11 Averaging

11.1 Introduction

In this chapter we will discuss neighborhood operations for performing the elementary task of averaging. This operation is of central importance for low-level image processing. It is one of the building blocks for the more complex feature extraction operators discussed in Chapters 13–15.

In the simplest case, objects are identified as *regions* of constant radiance, i. e., gray values. Then, averaging gives adequate mean values of the gray values within the object. This approach, of course, implies a simple model of the image content. The objects of interest must indeed be characterized by constant gray values that are clearly different from the background and/or other objects.

However, this assumption is seldom met in real-world applications. The intensities will generally show some variations. These variations may be an inherent feature of the object or could be caused by the image formation process. Typical cases are *noise*, a *non-uniform illumination*, or *inhomogeneous background*.

In complex cases, it is not possible to distinguish objects from the background with just one feature. Then it may be a valid approach to compute more than one feature image from one and the same image. This results in a multicomponent or *vectorial feature image*.

The same situation arises when more than one image is taken from a scene as with *color images* or any type of *multispectral image*. Therefore, the task of averaging must also be applied to vectorial images. In image sequences, averaging is extended into the time coordinate to a spatiotemporal averaging.

11.2 General Properties of Averaging Filters

Convolution provides the framework for all elementary averaging filters. These filters have a number of properties in common that are discussed in this section.

B. Jähne, Digital Image Processing
ISBN 3-540-24035-7

11.2.1 Zero Shift

With respect to object detection, the most important feature of a smoothing convolution operator is that it must not shift the object position. Any shift introduced by a preprocessing operator would cause errors in the estimates of the position and possibly other geometric features of an object. In order to cause no shift, the transfer function of a filter must be real. A filter with this property is known as a *zero-phase filter*, because it does not introduce a phase shift in any of the periodic components of an image. A real transfer function implies a symmetric filter mask (Section 2.3). A W-dimensional symmetric convolution mask is defined by

$$\begin{aligned} &\text{1-D:} \quad h_{-n} = h_n \\ &\text{2-D:} \quad h_{-m,n} = h_{m,n},\ h_{m,-n} = h_{m,n} \\ &\text{3-D:} \quad h_{-l,m,n} = h_{l,m,n},\ h_{l,-m,n} = h_{l,m,n},\ h_{l,m,-n} = h_{l,m,n}. \end{aligned} \tag{11.1}$$

The symmetry relations also significantly ease the computation of the transfer functions as only the cosine term of the complex exponential from the Fourier transform remains in the equations. The transfer function for 1-D symmetric masks with an odd number of coefficients $(2R + 1)$ is

$$\boxed{\hat{h}(\tilde{k}) = h_0 + 2\sum_{v=1}^{R} h_v \cos(v\pi\tilde{k}).} \tag{11.2}$$

With an even number of coefficients $(2R)$, the transfer function of a 1-D symmetric mask is given by

$$\boxed{\hat{h}(\tilde{k}) = 2\sum_{v=1}^{R} h_v \cos((v - 1/2)\pi\tilde{k}).} \tag{11.3}$$

Note that the wave numbers are half-integers $v = 1/2, 3/2, ...$, because for symmetry reasons the result of the convolution with an even-sized mask lies on the intermediate grid.

For a 2-D symmetric mask with an odd number of coefficients in both directions we obtain correspondingly:

$$\begin{aligned} \hat{h}(\tilde{\boldsymbol{k}}) &= h_{00} \\ &+ 2\sum_{v=1}^{r} h_{0v} \cos(v\pi\tilde{k}_1) + \sum_{u=1}^{R} h_{u0} \cos(u\pi\tilde{k}_2) \\ &+ 4\sum_{u=1}^{R}\sum_{v=1}^{R} h_{uv} \cos(v\pi\tilde{k}_1)\cos(u\pi\tilde{k}_2). \end{aligned} \tag{11.4}$$

A further discussion of the properties of symmetric masks up to three dimensions can be found in Jähne [89].

11.2.2 Preservation of Mean

A smoothing operator must preserve the mean value. This condition says that the transfer function for the zero wave number is 1 or, equivalently, that the sum of all coefficients of the mask is 1:

$$\begin{aligned} &\text{1-D:} \quad \hat{h}(0) = 1 \quad \sum_n h_n = 1 \\ &\text{2-D:} \quad \hat{h}(\mathbf{0}) = 1 \quad \sum_m \sum_n h_{mn} = 1 \\ &\text{3-D:} \quad \hat{h}(\mathbf{0}) = 1 \quad \sum_l \sum_m \sum_n h_{lmn} = 1. \end{aligned} \tag{11.5}$$

11.2.3 Monotonically Decreasing Transfer Function

Intuitively, we expect that a smoothing operator attenuates smaller scales more strongly than coarser scales. More specifically, a smoothing operator should not completely annihilate a certain scale while smaller scales still remain in the image. Mathematically speaking, this means that the transfer function decreases monotonically with the wave number:

$$\hat{h}(\tilde{k}_2) \leq \hat{h}(\tilde{k}_1) \quad \text{if} \quad \tilde{k}_2 > \tilde{k}_1. \tag{11.6}$$

We may impose the more stringent condition that for the highest wave numbers the transfer function is identical to zero:

$$\begin{aligned} &\text{1-D:} \quad \hat{h}(1) = 0 \\ &\text{2-D:} \quad \hat{h}(\tilde{k}_1, 1) = 0, \quad \hat{h}(1, \tilde{k}_2) = 0 \\ &\text{3-D:} \quad \hat{h}(\tilde{k}_1, \tilde{k}_2, 1) = 0, \quad \hat{h}(\tilde{k}_1, 1, \tilde{k}_3) = 0, \quad \hat{h}(1, \tilde{k}_2, \tilde{k}_3) = 0. \end{aligned} \tag{11.7}$$

Together with the monotonicity condition and the preservation of the mean value, this means that the transfer function decreases monotonically from one to zero for each averaging operator.

11.2.4 Isotropy

In most applications, the smoothing should be the same in all directions in order not to prefer any direction. Thus, both the filter mask and the transfer function should be isotropic. Consequently, the filter mask depends only on the magnitude of the distance from the center pixel and the transfer function on the magnitude of the wave number:

$$h(\boldsymbol{x}) = h(|\boldsymbol{x}|) \quad \text{and} \quad \hat{h}(\tilde{\boldsymbol{k}}) = \hat{h}(|\tilde{\boldsymbol{k}}|). \tag{11.8}$$

In discrete space, of course, this condition can only be met approximately. Therefore, it is an important design goal to construct discrete masks with minimum deviation from isotropy.

11.3 Box Filter

11.3.1 Introduction

It is obvious that smoothing filters will average pixels within a small neighborhood. The simplest method is to add all the pixels within the filter mask and to divide the sum by the number of pixels. Such a simple filter is called a *box filter*. Box filters are an illustrative example as to how to design a filter properly. As an introduction, we consider a 1×3 box filter

$$ {}^3\boldsymbol{R} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}. \tag{11.9} $$

The factor $1/3$ scales the result of the convolution sum in order to preserve the mean value (Section 11.2.2). Otherwise the gray value in a region with constant gray values is not preserved. We apply this mask to a vertical edge

$$ \begin{matrix} \vdots & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \cdots & 0 & 0 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix} \vdots & \vdots & \vdots & \vdots & \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \cdots & 0 & 1/3 & 2/3 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} $$

As expected for a smoothing operation, the sharp edge is transformed into a smoother ramp with a gradual transition from 0 to 1. Smoothing filters attenuate structures with high wave numbers. Let us test this first with a vertical structure with a wavelength of 3 pixel distance:

$$ \begin{matrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 1 & -2 & 1 & 1 & -2 & 1 & \cdots \\ 1 & -2 & 1 & 1 & -2 & 1 & \cdots \\ 1 & -2 & 1 & 1 & -2 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{matrix} $$

It turns out that the 1×3 box filter completely removes a structure with the wavelength 3. As already discussed in Section 11.2.3, we expect that all structures with a wave number above a certain threshold are removed by a good smoothing filter. This is not the case for the 1×3 box filter. A structure with the wavelength 2 is only attenuated by a factor of 3:

$$ \begin{matrix} \vdots & \vdots & \vdots & \vdots & \\ \cdots & 1 & -1 & 1 & -1 & \cdots \\ \cdots & 1 & -1 & 1 & -1 & \cdots \\ \cdots & 1 & -1 & 1 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} * \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = \begin{matrix} \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & -1/3 & 1/3 & -1/3 & 1/3 & \cdots \\ \cdots & -1/3 & 1/3 & -1/3 & 1/3 & \cdots \\ \cdots & -1/3 & 1/3 & -1/3 & 1/3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{matrix} $$

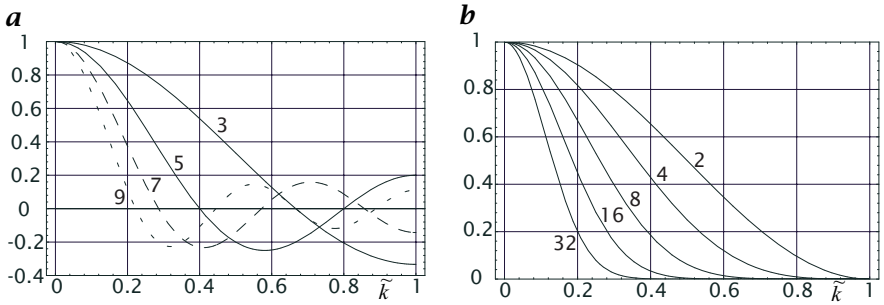

Figure 11.1: *Transfer functions of one-dimensional smoothing filters:* ***a*** *box filters with 3, 5, 7, and 9 coefficients;* ***b*** *binomial filters $\mathcal{B}^p$ with $p = 2$, 4, 8, 16, and 32.*

11.3.2 1-D Box Filter

After this qualitative introduction, we discuss box filters quantitatively by computing the transfer function. For sake of simplicity, we start with 1-D filters. The mask of the box filter Eq. (11.9) is of even symmetry. According to the considerations in Section 4.2.6, we can apply Eq. (4.25) to compute the transfer function of the one-dimensional 1×3 box filter. Only the coefficients $h_0 = h_1 = 1/3$ are unequal to zero and the transfer function reduces to

$$^3\hat{r}(\tilde{k}) = \frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k}). \tag{11.10}$$

The transfer function is shown in Fig. 11.1a. Our quick computation at the beginning of this section is verified. The transfer function shows a zero at $\tilde{k} = 2/3$. This corresponds to a wave number that is sampled 3 times per wavelength. The smallest possible wavelength ($\tilde{k} = 1$), which is sampled twice per wavelength, is only damped by a factor of three. The transfer function is negative for $\tilde{k} > 2/3$. A negative transfer function means an interchange of minimum and maximum values, equal to a phase shift of 180°. In conclusion, the 1×3 box filter is not a good lowpass filter. It is disturbing that the attenuation does not increase monotonically with the wave number but oscillates. Even worse, structures with the largest wave number are not attenuated strongly enough.

Larger box filters

$$^R\boldsymbol{R} = \frac{1}{R}\left[\underbrace{1\ 1\ \ldots\ 1}_{R \text{ times}}\right] \tag{11.11}$$

with R coefficients and the transfer function

$$\boxed{^R\hat{r}(\tilde{k}) = \frac{\sin(\pi R\tilde{k}/2)}{R\sin(\pi\tilde{k}/2)}} \tag{11.12}$$

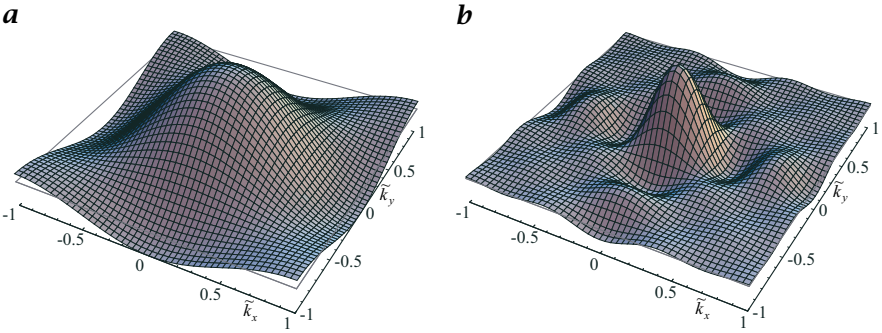

Figure 11.2: *Transfer functions of two-dimensional box filters shown in a pseudo 3-D plot:* ***a*** 3×3 *box filter;* ***b*** 7×7 *box filter.*

do not show a significant improvement (Fig. 11.1a). On the contrary, the oscillatory behavior is more pronounced and the attenuation is only proportional to the wave number. For large filter masks, the discrete mask with R coefficients comes close to a continuous box function of width R. Therefore the transfer function approximates the sinc function (≻ R5) at low wave numbers $\tilde{k} \ll 1$:

$$^R\hat{r}_x(\tilde{k}) \approx \frac{\sin(\pi R\tilde{k}/2)}{\pi R\tilde{k}/2} = \operatorname{sinc}(R\tilde{k}/2). \tag{11.13}$$

11.3.3 2-D Box Filter

Now we turn to two-dimensional box filters. To simplify the arithmetic, we utilize the fact that the filter is separable and decompose it into vertical and horizontal 1-D components:

$$^3\boldsymbol{R} = {}^3\boldsymbol{R}_x * {}^3\boldsymbol{R}_y = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

The transfer function of the one-dimensional filters is given by Eq. (11.10) (replacing $\tilde{k}_x$ by $\tilde{k}_y$ for the vertical filter). As convolution in the space domain corresponds to multiplication in the wave number domain, the transfer function of $\boldsymbol{R}$ is

$$^3\hat{r} = {}^3\hat{r}_x\,{}^3\hat{r}_y = \left[\frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k}_x)\right]\left[\frac{1}{3} + \frac{2}{3}\cos(\pi\tilde{k}_y)\right]. \tag{11.14}$$

11.3.4 Evaluation

From Eq. (11.14) and Fig. 11.2a, we can conclude that 2-D box filters are also poor lowpass filters. A larger box filter, for example one with a

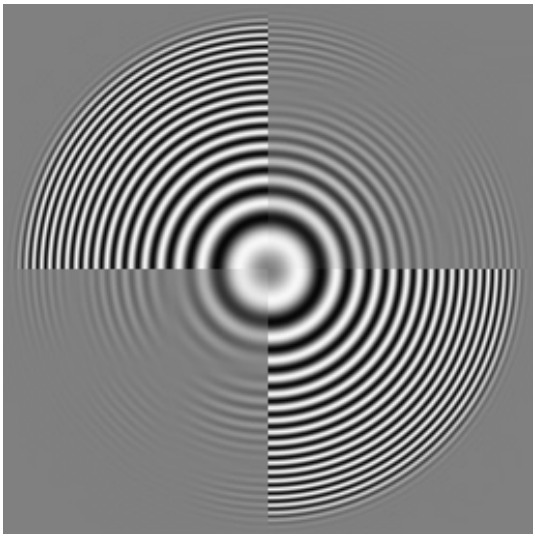

Figure 11.3: *Test of the smoothing with a* 5×5 *(upper right quadrant) and a* 9×9 *box filter (lower left quadrant) using a test image with concentric sinusoidal rings. The maximum wave number* $\tilde{k}$ *at the edge of the pattern is 0.6.*

7×7 mask (Fig. 11.2b), does not perform any better. Besides the disadvantages already discussed for the one-dimensional case, we are faced with the problem that the transfer function is not *isotropic*, i. e., it depends, for a given wave number, on the direction of the wave number.

When we apply a box filter to an arbitrary image, all these disadvantages affect the image, but it is difficult to observe them quantitatively (Fig. 11.6). They are revealed immediately, however, if we use a carefully designed *test image*. This image contains concentric sinusoidal rings. The wavelength of the rings decreases with distance from the center. With this test image, we map the Fourier domain onto the space domain. Thus, we can directly see the transfer function, i. e., the change in the amplitude and the phase shift, when a filter is applied. When we convolve this image with a 5×5 or 9×9 box filter, the deviations from an isotropic transfer function become readily visible (Fig. 11.3). We can observe the wave numbers that vanish entirely and the change of gray value maxima into gray value minima and vice versa in some regions, indicating the 180° phase shift caused by negative values in the transfer function.

From this experience, we can learn an important lesson. We must not rate the properties of a filter operation from its effect on arbitrary images, even if we think that they seem to work correctly. Obviously, the eye perceives a rather qualitative impression, but for quantitative extraction of image features a quantitative analysis of the filter proper-

ties is required. This involves a careful analysis of the transfer function and the application of the filters to carefully designed test images.

Now we turn back to the question of what went wrong with the box filter. We might try to design a better smoothing filter directly in the wave number space. An ideal smoothing filter would cut off all wave numbers above a certain threshold value. We could use this ideal transfer function and compute the filter mask by an inverse Fourier transform. However, we run into two problems, which can be understood without explicit calculations. The inverse Fourier transform of a box function is a sinc function. This means that the coefficients decrease only proportionally to the distance from the center pixel. We would be forced to work with large filter masks. Furthermore, the filter has the disadvantage that it overshoots at the edges.

11.3.5 Fast Computation

Despite all the disadvantages of box filters, they show one significant advantage. According to the following equation, the convolution with a one-dimensional box filter can be computed independently of its size with only three operations as a recursive filter operation:

$$g'_m = g'_{m-1} + \frac{1}{2r+1}(g_{m+r} - g_{m-r-1}). \tag{11.15}$$

This recursion can be understood by comparing the computations for the convolution at neighboring pixels. When the box mask is moved one position to the right, it contains the same weighting factor for all pixels except for the last and the first pixel. Thus, we can simply take the result of the previous convolution, (g'_{m-1}), subtract the first pixel that just moved out of the mask, (g_{m-r-1}), and add the gray value at the pixel that just came into the mask, (g_{m+r}). In this way, the computation of a box filter does not depend on its size and the number of computations is $O(r^0)$. Only one addition, one subtraction, and one multiplication are required to compute the filter result.

11.4 Binomial Filter

11.4.1 Basics

From our experience with box filters, we conclude that the design of filters is a difficult optimization problem. If we choose a small rectangular filter mask, we get a poor transfer function. If we start with an ideal transfer function, we get large filter masks and overshooting filter responses. The reason for this behavior is the fundamental relation between smoothness and compactness of the Fourier transform pairs (Section 2.3.4). An edge constitutes a discontinuity. A discontinuity

leads to an impulse in the first derivative. The Fourier transform of an impulse is evenly spread over the whole Fourier domain. Using the integral property of the Fourier transform (Section 2.3), an integration of the derivative in the space domain means a division by $\boldsymbol{k}$ in the Fourier domain (≻ R5). Then we know without any detailed calculation that in the one-dimensional case the envelope of the Fourier transform of a function which shows discontinuities in the space domain will decline with k^{-1} in the wave number domain. This was exactly what we found for the box function. Its Fourier transform is the sinc function (≻ R5).

Considering this basic fact, we can design better smoothing filters. One condition is that the filter masks should gradually approach zero.

11.4.2 1-D Binomial Filter

Here we will introduce a class of smoothing filters that meets this criterion and can be calculated very efficiently. Furthermore, these filters are an excellent example of how more complex filters can be built from simple components. The simplest and most elementary smoothing mask we can think of is

$$\boldsymbol{B} = \frac{1}{2}[1\ 1]. \tag{11.16}$$

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

$$\frac{1}{2^R}\underbrace{[1\ 1] * [1\ 1] * \ldots * [1\ 1]}_{R\ \text{times}}, \tag{11.17}$$

or, written as an operator equation,

$$\mathcal{B}^R = \underbrace{\mathcal{B}\mathcal{B}\ldots\mathcal{B}}_{R\ \text{times}}. \tag{11.18}$$

Some examples of the resulting filter masks are:

$$\begin{aligned}
\boldsymbol{B}^2 &= 1/4\,[1\ 2\ 1] \\
\boldsymbol{B}^3 &= 1/8\,[1\ 3\ 3\ 1] \\
\boldsymbol{B}^4 &= 1/16\,[1\ 4\ 6\ 4\ 1] \\
\boldsymbol{B}^8 &= 1/256\,[1\ 8\ 28\ 56\ 70\ 56\ 28\ 8\ 1].
\end{aligned} \tag{11.19}$$

Because of symmetry, only the odd-sized filter masks are of interest.

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

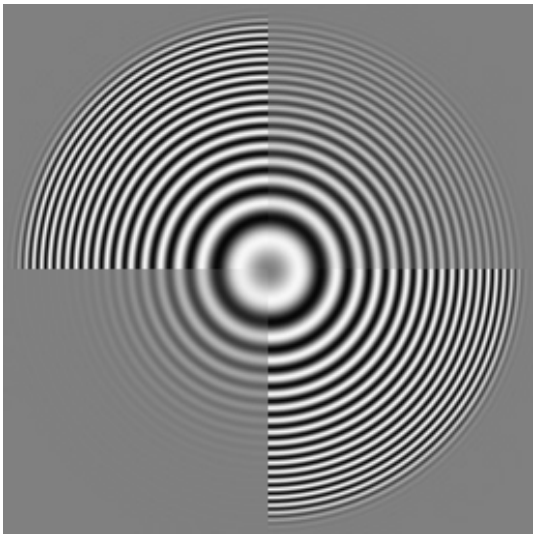

Figure 11.4: *Test of the smoothing with a $\mathcal{B}^4$ and $\mathcal{B}^{16}$ binomial filter using a test image with concentric sinusoidal rings.*

Pascal's triangle:

R	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

(11.20)

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

The computation of the transfer function of a binomial mask is also very simple since we only need to know the transfer function of $\mathcal{B}$. The transfer function of $\mathcal{B}^R$ is then given as the Rth power:

$$\boxed{\hat{b}^R(\tilde{k}) = \cos^R(\pi\tilde{k}/2),} \tag{11.21}$$

which can be approximated for small wave numbers by

$$\hat{b}^R(\tilde{k}) = 1 - \frac{R}{8}(\pi\tilde{k})^2 + \left(\frac{3R^2 - 2R}{384}\right)(\pi\tilde{k})^4 + O(\tilde{k}^6). \tag{11.22}$$

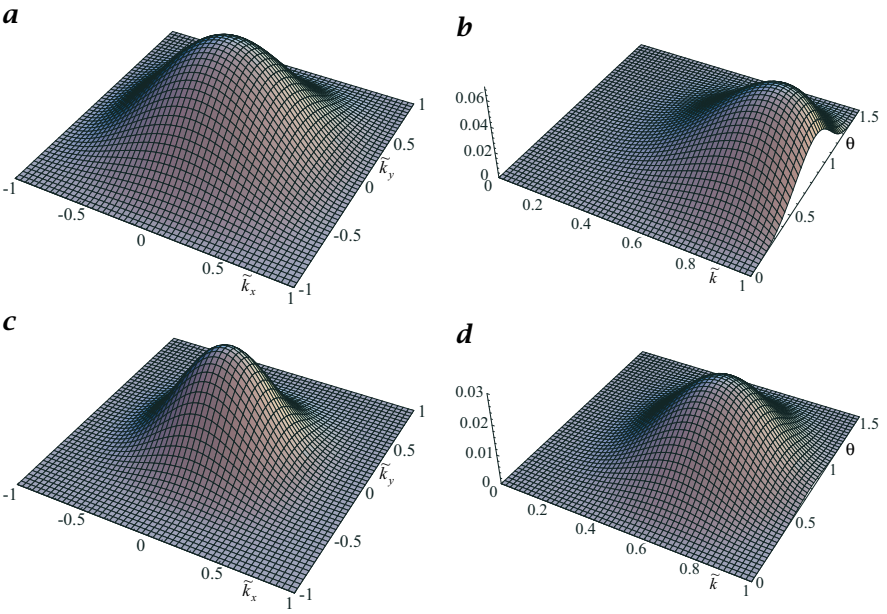

Figure 11.5: *Transfer function of two-dimensional binomial filters:* **a** $\mathcal{B}^2$*;* **b** *anisotropy* $\hat{B}^2(\tilde{k},\theta) - \hat{B}^2(\tilde{k},0)$ *in a* (k,θ) *diagram;* **c** $\mathcal{B}^4$*;* **d** *anisotropy for* $\mathcal{B}^4$ *as in* **b**.

The graphical representation of the transfer function in Fig. 11.1b reveals that binomial filters are much better smoothing filters than box filters. The transfer function decreases monotonically and approaches zero at the largest wave number. The smallest mask, $\mathcal{B}^2$, has a halfwidth of $\tilde{k}/2$. This is a periodic structure that is sampled four times per wavelength. For larger masks, both the transfer function and the filter masks approach the Gaussian distribution with an equivalent variance. Larger masks result in smaller half-width wave numbers according to the *uncertainty relation* (Section 2.3.4).

11.4.3 2-D Binomial Filter

Two-dimensional binomial filters can be composed from a horizontal and a vertical 1-D filter:

$$\mathcal{B}^R = \mathcal{B}_x^R \mathcal{B}_y^R. \tag{11.23}$$

The smallest mask of this kind is a 3×3-binomial filter ($R = 2$):

$$\boldsymbol{B}^2 = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{11.24}$$

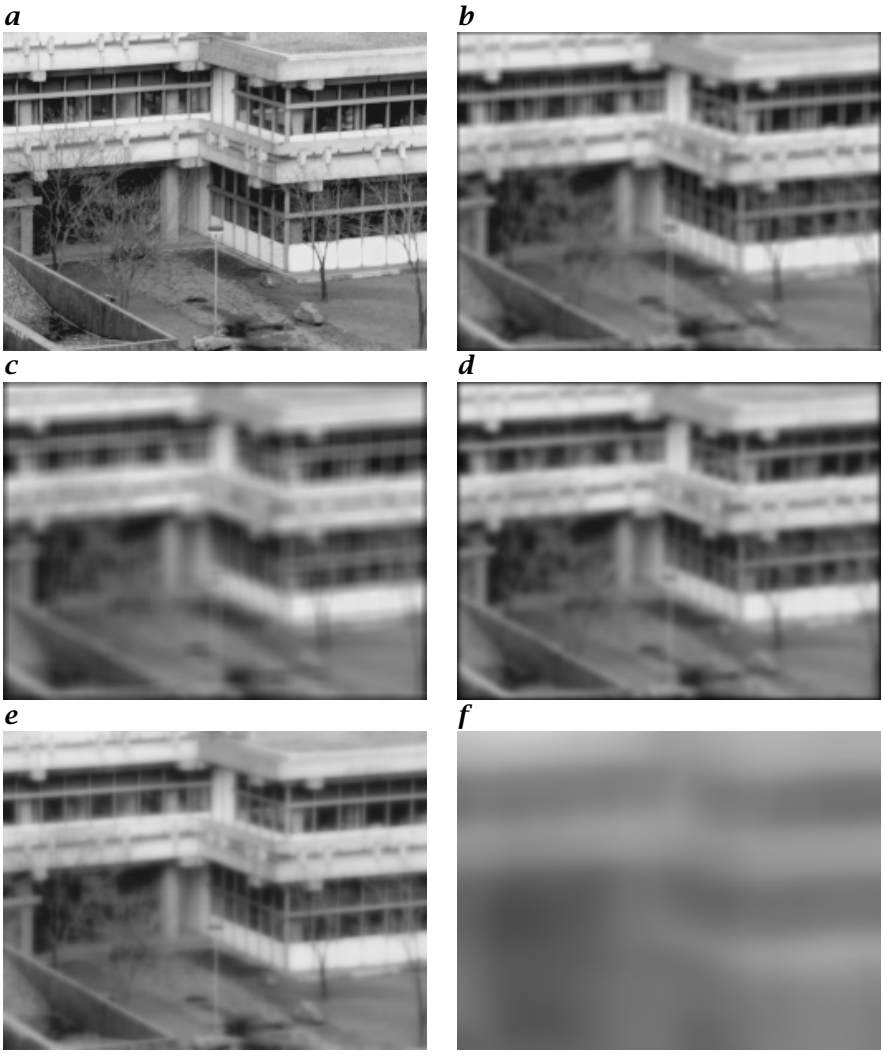

Figure 11.6: *Application of smoothing filters:* ***a*** *original image;* ***b*** 5×5 *box filter;* ***c*** 9×9 *box filter;* ***d*** 17×17 *binomial filter* ($\mathcal{B}^{16}$)*; a set of recursive filters Eq. (11.38) running in horizontal and vertical directions;* ***e*** $R = 2$*;* ***f*** $R = 16$.

The transfer function of the 2-D binomial filter $\mathcal{B}^R$ with $(R+1) \times (R+1)$ coefficients is easily derived from the transfer functions of the 1-D filters Eq. (11.21) as

$$\hat{b}^R = \hat{b}_y^R \hat{b}_x^R = \cos^R(\pi \tilde{k}_y/2) \cos^R(\pi \tilde{k}_x/2), \tag{11.25}$$

and correspondingly for a 3-D filter as

$$\hat{b}^R = \hat{b}_z^R \hat{b}_y^R \hat{b}_x^R = \cos^R(\pi \tilde{k}_z/2) \cos^R(\pi \tilde{k}_y/2) \cos^R(\pi \tilde{k}_x/2). \tag{11.26}$$

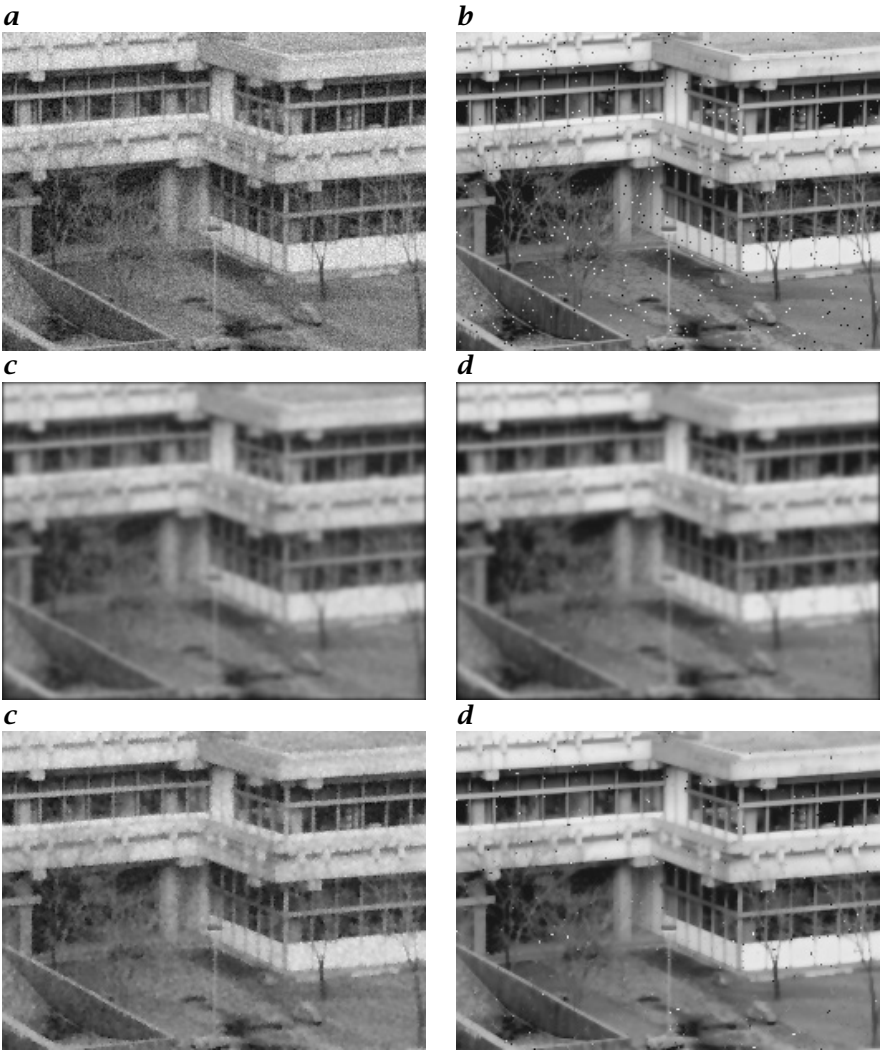

Figure 11.7: *Suppression of noise with smoothing filters:* ***a*** *image from Fig. 11.6a with Gaussian noise;* ***b*** *image with binary noise;* ***c*** *image* ***a*** *and* ***d*** *image* ***b*** *filtered with a* 9×9 *binomial filter* ($\mathcal{B}^8$)*;* ***e*** *image* ***a*** *and* ***f*** *image* ***b*** *filtered with a* 3×3 *median filter (Section 11.6.1).*

The transfer functions of $\mathcal{B}^2$ and $\mathcal{B}^4$ are shown in Fig. 11.5. Already the small 3×3 filter is remarkably isotropic. Larger deviations from the circular contour lines can only be recognized for larger wave numbers, when the transfer function has dropped to 0.3 (Fig. 11.5a). This property can be shown by expanding Eq. (11.25) in a Taylor series using cylindrical

coordinates $\tilde{\boldsymbol{k}} = [\tilde{k}, \theta]^T$:

$$\hat{b}^R \approx 1 - \frac{R}{8}(\pi\tilde{k})^2 + \frac{2R^2 - R}{256}(\pi\tilde{k})^4 - \frac{R\cos 4\theta}{768}(\pi\tilde{k})^4. \tag{11.27}$$

Only the second-order term is isotropic. In contrast, the fourth-order term contains an anisotropic part which increases the transfer function in the direction of the diagonals (Fig. 11.5a). A larger filter (larger R) is less anisotropic as the isotropic term with $\tilde{k}^4$ increases quadratically with R while the anisotropic term with $\tilde{k}^4 \cos 4\theta$ increases only linearly with R. Already the 5×5 filter (Fig. 11.5b) is remarkably isotropic. The insignificant anisotropy of the binomial filters also becomes apparent when applied to the test image in Fig. 11.4.

11.4.4 Evaluation

Figure 11.6b, c show smoothing with two different binomial filters. We observe that the edges get blurred. Fine structures as in the branches of the tree are lost. Smoothing suppresses noise. Binomial filters can reduce the noise level of zero-mean *Gaussian noise* (Section 3.4.2) considerably but only at the price of blurred details (Fig. 11.7a, c). *Binary noise* (also called *impulse noise*), which causes wrong gray values for a few randomly distributed pixels (Fig. 11.7b) (for instance due to transmission errors), is suppressed only poorly by linear filters. The images are blurred, but the error caused by the binary noise is not eliminated but only distributed.

11.4.5 Fast Computation

We close our consideration of binomial filters with some remarks on fast algorithms. A direct computation of a $(R + 1) \times (R + 1)$ filter mask requires $(R + 1)^2$ multiplications and $(R + 1)^2 - 1$ additions. If we decompose the binomial mask into elementary smoothing masks $1/2\,[1\ 1]$ and apply this mask in horizontal and vertical directions R times each, we only need $2R$ additions. All multiplications can be handled much more efficiently as shift operations. For example, the computation of a 17×17 binomial filter requires only 32 additions and some shift operations compared to 289 multiplications and 288 additions needed for the direct approach.

11.5 Efficient Large-Scale Averaging

Despite the efficient implementation of binomial smoothing filters $\mathcal{B}^r$ by cascaded convolution with $\mathcal{B}$, the number of computations increases

dramatically for smoothing masks with low cutoff wave numbers, because the standard deviation of the filters is proportional to the square root of R according to Eq. (3.43):

$$\sigma = \sqrt{R/4}. \tag{11.28}$$

Let us consider a smoothing operation over a circle with a radius of about only 1.73 pixels, corresponding to a variance $\sigma^2 = 3$. According to Eq. (11.28) we need to apply $\mathcal{B}^{12}$ which — even in an efficient separable implementation — requires 24 (36) additions and 2 (3) shift operations for each pixel in a 2-D (3-D) image. If we want to smooth over the double distance ($\sigma^2 = 12$, radius ≈ 3.5, $\mathcal{B}^{48}$) the number of additions quadruples to 96 (144) per pixel in 2-D (3-D) space.

11.5.1 Multistep Averaging

The problem of slow large-scale averaging originates from the small distance between the pixels averaged in the elementary $\boldsymbol{B} = 1/2\,[1\ 1]$ 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 ($\sigma \cdot \sqrt{2}$):

$$\boldsymbol{B}_{x+y} = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B}_{x-y} = \frac{1}{4}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{11.29}$$

or, with double step width along axes ($\sigma \cdot 2$) and in three dimensions,

$$\boldsymbol{B}_{2x} = \frac{1}{4}\,[1\ 0\ 2\ 0\ 1], \quad \boldsymbol{B}_{2y} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{B}_{2z} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}_z. \tag{11.30}$$

The subscripts in these masks denote the stepping width and coordinate direction. $\boldsymbol{B}_{x+y}$ averages the gray values at two neighboring pixels in the direction of the main diagonal. $\boldsymbol{B}_{2x}$ 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

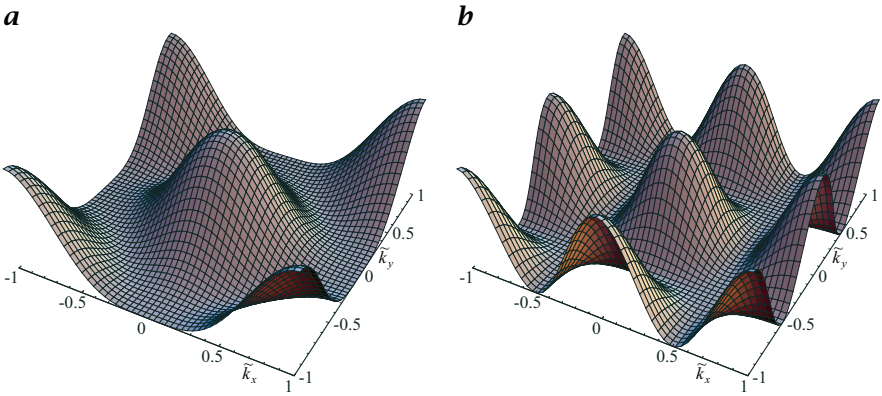

Figure 11.8: *Transfer function of the binomial mask applied* ***a*** *in the diagonal direction ($\mathcal{B}^2_{x+y}\mathcal{B}^2_{x-y}$) and* ***b*** *with double step width in axis directions ($\mathcal{B}^2_{2x}\mathcal{B}^2_{2y}$).*

on a grid with a doubled grid constant in the spatial domain. Hence, the reciprocal grid in the wave number space has half the grid width and the transfer function is periodically replicated once in both directions (Fig. 11.8). Generally, the zero lines of the transfer function of masks with larger step width reflect this reciprocal grid. For convolution with two neighboring pixels in the direction of the two diagonals, the reciprocal grid is turned by 45°. The grid constant of the reciprocal grid is a factor $\sqrt{2}$ smaller than that of the original grid.

Used individually, these filters are not of much help. But we can use them in cascade, starting with directly neighboring pixels. Then the zero lines of the transfer functions, which lie differently for each pixel distance, efficiently force the transfer function close to zero for large wave number ranges.

Cascaded multistep binomial filtering leads to a significant performance increase for large-scale smoothing. For normal separable binomial filtering, the number of computations is proportional to $\sigma^2 (O(\sigma^2))$. For multistep binomial filtering it depends only logarithmically on σ $(O(\mathrm{ld}\sigma^2))$ if a cascade of filter operations with recursive step width doubling is performed:

$$\underbrace{\mathcal{B}^R_{2^{S-1}x} \cdots \mathcal{B}^R_{8x}\mathcal{B}^R_{4x}\mathcal{B}^R_{2x}\mathcal{B}^R_{x}}_{S \text{ times}}. \tag{11.31}$$

Such a mask has the standard deviation

$$\sigma^2 = \underbrace{R/4 + R + 4R + \ldots + 4^{S-1}R}_{S \text{ times}} = \frac{R}{12}(4^S - 1) \tag{11.32}$$

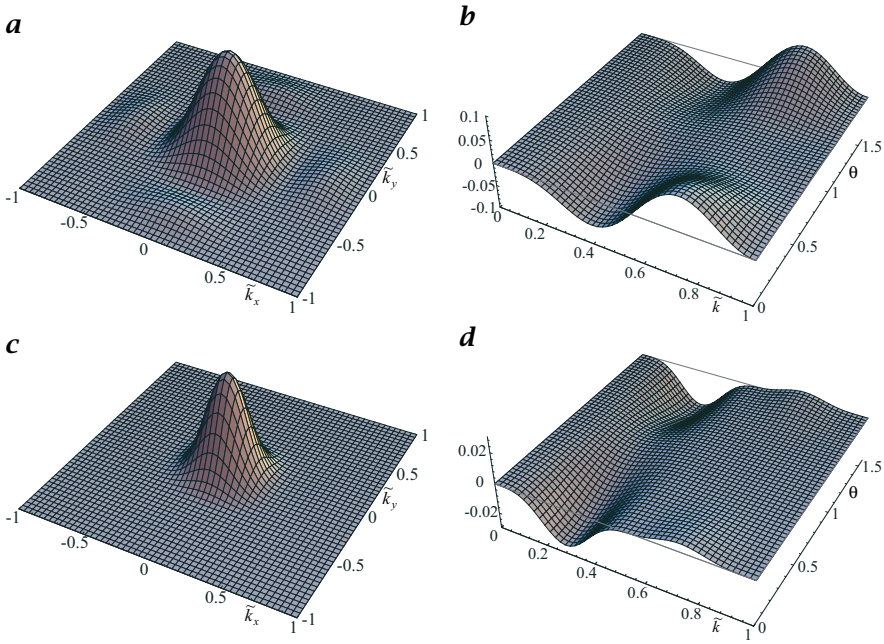

Figure 11.9: *Transfer function of cascaded multistep binomial filters and their anisotropy:* $\boldsymbol{a}$ $\mathcal{B}_2^2\mathcal{B}_1^2$, $\boldsymbol{b}$ $\hat{B}_2^2\hat{B}_1^2(\tilde{k},\theta)-\hat{B}_2^2\hat{B}_1^2(\tilde{k},0)$, $\boldsymbol{c}$ $\mathcal{B}_2^4\mathcal{B}_1^4$, $\boldsymbol{d}$ $\hat{B}_2^4\hat{B}_1^4(\tilde{k},\theta)-\hat{B}_2^4\hat{B}_1^2(\tilde{k},0)$. *The anisotropy is shown in polar coordinates* $(\tilde{k},\theta)$ *as the deviation from the transfer function in the* x *direction.*

and the transfer function

$$\prod_{s=0}^{S-1}\cos^R(2^{s-1}\pi\tilde{k}). \tag{11.33}$$

Thus, for S steps only RS additions are required while the standard deviation grows exponentially with $\approx\sqrt{R/12}\cdot 2^S$.

With the parameter R, we can adjust the degree of isotropy and the degree of residual inhomogeneities in the transfer function. A very efficient implementation is given by using R = 2 ($\boldsymbol{B}^2 = 1/4[1\ 2\ 1]$ in each direction). However the residual side peaks at high wave numbers with maximal amplitudes up to 0.08 are still significant disturbances (Fig. 11.9a, b, Fig. 11.10a, b).

With the next larger odd-sized masks (R = 4, $\boldsymbol{B}^4 = 1/16[1\ 4\ 6\ 4\ 1]$ in each direction) these residual side peaks at high wave numbers are suppressed well below 0.005 (Fig. 11.9c, d, Fig. 11.10c, d). This is about the relative resolution of 8-bit images and should therefore be sufficient for most applications. With still larger masks, they could be suppressed even further. Figure 11.11 shows the first four steps of multistep aver-

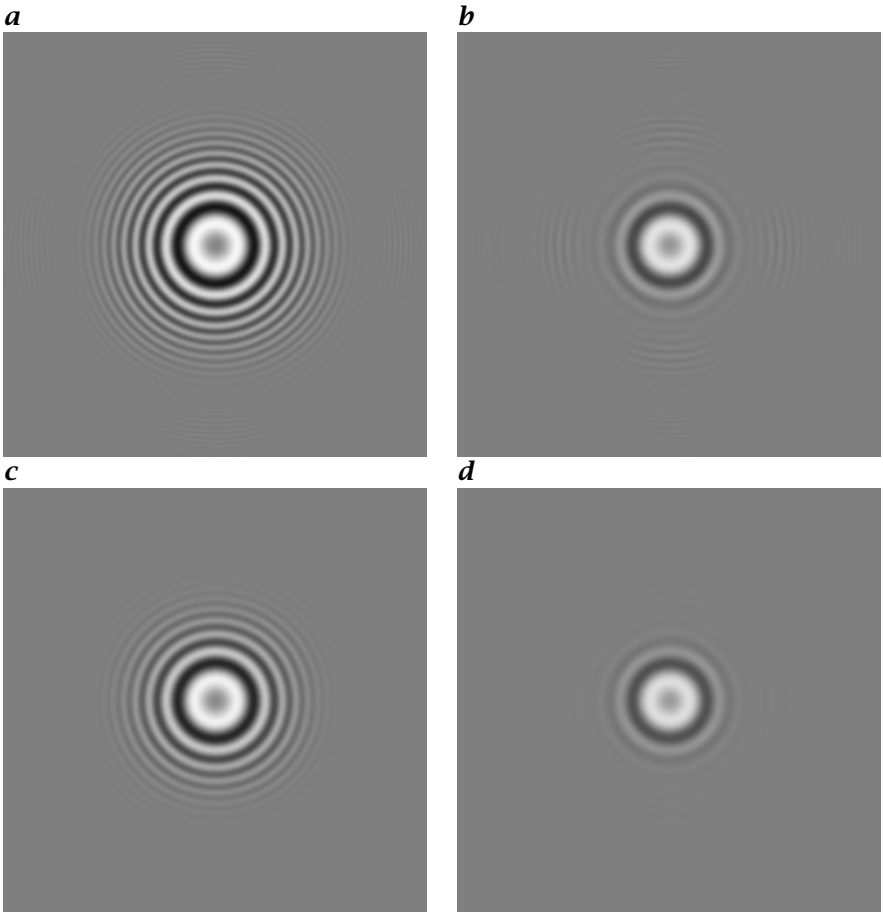

Figure 11.10: *Cascaded multistep averaging with step width doubling according to Eq. (11.31), applied to the ring test pattern:* **a** $\mathcal{B}_2^2\mathcal{B}_1^2$, **b** $\mathcal{B}_4^2\mathcal{B}_2^2\mathcal{B}_1^2$, **c** $\mathcal{B}_2^4\mathcal{B}_1^4$, *and* **d** $\mathcal{B}_4^4\mathcal{B}_2^4\mathcal{B}_1^4$.

aging with the $\boldsymbol{B}^4$ mask, illustrating how quickly the smoothing reaches large scales.

11.5.2 Multigrid Averaging

Multistep cascaded averaging can be further enhanced by converting it into a multiresolution technique. The idea of multigrid smoothing is very simple. When a larger-step mask is involved, this operation can be applied on a correspondingly coarser grid. This means that the last operation before using the larger-step mask needs to compute the convolution only at the grid points used by the following coarser grid operator.

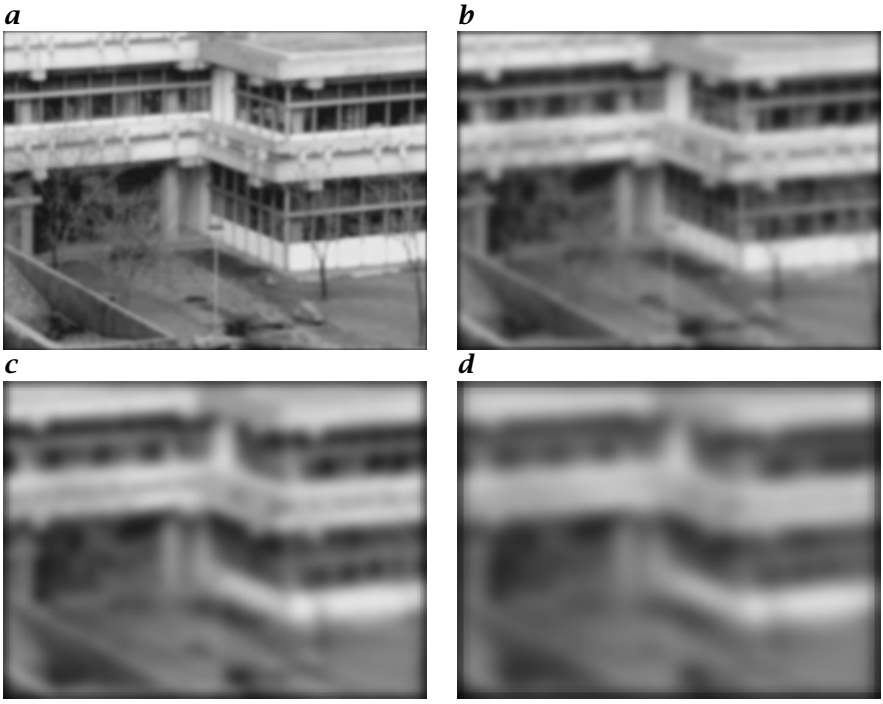

Figure 11.11: *Cascaded multistep averaging with step width doubling according to Eq. (11.31), applied to image Fig. 11.6a with* ***a*** *one,* ***b*** *two,* ***c*** *three, and* ***d*** *four steps using the* $\mathbf{B}^4$ *filter.*

This sampling procedure is denoted by a special syntax in the operator index. $\mathcal{O}_{x|2}$ means: Apply the operator in the x direction and advance the mask two pixels in the x direction. Thus, the output of the filter operator has only half as many pixels in the x direction as the input.

Multigrid smoothing makes the number of computations essentially independent of the standard deviation of the smoothing mask. We again consider a sequence of 1-D binomial filters:

$$\underbrace{\mathcal{B}^R_{x\downarrow 2} \cdots \mathcal{B}^R_{x\downarrow 2}\mathcal{B}^R_{x\downarrow 2}}_{S \text{ times}}.$$

Since $\mathcal{B}^R_{x|2}$ takes R operations, the operator sequence takes

$$R\sum_{s=1}^{S}\frac{1}{2^{s-1}} = R\left(1-\frac{1}{2^{S-1}}\right) < 2R$$

As for the multistep approach, Eq. (11.32), the standard deviation of the operator sequence is

$$\sigma^2 = \frac{R}{12}(4^S - 1). \tag{11.34}$$

Thus, smoothing to any degree takes not more than twice as many operations as smoothing at the first step! As for multistep binomial filters, the standard deviation grows by a factor of two. Also — as long as $\hat{B}^R(\tilde{k}) = 0 \quad \forall \tilde{k} \geq 1/2$ — the transfer functions of the filters are the same as for the multistep filters.

11.5.3 *Recursive Averaging*

A totally different approach to large-scale averaging is given by recursive filtering introduced in Section 4.5. The recursion essentially gives a convolution filter an infinite point spread function. The basic advantage of recursive filters is that they can easily be "tuned", as we have demonstrated with the simple lowpass filter in Section 4.5.5. In this section, the focus is on the design of averaging filters that meet the criteria we discussed earlier in Section 11.2, especially the zero-shift property (Section 11.2.1) that is not met by causal recursive filters.

Basically, recursive filters work the same as non-recursive filters. In principle, we can replace any recursive filter with a non-recursive filter whose filter mask is identical to the point spread function of the recursive filter. The real problem is the design of the recursive filter, i. e., the determination of the filter coefficients for a desired transfer function.

While the theory of one-dimensional recursive filters is standard knowledge in digital signal processing (see, for example, Oppenheim and Schafer [148]), the design of two-dimensional filters is still not adequately understood. The main reason is the fundamental difference between the mathematics of one- and higher-dimensional z-transforms and polynomials [124].

Despite these theoretical problems, recursive filters can be applied successfully in digital image processing. In order to avoid the filter design problems, we will use only very simple recursive filters which are easily understood and compose them to more complex filters, similar to the way we constructed the class of binomial filters from the elementary smoothing mask $1/2\,[1\ 1]$. In this way we will obtain a class of recursive filters that may not be optimal from the point of view of filter design but are useful in practical applications.

In the first composition step, we combine causal recursive filters to symmetric filters. We start with a general one-dimensional recursive filter with the transfer function

$$^{+}\hat{A} = a(\tilde{k}) + \mathrm{i}b(\tilde{k}). \tag{11.35}$$

The index + denotes the run direction of the filter in the positive coordinate direction. The transfer function of the same filter but running in the opposite direction is

$$^{-}\hat{A} = a(\tilde{k}) - \mathrm{i}b(\tilde{k}). \tag{11.36}$$

Only the sign of the imaginary part of the transfer function changes, as it corresponds to the odd part of the point spread function, while the real part corresponds to the even part.

We now have two possible ways to combine the forward and backward running filters into symmetric filters useful for averaging:

$$\begin{aligned} &\text{addition} && \hat{A} = \frac{1}{2}\left[{}^{+}\hat{A} + {}^{-}\hat{A}\right] && = a(\tilde{k}) \\ &\text{multiplication} && \hat{A} = {}^{+}\hat{A}\,{}^{-}\hat{A} && = a^2(\tilde{k}) + b^2(\tilde{k}). \end{aligned} \tag{11.37}$$

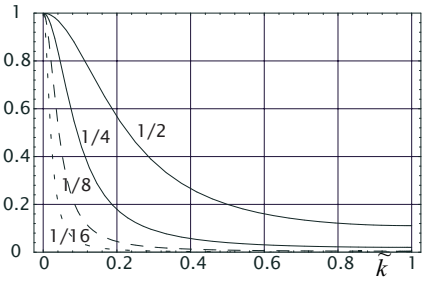

Figure 11.12: *Transfer function of the recursive lowpass filter Eq. (11.41) for different values of α = 1/2, 1/4, 1/8, and 1/16.*

Both techniques yield real transfer functions and thus even filters with zero shift that are suitable for averaging.

As the elementary recursive smoothing filter, we use the two-element lowpass filter we have already studied in Section 4.5.5:

$$^{\pm}\mathcal{A}_x: \; G'_{mn} = G'_{m,n\mp1} + \alpha(G_{mn} - G'_{m,n\mp1}) \quad \text{with} \quad 0 \le \alpha \le 1 \tag{11.38}$$

with the impulse response

$$(^{\pm}A_x)_{m,n} = \begin{cases} \alpha(1-\alpha)^n & n > 0, m = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11.39}$$

The transfer function of this filter can easily be calculated by taking into account that the Fourier transform of Eq. (11.39) forms a *geometric series*:

$$^{\pm}\hat{A}_x(\tilde{k}) = \frac{\alpha}{1-(1-\alpha)\exp(\mp\mathrm{i}\pi\tilde{k})}. \tag{11.40}$$

This relation is valid only approximately, since we broke off the infinite sum in Eq. (11.39) at $n = N-1$ because of the limited size of the image.

Consecutive filtering with a left and right running filter corresponds to a multiplication of the transfer functions

$$\hat{A}_x(\tilde{k}) = {}^{+}\hat{A}_x(\tilde{k})\;{}^{-}\hat{A}_x(\tilde{k}) \approx \frac{\alpha^2}{\alpha^2 + 2(1-\alpha)(1-\cos(\pi\tilde{k}))}. \tag{11.41}$$

The transfer function shows the characteristics expected for a lowpass filter (Fig. 11.12). At $\tilde{k} = 0$, $\hat{A}_x(\tilde{k}) = 1$; for small $\tilde{k}$, the transfer function falls off in proportion to $\tilde{k}^2$,

$$\hat{A}_x \approx 1 - \frac{1-\alpha}{\alpha^2}(\pi\tilde{k})^2 \quad \tilde{k} \ll 1, \tag{11.42}$$

and has a half-value wave number $\tilde{k}_c$ ($\hat{A}_x(\tilde{k}_c) = 1/2$) of

$$\tilde{k}_c \approx \frac{1}{\pi}\arcsin\frac{\alpha}{\sqrt{2(1-\alpha)}} \approx \frac{\alpha}{\sqrt{2}\pi}, \tag{11.43}$$

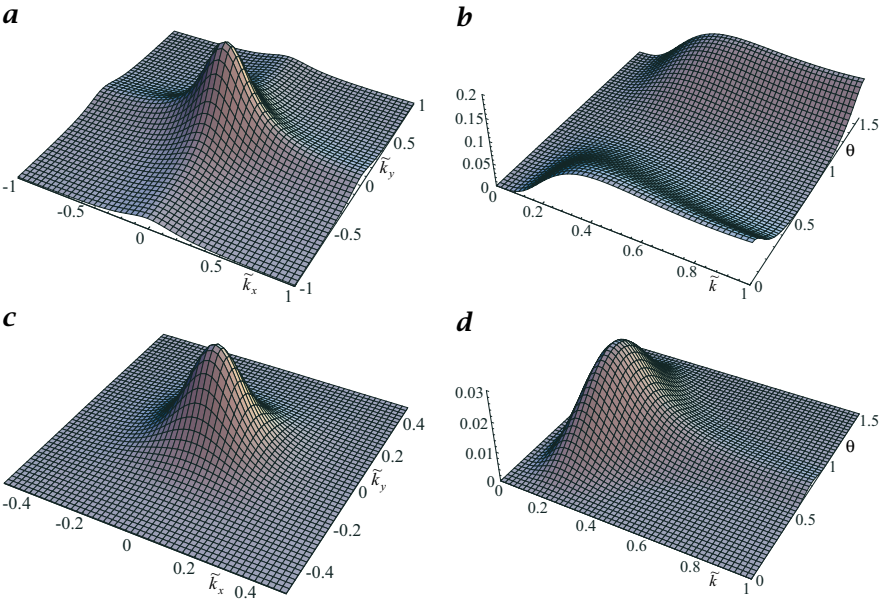

Figure 11.13: *Transfer functions of two-dimensional recursive lowpass filters:* ***a*** $\mathcal{A}$ *with* $\alpha = 1/2$, ***b*** *anisotropy of* ***a****:* $\hat{A}(k,\theta) - \hat{A}(k,\pi/4)$, ***c*** $\mathcal{A}'$ *with* $\alpha = 1/2$, *and* ***d*** *anisotropy of* ***c****:* $\hat{A}'(k,\theta) - \hat{A}'(k,0)$.

where the last approximation is only valid for $\alpha << 1$. At the highest wave number, $\tilde{k} = 1$, the transfer function has dropped off to

$$\hat{A}_x(1) \approx \frac{\alpha^2}{4(1-\alpha)+\alpha^2}. \tag{11.44}$$

In contrast to binomial filters, it is not exactly zero, but sufficiently small even for small values of α (Fig. 11.12).

Two-dimensional filters can be composed from one-dimensional filters running in the horizontal and vertical directions:

$$\mathcal{A} = \mathcal{A}_x\mathcal{A}_y = {}^+\mathcal{A}_x{}^-\mathcal{A}_x{}^+\mathcal{A}_y{}^-\mathcal{A}_y. \tag{11.45}$$

This filter (Fig. 11.13a, b) is significantly less isotropic than binomial filters (Fig. 11.5). High wave numbers are attenuated much less in coordinate directions than in the other directions. However, recursive filters have the big advantage that the computational effort does not depend on the degree of averaging. With the simple first-order recursive filter, we can select the degree of averaging with an appropriate choice of the filter parameter α (Eq. (11.43)). The isotropy of recursive filters can be further improved by running additional filters along the diagonals:

$$\mathcal{A}' = \mathcal{A}_x\mathcal{A}_y\mathcal{A}_{x-y}\mathcal{A}_{x+y}. \tag{11.46}$$

The subscripts $x-y$ and $x+y$ denote the main and second diagonal, respectively. The transfer function of such a filter is shown in Fig. 11.13c, d.

In contrast to non-recursive filters, the computational effort does not depend on the cut-off wave number. If $\alpha = 2^{-l}$ in Eq. (11.38), the filter can be computed without any multiplication:

$$G'_{mn} = \left[G'_{m,n\pm1} \cdot 2^l - G'_{m,n\pm1} + G_{mn} \right] \cdot 2^{-l}, \quad l > 1. \tag{11.47}$$

The two-dimensional filter $\mathcal{A}$ needs only 8 additions and shift operations per pixel, while the $\mathcal{A}'$ filter, running in 4 directions, needs twice as many operations. However, this is not more efficient than the multigrid approach with binomial masks (Section 11.5.2), which is a much better isotropic filter.

11.6 *Nonlinear Averaging*

The linear averaging filters discussed so far blur edges. Even worse, if the mask of the smoothing operator crosses an object edge it contains pixels from both the object and the background, giving a meaningless result from the filter. The same is true if averages are performed when a certain number of pixels in an image show erroneous values, e. g., because of a transmission error. The question, therefore, is whether it is possible to perform an averaging that does not cross object boundaries or that ignores certain pixels. Such a procedure can only be applied, of course, if we have already detected the edges or any distorted pixel.

In this section, we discuss three types of nonlinear averaging filter: the classical median filter (Section 11.6.1); weighted averaging, also known as normalized convolution (Section 11.6.2); and steerable averaging (Section 11.6.3), where we control the direction and/or degree of averaging with the local content of the neighborhood.

11.6.1 *Median Filter*

Linear filters effectively suppress Gaussian noise but perform very poorly in case of binary noise (Fig. 11.7). Using linear filters that weigh and sum up, we assume that each pixel carries some useful information. Pixels distorted by transmission errors have lost their original gray value. Linear smoothing does not eliminate this information but carries it on to neighboring pixels. Thus the appropriate operation to process such distortions is to detect these pixels and to eliminate them.

This is exactly what a *rank-value filter* does (Section 4.3). The pixels within the mask are sorted and one pixel is selected. In particular, the *median filter* selects the medium value. As binary noise completely changes the gray value, it is very unlikely that it will show the medium gray value in the neighborhood. In this way, the medium gray value of the neighborhood is used to restore the gray value of the distorted pixel.

The following examples illustrate the effect of a 1×3 median filter $\mathcal{M}$:

$$\mathcal{M}[\cdots\ 1\ 2\ 3\ 7\ 8\ 9\ \cdots] = [\cdots\ 1\ 2\ 3\ 7\ 8\ 9\ \cdots],$$

$$\mathcal{M}[\cdots\ 1\ 2\ 102\ 4\ 5\ 6\ \cdots] = [\cdots\ 1\ 2\ 4\ 5\ 5\ 6\ \cdots],$$

$$\mathcal{M}[\cdots\ 0\ 0\ 0\ 9\ 9\ 9\ \cdots] = [\cdots\ 0\ 0\ 0\ 9\ 9\ 9\ \cdots].$$

As expected, the median filter eliminates runaways. The two other gray value structures — a monotonically increasing ramp and an edge between two plateaus of constant gray value — are preserved. In this way a median filter effectively eliminates binary noise without significantly blurring the image (Fig. 11.7e). Gaussian noise is less effectively eliminated (Fig. 11.7f).

The most important deterministic properties of a one-dimensional $2N + 1$ median filter can be formulated using the following definitions.

- A *constant neighborhood* is an area with $N + 1$ equal gray values.
- An *edge* is a monotonically increasing or decreasing area between two constant neighborhoods.
- An *impulse* is an area of at most N points surrounded by constant neighborhoods with the same gray value.
- A *root* or *fix point* is a signal that is preserved under the median filter operation.

With these definitions, the deterministic properties of a median filter can be described very compactly:

- Constant neighborhoods and edges are fix points.
- Impulses are eliminated.

Iterative filtering of an image with a median filter results in an image containing only constant neighborhoods and edges. If only single pixels are distorted, a 3×3 median filter is sufficient to eliminate them. If clusters of distorted pixels occur, larger median filters must be used.

The statistical properties of the median filter can be illustrated with an image containing only constant neighborhoods, edges, and impulses. The impulse power spectrum of impulses is flat (*white noise*). As the median filter eliminates impulses, the power spectrum decreases homogeneously. The contribution of the edges to a certain wave number is not removed. This example also underlines the nonlinear nature of the median filter.

11.6.2 *Weighted Averaging*

In Section 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 independent 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 variance $w_n = 1/\sigma_n^2$. Then, an estimate of the mean value is given by

$$\overline{g} = \sum_{n=1}^{N} g_n/\sigma_n^2 \Bigg/ \sum_{n=1}^{N} 1/\sigma_n^2 \tag{11.48}$$

while the standard deviation of the mean is

$$\sigma_{\overline{g}}^2 = 1 \Bigg/ \sum_{n=1}^{N} 1/\sigma_n^2 \,. \tag{11.49}$$

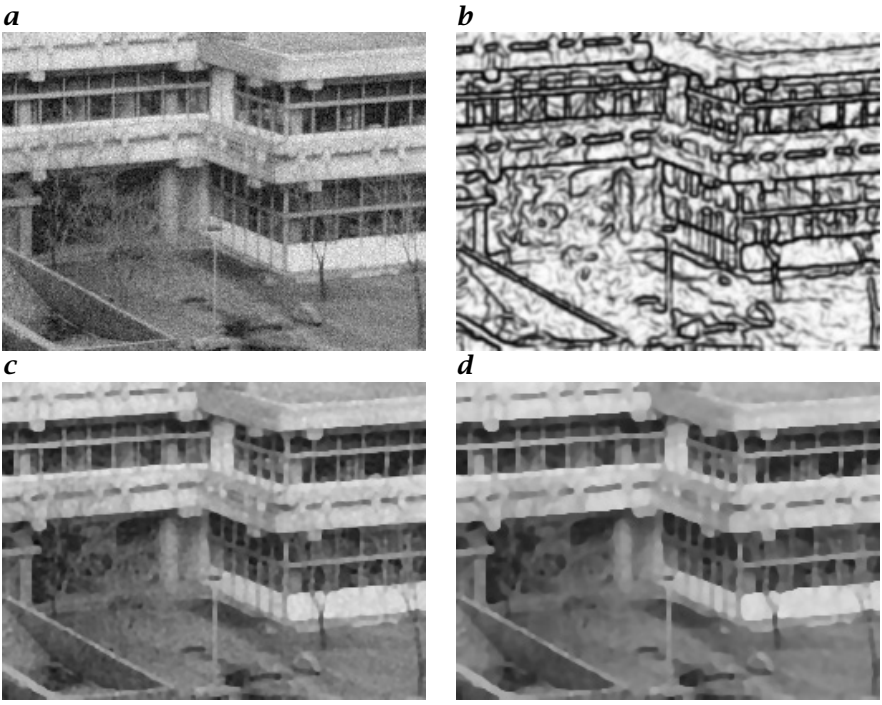

Figure 11.14: *Weighted averaging using the edge strength to hinder smoothing at edges:* ***a*** *image from Fig. 11.6a with added Gaussian noise;* ***b*** *weighting image after 5 convolutions; image after* ***c*** *two and* ***d*** *five normalized convolutions using a* $\mathcal{B}^2$ *binomial smoothing mask (compare with Fig. 11.7).*

The lower the statistical error of an individual data point, the higher is the weight in Eq. (11.48).

The application of *weighted averaging* to image processing is known as *normalized convolution* [64]. The averaging is now extended to a local neighborhood. Each pixel enters the convolution sum with a weighting factor associated with it. Thus, normalized convolution requires two images. One is the image to be processed, the other an image with the weighting factors.

By analogy to Eqs. (11.48) and (11.49), normalized convolution is defined by

$$\boxed{\boldsymbol{G}' = \frac{\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G})}{\boldsymbol{H} * \boldsymbol{W}},} \tag{11.50}$$

where $\boldsymbol{H}$ is any convolution mask, $\boldsymbol{G}$ the image to be processed, and $\boldsymbol{W}$ the image with the weighting factors. A normalized convolution with the mask $\boldsymbol{H}$ essentially transforms the set of the image $\boldsymbol{G}$ and the weighting image $\boldsymbol{W}$ into a new image $\boldsymbol{G}'$ and a new weighting image $\boldsymbol{W}' = \boldsymbol{H} * \boldsymbol{W}$ which can undergo further processing.

In this sense, normalized convolution is nothing complicated or special. It is just adequate consideration of pixels with spatially variable statistical errors.

"Standard" convolution can be regarded as a special case of normalized convolution. Then all pixels are assigned the same weighting factor and it is not required to use a weighting image, since the factor remains a constant.

The flexibility of normalized convolution is given by the choice of the weighting image. The weighting image is not necessarily associated with an error. It can be used to select and/or amplify pixels with certain features. In this way, normalized convolution becomes a versatile nonlinear operator.

As an example, Fig. 11.14 shows an noisy image that is filtered by normalized convolution using an weighting image that hinders smoothing at edges.

11.6.3 *Steerable Averaging*

The idea of *steerable filters* is to make the convolution mask dependent on the local image structure. This is a general concept which is not restricted to averaging but can be applied to any type of convolution process. The basic idea of steerable filters is as follows. A steerable filter has some freely adjustable parameters that control the filtering. These could be various properties such as the degree of smoothing, the direction of smoothing, or both. It is easy to write down a filter mask with adjustable parameters. We have done this already for recursive filters in Eq. (11.38) where the parameter α determines the degree of smoothing. However, it is not computationally efficient to convolve an image with masks that are different at every pixel. Then, advantage can no longer be taken of the fact that masks are separable.

An alternative approach is to seek a base of a few filters, and to use these filters to compute a set of filtered images. Then, these images are interpolated using adjustable parameters. In operator notation this reads

$$\boxed{\mathcal{H}(\alpha) = \sum_{p=1}^{P} f_p(\alpha)\mathcal{H}_p,} \tag{11.51}$$

where $\mathcal{H}_p$ is the pth filter and $f_p(\alpha)$ a scalar interpolation function of the steering parameter α. Two problems must be solved when using steerable filters. First, and most basically, it is not clear that such a filter base H_p exists at all. Second, the relation between the steering parameter(s) α and the interpolation coefficients f_p must be found. If the first problem is solved, we mostly get the solution to the second for free.

As an example, a directional smoothing filter is to be constructed with the following transfer function:

$$\hat{h}_{\theta_0}(k,\theta) = 1 - f(k)\cos^2(\theta - \theta_0). \tag{11.52}$$

In this equation, cylindrical coordinates (k,θ) are used in the Fourier domain. The filter in Eq. (11.52) is a *polar separable* filter with an arbitrary radial function $f(k)$. This radial component provides an arbitrary isotropic smoothing filtering.

The steerable angular term is given by $\cos^2(\theta - \theta_0)$. Structures oriented in the direction θ_0 remain in the image, while those perpendicular to θ_0 are completely filtered out. The angular width of the directional smoothing filter is $\pm 45^\circ$.

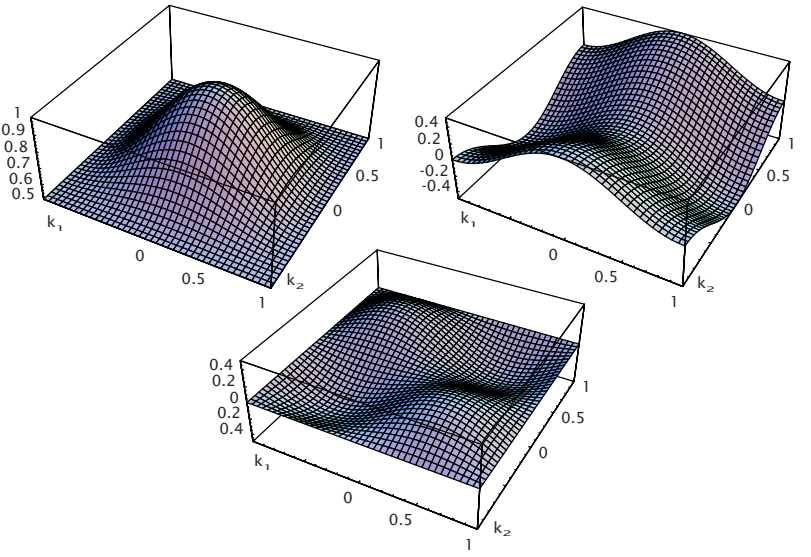

Figure 11.15: *Transfer functions for the three base filters for directional smoothing according to Eq. (11.56).*

We separate the cosine function in Eq. (11.52) into trigonometric functions that depend only on either θ or the steering angle θ_0 and obtain

$$\hat{h}_{\theta_0}(k,\theta) = 1 - \frac{1}{2} f(k)\,[1 + \cos(2\theta_0)\cos(2\theta) + \sin(2\theta_0)\sin(2\theta)] \tag{11.53}$$

with the base filters

$$\hat{h}_1 = 1 - \frac{1}{2} f(k), \quad \hat{h}_2 = -\frac{1}{2} f(k)\cos(2\theta), \quad \hat{h}_3 = -\frac{1}{2} f(k)\sin(2\theta) \tag{11.54}$$

and the interpolation functions

$$f_1(\theta_0) = 1, \quad f_2(\theta_0) = \cos(2\theta_0), \quad f_3(\theta_0) = \sin(2\theta_0). \tag{11.55}$$

Thus three base filters are required. The filter $\hat{h}_1$ is an isotropic smoothing filter, the other two are directional filters. Although the equations for this family of steerable directional smoothing filter are simple, it is not easy to implement polar separable base filters because they are not Cartesian separable and, thus, require careful optimization.

Nevertheless, it is even possible to implement this steerable smoothing filter with 3×3 base filters (Fig. 11.15). Because of symmetry properties of the transfer functions, we have not much choice to chose the filter coefficients and end up with the following three base filters:

$$\boldsymbol{H}_1 = \frac{1}{32}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 20 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad \boldsymbol{H}_2 = \frac{1}{32}\begin{bmatrix} 0 & -4 & 0 \\ 4 & 0 & 4 \\ 0 & -4 & 0 \end{bmatrix}, \quad \boldsymbol{H}_3 = \frac{1}{32}\begin{bmatrix} -2 & 0 & 2 \\ 0 & 0 & 0 \\ 2 & 0 & -2 \end{bmatrix}$$

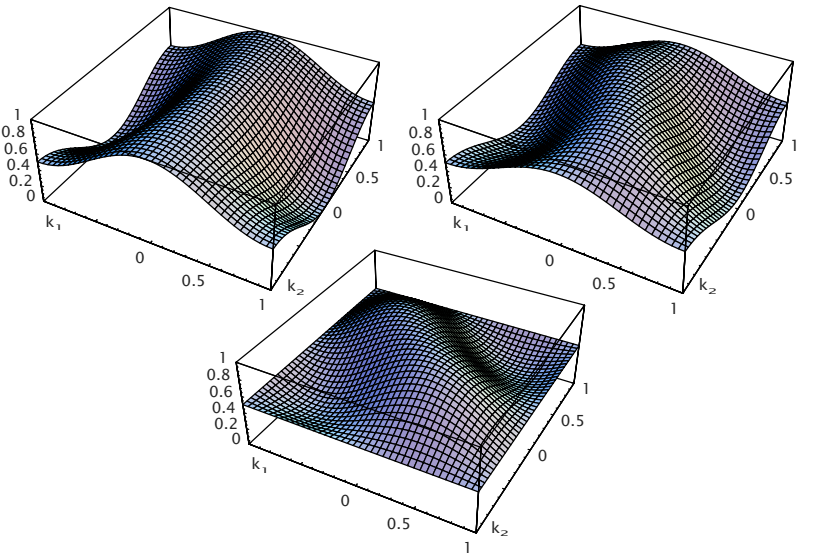

Figure 11.16: *Transfer functions for the steerable smoothing filter according to Eq. (11.53) using the base filters Eq. (11.56): smoothing in 0°, 22.5°, and 45°to the x axis.*

$$\begin{aligned}
\hat{h}_1 &= \frac{1}{2} + \frac{1}{2}\cos^2(\pi\tilde{k}_1/2)\cos^2(\pi\tilde{k}_2/2) &&\approx 1 - \frac{\pi^2\tilde{k}^2}{8}, \\
\hat{h}_2 &= \frac{1}{4}\left(\cos(\pi\tilde{k}_1) - \cos(\pi\tilde{k}_2)\right) &&\approx \frac{\pi^2\tilde{k}^2}{8}\cos(2\theta), \qquad (11.56)\\
\hat{h}_3 &= \frac{1}{8}\left(\cos(\pi(\tilde{k}_1+\tilde{k}_2)) - \cos(\pi(\tilde{k}_1-\tilde{k}_2))\right) &&\approx \frac{\pi^2\tilde{k}^2}{8}\sin(2\theta).
\end{aligned}$$

From Fig. 11.16 it is obvious that this simple implementation works well up to moderate wave numbers. At high wave number ($\tilde{k} > 0.5$), the directional filter does no longer work very well.

11.7 *Averaging in Multichannel Images*

At first glance, it appears that there is not much special about averaging of multichannel images: just apply the smoothing mask to each of the P channels individually:

$$\mathbf{G}' = \begin{bmatrix} \boldsymbol{G}'_1 \\ \boldsymbol{G}'_2 \\ \vdots \\ \boldsymbol{G}'_p \end{bmatrix} = \boldsymbol{H} * \mathbf{G} = \begin{bmatrix} \boldsymbol{H} * \boldsymbol{G}_1 \\ \boldsymbol{H} * \boldsymbol{G}_2 \\ \vdots \\ \boldsymbol{H} * \boldsymbol{G}_p \end{bmatrix}. \qquad (11.57)$$

This simple concept can also be extended to normalized convolution, discussed in Section 11.6.2. If the same smoothing kernel is applied to all components, it

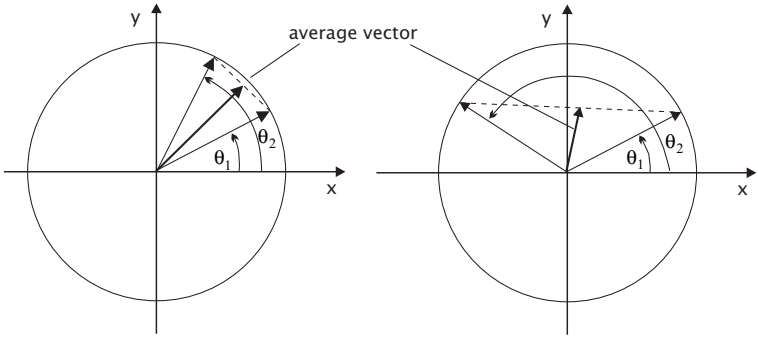

Figure 11.17: *Averaging of a cyclic quantity represented as a normal vector* $\bar{\boldsymbol{n}}_\theta = [\cos\theta, \sin\theta]^T$ *on the unit vector. The average vector* $(\bar{\boldsymbol{n}}_{\theta_1} + \bar{\boldsymbol{n}}_{\theta_2})/2$ *points in the correct direction* $(\theta_1 + \theta_2)/2$ *but its magnitude decreases with the difference angle.*

is sufficient to use one common weighting image that can be appended as the $(P+1)$th component of the multicomponent image.

$$\begin{bmatrix} \boldsymbol{G}'_1 \\ \boldsymbol{G}'_2 \\ \vdots \\ \boldsymbol{G}'_P \\ \boldsymbol{W}' \end{bmatrix} = \begin{bmatrix} (\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G}_1))/(\boldsymbol{H} * \boldsymbol{W}) \\ (\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G}_2))/(\boldsymbol{H} * \boldsymbol{W}) \\ \vdots \\ (\boldsymbol{H} * (\boldsymbol{W} \cdot \boldsymbol{G}_P))/(\boldsymbol{H} * \boldsymbol{W}) \\ \boldsymbol{H} * \boldsymbol{W} \end{bmatrix} \tag{11.58}$$

A special case of multicomponent images is given when they represent features that can be mapped to angular coordinates. Typically, such features include the direction of an edge or the phase of a periodic signal. Features of this kind are cyclic and cannot be represented well as Cartesian coordinates.

Also, they cannot be averaged in this representation. Imagine an angle of +175° and –179°. The mean angle is 178°, since –179° = 360° – 179° = 181° is close to 175° and not (175° – 179°) / 2 = –2°.

Circular features such as angles are, therefore, better represented as unit vectors in the form $\bar{\boldsymbol{n}}_\theta = [\cos\theta, \sin\theta]^T$.

In this representation, they can be averaged correctly as illustrated in Fig. 11.17. The average vector points to the correct direction but its magnitude is generally smaller than 1:

$$(\bar{\boldsymbol{n}}_{\theta_1} + \bar{\boldsymbol{n}}_{\theta_2})/2 = \begin{bmatrix} \cos[(\theta_1 + \theta_2)/2] \\ \sin[(\theta_1 + \theta_2)/2] \end{bmatrix} \cos[(\theta_2 - \theta_1)/2]. \tag{11.59}$$

For an angle difference of 180°, the average vector has zero magnitude. The decrease of the magnitude of the average vector has an intuitive interpretation. The larger the scatter of the angle is, the less is the certainty of the average value. Indeed, if all directions are equally probable, the sum vector vanishes, while it grows in length when the scatter is low.

These considerations can be extended to the averaging of circular features. To this end we set the magnitude of the vector equal to the certainty of the quantity that is represented by the angle of the vector. In this way, short vectors add little and long vectors add more to the averaging procedure.

This is a very attractive form of weighted convolution since in contrast to normalized convolution (Section 11.6.2) it requires no time-consuming division. Of course, it works only with features that can be mapped adequately to an angle.

Finally, we consider a measure to characterize the scatter in the direction of vectors. Figure 11.17 illustrates that for low scatter the sum vector is only slightly lower than the sum of the magnitudes of the vector. Thus, we can define an angular coherence measure as

$$c = \frac{|\boldsymbol{H} * \mathbf{G}|}{|\mathbf{G}|}, \tag{11.60}$$

where $\boldsymbol{H}$ is an arbitrary smoothing convolution operator. This measure is one if all vectors in the neighborhood covered by the convolution operator point in the same direction and zero if they are equally distributed. This definition of a coherence measure works not only in two-dimensional but also in higher-dimensional vector spaces. In one-dimensional vector spaces (scalar images), the coherency measure is, of course, always one.

11.8 *Exercises*

11.1: Box filters and binomial filters

Interactive demonstration of smoothing with box filters and binomial filters (dip6ex11.01)

11.2: Multistep smoothing with box filters and binomial filters

Interactive demonstration of multistep smoothing with box filters and binomial filters (dip6ex11.02)

11.3: *Box filter

Box filter were discussed in detail in Section 11.3. Answer the following questions:

1. Why are box filters bad smoothing filters? List all reasons!
2. Do the bad features improve if you apply the filters several times? Take the 3×3 box filter as an example.
3. What is the resulting filter if you apply the box filter many times to an image?

11.4: **Filter design

A filter should be designed with a small mask and optimal smoothing properties. Use a mask with 3 coefficients: $[\alpha, \beta, \gamma]$.
The filter should have the following properties:

a) Preservation of the mean value

b) No shift of gray value structures
c) Structures with the largest possible wave number should vanish

Questions and tasks:

1. Are the filter coefficients α, β und γ determined uniquely?
2. Compute the transfer function of the filter
3. Which constraints are imposed to a filter with five coefficents $[\alpha, \beta, \gamma, \delta, \epsilon]$?
4. Compute the transfer function of the filter.
5. Which values can the remaining free parameter take so that the transfer function remains monotonically decreasing for all wave numbers?
6. Which coefficients have the corresponding filter masks for the limiting values?

11.5: ****Fast computation for smoothing filters**

Examine the number of computations (additions and multiplications) for several methods to convolve an image with the following 2-D smoothing mask:

$$\boldsymbol{B}^4 = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

and with the equivalent 3-D mask

$$\frac{1}{16}\left[\boldsymbol{B}^4, 4\boldsymbol{B}^4, 6\boldsymbol{B}^4, 4\boldsymbol{B}^4, \boldsymbol{B}^4\right]_z .$$

1. Computation without any optimization directly using the convolution equation
2. Avoiding any unnecessary multiplications by making use of the fact that many coefficients have the same value.
3. Decomposition into 1-D masks
4. Decomposition of the 1-D masks into the elementary mask 1/2[1 1]
5. Do you have any other ideas for efficient computation schemes?

11.6: ****Noise suppression by smoothing**

1. Prove that it is not possible to improve the signal-to-noise ratio for a arbitrary *single* wave number with a linear smoothing filter $\mathcal{H}$. (Hint: write the image $\boldsymbol{G}$ as a sum of the signal part $\boldsymbol{S}$ and the noise part $\boldsymbol{N}$.)
2. Assume white noise (equally distributed over all wave numbers), but a spectrum of the signal that is only equally distributed up to half of the maximum wave number. Is it now possible to improve the signal-to-noise ratio integrated over all wave numbers? What is the shape of the transfer function that optimizes the signal-to-noise ratio?

11.7: ***Transfer function of the 1-D box filter**

Prove Equation (11.12) for the transfer function of the 1-D box filter. (Hint: there are at least to ways to do this. One is to write the transfer function that it can be seen as a geometric sequence $a_0(1 + q + q^2 + \ldots + q^{n-1})$ with the sum $a_0(q^n - 1)/(q - 1)$. The other solution is based on the recursive computation of the box filter given by Eq. (11.15).)

11.8: *Adaptive smoothing

A simple adaptive smoothing filter that reduces smoothing at edges has the following form:

$$(1 - \alpha)\mathcal{I} + \alpha\mathcal{B} = \mathcal{I} + \alpha(\mathcal{B} - \mathcal{I}),$$

where $\alpha \in [0, 1]$ depends on the steepness of the edge, e. g. $\alpha = \gamma^2/(\gamma^2 + |\nabla g|^2)$

Answer the following questions assuming that $\mathcal{B}$ is a 3×3 binomial filter:

1. Explicitly compute the nine coefficients of the adaptive 3×3-Filters as a function of α.
2. Compare the computational effort of this direct implementation of the adaptive filter with the implementation as a steerable filter. Do not take into account the effort to compute α.

11.9 *Further Readings*

The articles of Simonds [188] and Wells [216] discuss fast algorithms for large Gaussian kernels. Readers with an interest in the general principles of efficient algorithms are referred to the textbooks of Aho et al. [4] or Sedgewick [183]. Blahut [11] deals with fast algorithms for digital signal processing. Classical filter design techniques, especially for IIR-filter are discussed in the standard textbooks for signal processing, e. g., Proakis and Manolakis [159] or Oppenheim and Schafer [148]. Lim [124] specifically deals with the design of 2-D IIR filters. A detailed description of the deterministic and statistical properties of *median filters* can be found in Huang [83] or Arce et al. [6]. They are also discussed in detail in the monograph on nonlinear digital filters by Pitas and Venetsanopoulos [155]. The monograph of Granlund and Knutsson [64] on signal processing for computer vision deals also with weighted averaging (normalized convolution, Section 11.6.2). Steerable filters (Section 11.6.3) were introduced by the articles of Freeman and Adelson [55] and Simoncelli et al. [187].