
1 2 4 2 8
Starting position alignment
(j) 0 0 0 1 0 0 0 0
8 2 4 2 1
Starting position alignment
Zero padding
(c) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Starting position
(k) 0 0 0 0 0 1 0 0 0 0 0 0
8 2 4 2 1
Starting position
(d) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Position after 1 shift
(l) 0 0 0 0 0 1 0 0 0 0 0 0
8 2 4 2 1
Position after 1 shift
(e) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Position after 3 shifts
(m) 0 0 0 0 0 1 0 0 0 0 0 0
8 2 4 2 1
Position after 3 shifts
(f) 0 0 0 0 0 1 0 0 0 0 0 0
1 2 4 2 8
Final position
(n) 0 0 0 0 0 1 0 0 0 0 0 0
8 2 4 2 1
Final position
Correlation result
(g) 0 8 2 4 2 1 0 0
Convolution result
(o) 0 1 2 4 2 8 0 0
Extended (full) correlation result
(h) 0 0 0 8 2 4 2 1 0 0 0 0
Extended (full) convolution result
(p) 0 0 0 1 2 4 2 8 0 0 0 0

FIGURE 3.29 Illustration of 1-D correlation and convolution of a kernel, w, with a function f consisting of a discrete unit impulse. Note that correlation and convolution are functions of the variable x, which acts to *displace* one function with respect to the other. For the extended correlation and convolution results, the starting configuration places the rightmost element of the kernel to be coincident with the origin of f. Additional padding must be used.

FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.



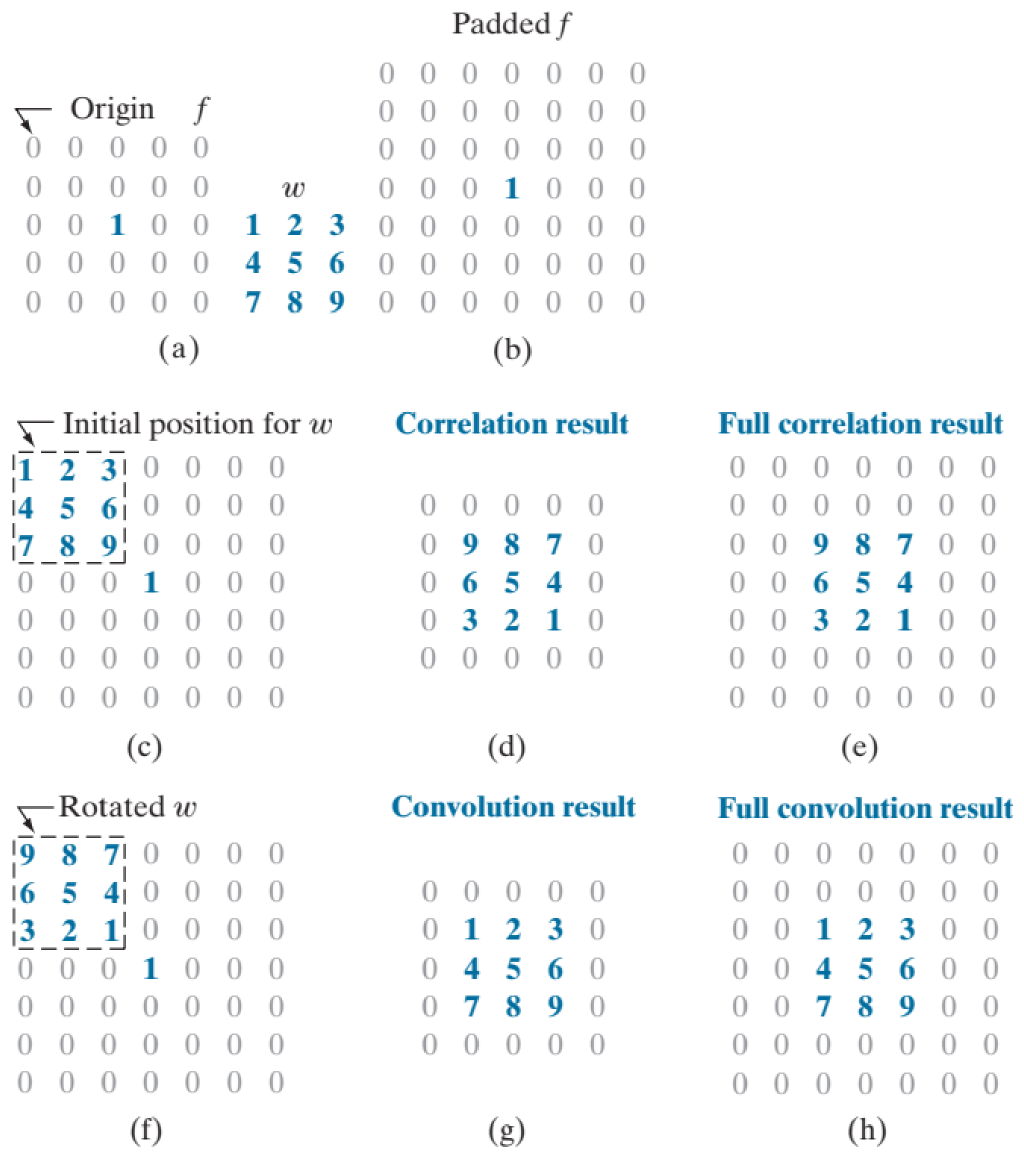
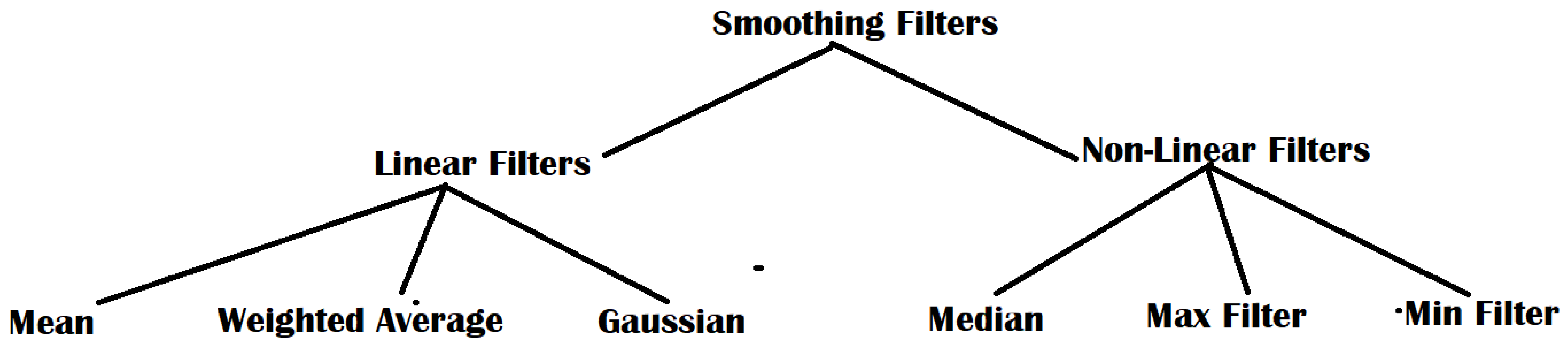


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse. The 0's are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of x and y. As these variable change, they *displace* one function with respect to the other. See the discussion of Eqs. (3-36) and (3-37) regarding full correlation and convolution.

Types of Smoothing Filters



Spatial filters classification filters

- *Spatial filters can be classified by effect:*
  - *Smoothing Filters: Aim to remove some small isolated detailed pixels by some form of averaging* of the pixels in the masked neighborhood. These are also called lowpass filters.

    Examples include Averaging, Weighted Average, Gaussian, order statistics filters etc.
  - *Sharpening Filters: aiming at highlighting some features such as edges or boundaries of image objects.*

    Examples include the Laplacian , Gradient filters etc.
- Spatial filters are also classified in terms of mask size (e.g. 3x3, 5x5, or 7x7).
- The maximum allowable mask size for spatial filters is 31x31.

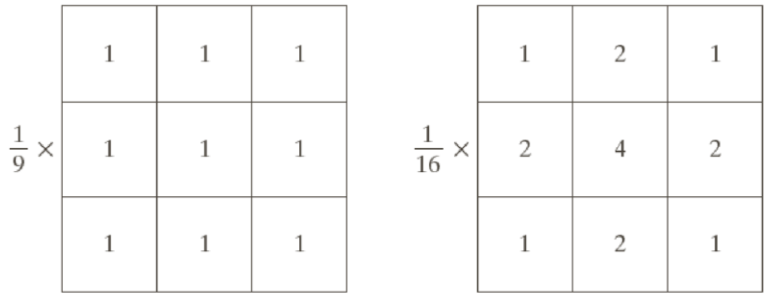
A weighted average filter – Example and averaging filter mask

For example, if $\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$ and

the neighbourhood subimage is $\begin{bmatrix}40 & 45 & 30\\ 41 & 50 & 20\\ 60 & 70 & 25\end{bmatrix}$

then the pixel value of the output image that corresponds to the central pixel in the neighbourhood is replaced with :

$$(40+2\times45+30+2\times41+4\times50+2\times20+60+2\times70+25)/16$$
$$=\text{int}(44.1875)=44.$$


$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{16}\times\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

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

Smoothing filters

- ***Smoothing Filters are particularly useful for blurring and noise reduction.***
- ***Smoothing filters work by reducing sharp transition in grey levels.***
- ***Noise reduction is accomplished by blurring with linear or non-linear filters (e.g. the order statistics filters).***
- **Beside reducing noise, smoothing filters often remove some significant features and reduce image quality.**
- **Increased filter size result in increased level of blurring which result in the reduction of noise and it also reduced image quality.**
- **Subtracting a blurred version of an image from the original may be used as a sharpening procedure.**

Gaussian smoothing filter

- A Gaussian smoothing filter is an image-filtering technique commonly used in image processing to reduce noise and smooth out an image. The filter works by convolving the image with a Gaussian function, a bell-shaped curve representing a probability distribution.
- To apply a Gaussian smoothing filter to an image, the image is convolved with the Gaussian function. The size of the Gaussian filter kernel is determined by the standard deviation of the Gaussian function. The kernel is centered on each pixel in the image, and the intensity value of each pixel is replaced by a weighted average of the intensities of the surrounding pixels, with the weights determined by the values of the Gaussian function at each pixel location.

Gaussian smoothing filter

- The Gaussian smoothing operator is **a 2-D convolution operator that is used to `blur' images and remove detail and noise**. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump.

Gaussian smoothing filter

- a case of weighted averaging
  – The coefficients are a 2D Gaussian.
  – Gives more weight at the central pixels and less weights to the neighbors.
  – The farther away the neighbors, the smaller the weight.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

How to implement the filter:

1. Design the kernel

The formula to design 2D Gaussian Kernel $= \frac{1}{2\pi\sigma^2} . e^{\frac{-(x^2+y^2)}{2\sigma^2}}$

Let us consider the standard deviation sigma = 0.6 of 3*3 filter

$$\frac{1}{2\pi\sigma^2} = \frac{1}{2\times 3.14\times 0.6\times 0.6} = \frac{1}{2.2619}$$

4. The width of the kernel is X = 3 and the height of the kernel is Y =3

i.e $X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ and $Y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

$$\frac{-(x^2+y^2)}{2\sigma^2} = \begin{bmatrix} -2.7778 & -1.3889 & -2.7778 \\ -1.3889 & 0 & -1.3889 \\ -2.7778 & -1.3889 & -2.7778 \end{bmatrix}$$

$$e^{-\frac{x^2+y^2}{2\sigma^2}} = \begin{bmatrix} 0.062 & 0.25 & 0.062 \\ 0.25 & 1 & 0.25 \\ 0.062 & 0.25 & 0.062 \end{bmatrix}$$

Filter designing

$$\frac{1}{2\pi\sigma^2}.e^{\frac{-(x^2+y^2)}{2\sigma^2}} = \frac{1}{2.2619} \times \begin{bmatrix} 0.062 & 0.25 & 0.062 \\ 0.25 & 1 & 0.25 \\ 0.062 & 0.25 & 0.062 \end{bmatrix}$$

Now the Gaussian Kernel = $\begin{bmatrix} 0.0275 & 0.1102 & 0.0275 \\ 0.1102 & 0.4421 & 0.1102 \\ 0.0275 & 0.1102 & 0.0275 \end{bmatrix}$

2. Convolve the kernel and the local region in the image

Consider a local region from an Image:

$$\begin{bmatrix} 72 & 68 & 88 & 159 \\ 69 & 66 & 87 & 162 \\ 70 & 66 & 83 & 161 \\ 70 & 66 & 78 & 154 \end{bmatrix}$$

Convolve the selected part and the kernel:

$$\begin{bmatrix} 68 & 88 & 159 \\ 66 & 87 & 162 \\ 66 & 83 & 161 \end{bmatrix} .* \begin{bmatrix} 0.0275 & 0.1102 & 0.0275 \\ 0.1102 & 0.4421 & 0.1102 \\ 0.0275 & 0.1102 & 0.0275 \end{bmatrix} = \begin{bmatrix} 1.8692 & 9.7009 & 4.3706 \\ 7.2757 & 38.4624 & 17.8585 \\ 1.8142 & 9.1497 & 4.4256 \end{bmatrix}$$

Add up the values in the vector: 1.8692 + 9.7009 + 4.3706 + 7.2757 + 38.4624 + 17.8585 + 1.84142 + 9.1497 + 4.4256 = 94.9269

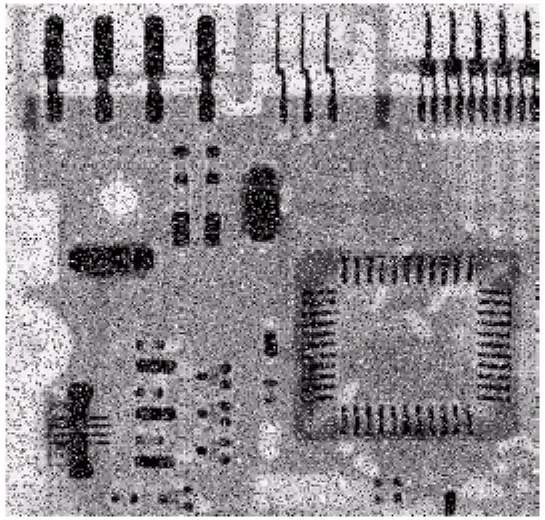
$$\begin{bmatrix} [\,] & [\,] & [\,] & [\,] \\ [\,] & [\,] & 94.9269 & [\,] \\ [\,] & [\,] & [\,] & [\,] \\ [\,] & [\,] & [\,] & [\,] \end{bmatrix}$$

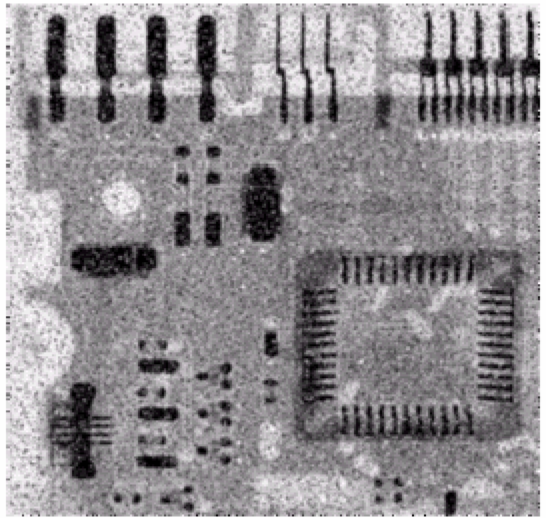
Perform the above operation on all the parts of the region. The final result after Gaussian filter is

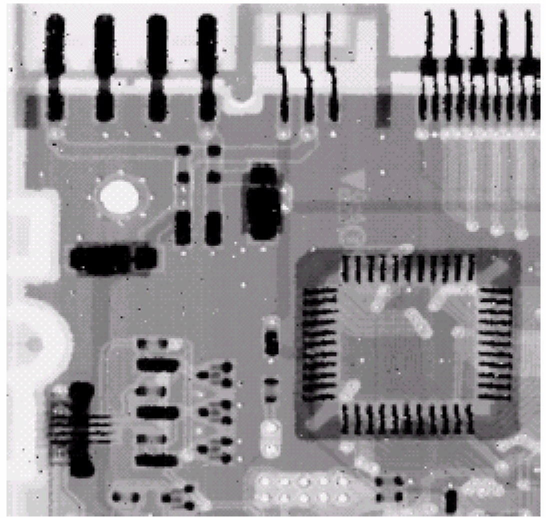
$$\begin{bmatrix} 48.7478 & 59.2645 & 79.7865 & 100.2444 \\ 57.1176 & 69.7512 & 94.9296 & 121.1870 \\ 57.1740 & 68.9526 & 92.2220 & 119.6981 \\ 47.7534 & 59.9750 & 74.1254 & 96.7113 \end{bmatrix}$$

Order Statistical filters

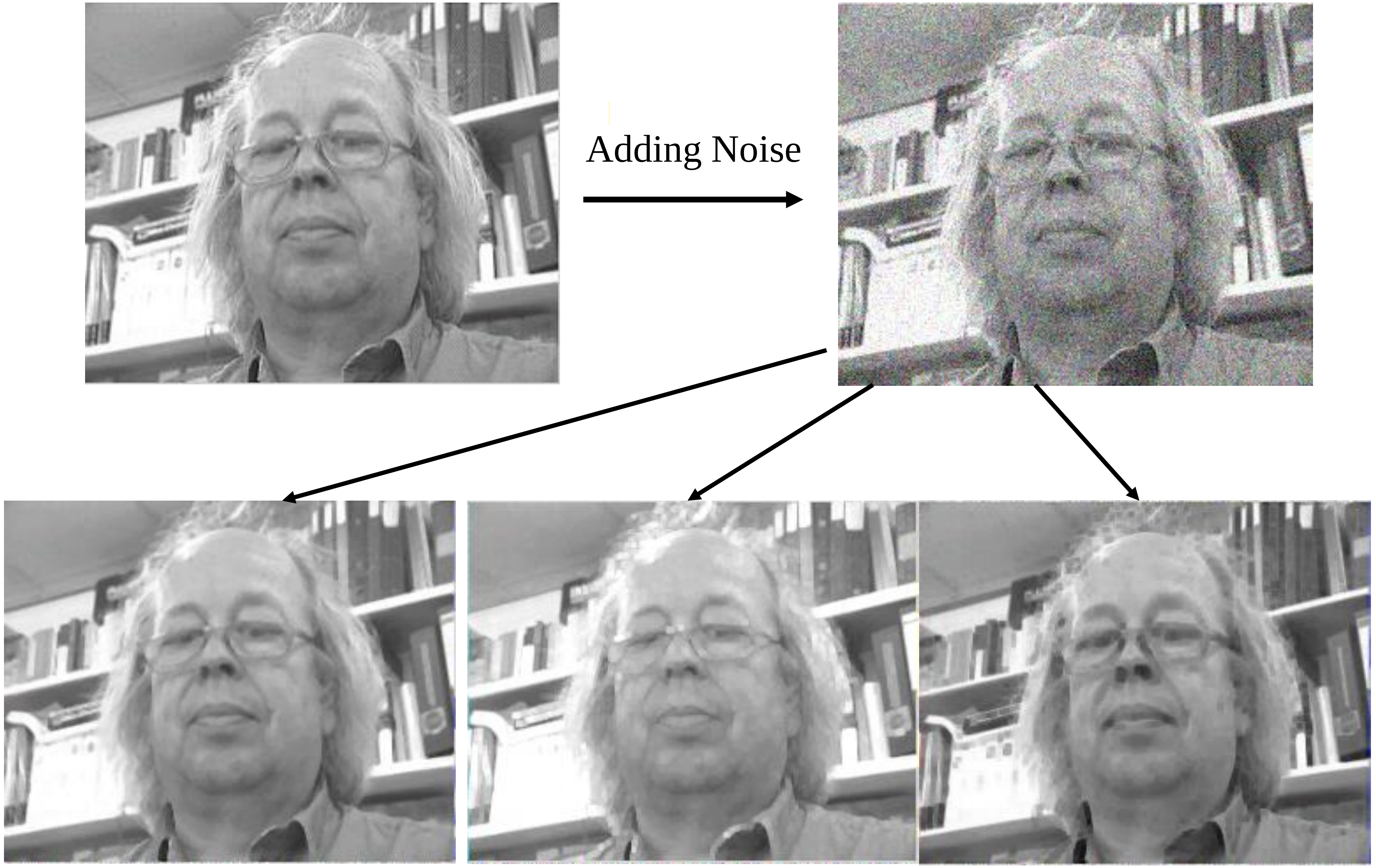
- ***These refer to non-linear filters whose response is based on ordering the pixels contained in the neighborhood. Examples include Max, Min, Median filters.***
- ***The median which replaces the value at the centre by the median pixel value in the neighbourhood, (i.e. the middle element when they are sorted.***
- ***Median filters are particularly useful in removing impulse noise, also known as salt-and-pepper.***


Noisy image


Averaging 3x3 filter


Median 3x3 filter

Example – Effect of different order statistics filters

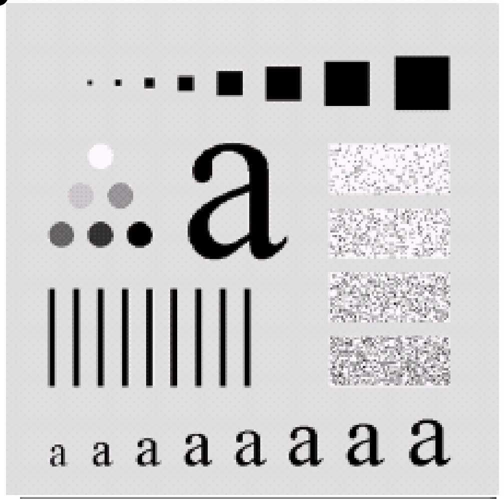


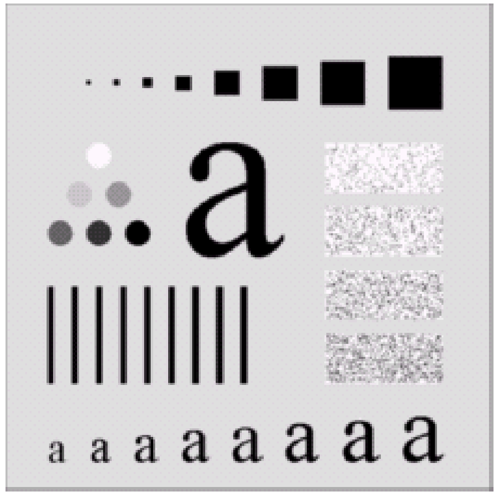
3x3 median

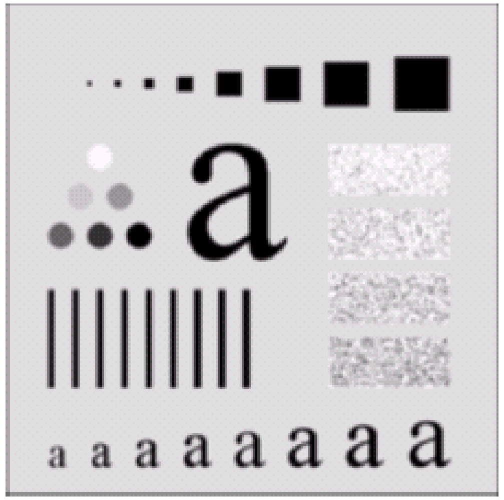
3x3 max

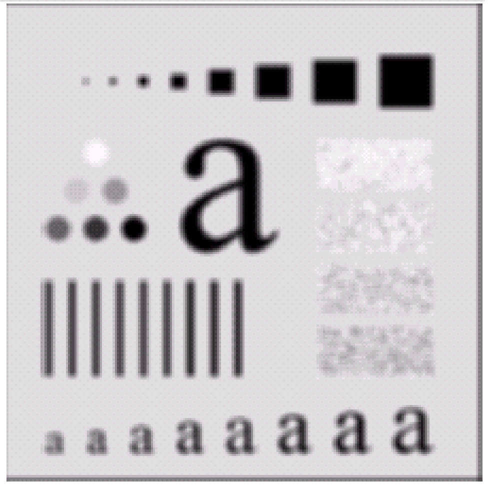
3x3 min

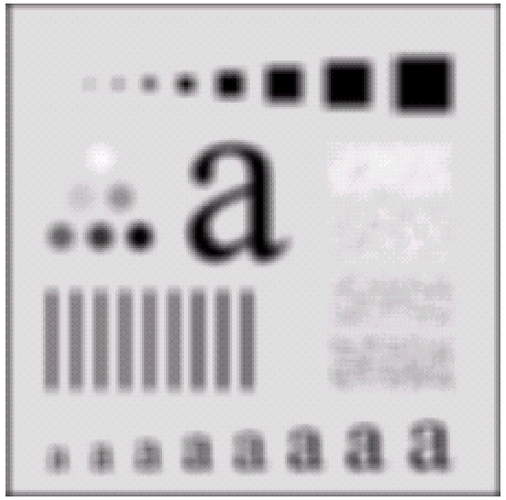
Effect of Averaging Linear Filters

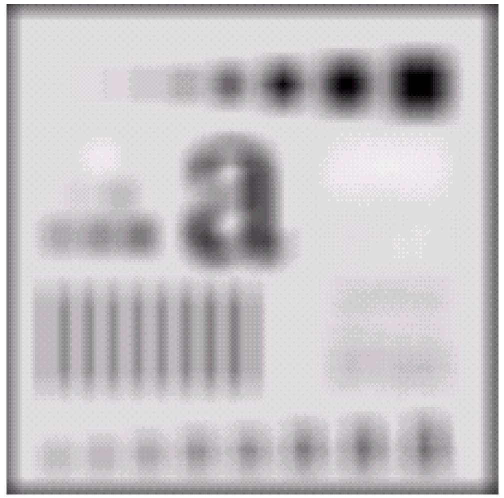

original


3x3 filter


5x5 filter


9x9 filter


15x15 filter


35x35 filter

The extent of burring increases the larger the filter is.

Sharpening Spatial Filters

- Sharpening aims to highlight fine details (e.g. edges) in an image, or enhance detail that has been blurred through errors or imperfect capturing devices.
- Image blurring can be achieved using averaging filters, and hence sharpening can be achieved by operators that invert averaging operators.
- In mathematics averaging is equivalent to the concept of integration along the gray level range:

$$s = T(r) = \int_0^r p(w)dw.$$

Integration inverts differentiation and as before, we need a digitized version of derivatives.

Derivatives of Digital functions of 2 variables

- **As we deal with images, which are represented by digital function of two variables, then we use the notion of partial derivatives rather than ordinary derivatives.**
- **The first order partial derivatives of the digital image f(x,y) in the x and in the y directions at (x,y) pixel are:**

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

$$f(x,y)$$

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

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

- **The first derivative is 0 along flat segments (i.e. constant gray values) in the specified direction.**
- **It is non-zero at the outset and end of sudden image discontinuities (edges or noise) or along segments of continuing changes (i.e. ramps).**

2nd order Derivatives & the Laplacian operator

- **The first derivative of a digital function f(x,y) is another digital image and thus we can define 2nd derivatives:**

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial\left(\partial f / \partial x\right)}{\partial x} = f(x+1, y) + f(x-1, y) - 2f(x, y), \quad \text{and}$$

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

- **Other second order partial derivatives can be defined similarly, but we will not use here.**
- **The Laplacian operator of an image f(x,y) is:**

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

Digital derivatives –Example

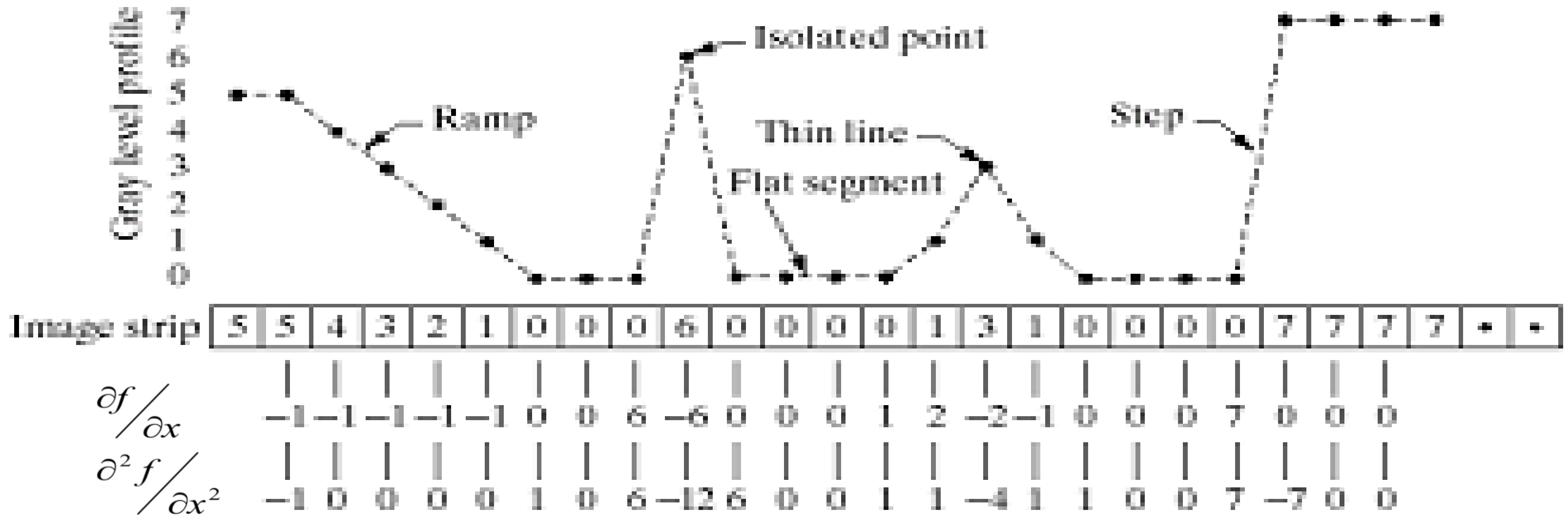


Image strip	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	•	•
$\partial f / \partial x$		−1	−1	−1	−1	−1	0	0	6	−6	0	0	0	1	2	−2	−1	0	0	0	7	0	0	0			
$\partial^2 f / \partial x^2$		−1	0	0	0	0	1	0	6	−12	6	0	0	1	1	−4	1	1	0	0	7	−7	0	0			

1. $\partial f / \partial x \neq 0$ along the entire ramp, but $\partial^2 f / \partial x^2 \neq 0$ at its ends.

   Thus $\partial f / \partial x$ detects thick edges while $\partial^2 f / \partial x^2$ detects thin edges.

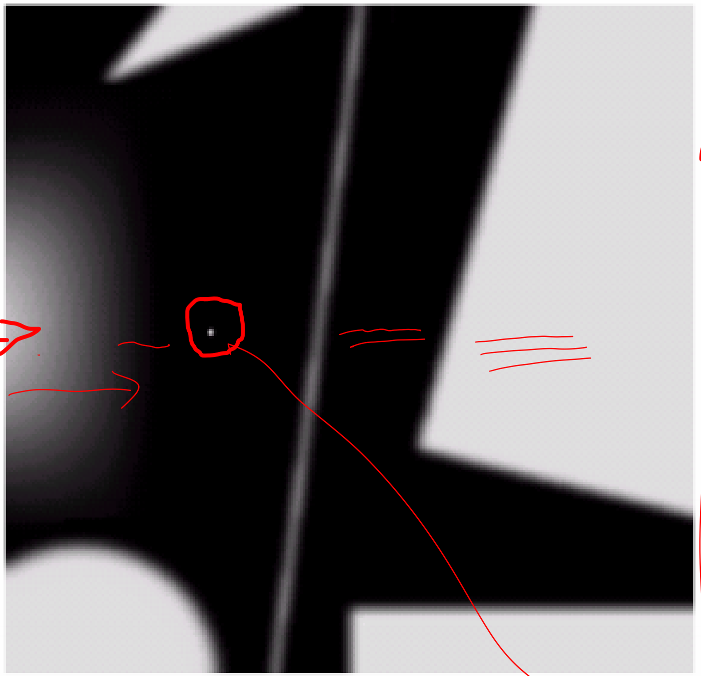
2. $\partial f / \partial x$ have stronger response to gray - level step,

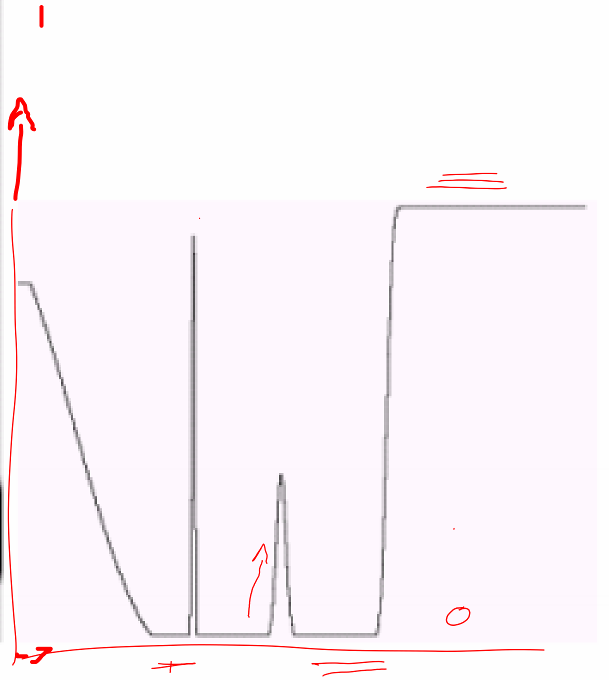
   but $\partial^2 f / \partial x^2$ produces a double response at gray - level steps.

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

a b
c

FIGURE 3.38 (a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).





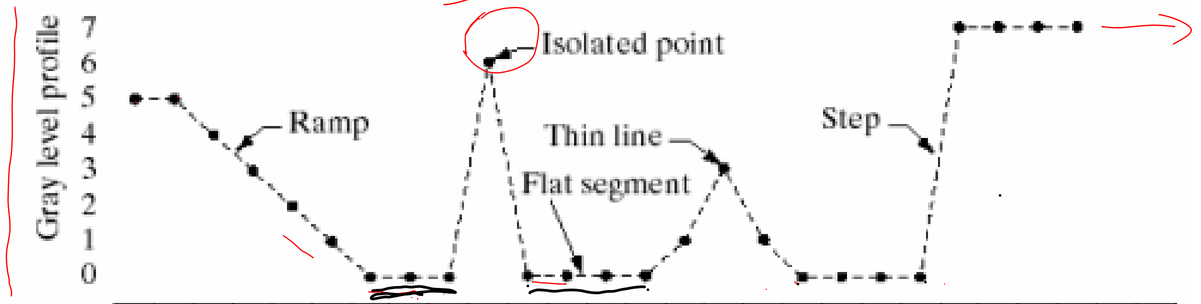


Image strip	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	•	•
First Derivative		–1	–1	–1	–1	–1	0	0	6	–6	0	0	0	1	2	–2	–1	0	0	0	7	0	0	0			
Second Derivative		–1	0	0	0	0	1	0	6	–12	6	0	0	1	1	–4	1	1	0	0	7	–7	0	0			

Observations

- 1st order derivatives produce thicker edges
- 2nd order derivatives have stronger response to finer details
- 1st order derivatives have stronger response to gray level step
- 2nd order derivatives produce double response to step changes
- Both 1st and 2nd order derivatives produce negative pixel values
    - Shift output image for display
    - Some applications use only the absolute value
    - overall 2nd order is better for most of the cases

Laplacian operator

- Usually the sharpening filters make use of the second order operators.
  - A second order operator is more sensitive to intensity variations than a first order operator.
- Besides, partial derivatives has to be considered for images
  - The derivative in a point depends on the direction along which it is computed.
- Operators that are invariant to rotation are called isotropic.
  - Rotate and differentiate (or filtering) has the same effects of differentiate and rotate.
- The Laplacian is the simpler isotropic derivative operator (wrt. the principal directions):
-

Laplacian filter

0	1	0
1	−4	1
0	1	0

Laplacian filter invariant to 90° rotations

1	1	1
1	−8	1
1	1	1

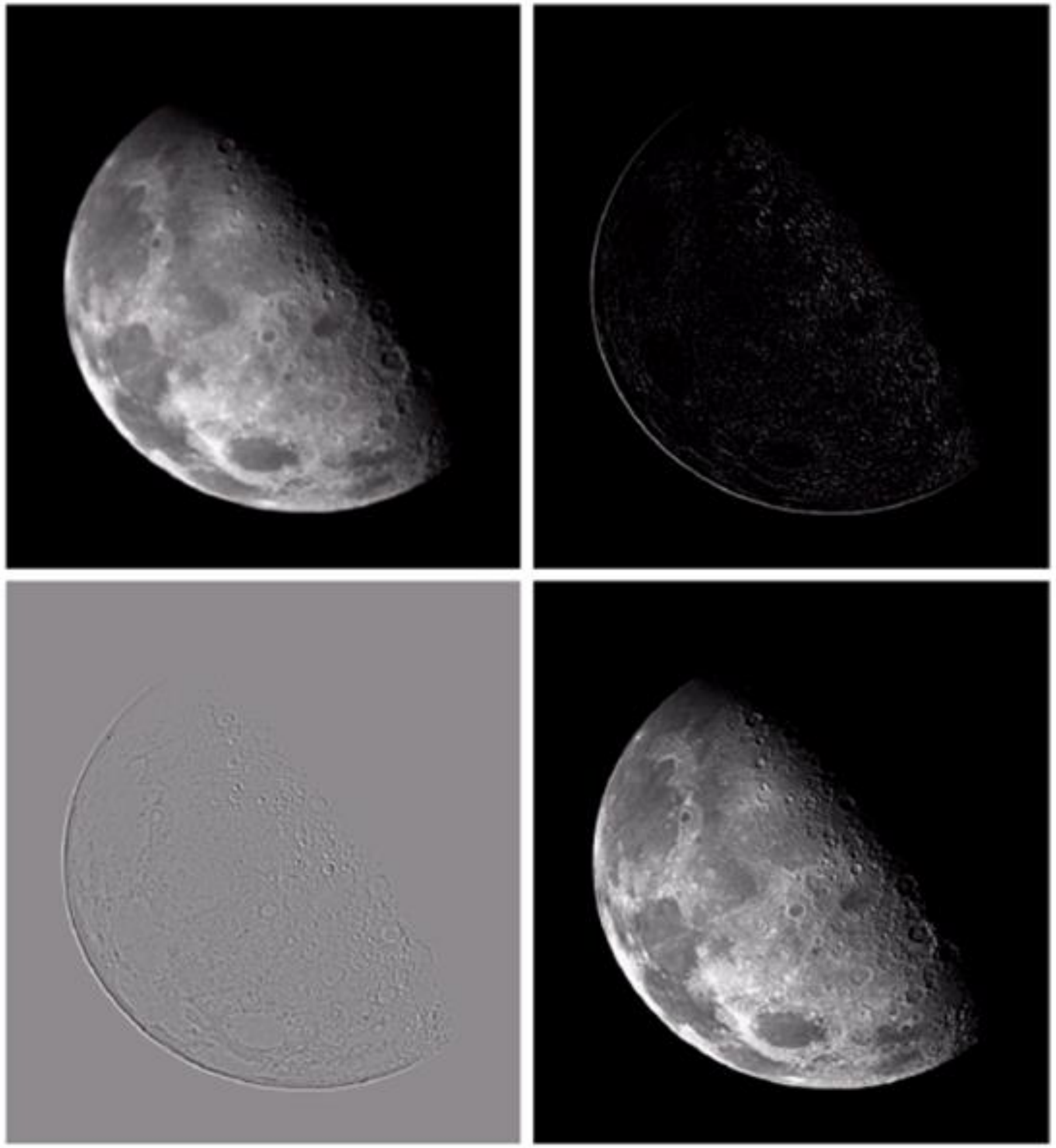
Laplacian filter invariant to 45° rotations

In order to recover background features, while still preserving the sharpening effect...

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 \underline{f(x,y)}, & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x,y) + \nabla^2 f(x,y), & \text{if the center coefficient of the Laplacian mask is positive} \end{cases}$$

a b
c d

FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)



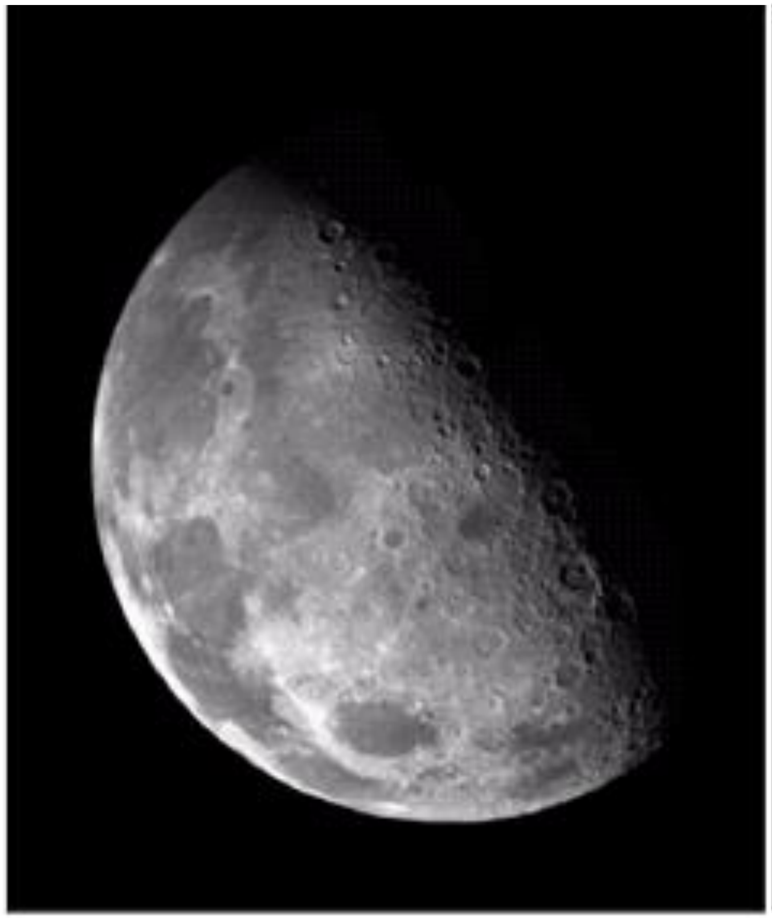
The Laplacian Filter

- **The Laplacian operator, applied to an Image can be implemented by the 3x3 filter:**

$$\nabla^2(f) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

- **Image enhancement is the result of applying the difference operator:**

Laplacian filter –Example



Original Image

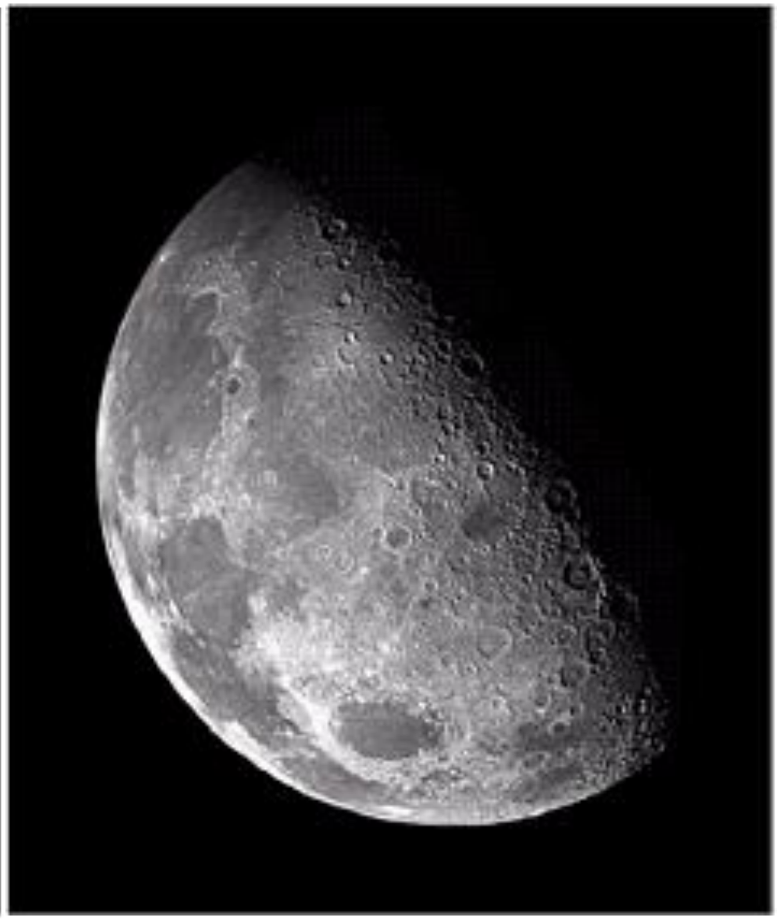
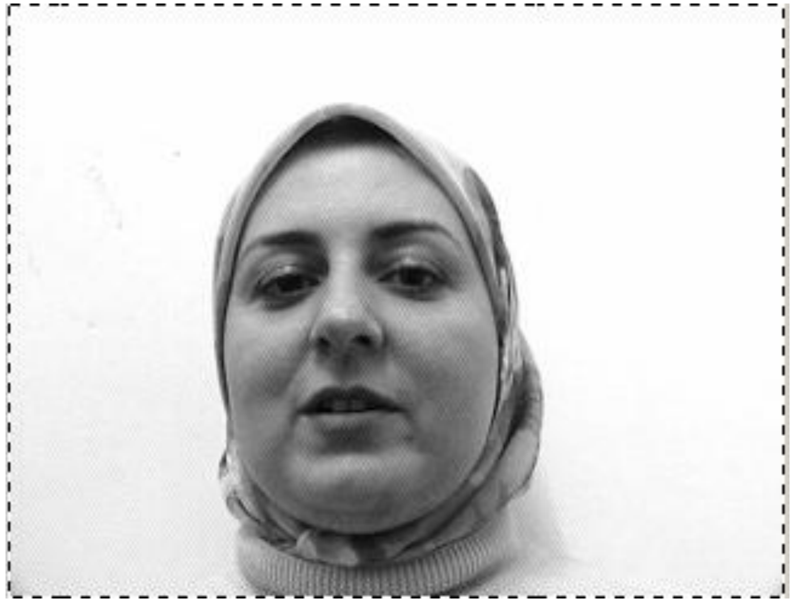


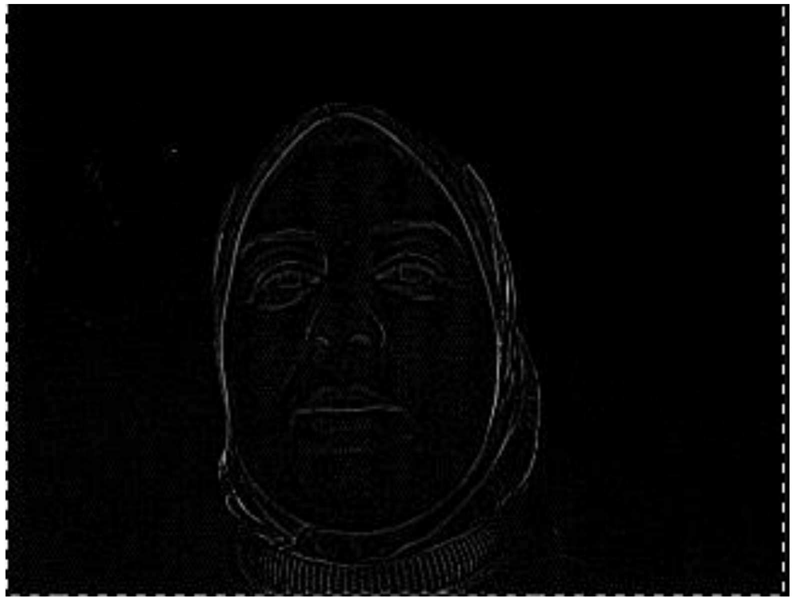
Image after Laplacian filer

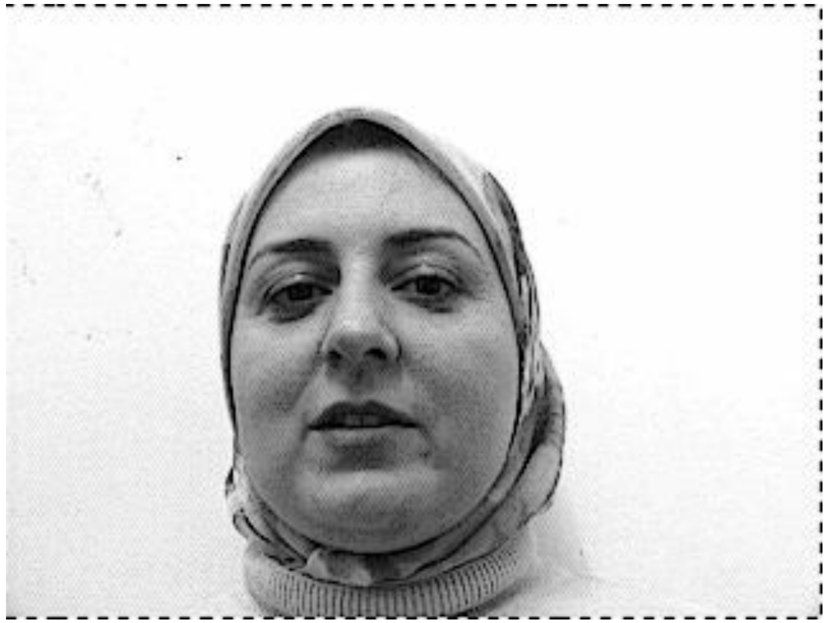
Note the enhanced details after applying the Laplacian filter.

Example



Laplacian Operator $\longrightarrow$

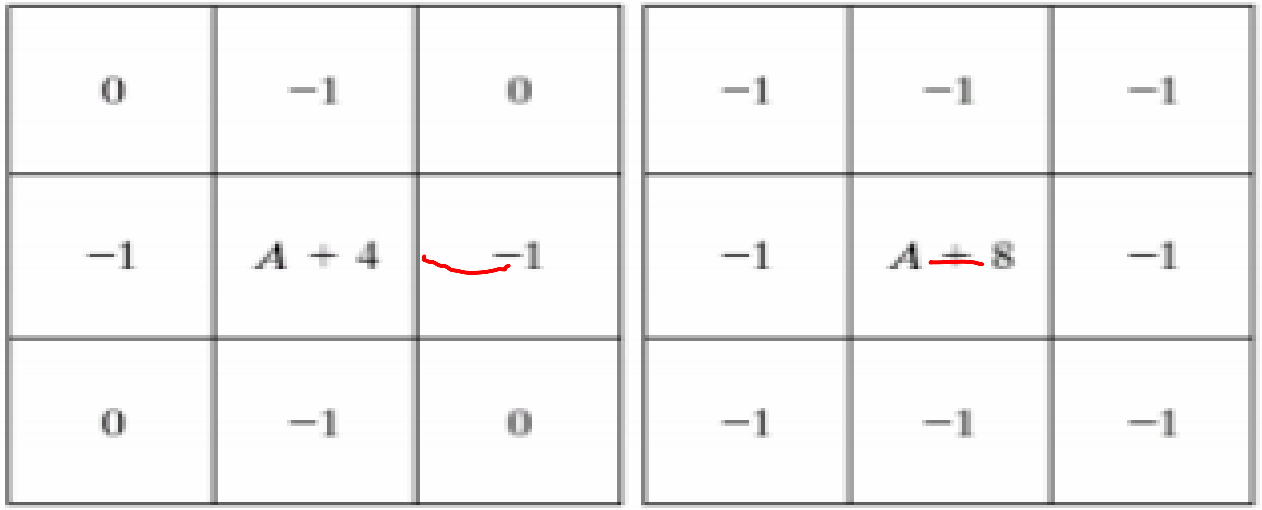




Laplacian Enhanced

Image = f - ∇^2f.

Mask of High Boost



0	−1	0
−1	$A+4$	−1
0	−1	0

−1	−1	−1
−1	$A+8$	−1
−1	−1	−1

a b

FIGURE 3.42 The high-boost filtering technique can be implemented with either one of these masks, with $A \geq 1$.

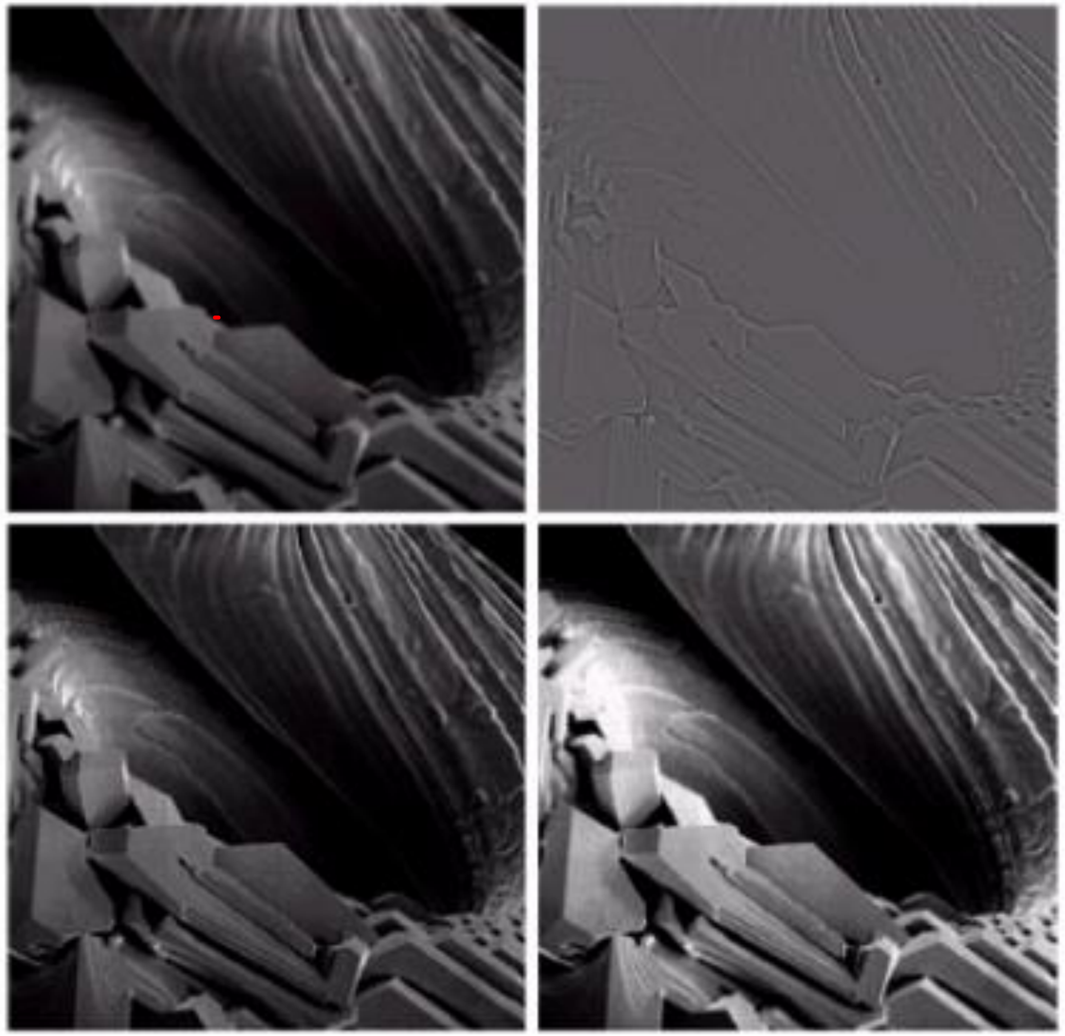


FIGURE 3.43 (a) Same as Fig. 3.41(c), but darker. (a) Laplacian of (a) computed with the mask in Fig. 3.42(b) using $A = 0$. (c) Laplacian enhanced image using the mask in Fig. 3.42(b) with $A = 1$. (d) Same as (c), but using $A = 1.7$.

(a) Original (b) $A = 0$ (c) $A = 1$ (d) $A = 1.7$

Combining various enhancement filters

- The effect of the various spatial enhancement schemes doesn’t always match the expectation, and depends on input image properties.

  e.g. histogram equalization introduces some noise.

- The application of any operator at any pixel does not depend on the position of the pixel, while the desired effect is often required in certain regions.

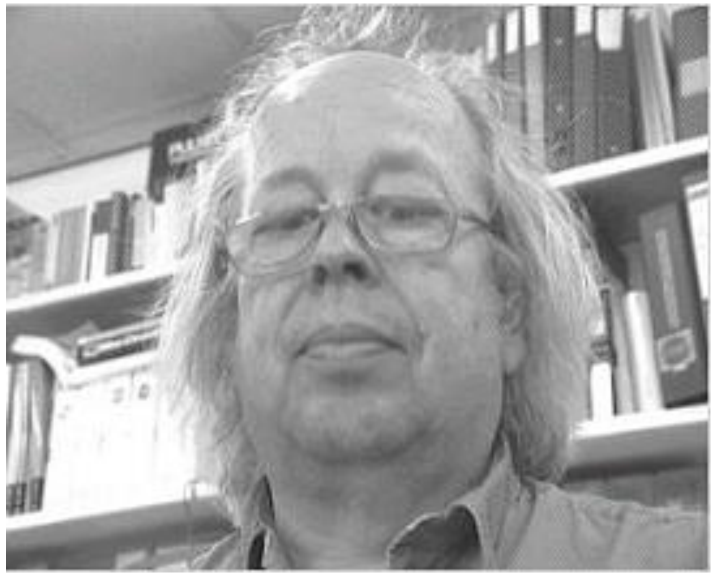
  e.g. an averaging filters blurs smooth areas as well as significant feature regions.

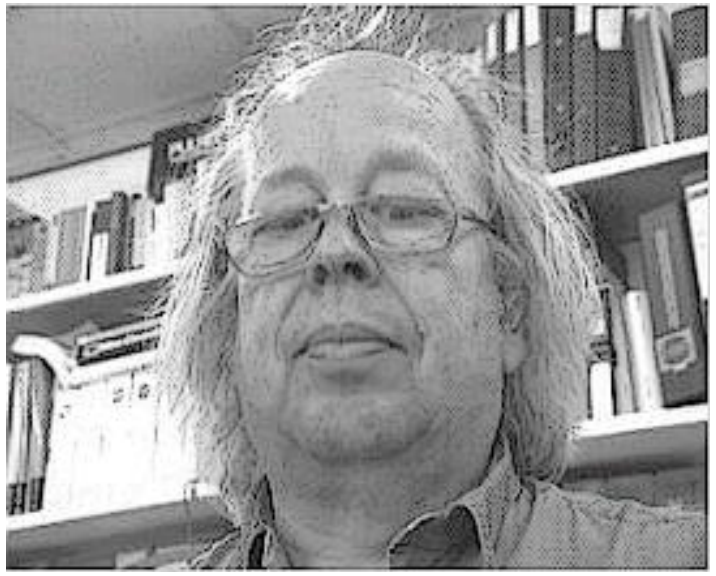
- Enhancing an image is often a trial & error process.

  In practice one may have to combine few such operations in an iterative manner .

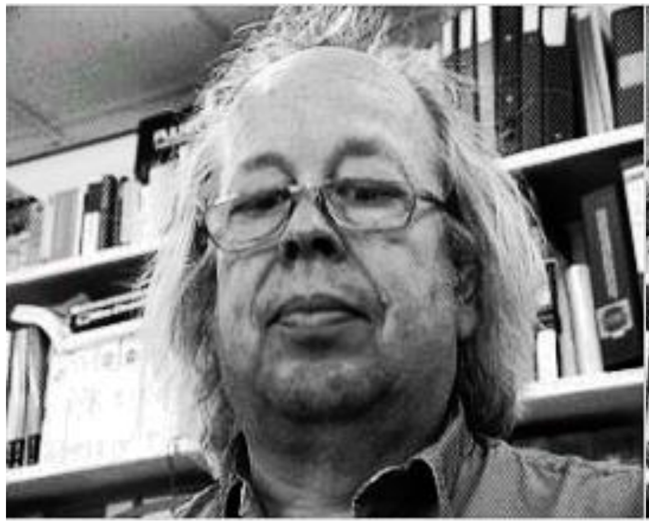
  In what follows we try few combined operations.

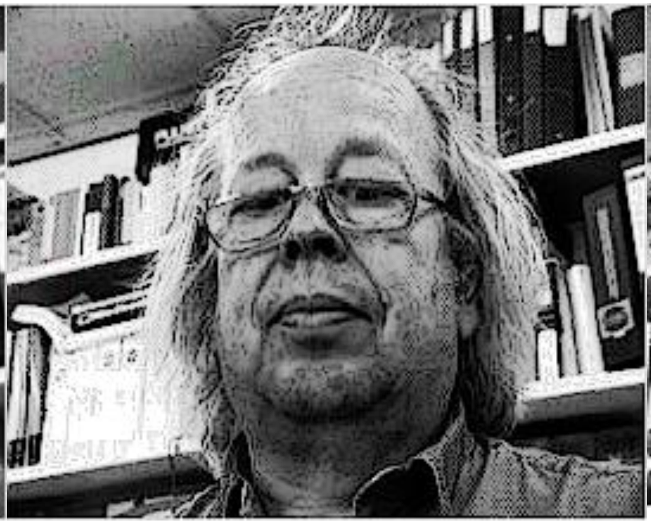
Combining Laplacian and HE

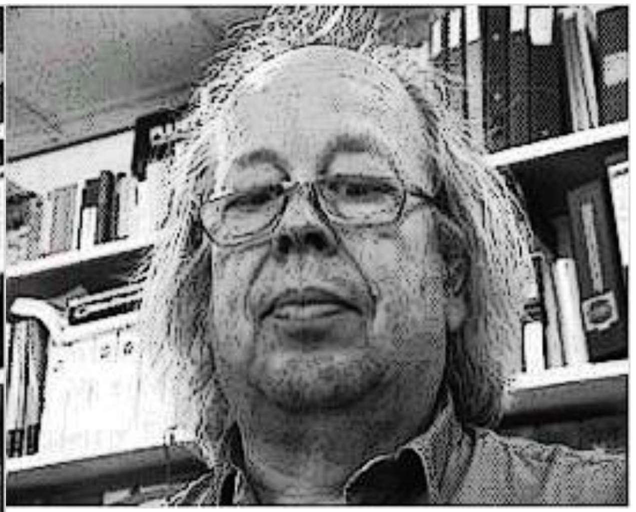




LAPChris







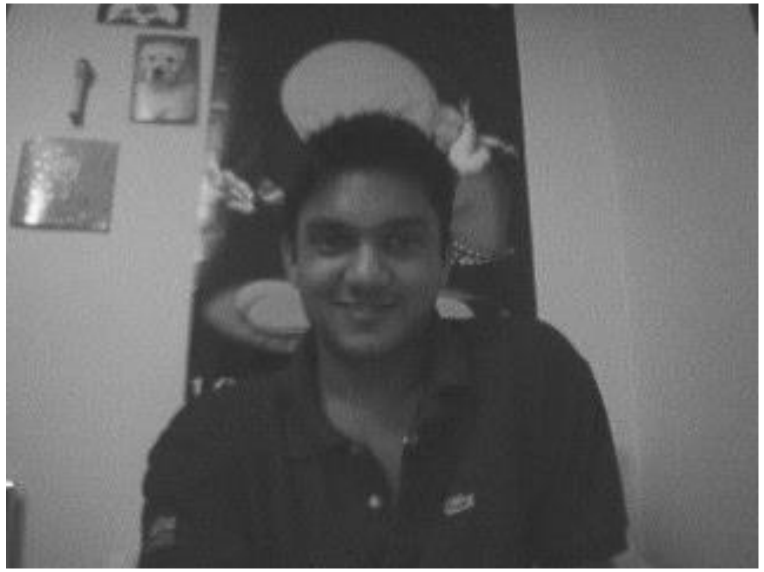
HEChris

LAPHEChris

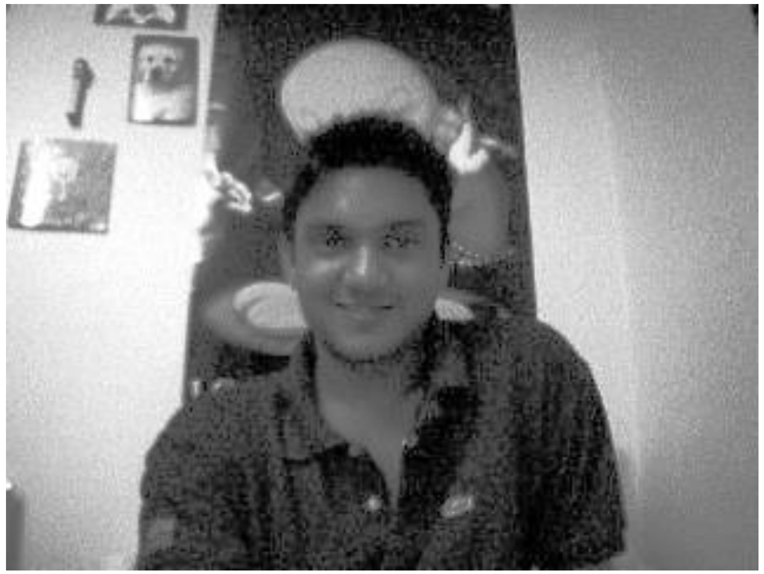
HELAPChris

Combined Laplacian & HE

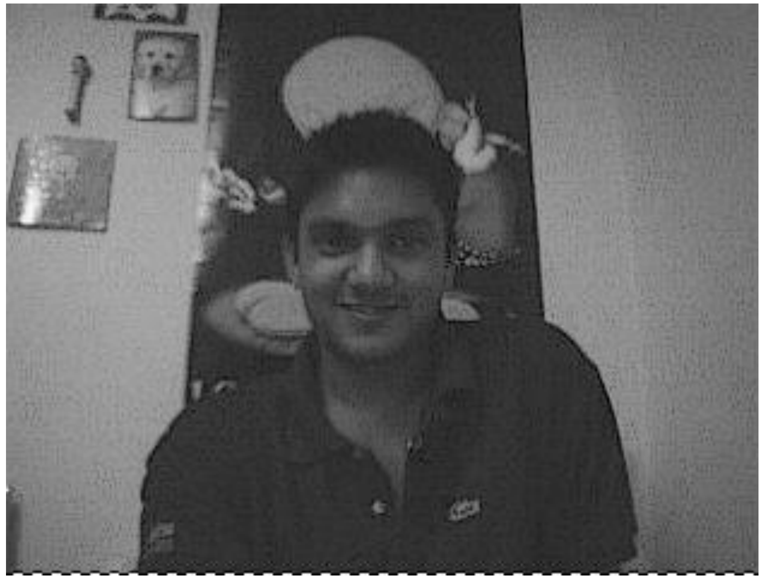
(a)



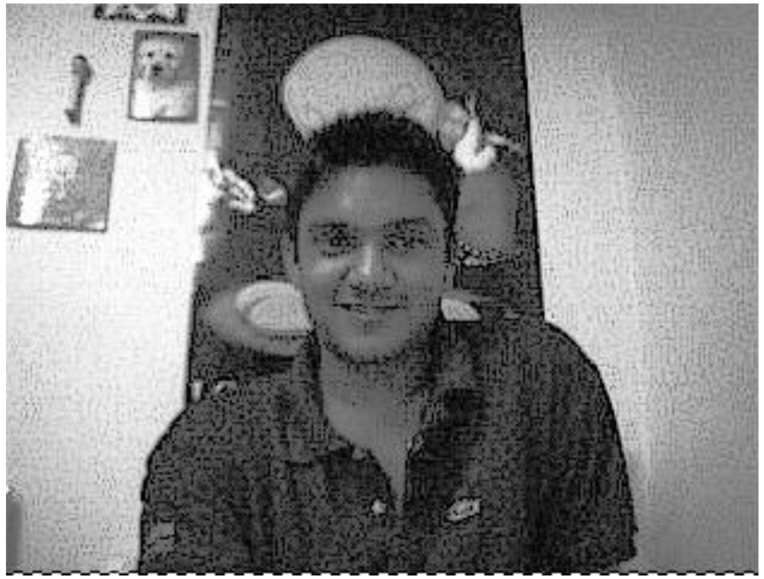
(b)



(c)



(d)



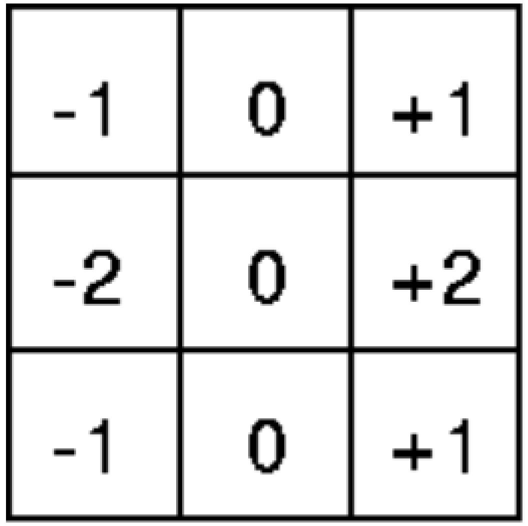
(a) Original, (b) After HE, (c) After Laplacian, (d) HE after Laplacian

Gradient in edge detection

- An image gradient is **a directional change in the intensity or color in an image**. The gradient of the image is one of the fundamental building blocks in image processing.

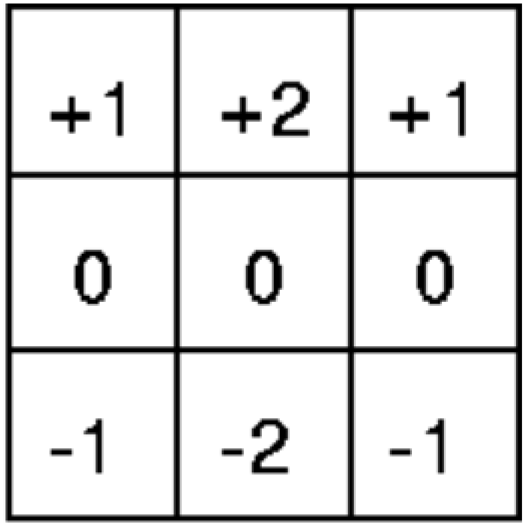
- The Sobel filter is used for **edge detection**. It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction.

Sobel convolution kernel



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

Gx



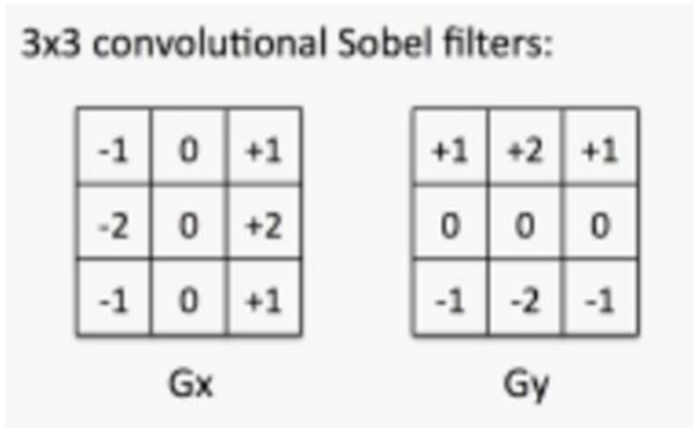
+1	+2	+1
0	0	0
-1	-2	-1

Gy

Sobel convolution kernal

$$G = \sqrt{Gx^2 + Gy^2} \; or \; G = |Gx| + |Gy|$$

Gradient magnitude calculation

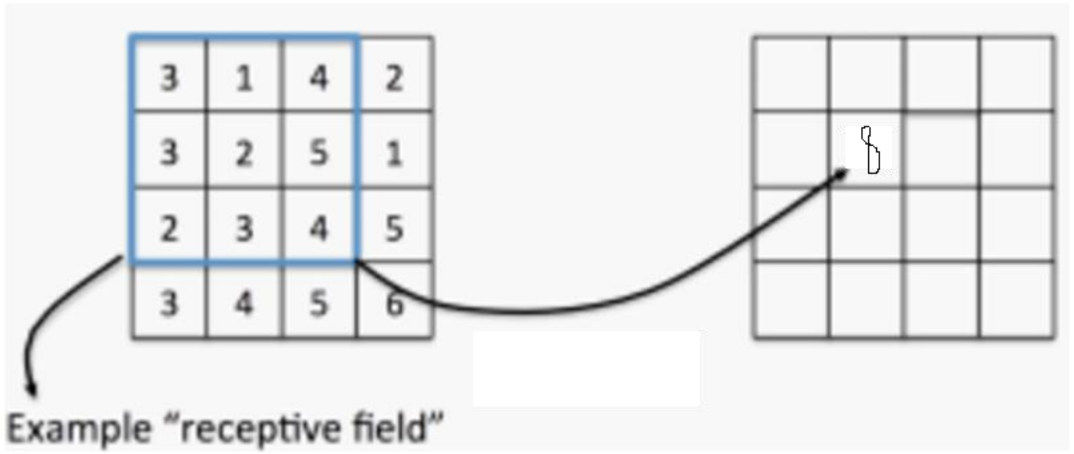


3x3 convolutional Sobel filters:

$$Gx = \begin{array}{|c|c|c|} \hline -1 & 0 & +1 \\ \hline -2 & 0 & +2 \\ \hline -1 & 0 & +1 \\ \hline \end{array} \qquad Gy = \begin{array}{|c|c|c|} \hline +1 & +2 & +1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Gx = (-1*3)+(1*4)+(-2*3)+(2*5)+(-1*2)+(1*4) = 7

Gy = (1*3)+(2*1)+(1*4)+(-1*2)+(-2*3)+(-1*4) = -3



3	1	4	2
3	2	5	1
2	3	4	5
3	4	5	6

Prewitt gradient edge detector

- It simply works like as **first order derivate and calculates the difference of pixel intensities in a edge region**. As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge.

Masks for the Prewitt gradient edge detector

-1	0	+1
-1	0	+1
-1	0	+1

Gx

+1	+1	+1
0	0	0
-1	-1	-1

Gy

$$g(x,y) = \sqrt{Gx^2 + Gy^2}$$