NOISE

What is noise? *Noise* is any undesired information that contaminates an image. Noise appears in images from a variety of sources. The digital image acquisition process, which converts an optical image into a continuous electrical signal that is then sampled, is the primary process by which noise appears in digital images. At every step in the process there are fluctuations caused by natural phenomena that add a random value to the exact brightness value for a given pixel. In typical images the noise can be modeled with either a gaussian ("normal"), uniform, or salt-and-pepper ("impulse") distribution. The shape of the distribution of these noise types as a function of gray level can be modeled as a histogram and can be seen in Figure 3.2-1. In Figure 3.2-1a we see the bell-shaped curve of the gaussian noise distribution, which can be analytically described by

$$\text{HISTOGRAM}_{\text{Gaussian}} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(g-m)^2/2\sigma^2}$$

where g = gray level

m = mean (average)

σ = standard deviation (σ^2 = variance)

About 70% of all the values fall within the range from one standard deviation (σ) below the mean (m) to one above, and about 95% fall within two standard deviations.

Umbaugh

Theoretically, this equation defines values from $-\infty$ to $+\infty$, but because the actual gray levels are only defined over a finite range, the number of pixels at the lower and upper values will be higher than this equation predicts. This is a result of the fact that all the noise values below the minimum will be clipped to the minimum, and those above the maximum will be clipped at the maximum value. This is a factor that must be considered with all theoretical noise models, when applied to a fixed, discrete range such as with digital images (e.g., 0 to 255). In Figure 3.2-1b is the uniform distribution:

$$\text{HISTOGRAM}_{\text{Uniform}} = \begin{cases} \dfrac{1}{b-a} & \text{for } a \le g \le b \\ 0 & \text{elsewhere} \end{cases}$$

$$\text{mean} = \frac{a+b}{2}$$

$$\text{variance} = \frac{(b-a)^2}{12}$$

With the uniform distribution, the gray-level values of the noise are evenly distributed across a specific range, which may be the entire range (0 to 255 for 8-bits), or a smaller portion of the entire range. In Figure 3.2-1c is the salt-and-pepper distribution:

$$\text{HISTOGRAM}_{\text{Salt \& Pepper}} = \begin{cases} A & \text{for } g = a \text{ (“pepper”)} \\ B & \text{for } g = b \text{ (“salt”)} \end{cases}$$

In the salt-and-pepper noise model there are only two possible values, a and b, and the probability of each is typically less than 0.1—with numbers greater than this, the noise will dominate the image. For an 8-bit image, the typical value for pepper noise is 0 and for salt-noise, 255.

The gaussian model is most often used to model natural noise processes, such as those occurring from electronic noise in the image acquisition system. The salt-and-pepper type noise is typically caused by malfunctioning pixel elements in the camera sensors, faulty memory locations, or timing errors in the digitization process. Uniform noise is useful because it can be used to generate any other type of noise distribution and is often used to degrade images for the evaluation of image restoration algorithms because it provides the most unbiased or neutral noise model. In Figure 3.2-2, we see examples of these three types of noise, and how they appear in images. Visually, the gaussian and uniform noisy images appear similar, but the image with added salt-and-pepper noise is very distinctive.

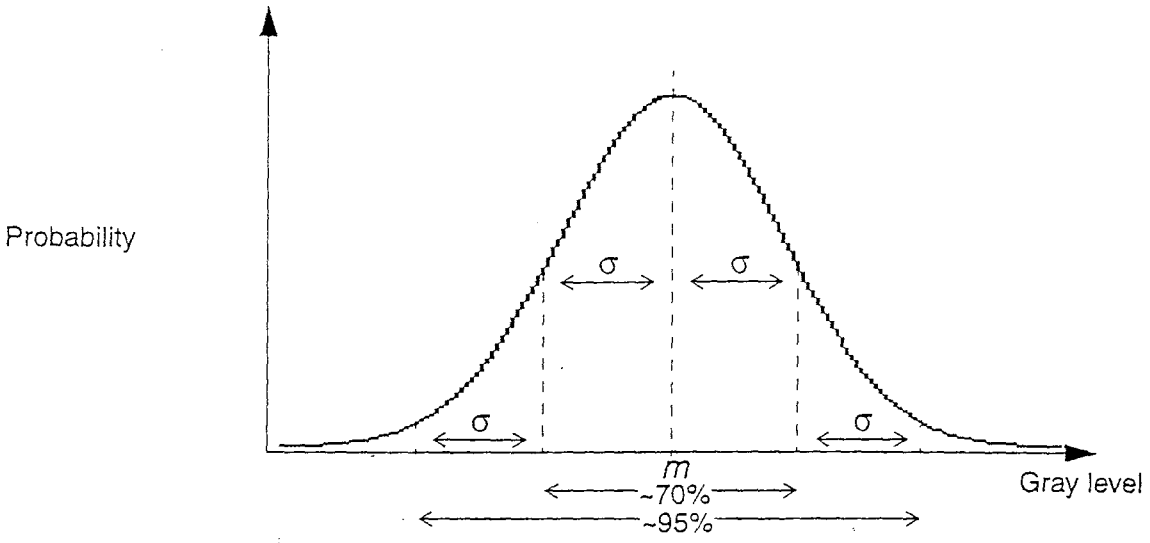
In addition to the gaussian, other noise models based on exponential distributions are useful for modeling noise in certain types of digital images. Radar range and velocity images typically contain noise that can be modeled by the Rayleigh distribution, defined by

$$\text{HISTOGRAM}_{\text{Rayleigh}} = \frac{2g}{\alpha} e^{-g^2/\alpha}$$

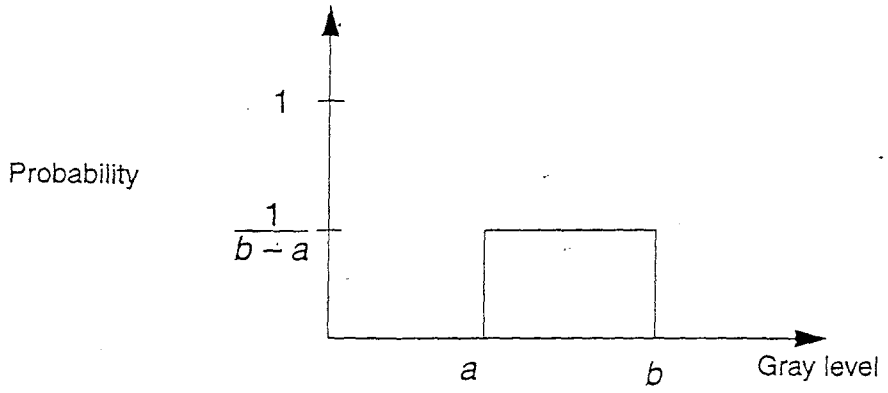
$$\text{where} \quad \text{mean} = \sqrt{\frac{\pi\alpha}{4}}$$

$$\text{variance} = \frac{\alpha(4-\pi)}{4}$$

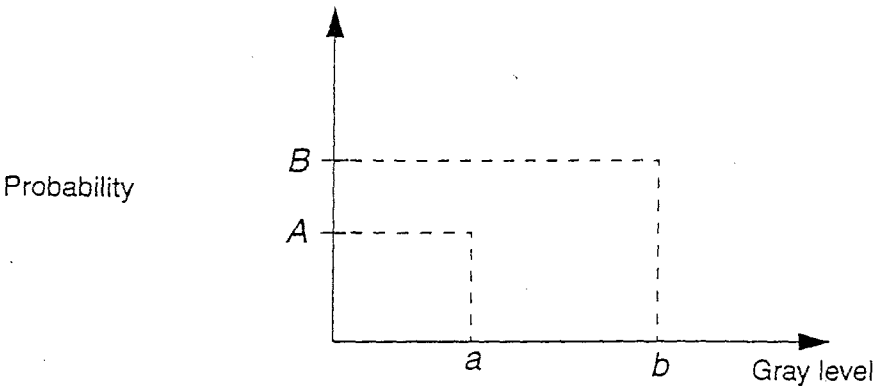
gure 3.2-1 Noise Distribution



a. Gaussian noise.

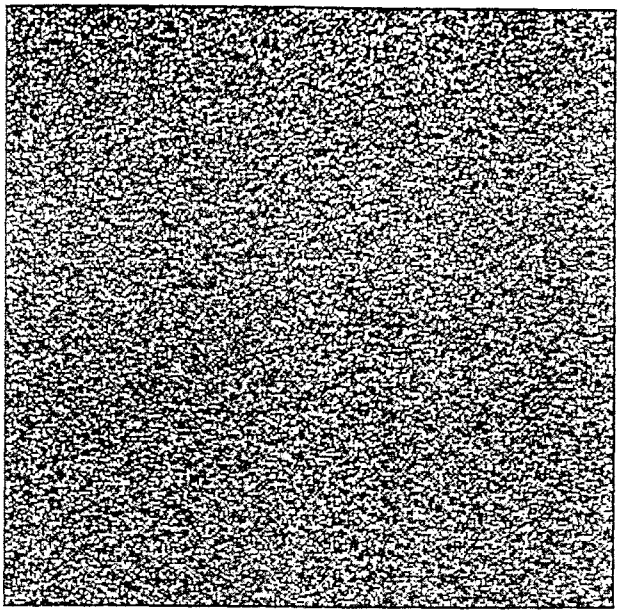


b. Uniform noise.

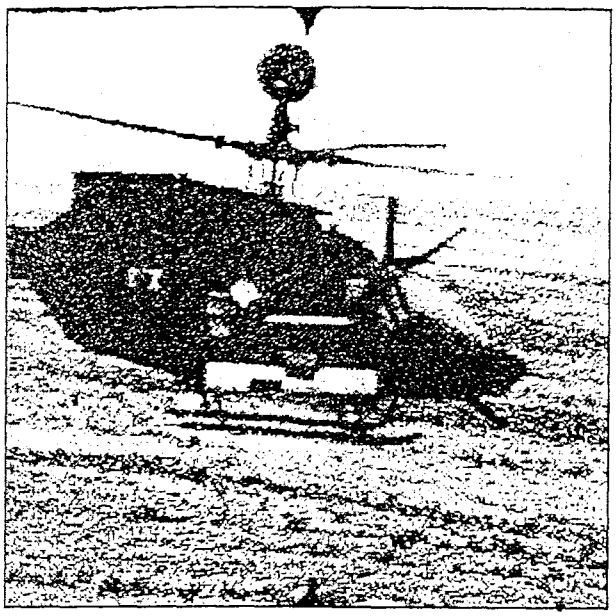


c. Salt-and-pepper noise.

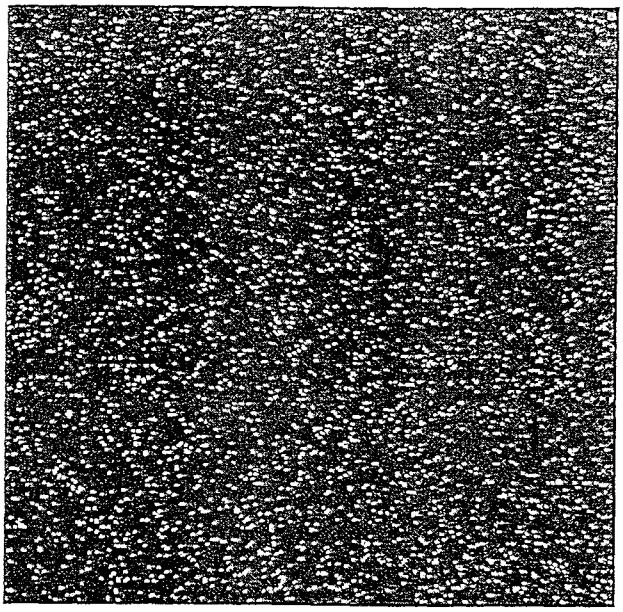
ıre 3.2-2 (Continued)



d. Uniform noise—$a = -47$; $b = +47$.



e. Original image with added uniform noise.



f. Salt-and-pepper noise—probability of salt = .05; probability of pepper = .05.

g. Original image with added salt-and-pepper noise.

The equation for gamma noise:

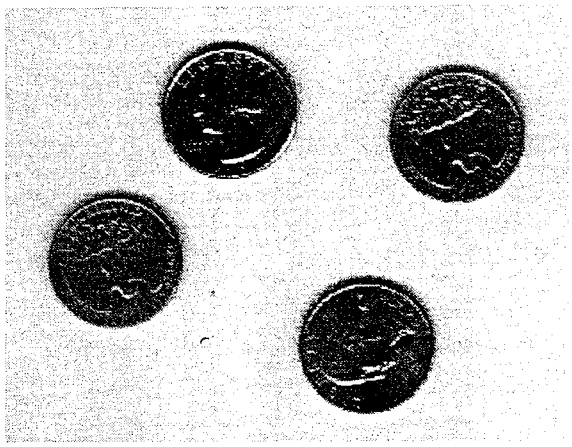
$$\text{HISTOGRAM}_{\text{Gamma}} = \frac{g^{\alpha-1}}{(\alpha-1)!a^{\alpha}} e^{-\frac{g}{a}}$$

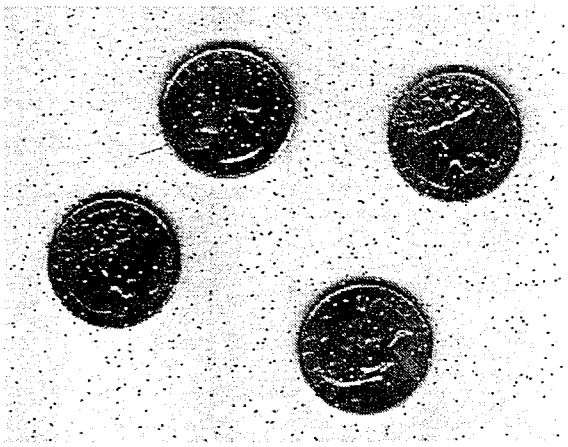
where variance $= a^2\alpha$

The histograms (distributions) for these can be seen in Figure 3.2-3. Note that negative exponential noise is actually gamma noise with the peak moved to the origin ($\alpha = 1$).

First, read in the image and add noise to it.

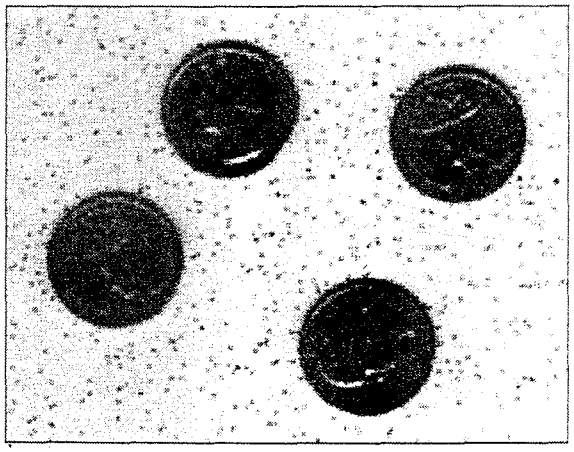
```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
imshow(I)
figure, imshow(J)
```

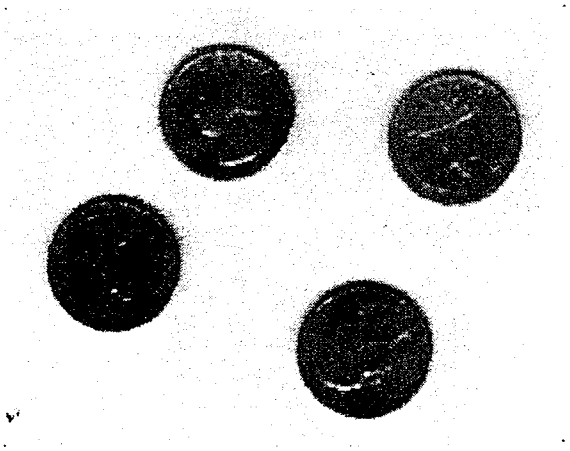




Now filter the noisy image and display the results. Notice that `medfilt2` does a better job of removing noise, with less blurring of edges.

```
K = filter2(fspecial('average',3),J)/255;
L = medfilt2(J,[3 3]);
imshow(K)
figure, imshow(L)
```





Median filtering is a specific case of *order-statistic filtering*. For information about order-statistic filtering, see the reference entry for the `ordfilt2` function in Chapter 11.

Definition: A function $f(x,y)$ is invariant if $f(x,y) = f(x-y)$

Definition: **Transfer function** H

Let $f(x,y)$ = picture

$h(x,y)$ = linear invariant operator

$H(u,v)$ = Fourier transform of $h(x,y)$

Then the output is $g = h * f$ and $G = H \bullet F$

Definition: **Impulse Response function:**

$h(x,y)$ is the response when the picture is a delta function

Definition: **Point Spread function**

Fourier transform of the Impulse Response function

Definition: **Finite Impulse Response Filter (FIR)**

Delta function $\rightarrow$ finite number of points

Definition: **Infinite Impulse Response Filter (IIR)**

Delta function $\rightarrow$ infinite number of points

Example: Recursive filter

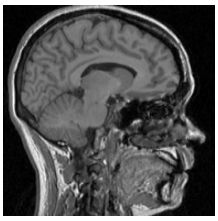
$$g'_n = (1-\alpha)g'_{n-1} + \alpha g_n$$

Definition: **Casual filter**

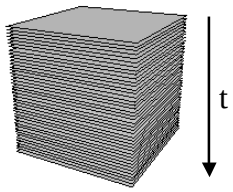
Depend only on previous data

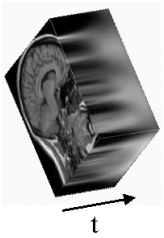
Dr. Yoram Tal

Image filtering



t = 0

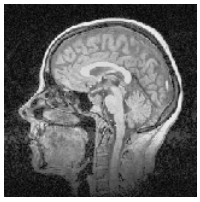




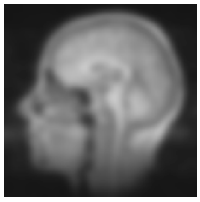
linear 'scale-space'
(Witkin 1983, Koenderink 1984)

Dr. Yoram Tal

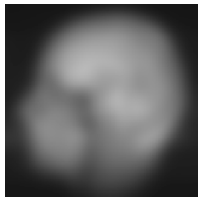
"Time" evolution



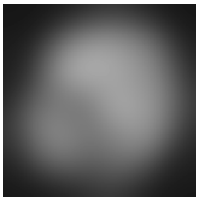
t = 0



t = 12.5



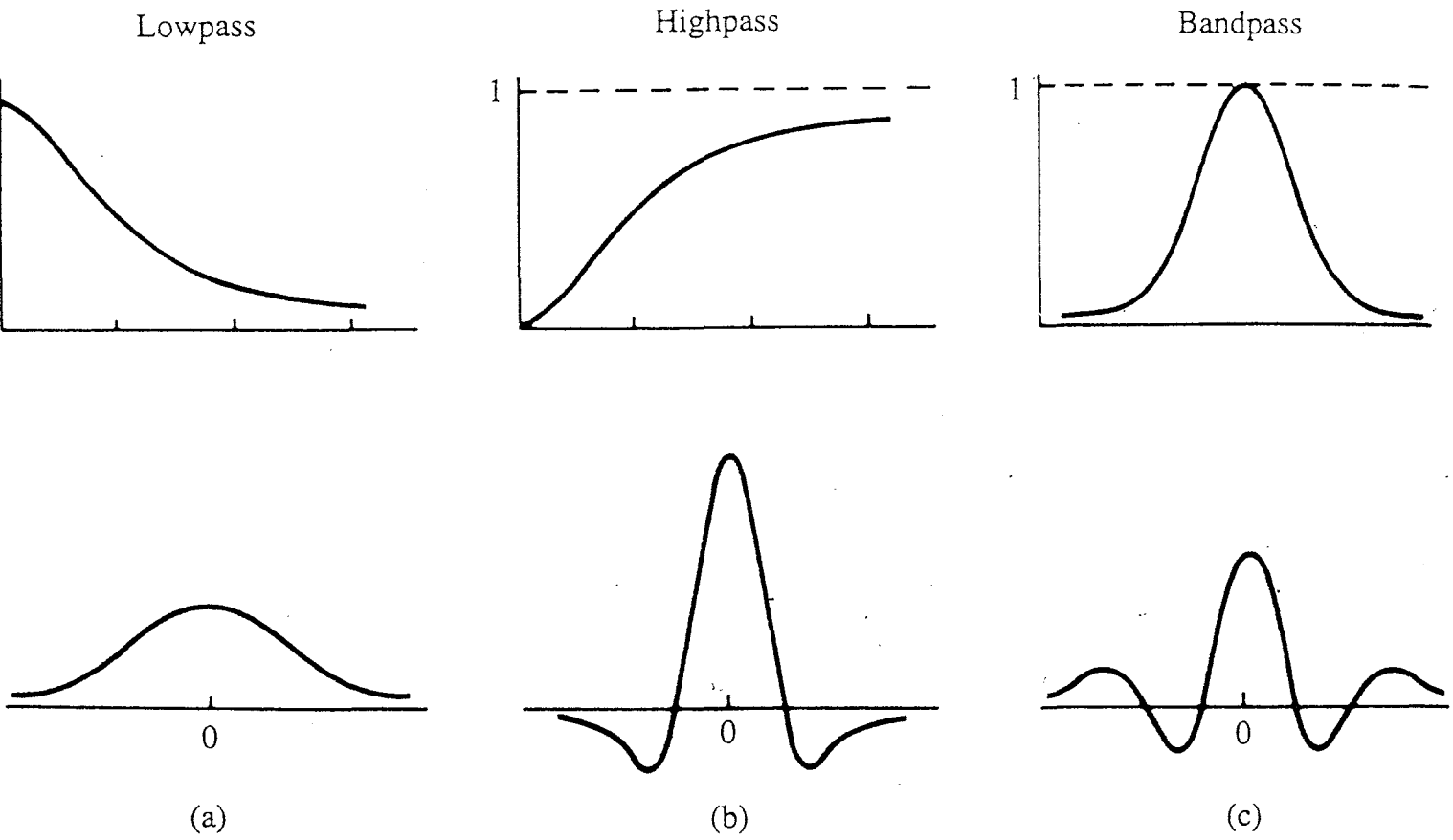
t = 50



t = 200

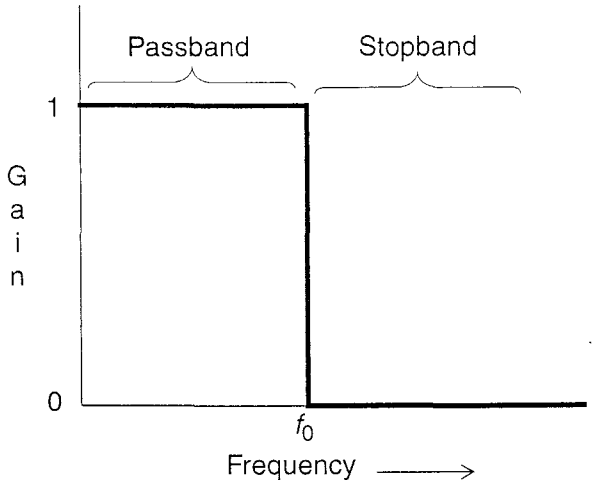
Time

Click



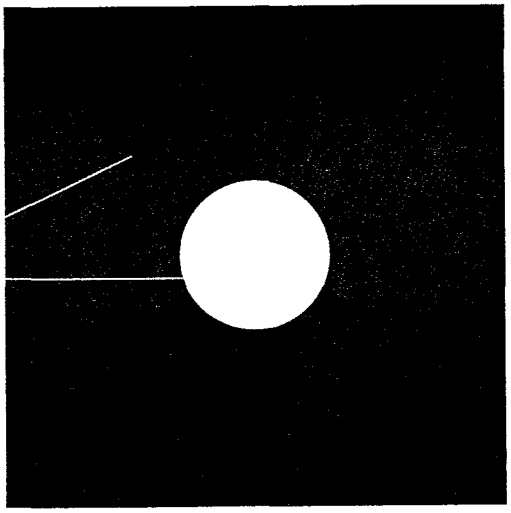
gure 4.19 *Top: cross sections of basic shapes for circularly symmetric frequency domain ers. Bottom: cross sections of corresponding spatial domain filters.*

Figure 2.5-18 Ideal Lowpass Filters

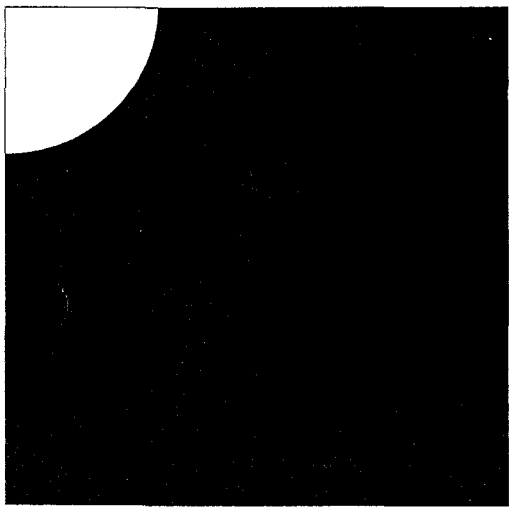


a. 1-D lowpass ideal filter.

black = 0
white = 1



b. 2-D lowpass ideal filter shown as an image.



c. 2-D lowpass ideal filter for Walsh-Hadamard and cosine functions.

Note that for ideal filters in Figure 2.5-18 the $H(u, v)$ matrix will contain only 1's and 0's, but, as in the preceding example, the matrix can contain any numbers.

The ideal filter is called ideal because the transition from the passband to the stopband in the filter is perfect—it goes from 0 to 1 instantly. Although this type of filter is not realizable in physical systems, such as with electronic filters, it is a reality for digital image processing applications because all we are doing is multiplying numbers in software. However, the ideal filter leaves undesirable artifacts in images. This artifact appears in the lowpass filtered image in Figure 2.5-17c as ripples, or waves, wherever there is a boundary in the image. This problem can be avoided by using a "nonideal" filter that does not have perfect transition, as is shown in Figure 2.5-19. The image created in Figure 2.5-17b was generated using a nonideal filter of a type called a Butterworth filter.

$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ is **not** a matrix

It corresponds to $Ux = x_{i+1,j+1} + 2x_{i+1,j} + x_{i+1,j-1} - (x_{i-1,j+11} + 2x_{i-1,j} + x_{i-1,j-1})$

Taking the Fourier transform

$$\begin{aligned} \widehat{Ux} &= e^{\frac{2\pi im}{M}} e^{\frac{2\pi in}{N}} + 2e^{\frac{2\pi im}{M}} + e^{\frac{2\pi im}{M}} e e^{-\frac{2\pi in}{N}} \\ &= 4i \sin(\frac{2\pi im}{M}) \left[1 + \cos(\frac{2\pi in}{N}) \right] \end{aligned}$$

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

1	1	1
1	1	1
1	1	1

(a)

$\frac{1}{25} \times$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(b)

$\frac{1}{49} \times$

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

(c)

.21 *Spatial lowpass filters of various sizes.*

Averaging

$$\widehat{x}_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i k n}{N}}$$

$$\widehat{y}_k = \sum_{n=0}^{N-1} x_{n+1} e^{-\frac{2\pi i k n}{N}} = \sum_{n=-1}^{N-2} x_{n+1} e^{-\frac{2\pi i k (n-1)}{N}} = e^{\frac{2\pi i k}{N}} \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i k n}{N}} = e^{\frac{2\pi i k}{N}} \widehat{x}_k$$

Therefore,

$$\frac{x_{k+1} \widehat{+} x_{k-1}}{2} = \frac{\widehat{x}_{k+1} + \widehat{x}_{k-1}}{2} = \frac{e^{\frac{2\pi i k}{N}} + e^{-\frac{2\pi i k}{N}}}{2} \widehat{x}_k = \cos\left(\frac{2\pi k}{N}\right) \widehat{x}_k = \sigma_k \widehat{x}_k$$

Example:
Highest frequency: $k = \frac{N}{2}$ then $\sigma_k = \cos(\pi) = -1$
half way : $k = \frac{N}{4}$ then $\sigma_k = \cos(\frac{\pi}{2}) = 0$
low frequency : $\frac{k}{N} \ll 1$ then $\sigma_k = \cos(\frac{2\pi k}{N}) \simeq 1 - \frac{1}{2}\left(\frac{2\pi k}{N}\right)^2$

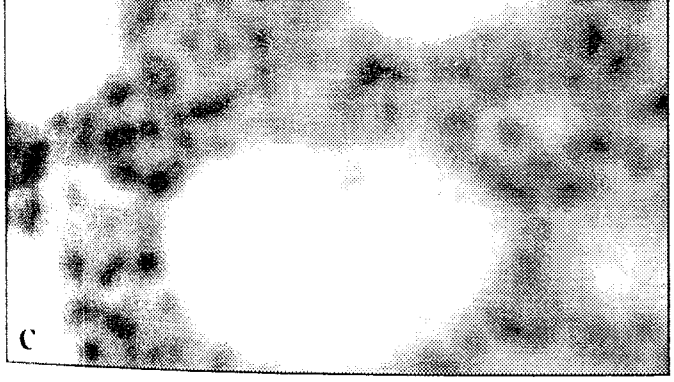
Other possibilities:

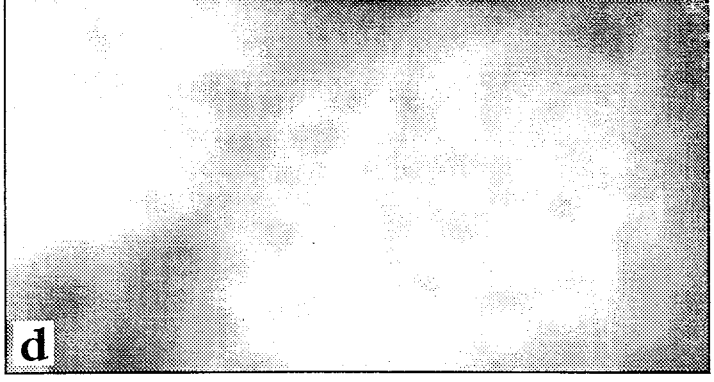
$$\frac{x_{k+1} + \widehat{x_k} + x_{k-1}}{3} = \frac{e^{\frac{2\pi i k}{N}} + 1 + e^{-\frac{2\pi i k}{N}}}{3} \widehat{x}_k = \frac{1 + 2\cos\left(\frac{2\pi k}{N}\right)}{3} \widehat{x}_k = \sigma_k \widehat{x}_k$$

Now
$\sigma_{\frac{N}{2}} = -\frac{1}{3}$
$\sigma_{\frac{N}{4}} = +\frac{1}{3}$
$\sigma_{\frac{k}{N} \ll 1} \simeq 1 - \frac{1}{3}\left(\frac{2\pi k}{N}\right)^2$

$$\frac{x_{k+1} + \widehat{2x_k} + x_{k-1}}{4} = \frac{e^{\frac{2\pi i k}{N}} + 2 + e^{-\frac{2\pi i k}{N}}}{4} \widehat{x}_k = \frac{1 + \cos\left(\frac{2\pi k}{N}\right)}{2} \widehat{x}_k = \sigma_k \widehat{x}_k$$

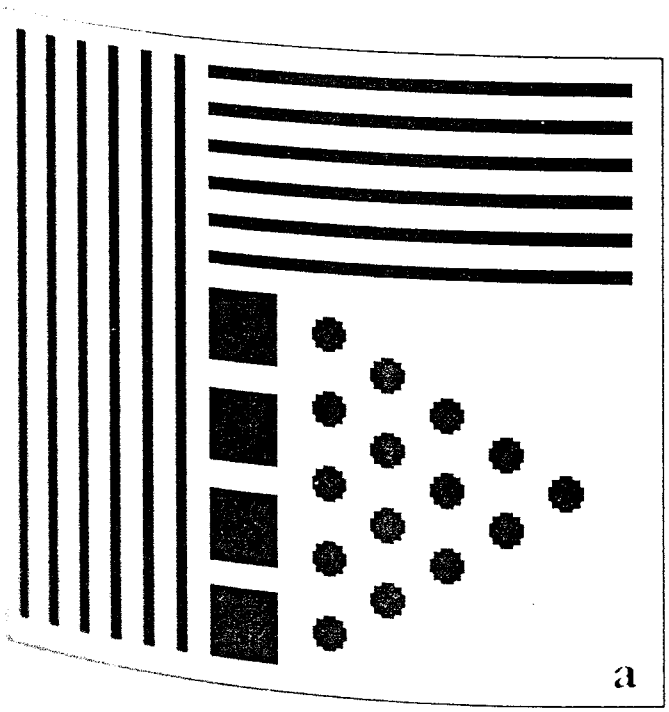
Now
$\sigma_{\frac{N}{2}} = 0$
$\sigma_{\frac{N}{4}} = +\frac{1}{2}$
$\sigma_{\frac{k}{N} \ll 1} \simeq 1 - \frac{1}{4}\left(\frac{2\pi k}{N}\right)^2$

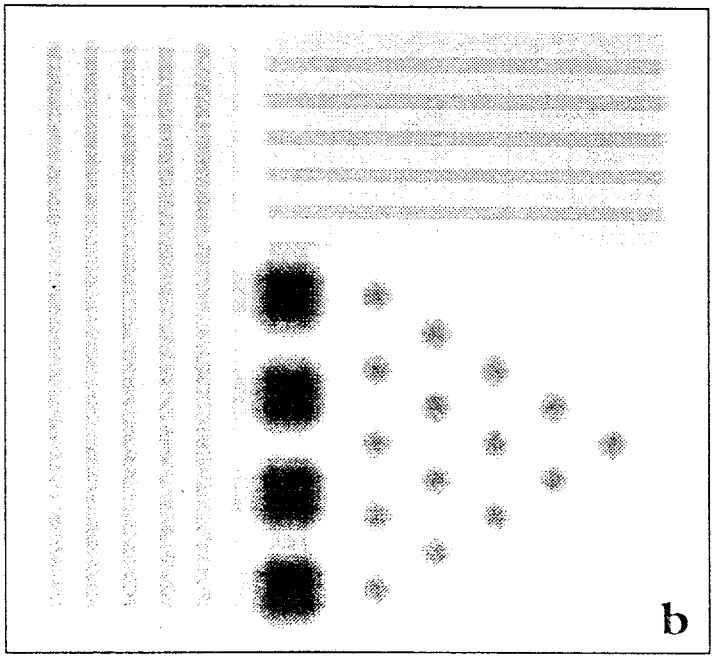




noise variations in uniform regions and the sharpness of boundaries between different structures. Applying smoothing with the 5×5 kernel corresponding to a Gaussian shape with a standard deviation of 0.625 pixels produces the improvement in quality shown in **Figure 5** (for two applications of the kernel). Using a larger kernel (the 9×9 kernel with standard deviation of 1.0 pixels, applied once) produces the result shown.

This type of averaging can reduce visible noise in the image, but it also blurs edges, displaces boundaries, and reduces contrast. It can even introduce an artefact often called "pseudo-resolution" when two nearby structures are averaged together in a way that creates an apparent feature between them. **Figure 6** shows an example in which the lines of the test pattern are blurred by the 11×11 averaging window causing false lines to appear between them.





Gibbs Phenomena

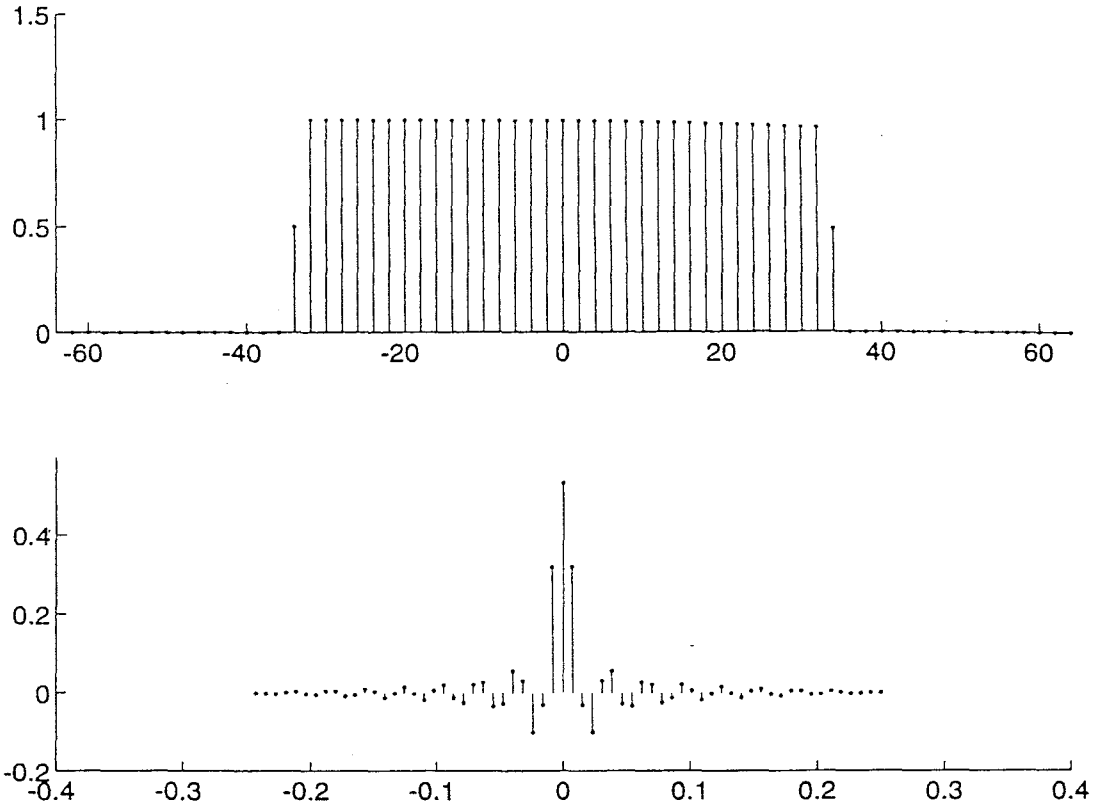
Given a discontinuity (רציף למקוטעין) the Fourier series converges but not uniformly

Thus, there is a constant overshoot of about 8% but the width narrows as we take more terms in the series

http://cnx.org/content/m10092/latest/

http://gaussianwaves.blogspot.com/2010/04/gibbs-phenomena-demonstration.html

- Affects filters
- Affects Fourier transform of pictures since the edges are not periodic



. 7.9. *The amplitude of the idealized low-pass filter in the frequency domain (top) and time domain (bottom) are shown in this figure. Observe that in the frequency domain the r values at the cut-off frequencies are* $1/2$*, satisfying the AVED condition.*

raightforward fashion. It is given by

$$H_k = \begin{cases} 1 & \text{if} \quad |k| < k_c, \\ \frac{1}{2} & \text{if} \quad |k| = k_c, \\ 0 & \text{if} \quad k_c < |k| \leq \frac{N}{2}, \end{cases}$$

ere k_c is the index associated with the desired cut-off frequency. Note that the ues of $H_{\pm k_c} = \frac{1}{2}$ must be used since they are the average values at the discontinuity ED).

It is a direct calculation (problem 156 or a fact from *The Table of DFTs*) to show t the time domain representation of the discrete filter is given by

$$h_n = \frac{\sin\left(\frac{\pi n k_c}{N}\right) \sin\left(\frac{2\pi n}{N}\right)}{2\sin^2\left(\frac{\pi n}{N}\right)}.$$

ure 7.9 displays a low-pass filter in both the time and frequency domains. The r is generated using $N = 64$ and a time sample rate $\Delta t = 1/128$, so that the total gth of the filter is $T = N\Delta t = 0.5$ seconds and the frequency sample rate (by the

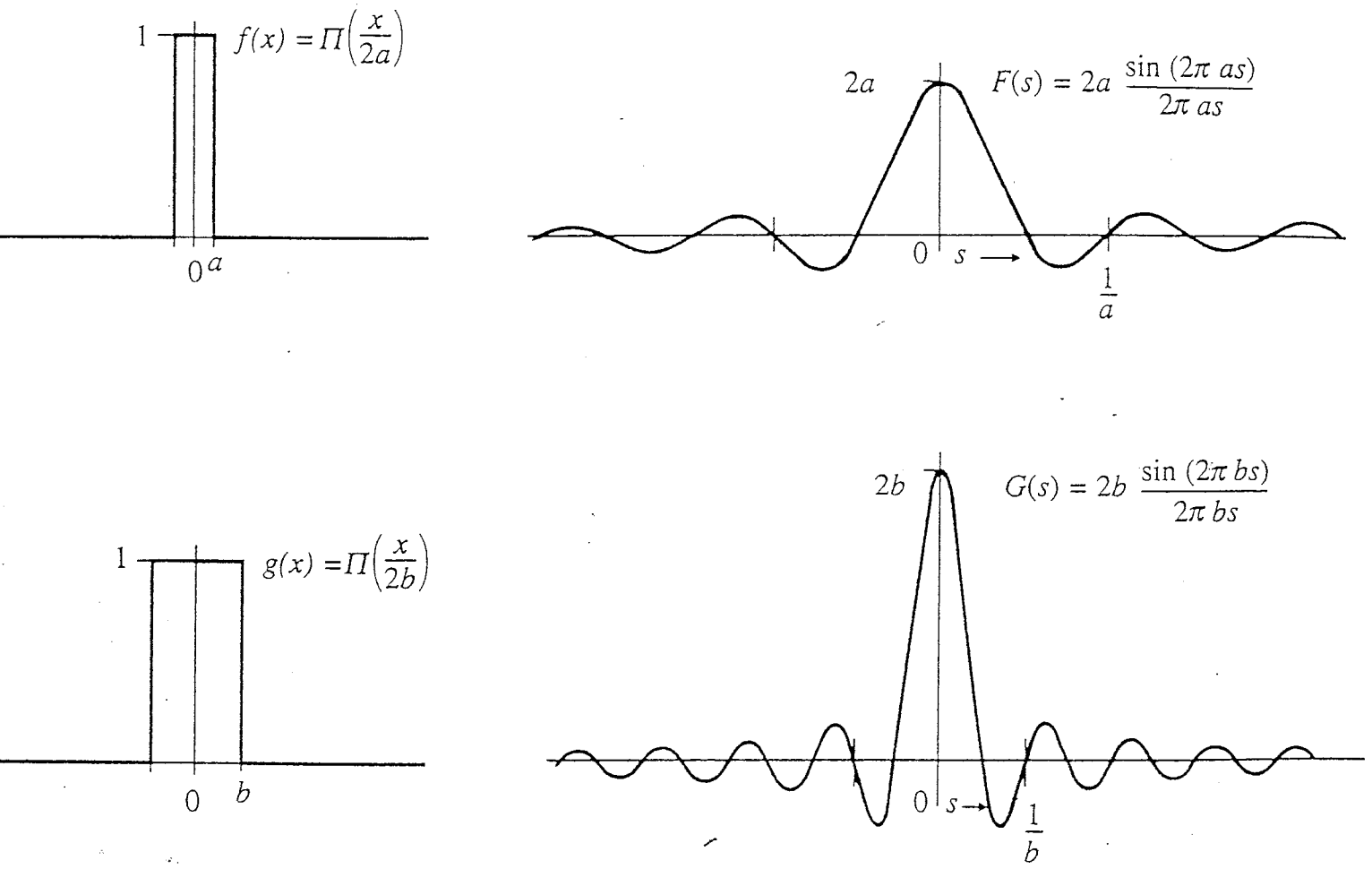


Figure 10–2 The similarity theorem

(a) Original

(b) noisy

(c) 3 × 3 filter

(d) 7 × 7 filter

Figure 7.18 Spatial averaging filters for smoothing images containing Ga noise.

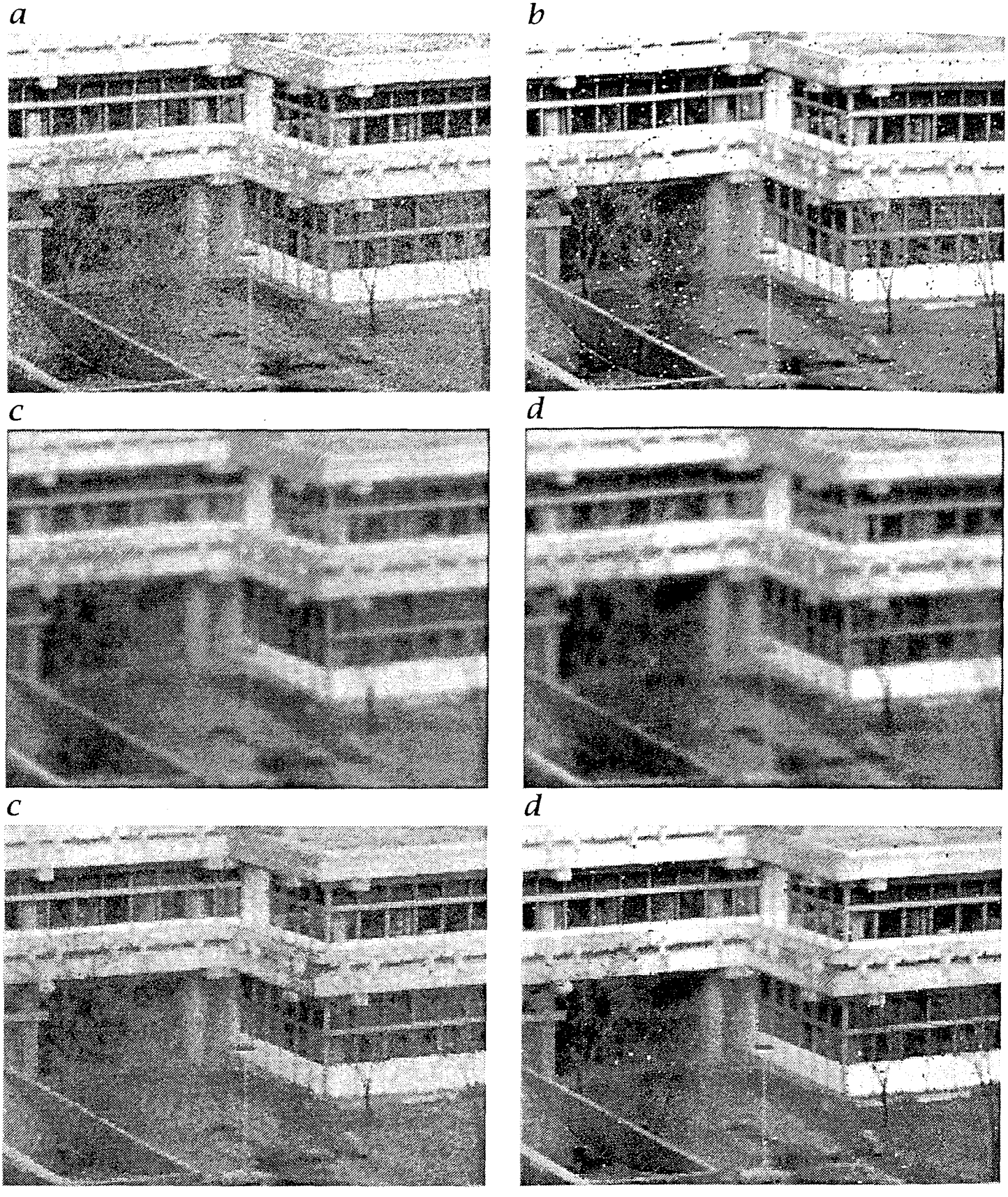


Figure 10.7: Suppression of noise with smoothing filters (exercise 10.2): a image from Fig. 10.6a with Gaussian noise; b image with binary noise; c image a and d image b filtered with a 9×9 binomial filter ($\mathcal{B}^8$); e image a and f image b filtered with a 3×3 median filter (Sect. 10.6.1).

Ideal low Pass Filter

$$H(U,V) = \begin{cases} 1 & \sqrt{U^2+V^2} < R^2 \\ 0 & \text{otherwise} \end{cases}$$

or

$$H(U,V) = \begin{cases} 1 & |U| < R_x \quad \text{and} \quad |V| < R_y \\ 0 & \text{otherwise} \end{cases}$$

Then

$$h(m,n) = R_x R_y sinc(2\pi U m) sinc(2\pi V n)$$

Butterworth low Pass Filter

$$H(U,v) = \frac{1}{1+(\frac{U^2+V^2}{R})^K}$$

Gaussian low Pass Filter

$$H(U,V) = e^{-2\pi^2\sigma^2(U^2+V^2)}$$

$$h(m,n) = \frac{1}{2\pi\sigma^2} e^{-\frac{m^2+n^2}{2\sigma^2}}$$

note: low frequencies centered at origin!

Butterworth filter

The transfer function of the Butterworth lowpass filter (BLPF) of order n and with cutoff frequency locus at a distance D_0 from the origin is defined by the relation

$D(u,v) = \sqrt{u^2+v^2}$ $\qquad H(u, v) = \dfrac{1}{1 + [D(u, v)/D_0]^{2n}}$ (4.4-4)

where $D(u, v)$ is given by Eq. (4.4-3). A perspective plot and cross section of the BLPF function are shown in Fig. 4.34.

note: $D(u,v) = D_0$
$\Rightarrow H(u,v) = 1/2$

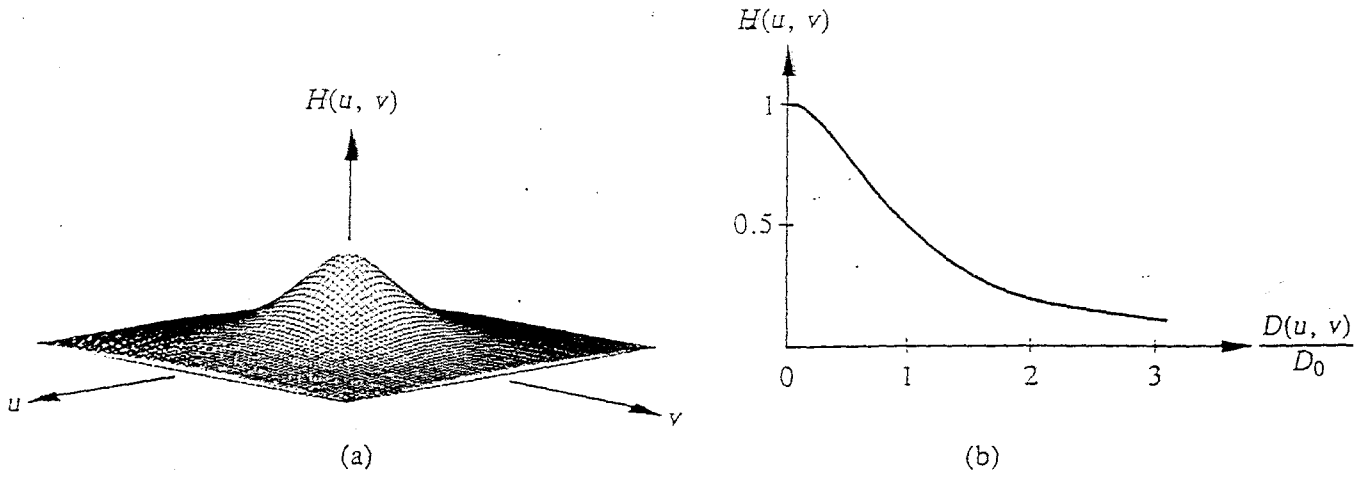


Figure 4.34 *(a) A Butterworth lowpass filter; (b) radial cross section for $n = 1$.*

$n \to \infty$ approaches "ideal" filter

Modification

$$H(u,v) = \frac{1}{1+\left[\sqrt{2}-1\right]\left[\frac{D(u,v)}{D_0}\right]^{2n}}$$

$$\simeq \frac{1}{+0.414\left[\frac{D(u,v)}{D_0}\right]^{2n}}$$

Figure 2.5-20 Lowpass Butterworth Filters

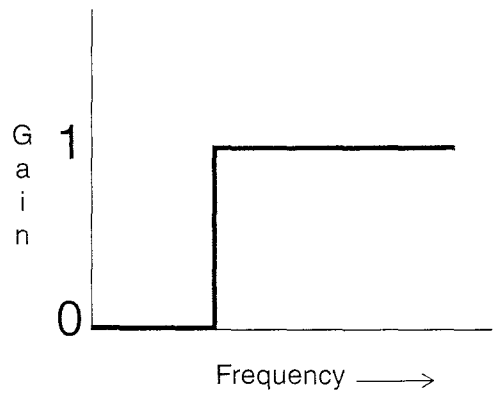
a. Filter order = 1.

b. Filter order = 3.

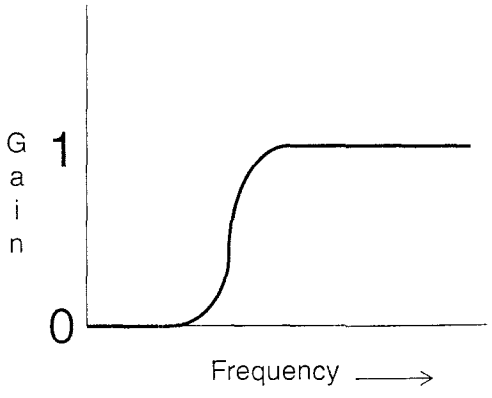
c. Filter order = 6.

d. Filter order = 8.

Figure 2.5-21 Highpass Filter Functions



a. 1-D ideal highpass filter.



b. 1-D nonideal highpass filter.

Homework:

Take picture: Try

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$\frac{1}{4}$

0	1	0
1	0	1
0	1	0

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

at border use
- (a) zero fill
- (b) periodic
- (c) extrapolation $X_{-1} = 2X_0 + X_1$

Try
- (a) FFT
- (b) multiply by ∇_n
- (c) IFFT

with 1) $\nabla_n = 1 + \cos\left(\frac{2\pi n}{N}\right)$

2) $\nabla_n = e^{-\alpha n/N}$ various α

but $\nabla_N = e^{-\alpha} << 1$

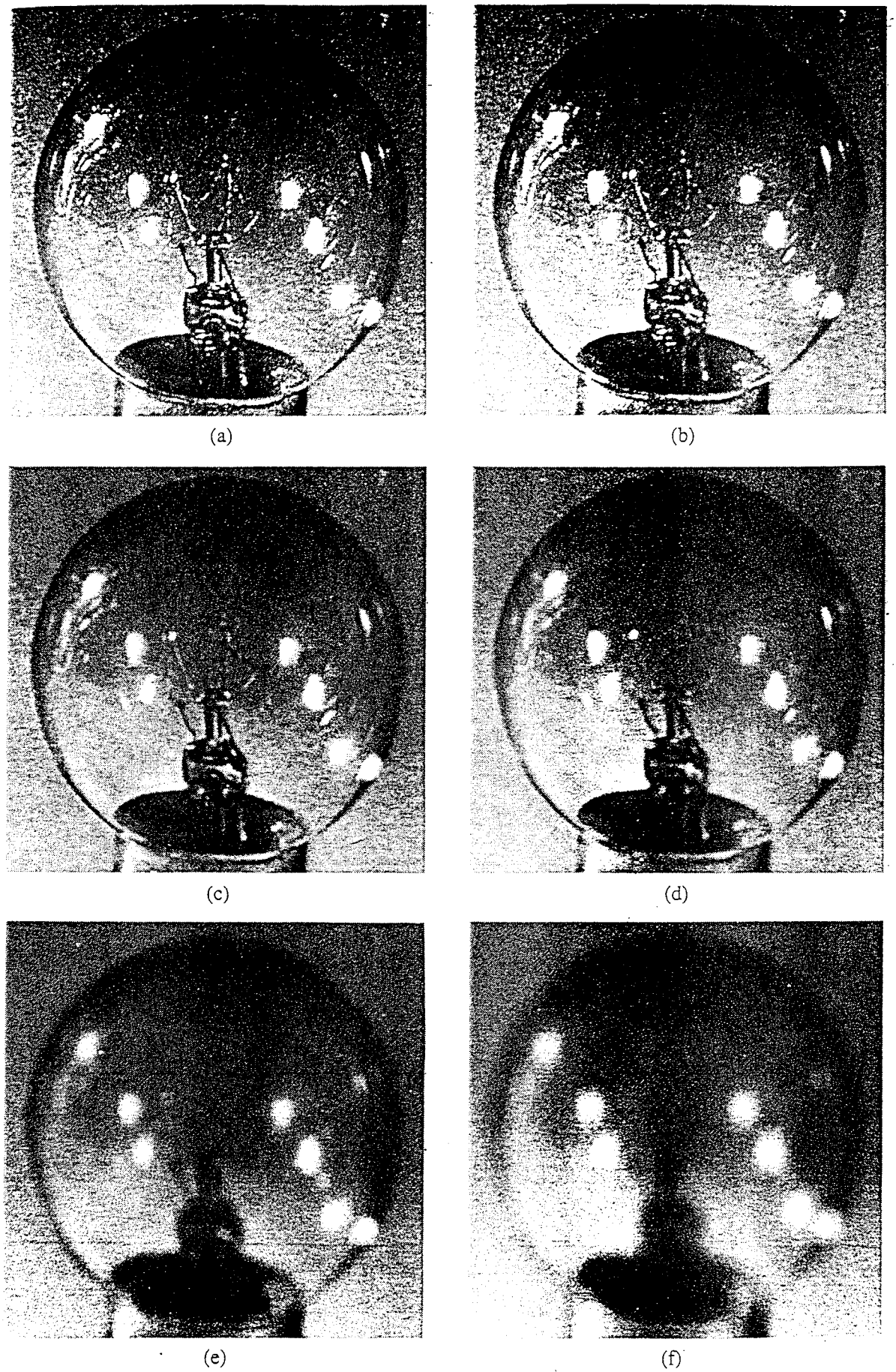
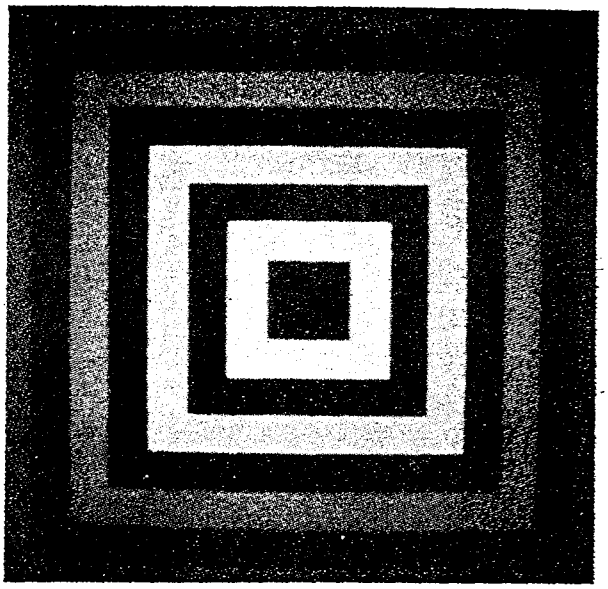
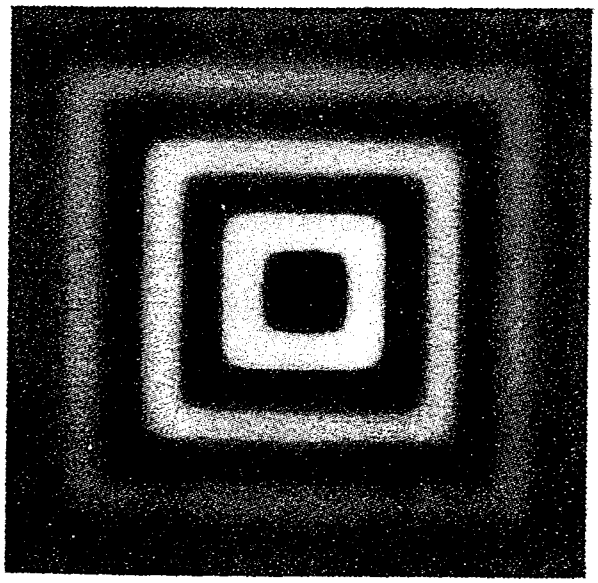


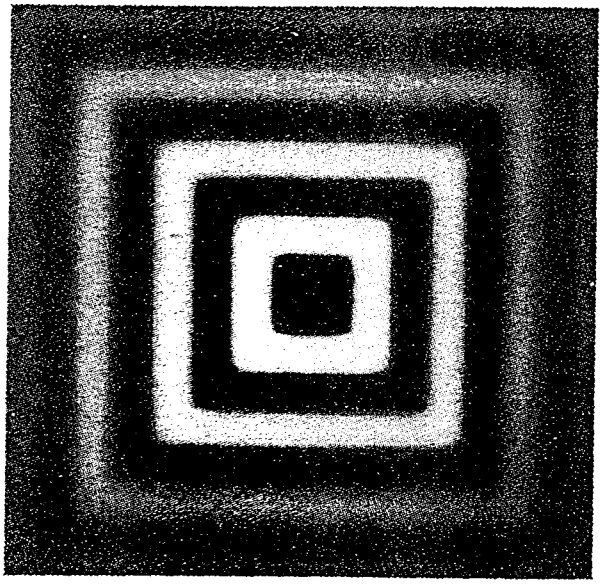
Figure 4.22 *(a) Original image; (b)–(f) results of spatial lowpass filtering with a mask of size $n \times n$, $n = 3, 5, 7, 15, 25$.*



(a)



(b)



(c)

Figure 4.43 *(a) Original image; (b) blurred image obtained with a Butterworth lowpass filter of order 1 in the frequency domain; (c) image blurred spatially by a 9 × 9 convolution mask obtained using Eq. (4.5-12). (From Meyer and Gonzalez [1983].)*

10.6.2 Weighted Averaging

In Sect. 3.1, we saw that gray values at pixels, just like any other experimental data, may be characterized by individual errors that have to be considered in any further processing. As an introduction, we first discuss the averaging of a set of N data g_n with standard deviations σ_n. From elementary statistics, it is known that appropriate averaging requires the weighting of each data point g_n with the inverse of the vari-

a local neighborhood. Results are acceptable if the noise is smaller in size than the smallest objects of interest in the image, but blurring of edges is a serious disadvantage. In the case of smoothing within a single image, one has to assume that there are no changes in the gray-levels of the underlying image data. This assumption is clearly violated at locations of image edges, and edge blurring is a direct consequence of violating the assumptions. Averaging is a special case of discrete convolution [equation (4.24)]. For a 3×3 neighborhood, the convolution mask h is

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.28}$$

The significance of the pixel in the center of the convolution mask h or its 4-neighbors is sometimes increased, as it better approximates the properties of noise with a Gaussian probability distribution (Gaussian noise, see Section 2.3.5).

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.29}$$

Larger convolution masks for averaging are created analogously according to the Gaussian distribution formula [equation (4.52)] and the mask coefficients are normalized to have a unit sum.

An example will illustrate the effect of this noise suppression. Images with low resolution (256×256) were chosen deliberately to show the discrete nature of the process. Figure 4.10a shows an original image of Prague castle with 256 brightnesses; Figure 4.10b shows the same image with superimposed additive noise with Gaussian distribution; Figure 4.10c shows the result of averaging with a 3×3 convolution mask (4.29)—noise is significantly reduced and the image is slightly blurred. Averaging with a larger mask (7×7) is demonstrated in Figure 4.10d, where the blurring is much more serious.

Alternative techniques, which are mostly non-linear, will now be discussed. These attempt not to blur sharp edges by avoiding averaging across edges.

Considering this basic fact, we can design better smoothing filters. e condition is that the filter masks should gradually approach zero. Here we will introduce a class of smoothing filters that meets this terion and can be calculated very efficiently. Furthermore these filters : an excellent example of how more complex filters can be built from ıple components. The simplest and most elementary smoothing mask can think of is

$$B = \frac{1}{2}[1\ 1]. \tag{10.15}$$

averages the gray values of two neighboring pixels. We can use this ısk p times in a row on the same image. This corresponds to the filter ısk

$$\frac{1}{2^p}\underbrace{[1\ 1] * [1\ 1] * \ldots * [1\ 1]}_{p \text{ times}}, \tag{10.16}$$

written as an operator equation,

$$\mathcal{B}^p = \underbrace{\mathcal{B}\mathcal{B}\ldots\mathcal{B}}_{p \text{ times}}. \tag{10.17}$$

me examples of the resulting filter masks are:

$$\begin{aligned} B^2 &= 1/4[1\ 2\ 1] \\ B^3 &= 1/8[1\ 3\ 3\ 1] \\ B^4 &= 1/16[1\ 4\ 6\ 4\ 1] \\ B^8 &= 1/256[1\ 8\ 28\ 56\ 70\ 56\ 28\ 8\ 1]. \end{aligned} \tag{10.18}$$

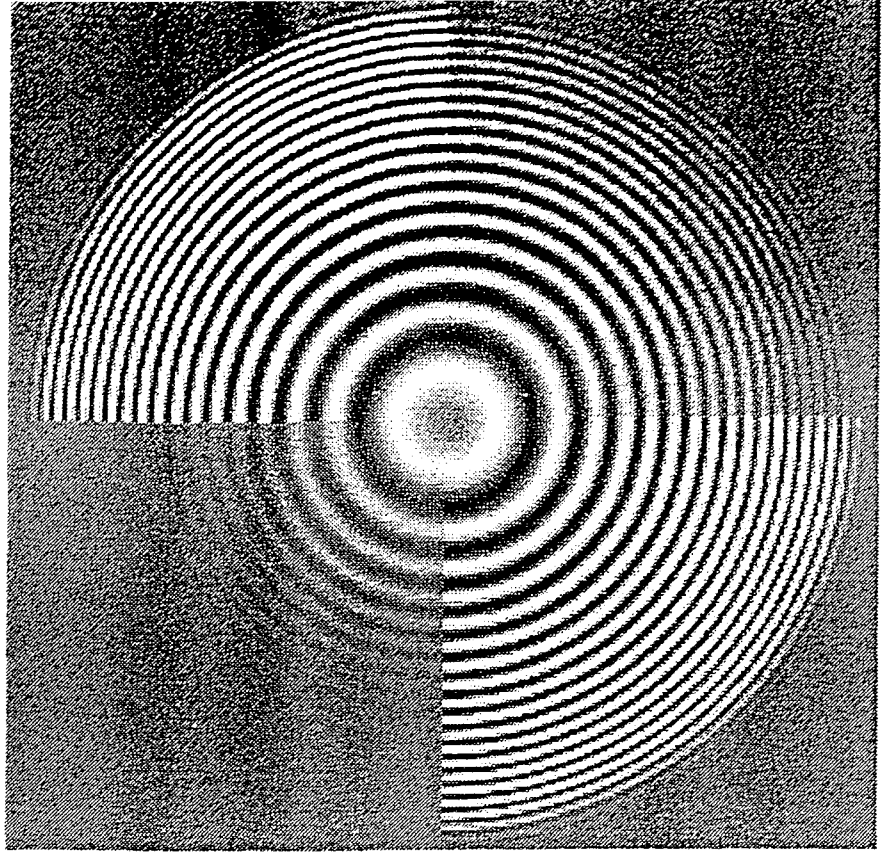


Figure 10.4: Test of the smoothing with a $\mathcal{B}^4$ and $\mathcal{B}^{16}$ binomial filter using a test image with concentric sinusoidal rings.

Because of symmetry, only the odd-sized filter masks are of interest. In order to perform a convolution with the asymmetric mask 1/2 [1 1] correctly, we store the result in the right and left pixel alternately.

The masks contain the values of the discrete *binomial distribution*. Actually, the iterative composition of the mask by consecutive convolution with the 1/2 [1 1] mask is equivalent to the computation scheme of *Pascal's triangle*:

p	f		σ^2
0	1	1	0
1	1/2	1 1	1/4
2	1/4	1 2 1	1/2
3	1/8	1 3 3 1	3/4
4	1/16	1 4 6 4 1	1
5	1/32	1 5 10 10 5 1	5/4
6	1/64	1 6 15 20 15 6 1	3/2
7	1/128	1 7 21 35 35 21 7 1	7/4
8	1/256	1 8 28 56 70 56 28 8 1	2

(10.19)

where p denotes the order of the binomial, f the scaling factor 2^{-p}, and σ^2 the variance, i. e., effective width, of the mask.

The problem of slow large-scale averaging originates from the small distance between the pixels averaged in the elementary $B = 1/2\,[1\ 1]$ mask. In order to overcome this problem, we may use the same elementary averaging process but with more distant pixels and increase the standard deviation for smoothing correspondingly. In two dimensions, the following masks could be applied along diagonals ($\sigma \cdot \sqrt{2}$):

$$B_{x+y} = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B_{x-y} = \frac{1}{4}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{10.27}$$

or, with double step width along axes ($\sigma \cdot 2$),

$$B_{2x} = \frac{1}{4}[1\ 0\ 2\ 0\ 1], \quad B_{2y} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}. \tag{10.28}$$

In three dimensions, the multistep masks with double step width along the axes are:

$$B_{2x} = \frac{1}{4}[1\ 0\ 2\ 0\ 1], \quad B_{2y} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}, \tag{10.29}$$

$$B_{2z} = \frac{1}{4}[[1],[0],[2],[0],[1]].$$

The subscripts in these masks denote the number of steps along the coordinate axes between two pixels to be averaged. B_{x+y} averages the gray values at two neighboring pixels in the direction of the main diagonal. B_{2x} computes the mean of two pixels at a distance of 2 in the x direction. The standard deviation of these filters is proportional to the distance between the pixels. The most efficient implementations are multistep masks along the axes. They have the additional advantage that because of separability, the algorithms can be applied to image data of arbitrary dimensions.

The problem with these filters is that they perform a subsampling. Consequently, they are no longer filters for larger wave numbers. If we take, for example, the symmetric 2-D $\mathcal{B}_{2x}^2\mathcal{B}_{2y}^2$ filter, we effectively work on a grid with a doubled grid constant in the spatial domain. Hence, the

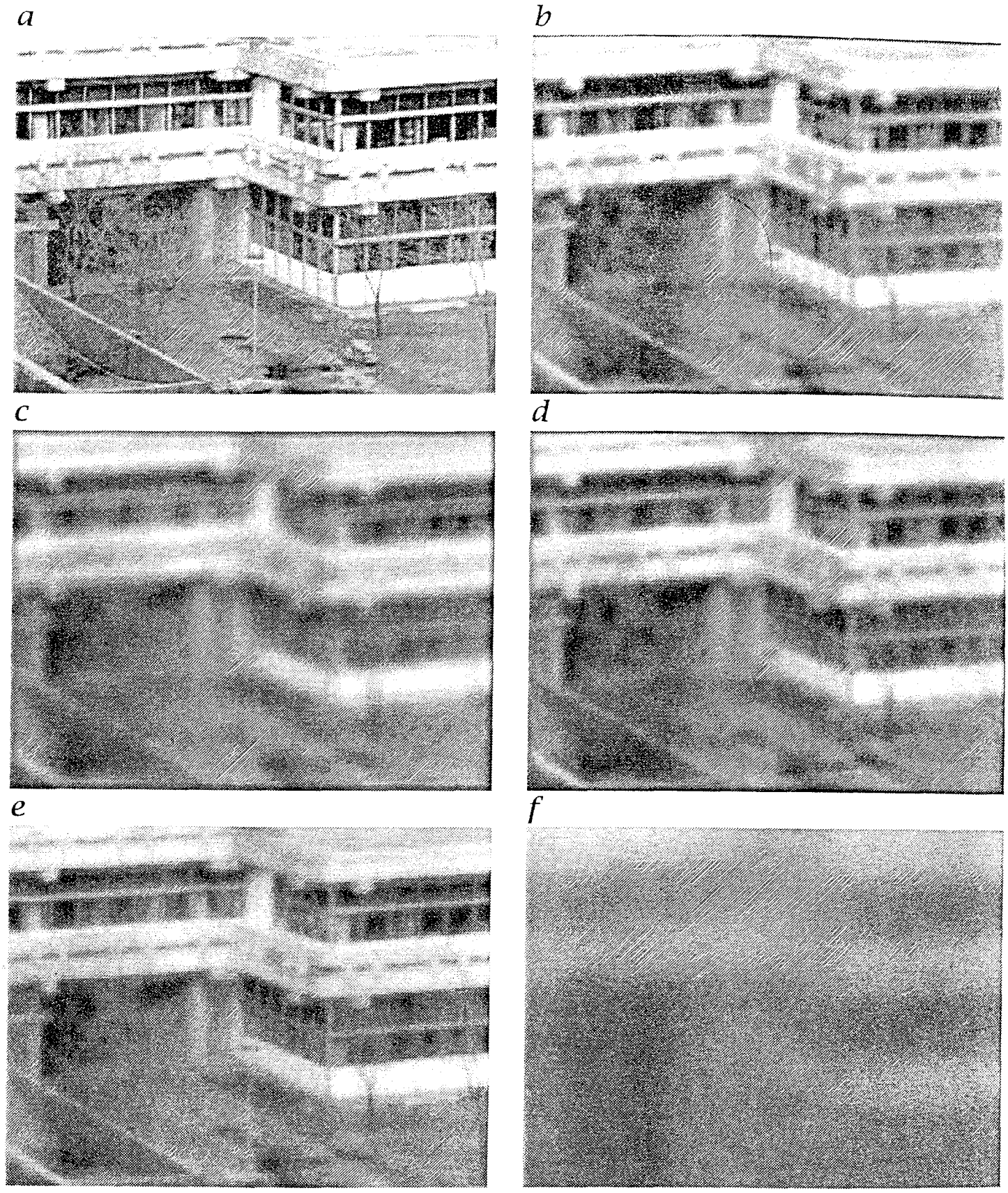


*Figure 10.6: Application of smoothing filters (**exercise 10.1**): **a** original image; **b** 5×5 box filter; **c** 9×9 box filter; **d** 17×17 binomial filter ($\mathcal{B}^{16}$); a set of recursive filters (10.37) running in horizontal and vertical directions; **e** $p = 2$; **f** $p = 16$.*

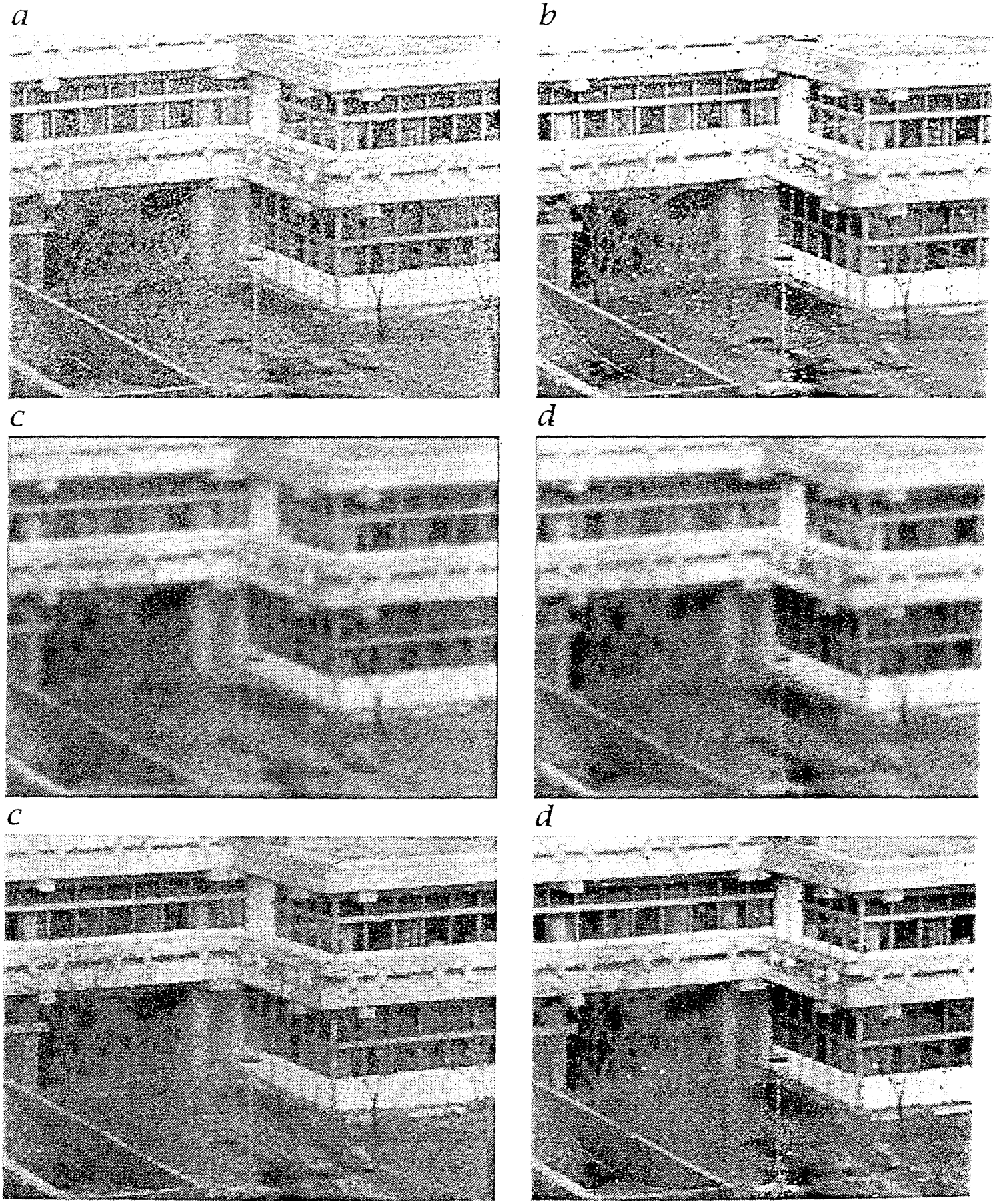
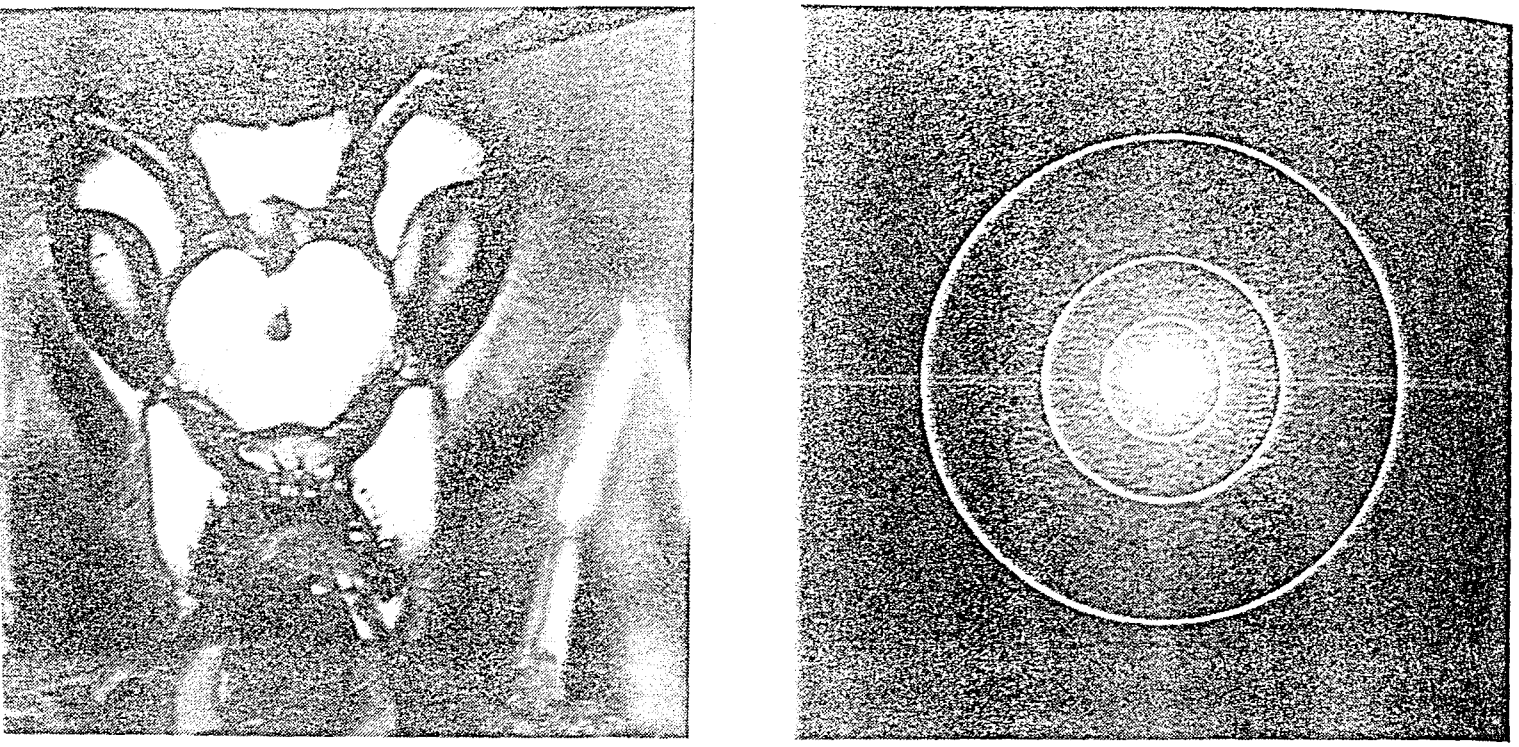


Figure 10.7: Suppression of noise with smoothing filters (exercise 10.2): a im from Fig. 10.6a with Gaussian noise; b image with binary noise; c image a a image b filtered with a 9 × 9 binomial filter ($\mathcal{B}^8$); e image a and f image b filt with a 3 × 3 median filter (Sect. 10.6.1).



(a) (b)

ure 4.31 *(a) 512 × 512 image and (b) its Fourier spectrum. The superimposed circles, ch have radii equal to 8, 18, 43, 78, and 152, enclose 90, 93, 95, 99, and 99.5 percent of image power, respectively.*

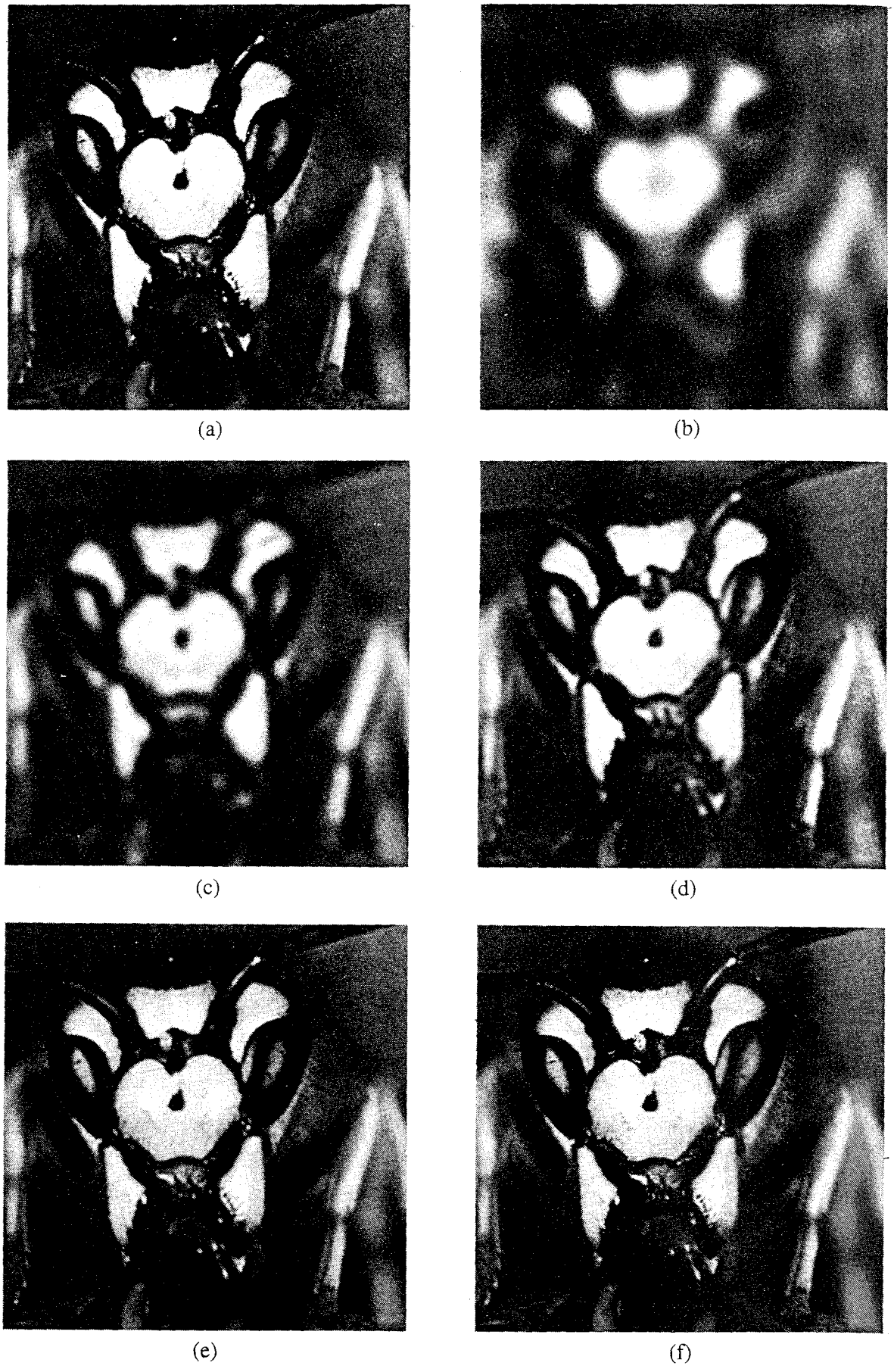


Figure 4.32 *(a) Original image; (b)–(f) results of ideal lowpass filtering with the cu frequency set at the radii shown in Fig. 4.31(b).*

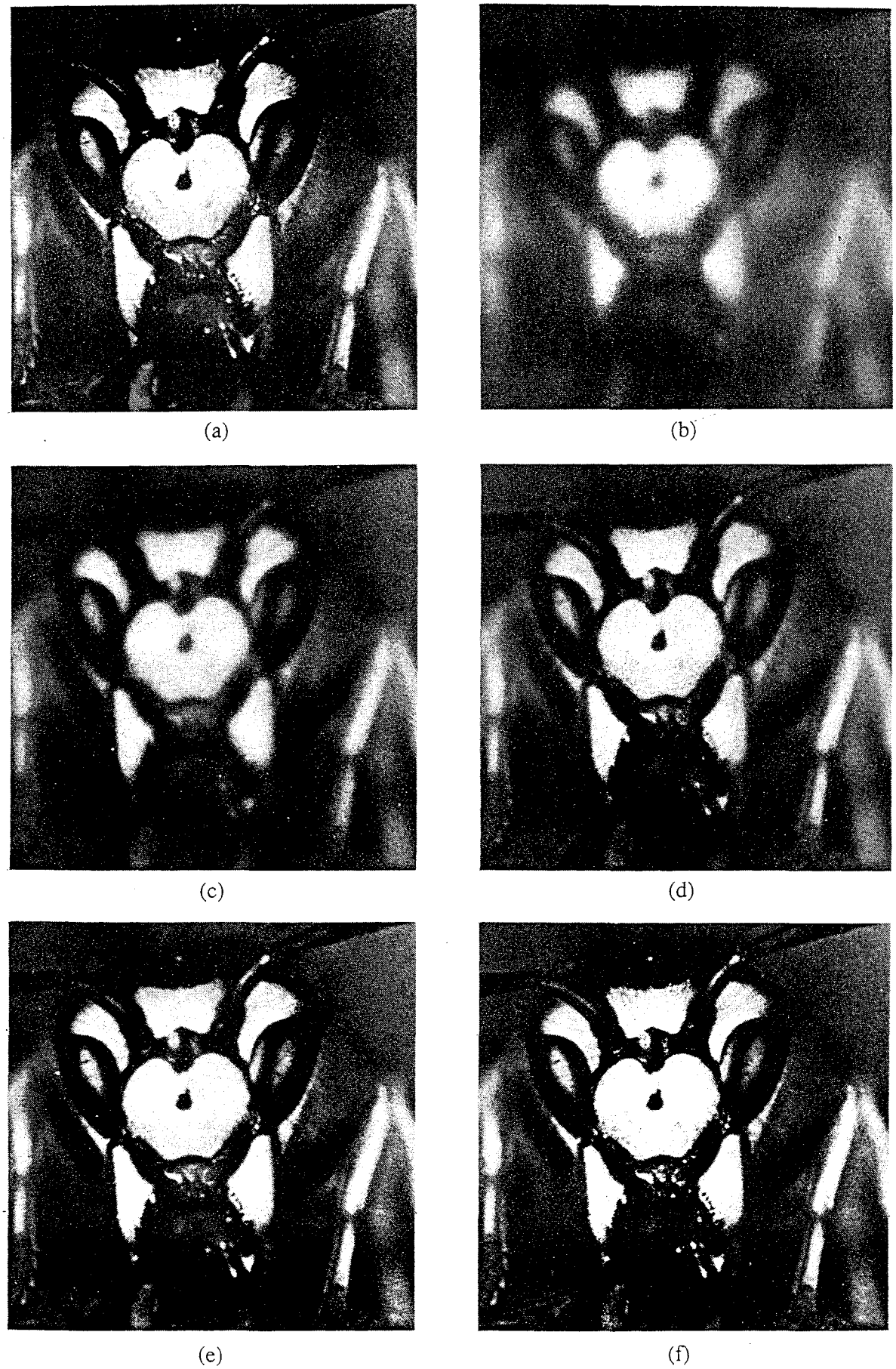


Figure 4.35 *(a) Original image; (b)–(f) results of Butterworth lowpass filtering with the cutoff point set at the radii shown in Fig. 4.31(b).*

Inhomogeneous filtering

In image processing we identify the concentration with the gray value at a certain location.

If the diffusion tensor is constant over the whole image domain, one speaks of *homogeneous diffusion*.

If the diffusion tensor is space-dependent, it is called an *inhomogeneous diffusion*.

Inhomogeneous filtering

The idea is to define a *diffusivity* function

$$D = g(|\nabla f|)$$

Where $|\nabla f|$ is a *fuzzy* edge detector:

$$g(|\nabla f|) = \frac{1}{\sqrt{1+|\nabla f|^2/\lambda^2}}$$

or $$g(|\nabla f|) = \exp\left(-|\nabla f|^2/\lambda^2\right)$$

Inhomogeneous filtering (2)

The resulting equation is no longer *homogeneous* (though it may still be *linear*):

$$\partial_t u = div\left(g(|\nabla f|) \cdot \nabla u\right)$$

See:

B. ter Haar Romeny, L. Florack, J. Koenderink, M. Viergever (Eds.), Scale-Space Theory in Computer Vision, Lecture Notes in Computer Science, Vol. 1252, Springer, Berlin, pp. 3–28, 1997. Invited paper.

A Review of Nonlinear Diffusion Filtering

Joachim Weickert*

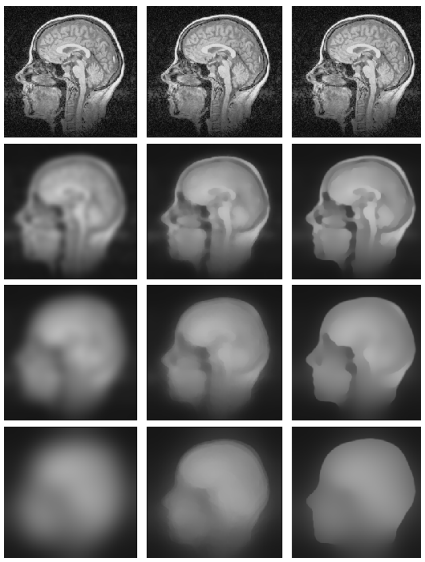
Image Sciences Institute,
Utrecht University Hospital,
E.01.334, Heidelberglaan 100,
3584 CX Utrecht, The Netherlands.
E-mail: Joachim.Weickert@cv.ruu.nl

Inhomogeneous filtering (3)

First column:
homogeneous

Second column:
inhomogeneous & linear

Third column:
inhomogeneous & non-linear



Adaptive filtering

Click

See also Wiener filter (wiener2)

Image Enhancement & Restoration

Enhancement: Speckle Noise

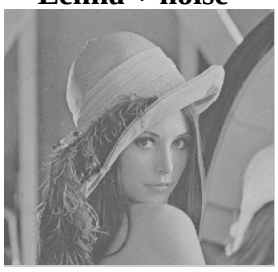
Lenna

Y

```
S = sprandn(512,512,0.01)*50;
Y = Lenna + S;
imagesc(Y); colormap gray
```


Solution 1: Gaussian smoothing

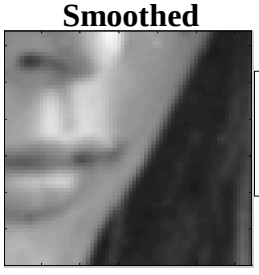
Lenna + noise



Smoothed

Lenna

Smoothed



Removes the noise - but smears the image

Solution 2: Median filtering

Lenna + noise

Filtered

Lenna

Filtered

Removes the noise - but looses image details

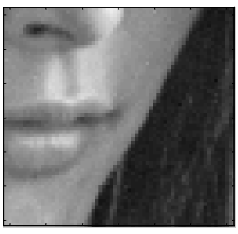
Solution 3: Selective filtering

Y

```
S = sprandn(512,512,0.01)*50;
Y = Lenna + S;
h = fspecial('gauss',7,1);
Ys = filter2(h,Y, same );
```

```
k = find(abs(Y-Ys)>20);
Z = Y;
Z(k) = Ys(k);
```

Z



Selective filtering - ROI

ROIFILT2 apply filter2 to a polygonal area of the image

```
y = roifilt2(ones(9),x,bw);
```

x → **bw** → **y**

Image Deblurring

Examples
(see MATLAB's help)

```
I = imread('flowers.tif');
I = I(10+[1:256],222+[1:256],:); % crop the image
figure;imshow(I);title('Original Image');

LEN = 31;
THETA = 11;
PSF = fspecial('motion',LEN,THETA);
Blurred = imfilter(I,PSF,'circular','conv');
figure; imshow(Blurred);title('Blurred Image');
```

Original image

Blurred image

Deblurring with Wiener

```
wnr1 = deconvwnr(Blurred,PSF);
figure;
imshow(wnr1);
title('Restored, True PSF');
```

Original image

Blurred image

Image restored by Wiener filter

High Pass Filters

Fourier

Ideal

$$H(u,v) = \begin{cases} 0 & \sqrt{u^2+v^2} \leq R^2 \\ 1 & \sqrt{u^2+v^2} \geq R^2 \end{cases}$$

or

$$H(u,v) = \begin{cases} 0 & |u| \leq u_0 \text{ and } |v| \leq v_0 \\ 1 & \text{otherwise} \end{cases}$$

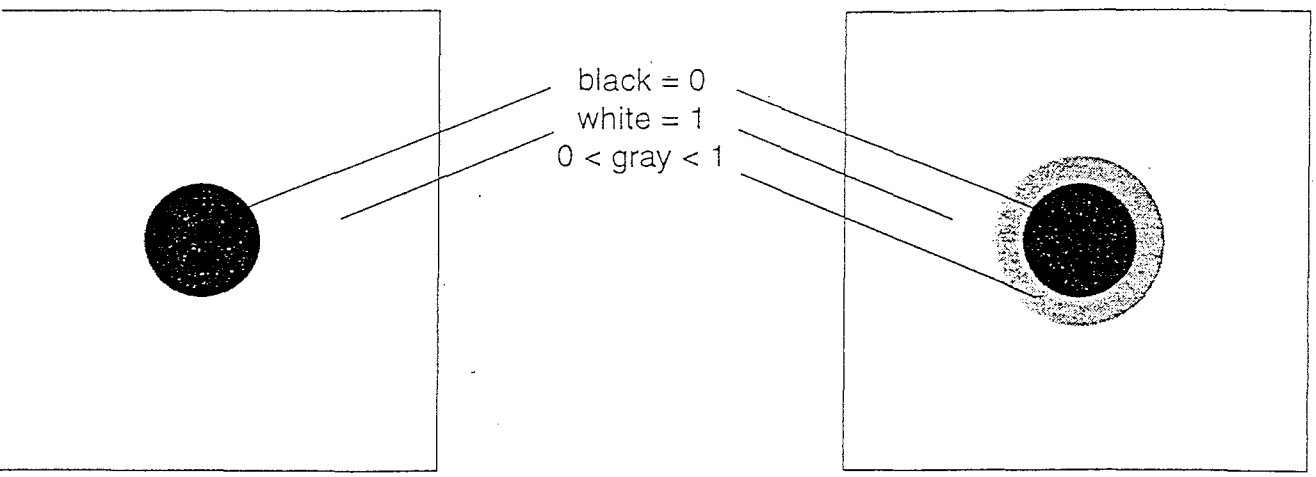
Butterworth

$$H(u,v) = \frac{1}{1 + \left(\frac{R^2}{\sqrt{u^2+v^2}}\right)^{2K}}$$

or

$$H(u,v) = \frac{1}{1 + (\sqrt{(2)} - 1) \cdot \left(\frac{R^2}{\sqrt{u^2+v^2}}\right)^{2K}}$$

(Continued)



2-D ideal highpass filter shown as an image.

d. 2-D nonideal highpass filter shown as an image.

Butterworth filter

The transfer function of the Butterworth highpass filter (BHPF) of order n and with cutoff frequency locus at a distance D_0 from the origin is defined by the relation

$$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}} \tag{4.4-7}$$

where $D(u, v)$ is given by Eq. (4.4-3). Figure 4.38 shows a perspective plot and cross section of the BHPF function.

Note that when $D(u, v) = D_0$, $H(u, v)$ is down to ½ of its maximum value. As in the case of the Butterworth lowpass filter, common practice is to select the cutoff frequency locus at points for which $H(u, v)$ is down to $1/\sqrt{2}$ of its maximum value. Equation (4.4-7) is easily modified to satisfy this constraint by using the following scaling:

$$H(u, v) = \frac{1}{1 + [\sqrt{2} - 1][D_0/D(u, v)]^{2n}} \tag{4.4-8}$$

$$= \frac{1}{1 + 0.414[D_0/D(u, v)]^{2n}}.$$

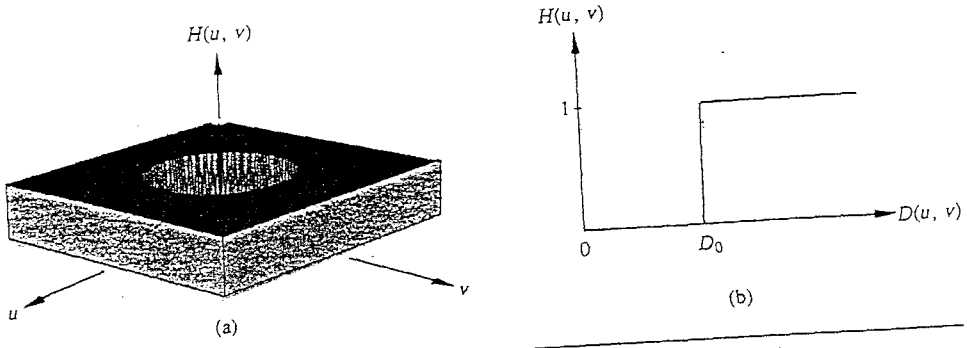


Figure 4.37 *Perspective plot and radial cross section of ideal highpass filter.*

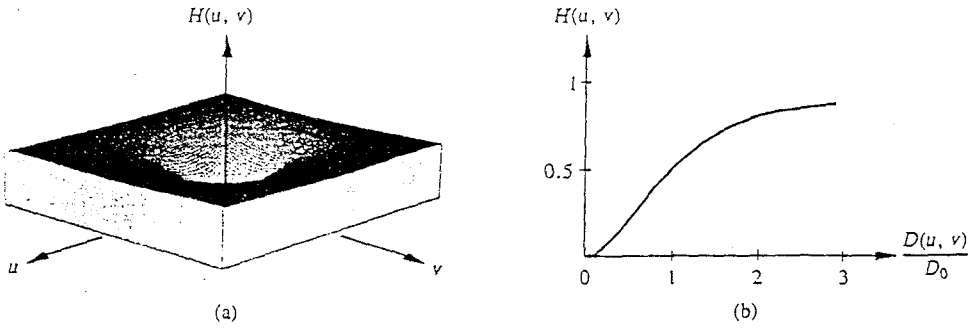


Figure 4.38 *Perspective plot and radial cross section of Butterworth highpass filter for n = 1.*

Example: Figure 4.39(a) shows a chest x-ray that was poorly developed, and Fig. 4.39(b) shows the image after it was processed with a highpass Butterworth filter of order 1. Only the edges are predominant in this image because the low-frequency components were severely attenuated, thus making different (but smooth) gray-level regions appear essentially the same.

A technique often used to alleviate this problem consists of adding a constant to a highpass filter transfer function in order to preserve the low-frequency components. This addition, of course, amplifies the high-frequency components

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

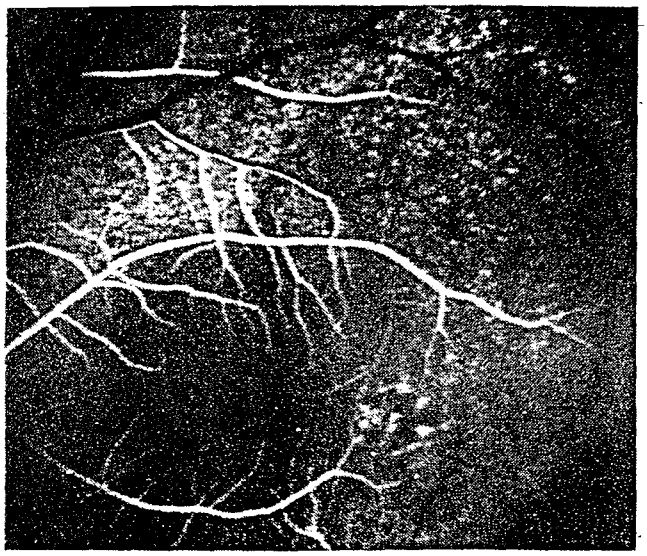
Figure 4.24 *A basic highpass spatial filter.*

$$\frac{1}{9} \times \begin{bmatrix} -1 & -1 & -1 \\ -1 & w & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

ure 4.26 *Mask used for high-boost spatial filtering. The value of the c*
$9A - 1$, with $A \geq 1$.

$W = 9A - 1 \quad A \geq 1$

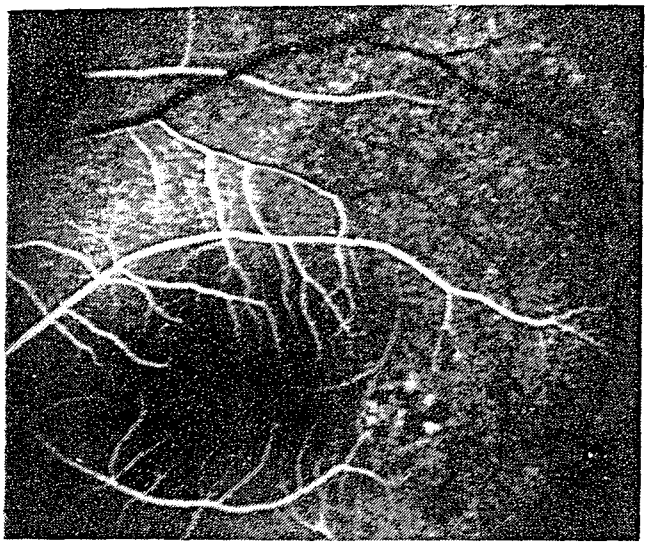
$\Rightarrow W \geq 8$



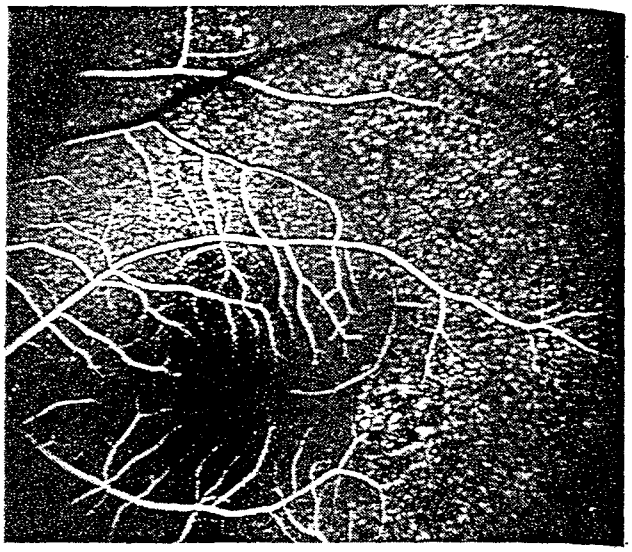
(a)

(b)

Figure 4.25 *(a) Image of a human retina; (b) highpass filtered result using the mask in Fig. 4.24.*



(a)



(b)

(c)

(d)

Figure 4.27 *(a) Original image; (b)–(d) result of high-boost filtering using the mask in Fig. 4.26, with A = 1.1, 1.15, and 1.2, respectively. Compare these results with those shown in Fig. 4.25.*

a. Original image.

b. Butterworth filter—Order = 2; Cutoff = 32.

c. Ideal filter—Cutoff = 32.

d. High-frequency emphasis filter—Offset = 0.5; Order = 2; Cutoff = 32.

e. High-frequency emphasis filter—Offset = 1.5; Order = 2; Cutoff = 32.

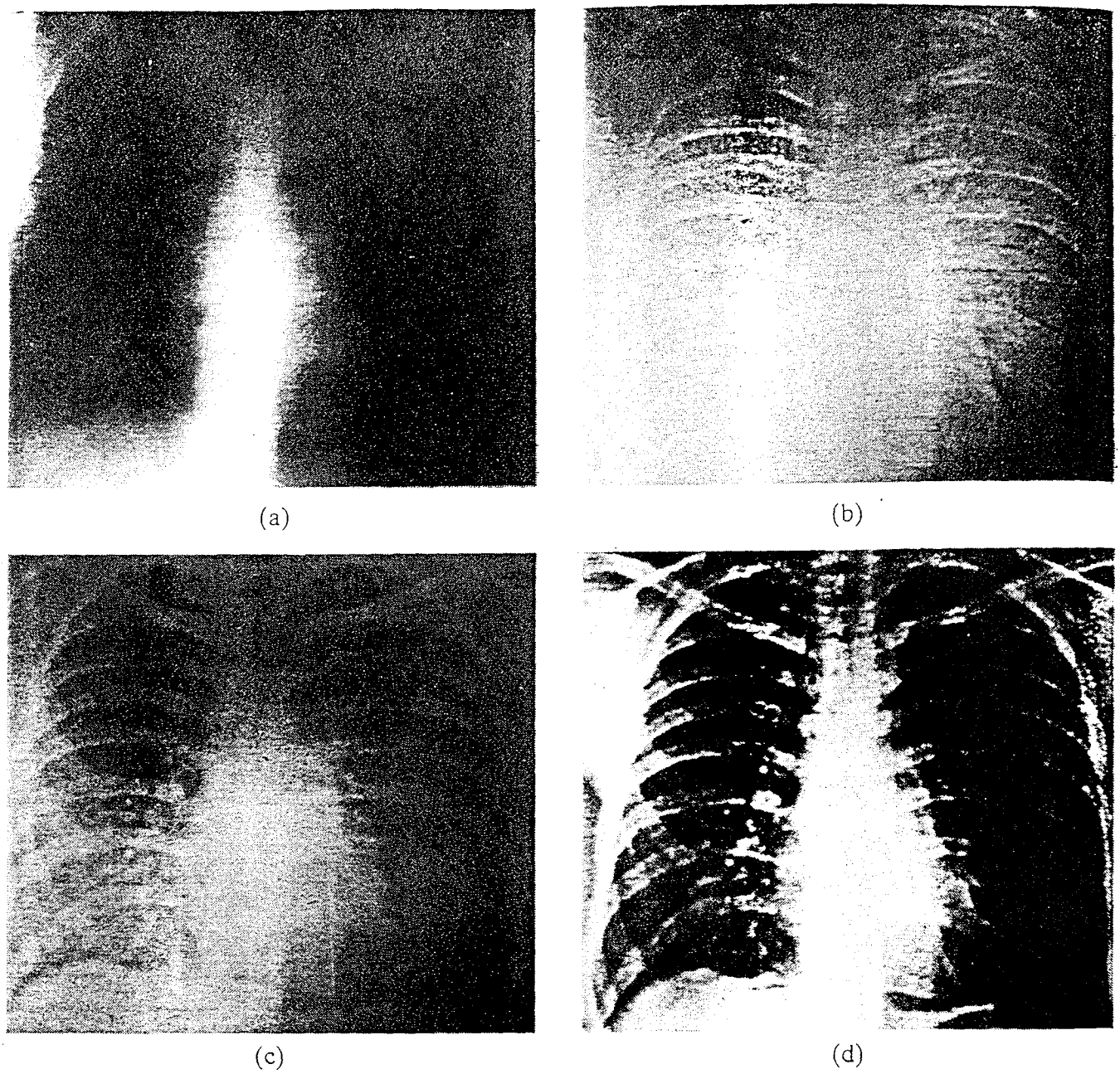


Figure 4.39 *Example of highpass filtering: (a) original image; (b) image processed with a highpass Butterworth filter; (c) result of high-frequency emphasis; (d) high-frequency emphasis and histogram equalization. (From Hall et al. [1971].)*

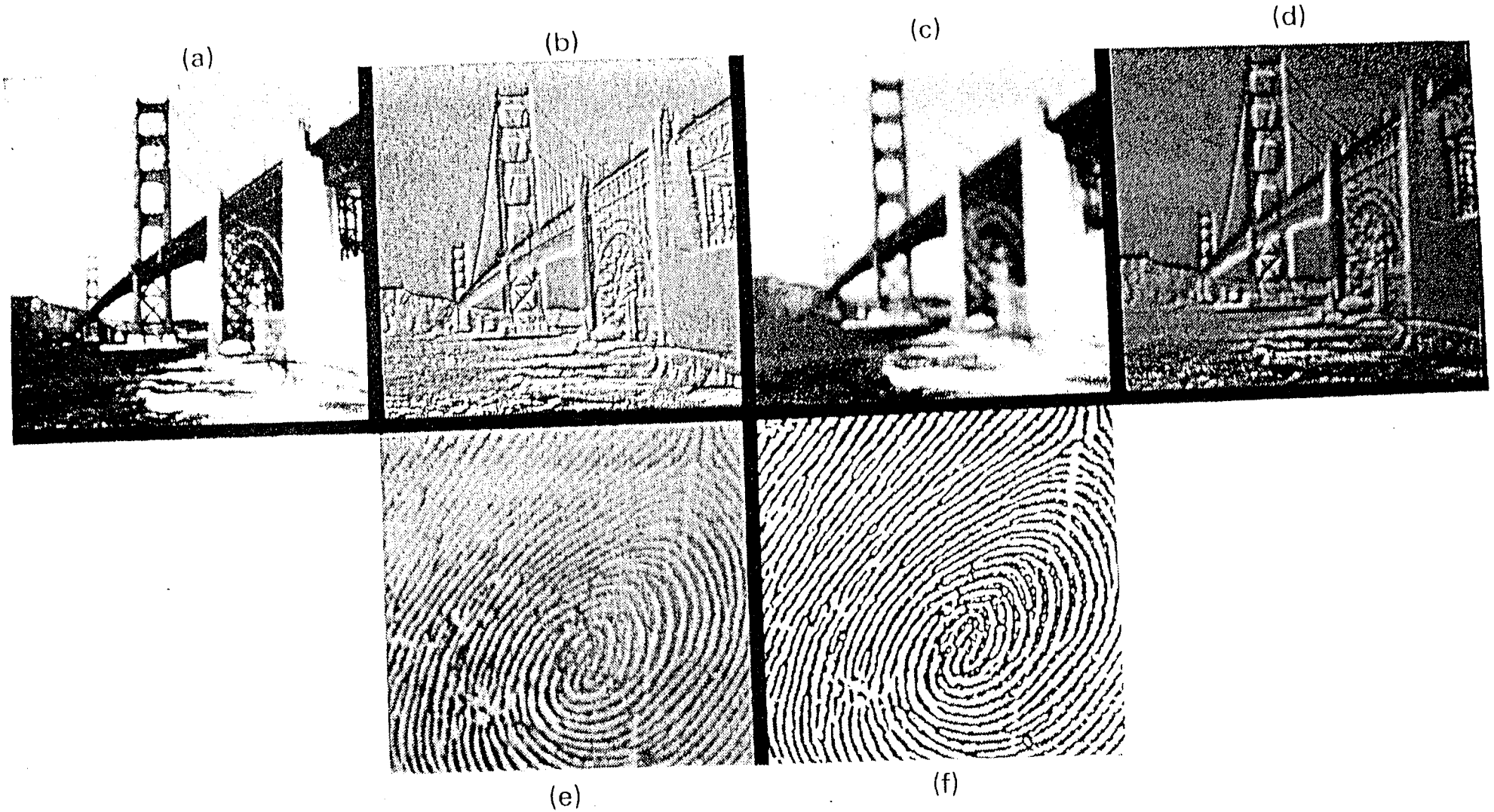
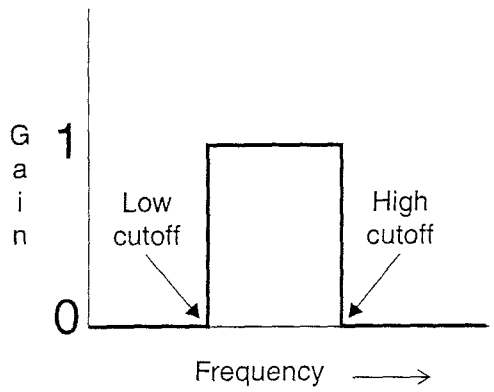
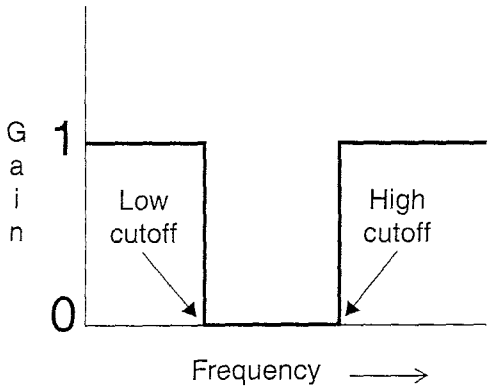


Figure 7.26 Spatial filtering examples.
Top row: original, high-pass, low-pass and band-pass filtered images.
Bottom row: original and high-pass filtered images.

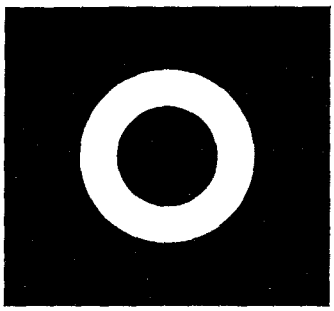
Figure 2.5-24 Bandpass, Bandreject, and Notch Filters



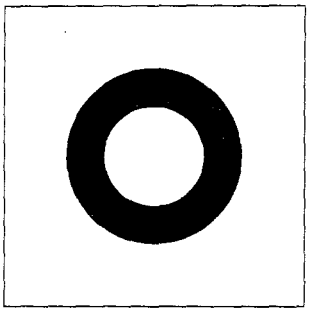
a. 1-D ideal bandpass filter.



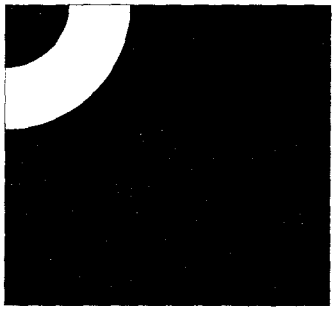
b. 1-D ideal bandreject filter.



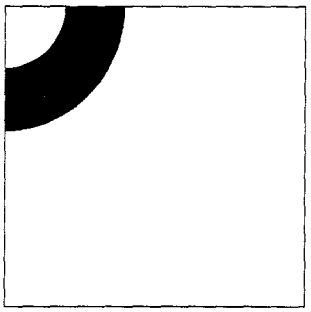
c. 2-D ideal bandpass filter shown as an image.



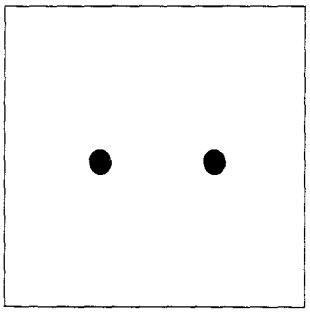
d. 2-D ideal bandreject filter shown as an image.



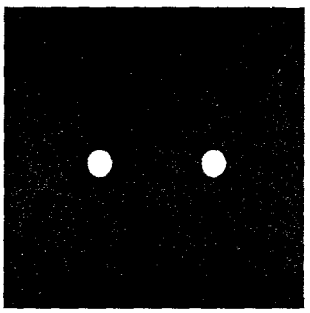
e. 2-D ideal bandpass filter for Walsh-Hadamard and cosine functions.



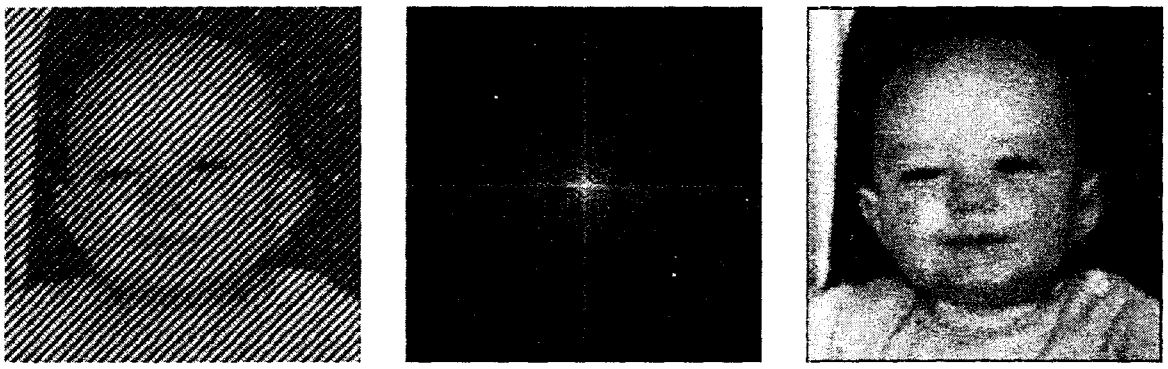
f. 2-D ideal bandreject filter for Walsh-Hadamard and cosine functions.



g. 2-D ideal notch filter for rejecting specific frequencies.



h. 2-D ideal notch filter for passing specific frequencies.



(a) (b) (c)

Figure 6.11 Example of structured noise removal. (a) Face image with an imposed sinusoidal pattern. (b) Fourier transform of the noisy image, showing the two spikes responsible for the pattern (Circled in white in the above image). (c) Restored image; the spikes were set to zero and then the inverse Fourier transform was computed (has been contrast enhanced).

As a simple example, Figure 6.11a shows the face image with high-frequency sinusoidal interference imposed on it. The Fourier transform of this image (Figure 6.11b) shows a number of bright spots (*spikes*). A matching pair of spikes appear in the upper left and lower right quarters of the image, and these correspond to the periodic signal causing the pattern of diagonal lines. To correct this, edit the Fourier domain image and set the two spikes to zero; then apply an inverse FFT to obtain the space domain image. The result, after some contrast improvement, appears in Figure 6.11c.

There are a number of questions still to be addressed. First among these concerns exactly which spikes to remove, and unfortunately there is no good answer. Experience with the appearance of Fourier domain images will help, and for this purpose the fftlib.c procedures will be very useful. In particular, it will be quickly learned that the peak in the center of the image contains much of the interesting information in the image, and must not be removed. Periodic signals cause symmetrical spikes on each side of the central peak, and though there can be many of these, they can be removed in pairs to see what happens. When the correct spikes are found, the image will be improved by their removal. Depending on the type of noise there could be more than one pair of spikes to be removed.

The restoration in this figure was performed by the program snr.c (Structured Noise Removal). This program computes the Fourier transform of the input image, and then interactively asks for the coordinates of small regions to be set to zero. After clearing the specified regions the image is back-transformed and saved. The restored image in Figure 6.11c was created by the following interactive session:

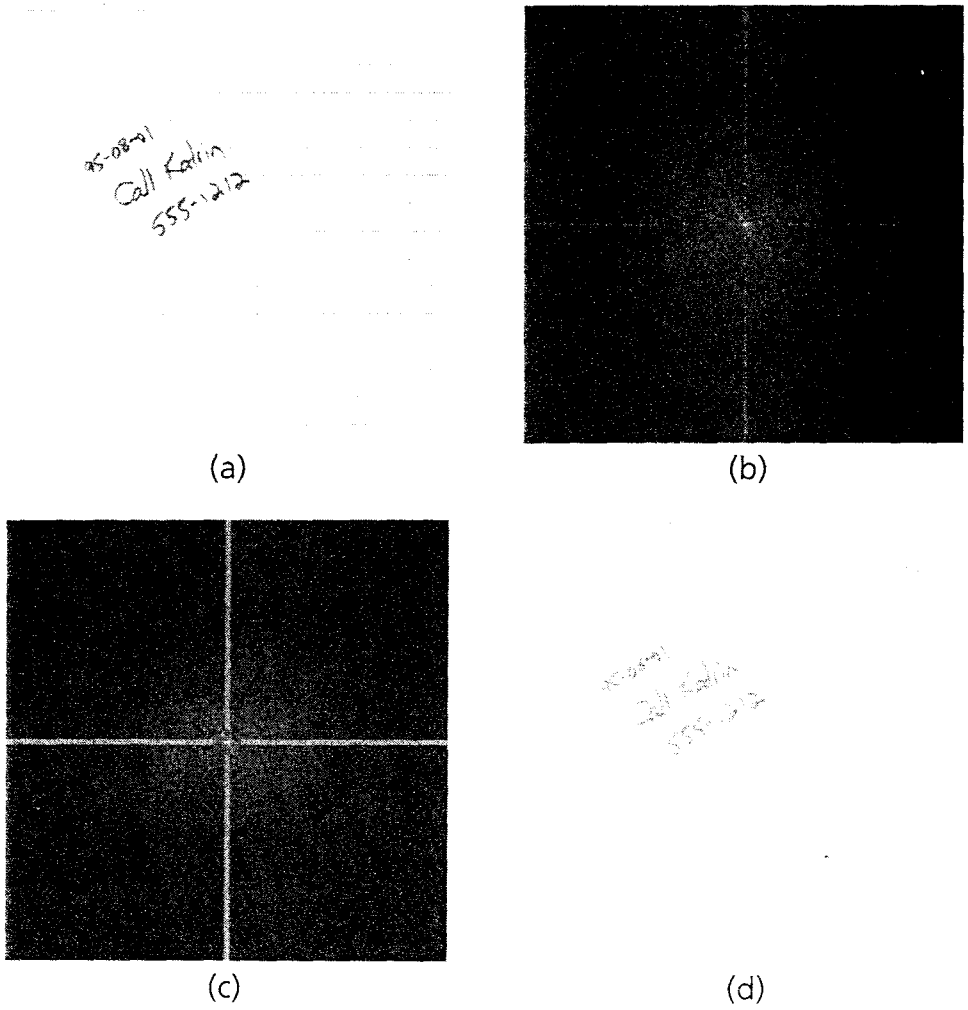


Figure 6.12 Removal of grid lines using a notch filter. (a) Original scanned image. (b) Fourier transform of scanned image. (c) Notches along the center lines, removing both the vertical and horizontal lines. (d) The restored image.

6.6 MOTION BLUR—A SPECIAL CASE

If an image has been blurred due to the motion of either the camera or the object, the point spread function will be an extended blob with the long axis indicating the direction of motion. While it is possible to use an inverse or Wiener filter restoration in these cases, there is a special solution that should not be as susceptible to noise.

If the motion can be assumed to be uniform and in the x (horizontal) direction, a very nice expression (Gonzalez 1992; Sondhi 1972) can be used to remove most of the blur without resorting to a Fourier transform:

11.4 HIGH-FREQUENCY ENHANCEMENT FILTERS

The term *high-frequency enhancement filter*, or *highpass filter*, is generally taken to describe a transfer function that is unity at zero frequency and increases with increasing frequency. Such a transfer function may either level off at some value greater than unity or, more commonly, fall back toward zero at higher frequencies. In the latter case, the high-frequency enhancement filter is actually a type of bandpass filter with the restriction of unity gain at zero frequency.

In practice, it is sometimes desired to have less than unity gain at zero frequency, so as to reduce the contrast of large, slowly varying components of the image. If the transfer function passes through the origin, it may be called a *Laplacian filter.*

11.4.1 The Difference-of-Gaussians Filter

We can produce a high-frequency enhancement transfer function by expressing it as the difference of two Gaussians of different widths:

$$G(s) = Ae^{-s^2/2\alpha_1^2} - Be^{-s^2/2\alpha_2^2} \qquad A \geq B,\ \alpha_1 > \alpha_2 \tag{12}$$

This is shown in Figure 11–8. The impulse response of such a filter is

$$g(t) = \frac{A}{\sqrt{2\pi\sigma_1^2}} e^{-t^2/2\sigma_1^2} - \frac{B}{\sqrt{2\pi\sigma_2^2}} e^{-t^2/2\sigma_2^2} \qquad \sigma_i = \frac{1}{2\pi\sigma_i} \tag{13}$$

and is graphed in Figure 11–9. Notice that the broad Gaussian in the frequency domain produces a narrow Gaussian in the time domain and vice versa. The impulse response shown in Figure 11–9 is typical of bandpass and highpass filters, having a positive pulse situated in a negative dish.

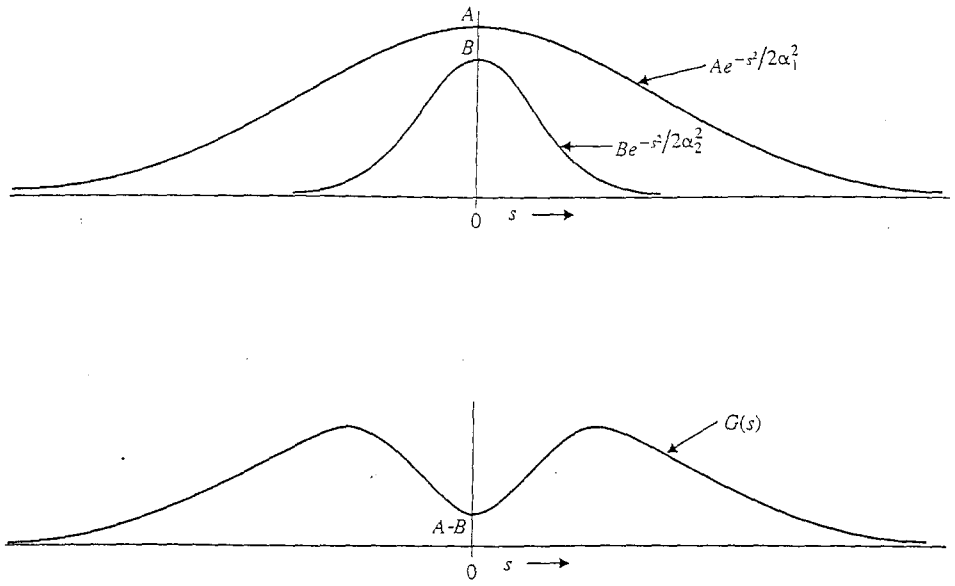


Figure 11–8 The Gaussian high-frequency enhancement transfer function

If we let α_1 approach infinity, the narrow Gaussian in the time domain narrows further to an impulse, and the filter has the form shown in Figure 11–10. Notice that the difference between a filter that rolls off (returns toward zero) at high frequencies and one that does not is the width of the central pulse in the time domain. In fact, the broader that central pulse, the faster the transfer function rolls off.

11.4.2 Rules of Thumb for Highpass Filter Design

In this section, we develop two rules that hold approximately for estimating the behavior of high-frequency enhancement filters. Suppose the impulse response of the filter is expressed as a narrow pulse minus a broad pulse, i.e.

11.3.3 The General Bandpass Filter

We now consider a class of bandpass filters constructed in the following way: We select a nonnegative unimodal function $K(s)$ and convolve it with an even impulse pair at frequency s_0. This yields a bandpass transfer function, as shown in Figure 11–6. That transfer function is given by

$$G(s) = K(s) * [\delta(s - s_0) + \delta(s + s_0)] \tag{8}$$

and the impulse response by

$$g(t) = 2k(t) \cos(2\pi s_0 t) \tag{9}$$

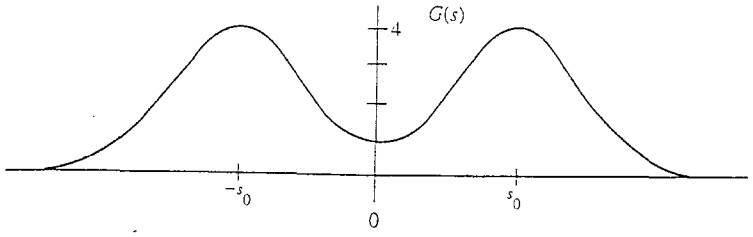


Figure 11–6 The general bandpass filter

This impulse response is a cosine of frequency s_0 in an envelope that is the inverse Fourier transform of $K(s)$.

Suppose, for example, that $K(s)$ is a Gaussian

$$G(s) = Ae^{-s^2/2\alpha^2} * [\delta(s - s_0) + \delta(s + s_0)] \tag{10}$$

Then the impulse response becomes

$$\sigma = \frac{1}{2\pi\alpha} \qquad g(t) = \frac{2A}{\sqrt{2\pi\sigma^2}} e^{-t^2/2\sigma^2} \cos(2\pi s_0 t) \tag{11}$$

This impulse response, a cosine in a Gaussian envelope, is graphed in Figure 11–7. Notice that we could easily generate a class of bandstop filters as well by this technique.

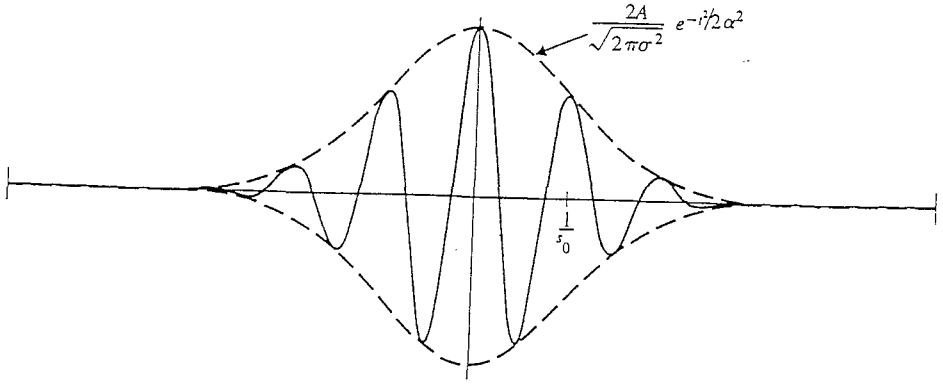


Figure 11–7 The Gaussian bandpass filter

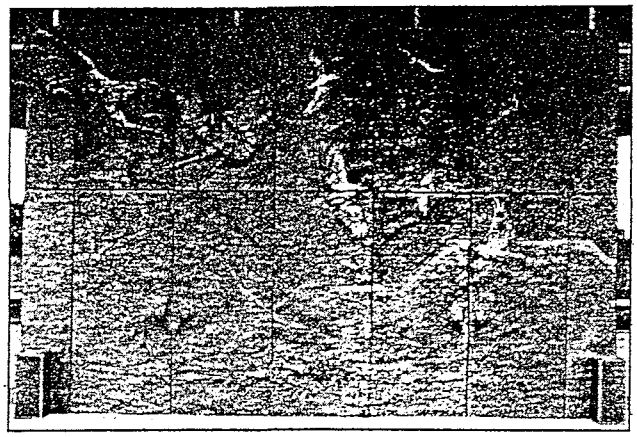
11.4 HIGH-FREQUENCY ENHANCEMENT FILTERS

The term *high-frequency enhancement filter,* or *highpass filter,* is generally taken to describe a transfer function that is unity at zero frequency and increases with increasing frequency. Such a transfer function may either level off at some value greater than unity or, more commonly, fall back toward zero at higher frequencies. In the latter case, the high-frequency enhancement filter is actually a type of bandpass filter with

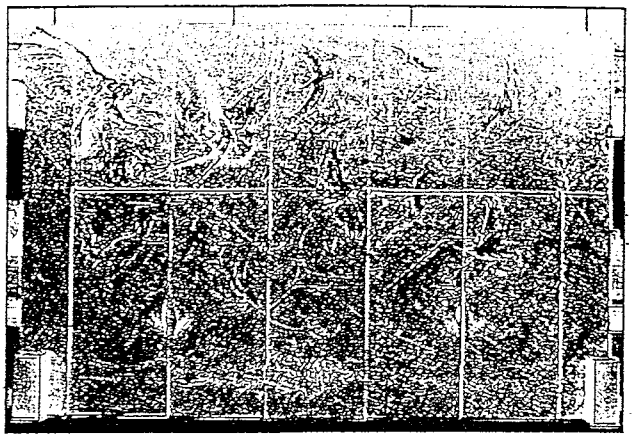
4.3.3 Homomorphic Filtering

The digital images we process are created from optical images. Optical ima consist of two primary components, the lighting component and the reflectance com nent. The lighting component results from the lighting conditions present when

Figure 4.3-1 High Boost Spatial Filter

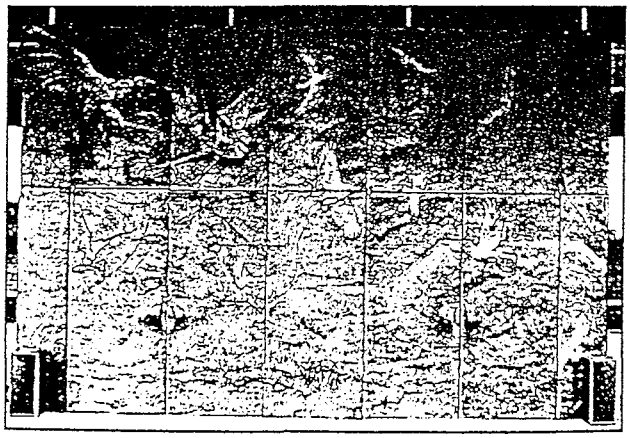


a. Original image.



b. High boost filter, 3 × 3 mask. A small center value, $x = 5$, results in a negative of the original.

c. High boost filter, 3 × 3 mask. A center value of $x = 8$ results in enhanced details.



d. High boost filter, 3 × 3 mask. A large center value, $x = 15$, retains more of the original image.

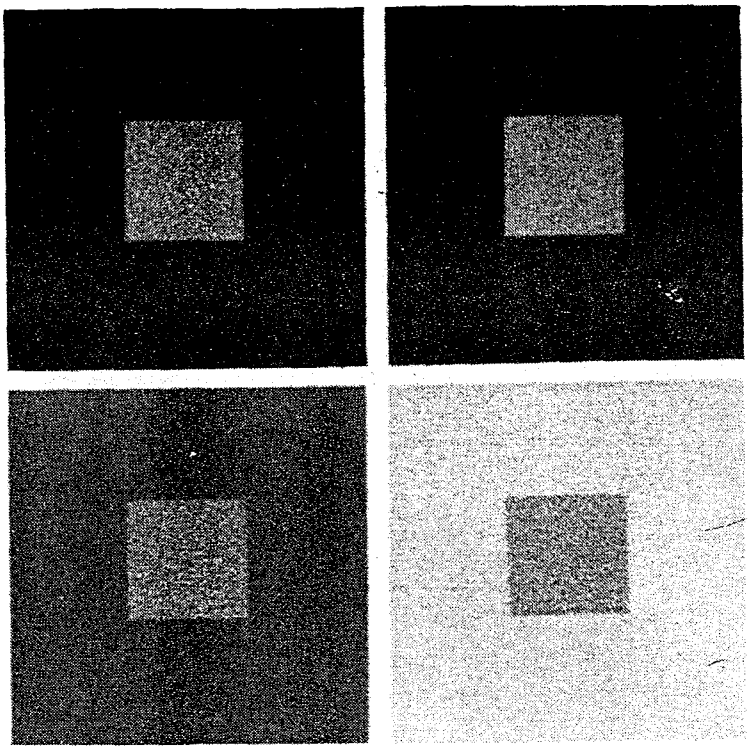


Figure 2.8 *Example of simultaneous contrast. All the small squares have exactly the same intensity, but they appear progressively darker as the background becomes lighter.*

the intensity (brightness) of the image at that point. As light is a form of energy, $f(x, y)$ must be nonzero and finite, that is,

$$0 < f(x, y) < \infty. \tag{2.2-1}$$

The images people perceive in everyday visual activities normally consist of light reflected from objects. The basic nature of $f(x, y)$ may be characterized by two components: (1) the amount of source light incident on the scene being viewed and (2) the amount of light reflected by the objects in the scene. Appropriately, they are called the *illumination* and *reflectance components*, and are denoted by $i(x, y)$ and $r(x, y)$, respectively. The functions $i(x, y)$ and $r(x, y)$ combine as a product to form $f(x, y)$:

$$f(x, y) = i(x, y)r(x, y) \tag{2.2-2}$$

where

$$0 < i(x, y) < \infty \tag{2.2-3}$$

and

$$0 < r(x, y) < 1. \tag{2.2-4}$$

Equation (2.2-4) indicates that reflectance is bounded by 0 (total absorption)

transform of the product of two functions is not separable; in other words,

$$\mathfrak{F}\{f(x, y)\} \neq \mathfrak{F}\{i(x, y)\}\mathfrak{F}\{r(x, y)\}.$$

Suppose, however, that we define

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y). \end{aligned} \tag{4.4-10}$$

Then,

$$\begin{aligned} \mathfrak{F}\{z(x, y)\} &= \mathfrak{F}\{\ln f(x, y)\} \\ &= \mathfrak{F}\{\ln i(x, y)\} + \mathfrak{F}\{\ln r(x, y)\} \end{aligned} \tag{4.4-11}$$

or

$$Z(u, v) = I(u, v) + R(u, v) \tag{4.4-12}$$

where $I(u, v)$ and $R(u, v)$ are the Fourier transforms of $\ln i(x, y)$ and $\ln r(x, y)$, respectively.

If we process $Z(u, v)$ by means of a filter function $H(u, v)$ then, from Eq. (4.1-4),

$$\begin{aligned} S(u, v) &= H(u, v)Z(u, v) \\ &= H(u, v)I(u, v) + H(u, v)R(u, v) \end{aligned} \tag{4.4-13}$$

where $S(u, v)$ is the Fourier transform of the result. In the spatial domain,

$$\begin{aligned} s(x, y) &= \mathfrak{F}^{-1}\{S(u, v)\} \\ &= \mathfrak{F}^{-1}\{H(u, v)I(u, v)\} + \mathfrak{F}^{-1}\{H(u, v)R(u, v)\}. \end{aligned} \tag{4.4-14}$$

By letting

$$i'(x, y) = \mathfrak{F}^{-1}\{H(u, v)I(u, v)\} \tag{4.4-15}$$

and

$$r'(x, y) = \mathfrak{F}^{-1}\{H(u, v)R(u, v)\} \tag{4.4-16}$$

Eq. (4.4-14) can be expressed in the form

$$s(x, y) = i'(x, y) + r'(x, y). \tag{4.4-17}$$

Finally, as $z(x, y)$ was formed by taking the logarithm of the original image $f(x, y)$, the inverse operation yields the desired enhanced image $g(x, y)$; that

is,

$$\begin{aligned} g(x, y) &= \exp[s(x, y)] \\ &= \exp[i'(x, y)] \cdot \exp[r'(x, y)] \\ &= i_0(x, y) r_0(x, y) \end{aligned} \tag{4.4-18}$$

where

$$i_0(x, y) = \exp[i'(x, y)] \tag{4.4-19}$$

and

$$r_0(x, y) = \exp[r'(x, y)] \tag{4.4-20}$$

are the illumination and reflectance components of the output image.

The enhancement approach using the foregoing concepts is summarized in Fig. 4.40. This method is based on a special case of a class of systems known as *homomorphic systems*. In this particular application, the key to the approach is that separation of the illumination and reflectance components is achieved in the form shown in Eq. (4.4-12). The *homomorphic filter function* $H(u, v)$ can then operate on these components separately, as indicated in Eq. (4.4-13).

The illumination component of an image is generally characterized by slow spatial variations, while the reflectance component tends to vary abruptly, particularly at the junctions of dissimilar objects. These characteristics lead to associating the low frequencies of the Fourier transform of the logarithm of an image with illumination and the high frequencies with reflectance. Although these associations are rough approximations, they can be used to advantage in image enhancement.

A good deal of control can be gained over the illumination and reflectance components with a homomorphic filter. This control requires specification of a filter function $H(u, v)$ that affects the low- and high-frequency components of the Fourier transform in different ways. Figure 4.41 shows a cross section of such a function. A complete specification of $H(u, v)$ is obtained by rotating the cross section 360° about the vertical axis. If the parameters γ_L and γ_H are chosen so that $\gamma_L < 1$ and $\gamma_H > 1$, the filter function shown in Fig. 4.41 tends

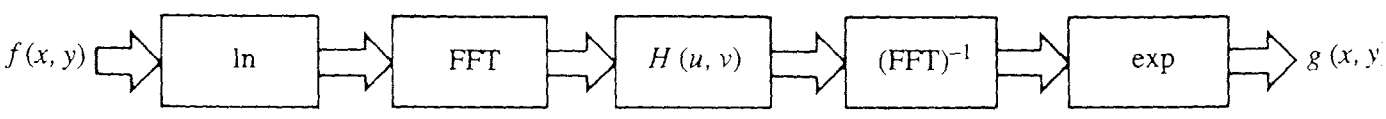


Figure 4.40 *Homomorphic filtering approach for image enhancement.*

image is captured and can change as the lighting conditions change. The reflectance component results from the way the objects in the image reflect light and are determined by the intrinsic properties of the object itself, which (normally) do not change. In many applications it is useful to enhance the reflectance component, while reducing the contribution from the lighting component. *Homomorphic filtering* is a frequency domain filtering process that compresses the brightness (from the lighting conditions) while enhancing the contrast (from the reflectance properties of the objects).

The image model for homomorphic filters is as follows:

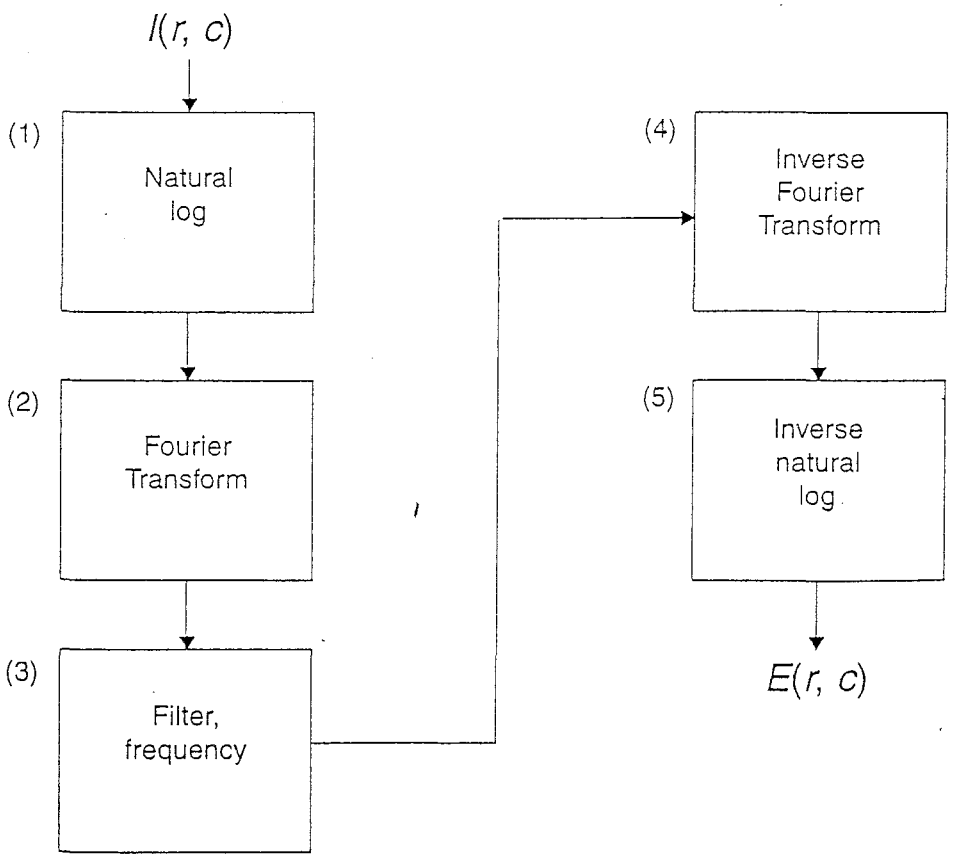
$$I(r, c) = L(r, c)R(r, c)$$

where $L(r, c)$ represents contribution of the lighting conditions
$R(r, c)$ represents contribution of the reflectance properties of the objects

The homomorphic filtering process assumes that $L(r, c)$ consists of primarily slow spatial changes (low spatial frequencies) and is responsible for the overall range of the brightness in the image. The assumptions for $R(r, c)$ are that it consists primarily of high spatial frequency information, which is especially true at object boundaries, and that it is responsible for the local contrast (the spread within a small spatial area). These simplifying assumptions are valid for many types of real images.

The homomorphic filtering process consists of five steps: 1) a natural log transform (base e), 2) the Fourier transform, 3) filtering, 4) the inverse Fourier transform, and 5) the inverse log function—the exponential. This process is illustrated in a block diagram in Figure 4.3-2. The first step allows us to decouple the $L(r, c)$ and $R(r, c)$ components because the logarithm function changes a product into a sum. Step 2 puts

Figure 4.3-2 The Homomorphic Filtering Process



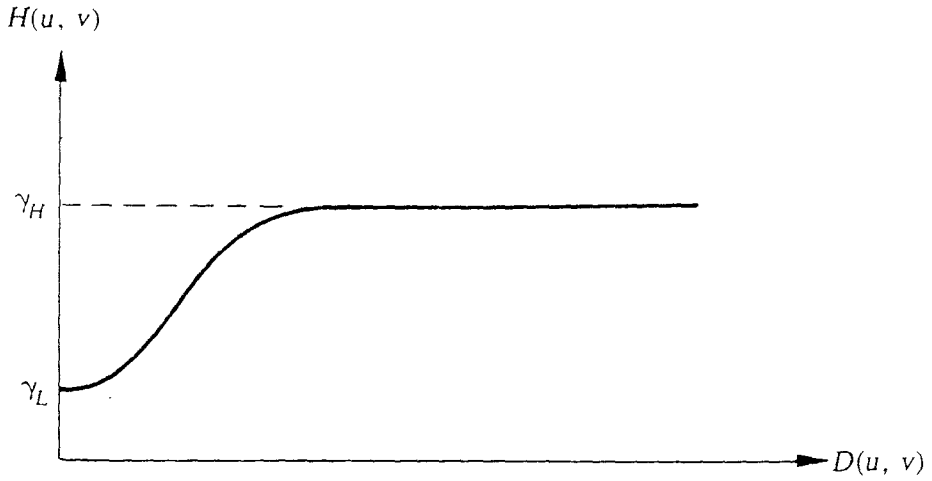
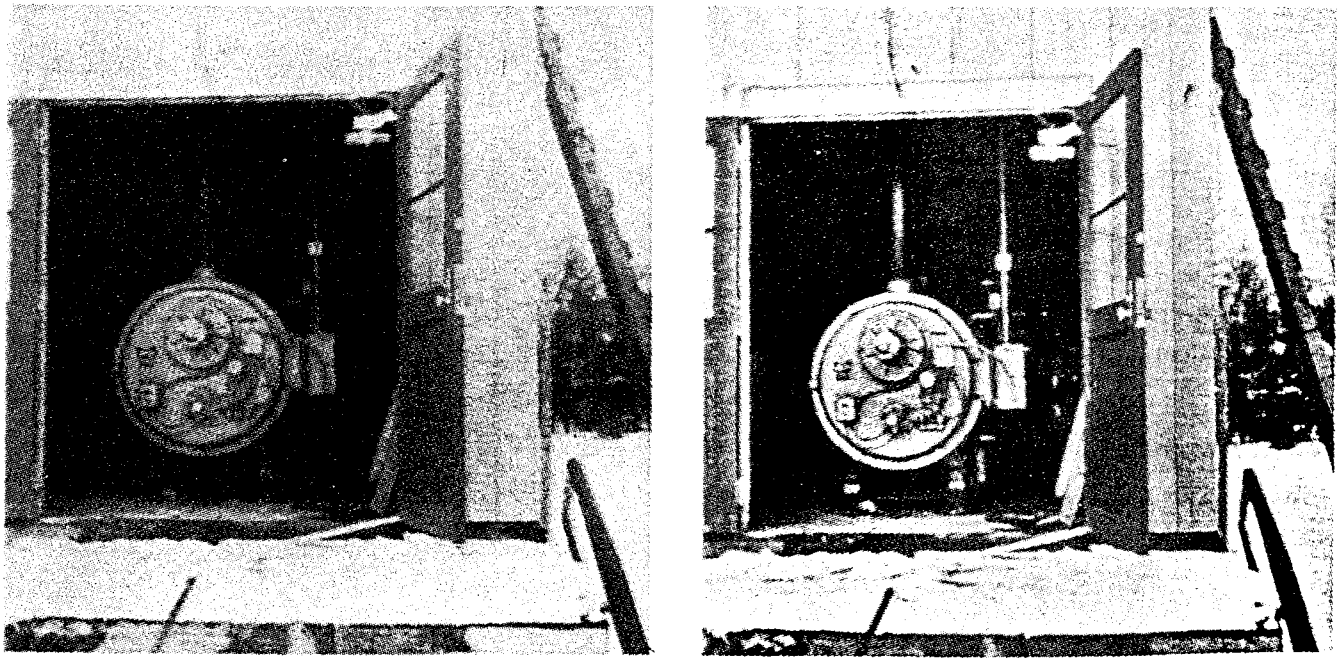


Figure 4.41 *Cross section of a circularly symmetric filter function for use in homomorphic filtering. D(u, v) is the distance from the origin.*

to decrease the low frequencies and amplify the high frequencies. The net result is simultaneous dynamic range compression and contrast enhancement.

Example: Figure 4.42 is typical of the results that can be obtained with the homomorphic filter function shown in Fig. 4.41. In the original image, Fig. 4.42(a), the details inside the room are obscured by the glare from the outside walls. Figure 4.42(b) shows the result of processing this image by homomorphic



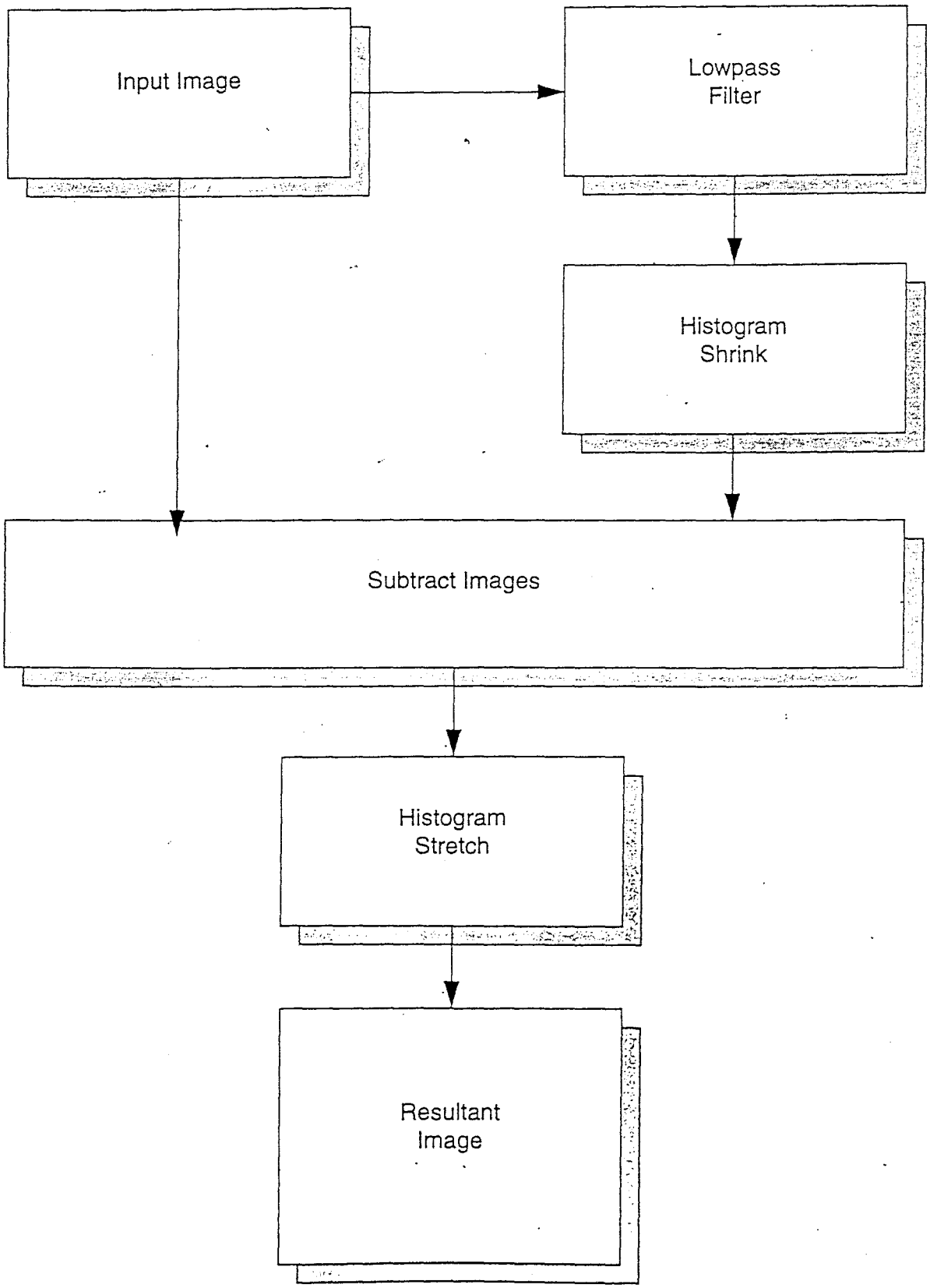
(a) (b)

Figure 4.42 *(a) Original image; (b) image processed by homomorphic filtering to achieve simultaneous dynamic range compression and contrast enhancement. (From Stockham [1972].)*

4.3.4 Unsharp Masking

The unsharp masking enhancement algorithm is representative of practical image sharpening methods. It combines many of the operations discussed, including filtering and histogram modification. A flowchart for this process is shown in Figure 4.3-5. Here we see that the original image is lowpass filtered, followed by a histogram shrink to the lowpass-filtered image. The resultant image from these two operations is then subtracted from the original image, and the result of this operation undergoes a histogram stretch to restore the image contrast. This process works because subtracting a slowly changing edge (the lowpass-filtered image) from faster changing edges (in the original) has the visual effect of causing overshoot and undershoot at the edges which has the effect of emphasizing the edges. By scaling the lowpassed image with a histogram shrink, we can control the amount of edge emphasis desired (see Image Processing exercise #7 in Chapter 8). In Figure 4.3-6 are the results of applying the unsharp masking algorithm with different ranges for the histogram shrink process. Here we see that as the range for the histogram shrink is increased, the resulting image has a greater edge emphasis.

Figure 4.3-5 Unsharp Masking Enhancement



4.3-6 Unsharp Masking

a. Original image.

b. Unsharp masking with lower limit = 0, upper = 100, with 2% low and high clipping.

c. Unsharp masking with lower limit = 0, upper = 150, with 2% low and high clipping.

d. Unsharp masking with lower limit = 0, upper = 200, with 2% low and high clipping.