ISEL
INSTITUTO SUPERIOR
DE ENGENHARIA DE LISBOA

PROCESSAMENTO DE IMAGEM E BIOMETRIA

IMAGE PROCESSING AND BIOMETRICS

5. **SPATIAL FILTERING (part 1)**

Summary

- Spatial filtering - neighborhood of a pixel
- Mask, window, kernel
- Linear spatial filtering operations
- Non-linear spatial filtering
- Laplacian operators

Neighborhood of a pixel

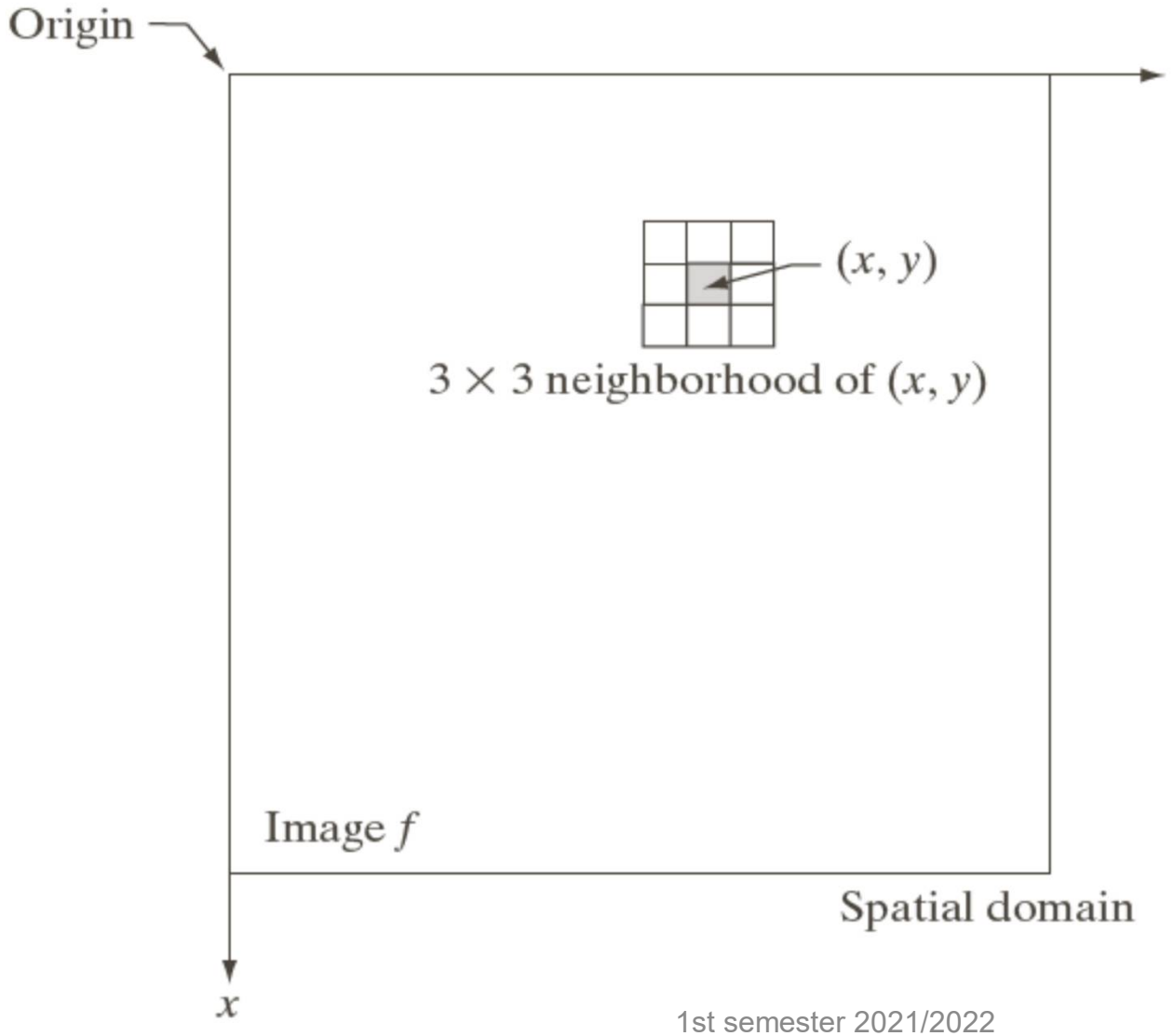


FIGURE 3.1 A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

The mechanics of spatial filtering (1)

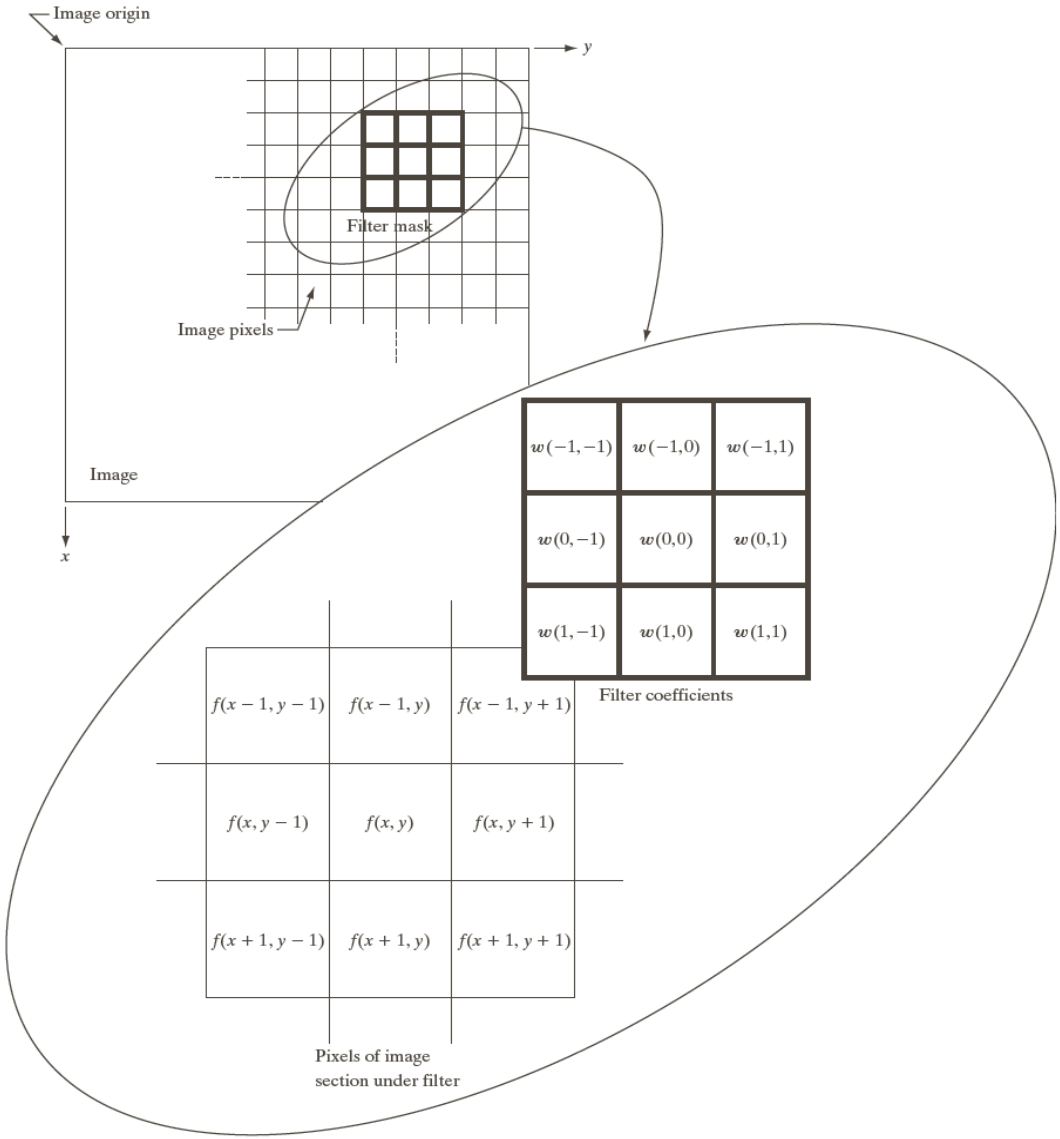
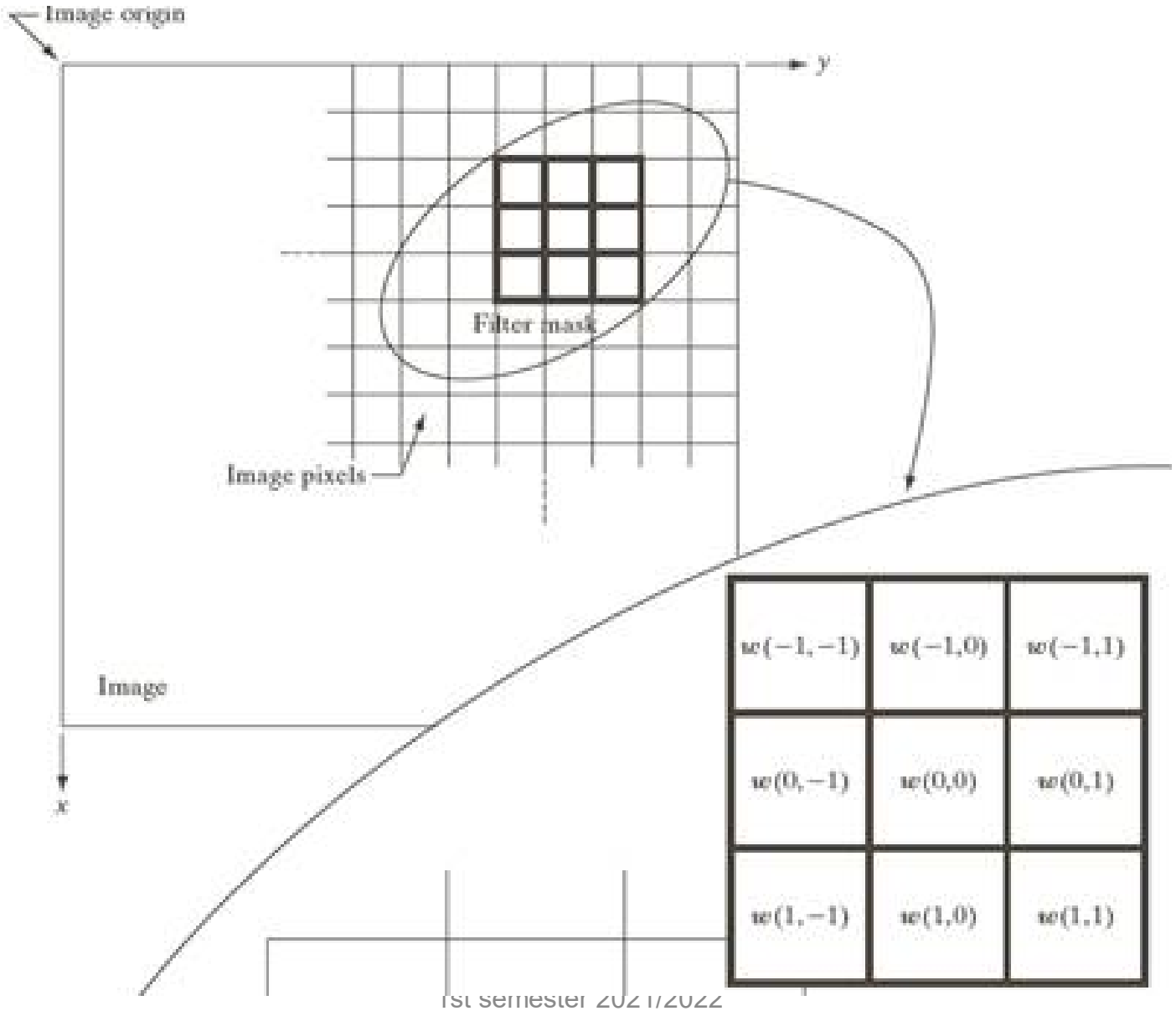
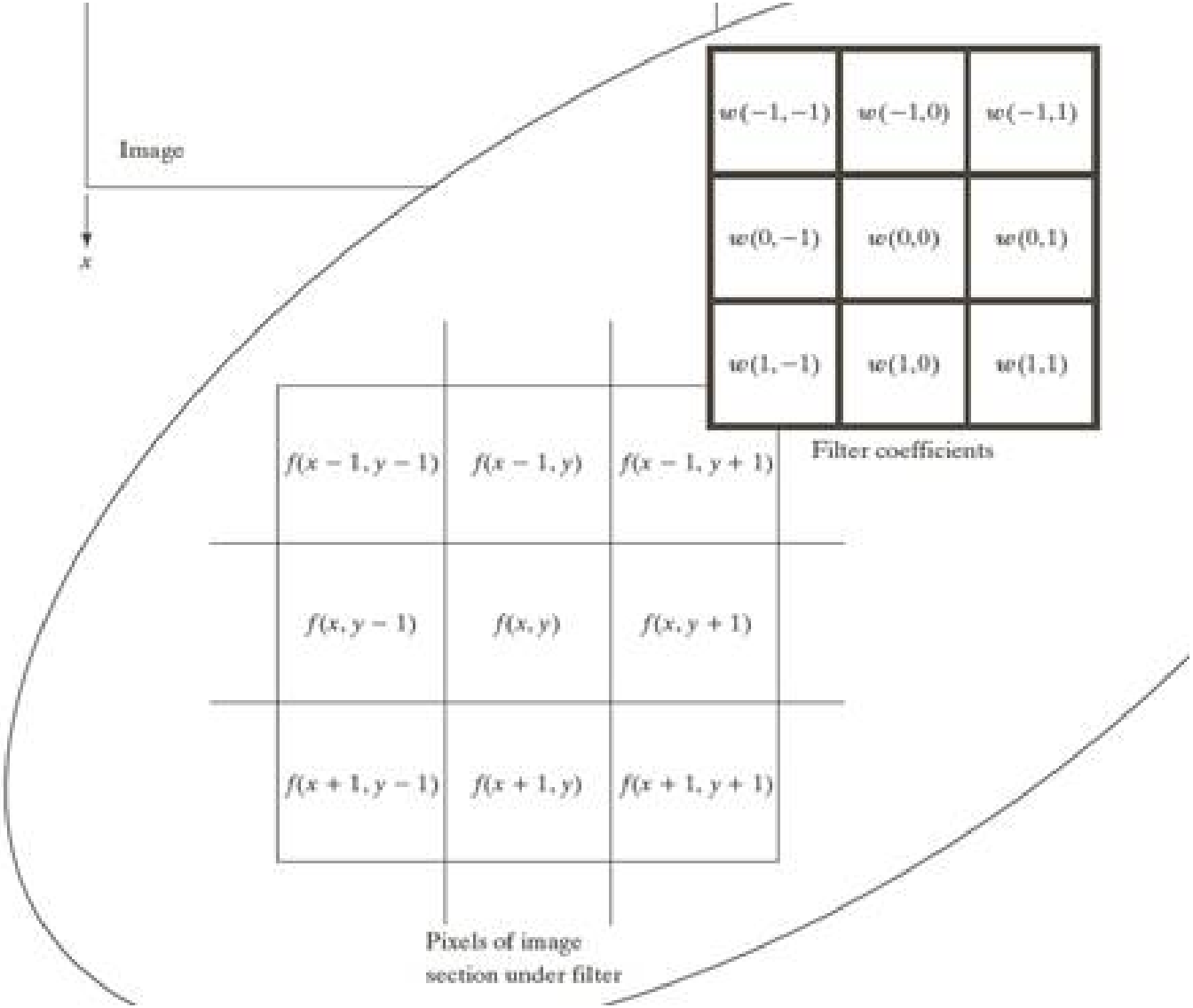


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

The mechanics of spatial filtering (2)



The mechanics of spatial filtering (3)



The mechanics of spatial filtering (4)

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 3.31 Another representation of a general 3×3 filter mask.

The mechanics of spatial filtering (5)

Spatial filtering can be:

- **Linear**, in which the window/kernel/mask is defined by a set of numbers. The output pixel value is the linear combination of the image pixel values contained in the mask

- **Non-linear**, in which the window/kernel/mask is defined by horizontal and vertical dimensions and a function

The mechanics of spatial filtering (6)

On **linear spatial filtering** the output pixel value, z, is given by (for a 3x3 window):

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

- The window coefficients define the type of filter/operation:
  - Smoothing (suavização)
  - Sharpening (realce)
  - Edge detection (deteção de contornos)

The mechanics of spatial filtering (7)

On **non-linear spatial filtering** the output pixel value, z, is given by (for a L_M x L_N window):

z = function (all_the_image_pixels_contained_in_the_window)

- The **function** and the dimensions of the window define the action

- Typical actions/functions are:
  - Median
  - Minimum
  - Maximum

Correlation and Convolution (1)

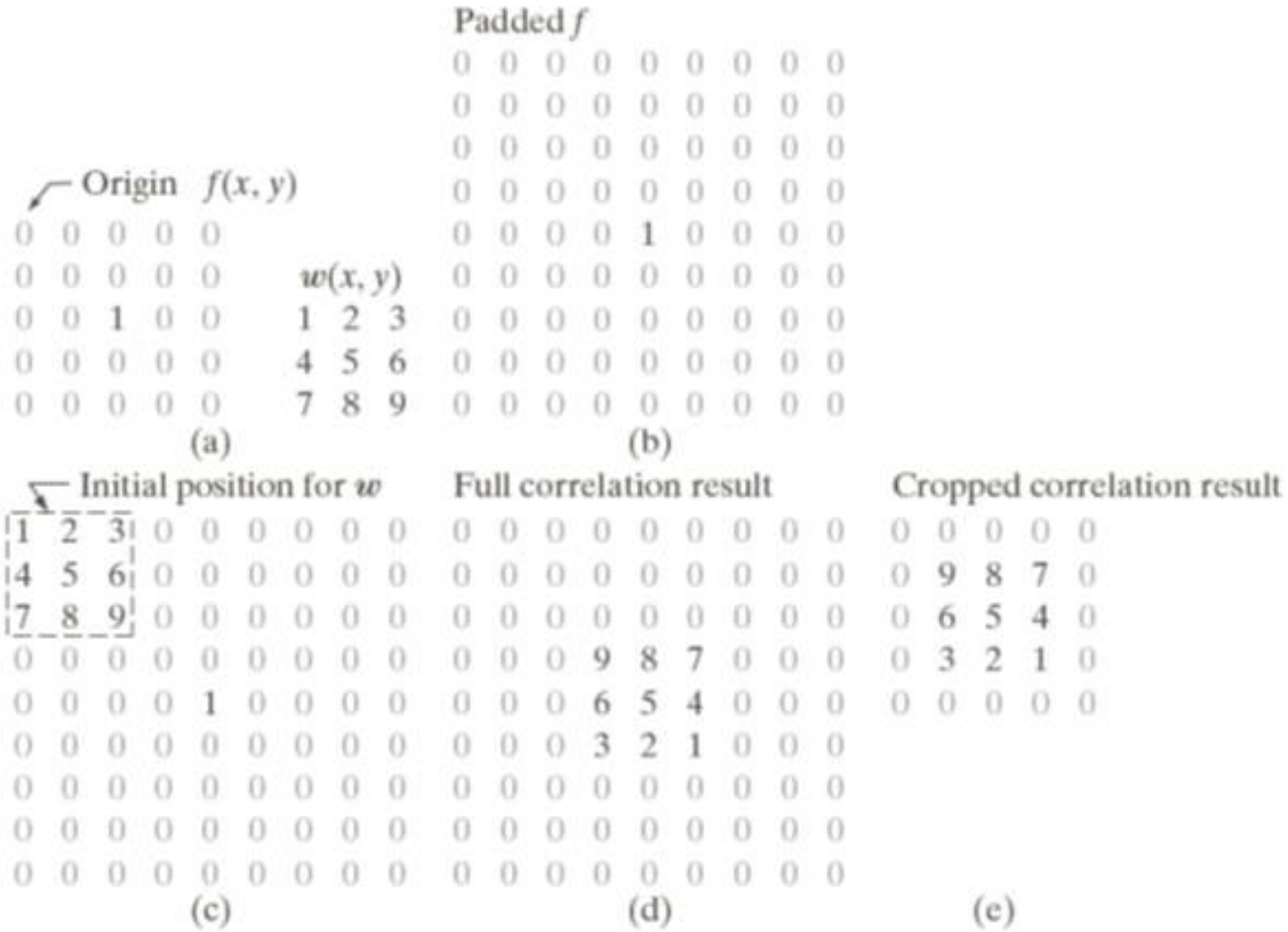


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

Correlation and Convolution (2)

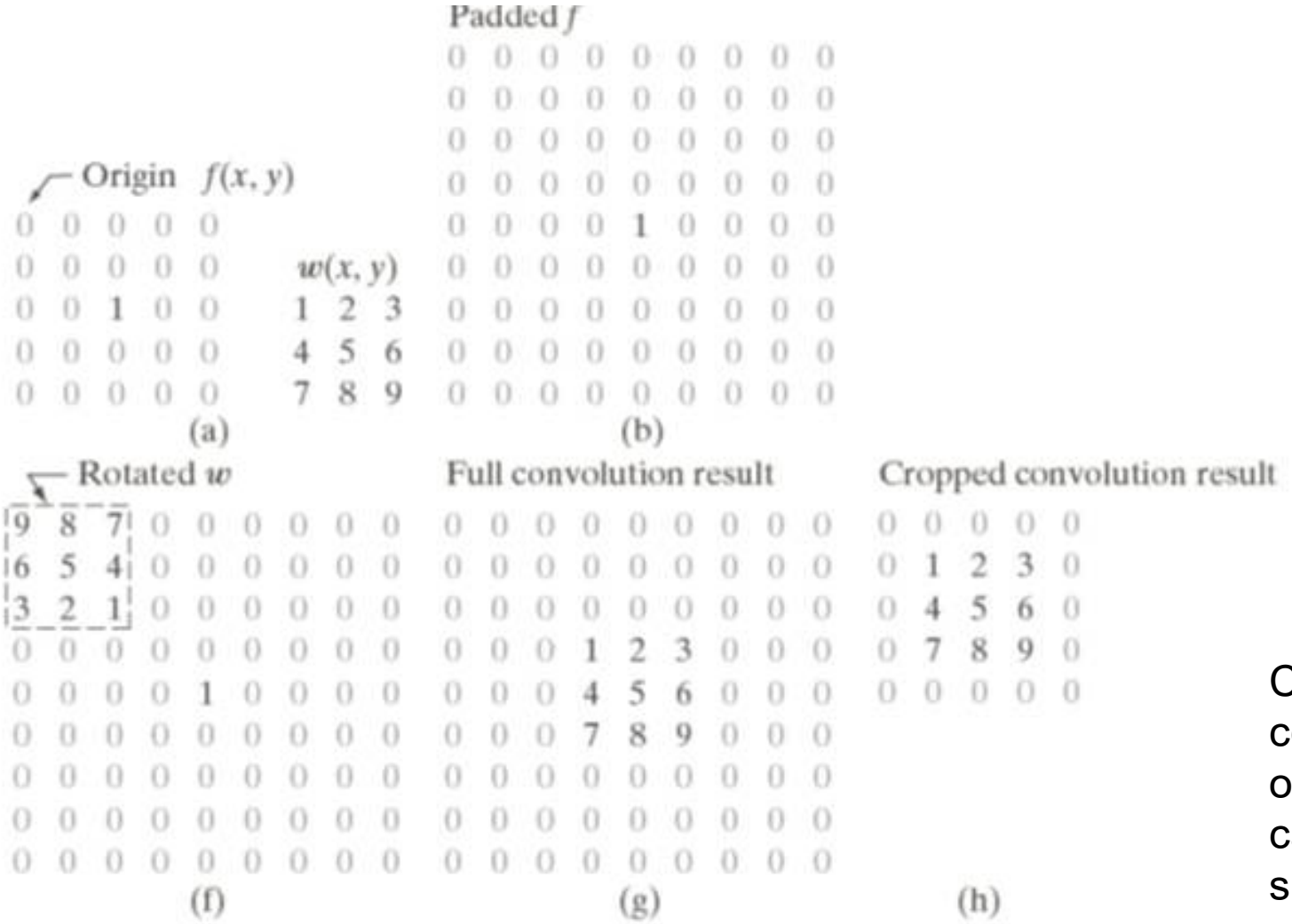


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

Correlation and convolution operations to carry out linear spatial filtering

Mask, kernel, or window (1)

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

1	1	1
1	1	1
1	1	1

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

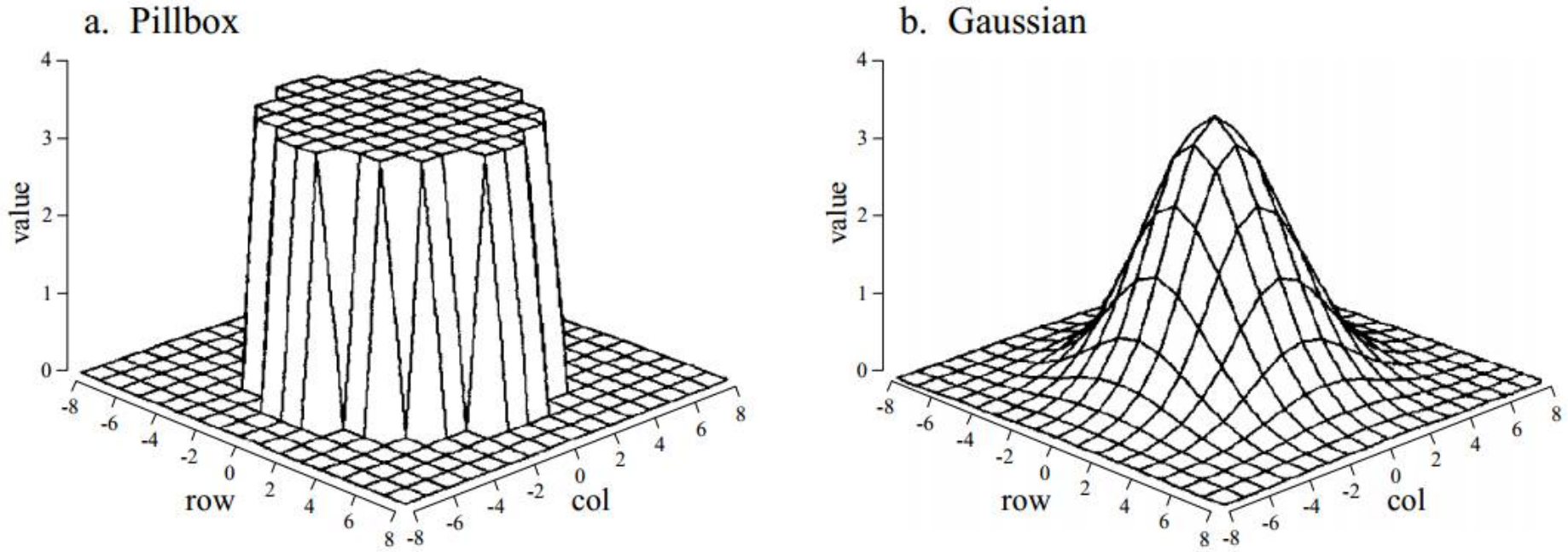
1	2	1
2	4	2
1	2	1

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.

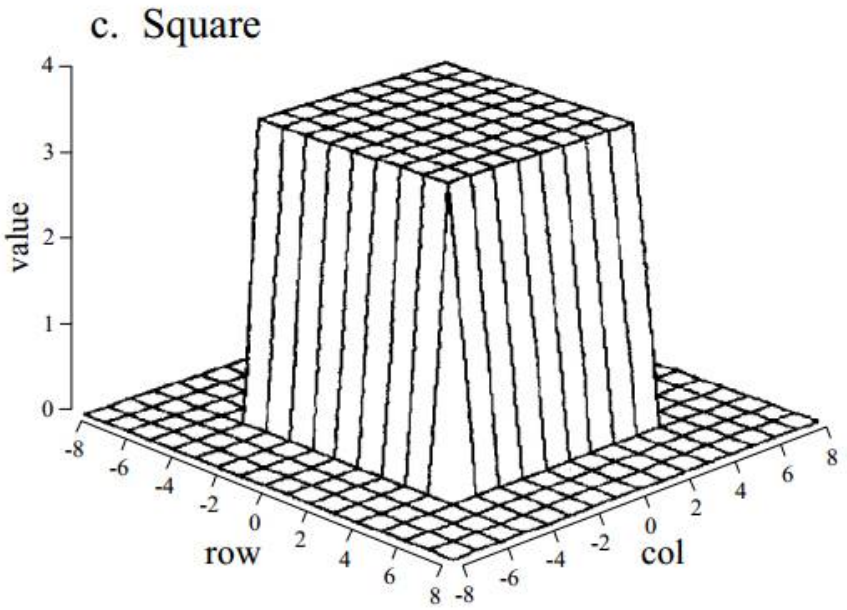
- Commonly used 3x3 masks for smoothing
- The right-hand-side mask can be applied without any products (it uses the shift operation)

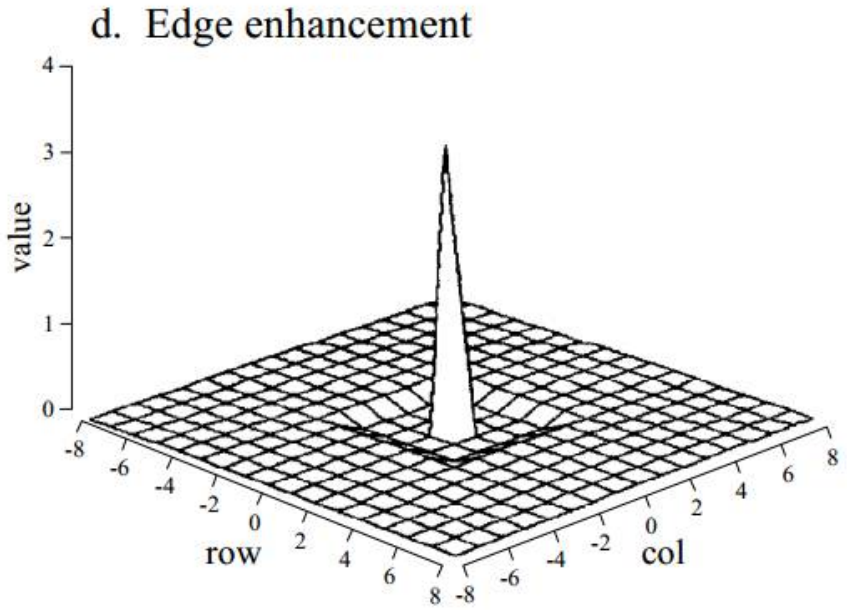
Mask, kernel, or window (2)



- Commonly used 3x3 masks for smoothing, seen as a surface
- The ‘pillbox’ mask is also known as ‘disk’ or ‘circular’ mask

Mask, kernel, or window (3)

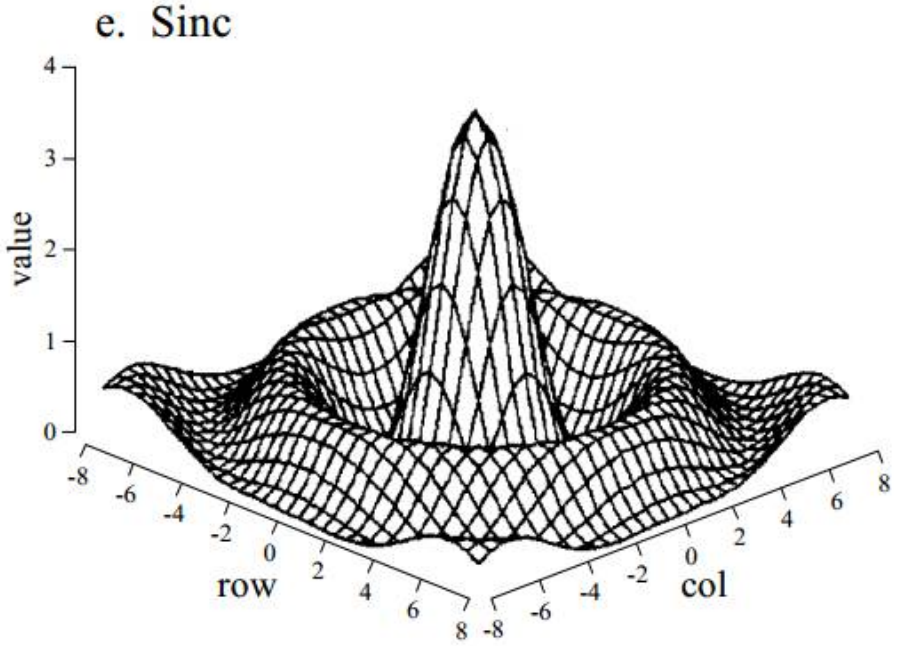




- Commonly used 3x3 masks for smoothing and edge enhancement, respectively

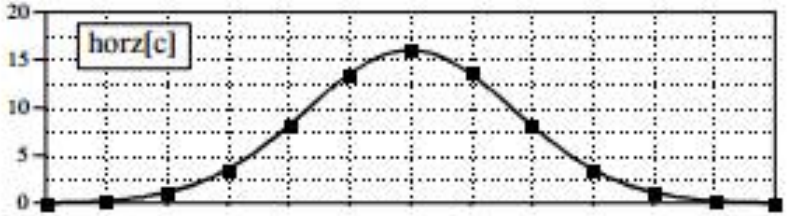
Mask, kernel, or window (4)

FIGURE 24-3
Common point spread functions. The pillbox, Gaussian, and square, shown in (a), (b), & (c), are common smoothing (low-pass) filters. Edge enhancement (high-pass) filters are formed by subtracting a low-pass kernel from an impulse, as shown in (d). The sinc function, (e), is used very little in image processing because images have their information encoded in the spatial domain, not the frequency domain.

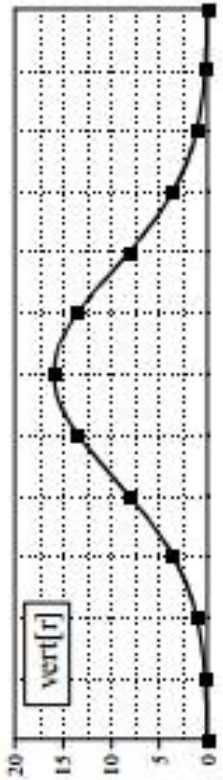


Mask, kernel, or window (5) - Separability

FIGURE 24-7
Separation of the Gaussian. The Gaussian is the only PSF that is circularly symmetric *and* separable. This makes it a common filter kernel in image processing.



0.04	0.25	1.11	3.56	8.20	13.5	16.0	13.5	8.20	3.56	1.11	0.25	0.04



0.04	0	0	0	0	0	1	1	1	0	0	0	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
16.0	1	4	18	57	131	216	255	216	131	57	18	4	1
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
0.04	0	0	0	0	0	1	1	1	0	0	0	0	0

Linear spatial filtering - Some examples (1)

- Smoothing with the average 3x3 window

Linear spatial filtering - Some examples (2)

- Smoothing with the average 5x5 window

Linear spatial filtering - Some examples (3)

- Smoothing with the average 7x7 window

Blurred/Smoothed image

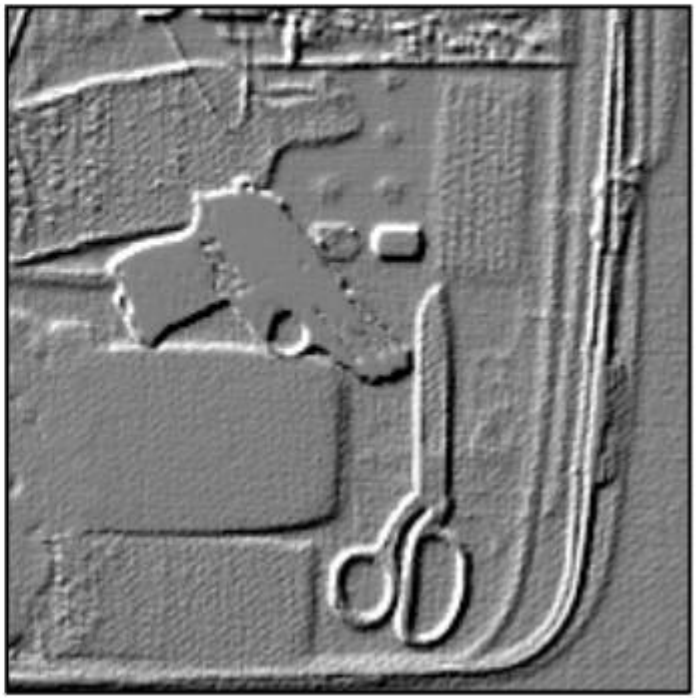
Linear spatial filtering - Some examples (4)

a. Delta function

0	0	0
0	1	0
0	0	0

b. Shift and subtract

0	0	0
0	1	0
0	0	-1



Linear spatial filtering - Some examples (5)

c. Edge detection

-1/8	-1/8	-1/8
-1/8	1	-1/8
-1/8	-1/8	-1/8

d. Edge enhancement

-k/8	-k/8	-k/8
-k/8	k+1	-k/8
-k/8	-k/8	-k/8

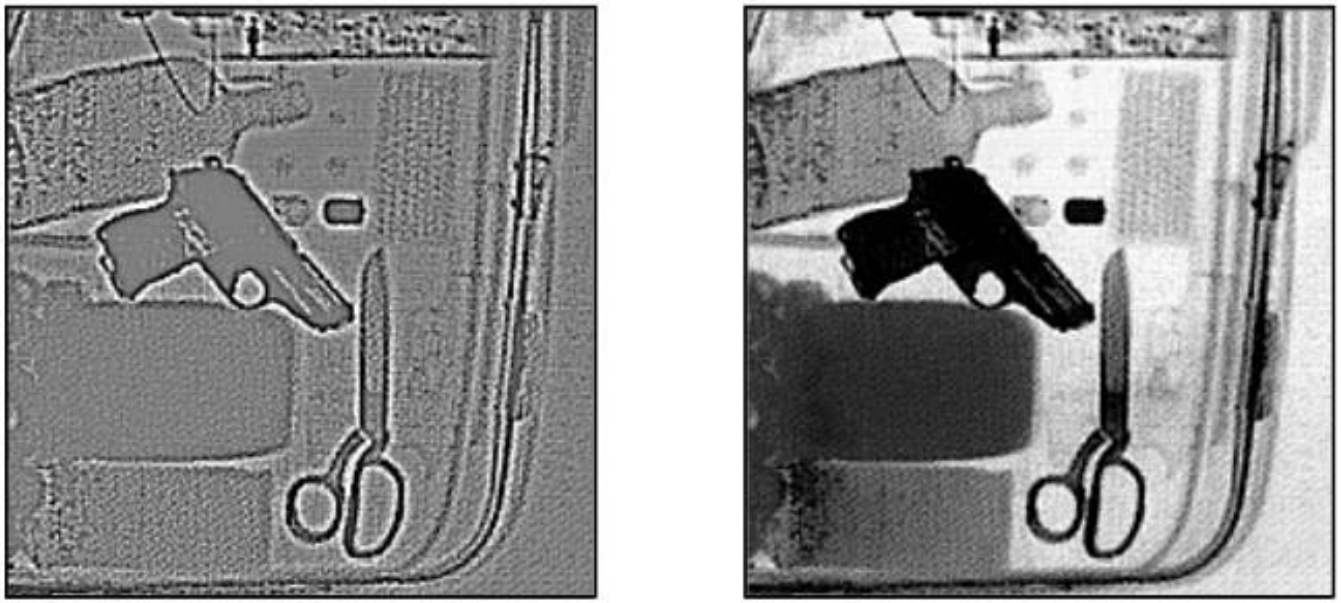


FIGURE 24-4
3×3 edge modification. The original image, (a), was acquired on an airport x-ray baggage scanner. The shift and subtract operation, shown in (b), results in a pseudo three-dimensional effect. The edge detection operator in (c) removes all contrast, leaving only the edge information. The edge enhancement filter, (d), adds various ratios of images (a) and (c), determined by the parameter, k. A value of $k = 2$ was used to create this image.

Linear spatial filtering - Some examples (6)

Test Image

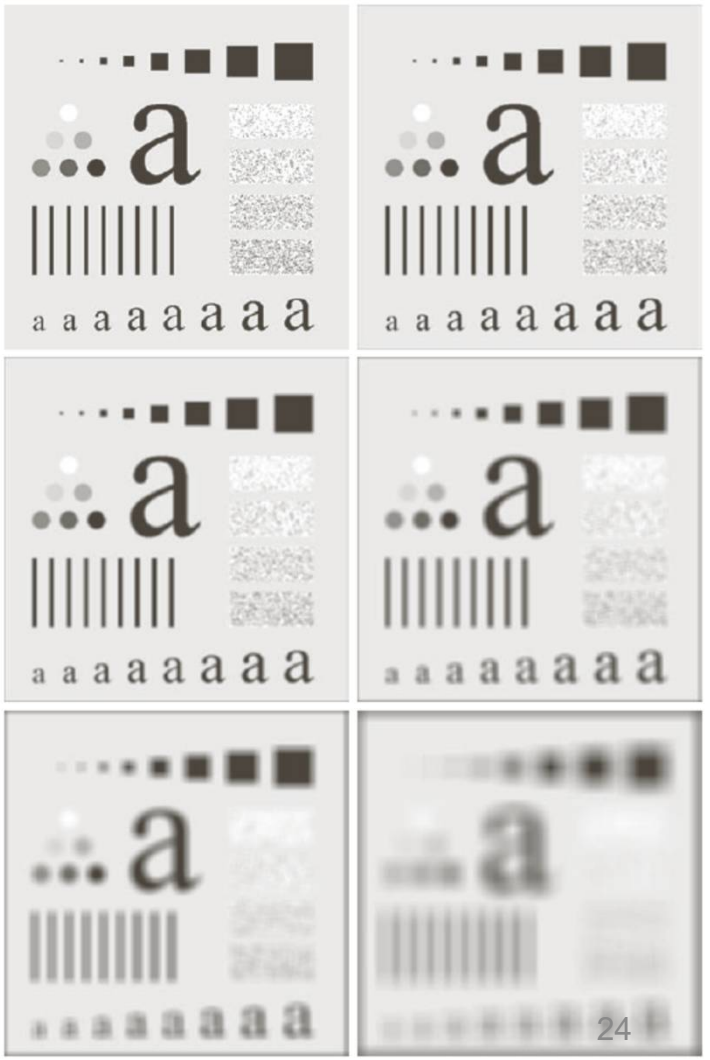
FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart.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

Linear spatial filtering - Some examples (7)

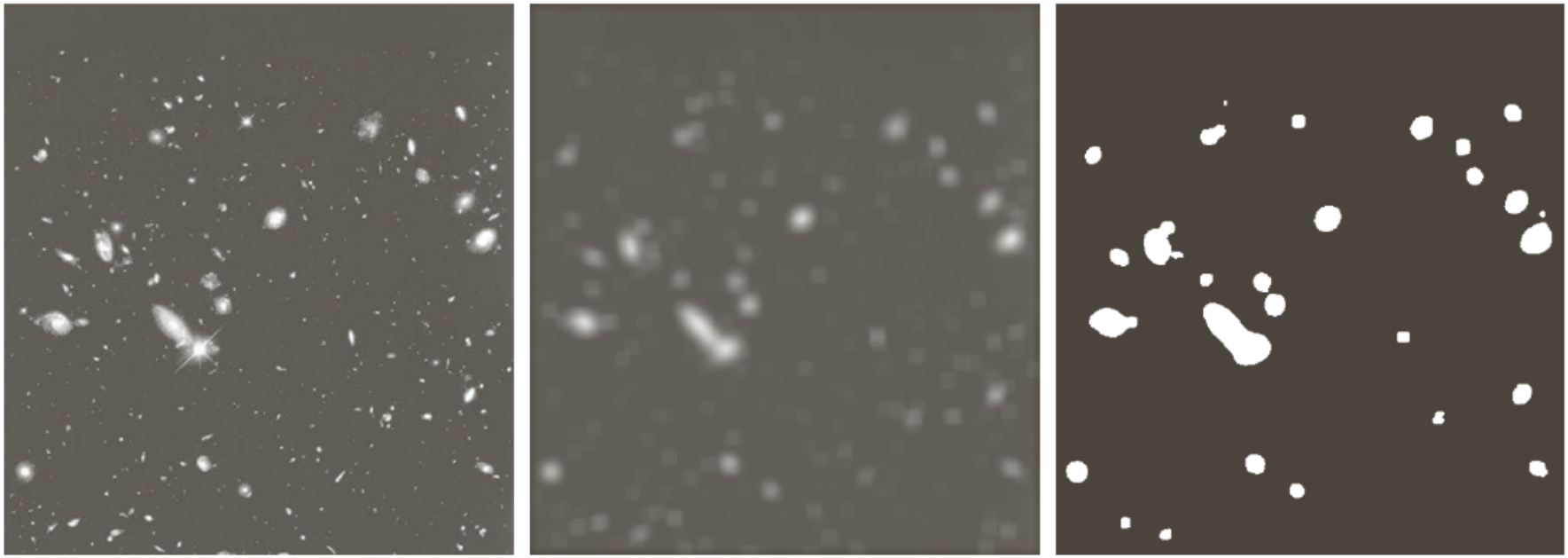
FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart.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



Linear spatial filtering - Some examples (8)

- Spatial filtering (smoothing) followed by intensity transformation (thresholding function)
- On the output image, only the larger objects are kept

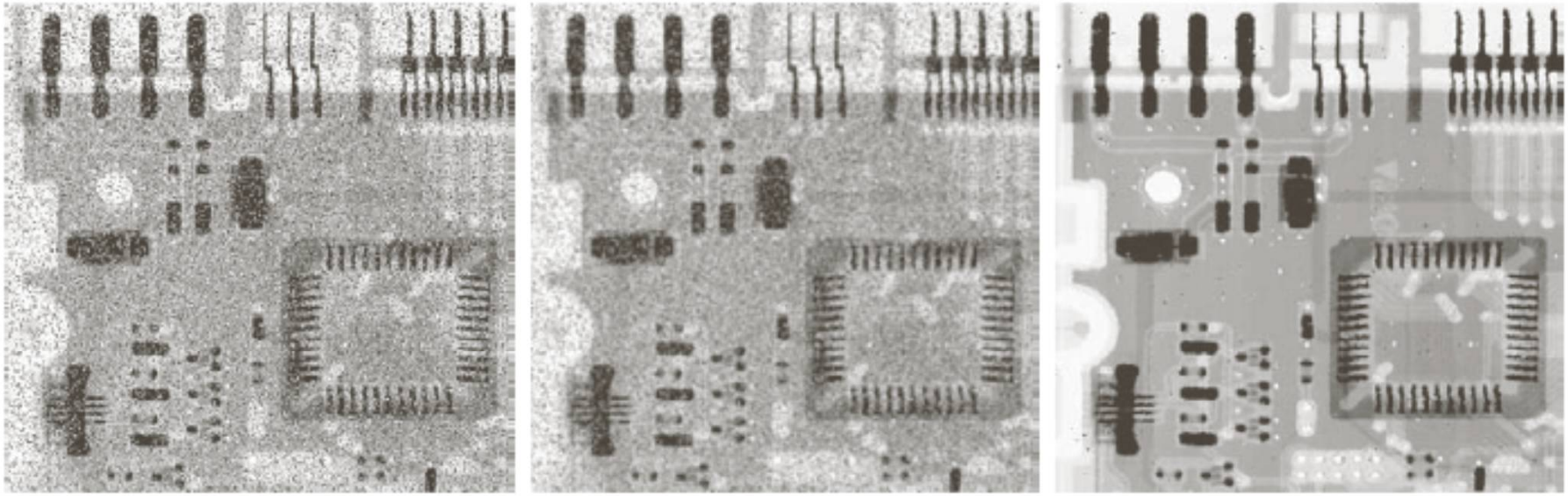


a b c

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.)

Linear and non-linear spatial filtering

- Linear spatial filtering (average, smoothing)
- Non-linear spatial filtering (median) efficiently removes the salt&pepper noise



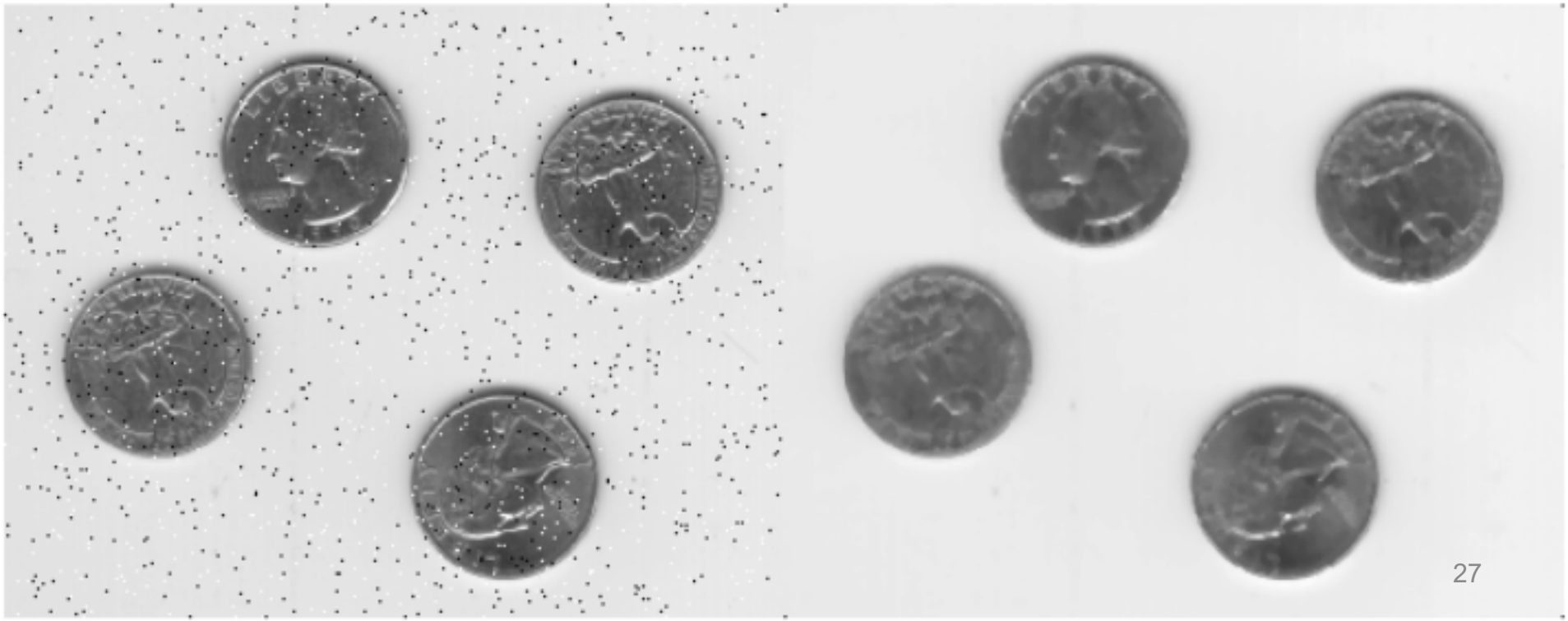
a b c

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.)

Non-linear spatial filtering (1)

- Median filtering to remove *salt&pepper* noise (3x3 mask)

```
I  = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
K = medfilt2(J);
imshowpair(J,K,'montage')
```



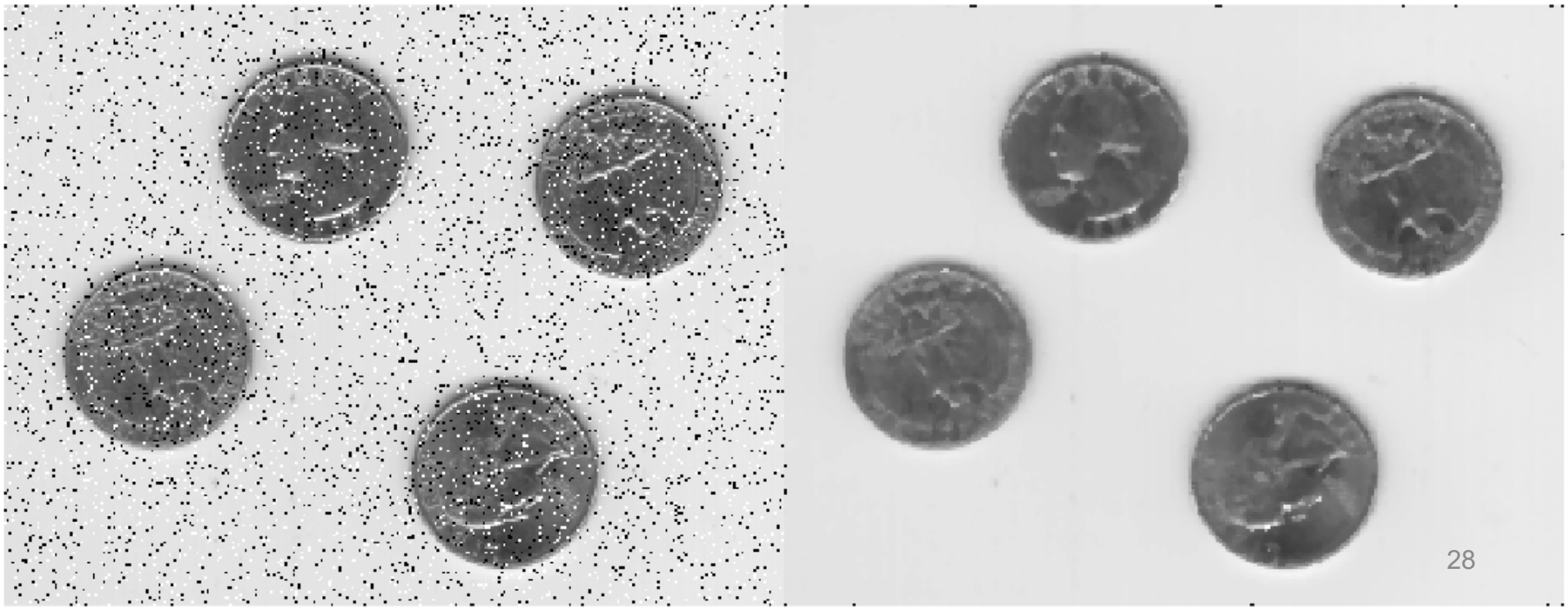
Non-linear spatial filtering (2)

- Median filtering to remove *salt&pepper* noise (3x3 mask)

```
I  = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.08);
K = medfilt2(J);
imshowpair(J,K,'montage')
```

Higher noisy density

Some failures on the boundaries



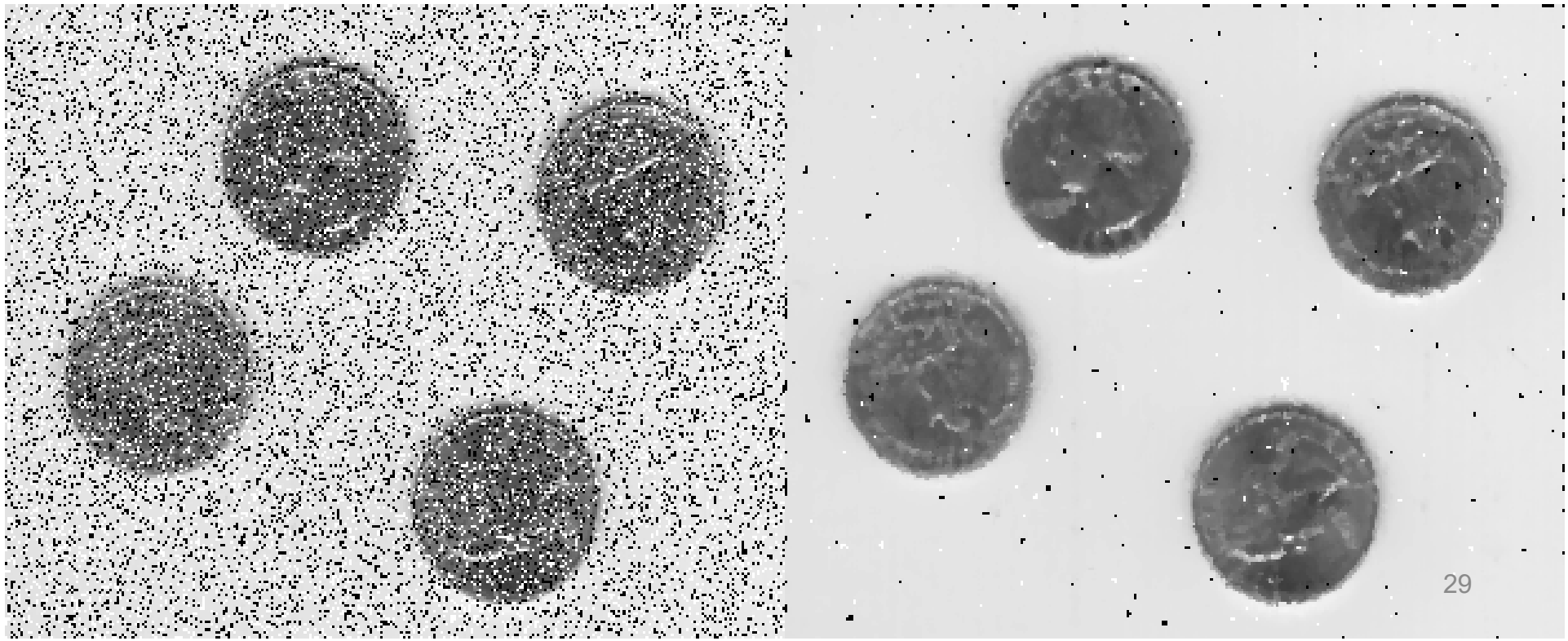
Non-linear spatial filtering (3)

- Median filtering to remove *salt&pepper* noise (3x3 mask)

```
I  = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.25);
K = medfilt2(J);
imshowpair(J,K,'montage')
```

Even higher noisy density

Some failures on the boundaries and on the entire image



Sharpening with the unsharp masking procedure (1)

- The unsharp masking is a sharpening procedure
- It enhances the edges of the image by adding a mask (of edge detail) on the original image

- The algorithm has 3 steps:

1) From the input image f(x,y) compute its blurred/smoothed version, g(x,y), using a smoothing filter (e.g. with an average mask)

2) Compute the image mask m(x,y) = f(x,y) – g(x,y)

3) Add a proportion ‘k’ of the mask to the original image yielding the final image fs(x,y) = f(x,y) + k m(x,y)

Sharpening with the unsharp masking procedure (2)

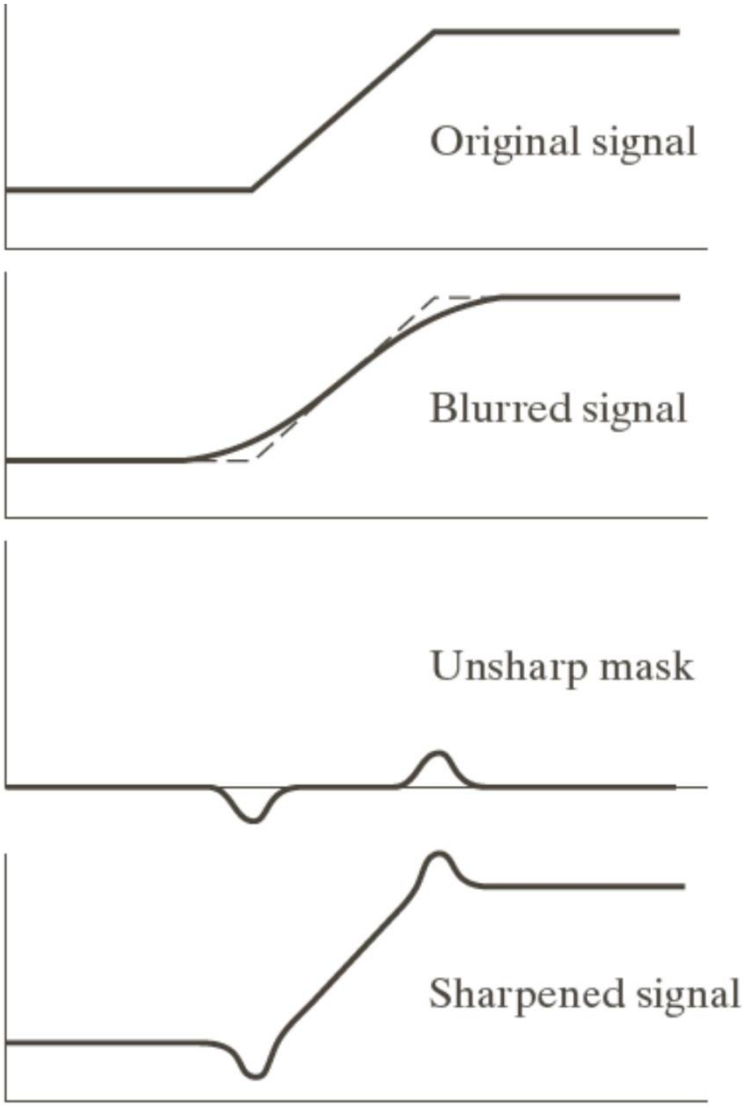
- The third step of the previous algorithm can be generalized as

$$fs(x,y) = f(x,y) + k\, m(x,y),$$

  where k is a suitable constant:

  - k = 1, unsharp masking
  - k > 1, high-boost filtering
  - k < 1, less emphasis on the mask

Sharpening with the unsharp masking procedure (3)



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).

Sharpening with the unsharp masking procedure (4)

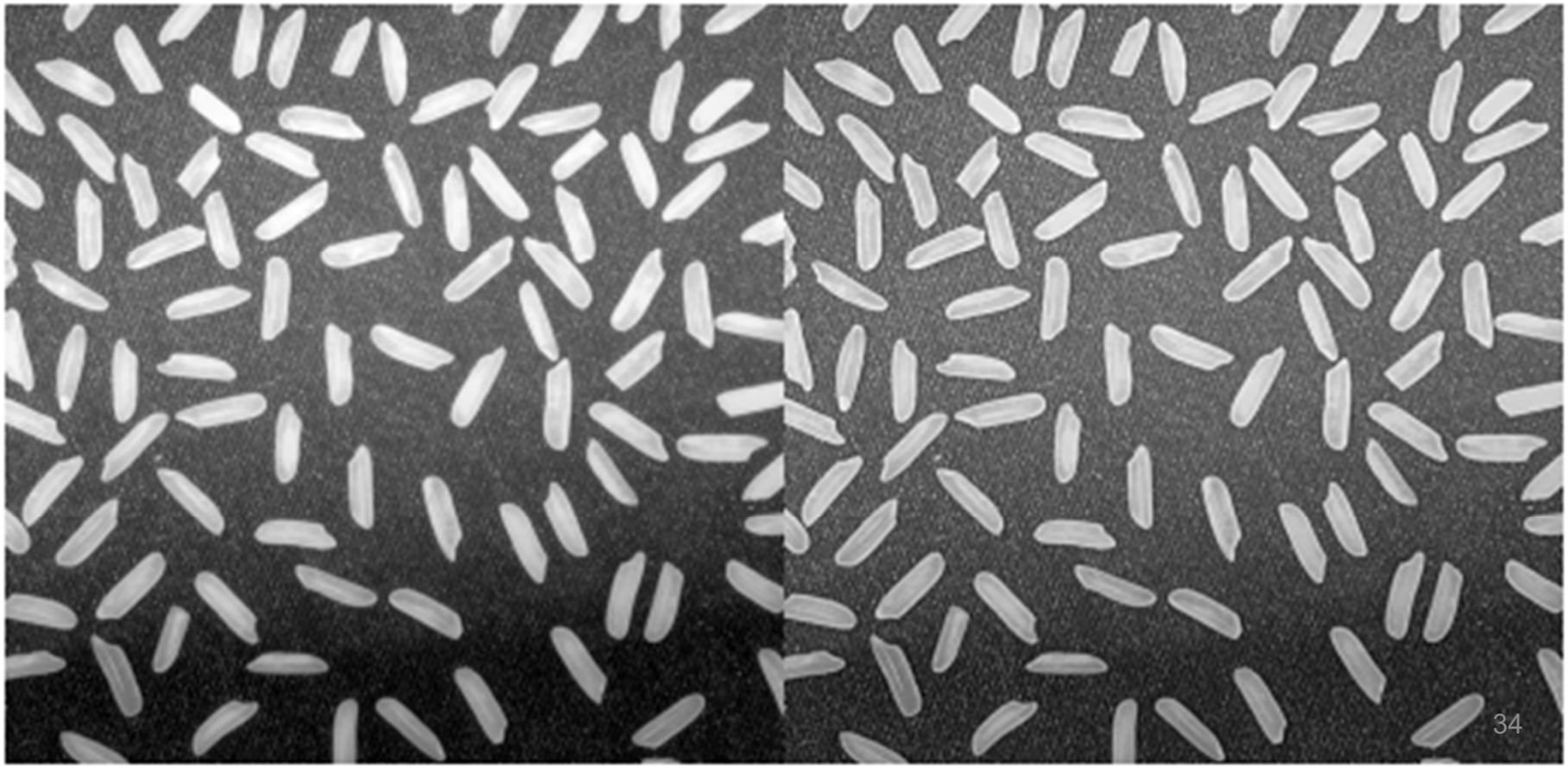
a
b
c
d
e

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.

MATLAB: imsharpen.m

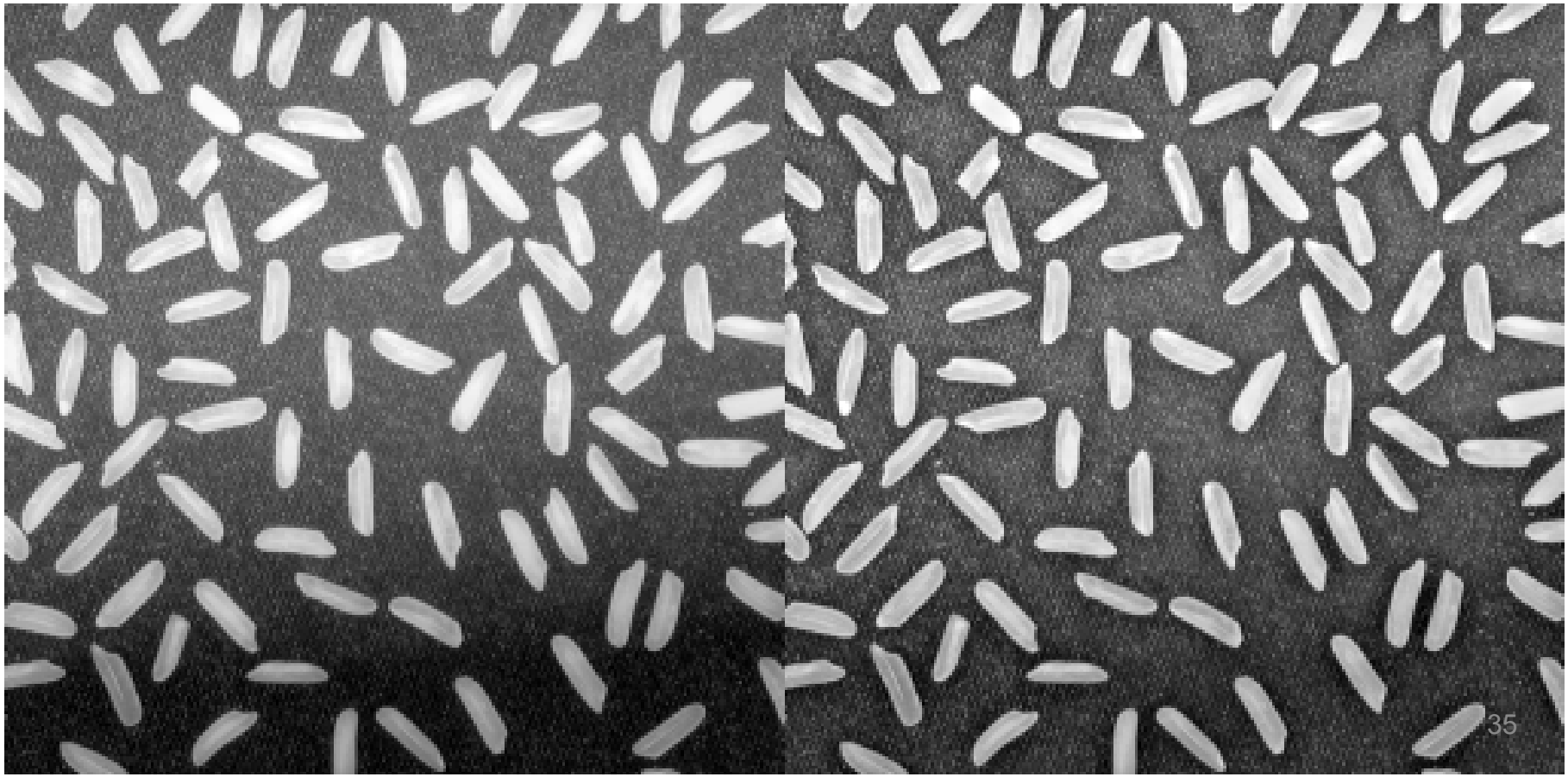
Sharpening with the unsharp masking procedure (5)

```
a = imread('rice.png');
b = imsharpen(a);
imshowpair(a,b,'montage')
```

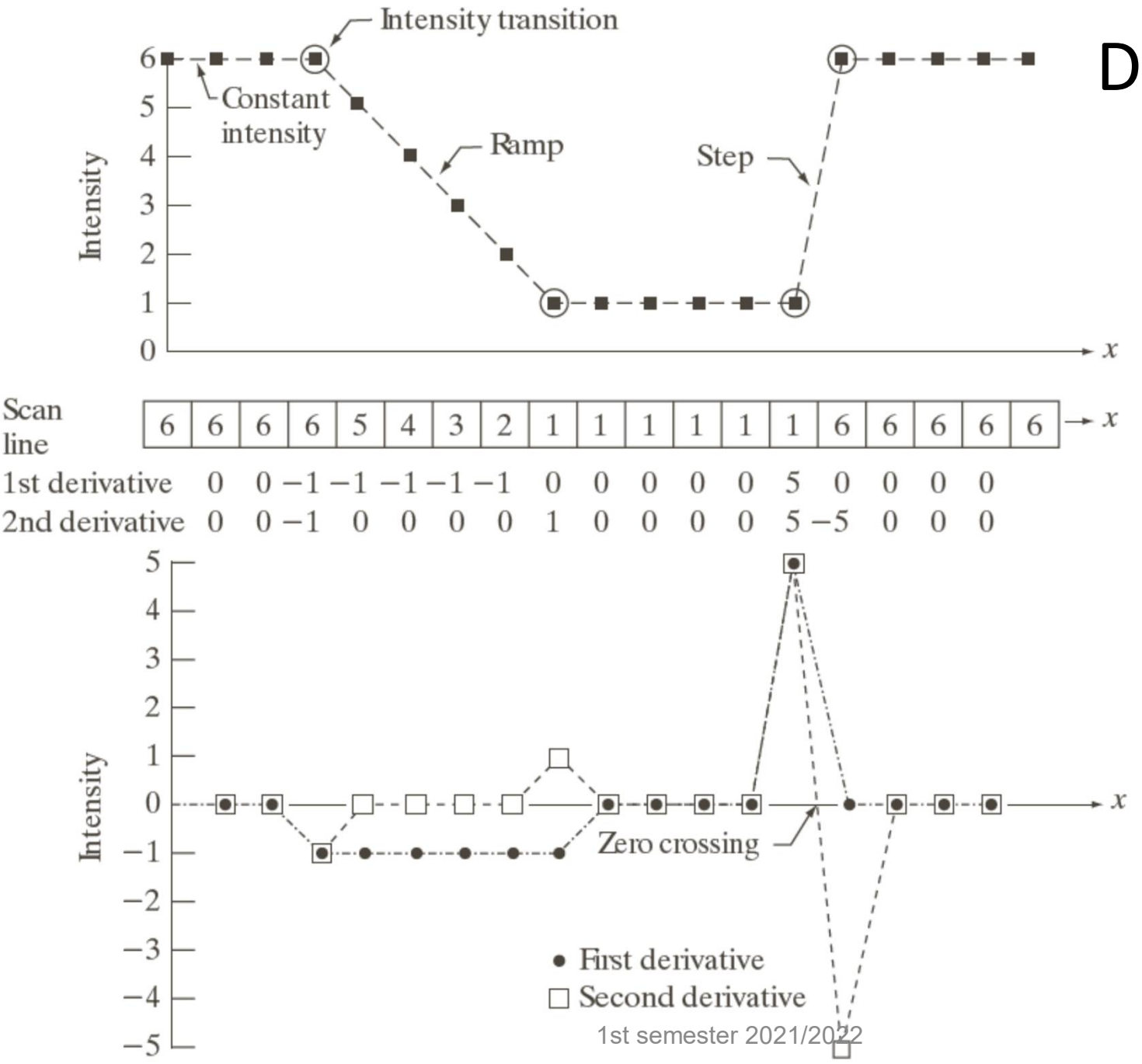


Sharpening with the unsharp masking procedure (6)

```
a = imread('rice.png');
b = imsharpen(a,'Radius',5,'Amount',1);    % Gaussian smoothing filter
imshowpair(a,b,'montage')
```



Derivatives (1)

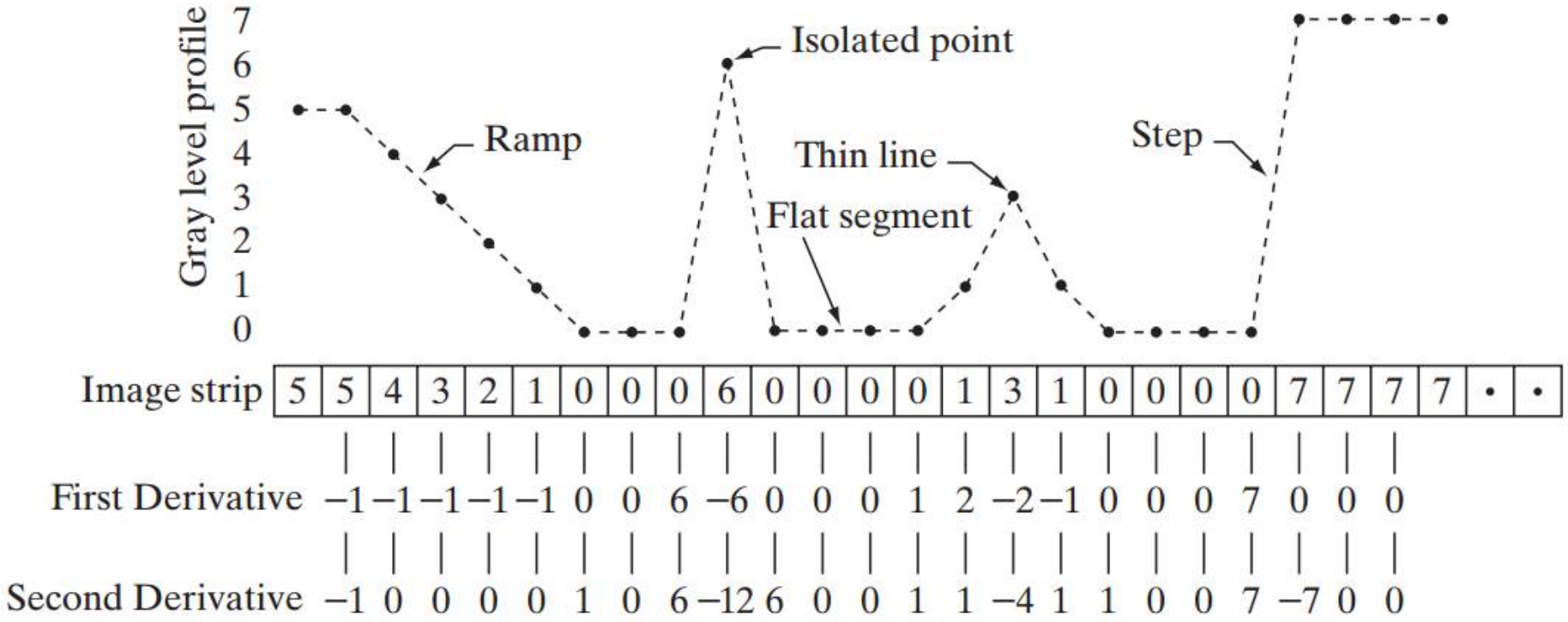


a
b
c

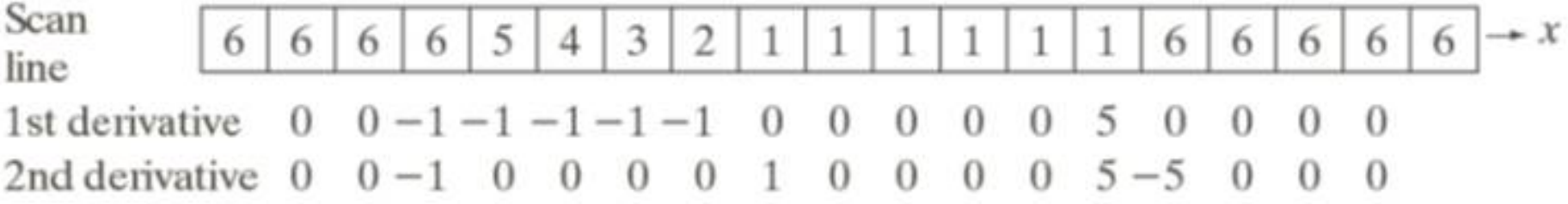
FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

Derivatives (2)

A similar example



First order and second order derivatives (1)



At each point:

- the first order derivative is given by

$f'(x) = f(x+1) - f(x)$, with coefficients [1 -1]

- the second order derivative is given by

$f''(x) = f(x+1) - 2f(x) + f(x-1)$, with coefficients [1 -2 1]

First order and second order derivatives (2)

The Laplacian operator (2nd order derivative):

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

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

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

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y).$$

First order and second order derivatives (3)

The Laplacian operator (2nd order derivative):

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y).$$

0	1	0
1	−4	1
0	1	0

Laplacian masks (1)

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

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.

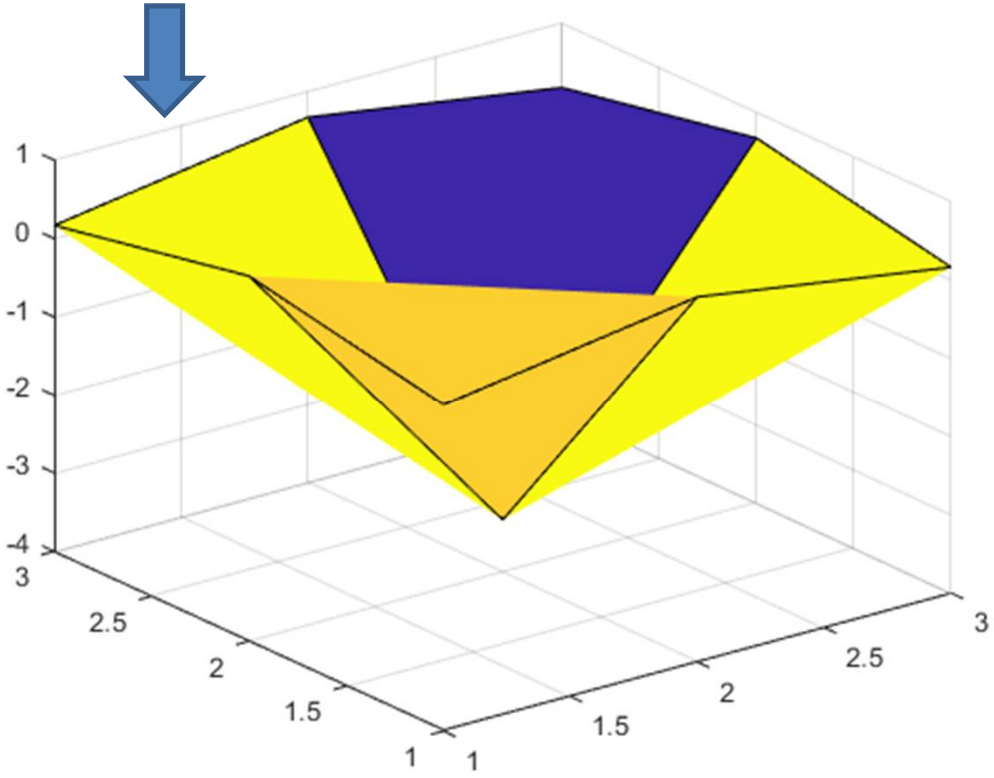
Laplacian masks (2)

Laplacian filters:

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

$$\nabla^2 = \frac{4}{(\alpha+1)} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

h = fspecial('laplacian',0);
surf(h)



https://www.mathworks.com/help/images/ref/fspecial.html

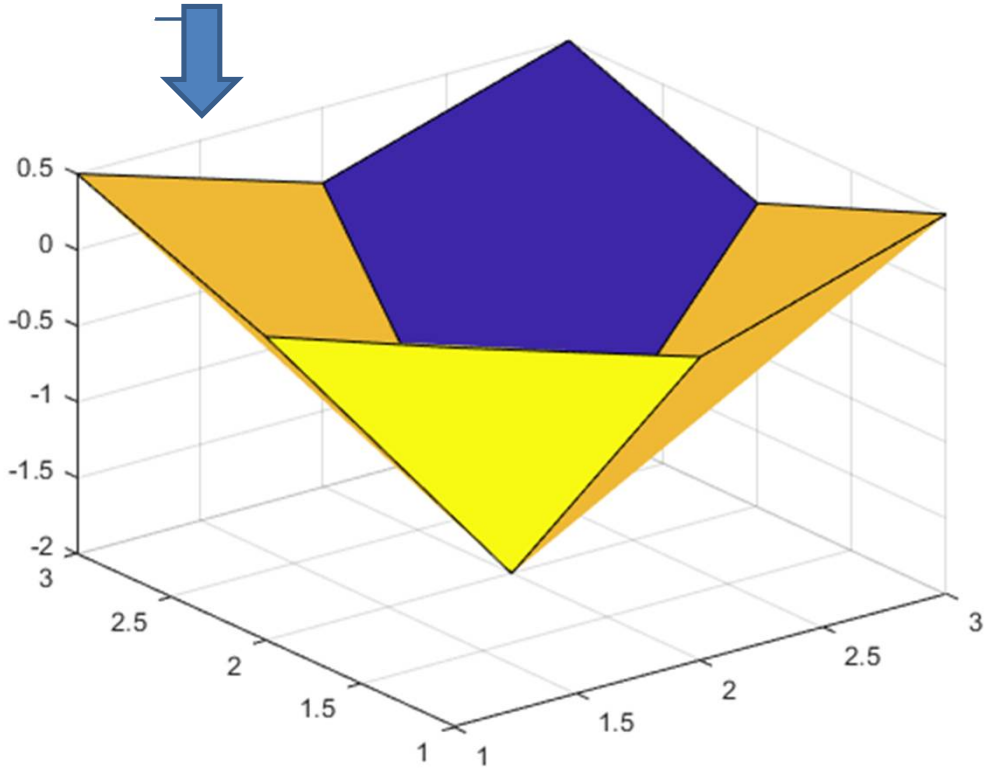
Laplacian masks (3)

Laplacian filters:

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

$$\nabla^2 = \frac{4}{(\alpha+1)} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

h = fspecial('laplacian‘,1);
surf(h)



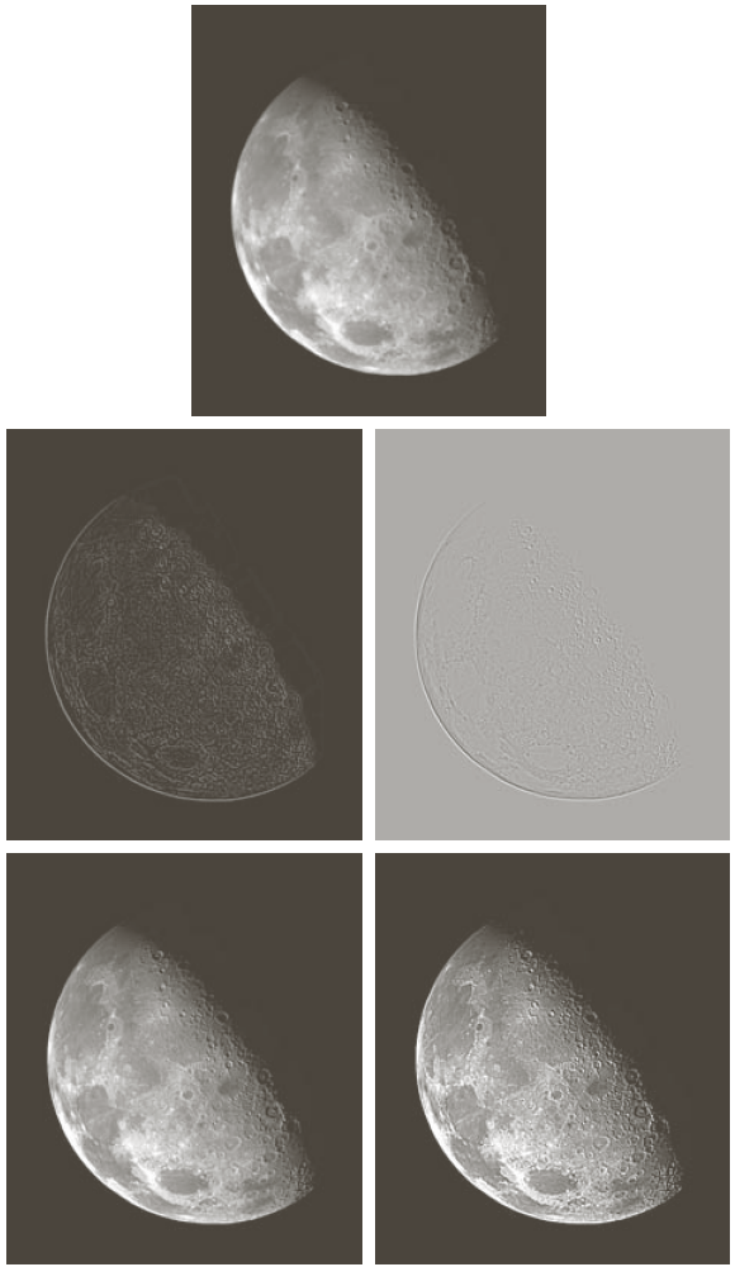
https://www.mathworks.com/help/images/ref/fspecial.html

Image sharpening with the Laplacian (1)

Rationale for sharpening:

add the Laplacian to the original image

subtract the Laplacian from the original image

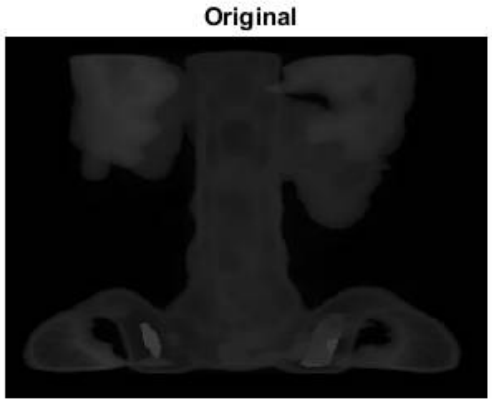


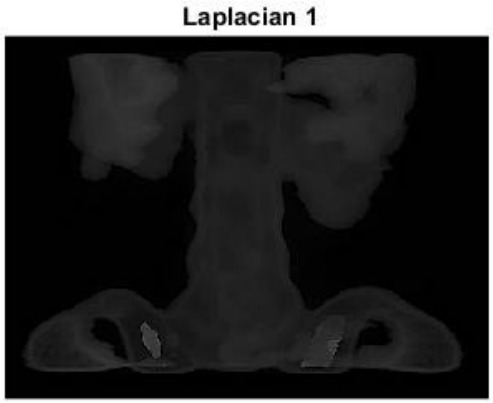
a
b c
d e

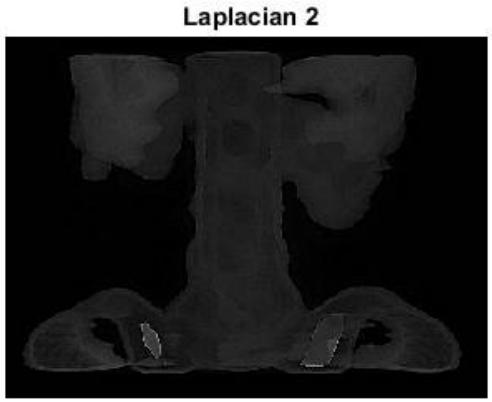
FIGURE 3.38
(a) Blurred image of the North Pole of the moon. (b) Laplacian without scaling. (c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a). (e) Result of using the mask in Fig. 3.37(b). (Original image courtesy of NASA.)

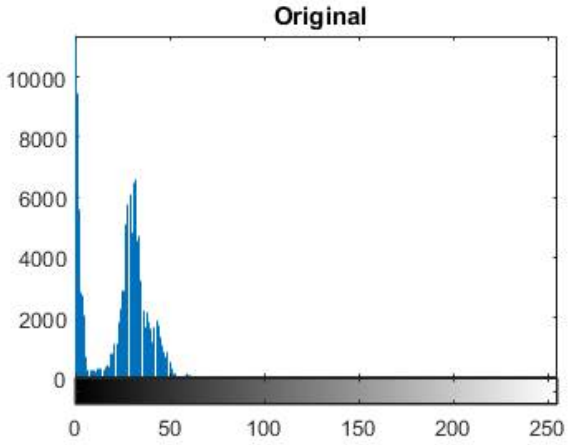
Image sharpening with the Laplacian (2)

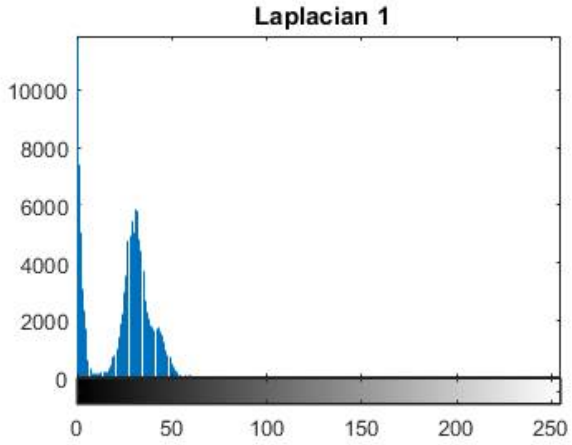
Laplacian 1, is the mask on Figure 3.37 (a)
Laplacian 2, is the mask on Figure 3.37 (b)











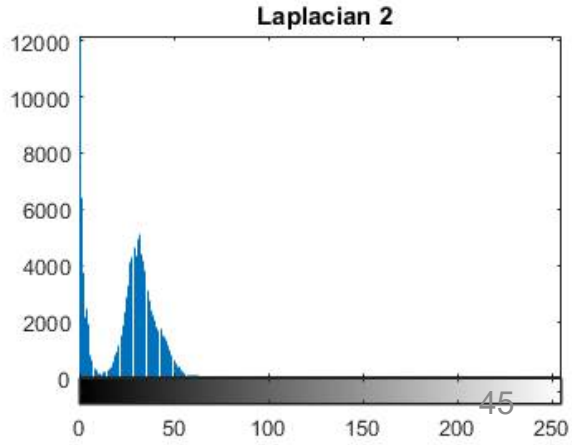
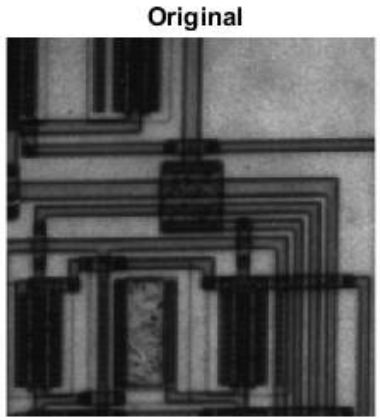
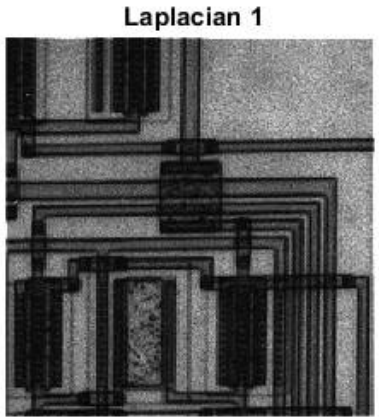
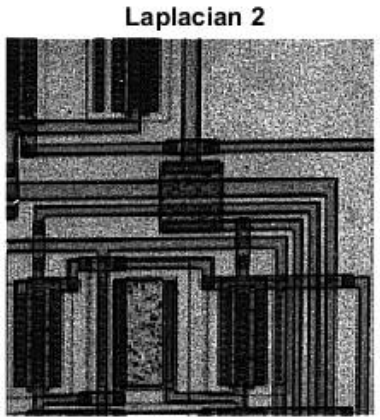


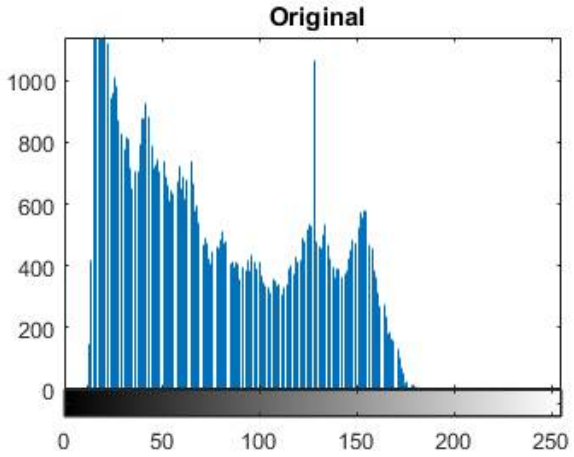
Image sharpening with the Laplacian (3)

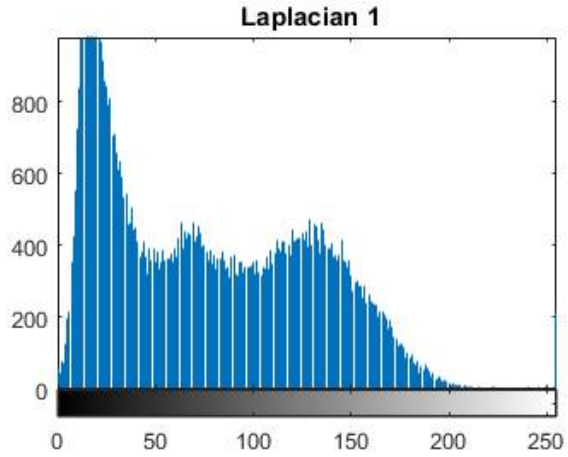
Laplacian 1, is the mask on Figure 3.37 (a)
Laplacian 2, is the mask on Figure 3.37 (b)

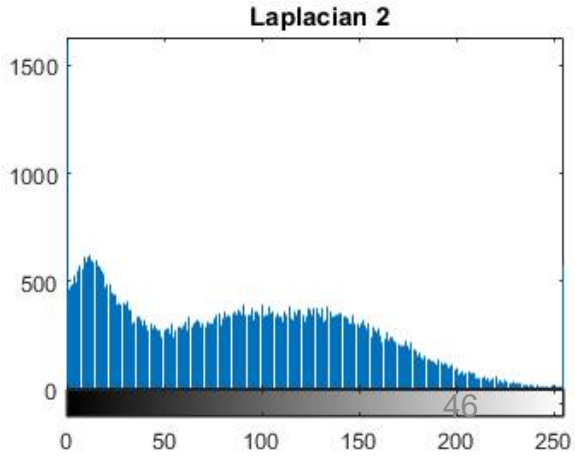












The LoG – Laplacian of Gaussian (1)

- The Laplacian operator is very sensitive to noise (2nd order derivative); it reacts to noise in a way as to edges
- To cope with the noise, the image is first filtered with a low-pass/smoothing Gaussian filter, to remove high-frequency noise
- The Gaussian filter, is defined by the standard deviation sigma, σ
- Then, the Laplacian operator is applied on the smoothed Gaussian-filtered image, with less noise

The LoG – Laplacian of Gaussian (2)

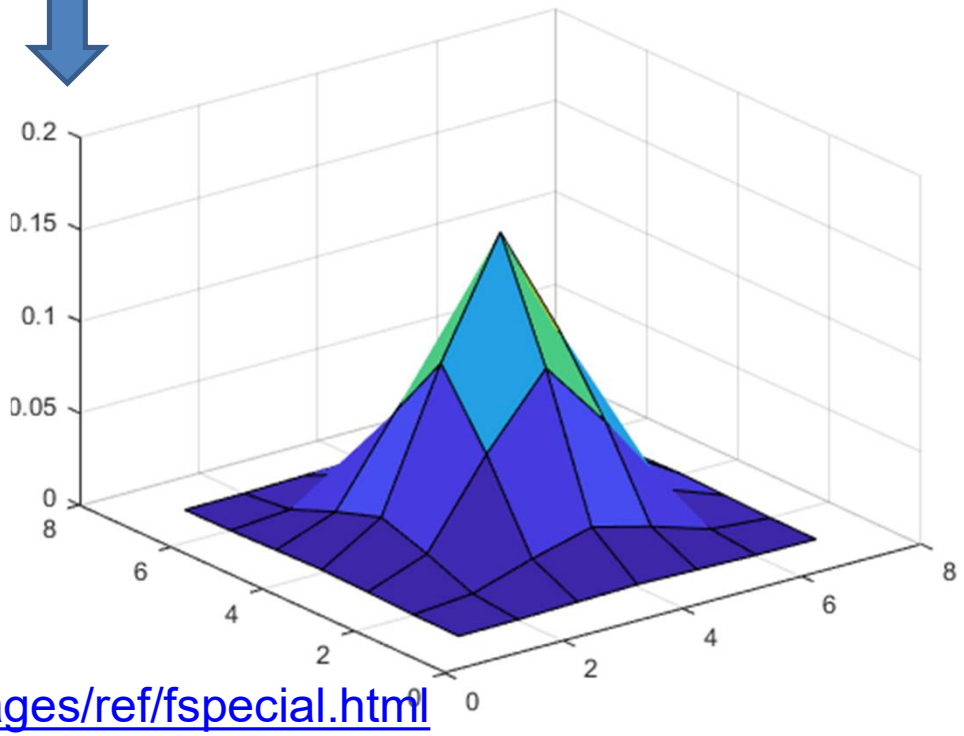
- The Gaussian filter and the standard deviation sigma, σ

Gaussian filters:

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2+n_2^2)}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g}$$

```
h = fspecial('gaussian',7,1);
surf(h)
```



https://www.mathworks.com/help/images/ref/fspecial.html

The LoG – Laplacian of Gaussian (3)

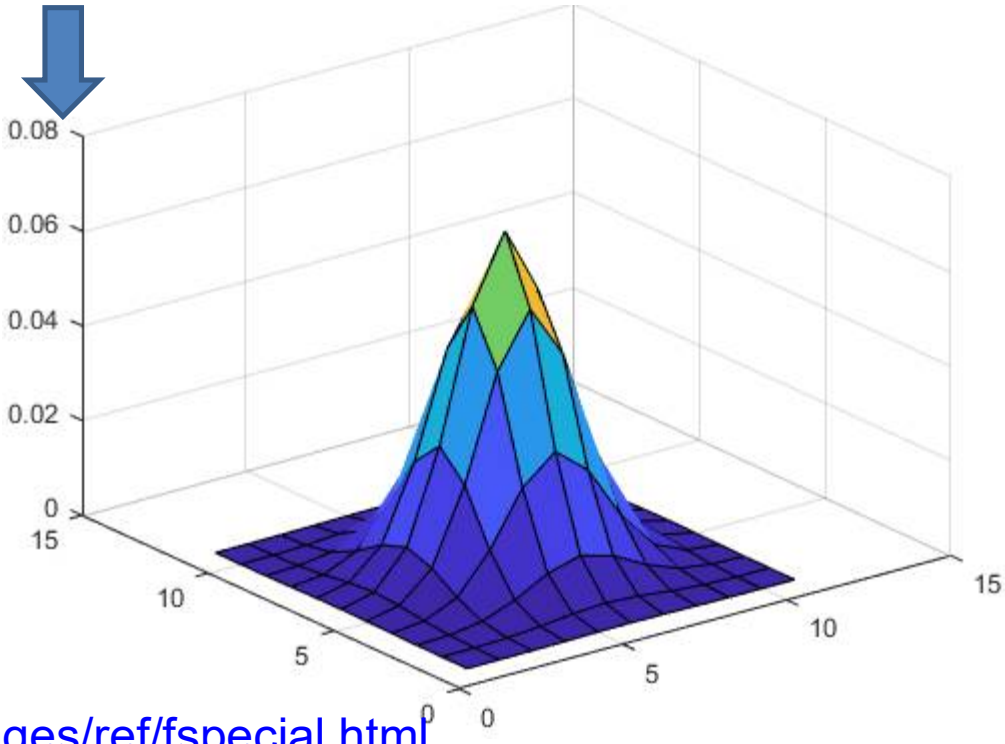
- The Gaussian filter and the standard deviation sigma, σ

Gaussian filters:

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2+n_2^2)}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g}$$

h = fspecial('gaussian',11,1.5);
surf(h)



https://www.mathworks.com/help/images/ref/fspecial.html

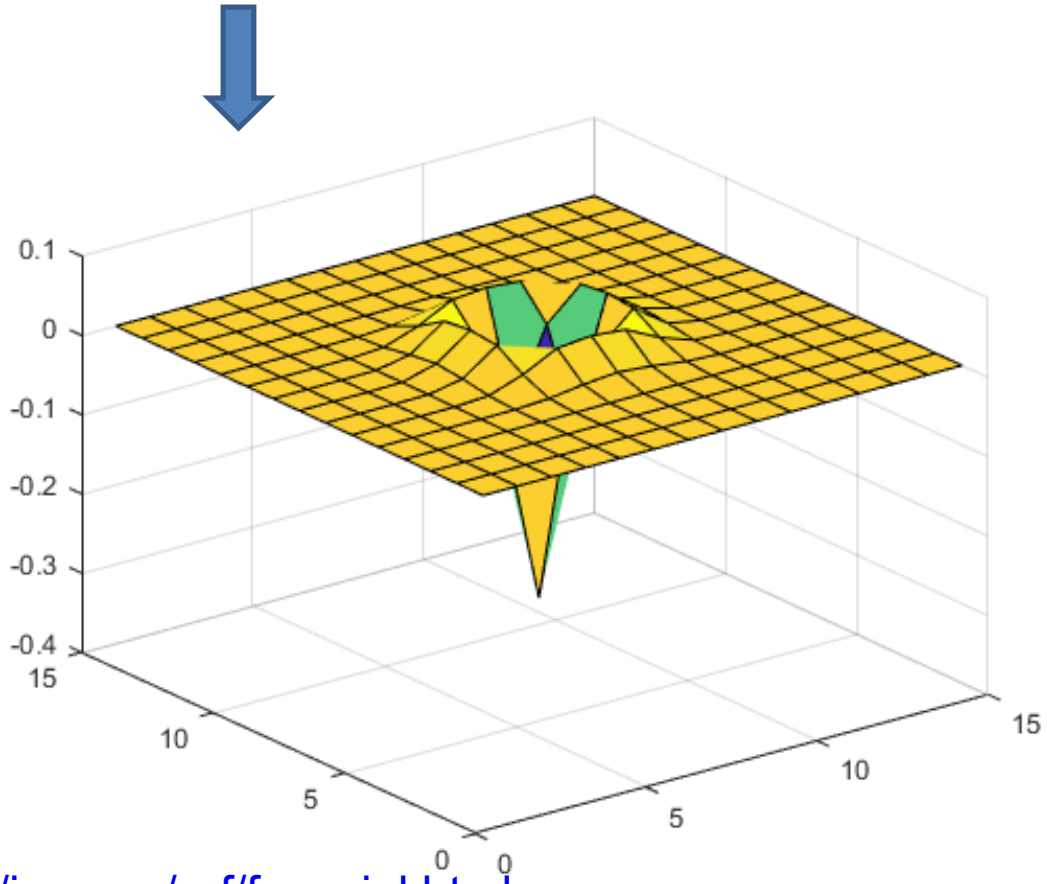
The LoG – Laplacian of Gaussian (4)

Laplacian of Gaussian (LoG) filters:

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2+n_2^2)}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{(n_1^2 + n_2^2 - 2\sigma^2)h_g(n_1, n_2)}{\sigma^4 \sum_{n_1} \sum_{n_2} h_g}$$

h = fspecial('log', 15, 1);
surf(h)



https://www.mathworks.com/help/images/ref/fspecial.html

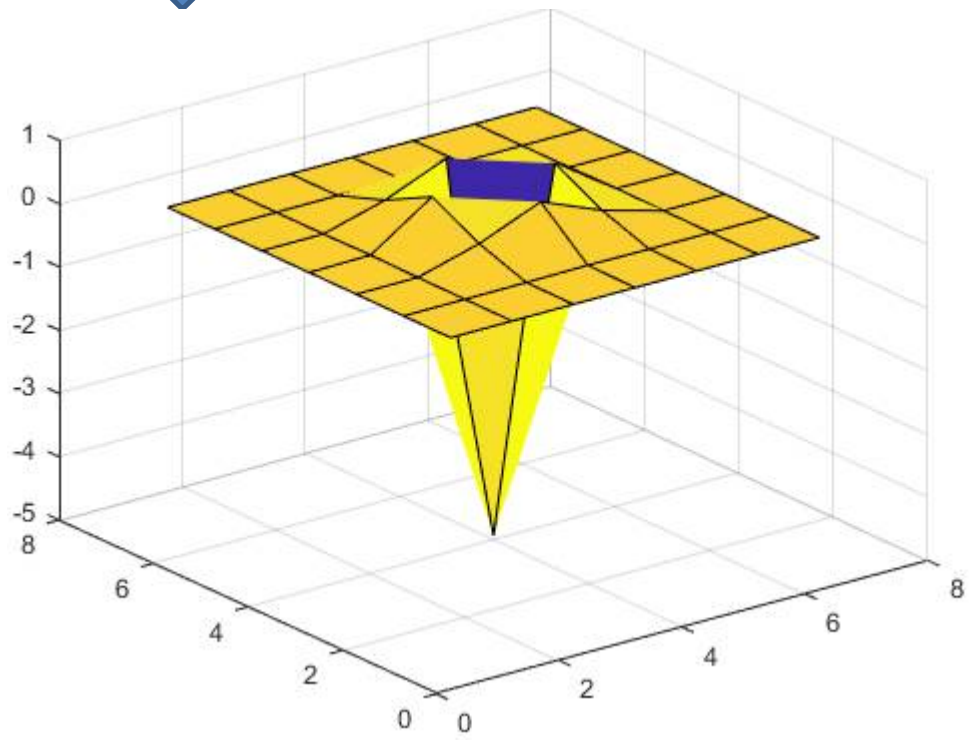
The LoG – Laplacian of Gaussian (5)

Laplacian of Gaussian (LoG) filters:

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2+n_2^2)}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{(n_1^2 + n_2^2 - 2\sigma^2)h_g(n_1, n_2)}{\sigma^4 \sum_{n_1} \sum_{n_2} h_g}$$

h = fspecial('log', 7, .5);
surf(h)



https://www.mathworks.com/help/images/ref/fspecial.html

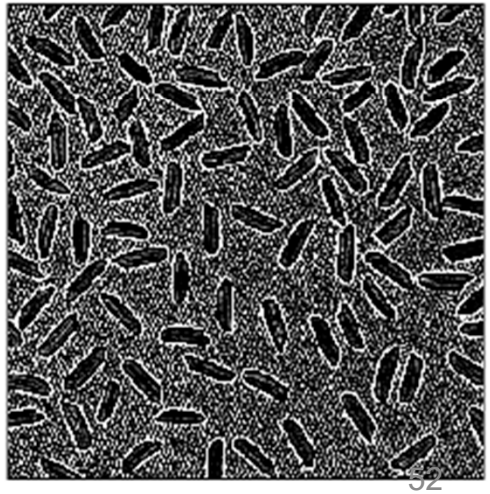
Laplacian and LoG – Some Results (1)

Original

Laplacian 1

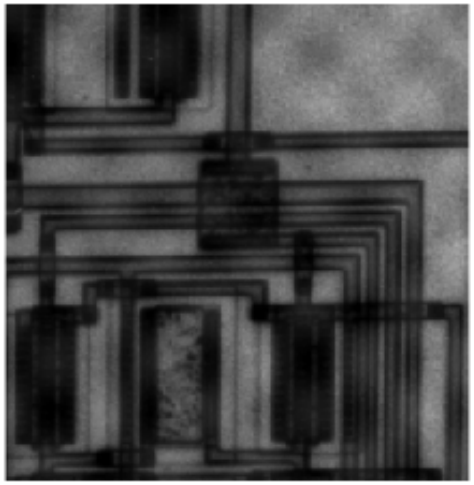
Laplacian 2

LoG

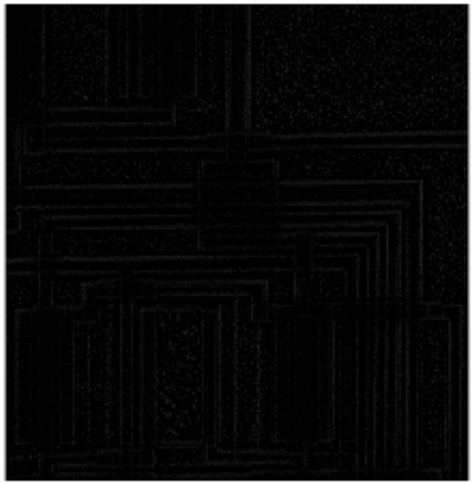


Laplacian and LoG – Some Results (2)

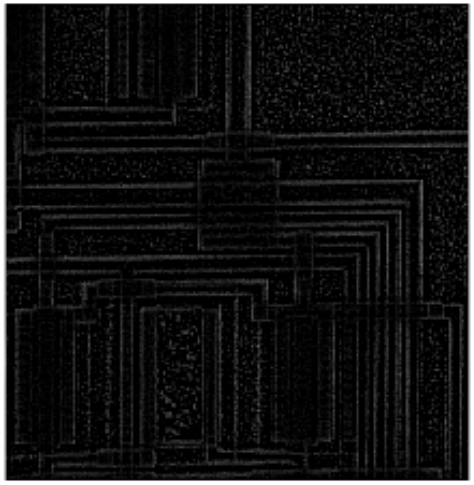
Original



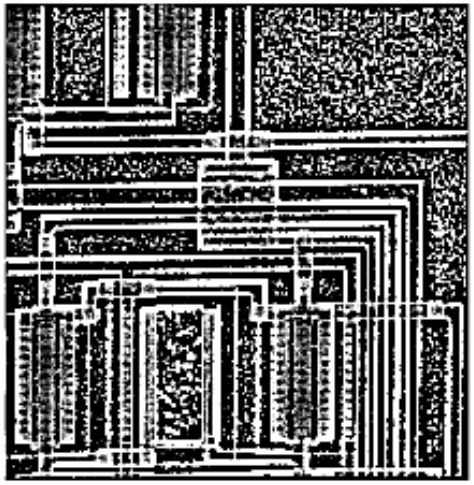
Laplacian 1



Laplacian 2

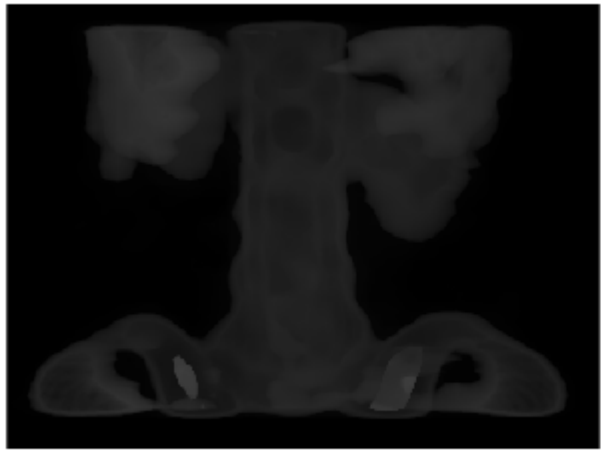


LoG



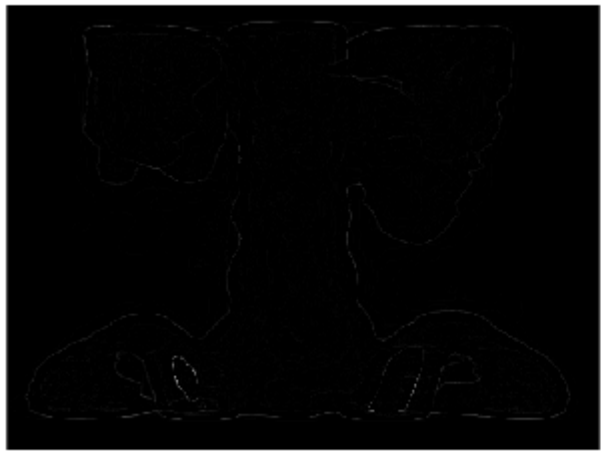
Laplacian and LoG – Some Results (3)

Original



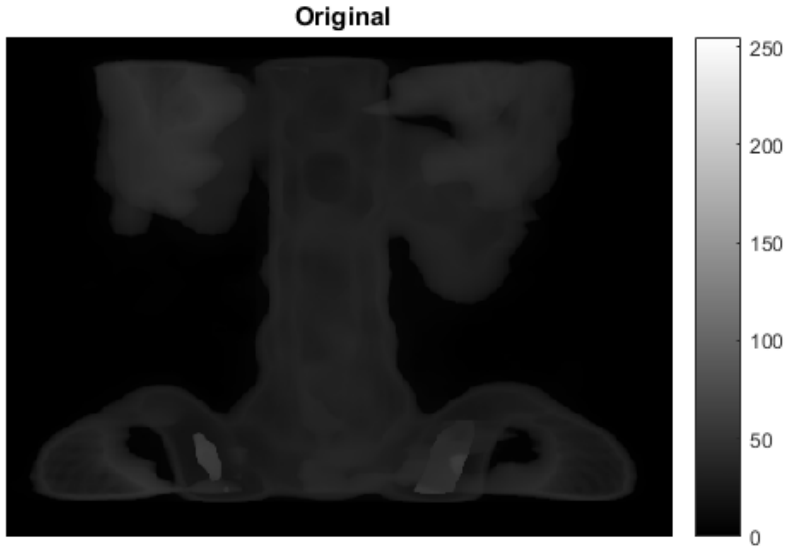
Laplacian 1

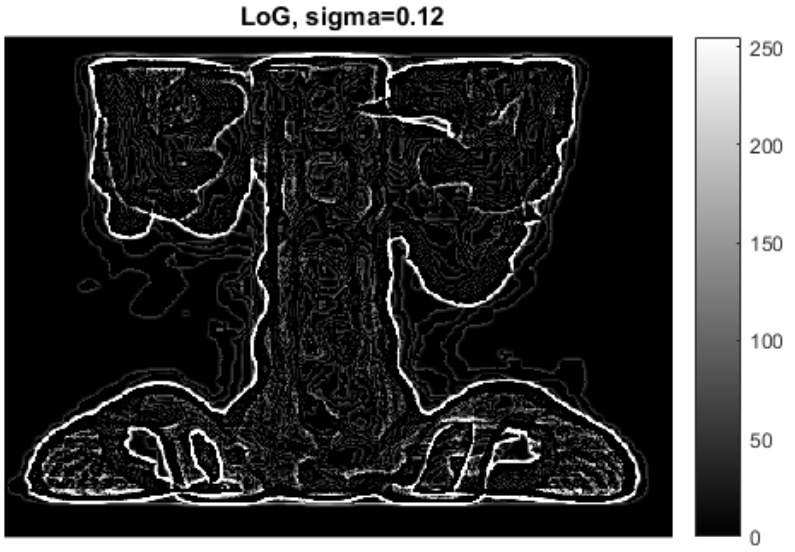
Laplacian 2

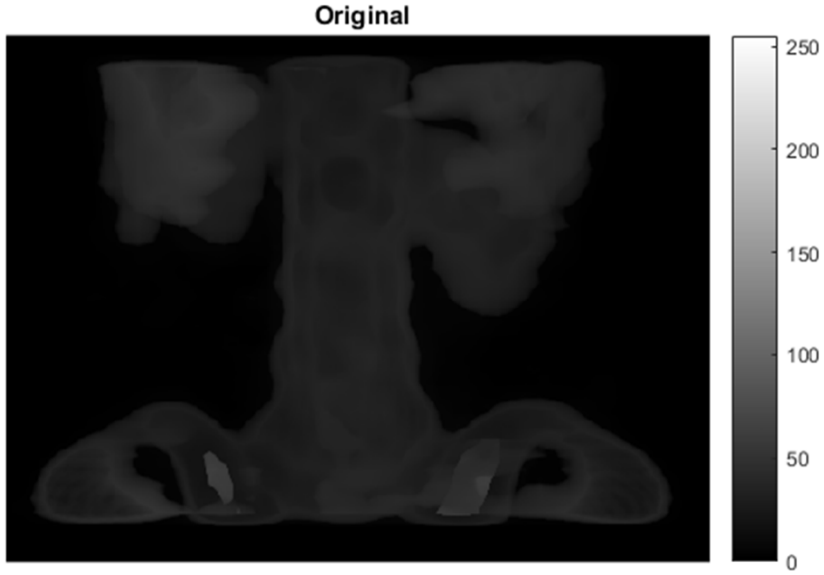


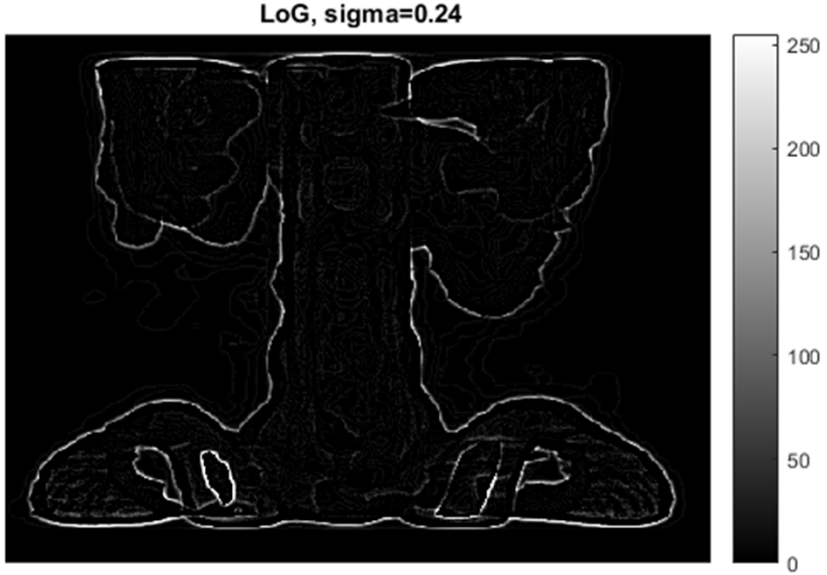
LoG

Laplacian and LoG – Some Results (4)









MATLAB
Image Processing Toolbox functions

https://www.mathworks.com/products/image.html

- *filter2, imfilter,* image filtering
- *medfilt2,* median filter
- *imsharpen,* unsharp masking technique
- *fspecial,* kernels/windows/masks for common filters

Bibliography

- The images displayed in these slides are from:
  - R. Gonzalez, R. Woods, *Digital Image Processing*, 4th edition, Prentice Hall, 2018, ISBN 0133356728
  - S. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, Newnes, 2003, ISBN 0-750674-44-X [chapter 23]
  - O. Filho, H. Neto, Processamento Digital de Imagens, Rio de Janeiro: Brasport, 1999, ISBN 8574520098.
  - Wikipedia and Mathworks web pages