AREL ÜNİVERSİTESİ
BİYOMEDİKAL GÖRÜNTÜ İŞLEME

UZAYSAL FİLTRELEME

DR. GÖRKEM SERBES

Lokal Komşuluk İlişkileri

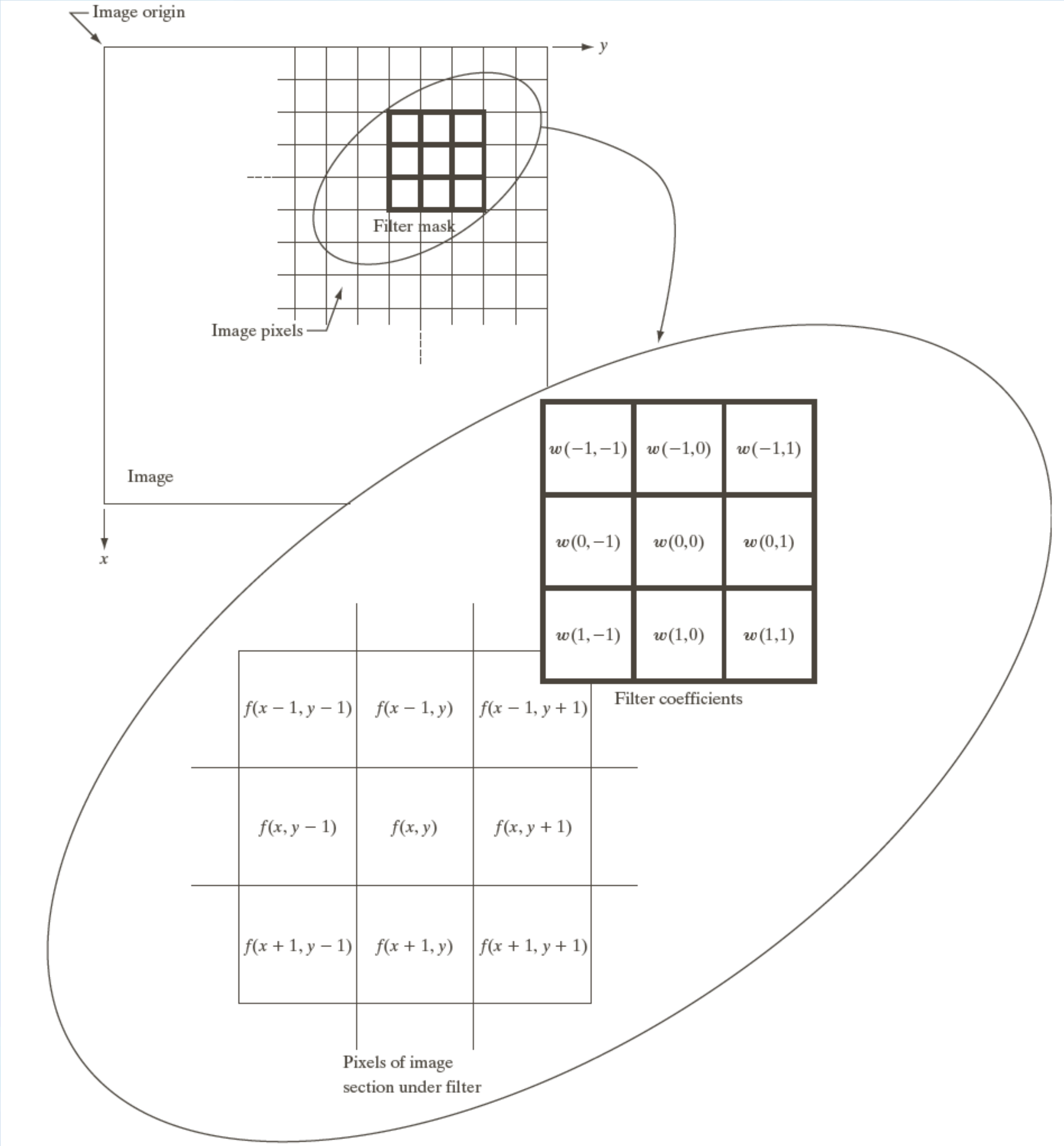


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

Filtre Maskesi

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.

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

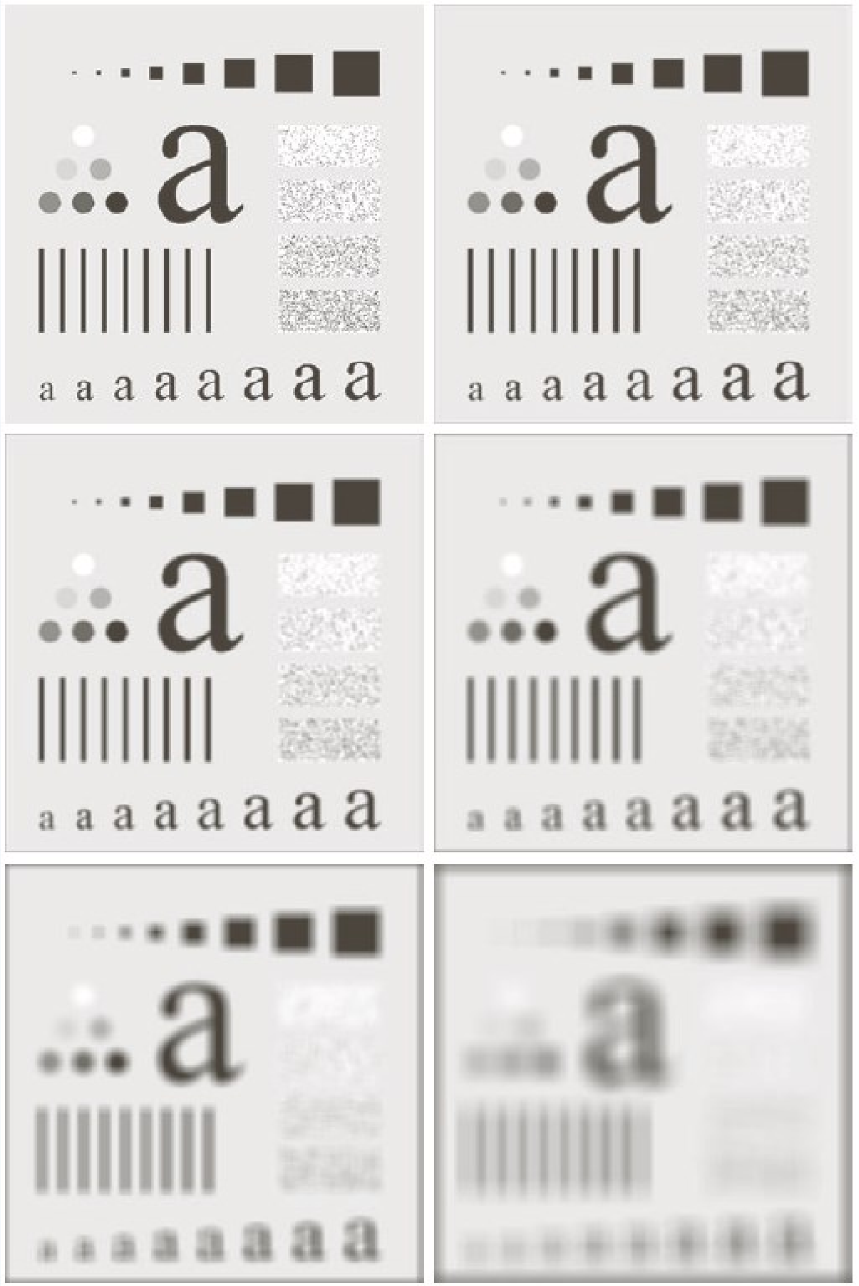
a b

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.

Filtre Boyutu Etkisi

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



Ortalama Alan Filtre Örneği

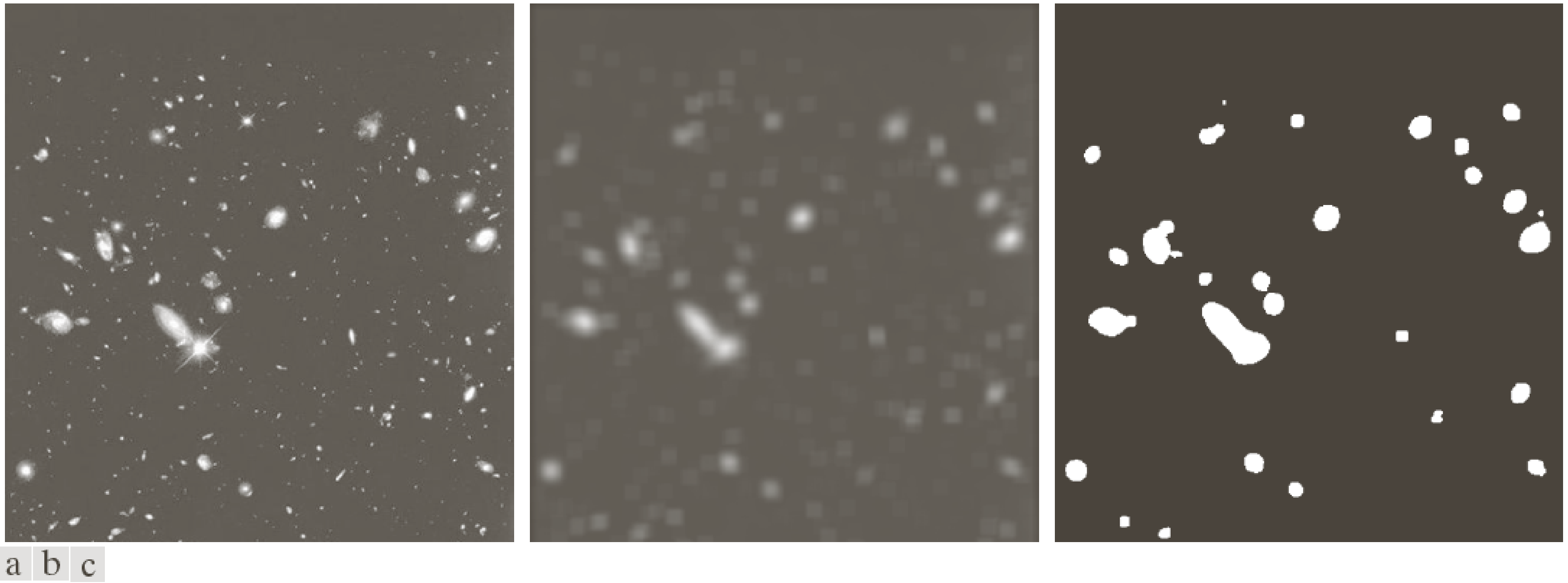
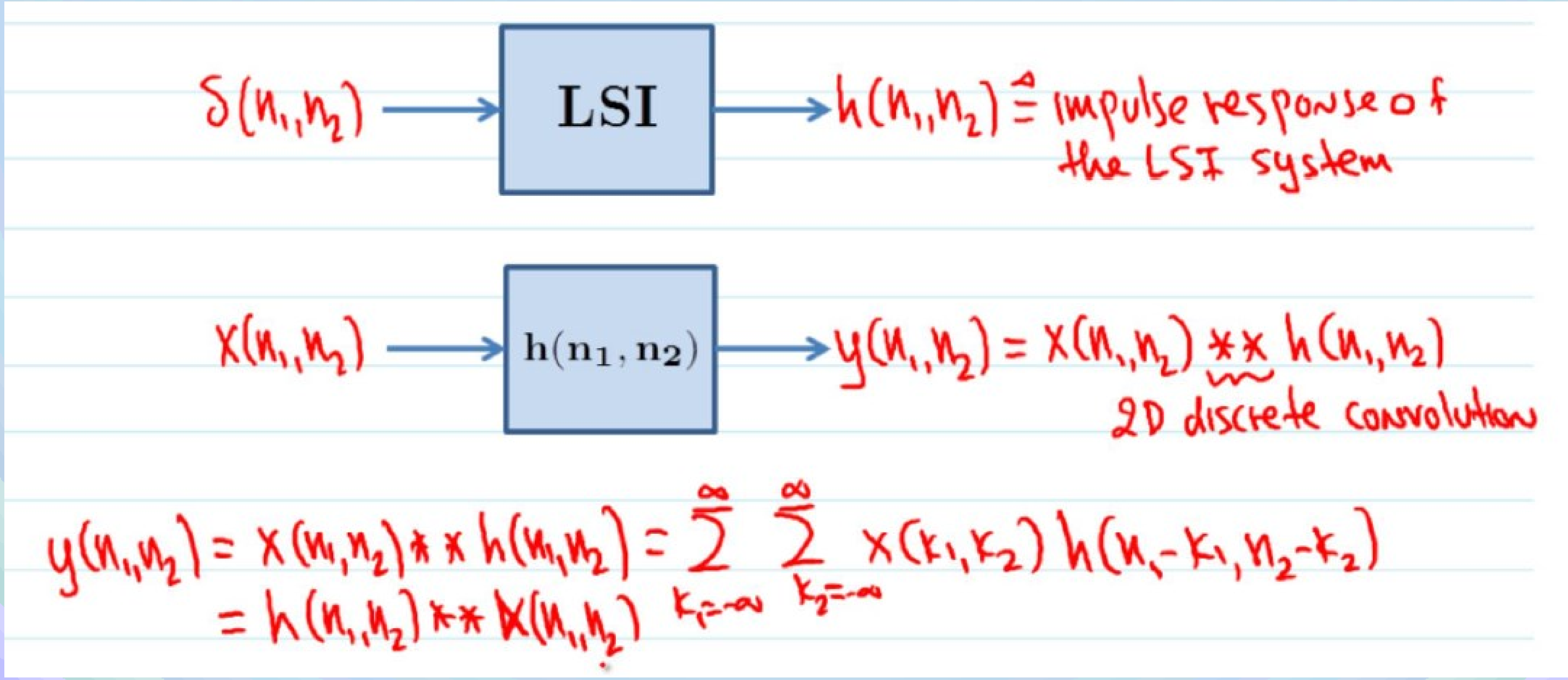


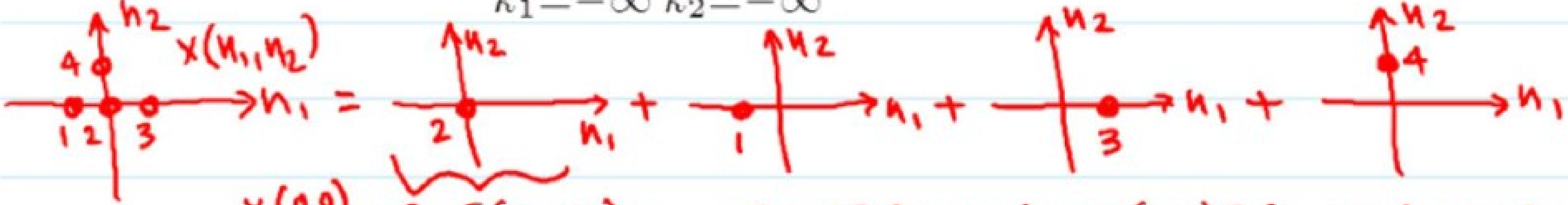
FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Konvolüsyon



Konvolüsyon Formül Çıkarımı

$$\underline{x(n_1,n_2)} = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} \underline{x(k_1,k_2)}\underbrace{\delta(n_1-k_1,n_2-k_2)}$$



$$\underbrace{x(0,0)}\longrightarrow 2\cdot\delta(n_1,n_2) + x(-1,0)\,\delta(n_1+1,n_2) + x(1,0)\,\delta(n_1-1,n_2) + x(0,1)\,\delta(n_1,n_2-1)$$

$$y(n_1,n_2) = T[x(n_1,n_2)] = T\Big[\sum_{k_1}\sum_{k_2} \overbrace{x(k_1,k_2)}^{\text{weight}}\underbrace{\delta(n_1-k_1,n_2-k_2)}\Big]$$

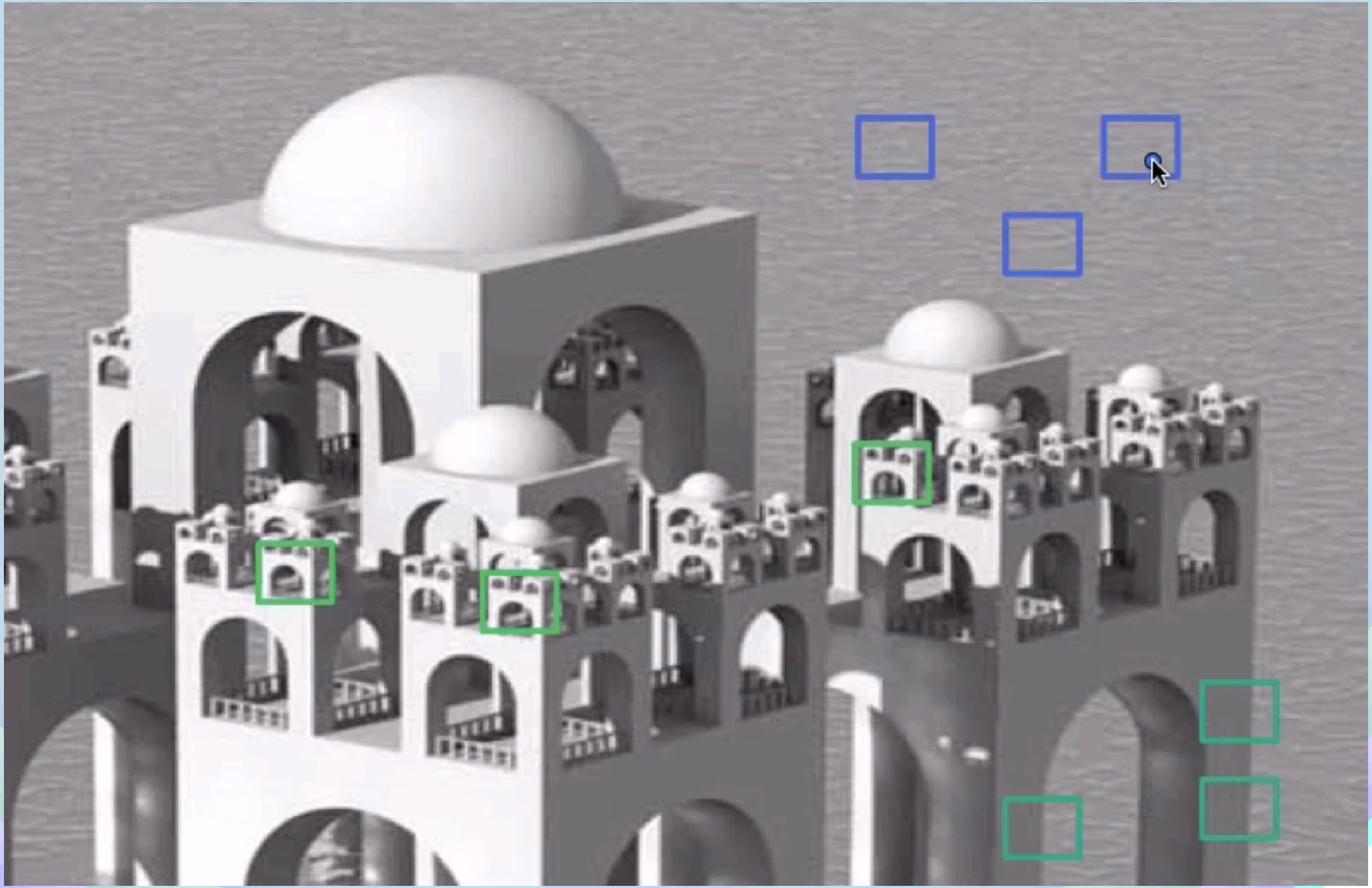
$$\overset{\text{linearity}}{=} \sum_{k_1}\sum_{k_2} x(k_1,k_2)\, T[\delta(n_1-k_1,n_2-k_2)]$$

$$\overset{\text{SI}}{=} \sum_{k_1}\sum_{k_2} x(k_1,k_2)\cdot h(n_1-k_1,n_2-k_2) = x(n_1,n_2) ** h(n_1,n_2)$$

$$= h(n_1,n_2) ** x(n_1,n_2)$$

Konvolüsyon - Demo

Lokal Olmayan Ortalama



Lokal Olmayan Ortalama - Demo

www.ipol.im/pub/art/2011/bcm_nlm/

IPOL Journal · Image Processing On Line

HOME · ABOUT · ARTICLES · PREPRINTS · WORKSHOPS · NEWS · SEARCH

Non-Local Means Denoising

Antoni Buades, Bartomeu Coll, Jean-Michel Morel

article | demo | archive

published • 2011-09-13 → BibTeX

reference • Antoni Buades, Bartomeu Coll, and Jean-Michel Morel, *Non-Local Means Denoising*, Image Processing On Line, 1 (2011), pp. 208–212. https://doi.org/10.5201/ipol.2011.bcm_nlm

Communicated by Guoshen Yu
Demo edited by Miguel Colom

Abstract

We present in this paper a new denoising method called non-local means. The method is based on a simple principle: replacing the color of a pixel with an average of the colors of similar pixels. But the most similar pixels to a given pixel have no reason to be close at all. It is therefore licit to scan a vast portion of the image in search of all the pixels that really resemble the pixel one wants to denoise. The paper presents two implementations of the method and displays some results.

Download

- full text manuscript: PDF (663K)
- source code: TAR/GZ

Lokal Olmayan Ortalama - Demo

article | demo | archive

Please cite the reference article if you publish results obtained with this online demo.

The algorithm result is displayed hereafter. It ran in 15.29s.
You can run again this algorithm with new data.

Run again?: new input | different parameter or subimage

Results (sigma: 35)

- Noisy
- Denoised
- Original
- Difference

Lokal Olmayan Ortalama - Demo

article | demo | archive

Please cite the reference article if you publish results obtained with this online demo.

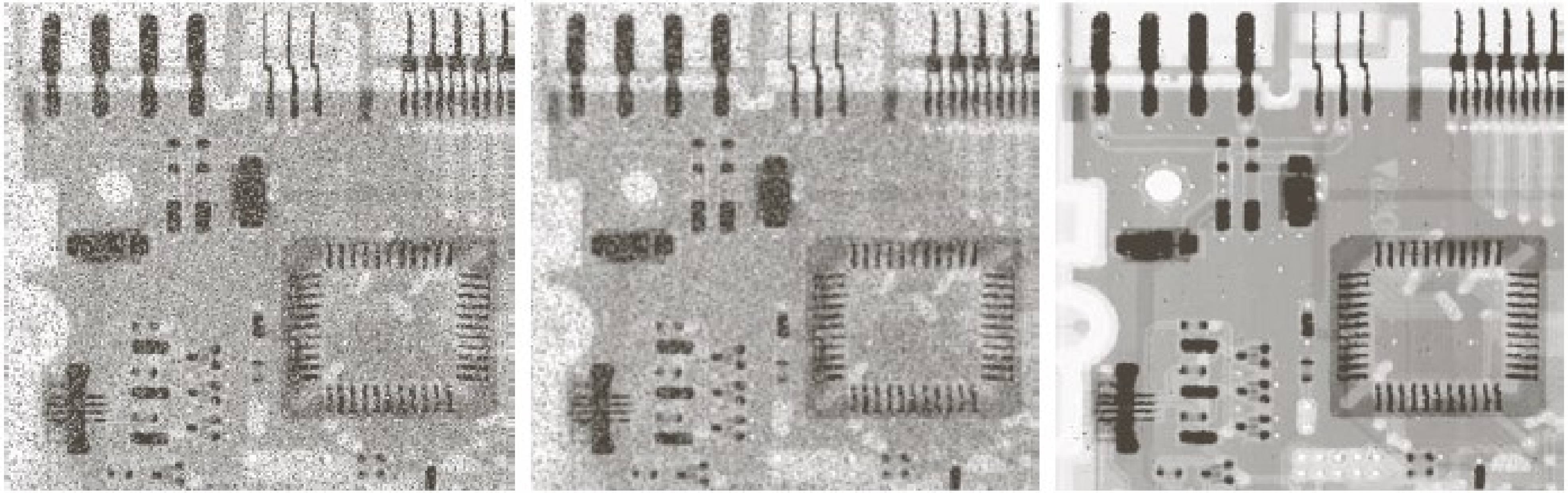
The algorithm result is displayed hereafter. It ran in 15.29s.
You can run again this algorithm with new data.

Run again?: new input | different parameter or subimage

Results (sigma: 35)

- Noisy
- Denoised
- Original
- Difference

Medyan Filtre

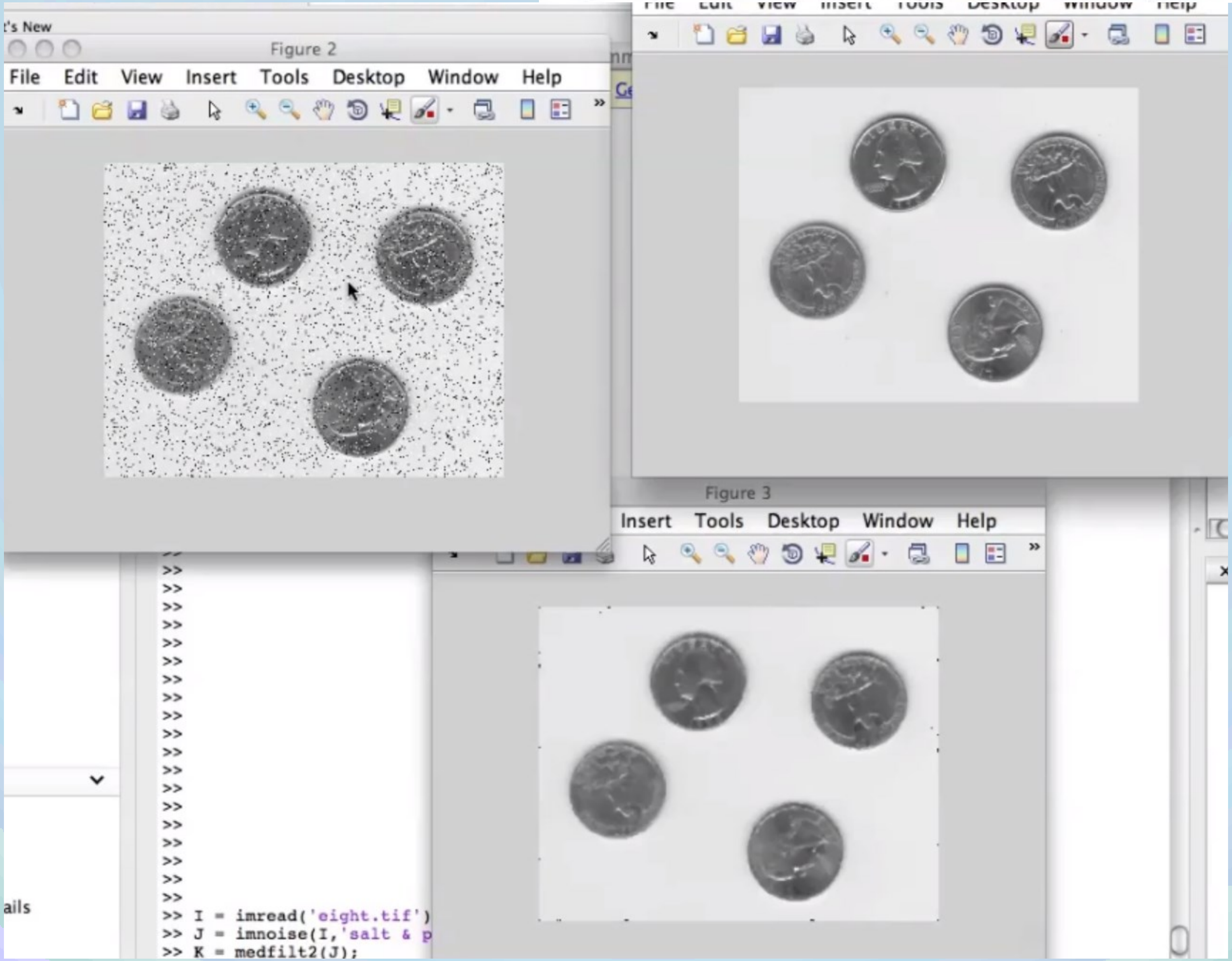


a b c

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.)

Medyan Filtre Demo

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.09);
K = medfilt2(J);
figure, imshow(I); figure, imshow(J), figure, imshow(K)
```



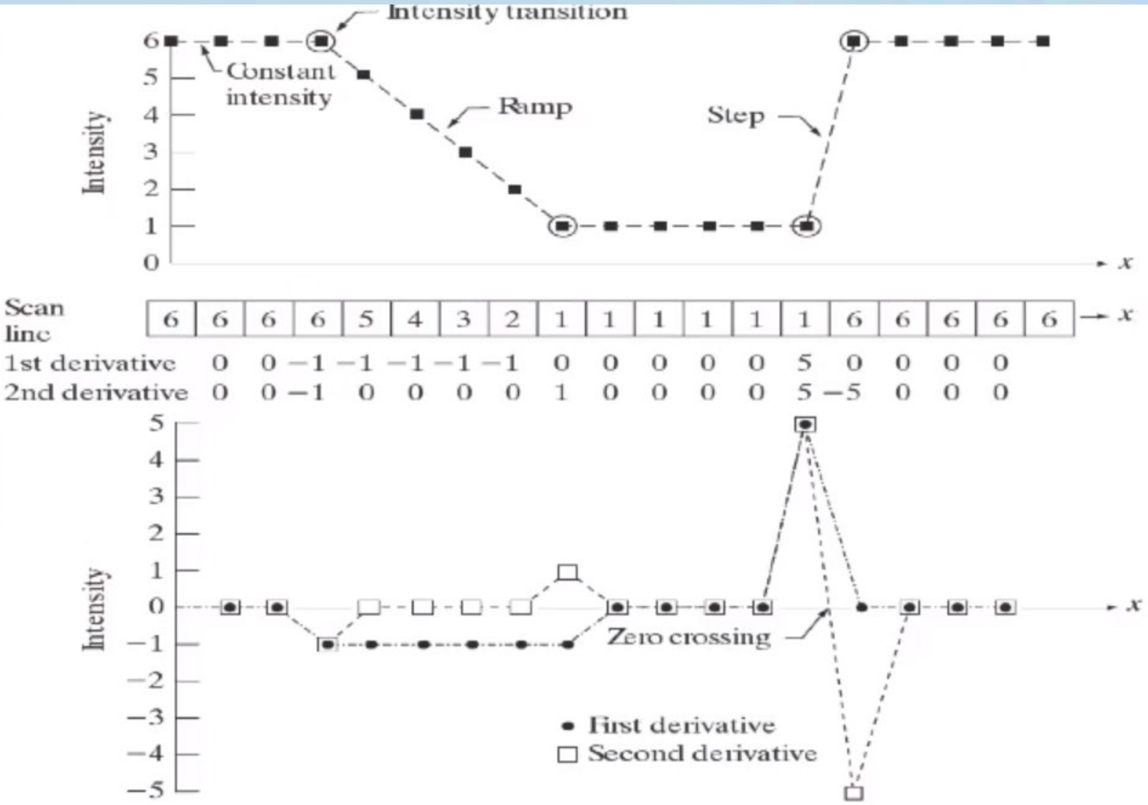
Türev ve Laplacian

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

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

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

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$



Türev ve Laplacian

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

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

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

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

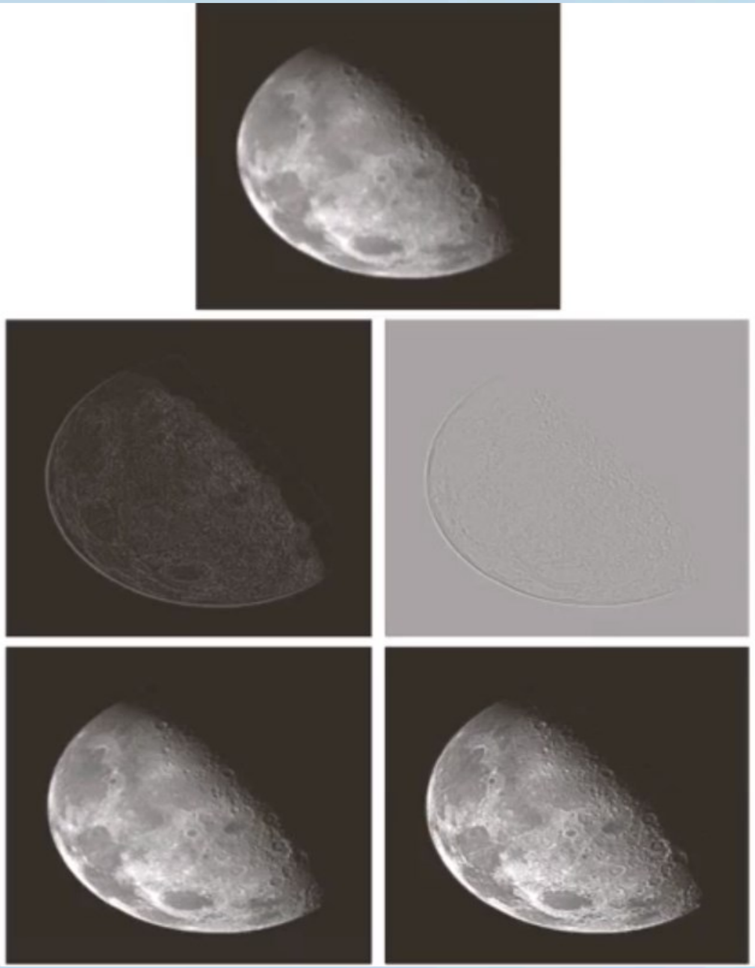
Türev ve Laplacian

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



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.)

Türev ve Laplacian - Demo

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper', 0.09);
K = medfilt2(J);
figure, imshow(I); figure, imshow(J), figure, imshow(K), figure, imshow(I-K), figure, imshow((I-K).^(2));
```