Chapter 3

FILTERS

Most images are affected to some extent by **noise**, that is unexplained variation in data: disturbances in image intensity which are either uninterpretable or not of interest. Image analysis is often simplified if this noise can be *filtered out*. In an analogous way filters are used in chemistry to free liquids from suspended impurities by passing them through a layer of sand or charcoal. Engineers working in signal processing have extended the meaning of the term **filter** to include operations which accentuate features of interest in data. Employing this broader definition, image filters may be used to emphasise **edges** — that is, boundaries between objects or parts of objects in images. Filters provide an aid to visual interpretation of images, and can also be used as a precursor to further digital processing, such as segmentation (chapter 4).

Most of the methods considered in chapter 2 operated on each pixel separately. Filters change a pixel's value taking into account the values of neighbouring pixels too. They may either be applied directly to recorded images, such as those in chapter 1, or after transformation of pixel values as discussed in chapter 2. To take a simple example, Figs 3.1(b)–(d) show the results of applying three filters to the cashmere fibres image, which has been redisplayed in Fig 3.1(a).

- Fig 3.1(b) is a display of the output from a 5×5 **moving average filter**. Each pixel has been replaced by the average of pixel values in a 5×5 square, or **window** centred on that pixel. The result is to reduce noise in the image, but also to blur the edges of the fibres. A similar effect can be produced by looking at Fig 3.1(a) through half-closed eyes.
- If the output from the moving average filter is subtracted from the original image, on a pixel-by-pixel basis, then the result is as shown in Fig 3.1(c) (which has been displayed with the largest negative pixel values shown as black and the largest positive pixel values shown as white). This filter (the original image minus its smoothed version) is a **Laplacian filter**. It has had the effect of emphasising edges in the image.
- Fig 3.1(d) shows the result produced when output from the Laplacian filter is added to the original image, again on a pixel-by-pixel basis. To the eye, this image looks clearer than Fig 3.1(a) because transitions at edges have been magnified — an effect known as **unsharp masking**.

We will consider these three filters in more detail in §3.1.

The above filters are all **linear**, because output values are linear combinations of the pixels in the original image. Linear methods are far more amenable to mathematical analysis than are nonlinear ones, and are consequently far better understood. For example, if a linear filter is applied to the output from another linear filter, then the result is a third linear filter. Also, the result would be the same if the order in which the two filters were applied was reversed. There are two, complementary, ways of studying linear filters, namely in the **spatial** and **frequency** domains. These approaches are considered in §3.1 and §3.2 respectively. The less mathematical reader may prefer to skip §3.2. This can be done without losing the sense of the rest of the chapter.

Nonlinear filters — that is, all filters which are not linear — are more diverse and difficult to categorize, and are still an active area of research. They are potentially more powerful than linear filters because they are able to reduce noise levels without simultaneously blurring edges. However, their theoretical foundations are far less secure and they can produce features which are entirely spurious. Therefore care must be taken in using them. In §3.3, some nonlinear smoothing filters are considered, and in §3.4, nonlinear edge-detection filters are introduced.

Finally, the key points of the chapter are summarized in §3.5.

3.1 Linear filters in the spatial domain

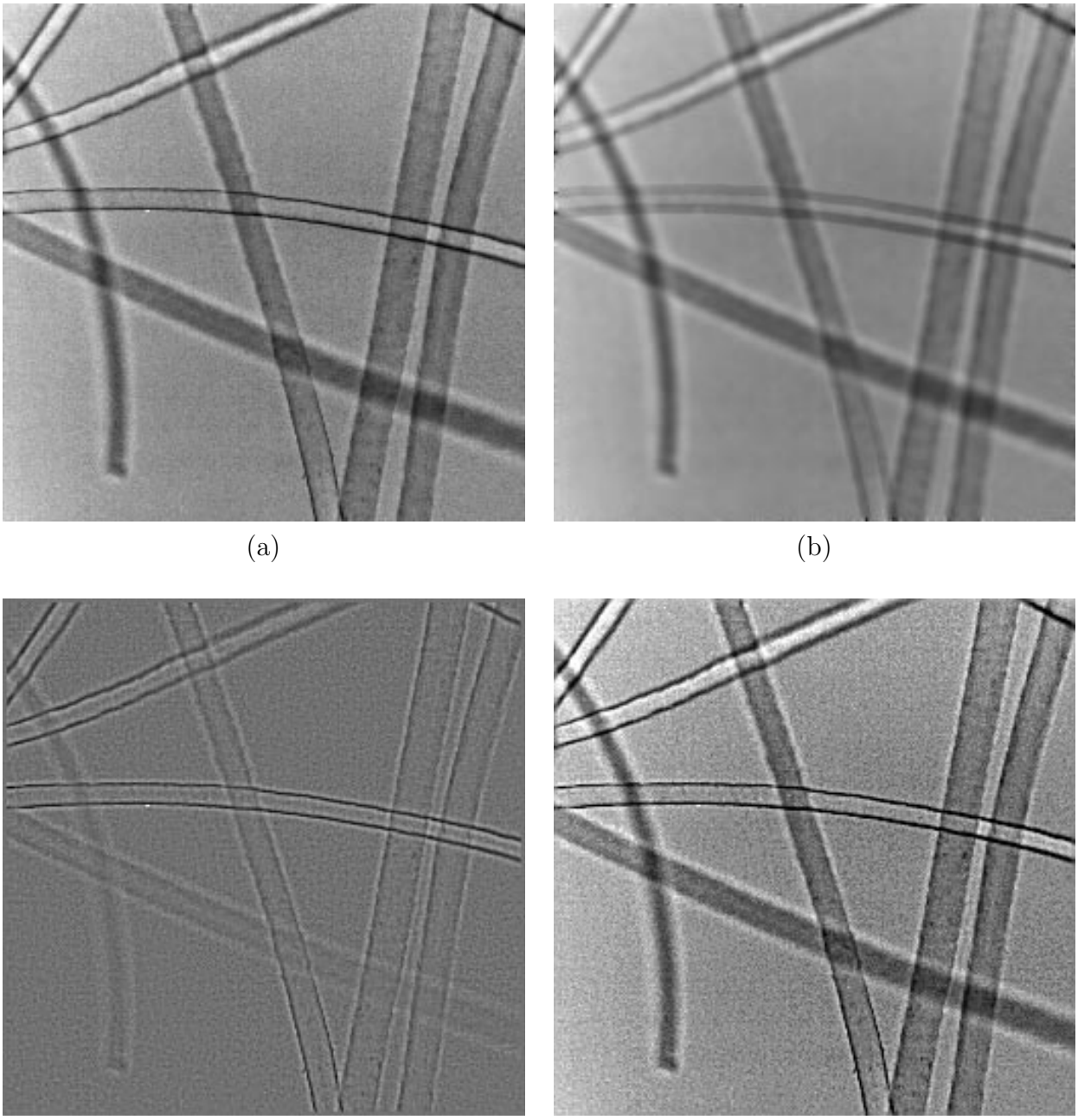
The moving average, or **box filter**, which produced Fig 3.1(b) is the simplest of all filters. It replaces each pixel by the average of pixel values in a square centred at that pixel. All linear filters work in the same way except that, instead of forming a simple average, a weighted average is formed. Using the terminology of chapter 1, let f_{ij}, for $i, j = 1, \ldots, n$, denote the pixel values in the image. We will use g, with pixel values g_{ij}, to denote the output from the filter. A linear filter of size $(2m+1) \times (2m+1)$, with specified weights w_{kl} for $k, l = -m, \ldots, m$, gives

$$g_{ij} = \sum_{k=-m}^{m} \sum_{l=-m}^{m} w_{kl} f_{i+k,j+l} \qquad \text{for } i, j = (m+1), \ldots, (n-m).$$

For example, if $m = 1$, then the window over which averaging is carried out is 3×3, and

$$\begin{aligned} g_{ij} = \quad & w_{-1,-1} \; f_{i-1,j-1} \quad + w_{-1,0} \; f_{i-1,j} \quad + w_{-1,1} \; f_{i-1,j+1} \\ & + w_{0,-1} \; f_{i,j-1} \quad + w_{0,0} \; f_{i,j} \quad + w_{0,1} \; f_{i,j+1} \\ & + w_{1,-1} \; f_{i+1,j-1} \quad + w_{1,0} \; f_{i+1,j} \quad + w_{1,1} \; f_{i+1,j+1}. \end{aligned}$$

For full generality, the weights (w) can depend on i and j, resulting in a filter which varies across the image. However, the linear filters considered in this chapter will all be spatially invariant. Also, all the filters will have windows composed of odd numbers of rows and columns. It is possible to have even-sized windows, but then there is a half-pixel displacement between the input and output images.



(a) (b)

(c) (d)

Figure 3.1: Application of linear filters to cashmere image: **(a)** original image, **(b)** output from 5×5 moving average filter, **(c)** result of subtracting output of 5×5 moving average filter from original image, **(d)** result of adding original image to the difference between the output from 5×5 moving average filter and the original image.

Note that the **borders** of g, that is

$$g_{ij} \quad \text{where either } i \text{ or } j = 1, \ldots, m \quad \text{or} \quad (n-m+1), \ldots, n,$$

have not been defined above. Various possibilities exist for dealing with them:

1. They could be discarded, resulting in g being smaller than f.
2. The pixels in the borders of g could be assigned the same values as those in the borders of f.
3. The border pixels in g could be set to zero.
4. The filter could be modified to handle incomplete neighbourhoods, for example:
   (a) by ignoring those parts of the neighbourhood which lie outside the image,
   (b) by reflecting the input image (f) along its first and last row and column, so that pixel values $f_{i,n+1} = f_{i,n-1}$ etc,
   (c) by wrapping-round the input image so that $f_{i,n+1} = f_{i,1}$ etc, as though it were on a torus.

In this chapter we will take option 2 for smoothing filters and option 3 for edge-detection filters, except in §3.2 where we will make use of option 4(c). Strictly speaking, this wrap-round approach is the only valid option for the mathematical results on linear filters to be applicable over the whole image.

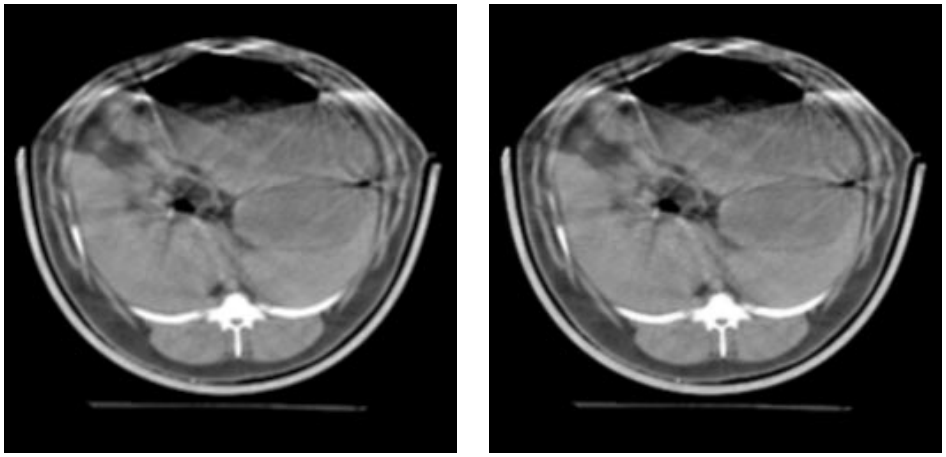
If all the elements in w are *positive*, then the effect of the filter is to smooth the image. The two most commonly-used filters of this type, the moving average and the Gaussian, will be considered in §3.1.1. If some weights are *negative*, then the filter outputs a difference between pixel values, which can have the effect of emphasising edges. Filters of this type will be presented in §3.1.2.

3.1.1 Smoothing

For the moving average filter, $w_{kl} = 1/(2m+1)^2$. Figs 3.2(a), (c) and (e) show the results of applying moving average filters with windows of size 3×3, 5×5 and 9×9 to the transformed X-ray image in Fig 2.3(b). As can be seen, the bigger the window the greater the noise reduction and blurring.

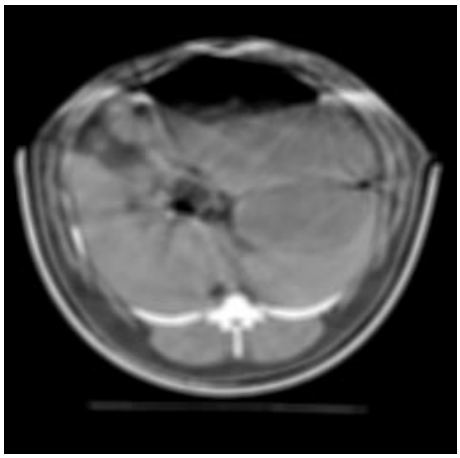
Computational efficiency is an important consideration in image analysis because of the size of data sets. In total, there are $(2m+1)^2$ additions and multiplications per pixel involved in deriving g from f. However, some filters can be computed more quickly. A filter is said to be **separable** if it can be performed by first filtering the image inside a $(2m+1) \times 1$ window, and then inside a $1 \times (2m+1)$ window. In other words, it can be separated into a column operation:

$$h_{ij} = \sum_{k=-m}^{m} w_k^c f_{i+k,j} \qquad \text{for } i = (m+1), \ldots, (n-m); \quad j = 1, \ldots, n,$$

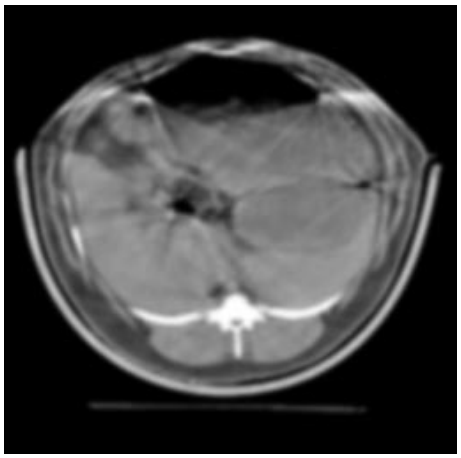


(a)

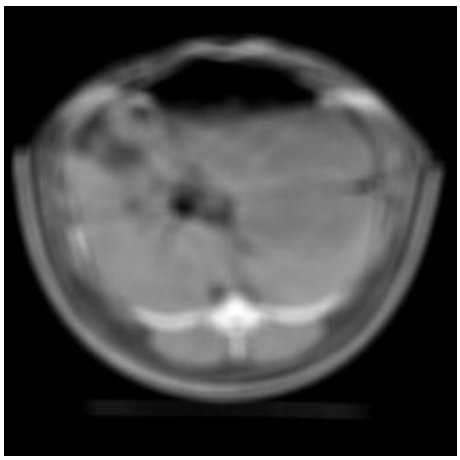
(b)



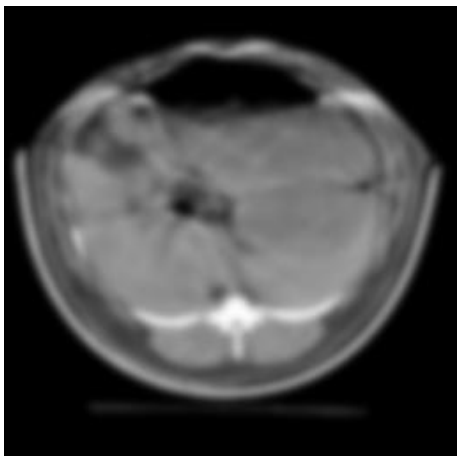
(c)



(d)



(e)



(f)

Figure 3.2: Linear smoothing filters applied to X-ray image: **(a)** 3×3 moving average, **(b)** Gaussian, $\sigma^2 = \frac{2}{3}$, **(c)** 5×5 moving average, **(d)** Gaussian, $\sigma^2 = 2$, **(e)** 9×9 moving average, **(f)** Gaussian, $\sigma^2 = 6\frac{2}{3}$.

using column weights $w^c_{-m}, \ldots, w^c_m$, followed by a row operation:

$$g_{ij} = \sum_{l=-m}^{m} w^r_l h_{i,j+l} \qquad \text{for } i, j = (m+1), \ldots, (n-m),$$

using row weights $w^r_{-m}, \ldots, w^r_m$. In order for this to be possible, the array of weights, w_{kl}, must be expressible as the product of the column and row weights, as follows:

$$w_{kl} = w^c_k w^r_l \qquad \text{for } k, l = -m, \ldots, m.$$

The number of operations per pixel has been reduced from $(2m+1)^2$ to $2(2m+1)$. Therefore a separable filter can be computed more quickly than one that is not separable, even when $m = 1$.

Although the moving-average filter is separable (with $w^c_k = w^r_l = 1/(2m+1)$), there exists a yet more efficient algorithm. This uses a **recursive** implementation, that is, one in which the output from the filter at location (i, j) is updated to obtain the output at location $(i+1, j)$. In contrast, the formulae we have considered so far involve calculating from scratch the output at each location. Specifically, the first $(2m+1)$ pixel values in column j are averaged:

$$h_{m+1,j} = \frac{1}{2m+1} \sum_{k=1}^{2m+1} f_{kj}.$$

Then, the pixel value in the first row (f_{1j}) is dropped from the average and the pixel in row $(2m+2)$ is added. This operation is repeated for every value in column j, so that:

$$h_{ij} = h_{i-1,j} + \frac{f_{i+m,j} - f_{i-m-1,j}}{2m+1} \qquad \text{for } i = (m+2), \ldots, (n-m).$$

This procedure is repeated for each column $j = 1, \ldots, n$, to obtain h. Then the same algorithm is applied along each row of h, to obtain g. The number of operations per pixel has been reduced to 4 *irrespective of the filter size* (m).

Table 3.1 gives times for the general, separable and moving average algorithms considered above, implemented in a Fortran77 program to run on a SUN Sparc2 computer. (Timings for filters to be discussed later in the chapter are also included.) Separable and, in particular, moving average filters run much more quickly than the general linear filter, particularly when image and window sizes are large.

Although the moving average filter is simple and fast, it has two drawbacks:

1. It is not **isotropic** (i.e. circularly symmetric), but smooths further along diagonals than along rows and columns.
2. Weights have an abrupt cut-off rather than decaying gradually to zero, which leaves discontinuities in the smoothed image.

Artefacts introduced by the square window can be seen in Fig 3.2(e), particularly around the sheep's backbone. Drawback 1 could be overcome by calculating the average in a lattice

approximation to a circular, rather than a square, neighbourhood. Such a filter with constant weights would not be separable, but could be implemented reasonably efficiently using a 2-D version of the recursive algorithm.

Gaussian filters are the only ones which are separable and, at least to a lattice approximation, circularly symmetric. They also overcome the other stated drawback of moving average filters because weights decay to zero. Gaussian filters have weights specified by the probability density function of a bivariate Gaussian, or Normal, distribution with variance σ^2, that is

$$w_{ij} = \frac{1}{2\pi\sigma^2} \exp\left\{\frac{-(i^2+j^2)}{2\sigma^2}\right\} \qquad \text{for } i,j = -[3\sigma],\ldots,[3\sigma],$$

for some specified positive value for σ^2. Here 'exp' denotes the exponential function and $[3\sigma]$ represents the 'integer part' of 3σ. Limits of $\pm 3\sigma$ are chosen because Gaussian weights are negligibly small beyond them. Note, that the divisor of $2\pi\sigma^2$ ensures that the weights sum to unity (approximately), which is a common convention with smoothing filters. If $\sigma^2 = 1$, the array of weights is

$$w = \frac{1}{1000}\begin{pmatrix} 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 3 & 13 & 22 & 13 & 3 & 0 \\ 1 & 13 & 59 & 97 & 59 & 13 & 1 \\ 2 & 22 & 97 & 159 & 97 & 22 & 2 \\ 1 & 13 & 59 & 97 & 59 & 13 & 1 \\ 0 & 3 & 13 & 22 & 13 & 3 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \end{pmatrix}.$$

(For succinctness of notation, we show a 7×7 array to represent the weights w_{kl} for $k,l = -3,\ldots,3$, and we have specified the weights only to three decimal places — hence the divisor of 1000 at the beginning of the array.)

Figs 3.2(b), (d) and (f) show the results of applying Gaussian filters with $\sigma^2 = \frac{2}{3}, 2$ and $6\frac{2}{3}$ to the X-ray image. These three values of σ^2 give filters which average to the same extents (have the same variance) as the moving average filters already considered. Figs 3.2(a) and (b) can be seen to be very similar, as can Figs 3.2(c) and (d). However, the Gaussian filter has produced a smoother result in Fig 3.2(f), when compared with the moving average filter in Fig 3.2(e).

For certain values of σ^2, Gaussian filters can be implemented efficiently by approximating them by repeated applications of moving average filters. For example, in one dimension, four iterations of a moving average filter of width 3 is equivalent to filtering using weights of:

$$\tfrac{1}{81}(\ 0 \qquad 1 \qquad 4 \qquad 10 \qquad 16 \qquad 19 \qquad 16 \qquad 10 \qquad 4 \qquad 1 \qquad 0\ )$$

A Gaussian filter with a variance of $\sigma^2 = \frac{8}{3}$, has weights:

$\frac{1}{81}$(0.2 1.0 3.7 9.4 16.4 19.8 16.4 9.4 3.7 1.0 0.2)

The agreement is seen to be very good. In general, four repeats of a moving average filter of size $(2m+1) \times (2m+1)$ approximates a Gaussian filter with $\sigma^2 = \frac{4}{3}(m^2+m)$ (Wells, 1986). Therefore, in order to use the approximation we must have $\sigma^2 \geq \frac{8}{3}$, for which the window is at least 9×9. If four iterations of a moving average filter are used, the number of operations per pixel is 16, irrespective of the value of σ^2. Table 3.1 gives computer timings for this approximation of a Gaussian filter. Provided m can be found such that $\frac{4}{3}(m^2+m)$ is an acceptable value for σ^2, the iterated algorithm is faster than a separable implementation in a $(2[3\sigma]+1) \times (2[3\sigma]+1)$ window.

As an aside, it may be noted that the Gaussian filter can also be used to simultaneously smooth and interpolate between pixels, provided that $\sigma^2 \geq 1$. At location (y, x), for non-integer row index y and column index x, the estimated intensity is an average of local pixel values, weighted by the Gaussian density function:

$$g(y,x) = \sum_{i=[y-3\sigma]}^{i=[y+3\sigma]} \sum_{j=[x-3\sigma]}^{j=[x+3\sigma]} \frac{f_{ij}}{2\pi\sigma^2} \exp\left\{\frac{-\{(i-y)^2+(j-x)^2\}}{2\sigma^2}\right\}.$$

This is an example of **kernel regression** (Eubank, 1988).

Any set of weights, w_{ij} , which are all positive will smooth an image. Alternative strategies for choosing weights include template matching, local fitting of polynomial surfaces, and minimum error predictors. The latter case will be returned to in §3.2.3 as an example of a Wiener filter. Hastie and Tibshirani (1990, chapter 2) review alternative statistical approaches to smoothing. As with all smoothing operations, there is the fundamental trade-off between *variance* and *bias*: a filter which operates in a larger neighbourhood will be more effective at reducing noise but will also blur edges.

3.1.2 Edge detection

To emphasise edges in an image it is necessary for some of the weights in the filter to be negative. In particular, sets of weights which sum to zero produce, as output, differences in pixel values in the input image. A **first-derivative row filter** gives, at a position in the output image, the difference in pixel values in columns on either side of that location in the input image. Therefore the output from the filter will be large in magnitude (either negative or positive) if there is a marked difference in pixel values to the left and right of a pixel location. One set of weights, which also involves some smoothing in averaging over 3 rows, is given by

$$w = \frac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}.$$

This choice of weights gives a separable filter, achieves some noise reduction and, in a sense to be explained below, estimates the first-derivative of the image intensity in the row direction.

To show that the output does estimate the first-derivative, consider the image intensity $f(y, x)$ to be specified for continuously varying row index y and column index x, and to be differentiable. By a Taylor's series expansion, f near (i, j) can be approximated by f_{ij}, together with a sum of partial derivatives of f with respect to y and x evaluated at (i, j):

$$f_{i+k,j+l} \approx f_{ij} + k\frac{\partial f_{ij}}{\partial y} + l\frac{\partial f_{ij}}{\partial x} + \frac{k^2}{2}\frac{\partial^2 f_{ij}}{\partial y^2} + \frac{l^2}{2}\frac{\partial^2 f_{ij}}{\partial x^2} + kl\frac{\partial^2 f_{ij}}{\partial y \partial x}.$$

It is found, after some messy algebra in which most terms cancel, that for the above array of weights:

$$\sum_{k=-1}^{1}\sum_{l=-1}^{1} w_{kl} f_{i+k,j+l} \approx \frac{\partial f_{ij}}{\partial x},$$

i.e. a first-derivative in the row direction.

Fig 3.3(a) shows the result when the first-derivative row filter is applied to the X-ray image, and Fig 3.3(b) shows the column counterpart, a **first-derivative column filter** obtained using

$$w = \frac{1}{6}\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

(These images have been displayed with zero pixel values shown as mid-grey, positive values as lighter greys and negative values as darker grey.) The results are strikingly similar to views of a landscape illuminated by a low sun. Our eyes cannot resist seeing the resultant image as 3-D, even though it isn't. Note that the row filter produces non-zero values in response to edges which are approximately vertical (i.e. down columns of the image), whereas the column filter responds most to horizontal (along row) edges.

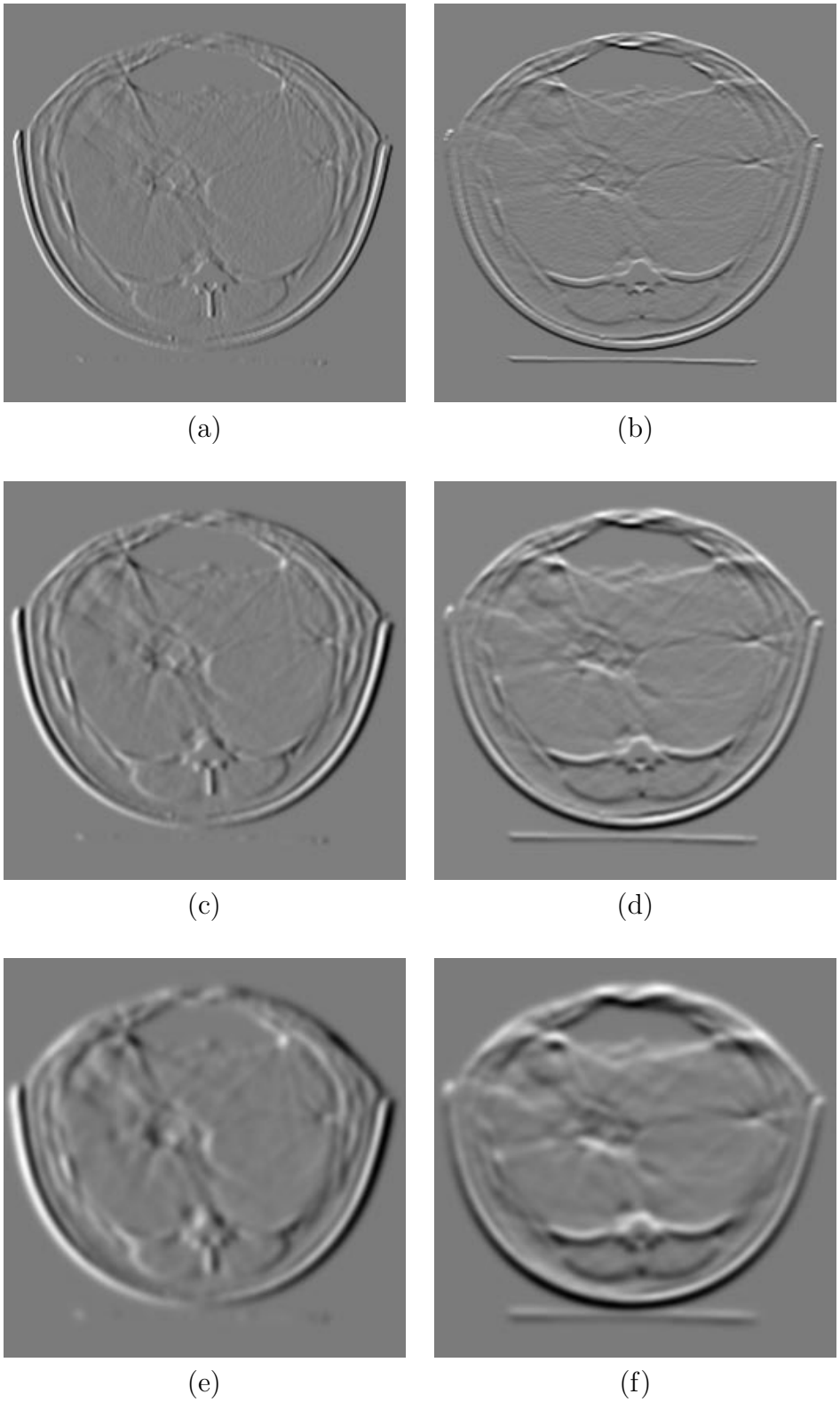
Although these filters do well at emphasising edges they have the disadvantage of also emphasising noise, as can be seen in Figs 3.3(a) and (b). The effects of noise can be diminished by evaluating the filters over larger sizes of window. A simple way to achieve this is by applying the derivative filters to the output from a smoothing filter such as the Gaussian already considered. No further smoothing is necessary, so w can be simplified to a 1×3 filter

$$w = \frac{1}{2}(\ -1 \quad 0 \quad 1\ ),$$

for the first-derivative row filter, and its transpose, a 3×1 filter

$$w = \frac{1}{2}\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix},$$

for the first-derivative column filter. The results of applying these filters to images, after Gaussian smoothing with $\sigma^2 = 2$ and $6\frac{2}{3}$ (i.e. the images displayed in Figs 3.2(d) and (f)) are shown in Figs 3.3(c)-(f). The combined filter has weights which are a **convolution** of the



(a) (b) (c) (d) (e) (f)

Figure 3.3: First-derivative filters applied to X-ray image: **(a)** 3×3 row filter, **(b)** 3×3 column filter, **(c)** row filter after Gaussian, $\sigma^2 = 2$, **(d)** column filter after Gaussian, $\sigma^2 = 2$, **(e)** row filter after Gaussian, $\sigma^2 = 6\frac{2}{3}$, **(f)** column filter after Gaussian, $\sigma^2 = 6\frac{2}{3}$.

weights in the two constituent filters. In the case of the first-derivative row filter with $\sigma^2 = 2$, the weights are:

$$w = \frac{1}{1000}\begin{pmatrix} 0 & -1 & -3 & -3 & 0 & 3 & 3 & 1 & 0 \\ -2 & -5 & -10 & -9 & 0 & 9 & 10 & 5 & 2 \\ -3 & -11 & -21 & -20 & 0 & 20 & 21 & 11 & 3 \\ -4 & -14 & -27 & -25 & 0 & 25 & 27 & 14 & 4 \\ -3 & -11 & -21 & -20 & 0 & 20 & 21 & 11 & 3 \\ -2 & -5 & -10 & -9 & 0 & 9 & 10 & 5 & 2 \\ 0 & -1 & -3 & -3 & 0 & 3 & 3 & 1 & 0 \end{pmatrix}.$$

Therefore the filter output at a pixel is again the difference in pixel values to the left and right of it, but here more values have been included in the averaging process on each side.

Linear combinations of first-derivative row and column filters can be used to produce first-derivative filters in other directions. But unfortunately they have to be combined in a nonlinear way in order to be sensitive to edges at all orientations simultaneously. Hence, further discussion of them will be deferred until we consider nonlinear edge filters in §3.4.

Second-derivative filters, unlike first-derivative ones, can be isotropic, and therefore responsive to edges in any direction, while remaining linear. Consider the array of weights

$$w = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

This is the same type of filter as that used to produce Fig 3.1(c), i.e. the difference between the output from a moving average filter and the original image. It can be shown that with these weights,

$$\sum_{k=-1}^{1}\sum_{l=-1}^{1} w_{kl} f_{i+k,j+l} \approx \frac{\partial^2 f_{ij}}{\partial x^2} + \frac{\partial^2 f_{ij}}{\partial y^2},$$

which is the **Laplacian** transform of f. Fig 3.4(a) shows the result of applying this filter to the X-ray image. (As with Fig 3.3, zero pixel values are displayed as mid-grey.) Edges in the original image manifest themselves as **zero-crossings** in the Laplacian — that is, on one side of an edge the Laplacian will be positive while on the other side it will be negative. This is because an edge produces a maximum or minimum in the first derivative, and therefore it generates a zero value in the second derivative. Fig 3.4(b) provides an aid to identification of zero-crossings: negative values from the Laplacian filter are displayed as black and positive values as white. Therefore zero-crossings correspond to black/white boundaries. As with the first-derivative filters, noise has unfortunately been accentuated together with edges.

To reduce the effects of noise, the above filter or a simpler version of the Laplacian filter:

$$w = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

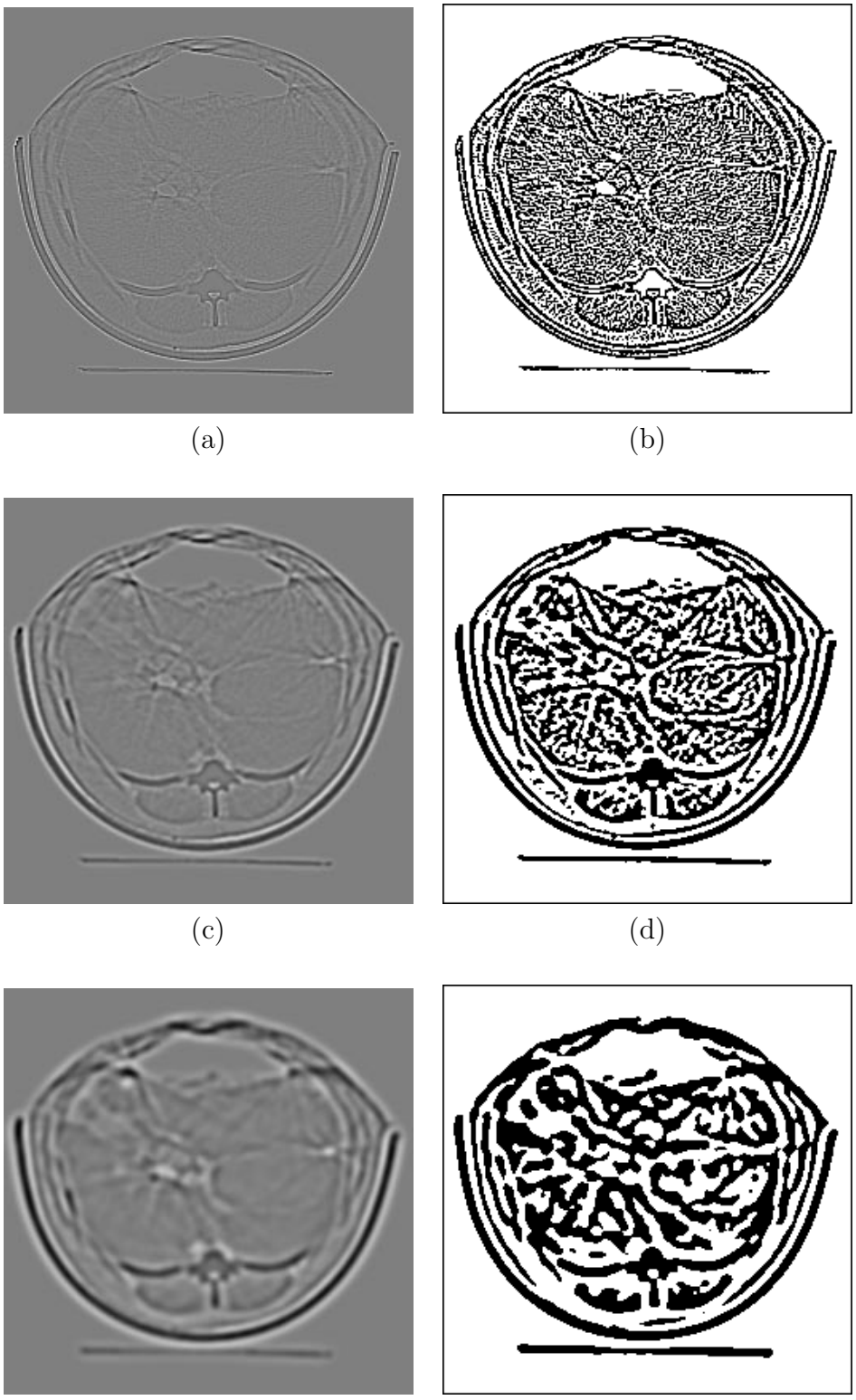


Figure 3.4: Second-derivative filters applied to X-ray image: **(a)** 3×3 Laplacian filter, **(b)** zero-thresholded version of (a), **(c)** Laplacian-of-Gaussian filter, $\sigma^2 = 2$, **(d)** zero-thresholded version of (c), **(e)** Laplacian-of-Gaussian filter, $\sigma^2 = 6\frac{2}{3}$, **(f)** zero-thresholded version of (e).

can be applied to the image produced after Gaussian smoothing of the original. The results with $\sigma^2 = 2$ and $6\frac{2}{3}$ are shown in Figs 3.4(c) and (e), and zero-crossings are shown as the black/white boundaries in Figs 3.4(d) and (f). As σ^2 increases, the number of apparent edges decreases. These combined filters, obtained by Gaussian smoothing and then applying the Laplacian filter, are termed **Laplacian-of-Gaussian** filters. They were proposed by Marr and Hildreth (1980) in a seminal paper on vision. It is of interest to note that the weights, which have the following expression:

$$w_{ij} = \frac{1}{2\pi\sigma^6}(i^2 + j^2 - 2\sigma^2) \exp\left\{\frac{-(i^2 + j^2)}{2\sigma^2}\right\},$$

can be re-expressed as the derivative of the Gaussian weights with respect to σ^2:

$$2\frac{\partial}{\partial\sigma^2}\left(\frac{1}{2\pi\sigma^2} \exp\left\{\frac{-(i^2 + j^2)}{2\sigma^2}\right\},\right)$$

and so can be approximated by the difference between a pair of Gaussian filters with different values of σ^2, the so-called **difference-of-Gaussian** (**DoG**) **filter**. An image can be decomposed into a series of differences between outputs of Gaussian filters at a range of variances, to give a **multiresolution representation** of an image. The new field of wavelets extends these ideas and gives them a more rigorous foundation (see, for example, Mallat, 1989).

All the filters we have considered so far have had weights which are either all positive, or which sum to zero. However, such restrictions are not necessary. For example, the **unsharp masking filter** which was used to produce Fig 3.1(d) has weights:

$$w = \frac{1}{9}\begin{pmatrix} -1 & -1 & -1 \\ -1 & 17 & -1 \\ -1 & -1 & -1 \end{pmatrix}.$$

This filter emphasises edges, but (approximately) retains the original pixel values in edge-free regions of an image. It is used to aid visual interpretation of images rather than as an edge-detection filter.

3.2 Linear filters in the frequency domain

Instead of representing an image as an $n \times n$ array of pixel values, we can alternatively represent it as the sum of many sine waves of different frequencies, amplitudes and directions. This is referred to as the **frequency domain** or **Fourier representation**. The parameters specifying the sine waves are termed **Fourier coefficients**. For some scientists, particularly engineers, this seems an obvious thing to do — for others it may take some getting used to. The reasons why we are taking this approach are:

- Extra insight can be gained into how linear filters work by studying them in the frequency domain.

- Some linear filters can be computed more efficiently in the frequency domain, by using the Fast Fourier Transform (FFT).
- New filters can be identified.

In §3.2.1 we will present the basic theory of Fourier transforms. Then, in §3.2.2 we will look again at the linear filters already considered in §3.1. Finally, in §3.2.3 we will develop some new filters by specifying them in the frequency domain. The less-mathematical reader may prefer to skip the rest of this section, which can be done without losing the sense of the rest of the chapter.

3.2.1 Theory of the Fourier transform

The **Fourier transform** of f is the $n \times n$ array, f^* , defined by

$$f^*_{kl} = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} \exp\left\{\frac{-2\pi\imath}{n}(ik+jl)\right\} \qquad \text{for } k,l = 1,\ldots,n,$$

where $\imath$ is the square-root of -1. The inverse transform is very similar, but recovers f from f^*. It is

$$f_{ij} = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} f^*_{kl} \exp\left\{\frac{2\pi\imath}{n}(ik+jl)\right\} \qquad \text{for } i,j = 1,\ldots,n.$$

It is arbitrary where the divisor n^2 appears in the two equations above, and different authors make different choices. (It should however be borne in mind that this choice affects some of the equations which follow.)

Alternatively, we can avoid using $\imath$. If the real part of f^* (which is a complex array) is denoted by a and the imaginary part (that is the term involving $\imath$) is denoted by b, then

$$a_{kl} = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\cos\left\{\frac{2\pi}{n}(ik+jl)\right\}, \quad b_{kl} = -\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij}\sin\left\{\frac{2\pi}{n}(ik+jl)\right\},$$

$$\begin{aligned}\text{and } f_{ij} &= \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n}\left(a_{kl}\cos\left\{\frac{2\pi}{n}(ik+jl)\right\} - b_{kl}\sin\left\{\frac{2\pi}{n}(ik+jl)\right\}\right),\\ &= \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} r_{kl}\cos\left\{\frac{2\pi}{n}(ik+jl) \quad + \quad \theta_{kl}\right\},\end{aligned}$$

where r_{kl} and θ_{kl} are the **amplitude** and **phase** of the Fourier transform at frequency (k,l): $r_{kl} = \sqrt{a_{kl}^2 + b_{kl}^2}$, $\theta_{kl} = \tan^{-1}(b_{kl}/a_{kl})$. (Here, and subsequently, '$\tan^{-1}$' must produce output over a range of 2π radians, to ensure that $\cos\theta_{kl} = a_{kl}/r_{kl}$ and $\sin\theta_{kl} = b_{kl}/r_{kl}$. The Fortran77 and C functions 'atan2' will achieve this). Therefore, as we can see, f is a sum of sine (or cosine) terms.

Let us take a simple example, composed of only a few sine terms, to see what this transformation looks like. If f is the 4×4 array:

$$f = \begin{pmatrix} 2 & 5 & 2 & 7 \\ 1 & 6 & 3 & 6 \\ 2 & 7 & 2 & 5 \\ 3 & 6 & 1 & 6 \end{pmatrix},$$

then by substituting numbers into the above equations we obtain the result that the real and imaginary parts of f^* are:

$$a = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 32 & 0 & 64 \end{pmatrix} \quad \text{and } b = \begin{pmatrix} -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

These can be re-expressed as amplitudes and phases of:

$$r = \begin{pmatrix} 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 32 & 0 & 64 \end{pmatrix} \quad \text{and } \theta = \frac{\pi}{2} \begin{pmatrix} -1 & * & * & * \\ * & * & * & * \\ * & * & 1 & * \\ * & 0 & * & 0 \end{pmatrix},$$

where '$*$' denotes θ_{kl} which are undefined because $a_{kl} = b_{kl} = 0$. Therefore, f is the sum of four cosines, namely for $i, j = 1, \ldots, 4$:

$$f_{ij} = \frac{r_{11}}{16} \cos\left\{\frac{\pi}{2}(i+j) + \theta_{11}\right\} + \frac{r_{33}}{16} \cos\left\{\frac{\pi}{2}(3i+3j) + \theta_{33}\right\}$$
$$+ \frac{r_{42}}{16} \cos\left\{\frac{\pi}{2}(4i+2j) + \theta_{42}\right\} + \frac{r_{44}}{16} \cos\left\{\frac{\pi}{2}(4i+4j) + \theta_{44}\right\}.$$

This works out as:

$$f = \frac{1}{2} \begin{pmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}$$
$$+2 \begin{pmatrix} -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} + 4 \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Notice that the first two arrays are identical, and both represent a sinusoidal wave in a direction from top-left to bottom-right of the arrays. The third array is a sinusoidal wave between columns, and at a higher frequency than the previous two, whereas the fourth array provides an overall mean.

We shall now consider Fourier transforms of some images. Figs 3.5(a) and (b) show r and θ for the X-ray image, respectively. Following common practice, in order to aid interpretation, these

arrays have been displayed in the range $k, l = -\frac{n}{2}, \ldots, \frac{n}{2} - 1$. Therefore, the zero-frequency coefficient is positioned in the centres of the displays. (Because of the periodic nature of sines and cosines, the range of summations in the Fourier transforms is arbitrary, subject to it being over an $n \times n$ square. Therefore, f^*_{kl} is equal to $f^*_{k+in,l+jn}$ for any values of i and j, and summation can range from $-\frac{n}{2}$ to $\frac{n}{2} - 1$.) The amplitude image has been displayed on a logarithmic scale, because otherwise all except the large values at low frequencies would appear as black. In the phase image, an angle of $-180°$ is displayed as black and one of $+180°$ is displayed as white. Note that r has 180° rotational symmetry and θ has 180° rotational antisymmetry (i.e. $r_{kl} = r_{-k,-l}$ and $\theta_{kl} = -\theta_{-k,-l}$) in these and all subsequent Fourier transforms. The amplitude display shows high coefficients at low frequencies, as is typical of images in general, whereas little can be discerned from the phase display.

Figs 3.5(c) and (d) show Fourier coefficients for the cashmere fibres image. The radiating spokes visible in Fig 3.5(c) correspond to straight edges of fibres. Again, amplitude shows high coefficients at low frequencies and the phase display shows no discernible pattern.

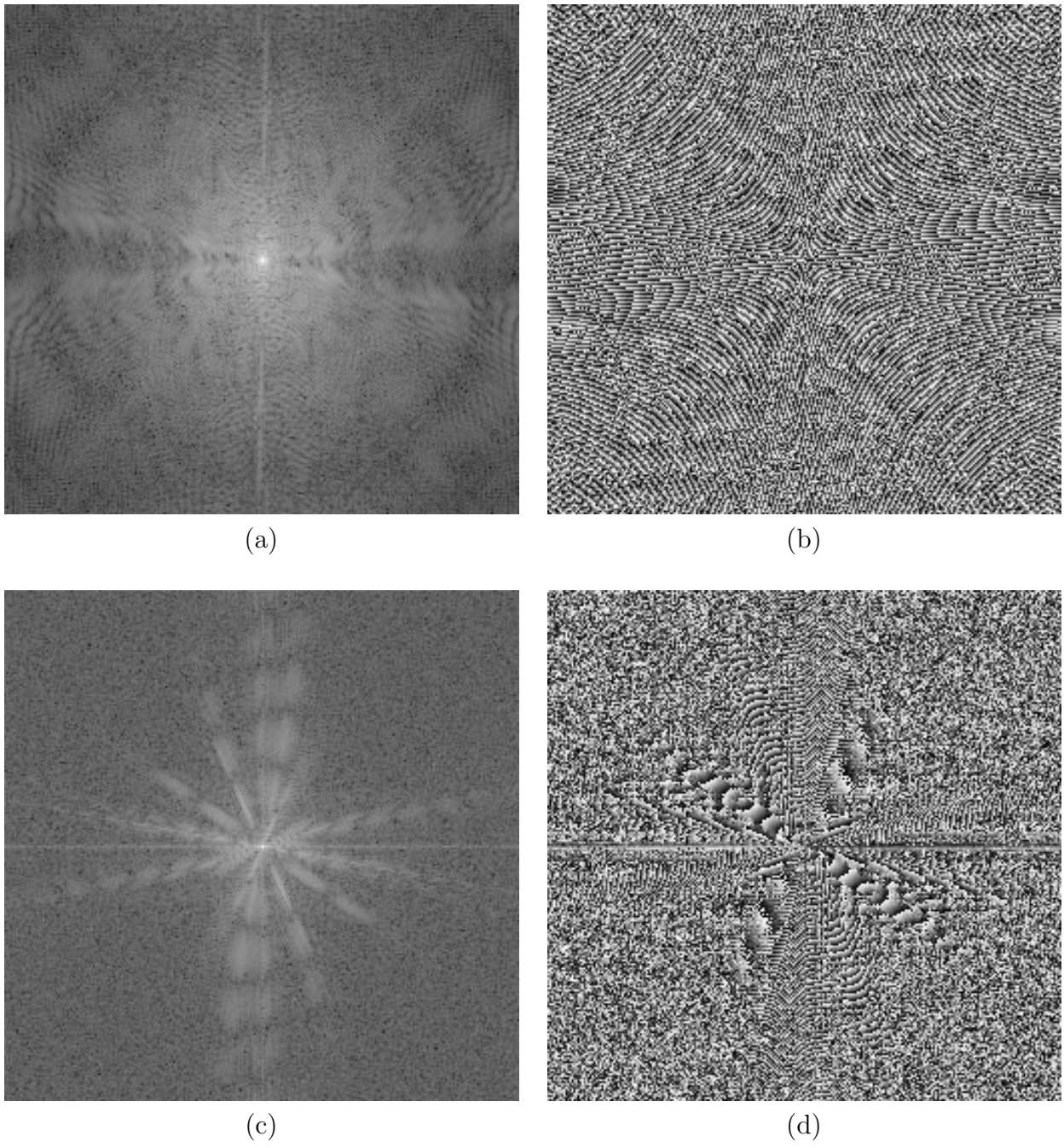
To illustrate the information contained in r and θ separately, let us look at what happens to the images if r is modified before back-transforming to f. Fig 3.6(a) shows the effect on the X-ray image of taking square-roots of all the Fourier amplitudes, and then applying the inverse transform to produce a new image. Fig 3.6(b) is the result of setting all amplitudes equal and then back-transforming. Therefore, these images emphasise the information content of the phases *alone*. In both cases, edges have been emphasised in a way akin to Figs 3.1(c) and (d). (Note that the operations which produced Figs 3.6(a) and (b) are both nonlinear filters.) Contrary to the the impression given in Fig 3.5, the information content is generally greater in θ than in r for Fourier transforms of images. Fig 3.6(c) further illustrates the point. It is the image produced when the cashmere fibre Fourier amplitudes are combined with the X-ray phases. The resemblance to the original X-ray image is much closer than to the cashmere image. Conversely, Fig 3.6(d) has the X-ray amplitude and cashmere phases, and looks more like the cashmere image. Stark (1987, chapters 6-8) considers the problem of reconstructing an image when only phase information, or only amplitude information, is available, which occurs with certain sensors.

Finally in this subsection on the theory of Fourier transforms we will consider computational matters. The Fourier transform is separable. This, combined with the existence of a highly-efficient algorithm, the **Fast Fourier Transform** (**FFT**) (see for example, Chatfield, 1989) leads to very fast implementations of the transform. The column transform

$$f'_{kj} = \sum_{i=1}^{n} f_{ij} \exp\left\{\frac{-2\pi\imath}{n} ik\right\} \qquad \text{for } k = 1, \ldots, n,$$

can be carried-out separately for each column, j, from 1 to n, using a real-only implementation of the FFT. The subsequent row transform

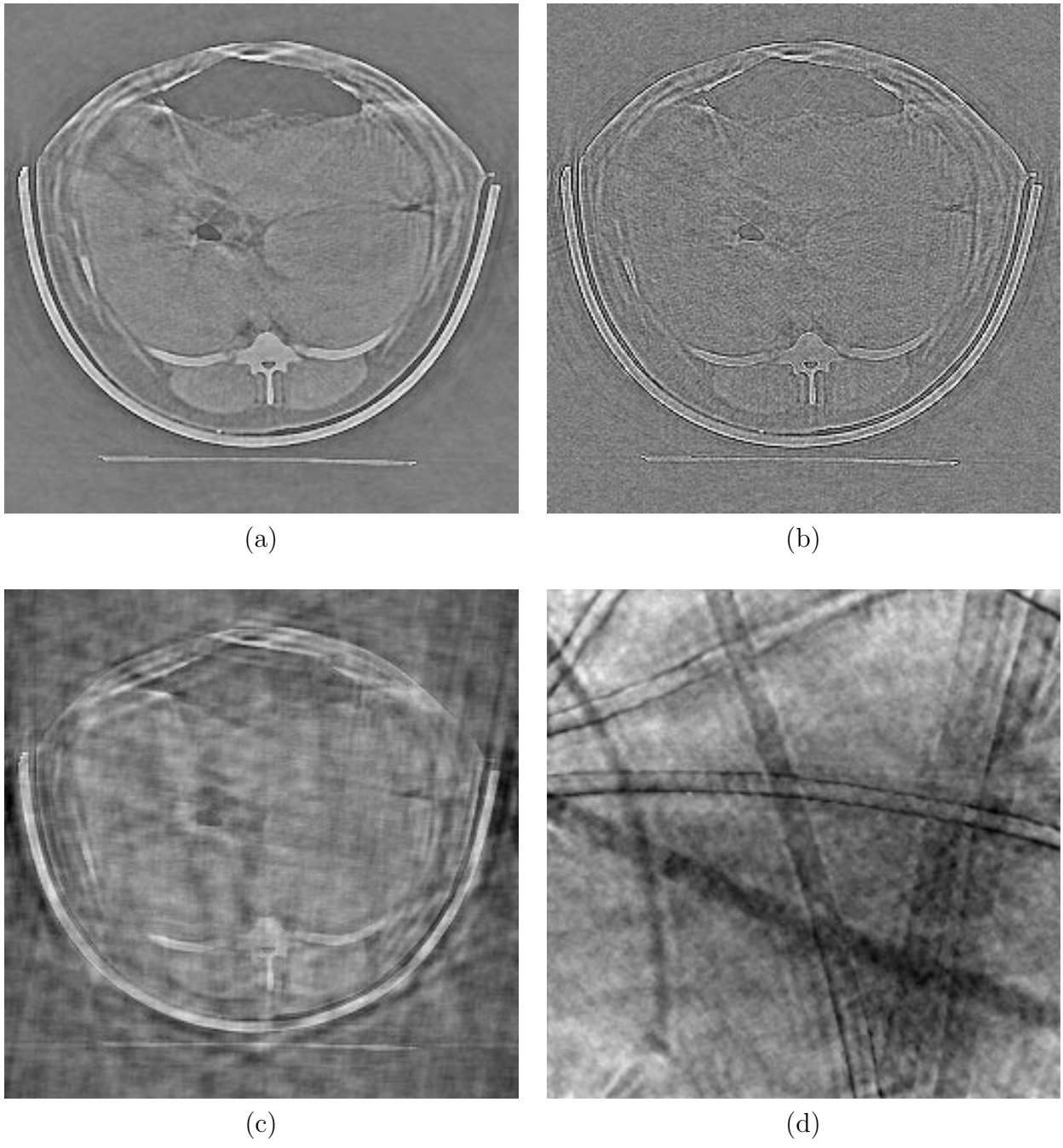
$$f^*_{kl} = \sum_{j=1}^{n} f'_{kj} \exp\left\{\frac{-2\pi\imath}{n} jl\right\} \qquad \text{for } l = 1, \ldots, n,$$



(a) (b)

(c) (d)

Figure 3.5: Fourier transforms of images: **(a)** amplitudes of Fourier transform of X-ray image, **(b)** phases from X-ray image, **(c)** amplitudes of Fourier transform of cashmere image, **(d)** phases from cashmere image.



(a) (b)

(c) (d)

Figure 3.6: Recovered images from Fourier domain after back-transforming: **(a)** X-ray image with Fourier amplitudes replaced by their square-roots, **(b)** X-ray image with constant Fourier amplitudes, **(c)** cashmere amplitudes combined with X-ray phases, **(d)** X-ray amplitudes and cashmere phases.

requires a complex FFT algorithm (that is, one which operates on complex numbers), but need only be performed for $k = \frac{n}{2}, \ldots, n$, because of the structure of f^* . The inverse transform similarly needs $\frac{n}{2}+1$ complex FFTs, but this time followed by n FFTs of Hermitian series, which are a restricted class of complex series. Numerical Algorithms Group (1991) (NAG) routines CO6FPF, CO6FQF and CO6FRF are Fortran77 implementations of such FFTs, which work for any values of n. Yet more efficient FFT algorithms exist, which require n to be a power of 2 or to have only small prime factors. Also, hardware implementations of FFTs exist in some computers.

3.2.2 Filters already considered

Linear filters have a very simple representation in the frequency domain. We need to use a slightly different definition of a filter from that used in §3.1. We will use

$$g_{ij} = \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} w_{kl} f_{i+k,j+l} \qquad \text{for } i, j = 1, \ldots, n,$$

where $(i+k)$ is replaced by $(i+k\pm n)$, if it falls outwith the range 1 to n. A similar transformation is applied to $(j+l)$. In effect, the borders of f are wrapped round to meet each other as though the image lay on the surface of a torus, something we mentioned at the beginning of §3.1. Normally w is non-zero only for (k, l) near (0,0), in which case the difference in results from those obtained using other border conditions will be slight. Remarkably, the Fourier transforms of f, g and w satisfy

$$g^*_{kl} = w^*_{kl} f^*_{kl} \qquad \text{for } k, l = 1, \ldots, n,$$

provided that w and f are arrays of the same size. A filtering operation in the spatial domain has become simply a *point-by-point multiplication* in the frequency domain. Therefore, an alternative way of applying a linear filter, instead of using the methods of §3.1, is to compute Fourier transforms of f and w, multiply their Fourier coefficients and back-transform to obtain g.

If w has 180° rotational symmetry, (that is $w_{kl} = w_{-k,-l}$), as have the moving average, Gaussian and Laplacian filters, then w^* is real. Therefore g^* has the same phases, θ, as f^* — only the amplitudes are different. Another possibility is that w is antisymmetric, (that is $w_{kl} = -w_{-k,-l}$), as were the first-derivative filters. In this case, w^* is purely imaginary. Such filters rescale the amplitudes and add or subtract $\frac{\pi}{2}$ from the phases. Fig 3.7 shows w^* for all the filters used to produce Figs 3.2-4. In the case of the first-derivative filters the imaginary parts of w^* are shown. Zero is shown as mid-grey in each display.

The smoothing filters, that is the moving average and Gaussian, are **low-pass filters**, because these filters only let through low-frequency terms. All high-frequency terms in g^* are almost reduced to zero because w^*_{kl} decays to zero as (k, l) moves away from (0,0). As the extent of smoothing is increased (i.e. as m or σ^2 is increased) the decay is more rapid, therefore progressively more of the higher frequencies are filtered out of the image. The Fourier transform

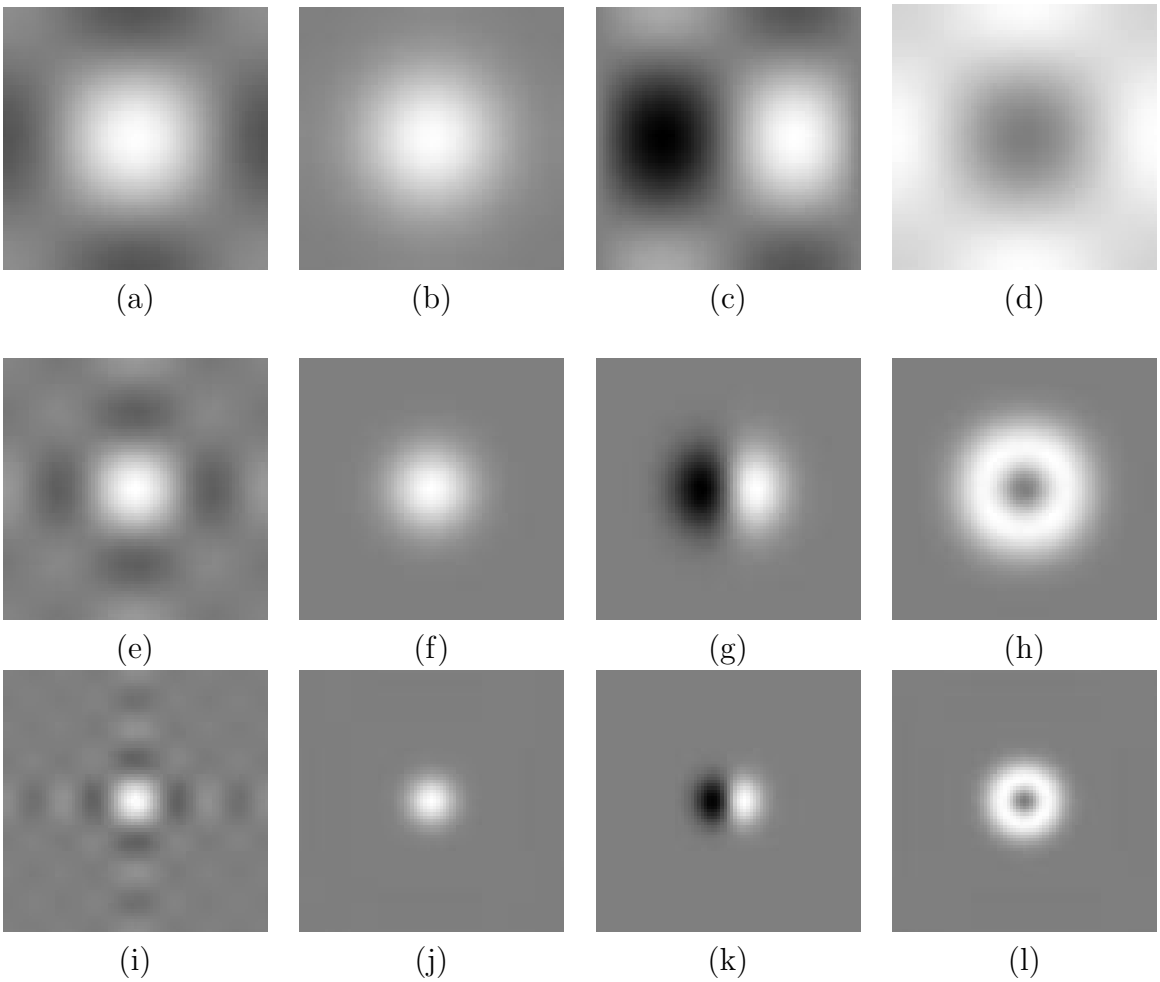


Figure 3.7: Fourier transforms of linear filters used in Figs 3.2-4 (all transforms are real, except for the first-derivative filters, which are purely imaginary. In the latter case, the imaginary term is displayed): **(a)** 3×3 moving average, **(b)** Gaussian, $\sigma^2 = \frac{2}{3}$, **(c)** 3×3 first-derivative row filter, **(d)** Laplacian, **(e)** 5×5 moving average, **(f)** Gaussian, $\sigma^2 = 2$, **(g)** first-derivative row filter after Gaussian, $\sigma^2 = 2$. **(h)** Laplacian-of-Gaussian, $\sigma^2 = 2$, **(i)** 9×9 moving average, **(j)** Gaussian, $\sigma^2 = 6\frac{2}{3}$, **(k)** first-derivative row filter after Gaussian, $\sigma^2 = 6\frac{2}{3}$, **(l)** Laplacian-of-Gaussian, $\sigma^2 = 6\frac{2}{3}$.

of the array of moving average weights is approximately given by

$$w^*_{kl} = w^*_k w^*_l \qquad \text{where } w^*_k = \frac{\sin\{\frac{\pi k}{n}(2m+1)\}}{\frac{\pi k}{n}(2m+1)}.$$

It is not isotropic, and some elements in w are negative. (The expression for w^*_k is sometimes referred to as a *sinc* function.) The Fourier transform of the Gaussian weights can be approximated by

$$w^*_{kl} = \exp\left\{\frac{-(k^2+l^2)}{2\{n/(2\pi\sigma)\}^2}\right\}.$$

It is a scaled version of a Gaussian density, but with a different spread, and is both isotropic and non-negative.

The Laplacian filter, whose Fourier transform is shown in Fig 3.7(d), is a **high-pass filter**, because it filters out only frequencies near zero. (Note that Fig 3.7(d) is the complement of Fig 3.7(a): element w^*_{kl} in one is equal to $1 - w^*_{kl}$ in the other one.) The Laplacian-of-Gaussian filters, Figs 3.7(h) and (l), are **band-pass filters** because they remove both low and high frequencies from an image. Therefore they operate as edge detectors, whilst managing to smooth out some of the noise. The first-derivative row filters are also band-pass filters, as can be seen in Figs 3.7(c), (g) and (k).

For the case where w^* is known (and therefore w does not need Fourier transforming) the computer timings for implementing filters using Fourier transforms are given in Table 3.1. Such implementations would appear to be worthwhile only for filters which are non-separable and bigger than 7×7. Faster times are possible if n is a power of 2 and a specialized FFT algorithm is used. However, if w^* has itself to be obtained by applying an FFT to w, then times will be slower. A further complication in making comparisons between spatial and frequency algorithms is that the former can often be achieved using integer arithmetic, while the latter always need floating-point operations, which are generally slower.

3.2.3 New filters and image restoration

So far in §3.2, we have only reviewed filters previously considered in §3.1. However, it is also possible to formulate filters in the frequency domain: a filter is specified by the array w^*, rather than by the array w. One class of isotropic filters, termed **'ideal' low-pass filters**, is given by setting w^*_{kl} to zero beyond a certain distance (R) from the origin, and otherwise setting it to unity:

$$w^*_{kl} = \begin{cases} 1 & \text{if } k^2 + l^2 < R^2, \\ 0 & \text{otherwise,} \end{cases} \qquad \text{for } k, l = -\frac{n}{2}, \ldots, \frac{n}{2} - 1.$$

'Ideal' high-pass filters are defined in a complementary way by setting w^*_{kl} to zero within a distance (R) from the origin:

$$w^*_{kl} = \begin{cases} 0 & \text{if } k^2 + l^2 < R^2, \\ 1 & \text{otherwise,} \end{cases} \qquad \text{for } k, l = -\frac{n}{2}, \ldots, \frac{n}{2} - 1.$$

Fig 3.8 shows the results when 'ideal' low-pass filters are applied to the X-ray image for $R = 60$, 30 and 15 — values which approximately match the extents of smoothing in Fig 3.2. A ripple or '*ringing effect*' is evident, particularly in Fig 3.8(b), which can be explained by examining the weights w of the equivalent spatial filter. Because w has radial symmetry we need only look at elements in a single row. For $R = 60$:

$l =$	0	1	2	3	4	5	6	7	8	9
$1000 \times w_{0l} =$	175	131	41	-17	-16	5	10	-1	-6	0

The negative weights are the cause of the ringing. It is the same phenomenon by which moving average filters produced ringing effects in the Fourier domain (see, for example, Fig 3.7(i)).

The Gaussian filter is in some ways a compromise between these filters — its weights remain positive but decrease smoothly in both spatial and frequency domains. Wang, Vagnucci and Li (1983) survey alternative, smooth specifications of w^* .

If an image has been contaminated by noise and blurring of known forms, then filters can be constructed which optimally (in some sense) restore the original image. Some authors regard **restoration** as being distinct from image filtering and devote whole chapters to it (see for example Rosenfeld and Kak, 1982, Gonzalez and Wintz, 1987). There are both linear and nonlinear restoration methods. Here we will consider only the fundamental linear method, called the **Wiener filter**.

The observed image, f, is assumed to be a blurred version of an underlying 'true' image, g, with noise added. We envisage that g is a clear image, unaffected by noise and with distinct boundaries between objects, whereas in f the edges are spread over several pixels, and there is noise. We can express the degradation process as

$$f_{ij} = \sum_{k=-\frac{n}{2}}^{\frac{n}{2}-1} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}-1} v_{kl} g_{i+k,j+l} \quad + \quad e_{ij} \qquad \text{for } i,j = 1,\ldots,n,$$

where v denotes the weights by which g is blurred and e denotes the noise.

We can use information about the nature of the degradations to design a filter which will smooth f and enhance the edges, so as to get as close as possible to restoring g. Provided that we can consider g to be the realisation of a random process, and subject to various technical conditions, the Wiener filter is the best linear predictor (in the least-squares sense) of g from f. The restored image, $\hat{g}$, has Fourier transform:

$$\hat{g}^*_{kl} = \frac{f^*_{kl}}{v^*_{kl}} \quad \frac{|v^*_{kl}|^2}{|v^*_{kl}|^2 + S^e_{kl}/S^g_{kl}} \qquad \text{for } k,l = -\frac{n}{2},\ldots,\frac{n}{2}-1.$$

For a derivation, see for example Rosenfeld and Kak (1982, §7.3). S^g_{kl} denotes the **spectrum** of g at frequency (k,l), that is the expected value of the squared amplitude of g^*_{kl}, after scaling:

$$S^g_{kl} = \frac{1}{n^2}\ \mathrm{E}(|g^*_{kl}|^2),$$

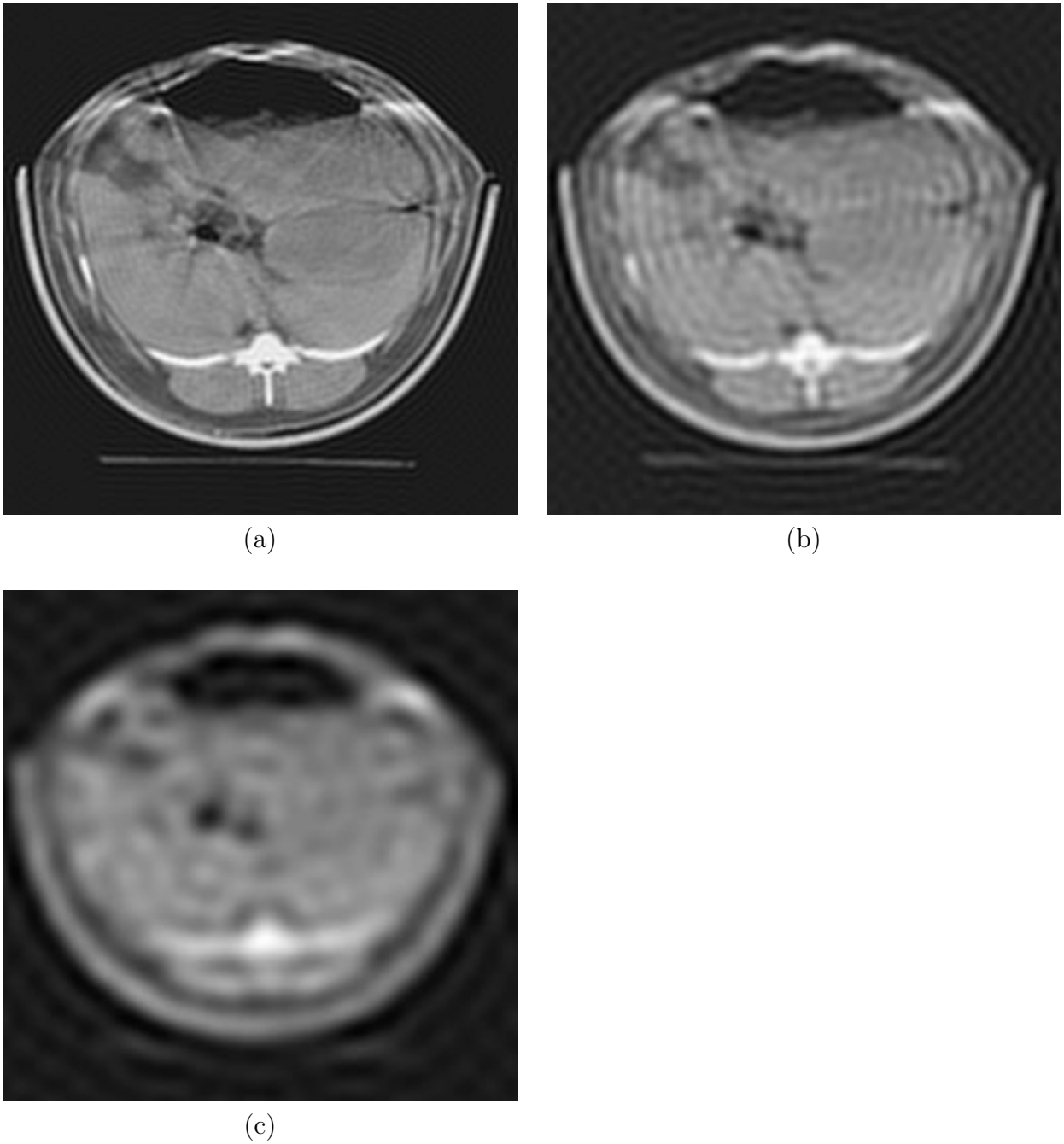
Similarly, the array S^e denotes the spectrum of e. S^g can alternatively be expressed as the Fourier transform of the **autocovariance function**, C^g, of g, that is

$$S^g = C^{g*},$$

where C^g_{kl} denotes the covariance between pixel values in g which are k rows and l columns apart:

$$C^g_{kl} = \ \mathrm{E}\{(g_{ij} - \bar{g})(g_{i+k,j+l} - \bar{g})\}$$

and $\bar{g}$ denotes the mean pixel value in g (see for example Chatfield, 1989).



(a) (b)

(c)

Figure 3.8: 'Ideal' low-pass filters applied to X-ray image: **(a)** $R = 60$, **(b)** $R = 30$, **(c)** $R = 15$.

In the *absence of blur*, $v^*_{kl} = 1$ for all values of k and l, and the Wiener filter simplifies to

$$\hat{g}^*_{kl} = \frac{f^*_{kl}}{1 + S^e_{kl}/S^g_{kl}},$$

which is typically a low-pass filter, also known in the spatial domain as **simple kriging** (Cressie, 1991, p 110). In the *absence of noise*, $S^e_{kl} = 0$ for all values of k and l, and the Wiener filter simplifies to a high-pass filter:

$$\hat{g}^*_{kl} = \frac{f^*_{kl}}{v^*_{kl}}$$

which reverses the blurring process. However, in practice this filter *never works* because there are always some values of v^*_{kl} which are close to zero. The presence of noise means that the highest-frequency components of g cannot be recovered from f, and the filter is band-pass.

To illustrate the low-pass case, consider the log-transformed SAR image (Fig 2.6). The speckle noise has known variance of 0.41 (§1.1.3) and zero autocovariances at other lags. Therefore the noise spectrum is

$$S^e_{kl} = 0.41 \qquad \text{for all } k, l.$$

There is assumed to be no blur, so S^g can be obtained either by estimating $(S^f - S^e)$, or indirectly by estimating C^g, which is equal to $(C^f - C^e)$. We will follow the latter approach. The sample autocovariances of f are given in Table 3.2. They were computed efficiently by taking the inverse Fourier transform of $\frac{1}{n^2}|f^*|^2$. Once a value of 0.41 (due to C^e) has been subtracted at $(k, l) = (0, 0)$, the statistics are consistent with an isotropic model with autocovariance function:

$$C^g_{kl} = 0.75 \times 0.9^{\sqrt{k^2+l^2}}$$

(Horgan, 1994). When these expressions are used in the Wiener filter, it simplifies to an isotropic spatial filter with weights:

$l =$	0	1	2	3
$1000 \times w_{0l} =$	204	77	24	6

The result of applying the filter to the log-transformed SAR image is shown in Fig 3.9. This filter is the best compromise in linear filters between a large window to smooth the speckle noise and a small window to avoid blurring edges such as field boundaries. It should, however, be borne in mind that although the Wiener filter is the optimal linear one, nonlinear filters to be considered in §3.3 can out-perform it.

To illustrate **deconvolution**, that is reversal of the blurring process, we will consider the DNA image and adopt an approach similar to that of Press, Flannery, Teukolsky and Veltering (1986, pp 417-420). Fig 3.10(a) shows a single column of the image (i.e. of f) after inversion of pixel

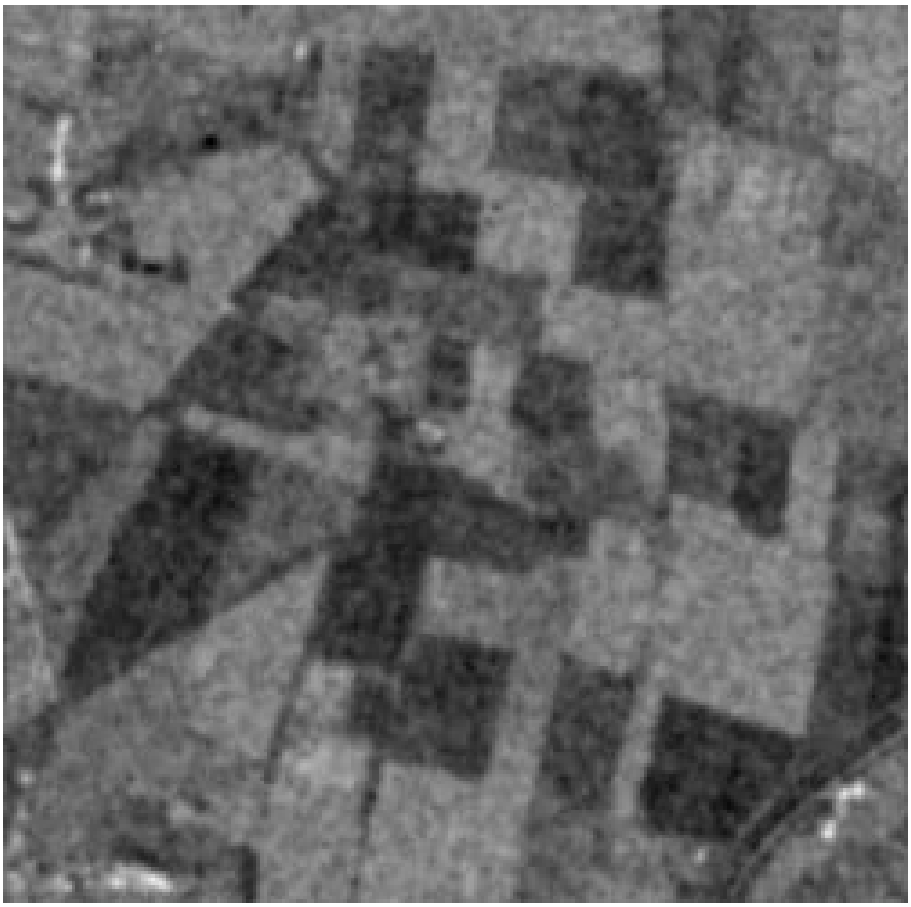


Figure 3.9: Wiener filter applied to SAR image.

values and subtraction of a linear background trend. Each band on the gel appears in this plot as a peak spread over several pixels, whereas the true pattern (g) is assumed to be distinct single-pixel-width spikes as shown in Fig 3.10(b), which correspond to the local maxima in Fig 3.10(a) above a threshold. The log-transformed amplitudes of the 1-D Fourier transforms of Figs 3.10(a) and (b), that is $\log|f^*|$ and $\log|g^*|$, are shown plotted against frequency in Figs 3.10(c) and (d), respectively. Fig 3.10(e) shows the phase spectrum, which is the difference in phases between f^* and g^* , modulo 2π. Phase differences are small at frequencies below 0.7, but above this limit, noise appears to dominate. The log-ratio, $\log(|f^*|/|g^*|)$, plotted in Fig 3.10(f), can be used to estimate $|v^*|$ at frequencies below 0.7. On a log-scale a quadratic function fits well, with

$$\log|v_k^*| = 1.19 - 3.13\left(\frac{\pi k}{n}\right)^2 \qquad \text{for } \left(\frac{\pi k}{n}\right) < 0.7.$$

The fitted curve is also displayed in Fig 3.10(f). This corresponds to the blur, v, being Gaussian. From Fig 3.10(d), the pattern underlying $|g^*|$ appears to be uniform, with a mean value of 5.89 on a log-scale, as plotted. We conclude that the spectrum of g is a constant:

$$S^g = \frac{1}{256}(e^{5.89})^2 = 510.$$

We shall assume, in the absence of any other information, that the noise is uncorrelated, so-called **white noise**, as was the SAR speckle noise. Its variance can be estimated from the high frequencies of f^*. The mean value of $\log|f^*|$, for frequencies greater than 1.0, is 4.05. This is plotted on Fig 3.10(c). It leads to the spectrum of e being

$$S^e = 12.9.$$

Fig 3.10(g) shows the result of applying the Wiener filter to the data in Fig 3.10(a), using the expressions identified above. The spread of the peaks has been reduced, but at the cost of a more variable signal.

If the 1-D model is assumed to apply isotropically in 2-D, the restored image after Wiener filtering is as given in Fig 3.11. Fig 3.12 shows the effect on the restoration of a small part of the DNA image of varying the noise-to-signal ratio, S^e/S^g. Fig 3.12(a) shows the original image, in which four bands are just discernible. Fig 3.12(c) shows the optimal restoration, and the four bands appear clearly. Fig 3.12(b) shows the restoration if S^e/S^g is ten times too small. Here, too many high frequencies have been used and noise dominates the result. Fig 3.12(d) shows the restoration with S^e/S^g ten times too big. Not as much deblurring has been performed as is possible. For the optimal isotropic filter:

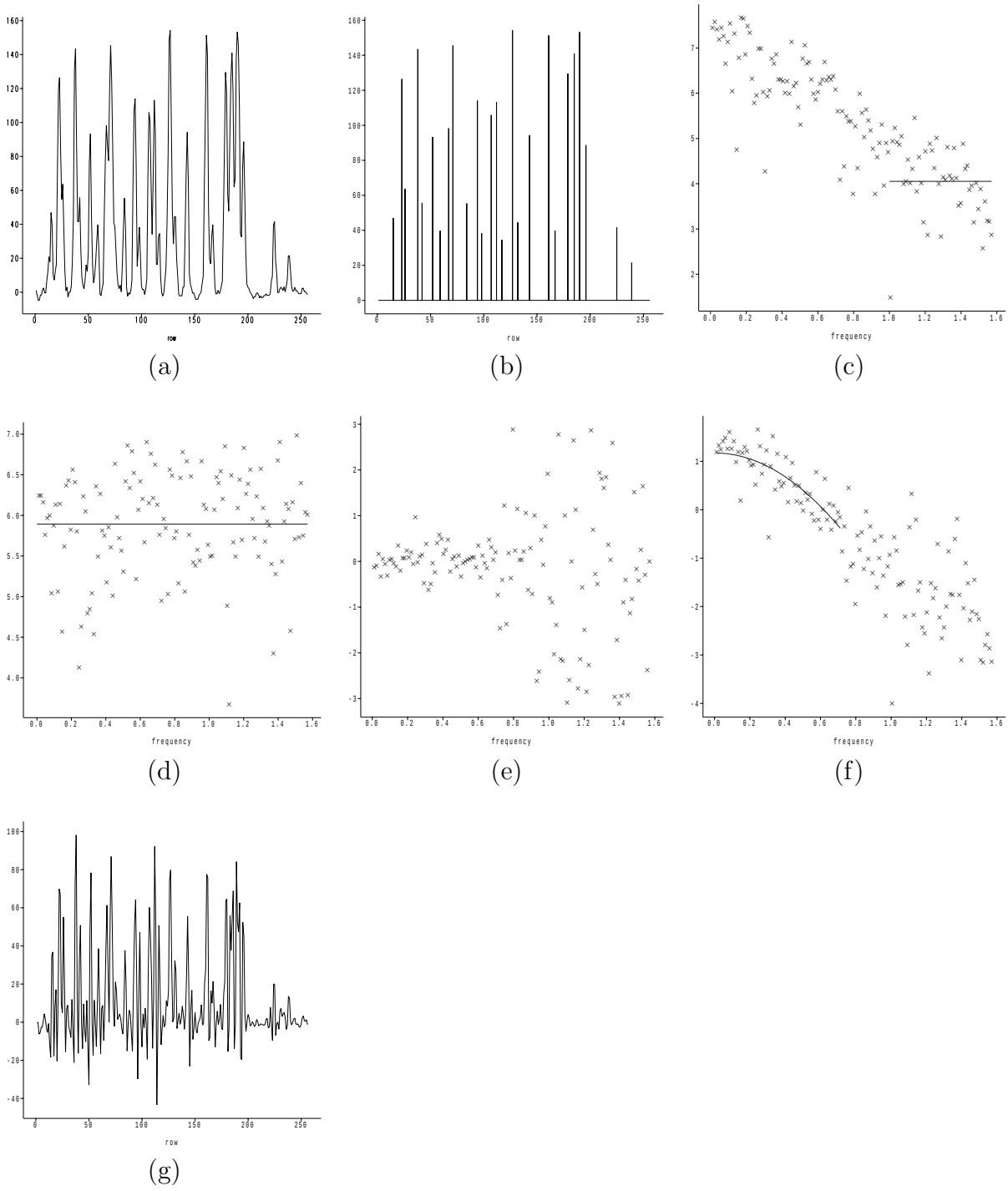
$l =$	0	1	2	3	4	5	6	7	8
$1000 \times w_{0l} =$	323	79	-82	20	14	-13	2	3	-2

Note that some weights are negative, as they have to be in a band-pass filter. These produce the ripple effects in the restored image, such as the white borders discernible around each band in Fig 3.11.

Finally in this section, note that nonlinear restoration algorithms can do better than linear ones, but require substantially more computation. For example, maximum entropy restoration (Skilling and Bryan, 1984) is one method which exploits the constraint that g is non-negative, which is appropriate to the DNA image after removal of background trend and greyscale inversion. However, see also Donoho, Johnstone, Hoch and Stern (1992) for a discussion of alternative methods. It is also possible to apply deconvolution algorithms to three-dimensional images, to reduce blurring such as occurs in fluorescence microscopy (Shaw and Rawlins, 1991). These topics are all areas of active research and further discussion of them is beyond the scope of this book.

3.3 Nonlinear smoothing filters

In filtering to reduce noise levels, linear smoothing filters inevitably blur edges, because both edges and noise are high-frequency components of images. Nonlinear filters are able to *simultaneously* reduce noise and preserve edges. However:



(a) (b) (c)

(d) (e) (f)

(g)

Figure 3.10: Interpretation of single column of DNA image in order to identify degradation model required by Wiener filter: **(a)** pixel values after removal of linear trend and inverting greyscale, **(b)** idealised version of (a), **(c)** log-transformed amplitudes of Fourier transform of (a) plotted against frequency, together with mean value at higher frequencies, **(d)** log-transformed amplitudes of Fourier transform of (b), together with mean value, **(e)** phase differences between Fourier transforms of (a) and (b), **(f)** log-transformed ratios of amplitudes of Fourier transform of (a) and (b), together with estimated quadratic function, **(g)** result of applying Wiener filter to data in (a).

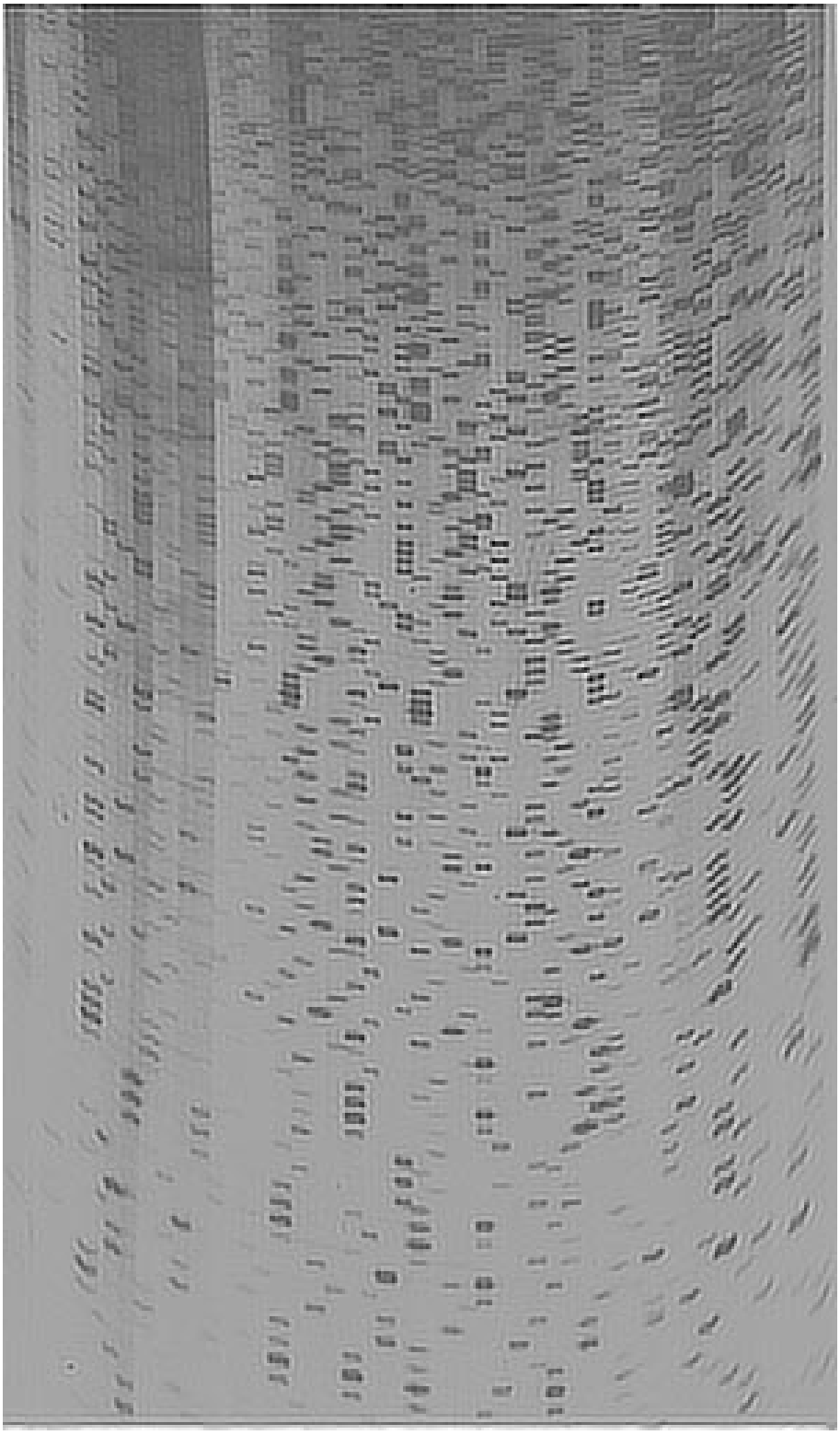
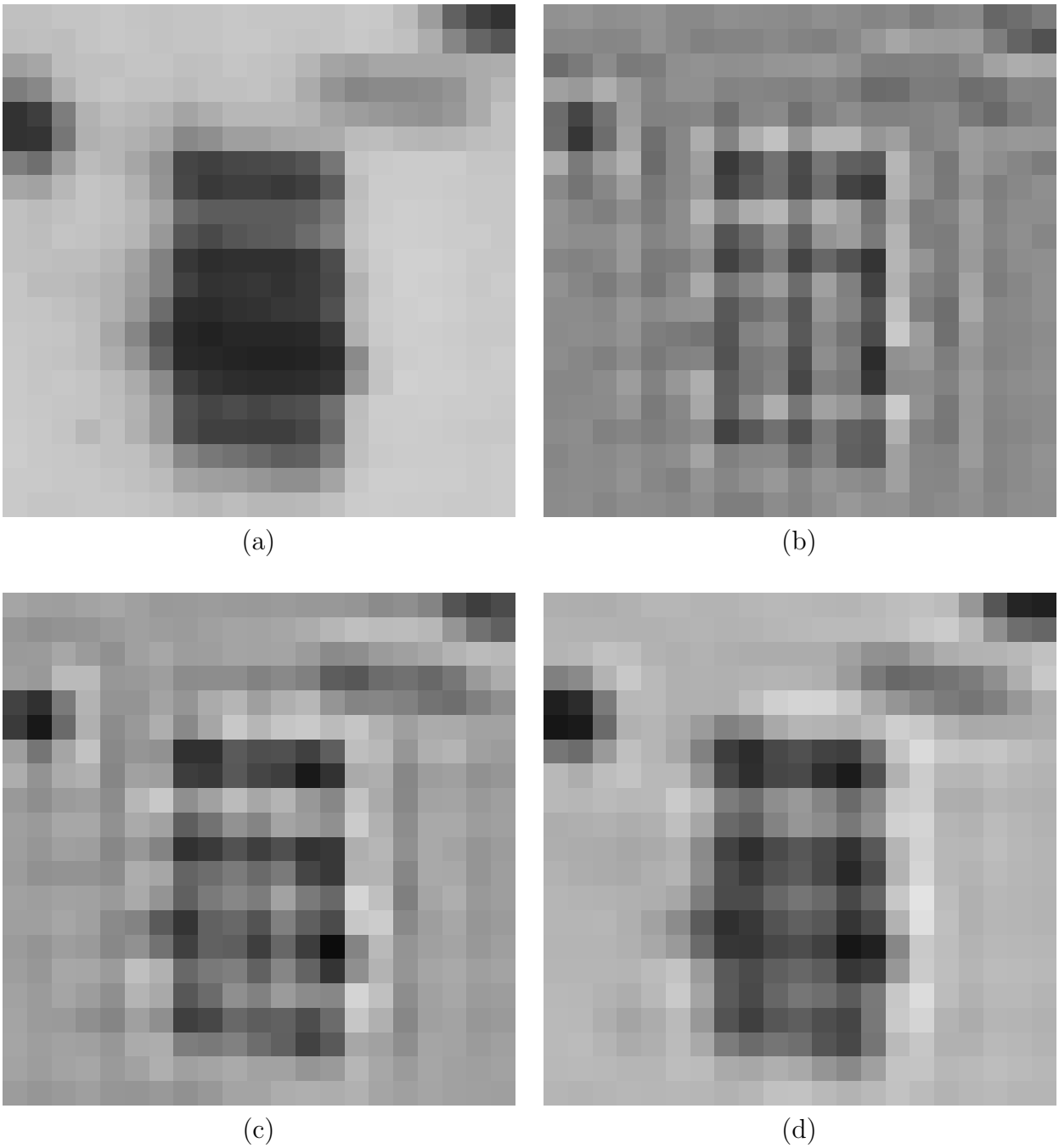


Figure 3.11: Optimal linear deconvolution (Wiener filter) to restore DNA image.



(a) (b)

(c) (d)

Figure 3.12: Detail from DNA image: **(a)** original, **(b)** sub-optimal restoration assuming too little noise, **(c)** optimal restoration, as in Fig 3.11, **(d)** sub-optimal restoration assuming too much noise.

- there are a bewildering number of filters from which to choose,
- they can be computationally expensive to use,
- they can generate spurious features and distort existing features in images.

Therefore they should be used with caution.

The simplest, most studied and most widely used nonlinear filter is the moving median. It will be considered in §3.3.1, together with similar filters based on the histogram of pixel values in a neighbourhood. In §3.3.2, a few of the many filters will be presented which make use of the spatial distribution of pixel values in a neighbourhood.

3.3.1 Histogram-based filters

The **moving median filter** is similar to the moving average filter, except that it produces as output at a pixel the median, rather than the mean, of the pixel values in a square window centred around that pixel. (The **median** of a set of numbers is the central one, once they have been sorted into ascending order. For example, the median of {1,0,4} is 1, whereas the mean is $\frac{5}{3}$.) For a filter of size $(2m+1) \times (2m+1)$ the output is

$$g_{ij} = \text{ median } \{f_{i+k,j+l} \ : \ k,l = -m,\ldots,m\} \qquad \text{for } i,j = (m+1),\ldots,(n-m).$$

Fig 3.13 shows a 1-D illustration of both moving average and moving median filters of width 19 applied to part of a row of pixels from the muscle fibres image. Both filters have reduced the noise level, the moving average more efficiently than the median, but the median filter has preserved the step-edge while the moving average filter has blurred it into a ramp of intermediate values.

An easily programmed way of computing the moving median filter would be to sort the set of pixel values $\{f_{i+k,j+l} \ : \ k,l = -m,\ldots,m\}$ into ascending order, then assign to g_{ij} the middle ranked value, and do this independently for every value of i and j from $(m+1)$ to $(n-m)$. This approach is computationally slow and grossly inefficient. Huang, Yang and Tang (1979) presented a fast recursive algorithm for applying the median filter provided that the number of intensity levels of the image is not too great (no greater than 256 for example). It is based on a local histogram which is updated from one neighbourhood of pixel values to the next and from which the median can be extracted without having to sort the set of $(2m+1)^2$ pixel values. Their algorithm is as follows.

For each row of the image, $i = (m+1),\ldots,(n-m)$, perform steps 1 to 9:

1. Set $j = m+1$
2. Calculate the histogram of $\{f_{i+k,j+l} \ : \ k,l = -m,\ldots,m\}$

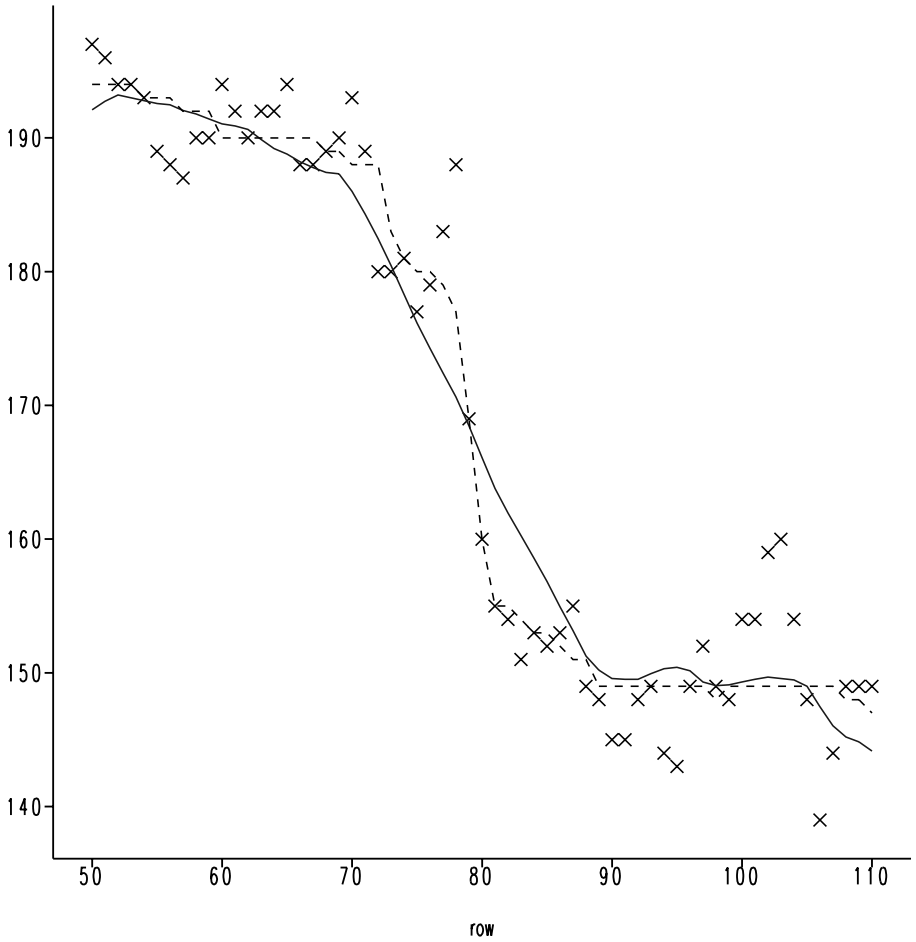


Figure 3.13: Part of a row of pixels (×) from muscle fibres image, together with moving average (———) and moving median (- - - - -) filters of width 19.

3. Set M = median value in the histogram

   and N = number of pixels in histogram which are less than M

   (Note that M and N are simple to compute from the histogram.)

4. Increase j by 1

5. Update the histogram

   to include $\{f_{i+k,j+m} \ : \ k = -m, \ldots, m\}$

   and exclude $\{f_{i+k,j-m-1} \ : \ k = -m, \ldots, m\}$

   and simultaneously update N to remain equal to the number of pixels in the histogram which are less than M

6. If $N > \frac{1}{2}(2m+1)^2$

   then reduce M by 1, update N and repeat this step as many times as is necessary until the inequality is no longer satisfied, then go to step 8.

7. If N + (number of pixels equal to M) $< \frac{1}{2}(2m+1)^2$

   then increase M by 1, update N and repeat this step as many times as is necessary until the inequality is no longer satisfied.

8. Set $g_{ij} = M$

9. Return to step 4 if $j < n - m$.

Computer timings, given in Table 3.1, show that with this algorithm the median filter is as fast as linear filters to compute. (Note that it can be applied equally efficiently in a circular window, or any other convex region. The algorithm can also be adapted to find neighbourhood minima and maxima, which will be used in §3.4). Juhola, Katajainen and Raita (1991) review other algorithms, which are appropriate when there are more than 256 intensity levels.

Figs 3.14(a), (c) and (e) show the effects of median filters of sizes 3×3, 5×5 and 9×9 on the X-ray image. Larger windows produce more smoothing, and fine features are lost, but not to the same extent as with the linear filters in Fig 3.2.

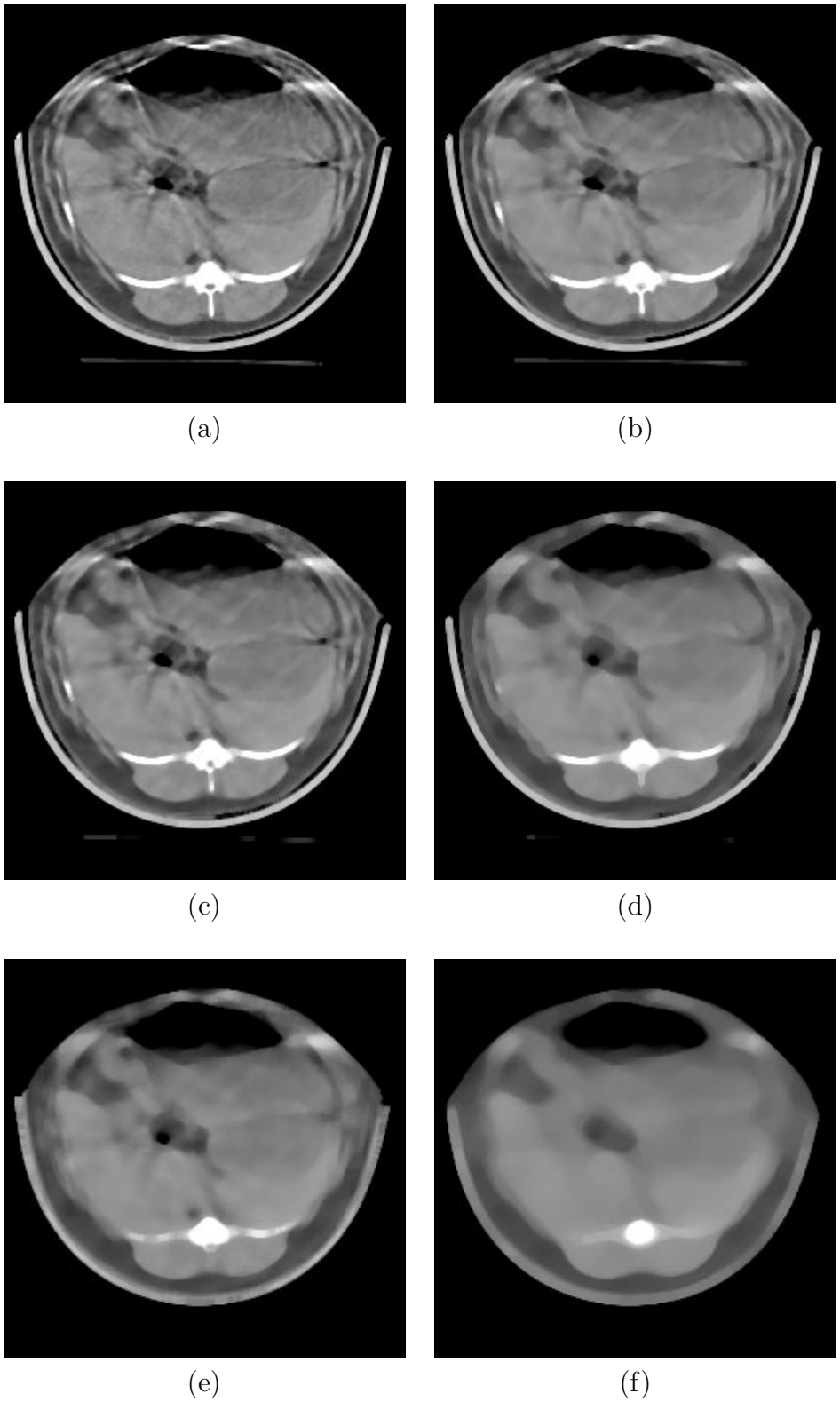
Nonlinear filters are not additive: repeated application of a median filter is not equivalent to a single application of a median filter using a different size of window. By making repeated use of a nonlinear smoothing filter with a small window, it is sometimes possible to improve on noise reduction while retaining fine details which would be lost if a larger window was used. Figs 3.14(b), (d) and (f) show the results after 5 iterations of the median filters corresponding to Figs 3.14(a), (c) and (e) respectively. Note the 'mottling effect' produced by regions of constant or near-constant pixel value, which are an inevitable consequence of median filtering and can lead to the identification of spurious boundaries.

Fig 3.15(a) shows 10 iterations of a 3×3 median filter applied to the log-transformed SAR image. Typically, each iteration produces less change in the image than the previous one until, after (potentially) many iterations, a stable image is found which is unaffected by the median filter. However, iterated median filters have been found to produce artefacts with some images, and also to distort edges, so it is advisable to perform at most a few iterations.

In the past 30 years, statisticians have considered many **robust estimators** of the centre of a histogram, of which the median is but one. These estimators are robust in the sense that they are less affected than the mean by the presence of extreme values in the distribution. Moreover, the median has been shown to be outperformed by some other estimators, any of which could be used as an image filter thus:

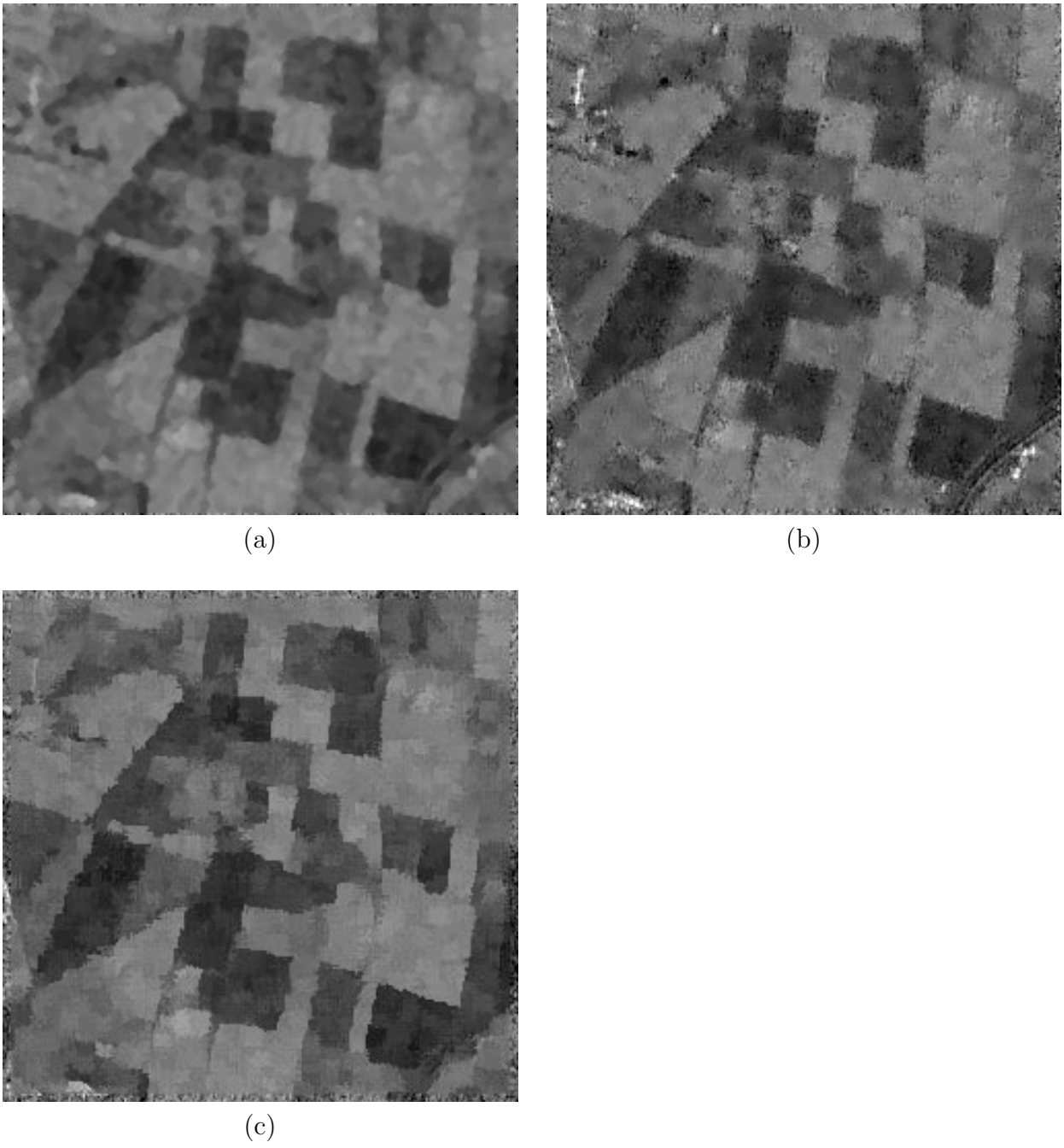
$$g_{ij} = \text{ robust estimator of centre of } \{f_{i+k,j+l} \ : \ k,l = -m,\ldots,m\}.$$

For example, a **trimmed mean** could be obtained by averaging the 50% of pixel values nearest to the median in each window. This should preserve edges, like the median, but should smooth more effectively. It can be programmed efficiently by making further use of the local histogram, using the kind of method described by Mitchell and Mashkit (1992). Because the speed of the



(a) (b) (c) (d) (e) (f)

Figure 3.14: Median filters applied to X-ray image: **(a)** 3×3 filter, **(b)** 5 iterations of 3×3 filter, **(c)** 5×5 filter, **(d)** 5 iterations of 5×5 filter, **(e)** 9×9 filter, **(f)** 5 iterations of 9×9 filter.



(a) (b)

(c)

Figure 3.15: Nonlinear smoothing filters applied to SAR image: **(a)** 10 iterations of 3×3 median filter, **(b)** 7×7 Lee's 1981 filter, **(c)** 9×9 minimum variance filter with 5×5 subwindows.

algorithm is image dependent, it was run on all the 512×512 images used in the book. Table 3.1 reports the range of times taken by a SUN Sparc2 computer for a range of window sizes and $n \times n$ subsets of the full images. The muscle fibres image usually took longest and the Landsat band 2 took the least time. Such robust estimators, many of which are reviewed by Fong, Pomalaza-Raez and Wang (1989) are inevitably slower than the median filter. The computational burden is even further increased if iterations are involved. Also the improvement over the median filter is often very small in practice.

3.3.2 Spatially-adaptive filters

Median filters can be improved upon by making use, not only of the local histogram of values, but also of the spatial distribution of pixel values. Since this information is available it seems sensible to make use of it and not simply form a histogram.

One class of filters just makes a distinction between the *central pixel* and the rest of the window. For example, as a variant on the robust method in the preceding subsection, the mean could be formed of the 50% of pixel values which are nearest f_{ij} in value. This is a case of **k-neighbours filters**, proposed by Davis and Rosenfeld (1978), and efficiently implemented by Mitchell and Mashkit (1992), in which in general the k pixel values nearest to f_{ij} are averaged. Computer times are almost identical to those for the robust method described in the previous subsection. Alternatively, pixels which differ from f_{ij} by less than the average intensity difference from f_{ij} could be averaged (Asano and Yokoya, 1981), or pixels could be weighted in inverse proportion to this difference (*et al*, 1981). If window size is varied, and filters are iterated, a vast number of filters may be generated.

If the noise variance (τ^2 say) is known, as with the SAR image, or can be estimated from homogeneous regions in an image, then use can be made of it. Lee (1983) proposed a filter in which smoothing is only performed where the output from a moving average filter is *consistent* with the original pixel value. Consistency is defined to be an output value within two standard deviations of f_{ij}. Specifically:

$$g_{ij} = \begin{cases} \bar{f}_{ij} & \text{if } |\bar{f}_{ij} - f_{ij}| < 2\tau, \\ f_{ij} & \text{otherwise,} \end{cases}$$

where $\bar{f}_{ij}$ denotes output from a moving average filter of size $(2m+1) \times (2m+1)$. In another paper, Lee (1981) proposed a variant on this idea, in which a weighted average is formed of f_{ij} and $\bar{f}_{ij}$:

$$g_{ij} = \bar{f}_{ij} + (f_{ij} - \bar{f}_{ij}) \frac{(S_{ij}^2 - \tau^2)}{S_{ij}^2},$$

where S_{ij}^2 is the sample variance in the window, that is

$$S_{ij}^2 = \frac{1}{(2m+1)^2 - 1} \left\{ \sum_{k=-m}^{m} \sum_{l=-m}^{m} f_{i+k,j+l}^2 \quad - \quad (2m+1)^2 \bar{f}_{ij}^2 \right\},$$

provided this exceeds τ^2, and S_{ij}^2 is otherwise set to τ^2 in order to ensure that $(S_{ij}^2 - \tau^2)$ is non-negative. (Note that the summation term can be obtained efficiently by applying the moving

average algorithm to an image with intensities f_{ij}^2 .) Therefore, in an edge-free neighbourhood, $S_{ij}^2 \approx \tau^2$ and $g_{ij} \approx \bar{f}_{ij}$, whereas near edges, $S_{ij}^2 \gg \tau^2$ and $g_{ij} \approx f_{ij}$. The filter is derived as the minimum mean-square-error predictor of g_{ij} if f_{ij} has mean g_{ij} , variance τ^2 , and g_{ij} has mean $\bar{f}_{ij}$, variance $(S_{ij}^2 - \tau^2)$. Durand, Gimonet and Perbos (1987) compared several filters for smoothing SAR images and found this to be one of the best. The result is shown in Fig 3.15(b) for a 7×7 window, which is the minimum size of window which can be used and still produce a reasonably precise estimate of S^2 . Notice that the centres of fields are smoothed, but the edges are left speckly. Also, bright spots on the top left and bottom right of the image remain unsmoothed, unlike in Figs 3.15(a) and (c).

The final class of filters we shall consider make greater use of the spatial distribution of pixels. Lev, Zucker and Rosenfeld (1977) proposed two sets of adaptive weights for a 3×3 window, which allow for the presence of edges and corners. Harwood, Subbarao, Hakalahti and Davis (1987) suggested forming a symmetric nearest-neighbours mean, by using the pixel value nearer f_{ij} for each pair of pixels symmetrically opposite in the window.

An elegant and fast method, originally proposed in a slightly different form by Tomita and Tsuji (1977), is the **minimum variance filter**. For this filter, the mean $(\bar{f})$ and variance (S) are evaluated in five $(2m+1) \times (2m+1)$ subwindows within a $(4m+1) \times (4m+1)$ window, and the filter output is defined to be the mean of that subwindow which has the smallest variance. Therefore

$$g_{ij} = \bar{f}_{kl} \qquad \text{where } (k,l) = \text{ argmin } \{S_{ij}^2, S_{i-m,j-m}^2, S_{i+m,j-m}^2, S_{i-m,j+m}^2, S_{i+m,j+m}^2\},$$

and 'argmin' means that $(k,l) = (i-m, j-m)$ if $S_{i-m,j-m}^2$ is the smallest of the five S^2s in the set, $(k,l) = (i+m, j-m)$ if $S_{i+m,j-m}^2$ is the smallest of the five S^2s, etc. The times taken by the computer to apply the filter are again given in Table 3.1. Times are unaffected by window size because they make use of the moving average algorithm given in §3.1.1.

The output produced when the minimum variance filter is applied to the SAR image, with $m = 2$, is shown in Fig 3.15(c). Areas within fields have been smoothed effectively, although there appears to be some distortion of field boundaries. Square windows also have the drawback that they blur corners, that is, where edges meet at acute angles. Nagao and Matsuyama (1979) proposed a filter using essentially the same idea but with other shapes of window, and at the cost of a much more computationally-intensive algorithm.

For descriptions of yet more filters, and comparative studies of performance, the interested reader is referred to the review papers of Chin and Yeh (1983), Wang, Vagnucci and Li (1983), Mastin (1985), Fong *et al* (1989), Imme (1991) and Wu, Wang and Liu (1992). Unfortunately, the reviews are ultimately inconclusive and contradictory, because images can be of such varying types and there is no unique measure of a filter's success.

3.4 Nonlinear edge-detection filters

Edges are boundaries between objects, or parts of objects, in images. An edge filter should produce a large response at a location if pixels in the neighbourhood show a systematic pattern of changes in value. Linear filters, considered in §3.1.2, are of use as directional edge filters, but the Laplacian filter is the only one we have considered up to now which can detect edges at any orientation. In §3.4.1 we shall consider some simple, nonlinear filters for detecting edges in all directions, before taking a more systematic approach based on estimated derivatives in §3.4.2.

3.4.1 Simple edge filters

The **variance filter** (S^2), already used in §3.3.2 in conjunction with Lee's filters, is a measure of edge presence which can be calculated very quickly. The result of its application to the X-ray image is shown in Fig 3.16(a). The standard deviation rather than variance is displayed, with large values shown as darker pixels. As may be expected, neighbourhoods containing edges have larger variances, but the noise in the original image can also be seen in the filter output.

The **range filter** produces as output the range of pixel values in a window, that is the difference between the maximum and minimum values:

$$g_{ij} = \max\{f_{i+k,j+l} \ : \ k,l = -m,\ldots,m\} \quad - \quad \min\{f_{i+k,j+l} \ : \ k,l = -m,\ldots,m\}.$$

This is also a quick and simple filter, which can be obtained by modifying the moving median algorithm of §3.3.1. Alternatively, the filter can be implemented by making use of the property that the 'max' and 'min' filters are both separable (see §3.1.1). The result, in Fig 3.16(b), is very similar to the variance filter for the X-ray image.

Roberts' filter is defined as the sum of the absolute differences of diagonally opposite pixel values:

$$g_{ij} = |f_{ij} - f_{i+1,j+1}| + |f_{i+1,j} - f_{i,j+1}|.$$

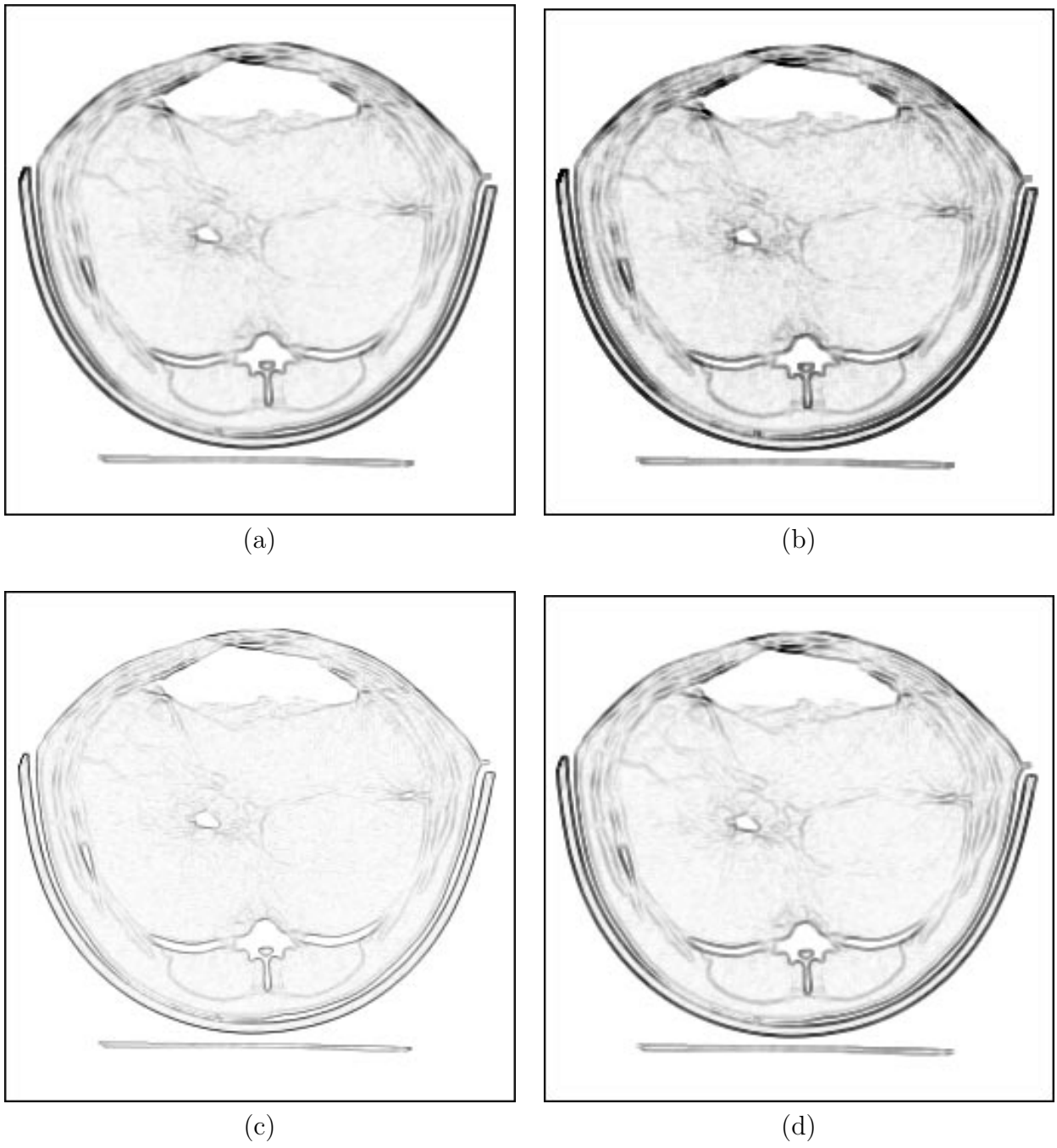
Because the window size is only 2×2, edges produce finer lines in the output, as illustrated in Fig 3.16(c).

The **Kirsch filter** is one of a family of **template matching filters** (for others, see Jain, 1989, §9.4) in which the filter output is the maximum response from a set of linear filters which are sensitive to edges at different orientations:

$$g_{ij} = \max_{z=1,\ldots,8} \sum_{k=-1}^{1} \sum_{l=-1}^{1} w_{kl}^{(z)} f_{i+k,j+l},$$

where

$$w^{(1)} = \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}, \quad w^{(2)} = \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}, \quad w^{(3)} = \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix},$$



(a) (b)

(c) (d)

Figure 3.16: Nonlinear edge filters applied to X-ray image: **(a)** square-root of variance filter, **(b)** range filter, **(c)** Roberts' filter, **(d)** Kirsch filter.

and each subsequent weights matrix, $w^{(4)}, \ldots, w^{(8)}$ has elements which are successively rotated through multiples of 45°. This is a much slower algorithm than the preceding three, as can be seen from the computer times in Table 3.1. However, the results are usually very similar, as in Fig 3.16(d).

All the above filters emphasise edges and noise simultaneously. To reduce the effect of noise, we need filters which also smooth. They could be obtained by applying the preceding filters to smoothed versions of images, such as those produced in §3.1.1 and §3.3, but a more elegant approach is motivated by estimation of gradients.

3.4.2 Gradient filters

Returning to the calculus notation of §3.1.2, the maximum gradient at point (i, j) is given by

$$\sqrt{\left(\frac{\partial f_{ij}}{\partial x}\right)^2 + \left(\frac{\partial f_{ij}}{\partial y}\right)^2}.$$

We can design a filter to estimate this gradient by replacing the partial derivatives above by estimates of them: the outputs from first-derivative row and column filters from §3.1.2

$$\widehat{\frac{\partial f_{ij}}{\partial x}} = \frac{1}{6}(f_{i-1,j+1} + f_{i,j+1} + f_{i+1,j+1} - f_{i-1,j-1} - f_{i,j-1} - f_{i+1,j-1})$$

and

$$\widehat{\frac{\partial f_{ij}}{\partial y}} = \frac{1}{6}(f_{i+1,j-1} + f_{i+1,j} + f_{i+1,j+1} - f_{i-1,j-1} - f_{i-1,j} - f_{i-1,j+1}).$$

This estimate of the maximum gradient is known as **Prewitt's filter**. Fig 3.17(a) shows the result of applying the filter to the X-ray image, which is equivalent to combining the images displayed in Figs 3.3(a) and (b), pixel-by-pixel, by taking the square-root of the sum of their squares. Larger pixel values of the filter output are shown darker in the display and zero values are shown as white. Notice that this filter, unlike the first-derivative ones considered in §3.1.2, responds to edges at all orientations.

Sobel's filter is very similar, except that in estimating the maximum gradient it gives more weight to the pixels nearest to (i, j), as follows:

$$\widehat{\frac{\partial f_{ij}}{\partial x}} = \frac{1}{8}(f_{i-1,j+1} + 2f_{i,j+1} + f_{i+1,j+1} - f_{i-1,j-1} - 2f_{i,j-1} - f_{i+1,j-1})$$

and

$$\widehat{\frac{\partial f_{ij}}{\partial y}} = \frac{1}{8}(f_{i+1,j-1} + 2f_{i+1,j} + f_{i+1,j+1} - f_{i-1,j-1} - 2f_{i-1,j} - f_{i-1,j+1}).$$

For both Prewitt's and Sobel's filter, the direction of maximum gradient can also be obtained as

$$\phi_{ij} = \tan^{-1}\left(\widehat{\frac{\partial f_{ij}}{\partial y}} \Big/ \widehat{\frac{\partial f_{ij}}{\partial x}}\right),$$

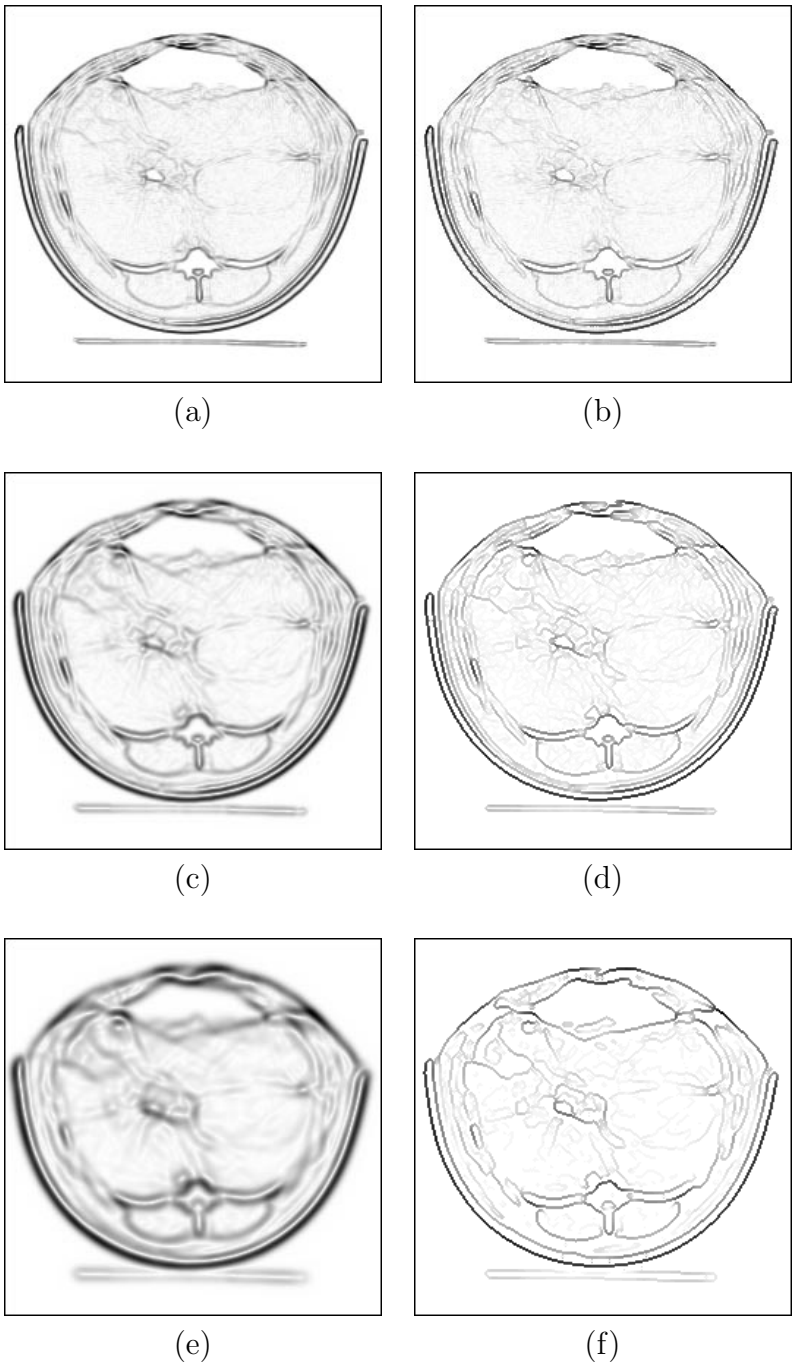


Figure 3.17: Gradient edge filters applied to X-ray image: **(a)** Prewitt's filter, **(b)** (a) restricted to zero-crossings of directional second-derivative, **(c)** maximum gradient after Gaussian, $\sigma^2 = 2$, **(d)** (c) restricted to zero-crossings, **(e)** maximum gradient after Gaussian, $\sigma^2 = 6\frac{2}{3}$, **(f)** (e) restricted to zero crossings.

measured clockwise from horizontal in the direction of increasing x. Fig 3.18(colour plate) shows a colour display of the result of applying Prewitt's filter to the DNA image, using a hue-intensity-saturation transformation (see §2.3.3): orientation (modulo π) is displayed as hue, the maximum gradient is assigned to image intensity and saturation is set to full. It can be seen that reddish colours predominate on the left side of the display, where edges are vertical, and green-blue for edges of bands which are oriented from top-left to bottom-right. Horizontal bands generate blue colours in the centre of the display, whereas on the right side magenta predominates, as a result of bands being oriented from bottom-left to top-right. Glasbey and Wright (1994) use the orientation information to 'unwarp' the image of the gel so that all bands are horizontal.

To increase the amount of smoothing in the filters, derivatives can be estimated after Gaussian filtering, as in §3.1.2. Fig 3.17(c) shows the result for $\sigma^2 = 2$, obtained by combining the estimated first-derivatives displayed in Figs 3.3(c) and (d). Similarly, Fig 3.17(e) shows the estimated gradient after Gaussian smoothing with $\sigma^2 = 6\frac{2}{3}$, derived from the data in Figs 3.3(e) and (f).

In an important paper, Canny (1986) shows that such filters are almost-optimal edge detectors. To overcome the problem of edge responses broadening in extent as smoothing increases, he suggested that positions of edges could be determined by finding zero-crossings of second-derivatives in the direction of steepest gradient, that is, of:

$$\cos^2\phi_{ij}\frac{\partial^2 f_{ij}}{\partial x^2} + \sin^2\phi_{ij}\frac{\partial^2 f_{ij}}{\partial y^2} + 2\sin\phi_{ij}\cos\phi_{ij}\frac{\partial^2 f_{ij}}{\partial x \partial y},$$

where ϕ_{ij} is defined above. The directional second-derivative can be estimated using an adaptive weights matrix:

$$w = \frac{1}{6}\begin{pmatrix} \frac{1}{2}\sin\phi_{ij}\cos\phi_{ij} & \sin^2\phi_{ij} & -\frac{1}{2}\sin\phi_{ij}\cos\phi_{ij} \\ \cos^2\phi_{ij} & -2 & \cos^2\phi_{ij} \\ -\frac{1}{2}\sin\phi_{ij}\cos\phi_{ij} & \sin^2\phi_{ij} & \frac{1}{2}\sin\phi_{ij}\cos\phi_{ij} \end{pmatrix}.$$

The effect is to thin the edges produced by the gradient filter. Figs 3.17(b), (d) and (f) show thinned versions of Figs 3.17(a), (c) and (e), in which all pixels have been set to zero except those in which the directional second-derivative changes sign between adjacent pixels.

The result is very similar to the Laplacian filter, shown in Fig 3.4. However, the Laplacian is the sum of the directional second-derivative and the second-derivative perpendicular to the direction of maximum slope. Because the perpendicular derivative is dominated by noise, Canny claims that the directional derivative is a better edge detector than the Laplacian. For further substantial papers on the theoretical underpinning of edge filters, see Haralick (1984) and Torre and Poggio (1986).

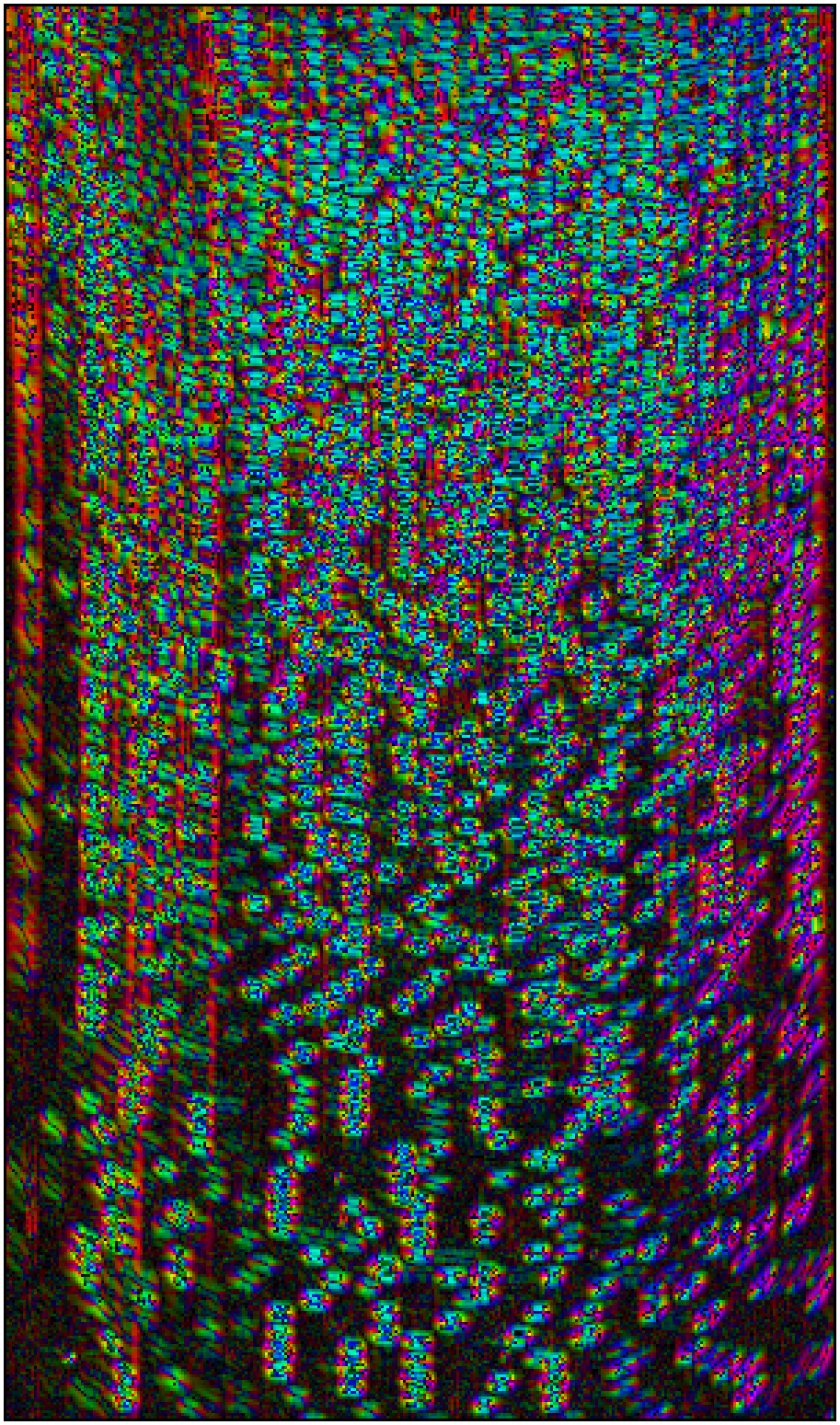


Figure 3.18: Prewitt's filter applied to DNA image, and displayed using intensity, hue and saturation: intensity is proportional to the square-root of the maximum gradient, hue represents the direction of maximum gradient (modulo 180°), and saturation is set to be full everywhere.

Summary

The key points about filters are:

- Filters re-evaluate the value of every pixel in an image. For a particular pixel, the new value is based of pixel values in a local neighbourhood, a window centred on that pixel, in order to:
  - reduce noise by smoothing, and/or
  - enhance edges.
- Filters provide an aid to visual interpretation of images, and can also be used as a precursor to further digital processing, such as segmentation (chapter 4).
- Filters may either be applied directly to recorded images, such as those in chapter 1, or after transformation of pixel values as discussed in chapter 2.
- Filters are linear if the output values are linear combinations of the pixels in the original image, otherwise they are nonlinear.
  - Linear filters are well understood and fast to compute, but are incapable of smoothing without simultaneously blurring edges.
  - Nonlinear filters can smooth without blurring edges and can detect edges at all orientations simultaneously, but have less secure theoretical foundations and can be slow to compute.

The main points about linear filters are:

- Two smoothing filters are:
  - moving average filter,
  - Gaussian filter.
- And four filters for edge detection are:
  - first-derivative row and column filters,
  - first-derivative row and column filters combined with the Gaussian filter,
  - Laplacian filter,
  - Laplacian-of-Gaussian filter.
- Linear filters can be studied in the frequency domain, as well as in the spatial domain. The benefits are:
  - efficient computation using the Fast Fourier Transform algorithm,

  - further insight into how filters work, such as categorizing them as low-pass, high-pass or band-pass,
  - opportunities to design new filters, such as:
    * 'ideal' low-pass and high-pass filters, and
    * the Wiener filter for image restoration.

The main points about nonlinear filters are:

- Five smoothing filters are:
  - moving median filter,
  - trimmed-mean filter, as an example of robust estimation,
  - k-neighbours filter,
  - Lee's filters,
  - minimum variance filter.
- Seven filters for edge detection are:
  - variance filter,
  - range filter,
  - Roberts' filter,
  - Kirsch's template filter,
  - Prewitt's gradient filter,
  - Sobel's gradient filter,
  - gradient filter combined with Gaussian filter, and thinned using Canny's method.

The output from filters in this chapter will be used in chapter 4 to segment images.

Comuter time (Seconds x 10)

n	128				256				512			
m	1	2	3	4	1	2	3	4	1	2	3	4
Linear filters												
moving average	1	1	1	1	4	4	4	4	16	16	16	16
Gaussian approximation	2	2	2	2	8	8	8	8	33	33	33	33
separable filter	2	2	2	2	6	7	8	9	25	31	36	40
general linear filter	2	3	5	7	8	13	21	31	31	53	85	126
Fourier implementation	6	6	6	6	24	24	24	24	112	112	112	112
Nonlinear smoothing filters												
median	1	1	1	2	4	5	5	6	13	18	22	26
robust - fastest	2	2	2	2	7	8	9	10	25	32	38	43
- slowest	4	4	4	5	12	15	17	19	48	61	71	79
minimum variance	4	4	4	4	17	17	17	17	74	74	74	74
Nonlinear edge filters												
Roberts	1	-	-	-	2	-	-	-	10	-	-	-
Kirsch	17	-	-	-	70	-	-	-	287	-	-	-

Table 3.1: Times for a SUN Sparc2 computer to process images of size n x n, by applying a range of filters using neighbourhoods of size (2m+1) x (2m+1)

100 x sample covariances

row	column								
	-4	-3	-2	-1	0	1	2	3	4
0					116	61	52	48	44
1	43	46	51	59	73	56	51	47	44
2	42	45	48	53	55	51	47	45	42
3	40	44	47	49	50	48	45	42	40
4	39	42	44	46	47	45	42	40	37

Table 3.2: Sample covariances of log-transformed pixel values in SAR image, for a range of row and/or column separations.