$$l = \text{floor}(x)\,, \quad a = x - l\,, \quad k = \text{floor}(y)\,, \quad b = y - k\,.$$

Linear interpolation can cause a small decrease in resolution, and blurring due to its averaging nature. The problem of step-like straight boundaries with the nearest-neighborhood interpolation is reduced.

Bi-cubic interpolation improves the model of the brightness function by approximating it locally by a bi-cubic polynomial surface; 16 neighboring points are used for interpolation. The one-dimensional interpolation kernel ('Mexican hat') is shown in Figure 5.9 and is given by

$$h_3 = \begin{cases} 1 - 2\,|x|^2 + |x|^3 & \text{for } 0 \le |x| < 1\,, \\ 4 - 8\,|x| + 5\,|x|^2 - |x|^3 & \text{for } 1 \le |x| < 2\,, \\ 0 & \text{otherwise.} \end{cases} \tag{5.23}$$

Bi-cubic interpolation does not suffer from the step-like boundary problem of nearest-neighborhood interpolation, and copes with linear interpolation blurring as well. Bi-cubic interpolation is often used in raster displays that enable zooming with respect to an arbitrary point. If the nearest-neighborhood method were used, areas of the same brightness would increase. Bi-cubic interpolation preserves fine details in the image very well.

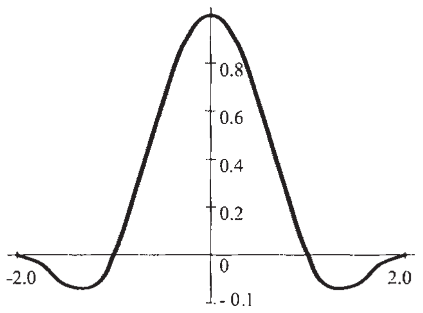

Figure 5.9: Bi-cubic interpolation kernel.

5.3 Local pre-processing

We shall consider methods that use a small neighborhood of a pixel in an input image to produce a new brightness value in the output image. Such pre-processing operations are called also **filtration** (or **filtering**) if signal processing terminology is used.

Local pre-processing methods can be divided into two groups according to the goal of the processing. First, **smoothing** aims to suppress noise or other small fluctuations in the image; it is equivalent to the suppression of high frequencies in the Fourier transform domain. Unfortunately, smoothing also blurs all sharp edges that bear important information about the image. Second, **gradient operators** are based on local derivatives of the image function. Derivatives are bigger at locations of the image where the image function undergoes rapid changes, and the aim of gradient operators is to indicate such locations in the image. Gradient operators have a similar effect to suppressing low frequencies in the Fourier transform domain. Noise is often high frequency in nature; unfortunately, if a gradient operator is applied to an image, the noise level increases simultaneously.

Clearly, smoothing and gradient operators have conflicting aims. Some pre-processing algorithms solve this problem and permit smoothing and edge enhancement simultaneously.

Another classification of local pre-processing methods is according to the transformation properties; **linear** and **non-linear** transformations can be distinguished.

Linear operations calculate the resulting value in the output image pixel $g(i,j)$ as a linear combination of brightnesses in a local neighborhood $\mathcal{O}$ of the pixel $f(i,j)$ in the input image. The contribution of the pixels in the neighborhood $\mathcal{O}$ is weighted by coefficients h:

$$f(i,j) = \sum\sum_{(m,n)\in\mathcal{O}} h(i-m, j-n)\, g(m,n)\,. \tag{5.24}$$

Equation (5.24) is equivalent to discrete convolution with the kernel h, which is called a **convolution mask**. Rectangular neighborhoods $\mathcal{O}$ are often used with an odd number of pixels in rows and columns, enabling specification of the central pixel of the neighborhood.

Local pre-processing methods typically use very little a priori knowledge about the image contents. It is very difficult to infer this knowledge while an image is being processed, as the known neighborhood $\mathcal{O}$ of the processed pixel is small. Smoothing operations will benefit if some general knowledge about image degradation is available; this might, for instance, be statistical parameters of the noise.

The choice of the local transformation, size, and shape of the neighborhood $\mathcal{O}$ depends strongly on the size of objects in the processed image. If objects are rather large, an image can be enhanced by smoothing of small degradations.

5.3.1 Image smoothing

Image smoothing is the set of local pre-processing methods whose predominant use is the suppression of image noise—it uses redundancy in the image data. Calculation of the new value is based on the averaging of brightness values in some neighborhood $\mathcal{O}$. Smoothing poses the problem of blurring sharp edges in the image, and so we shall concentrate on smoothing methods which are **edge preserving**. They are based on the general idea that the average is computed only from those points in the neighborhood which have similar properties to the point being processed.

Local image smoothing can effectively eliminate impulse noise or degradations appearing as thin stripes, but does not work if degradations are large blobs or thick stripes. The solution for complicated degradations may be to use image restoration techniques, described in Section 5.4.

Averaging, statistical principles of noise suppression

Assume that the noise value ν at each pixel is an independent random variable with zero mean and standard deviation σ. We might capture the same static scene under the same conditions n times. From each captured image a particular pixel value g_i, $i = 1, \ldots, n$ is selected; the index $_i$ indicates to which image the pixel value g_i belongs. An estimate of the correct value can be obtained as an average of these values, with corresponding noise values $\nu_1, \ldots, \nu_n$

$$\frac{g_1 + \ldots + g_n}{n} + \frac{\nu_1 + \ldots + \nu_n}{n}\,. \tag{5.25}$$

The second term here describes the effect of the noise, which is again a random value with zero mean and standard deviation $\sigma/\sqrt{n}$. Thus, if n images of the same scene are available, smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n} \sum_{k=1}^{n} g_k(i,j) \,. \tag{5.26}$$

This reasoning has roots in statistics which aims at estimating the most probable value from some population: a random sample is taken from the population and the corresponding sample mean value is calculated. If random samples were repeatedly selected and their sample mean values were calculated, we would obtain a distribution of sample mean values. This distribution of sample means has some useful properties:

- The mean value of the distribution of sample mean values is equal to the mean value of the population.
- The distribution of sample mean values has smaller variance than the original population. Suppose n samples are selected, and the standard deviation of the population is σ. Then the standard deviation of the distribution of sample mean values is $\sigma/\sqrt{n}$.
- If the original distribution is normal (Gaussian) then the distribution of sample mean values is also normal. In fact, no matter what the shape o the original distribution, the distribution of sample means converges to a normal distribution. This is the **central limit theorem**, which is one reason why the normal distribution plays such a critical role in statistics.
- From the practical point of view, it is important that not too many random selections have to be made. The central limit theorem tell us the distribution of sample mean values without the need to create them. In statistics, usually about 30 samples are considered the lowest limit of the necessary number of observations. The degree of belief in the parameter describing the population (we have only considered the mean so far) is expressed by a confidence interval.

In many cases only one image corrupted by noise is available, and averaging is then realized in 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 (5.24)]. For a 3×3 neighborhood, the convolution mask h is

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \,. \tag{5.27}$$

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 2.3.6)

$$h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \,. \tag{5.28}$$

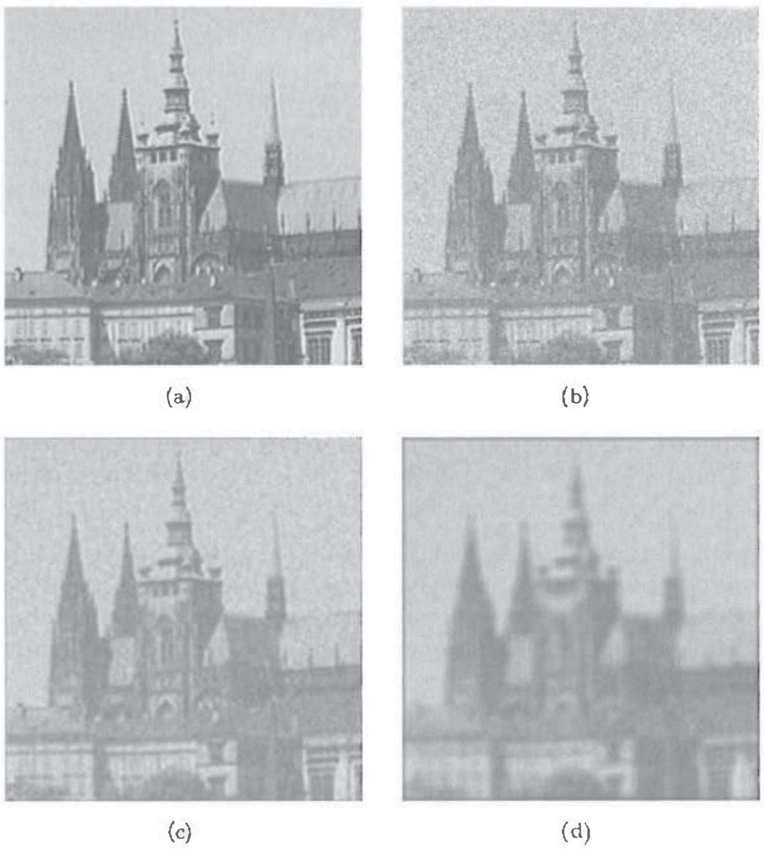

Figure 5.10: Noise with Gaussian distribution and averaging filters. (a) Original image. (b) Superimposed noise (random Gaussian noise characterized by zero mean and standard deviation equal to one-half of the gray-level standard deviation of the original image). (c) 3×3 averaging. (d) 7×7 averaging.

There are two commonly used smoothing filters whose coefficients gradually decrease to have the values close to zero at the limit of the window. This is the best way to minimize spurious oscillations in the frequency spectrum (see the discussion of the uncertainty principle (3.24)). These are the Gaussian and the Butterworth filters. Larger convolution masks for averaging by Gaussian filter are created according to the Gaussian distribution formula (5.47) and the mask coefficients are normalized to have a unit sum. The Butterworth filter will be explained in Section 5.3.8 dealing with the local pre-processing in the frequency domain.

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 5.10a shows an original image of Prague castle with 256 brightnesses; Figure 5.10b shows the same image with superimposed additive noise with Gaussian distribution; Figure 5.10c shows the result of averaging with a 3×3 convolution mask (5.28)—noise is significantly reduced and the image is slightly blurred. Averaging with a larger mask (7×7) is demonstrated in Figure 5.10d, where the blurring is much more serious.

Consider now the issue of computational cost. There is an important special case of convolution filters called **separable filters**. Separability in 2D means that the convolution kernel can be factorized as a product of two one-dimensional vectors, and theory provides a clue as to which convolution masks are separable. Every convolution mask with rank one is separable.

As an example, consider a binomic filter. Its elements are binomic numbers which are created as a sum of the corresponding two numbers in Pascal's triangle. Consider a 2D binomic filter of size 5×5. Such a filter can be decomposed into a product of two 1D vectors. h_1, h_2.

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} \\ h_1 \\ \\ \end{bmatrix} \begin{bmatrix} \quad h_2 \quad \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} .$$

Suppose a convolution kernel is of size $2N + 1$. Equation (5.24) allows the convolution to be rewritten taking account of special properties of a separable filter

$$g(x, y) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} h(m, n)\, f(x+m, y+n) = \sum_{m=-N}^{N} h_1(m) \sum_{n=-N}^{N} h_2(n)\, f(x+m, y+n) .$$

The direct calculation of the convolution according to Equation (5.24) would need, in our case of 5×5 convolution kernel, 25 multiplications and 24 additions for each pixel. If the separable filter is used then only 10 multiplications and 8 additions suffice.

The computational saving would be more pronounced in the case of convolution of a 3D image (e.g., from a computed tomography data). Consider the $9 \times 9 \times 9$ convolution kernel. There would be a need to use 729 multiplications and 728 additions per voxel if the convolution definition were followed. In the case of the separable filter, only 27 multiplications and 24 additions per voxel will do.

The smoothing discussed thus far was linear, which has the disadvantage that edges in the image are inevitably blurred. Alternative non-linear methods exist which reduce this. The neighborhood of the current pixel is inspected and divided into two subsets by a homogeneity criterion of the user's choice. One set consists of all pixels neighboring the current pixel or any pixel already included in this set, which satisfy the homogeneity criterion. The second set is the rest of the image. This selection operation is non-linear and causes the whole filter to be non-linear. Having selected the homogeneous subset containing the current pixel, the most probable value is sought in it by a linear or non-linear technique.

Averaging using a rotating mask

Averaging using a rotating mask is a non-linear smoothing method that avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood and the resulting image is in fact sharpened [Nagao and Matsuyama, 1980]. The brightness average is calculated only within this region; a brightness dispersion σ^2 is used as the region homogeneity measure. Let n be the number of pixels in a region R and g be the