An Introduction to Digital Image Processing *

Jos B.T.M. Roerdink
Department of Mathematics and Computing Science, University of Groningen,
P.O. Box 800, 9700 AV Groningen, The Netherlands
Tel. +31-50-3633931; Fax +31-50-3633800;
Email roe@cs.rug.nl; URL: http://www.cs.rug.nl/~roe

Abstract

Digital image processing is the process of manipulating images by computer with the purpose of improving image quality or extracting useful information. This chapter collects basic concepts and terminology necessary to understand the fundamentals of the field. Topics discussed are: classification of operations in image processing, digitisation, linear filtering, enhancement, restoration, mathematical morphology, segmentation, representation, description and measurement.

1 Introduction

Digital image processing is the process of manipulating images by computer with the purpose of extracting useful information about the objects which appear in the image, and is a multidisciplinary field involving elements of optics, electronics, mathematics and computer engineering. As such, it suffers from an extensive but often rather imprecise jargon. It is the purpose of this chapter to collect a number of basic concepts in order to introduce the reader to the fundamentals of digital image processing, so that it may be useful for later reference. It should be realized that a certain amount of mathematical machinery is unavoidable in this field. The mathematical level of the exposition in this chapter confirms to that in the average textbook on digital image processing: elementary linear algebra (matrix calculus) and analysis, i.e., mainly convolution and Fourier transforms. An exception is Section 6, where a knowledge of basic set theory is assumed.

Because of its introductory nature, this chapter will treat many aspects only in a very global way. Also, recent developments such as processing of three-dimensional images, parallel image processing, special purpose hardware or high resolution display systems will not be discussed. The same holds for new methodologies such as wavelet transforms for image compression or colour image processing. The reader is referred to Castleman (1996), Gonzalez & Woods

*In: *Digital Image Analysis of Microbes: Imaging, Morphometry, Fluorometry and Motility Techniques and Applications*, M.H.F. Wilkinson and F. Schut (eds.), Modern Microbiological Methods (M. Goodfellow series editor), John Wiley and Sons Ltd, 1998. Postscript version obtainable at http://www.cs.rug.nl/~roe/

(1992), Rosenfeld & Kak (1982), Haralick & Shapiro (1992) or Jain (1989) for a more complete treatment of the fundamentals of digital image processing.

The organisation of this chapter is as follows. In Section 2 a global sketch of the basic concepts and terminology is given. Section 3 introduces the basics of linear filtering, with its associated concepts of convolution and point spread function. Section 4 gives examples of image enhancement by point operations (e.g., contrast stretching) or spatial filtering (smoothing, sharpening, etc.). Image restoration, that is, the process of undoing degradations of the image which occurred during image acquisition (blurring, noise) is presented in Section 5. Nonlinear image filtering is described in Section 6 where a concise introduction to mathematical morphology is presented. The following sections describe a number of specific image processing tasks, such as segmentation (Section 7) and description or feature extraction (Section 8).

2 Basic concepts

In its most simple form, a digital image processing system contains an input device, or image sensor, to acquire the image, a computer upon which to process the image, and an output device, cf. Fig. 1. For input device one may take a camera or an image digitiser. Output may go to a computer display or to hardcopy devices such as printers or plotters. Other components involve communication and image storage. Here we are mostly concerned with the computer processing aspects.

2.1 Digitisation

An image is a spatial representation of an object or a two- or three-dimensional scene. Usually the image is represented as a function $(x, y) \rightarrow f(x, y)$, where the domain of (x, y)-values is a discrete grid of small rectangular regions called picture elements or *pixels*. The values $f(x, y)$ are called grey levels: if n bits per pixel are available, the number of possible grey levels is 2^n. Very common are 8-bit images, for which grey levels range from 0 to 255.

The process of converting the values of the continuous luminance distribution reaching the image sensor to a numerical form readable by a computer is called *digitisation* or *analog-to-digital conversion*. This involves a sampling of the image brightness at each pixel location and a quantisation of the value within the allowed grey level range. A simple approach is to use uniform spatial sampling and equally spaced grey levels. More advanced techniques use adaptive or nonuniform sampling and quantisation.

2.2 Digital imaging

Many different areas involve the formation and manipulation of images by computer, such as image processing, image analysis and computer graphics, see Table 1. *Image processing* is concerned with *transforming* images: input of the processing is an image, output is again an image. In *image analysis* we want to obtain *quantitative* data from the images, such as the number of cells in a microscopic image, diameter of blood vessels in a cardiac scan, etc; in this

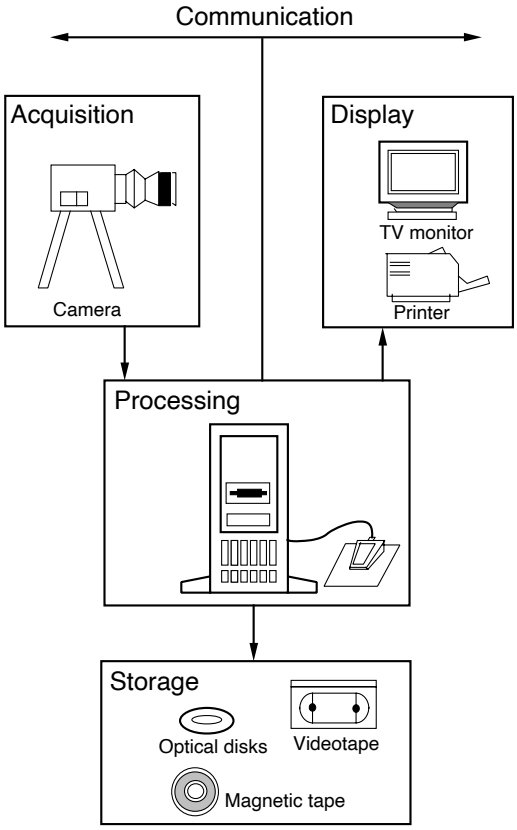


Figure 1: *A digital image processing system.*

case, input consists of one or more images, whereas the output contains numeric or symbolic data. Finally, in *computer graphics* the goal is image synthesis: starting from data derived from a mathematical description of objects, images are constructed on a computer display.

2.3 Classification of image operations

Assuming now that a digitised image is available in computer memory, various operations can be performed on the image, depending on the application. These operations can be classified with respect to their mathematical properties, or with respect to the goal of the operation (or set of operations).

A first classification considers the correspondence between the pixels of input and output images.

- *Point operation*: the grey level at each pixel of the output image f_{out} depends only on the grey level of the corresponding pixel in the input image f_{in}:

  $$f_{out}(x, y) = \mathcal{O}\left(f_{in}(x, y)\right),$$

  where $\mathcal{O}$ denotes the image operation. Examples are contrast stretching or histogram equalisation, see Section 4.1.

  Also in this category belong *algebraic* operations, where the output image is formed by pointwise addition, subtraction, multiplication or division of two input images. An example is subtraction of two images for shading correction.

- *Local operation*: the grey level at each pixel (x, y) of the output image depends on the grey levels of pixels which are contained in a small neighbourhood $\mathcal{B}(x, y)$ of the pixel (x, y) in the input image:

  $$f_{out}(x, y) = \mathcal{O}\left(\{f_{in}(x', y') : (x', y') \in \mathcal{B}(x, y)\}\right).$$

  See Section 4.2 for examples.

- *Global operation*: the grey level at each pixel of the output image depends on the grey levels of all pixels in the input image. An example of such an operation is a (discrete) Fourier transform of the input image (cf. Section 3).

- *Geometric operation*: this involves a spatial transformation of the image such as scaling, translation, rotation or perspective projection (e.g., used for distortion correction).

2.4 Connectivity

The above spatial relationships between pixels were expressed by using the concept of neighbourhood of a pixel. Although the notion of neighbourhood is intuitively obvious, a more precise formulation is needed for the case of digital images which are defined on discrete grids. In this

Table 1: *Subfields in digital imaging.*

	Image OUT	Data OUT
Image IN	image processing	image analysis
Data IN	computer graphics	data analysis

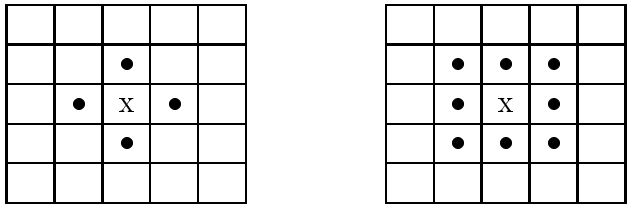


Figure 2: *Pixels* • *that are 4-connected (left) or 8-connected (right) to the centre pixel* x.

context the notion of *connectivity* is essential. When, on a rectangular pixel grid, only the pixels to the north, south, east and west of a pixel are considered as the neighbours of this pixel, we speak of *4-connectivity*. When in addition the diagonal pixels are considered as neighbours, one speaks of *8-connectivity*. We also say that the neighbours of a pixel are *adjacent* to that pixel. This is illustrated in Fig. 2.

A set M of pixels is called *connected* if and only if for every pair of pixels $p, q \in M$ there exists a path between p and q which only passes through pixels of M, where a path only moves between neighbouring pixels. A *connected component* is a nonempty connected set of pixels of maximal size. Note that the result of applying this definition depends on the type of neighbour relation used (4-connectivity or 8-connectivity). For binary images, where pixels have value 1 or 0, connected components can be defined both for foreground (1-pixels) and background (0-pixels). The *border* of a connected component C of 1-pixels is the set of pixels of C which are a neighbour of 0-pixels. We can also introduce (4- or 8) connectivity for the 0-pixels, and define the connected components of these pixels too. To distinguish both component types, let us speak about 0-components and 1-components. However, one has to be careful in the choice of connectivity for both types of pixels. It can be shown that if C is a 4-connected 1-component and D is an adjacent 8-connected 0-component, then either C surrounds D or D surrounds C (Rosenfeld 1970). This remains the case when we use 4-connectivity for the 0-pixels and 8-connectivity for the 1-pixels, but not when we use 4-connectivity (or 8-connectivity) for *both* 0- and 1-pixels.

Extraction of connected components is a step often used in segmentation or description, see Section 8. Efficient algorithms to perform this task based on some form of label propagation are known, for example the two-pass algorithm (Rosenfeld & Kak 1982).

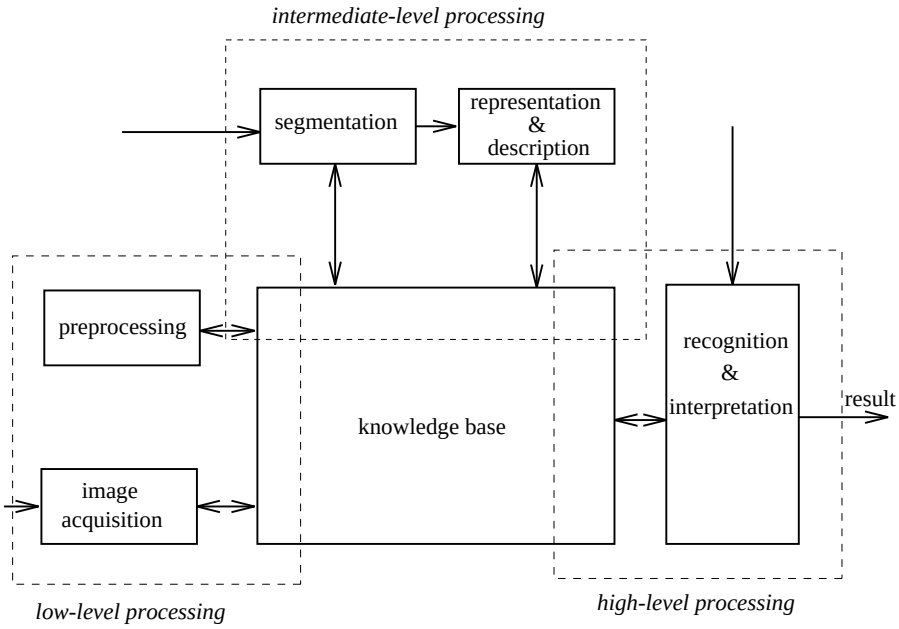


Figure 3: *Phases in digital image processing.*

2.5 Methodology

In digital image processing one may distinguish the following three levels:

- **low level**: the first step after image acquisition is preprocessing to obtain elementary or *atomic* features (representation in terms of *iconic* data)
- **intermediate level**: grouping of atomic features (segmentation, description, feature extraction)
- **high level**: recognition and interpretation of objects (representation in terms of *symbolic* data).

Let us expand a little bit upon the individual steps involved in a digital image processing system; cf. Fig. 3.

1. *image formation*: this is the first step, and a vital one. The quality of further processing depends in a crucial way upon the quality of the sensor, illumination conditions, digitisation, etc.
2. *preprocessing*: the assumption is that the image contains an informative part and variations which one wants to suppress. The preprocessing step, also called 'conditioning', encompasses enhancement, image restoration, noise suppression, background normalisation, etc.
3. *segmentation*: the assumption is that the image has a spatial structure. A first step is labelling where we try to organise the pixels into sets of correlated points. Examples

are thresholding, edge detection, corner detection. Next comes *grouping*: for example, connected component detection, segmentation on the basis of grey values, linking edges into borders. Note that there is a change of logical data structure here: from pixels into pixel *sets*.

4. *description*: in this step, also called *feature extraction*, pixel sets are grouped into lists of properties: area, centre of gravity, number of holes, curvature, relation to other groups.

5. *interpretation*: find the meaning of the extracted properties in terms of real world properties. Examples are: template matching; statistical pattern recognition (matching vectors of image properties to vectors of object properties); structural pattern recognition (matching vectors together with relations between them).

In the rest of this chapter we give a more precise definition and description of the different phases of image processing outlined above.

3 Linear filtering

Many of the operations commonly used for image enhancement involve the application of linear filters. There is a well-developed mathematical discipline underlying this, called linear system theory, which is an essential ingredient of linear signal and image processing. In this section we give the fundamentals of this theory, including a discussion of sampling, convolution, Fourier transforms, and filter design. For simplicity we first discuss one-dimensional signals, which are functions $f(t)$ of a single parameter, where t denotes the time. This is easily extended to images, which are — in the two-dimensional case — signals $f(x, y)$ of two spatial variables.

3.1 Linear shift-invariant systems

Two basic assumptions are made about the system which transforms signals to signals (or, for that matter, images to images). The first one is additivity: operating on the sum of two input signals gives the same result as operating on the two input signals separately and adding the results. A similar property holds with respect to scaling the intensity of the signal: first scaling and then operating on the signal gives the same result as first performing the signal operation and then scaling the result. A second basic assumption concerns time shifts: it makes no difference whether the system operates on a time-shifted signal, or on the original signal followed by a time shift. (In the case of 2D images there are two shift parameters, one for the horizontal and one for the vertical direction.) The usefulness of introducing systems with these two basic properties comes from the fact that such systems form a good approximation to many optical systems used in practice.

This leads to the following precise definition. A linear shift-invariant (LSI) system $\mathcal{O}$ maps an input signal $f(t)$ to an output signal $g(t)$, denoted as $f(t) \stackrel{\mathcal{O}}{\rightarrow} g(t)$, such that the following two conditions hold:

- *Linearity*: if $f_1(t) \xrightarrow{\mathcal{O}} g_1(t)$ and $f_2(t) \xrightarrow{\mathcal{O}} g_2(t)$, then, for arbitrary constants a, b,

$$a\, f_1(t) + b\, f_2(t) \xrightarrow{\mathcal{O}} a\, g_1(t) + b\, g_2(t)$$

- *Shift invariance*: if $f(t) \xrightarrow{\mathcal{O}} g(t)$, then for each time shift T,

$$f(t-T) \xrightarrow{\mathcal{O}} g(t-T)$$

3.1.1 Signals

The output of a LSI system is given by the *convolution* of the input with a kernel $k(t)$ which is characteristic for that system:

$$g(t) = (k * f)(t) = \int_{-\infty}^{\infty} k(t-s)\, f(s)\, \mathrm{d}s \tag{1}$$

If the input is an impulse, i.e., a delta function at time 0, then the output is precisely $k(t)$. Therefore the filter kernel $k(t)$ is usually called the *impulse response function*.

The 1D Fourier transform LSI systems can be conveniently described by using Fourier transforms. The Fourier transform of a function $f(t)$, denoted by $F(u)$, is defined by

$$F(u) = \int_{-\infty}^{\infty} e^{-i2\pi ut} f(t)\ \mathrm{d}t \tag{2}$$

For discrete and finite data the Fourier transform (2) is replaced by the forward discrete Fourier transform (DFT), which transforms a signal represented by the vector $(f(0), f(1), \ldots, f(N-1))$ into a transformed vector $(F(0), F(1), \ldots, F(N-1))$ by

$$F(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x)\, e^{-i2\pi ux/N}, \qquad u = 0, 1, \ldots, N-1$$

The original signal can be recovered from its discrete Fourier transform by the inverse discrete Fourier transform (IDFT):

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} F(u)\, e^{i2\pi ux/N}, \qquad x = 0, 1, \ldots, N-1$$

The DFT can be implemented efficiently by the Fast Fourier Transform (FFT) algorithm.

Introducing the Fourier transforms F, G and K of f, g and k, we can write the convolution formula (1) in the form:

$$G(u) = K(u)\, F(u).$$

So in the Fourier domain, convolution becomes multiplication, which is the basis of computer implementation using the FFT: first compute the DFT of k and f, then perform the multiplication of the Fourier transforms K and F, and finally compute the inverse DFT of G to obtain g. The function $K(u)$, which is the Fourier transform of the impulse response function, is called the *transfer function* of the linear system.

3.1.2 Images

In the 2D case, the convolution takes the form

$$g(x,y) = (k * f)(x,y) = \iint k(x-x', y-y')\, f(x',y')\, \mathrm{d}x'\, \mathrm{d}y' \tag{3}$$

Now $k(x,y)$ is often referred to as the *point spread function*. In the case of digital images, the convolution integral (3) becomes a summation:

$$g(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} k(x-m, y-n)\, f(m,n), \tag{4}$$

for $x = 0, 1, \ldots, M-1, y = 0, 1, \ldots, N-1$. Examples of discrete convolution for image filtering are given in Section 4.2.

The 2D Fourier transform For the case of images we use a two-dimensional DFT:

$$F(u,v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-i2\pi(ux/M + vy/N}, \quad u = 0, 1, \ldots, M-1; v = 0, 1, \ldots, N-1$$

with inverse

$$f(x,y) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(u,v)\, e^{i2\pi(ux/M + vy/N}, \quad x = 0, 1, \ldots, M-1; y = 0, 1, \ldots, N-1$$

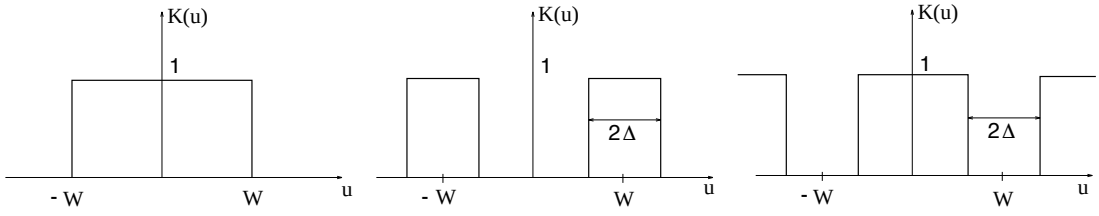
Introducing the 2D Fourier transforms F, G and K of f, g and k, we can again write the convolution formula (4) in the form of a product:

$$G(u,v) = K(u,v)\, F(u,v).$$

3.2 Filter design

Linear filters are commonly classified with respect to their frequency domain characteristics. *Lowpass* filters suppress high frequencies, and can therefore be used to suppress noise. The simplest case is the box filter k, for which $K(u) = 1$ for $-W \leq u \leq W$. Here W is the cutoff frequency. *Bandpass* or *bandstop* filters pass or suppress energy, respectively, if the frequency u is within a given window of size 2Δ. *Highpass* filters suppress low frequencies. This is useful for edge detection, for example (see Section 4.2.2). An example is the difference-of-Gaussians (DOG) filter, whose transfer function is given by

$$K(u) = A\, e^{-u^2/2\alpha_1^2} - B\, e^{-u^2/2\alpha_2^2}, \qquad A \geq B, \alpha_1 > \alpha_2.$$



(a) (b) (c)

Figure 4: *Ideal transfer functions. (a): lowpass. (b): bandpass. (c): bandstop.*

3.3 Sampling

Suppose the Fourier transform of a signal f is only nonzero in the interval $[-W, W]$. Such a signal is called *bandlimited*, and W is called the *bandwidth*. The famous sampling theorem of Shannon says that bandlimited signals can be represented by a sampling series:

$$f(x) = \sum_n f(n\Delta)\ \mathrm{sinc}_W(x - n\Delta), \tag{5}$$

where the *sinc-function* is defined by $\mathrm{sinc}_W(x) = \sin(2\pi W x)/(2\pi W x)$ for $x \neq 0$, and $\mathrm{sinc}_W(0) = 1$. Here $\Delta := 1/(2W)$ is the sampling interval and $f_s := 1/\Delta$ the corresponding sampling frequency. The Fourier transform of the sinc-function is the box function (see Fig. 5): $(\mathrm{sinc}_W)\widehat{\ }(u) = \Delta$ for $|u| < W$ and zero elsewhere. Formula (5) says that f is completely determined by its values at equidistant points $0, \pm\Delta, \cdots$. Note that a function with bandwidth W is certainly bandlimited on $[-W', W']$ with $W' > W$. So we can always use a smaller sampling step $\Delta' < \Delta$ in (5). The minimal sampling frequency required is f_s: this is called the *Nyquist frequency*. Note that the minimal frequency $f_s = 2W$ is twice the maximum frequency W contained in the signal f.

If one undersamples the signal by using a sampling frequency lower than the Nyquist frequency, distortions in the sampled signal appear; this is called *aliasing*. In the case of images visible artifacts appear which are called Moiré patterns. Unfortunately, real signals or images are defined on a bounded domain which implies that they cannot be strictly bandlimited. However, by a judicious choice of digitising parameters the so-called aliasing error can be reduced to an acceptable level.

4 Image enhancement

Image enhancement is the process of improving image quality so that the result is more suitable for a specific application. Examples are point operations such as contrast stretching, or spatial filtering for edge enhancement, noise suppression, smoothing and sharpening.

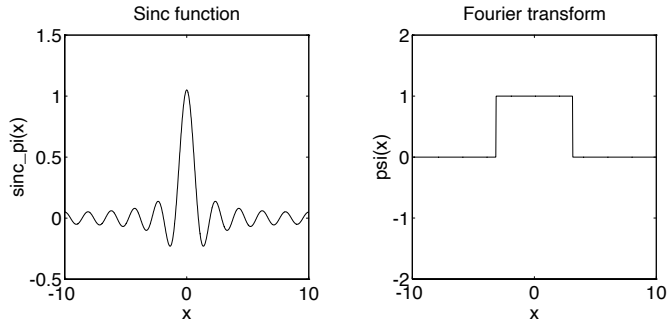


Figure 5: *Sinc function and its Fourier transform for* $W = 1/2$.

4.1 Point operations

Point operations are often used to improve the visual appearance of an image. As defined in Section 2, a point operation is distinguished by the fact that the value at each pixel of the output image depends only on the value of the corresponding pixel in the input image:

$$f_{out}(x, y) = \mathcal{O}\left(f_{in}(x, y)\right),$$

where $\mathcal{O}$ denotes the *grey scale transformation* (GST) function. This can be a linear or nonlinear function. We give two examples.

4.1.1 Contrast stretching

If the features of interest occupy only a small range of the available grey levels, one may use a point operation so that the output image spans a larger range. This is called contrast stretching. For example, let the input image f_{in} have minimum and maximum grey level m and M ($M > m \geq 0$), respectively. Then the following operation stretches the image to full grey level range $[0..255]$:

$$f_{out}(x, y) = \frac{255}{M - m}\left(f_{in}(x, y) - m\right).$$

See Fig. 6(b) for an example.

4.1.2 Histogram equalisation

The *histogram* $h(m)$ of a digital image f is the number of times the grey value m occurs in the image. Now suppose we want to apply a point operation to the image f which produces a flat histogram in the output, i.e., on the average an equal number of pixels at each grey level. So if the image has N pixels and grey level range $[0..M]$, the output image will have M/N pixels at

(a) (b) (c)

Figure 6: *(a): input image. (b): contrast stretch of (a). (c): histogram equalisation of (a).*

each grey level, cf. Fig. 6(c). It can be shown that the grey scale transformation which achieves this histogram equalisation is given by $\mathcal{O}(f(x,y)) = M\,\mathsf{P}(f(x,y))$, where

$$\mathsf{P}(\ell) = \frac{1}{N}\sum_{m=0}^{\ell} h(m)$$

is the normalised cumulative histogram function.

4.2 Spatial filtering

4.2.1 Smoothing filters

To suppress noise one may use lowpass filtering by local averaging within a small neighbourhood or *mask* surrounding each pixel. This can be implemented by convolution filtering, cf. Section 3. For example, take as the mask a 3×3 neighbourhood, with $k(m,n) = 1/9$ for $|m| \leq 1, |n| \leq 1$ in (4); this is called *uniform filtering*, see Fig. 7(b). This may be extended to more general filter kernels, such as a Gaussian function.

To obtain noise reduction without blurring of the edges, one may use *rank-order* (or *percentile*) filters based upon ranking the pixel values within the mask surrounding the centre pixel, cf. Fig. 7(c). An example is *median filtering*: sort the set of pixel values and assign the median (the value m such that half the values in the set are less than m and half are greater than m) to the centre pixel. This class of filters is nonlinear and therefore more difficult to analyse than linear filters based on convolution. More examples of such nonlinear filters are provided by the morphological filters, see Section 6.

4.2.2 Sharpening filters

The goal of sharpening is to enhance fine details such as edges in an image that has been blurred during image acquisition. This can be performed by using highpass filters based upon spatial

(a) (b) (c)

Figure 7: *Smoothing filters. (a): Original. (b): uniform filter. (c): percentile filter.*

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 8: *Laplacian convolution kernels.*

differentiation. Some of the most well-known examples are the following, cf. Fig. 11.

- **Laplace operator** A digital implementation of Laplacian filtering uses the discrete convolution kernels shown in Fig. 8.

- **Roberts operator** When $f(x, y)$ is the input image, this operator produces an output $g(x, y)$ given by

$$g(x, y) = \left\{ (f(x, y) - f(x + 1, y + 1))^2 + (f(x + 1, y) - f(x, y + 1))^2 \right\}^{\frac{1}{2}}$$

- **Sobel operator** Each point of the input image is convolved with two kernels, shown in Fig. 9, one for detecting horizontal edges and the other for vertical edges. The output of the operator is the maximum of the two convolutions; one may also take the square root of the sum of the squares.

- **Prewitt operator** This works in exactly the same way as the Sobel filter, except for the convolution kernels, which are now as shown in Fig. 10.

-1	-2	-1
0	0	0
+1	+2	+1

-1	0	+1
-2	0	+2
-1	0	+1

Figure 9: *Kernels for the Sobel operator.*

-1	-1	-1
0	0	0
+1	+1	+1

+1	0	-1
+1	0	-1
+1	0	-1

Figure 10: *Kernels for the Prewitt operator.*

4.3 Frequency domain filtering

Enhancement can also be performed in the frequency domain: compute the Fourier transform of the input image, multiply the result by a filter transfer function, and take the inverse Fourier transform. For example, to suppress noise one may use lowpass filtering (cf. Section 3.2) which attenuates the high-frequency components. On the other hand, edge sharpening is obtained by highpass filtering.

5 Image restoration

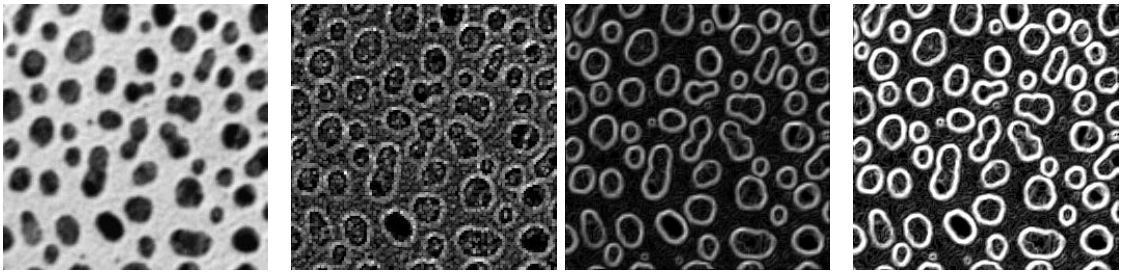
By image restoration we denote the process of removing or reducing image degradations which occur during image formation. Important degrading factors are: (i) *blurring* produced by the optical system or object motion during acquisition, and (ii) *noise* from electronic and optical devices.

5.1 Linear degradation model

A simple model assumes linear degradation of the ideal image $f(x, y)$ described by a convolution kernel $k(x, y)$ and additive noise $n(x, y)$, resulting in the degraded image $w(x, y)$:

$$w(x, y) = (k * f)(x, y) + n(x, y). \tag{6}$$

Restoration now has to undo the degradation due to convolution and noise.



(a) (b) (c) (d)

Figure 11: *Effect of sharpening filters. (a): Original. (b): Laplace filter. (c): Roberts filter. (d): Sobel filter.*

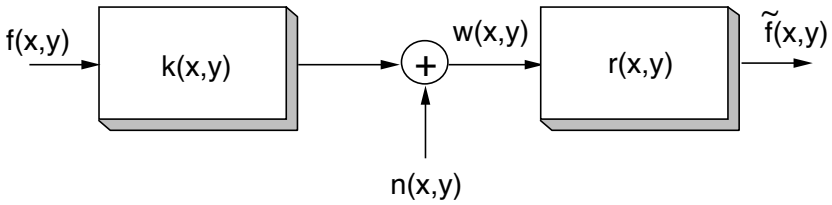


Figure 12: *Linear image restoration model.*

5.2 Linear restoration model

Using a linear model for restoration as well (for this reason, one often speaks of *deconvolution*), we compute an estimate $\tilde{f}(x, y)$ of the ideal image by using a convolution kernel $r(x, y)$:

$$\tilde{f}(x, y) = (r * w)(x, y)$$

The complete process of degradation and restoration is graphically represented in Fig. 12. A simplified restoration problem results if we consider images of finite size, say $N \times N$. In that case we can work with matrix-vector calculus (Andrews & Hunt 1977). The restoration problem (6) then has the form

$$\mathbf{w} = \mathbf{K}\,\mathbf{f} + \mathbf{n} \tag{7}$$

where $\mathbf{w}, \mathbf{f}, \mathbf{n}$ are column vectors of length N^2, and $\mathbf{K}$ is an $N^2 \times N^2$ matrix representing the convolution. To give an example, for an image of size 512×512 the matrix $\mathbf{K}$ contains over $60,000$ million entries, but in practice it is sparse, i.e., many entries are zero. For shift-invariant systems, the matrix $\mathbf{K}$ has a periodic structure which can be exploited to achieve efficient computation using the Fast Fourier Transform, cf. Section 3.

5.3 Constrained least squares restoration

A number of well-known restoration filters is captured by considering the following *constrained least squares restoration* model. First, the estimated solution vector $\tilde{\mathbf{f}}$ should satisfy (7); this gives the constraint that the norms of $\mathbf{w} - \mathbf{K}\tilde{\mathbf{f}}$ and $\mathbf{n}$ are equal:

$$\left\|\mathbf{w} - \mathbf{K}\tilde{\mathbf{f}}\right\|^2 = \|\mathbf{n}\|^2 \tag{8}$$

Second, one may subject the solution $\tilde{\mathbf{f}}$ to additional constraints; this can be realized by introducing a matrix $\mathbf{Q}$ such that the term

$$\left\|\mathbf{Q}\,\tilde{\mathbf{f}}\right\|^2 \tag{9}$$

is minimised. To take both contributions into account, we want to minimise

$$\mathcal{E}(\tilde{\mathbf{f}}) = \left\|\mathbf{Q}\,\tilde{\mathbf{f}}\right\|^2 + \gamma^{-1}\left(\left\|\mathbf{w} - \mathbf{K}\tilde{\mathbf{f}}\right\|^2 - \|\mathbf{n}\|^2\right) \tag{10}$$

where γ is a constant which determines the relative weight of the terms (8) and (9). The solution of (10) is:

$$\tilde{\mathbf{f}} = \left(\mathbf{K}^t\mathbf{K} + \gamma\,\mathbf{Q}^t\mathbf{Q}\right)^{-1}\mathbf{K}^t\mathbf{w} \tag{11}$$

where the superscript 't' denotes transposition of vectors or matrices. Special cases are the following:

Pseudoinverse filter

Take $\mathbf{Q} = \mathbf{I}$, the identity matrix. Thus we minimise the norm of $\tilde{\mathbf{f}}$ subject to the constraint (8). The solution (11) then reduces to:

$$\tilde{\mathbf{f}} = \left(\mathbf{K}^t\mathbf{K} + \gamma\,\mathbf{I}\right)^{-1}\mathbf{K}^t\mathbf{w}$$

An alternative formulation of this result makes use of the discrete Fourier transforms $K(u,v)$, $W(u,v)$, $F(u,v)$ of $k(x,y)$, $w(x,y)$, $f(x,y)$ in (6). The estimate $\tilde{F}(u,v)$ of $F(u,v)$ according to the pseudoinverse filter has the form:

$$\tilde{F}(u,v) = \left(|K(u,v)|^2 + \gamma\right)^{-1} K^*(u,v)\,W(u,v), \tag{12}$$

with $K^*(u,v)$ the complex conjugate of $K(u,v)$.

Inverse filter

For $\gamma = 0$ the formula for the pseudoinverse filter reduces to the 'inverse' filter:

$$\tilde{\mathbf{f}} = (\mathbf{K}^t\mathbf{K})^{-1}\mathbf{K}^t\mathbf{w} = \mathbf{K}^{-1}\mathbf{w}$$

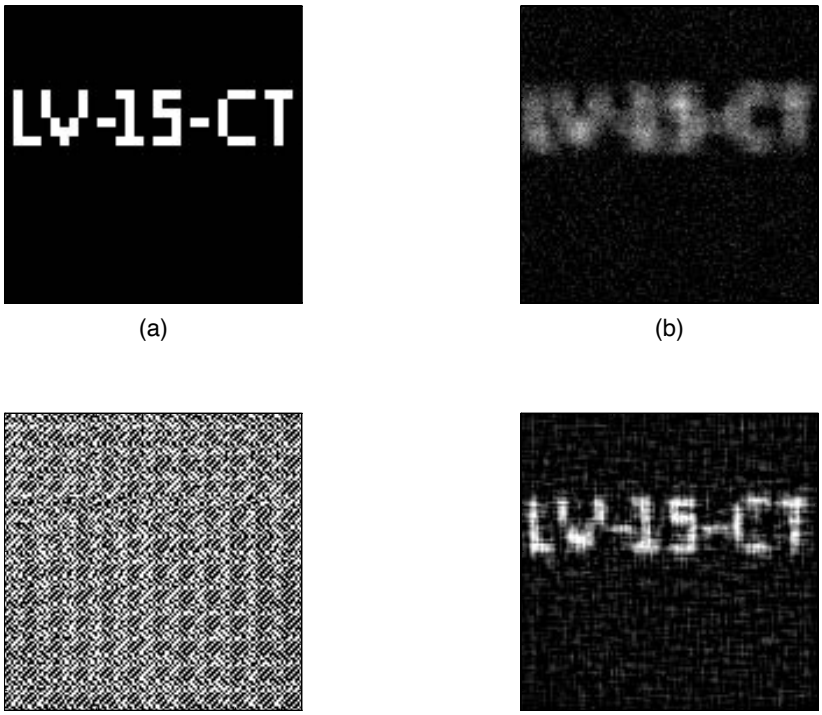


Figure 13: *Effect of restoration filters. (a): Original. (b): Degraded image. (c): Inverse filtering. (d): Pseudoinverse filtering ($\gamma = 0.01$).*

or, in terms of Fourier transforms,

$$\tilde{F}(u,v) = \frac{W(u,v)}{K(u,v)} = F(u,v) + \frac{N(u,v)}{K(u,v)}, \tag{13}$$

where $N(u,v)$ is the Fourier transform of the noise term $n(x,y)$ in (6). In practice, inverse filtering is often useless, the reason being that the matrix $\mathbf{K}$ may be (almost) singular. Equivalently, this means that the Fourier transform $K(u,v)$ has values which are zero or very small at certain points (u,v) in the frequency plane, leading to amplification of the noise term in (13). An example is given below.

Parametric Wiener filter

Assume $\mathbf{f}$ and $\mathbf{n}$ to be random vectors with zero mean and covariance matrices $\mathbf{R}_f = \mathbb{E}(\mathbf{f}\,\mathbf{f}^t)$ for the signal and $\mathbf{R}_n = \mathbb{E}(\mathbf{n}\,\mathbf{n}^t)$ for the noise. Let $\mathbf{Q}$ be the noise-to-signal ratio:

$$\mathbf{Q} = \left(\frac{\mathbf{R}_n}{\mathbf{R}_f}\right)^{\frac{1}{2}}$$

Then the solution (11) reads

$$\tilde{\mathbf{f}} = \left(\mathbf{K}^t\mathbf{K} + \gamma\,\mathbf{R}_f^{-1}\mathbf{R}_n\right)^{-1}\mathbf{K}^t\mathbf{w}$$

For shift-invariant blur and stationary signal and noise, this filter is the parametric Wiener filter, which for $\gamma = 1$ reduces to the ordinary Wiener filter.

As an example, we present in Fig. 13 the results of applying inverse and pseudoinverse filters to an image degraded by uniform blurring and Gaussian noise. As can be seen, the inverse filter is useless, because of noise amplification. The pseudoinverse filter, however, leads to satisfactory results.

6 Mathematical morphology

Mathematical morphology was developed at the Paris School of Mines as a set-theoretical approach to image analysis (Matheron 1967, Serra 1982). For a first introduction, see e.g. Giardina & Dougherty (1988), Haralick, Sternberg & Zhuang (1987) or Serra & Vincent (1992). The method works by translating small subsets B, called *structuring elements*, of various forms and sizes over the image plane and recording the locations where certain relations between the image and the translated structuring element are satisfied. This can be compared to the neighbourhood operations of linear filtering (cf. Section 4.2), the main difference being that the operations are now *nonlinear*. Morphological operations are usually performed in a sequence, which is chosen in such a way that the end result is in a form suitable for quantitative measurements. First we will look at some of the basic transforms for the case of binary images. Then the case of grey value images will be considered.

6.1 Morphology for binary images

Binary images have only two values, say, zero or one (black and white). This means that we can describe such images in terms of *sets* in the plane: all white pixels form a set X, and all black pixels the complement of X in the plane. The theory of mathematical morphology not only works for discrete images, but for continuous ones as well, with a unified formulation. In the continuous case one considers images as subsets of the Euclidean space $E = \mathbb{R}^2$, in the discrete case as subsets of the discrete grid $E = \mathbb{Z}^2$.

We add a brief reminder about sets and the basic set operations. A set X with elements $x_1, x_2, x_3, \ldots$ is denoted by $X = \{x_1, x_2, x_3, \ldots\}$. The expression $x \in X$ means that x is an element of X. The set Y consisting of those elements of X satisfying some condition C, is written $Y = \{x \in X : C\}$. For example, if $X = \{-2, -1, 0, 1, 2\}$, the elements of X which are positive can be selected as follows: $Y = \{x \in X : x > 0\} = \{1, 2\}$.

If X and Y are subsets of a certain universal set E (say, the plane), the union $X \cup Y$ contains all elements which occur in X or in Y (possibly in both), the intersection $X \cap Y$ contains all elements which occur both in X and Y, the set difference $X \backslash Y$ contains those elements of X which are not an element of Y, and the complement X^c contains all elements of E which are not in X, i.e., $X^c = E \backslash X$.

6.1.1 Dilation and erosion

Let A be a set in the plane (the structuring element). We write $x + y$ for the vector sum of x and y, and $-x$ for the reflected vector of x w.r.t. the origin. If we translate (i.e. shift) A along the vector h, we write the result as A_h, so

$$A_h = \{a + h : a \in A\}.$$

Also, if we reflect all points of a set A, we get the *reflected set* $\check{A}$, i.e.

$$\check{A} = \{-a : a \in A\}.$$

Now we can define the following elementary algebraic operations between two sets X and A:

$$\begin{aligned} Minkowski\ addition: &\quad X \oplus A = \{x + a : x \in X, a \in A\} = \bigcup_{a \in A} X_a = \bigcup_{x \in X} A_x \\ Minkowski\ subtraction: &\quad X \ominus A = \bigcap_{a \in A} X_{-a} \end{aligned}$$

When the set A is fixed, the mapping $X \mapsto \delta_A(X) = X \oplus A$ is called the *dilation* with structuring element A, and the mapping $X \mapsto \varepsilon_A(X) = X \ominus A$ is called the *erosion* with structuring element A. There is a simple geometrical interpretation of these operations, expressed by the following formulas:

$$\begin{aligned} Dilation: &\quad X \oplus A = \{x \in E : (\check{A})_x \cap X \neq \emptyset\} \\ Erosion: &\quad X \ominus A = \{x \in E : A_x \subseteq X\}. \end{aligned}$$

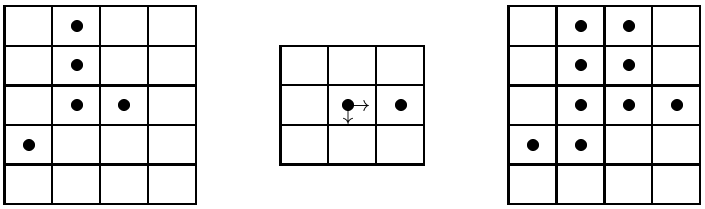


Figure 14: *Left: binary image X. Middle: structuring element A. Right: dilation of X by A.*

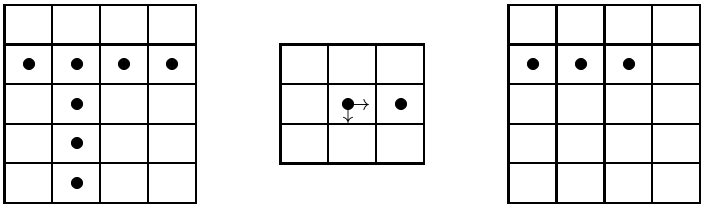


Figure 15: *Left: binary image X. Middle: structuring element A. Right: erosion of X by A.*

In words, the dilation of X by A is the collection of points h such that $\check{A}$ after translation over the vector h 'hits' the set X. Similarly, the erosion of X by A is the collection of points h such that after shifting A over the vector h it 'fits' into X. For the discrete case, an example for the dilation is given in Fig. 14, and for the erosion in Fig. 15. Pixels with value 1 (1-pixels) are represented by dots, pixels with value 0 by blanks, and the position of the origin in the structuring element is indicated by the symbol $\overset{\rightarrow}{\downarrow}$ (reflecting the fact that we use a row-column oriented coordinate system). Dilation and erosion are 'translation-invariant': if the origin of X is shifted over a vector h, followed by a dilation or erosion, the result is the same as first dilating or eroding the set, followed by translation. For example:

$$X_h \oplus A = (X \oplus A)_h$$

There exists a *duality relation* with respect to set-complementation (X^c denotes the complement of the set X):

$$X \oplus A = (X^c \ominus \check{A})^c,$$

i.e. dilating an image by A gives the same result as eroding the background by $\check{A}$, followed by taking the complement.

6.1.2 The hit-or-miss transform

The hit-or-miss transform of an image X by a composite structuring element (A^1, A^2) is the collection of points h such that the shifted set A^1_h fits into X and the shifted set A^2_h has no overlap with X. Expressed in a formula:

$$\begin{aligned} X \circledast (A^1, A^2) &= \{h \in E : A^1_h \subseteq X, A^2_h \subseteq X^c\} \\ &= (X \ominus A^1) \cap (X^c \ominus A^2) \end{aligned}$$

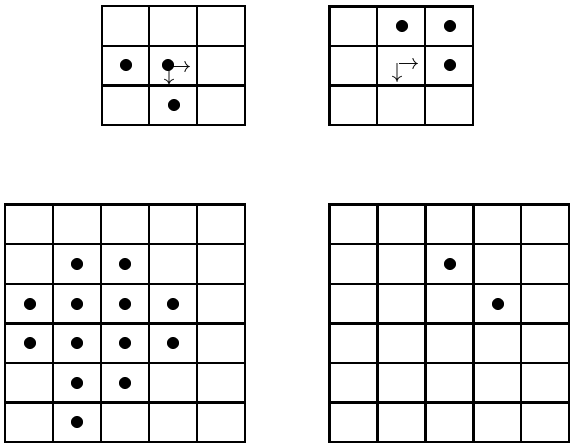


Figure 16: *Upper left: structuring element A^1. Upper right: structuring element A^2. Lower left: binary set X. Lower right: hit-or-miss transform of X.*

This is a form of 'template matching'. This transformation can be used to detect special points in an image. An example is given in Fig. 16, which shows an application where north-east pixels are detected.

The erosion, or the hit-or-miss transform, can also be used to compute the genus or the number of regions in a binary image (Haralick & Shapiro 1992), see also Section 8.2.2.

6.1.3 Openings and closings

The opening and closing by a structuring element A are obtained by an erosion followed by a dilation, and conversely.

$$Opening: \qquad X \circ A := (X \ominus A) \oplus A \tag{14}$$

$$Closing: \qquad X \bullet A := (X \oplus A) \ominus A \tag{15}$$

It can be shown that the opening is the union of all shifted copies of the structuring element A which are included in the set X. An opening smooths contours, cuts narrow bridges, removes small islands and sharp corners; a closing fills narrow channels and small holes. For the discrete case, an example is given in Fig. 17. It is easy to see that opening and closing are translation-invariant, just as dilation and erosion. A translation of the structuring element A, however, does not affect the outcome.

6.1.4 Thinnings and thickenings

Also much used are the *thinning* and *thickening* of a set X by $B = (B^1, B^2)$, where B^1 and B^2 should be disjoint:

$$Thinning: \qquad X \bigcirc B = X \backslash (X \circledast B) = X \cap (X \circledast B)^c$$

$$Thickening: \qquad X \odot B = X \cup (X \circledast B)$$

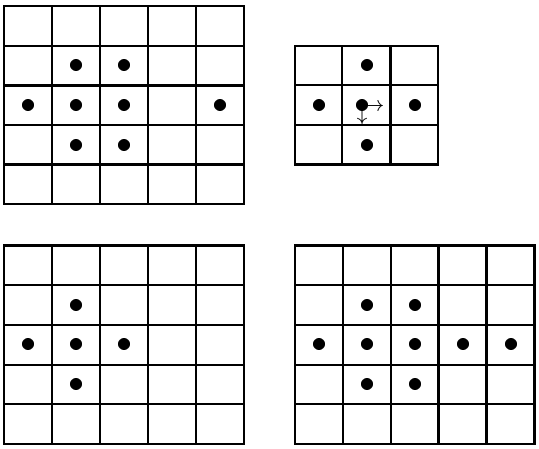


Figure 17: *Upper left: binary image X. Upper right: structuring element A. Lower left: opening of X by A. Lower right: closing of X by A.*

6.1.5 Distance transforms

On the discrete grid we use a *digital* distance function, which depends on the type of connectivity: if 4-connectivity is used, each of the 4 neighbours is by definition at distance 1 from the centre pixel. This is called the *city-block distance*. We note that other digital metrics have been considered involving so-called *chamfer distances*, which very well approximate the Euclidean distance (Borgefors 1984).

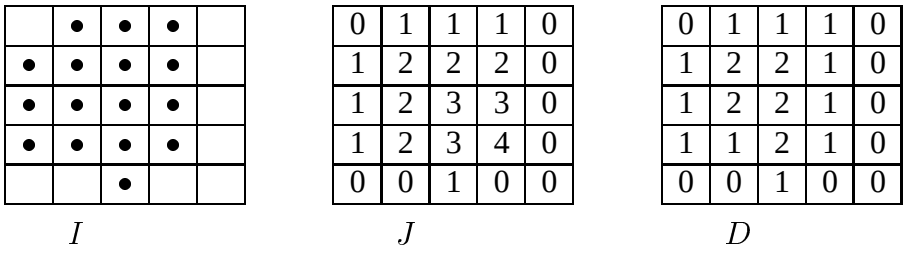
Paths on the grid are formed by steps in the horizontal and vertical directions (each of length 1). The length of a path is the sum of the lengths of the individual steps. The distance $d(x, y)$ between two points x and y on the grid is then the minimum of the lengths of all allowed paths from x to y on the grid. We also define the distance $d(x, A)$ between a point x and a set A as the minimum of the distances between x and points of A.

Let A be a discrete binary image. The *distance transform* of A is the grey value function $D : E \to \mathbb{N}$ which associates to point x the distance of x to the complement of A: $D(x) = d(x, A^c)$. Notice that $D(x) = 0$ for points $x \in A^c$. The distance transform can be computed by a 2-pass procedure (Rosenfeld & Pfaltz 1968), consisting of a forward scan (top-down, left-to-right) which takes only steps to the left or upwards into account, followed by a reverse scan (bottom-up, right-to-left) which considers steps to the right and downwards. An example is shown in Fig. 18.

6.1.6 Skeletonisation

For purposes of shape description and efficient coding it is desirable to extract features from binary images which represent only the 'backbone' or skeleton of the set. Roughly spoken these are connected lines within the binary image which are in the middle of the set and 'as thin as possible'.

For a binary image A one can define the so-called SKIZ ('skeleton by influence zones'),



	•	•	•	
•	•	•	•	
•	•	•	•	
•	•	•	•	
		•		

I

0	1	1	1	0
1	2	2	2	0
1	2	3	3	0
1	2	3	4	0
0	0	1	0	0

J

0	1	1	1	0
1	2	2	1	0
1	2	2	1	0
1	1	2	1	0
0	0	1	0	0

D

Figure 18: *Left: digital image I. Middle: forward scan output image J. Right: city-block distance transform of I produced by reverse scan of J.*

which is a skeleton of the background. Let

$$A = \bigcup_{n=1}^{N} C_n$$

with $C_1, \ldots, C_N$ the connected components of A (cf. Section 2.4). The zone of influence Z_n of C_n is the set of points closer to C_n than to any other component:

$$Z_n = \{x : d(x, C_n) < d(x, C_m), m \neq n\}.$$

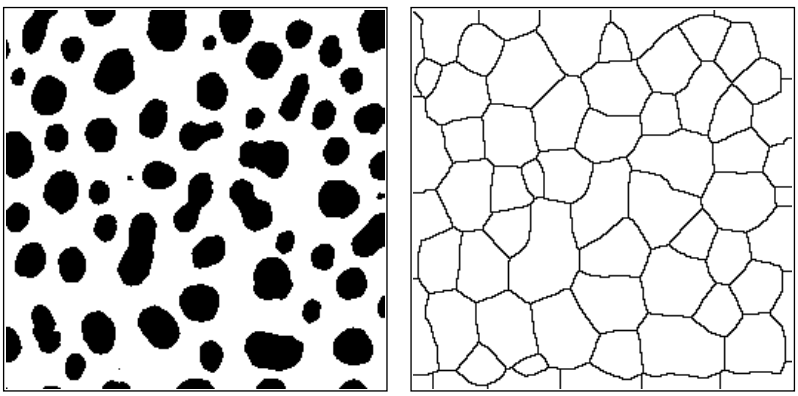
The points that do not belong to any Z_n (points that are equidistant to at least two components) form the SKIZ S_z of A:

$$S_z(A) = \left(\bigcup_{n=1}^{N} Z_n \right)^c.$$

See Fig. 19 for an example.

6.2 Morphology for grey value images

Morphological operations for grey value images can be defined in analogy with the binary case. The basic idea is to consider a grey value image as a landscape in three-dimensional space, where the vertical dimension denotes grey value. We can define morphological transformations acting on the image f by introducing the concept of *umbrae* ('shadows') (Sternberg 1986). For a given grey value image f, the umbra of f is the set $U(f)$ consisting of all points below the graph of f (when one imagines a light source above the landscape, the umbra is made up of all points 'in the shadow'). Now one may apply a binary morphological operation to the set $U(f)$, which yields a transformed set. To this set one can again associate a grey value image by looking at the *top surface* of the transformed set. More details can be found in Serra (1988) or Haralick & Shapiro (1992). For a mathematical treatment, see e.g. Heijmans (1994). Here we confine ourselves to a presentation of some of the most important morphological operations for grey value images.



(a) (b)

Figure 19: *Skeletonisation. (a): Binary image. (b): SKIZ of (a).*

6.2.1 Grey value dilation and erosion

Let f and k be two grey value images with domain F and K, respectively. The *grey value dilation* $f \oplus k$ and the *grey value erosion* $f \ominus k$ of f by k are the functions defined by

$$(f \oplus k)(x) = \max_{y \in K, x-y \in F} [f(x-y) + k(y)],$$
$$(f \ominus k)(x) = \min_{y \in K, x+y \in F} [f(x+y) - k(y)].$$

That is, $f \oplus k$ equals the maximum of $f(x-y) + k(y)$ when y runs over the set K, and similarly for $f \ominus k$.

A special case occurs for so-called *flat* structuring functions, which have a *constant* value of zero on a domain K. Then we get the flat grey value dilation and erosion, which simply replace each pixel by the maximum or minimum in a neighbourhood defined by a structuring element K:

$$(f \oplus K)(x) = \max_{y \in K, x-y \in F} [f(x-y)],$$
$$(f \ominus K)(x) = \min_{y \in K, x+y \in F} [f(x+y)].$$

For this reason these operations are called *minimum* and *maximum* filters.

6.2.2 Grey value opening and closing

By taking products of the grey value dilation and erosion we can construct grey value openings and closings just as in the binary case. For *flat* structuring functions with domain K, the *grey*

Figure 20: *Left: a grey value image. Right (reduced by factor of 2): upper left: grey value dilation; upper right: grey value erosion; lower left: grey value opening; lower right: grey value closing.*

value opening $f \circ K$ and the *grey value closing* $f \bullet K$ are defined by

$$(f \circ k)(x) = \max_{z \in K} \min_{y \in K} f(x - z + y)$$
$$(f \bullet k)(x) = \min_{z \in K} \max_{y \in K} f(x + z - y)$$

The grey value opening and closing filters are also called *min-max* and *max-min* filters, respectively. The opening eliminates peaks, the closing valleys. Figure 20 shows the result of the grey value dilation, erosion, opening and closing for a grey value image. If we apply the grey value opening with a *flat* structuring element to an image f and subtract the result from f, we get Meyer's *top-hat transform* (Serra 1982). This transform extracts the peaks from a grey value image.

7 Image segmentation

The process of isolating the important objects in the image from the background is called image segmentation (Haralick & Shapiro 1985). More precisely, image segmentation can be defined as the process of partitioning an image into disjoint regions. Each region should be uniform and homogeneous with respect to some property, such as grey value or texture, and differ significantly from neighbouring regions.

The following subdivision of image segmentation methods can be made:

- *Region based approach*: assign pixels to regions or objects
- *Boundary based approach*: locate boundaries between regions
- *Edge based approach*: identify edge pixels and link them to form boundaries

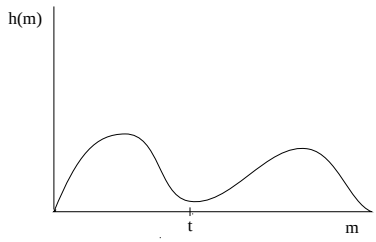


Figure 21: *Choice of the threshold t for a bimodal histogram.*

7.1 Region based approaches

7.1.1 Thresholding

Let $f(x, y)$ be a grey value image. Let t be a fixed grey value called the 'threshold'. Then we define a binary image f_t by thresholding as follows:

$$f_t(x, y) := 1 \text{ if } f(x, y) \geq t$$
$$f_t(x, y) := 0 \text{ if } f(x, y) < t$$

In this method, pixels above a certain value t — the 'threshold' — are declared to be 'object', those below the threshold 'background'. This will work fine when grey levels of both objects and background are relatively uniform, but distinct. When the background grey level is not constant one may use a threshold which varies over the image: this is called *adaptive thresholding*.

An important problem is to determine the threshold automatically. This can be done by considering the histogram of the grey values of the image (cf. Section 4.1.2). If the histogram is bimodal, i.e has two peaks, an obvious choice for the threshold is a value at the local minimum in the valley between the two maxima, see Fig. 21. When the histogram does not have a nice bimodal shape, one may determine a global threshold statistically, by minimising *within group variance* (Haralick & Shapiro 1992). This means splitting the set of grey values into two disjoint subsets V_1 and V_2 in such a way that the weighted sum of the variances of the groups V_1 and V_2 (which are measures of homogeneity) is minimal.

7.1.2 Watershed segmentation

The standard approach to image segmentation in morphological image processing (cf. Section 6) is by the watershed transform originally proposed by Lantuéjoul (Serra 1982, Vincent & Soille 1990, Beucher & Meyer 1993). The intuitive idea underlying this method is that of flooding a landscape or topographic relief with water, by piercing holes at the positions of local minima and immersing the landscape into a lake. Basins will fill up with water starting at local minima, and at points where water coming from different basins would meet, dams are built. When the water level has reached the highest peak in the landscape, the process is stopped. The set of dams thus obtained partitions the region into 'catchment basins' separated by dams. These dams

(more precisely, their projections on the horizontal plane) are called watershed lines or simply *watersheds*.

A recursive algorithm for computing the watershed transform was given by Vincent & Soille (1990). The basic structure of the algorithm is a loop in which the image is thresholded at successive grey levels. In every iteration the current basins belonging to the minima are extended by their influence zones within the binary image obtained by thresholding at the current grey level (cf. the computation of the SKIZ).

Other approaches exist based on a definition of watersheds in terms of distance functions (Meyer 1994, Najman & Schmitt 1994, Meijster & Roerdink 1996).

To separate the regions in a grey value image one usually first computes the derivative of the image using some edge detector, before applying the watershed. In practice, the watershed transform suffers from severe *over-segmentation*, that is, the creation of many small basins in regions with many local minima. Therefore, one usually has to use some preprocessing method to reduce the number of local minima, for example by first applying some smoothing filter.

As an application, we show in Fig. 22 segmentation of images of coffee beans, cf. Vincent (1990). The goal is to segment the image in non-overlapping regions each containing exactly one coffee bean. The most important steps to achieve this are the following. First segment the image in foreground and background by simple thresholding, and remove some small holes inside the beans, see Fig. 22(b). Note that directly using the connected components in this binary image will not work, because some of the beans overlap. To separate these overlapping beans we first apply a distance transform to the inverted image of Fig. 22(b). This will yield a single maximum in the centre of the isolated coffee beans, and two (or more) maxima in overlapping beans, cf. Fig. 22(c). Then invert the distance transformed image so that maxima becomes minima, and use these minima as points for starting a watershed algorithm. The resulting watershed lines are shown in Fig. 22(d). Finally, we show in Fig. 22(e) the original image with the watershed lines superimposed.

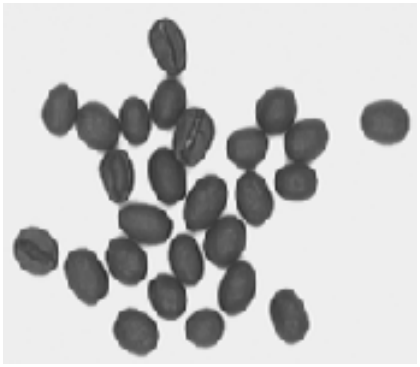
7.1.3 Region growing, split-and-merge

A merging method starts with many small regions subdividing the image. In each region one computes certain properties which represent membership to objects. Next boundaries are merged if their strength, defined according to certain criteria, is below a given threshold, that is, if the corresponding regions are similar enough. This process is repeated until no more merging takes place.

One can also start with the entire image as the initial region, and successively split regions into subregions if a region is not homogeneous enough. Since this method tends to produce suboptimal boundaries, one often uses a combination of splitting and merging operations, called the split-and-merge strategy.

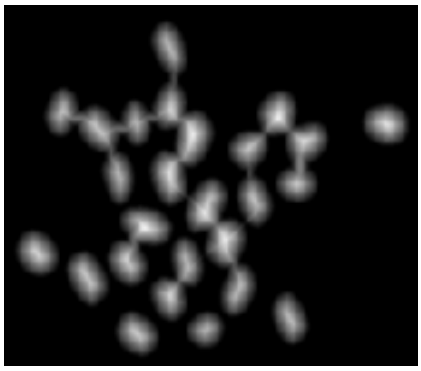
7.2 Boundary based approach

Boundary techniques attempt to find the edges directly by computing points with high gradient magnitude.



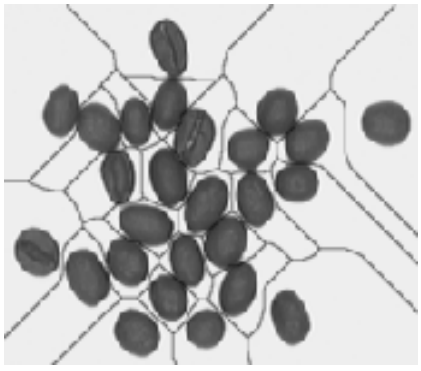
(a)

(b)



(c)

(d)



(e)

Figure 22: *Segmenting an image of coffee beans. (a): original image; (b): original after thresholding and small hole removal; (c): distance transform of the inverse of (b); (d): watershed lines of the inverse of (c); (e): original image with watershed lines superimposed.*

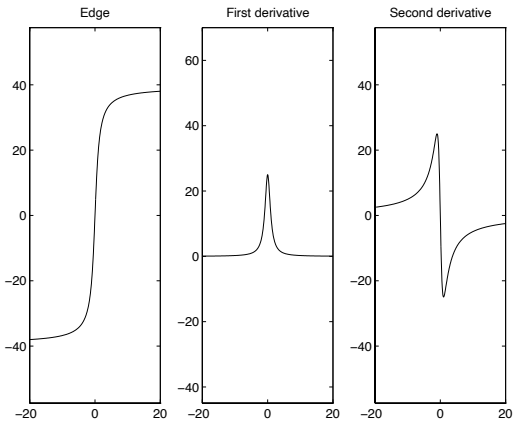


Figure 23: *Derivatives of an edge.*

7.2.1 Boundary tracking

After taking the gradient of the initial image, one locates the pixel with the highest value, and starts a boundary tracking process at this pixel. Then new boundary points are computed iteratively by searching in a local neighbourhood of the current point for the point—not yet chosen as a boundary point—with maximum grey level. If several candidate points are available, one chooses arbitrarily. To combat noise the image can be smoothed before tracking, or one may use a larger local window within which searching takes place.

7.2.2 Laplacian filtering

This filter was mentioned above as an sharpening filter for image enhancement, see Section 4.2. The Laplacian is a second derivative operator Δ defined by

$$\Delta f(x,y) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) f(x,y)$$

which is linear, shift-invariant and its transfer function is zero at the origin in frequency space. So after Laplacian filtering the image will have zero average grey value. The Laplacian filtering of an image will produce a zero-crossing at an edge, cf. Fig. 23. In the presence of noise, filters such as these which are based upon differentiation leads to very unstable results. One solution is to first convolve the image with a smoothing kernel, such as a Gaussian, prior to differentiation. All derivatives of the image undergo the same Gaussian smoothing process which is equivalent to convolving the image with derivatives of a Gaussian. The *Canny edge detector* computes the first derivatives of the Gaussian-smoothed image. Edges are identified as locations where the gradient magnitude has a maximum. Another often used edge detector is the *Marr-Hildreth operator*, which uses second derivatives; the convolution kernel $K_\sigma(x,y)$ is the Laplacian-of-

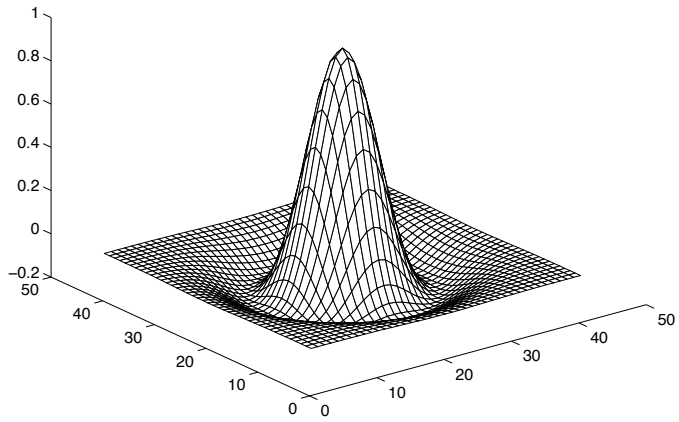


Figure 24: *Surface plot of the Laplacian-of-Gaussian function.*

Gaussian (LoG), with σ the standard deviation of the Gaussian kernel:

$$K_\sigma(x,y) = -\frac{2}{\pi\sigma^4}\left(1-\frac{r^2}{2\sigma^2}\right)e^{-r^2/2\sigma^2} \qquad (r^2 = x^2+y^2).$$

Edges of f are identified as *zero-crossings* of $K_\sigma f$. The Laplacian-of-Gaussian is a Mexican-hat shaped function, cf. Fig. 24.

7.3 Edge based approach

This approach consists of two phases: first determine for each pixel whether it is on the boundary of an object using some appropriate criterion. The result is called an *edge image*. This will in general not result in closed contours. To obtain these, a second step is required in which edge points are linked.

7.3.1 Detecting edge points

Edge points are determined by looking at magnitude and direction of the gradient of the input image. Most methods use local convolution with a set of directional derivative masks, as in the case discussed above for the Laplacian filter. Other examples of edge operators were presented in Section 4.2.

7.3.2 Edge linking

For strong edges and low noise one may threshold the edge image and apply a thinning (cf. Section 6.1.4) to the resulting binary image until closed contours are obtained with one-pixel

thickness. In general the edge image will contain gaps which must be closed. When the gaps are small one may use some heuristic search procedure for finding other pixels to link to. For sparse edge points one may use curve fitting to form boundaries. Straight-line fitting can be conveniently performed by using the *Hough transform*, which maps a straight line $y = mx + b$ to polar coordinates ρ, θ by the relation $\rho = x \cos\theta + y \sin\theta$. Consider a set of edge points (x_i, y_i) lying on a straight line with parameters ρ_0, θ_0. Each point is represented in ρ, θ space by a sinusoidal curve, but all these curves go through a common point ρ_0, θ_0. Thus, a straight line through a number of edge points can be found by looking for a local maximum of the histogram representing the points in ρ, θ space. Note that the Hough transform can also be used to detect other shapes, e.g. circles.

8 Description and measurement

After image segmentation the resulting collection of regions is usually represented and described in a form suitable for higher level processing. In this section we outline the most important representations based on shape or texture. Then, image descriptors based on the chosen representation are presented. A desirable property of these descriptors is that they should be insensitive to changes in size, translation or rotation. The actual measurement of features in digital images makes use of many of the techniques discussed above, such as linear or morphological image operators.

8.1 Representations

Roughly spoken, representations can be given in terms of the boundary or the interior of the regions (Castleman 1996, Ch. 11).

8.1.1 Boundary representations

Chain codes are strings of integers used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. The direction of each line segment is coded using a numbering scheme which may be adapted to the connectivity used. The accuracy of the straight-line representation depends on the spacing of the sampling grid.

A digital boundary can also be approximated by a polygon, possibly with minimum length. Computing such rubber band approximations directly within the grey value image has recently become popular under the name of active contour models, or *snakes* (Kass, Witkin & Terzopoulos 1988).

8.1.2 Region representations

An important shape representation of regions is obtained by computing a *skeleton*, based on the medial axis (Blum 1973), thinning algorithms (Section 6.1.4) or morphological skeletons, cf. Section 6.1.6. The goal of skeletonisation is reduction of information while keeping the essential characteristics of the image.

8.2 Descriptors

8.2.1 Boundary descriptors

Simple descriptors of boundaries are length, diameter (maximal distance between points on the contour) or curvature. Length can be simply approximated by counting the number of pixels along the contour. Curvature can be computed from the chain code.

Shape numbers of a boundary can be defined in terms of its chain code (Gonzalez & Woods 1992).

Fourier descriptors are based upon the representation of the boundary as a sequence $s(k) = x(k) + i\,y(k), k = 0, 1, \ldots N-1$, where $(x(k), y(k))$ are coordinates of points on the contour. Taking a discrete Fourier transform of the vector $(s(1), \ldots, s(N-1))$ yields N complex coefficients called the Fourier descriptors of the boundary. Approximations of the boundary are obtained by only using $M < N$ of the Fourier coefficients and performing an inverse discrete Fourier transform.

Fractal dimension of a (compact) set can be computed as follows (Barnsley 1988). Let $\mathcal{N}(A, \epsilon)$ denote the smallest number of closed balls of radius $\epsilon > 0$ needed to cover the set A. Then

$$D = \lim_{\epsilon \to 0} \left(\frac{\log(\mathcal{N}(A, \epsilon))}{\log(1/\epsilon)} \right)$$

is called the fractal dimension of A. The fractal dimension of a boundary can be estimated using morphological operations on real images through the concept of Minkowski dimension, which is based upon the Minkowski addition (Serra 1982, Ch. 5).

8.2.2 Regional descriptors

Simple descriptors based upon regions are area, perimeter or derived quantities such as perimeter squared divided by area, which is a *circularity measure* (Castleman 1996, Section 19.3).

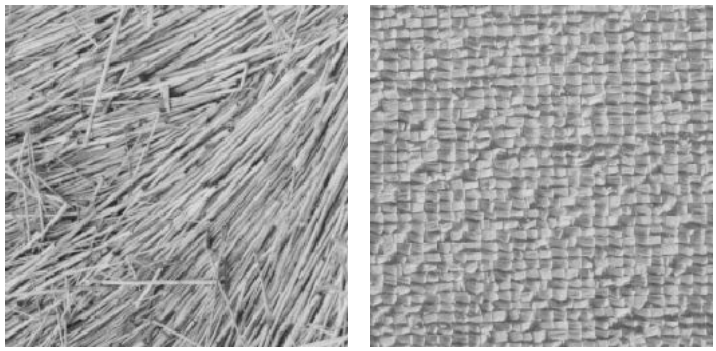
Topological descriptors are invariant to a large class of local deformations. Examples are: number of connected components; number of holes; or the genus or Euler number (number of connected components minus the number of holes). These quantities can be measured digitally be using erosions or hit-or-miss transforms as introduced in Section 6.1, see Haralick & Shapiro (1992).

Texture of a region refers to the spatial distribution of discrete grey value variations, described in terms of uniformity, coarseness, regularity and directionality. Some examples are given in Fig. 25. The three main approaches used to describe texture are: (i) statistical; (ii) structural, viewing a texture as an arrangement of texture primitives; and (iii) spectral, using the Fourier transform to detect global periodicities.

Moments of an image $f(x, y)$ are integrals over its domain with respect to polynomial weight functions:

$$M_{m,n} = \iint x^m\, y^n\, f(x, y)\, \mathrm{d}x\ \mathrm{d}y\,, \qquad m, n = 1, 2, \ldots$$

Invariant moments are combinations of several $M_{m,n}$ which are invariant to translation, rotation and scale-change.



(a) (b)

Figure 25: *Textures. (a): Straw. (b): Raffia.*

9 Summary

In this chapter we have introduced the basics of digital image processing, concentrating on the main concepts and terminology used in the field. Aspects of both linear and nonlinear image processing have been presented, such as image enhancement, spatial filtering, restoration, morphological image processing, segmentation and description.

Although many aspects have been treated only in a very global way, it is hoped that this chapter will provide to the reader sufficient background to understand the basis of the image processing techniques to be found in the later chapters of this volume.

References

Andrews, H. C. & Hunt, B. R. (1977), *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ.

Barnsley, M. F. (1988), *Fractals Everywhere*, Academic Press, New York.

Beucher, S. & Meyer, F. (1993), The morphological approach to segmentation: the watershed transformation, *in* E. R. Dougherty, ed., 'Mathematical Morphology in Image Processing', Marcel Dekker, New York, chapter 12, pp. 433–481.

Blum, H. (1973), 'Biological shape and visual science (part I)', *Journal of Theoretical Biology* **38**, 205–287.

Borgefors, G. (1984), 'Distance transformations in arbitrary dimensions', *Comp. Vis. Graph. Im. Proc.* **27**, 321–345.

Castleman, K. R. (1996), *Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Giardina, C. R. & Dougherty, E. R. (1988), *Morphological Methods in Image and Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Gonzalez, R. C. & Woods, R. E. (1992), *Digital Image Processing*, Addison Wesley, Reading, MA.

Haralick, R. M. & Shapiro, L. G. (1985), 'Survey : image segmentation techniques', *Comp. Vis. Graph. Im. Proc.* **29**, 100–132.

Haralick, R. M. & Shapiro, L. G. (1992), *Computer and Robot Vision (Volume I,II)*, Addison Wesley, Reading, MA.

Haralick, R. M., Sternberg, S. R. & Zhuang, X. (1987), 'Image analysis using mathematical morphology', *IEEE Trans. Patt. Anal. Mach. Intell.* **9**, 532–550.

Heijmans, H. J. A. M. (1994), *Morphological Image Operators*, Vol. 25 of *Advances in Electronics and Electron Physics, Supplement*, Academic Press, New York.

Jain, A. K. (1989), *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ.

Kass, M., Witkin, A. & Terzopoulos, D. (1988), 'Snakes: Active contour models', *International Journal of Computer Vision* **1**(4), 321–331.

Matheron, G. (1967), *Eléments pour une théorie des milieux poreux*, Masson, Paris.

Meijster, A. & Roerdink, J. B. T. M. (1996), Computation of watersheds based on parallel graph algorithms, *in* P. Maragos, R. W. Shafer & M. A. Butt, eds, 'Mathematical Morphology and its Applications to Image and Signal Processing', Kluwer Acad. Publ., Dordrecht, pp. 305–312.

Meyer, F. (1994), 'Topographic distance and watershed lines', *Signal Process.* **38**, 113–125.

Najman, L. & Schmitt, M. (1994), 'Watershed of a continuous function', *Signal Process.* **38**, 99–112.

Rosenfeld, A. (1970), 'Connectivity in digital pictures', *J. Ass. Comp. Mach.* **17**, 146–160.

Rosenfeld, A. & Kak, A. (1982), *Digital Picture Processing, Vols. I and II*, Academic Press, New York.

Rosenfeld, A. & Pfaltz, J. (1968), 'Distance functions on digital pictures', *Pattern Recognition* **1**, 33–61.

Serra, J. (1982), *Image Analysis and Mathematical Morphology*, Academic Press, New York.

Serra, J., ed. (1988), *Image Analysis and Mathematical Morphology. II: Theoretical Advances*, Academic Press, New York.

Serra, J. & Vincent, L. (1992), 'An overview of morphological filtering', *Circuits, Systems and Signal Processing* **11**, 47–108.

Sternberg, S. R. (1986), 'Grayscale morphology', *Comp. Vis. Graph. Im. Proc.* **35**, 333–355.

Vincent, L. (1990), Algorithmes Morphologiques a Base de Files d'Attente et de Lacets. Extension aux Graphes, PhD thesis, Ecole Nationale Supérieure des Mines de Paris, Fontainebleau.

Vincent, L. & Soille, P. (1990), 'Watersheds in digital spaces: an efficient algorithm based on immersion simulations', *IEEE Trans. Patt. Anal. Mach. Intell.* **13**(6), 583–598.