Contents

I Granulometries: Algorithms and Applications 3

1 Fast Opening Functions and Morphological Granulometries 3
   1.1 Introduction ...................................... 3
   1.2 Background, Previous Work ...................... 6
   1.3 Linear Granulometries in Binary Images ............. 6
   1.4 Granulometry Functions on Binary Images .......... 8
   1.5 Grayscale Granulometries ....................... 12
   1.6 Conclusion .................................. 17

2 Fast Grayscale Granulometry Algorithms 17
   2.1 Introduction, State of the Art .................... 20
   2.2 Linear Granulometries .......................... 21
   2.3 Granulometries with Maxima of Linear Openings .... 22
   2.4 Conclusion ................................ 24

3 Watersheds and Segmentation 28

4 Binary and Grayscale Particle Segmentation 28
   4.1 Introduction to Morphological Segmentation .......... 28

4.2 Maximal Balls, Skeletons, Ultimate Erosions .......... 31
   4.2.1 Definitions .................................. 31
   4.2.2 Quench Function ............................. 34
   4.2.3 Ultimate Erosion and Distance Function .......... 37
   4.2.4 Skeleton by Influence Zones .................... 42

4.3 Geodesic Transformations .......................... 42
   4.3.1 Introduction, Geodesic Distances ................. 42
   4.3.2 Geodesic Dilations and Erosions ................. 45
   4.3.3 Reconstruction and Applications ................ 45
   4.3.4 Grayscale Reconstruction ...................... 45
   4.3.5 Binary Segmentation .......................... 48

4.4 Watersheds and Grayscale Segmentation ............... 50
   4.4.1 Deriving a General Segmentation Approach ........ 50

5 Conclusions 62

III Morphological Algorithms 64

6 Introduction 64
   6.1 Discrete images and grids, notations .............. 65
   6.2 The distance function .......................... 65
   6.3 Estimating the quality of a morphological algorithm ...... 65
   6.4 Image structures and how to access them .......... 69
7 Parallel Algorithms 69
8 Sequential Algorithms 70
9 Loop and Chain Algorithms 75
10 Algorithms Based on Queues of Pixels 81
11 Conclusion and Summary 88
Part I
Granulometries: Algorithms and Applications


1. Fast Opening Functions and Morphological Granulometries

Abstract

Granulometries constitute one of the most useful and versatile sets of tools of morphological image analysis. They can be applied to a wide range of tasks, from feature extraction, to texture characterization, to size estimation, to image segmentation, etc. However, traditional granulometry algorithms—involving sequences of openings or closings with structuring elements of increasing size—are prohibitively costly on non-specialized hardware. This problem has prevented granulometries from reaching a high level of popularity in the image analysis community.

In this part, a comprehensive set of fast algorithms for computing granulometries in binary images is first proposed: linear granulometries (i.e., granulometries based on openings with line segments) constitute the easiest case, and are computed using image “run-length”. The 2-D case (granulometries with square or “diamond”-shaped structuring elements, or granulometries with unions of line-segments at different orientations) involves the determination of opening functions or granulometry functions. The grayscale case is then addressed, and a new algorithm for computing grayscale linear granulometries is introduced. This algorithm is orders of magnitude faster than any previously available technique. The techniques introduced in this part open up a new range of applications for granulometries, examples of which are described in the part.

1.1. Introduction

The concept of granulometry was introduced by G. Matheron in 1967 [65] as a new tool for studying porous media. The size of the pores in such media was characterized using series of openings with structuring elements of increasing size [100]. The theoretical study of these operations led Matheron to propose the following definition:

Definition 1 Let \( \Phi = (\phi_\lambda)_{\lambda \geq 0} \) be a family of image transformations depending on a unique parameter \( \lambda \). This family constitutes a granulometry if and only if the following properties are satisfied:

\[
\forall \lambda \geq 0, \quad \phi_\lambda \text{ is increasing,} \tag{1}
\]
\[
\forall \lambda \geq 0, \quad \phi_\lambda \text{ is anti-extensive,} \tag{2}
\]
\[
\forall \lambda \geq 0, \mu \geq 0, \quad \phi_\lambda \phi_\mu = \phi_{\mu \lambda} = \phi_{\max(\lambda, \mu)} \tag{3}
\]

Property (3) implies that for every \( \lambda \geq 0 \), \( \phi_\lambda \) is an idempotent transformation. Therefore, \( (\phi_\lambda)_{\lambda \geq 0} \) is nothing but a decreasing family of algebraic openings [100]. Conversely, one can prove that for any convex set \( B \), the family of the openings with respect to \( AB = \{ \lambda b \mid b \in B \} \), \( \lambda \geq 0 \), constitutes a granulometry [66].

More intuitively, suppose now that the transformations considered are acting on discrete binary images, or sets. In this context, a granulometry is a sequence of openings \( \phi_n \), indexed on an integer \( n \geq 0 \). Each opening is smaller than the previous one:

\[
\forall X, \quad \forall n \geq m \geq 0, \quad \phi_n(X) \subseteq \phi_m(X). \tag{4}
\]

The granulometric analysis of \( X \) with family of openings \( (\phi_n)_{n \geq 0} \) is often compared to a sifting
The granulometry function is central to the algorithms described in section 1.4. An example of square granulometry function is shown in Fig. 1.

The granulometries that have been described so far are often referred to as granulometries by openings. By duality, granulometries by closing can also be defined [136]; the granulometric analysis of a set $X$ with respect to a family of closings is strictly equivalent to the granulometric analysis of $X^C$ (complement of $X$) with the family of dual openings. Therefore, from now on, only granulometries by openings are considered. Similarly, these notions can be directly extended to grayscale images; in this context, the measure $m$ chosen is the “volume” of the image processed, i.e. the sum of all its pixel values.

The granulometric analysis of Fig. 1a with respect to a family of openings with squares (as was used for Fig. 1) is shown in Fig. 2. From the observed pattern spectrum, the typical size of the beans (as size of the largest square a bean can contain) in this image can easily be derived. Granulometries therefore allow one to extract size information without any need for prior segmentation: the beans in this image are highly overlapping, yet their size can be estimated without individually identifying each bean.

Granulometries have been used for a variety of other image analysis tasks, including shape characterization and feature extraction (see for example [144]), texture classification (see [25]), and even segmentation (see for example [31] and Fig. 3). Nonetheless, until recently, granulometric analysis involved performing a series of openings and/or closings of increasing size, which is prohibitively expensive for most applications, unless dedicated hardware is used.

In the next section, we first briefly review the literature on granulometry algorithms. A comprehensive set of fast binary granulometry algorithms is then proposed: section 1.3 is concerned with the simple case of linear granulometries. In section 1.4, more complex cases, such as granulometries with squares, are discussed. The corresponding algorithms involve the determination of granulometry functions, for which fast algorithms are proposed. Following prop. 4's result, pattern spectra are then derived by simple histogramming.
Figure 1. (a) original binary image of coffee beans, (b) square granulometry function of this image, in which dark regions correspond to higher pixel value (c) level lines of the granulometry function.

Figure 2. (a) Successive openings of a Fig. 1 using squares of increasing size as structuring elements. (b) Corresponding granulometric curve, or pattern spectrum: the peak at size 20 is indicative of the typical size of the beans in original image.
Lastly, the grayscale case is considered in section 1.5 a new algorithm is introduced for computing linear grayscale granulometries. The algorithm is several orders of magnitude faster than any previously available technique. It makes it therefore possible to use granulometries where previously unthinkable. We illustrate this point by using this new algorithm to efficiently extract size information directly from a grayscale image.

1.2. Background, Previous Work

The literature on mathematical morphology is not short of algorithms for computing erosions and dilations, openings and closings, with various structuring elements, in binary and in grayscale images. Reviewing them would be beyond the scope of this part. But no matter how efficient an opening algorithm is used, determining a pattern spectrum using a sequence of openings is a very time-consuming task given the number of operations involved. Furthermore, since the size of the structuring element increases with \( n \), so does the computation time of the corresponding opening. Even if we assume that the computation time of \( \phi_n(X) \) (\( n \)-th opening in the series) can be done in constant time (which is not always true depending on the structuring element and on the opening algorithm used), determining the pattern spectrum up to size \( n \) using openings is still an \( O(n) \) algorithm.

The few granulometry algorithms found in literature only deal with the binary case, and have in common the use of granulometry functions as an intermediate step. The algorithm proposed by Yuan [145] for determining binary square granulometries consists in first determining the quench function of the original set \( X \). The quench function maps each pixel \( p \) of the skeleton (medial axis) \( S_X \) of \( X \) to the size (radius) \( S_X(p) \) of the corresponding maximal square. An example is shown in Fig. 4 (See [124] for more details on these concepts). In a second step, each pixel \( p \) of the skeleton is replaced by a square centered at this pixel, with size \( S_X(p) \), and graylevel \( S_X(p) + 1 \). The pixelwise maximum of all these squares provides the granulometry function of \( X \). This algorithm is faster than the brute force method described in the previous paragraph, but still requires a significant amount of image scans. In addition, the more complicated the image or the larger the objects in it, the longer this method takes.

Surprisingly, a better algorithm can be found in an earlier paper by LaR [57], in which the author devotes a few lines to the description of a sequential algorithm [88, 127] based on the distance function [89, 20], and also using the granulometry function as an intermediate step. This algorithm still provides one of the most efficient implementations to date for binary granulometries with structuring elements such as squares and hexagons. In section 1.4, this technique is for the first time described in detail, and is extended to other types of binary granulometries.

The algorithm proposed in 1992 by Haralick et al is interesting in that it allows in principle to compute granulometry functions with respect to any family of homothetic elements. However, for simple structuring elements such as squares, this technique is not as efficient as the one mentioned in the previous paragraph, because its elementary steps (propagation and merging of lists of "propagators") are rather computationally intensive, therefore relatively slow.

1.3. Linear Granulometries in Binary Images

Linear granulometries in binary images constitute the simplest possible case of granulometries. Let us for example consider the horizontal granulometry, i.e., the granulometry by openings with the \( (L_n)_{n \geq 0} \) family of structuring elements, where:

\[
L_n = \underbrace{\bullet \cdot \cdot \cdot \cdot \bullet}_{n+1 \text{ pixels}}
\]

From now on, we use the convention that the center of a structuring element is marked using a thicker dot than is used for the other pixels. Note that the location of the center of the structuring elements used has no influence on the resulting granulometry.

Let us analyze the effect of an opening by \( L_n \), \( n \geq 0 \) on a discrete set \( X \) (binary image). The

\[1\text{st element does not even have to be convex!} \]
Figure 3. Segmentation of tri-phased binary texture using local granulometric moments (from [31]).

Figure 4. Quench function of Fig. 1a, dilated for clarity; dark skeleton pixels correspond to large values of the quench function.
following notations are used from now on: the neighbors of a given pixel \( p \) in the square grid are denoted \( N_0(p) \), \( N_1(p) \), \ldots, \( N_7(p) \), and the eight elementary directions are encoded in the following way:

\[
\begin{array}{c}
3 & 2 & 1 \\
4 & \bullet & 0 \\
5 & 6 & 7
\end{array}
\]

For a direction \( d \in \{0, 1, \ldots, 7\} \) and \( k \geq 0 \), we denote by \( N_d^{(k)}(p) \) the \( k \)-th order neighbor of pixel \( p \) in direction \( d \):

\[
N_d^{(0)}(p) = p, \quad \text{and} \quad k > 0 \implies N_d^{(k)}(p) = N_d(N_d^{(k-1)}(p)).
\]

(9)

The opposite of direction \( d \) is denoted \( \bar{d} \). For example, if \( d = 3 \), then \( \bar{d} = 7 \).

**Definition 5** The ray in direction \( d \) at pixel \( p \) in set \( X \) is given by:

\[
r_{X,d}(p) = \{ N_d^{(k)}(p) \mid k \geq 0 \text{ and } \forall 0 \leq j \leq k, N_d^{(j)}(p) \in X \}.
\]

(10)

With each pixel \( p \in X \), we also associate a run in direction \( d \), defined as the union of the rays in direction \( d \) and in direction \( \bar{d} \).

**Definition 6** The run in direction \( d \) at pixel \( p \) in set \( X \) is given by:

\[
R_{X,d}(p) = r_{X,d}(p) \cup r_{X,\bar{d}}(p).
\]

(11)

The number of pixels in a run \( R \) will be called length of this run and denoted \( l(R) \).

The following proposition is immediate:

**Proposition 7** The opening of \( X \) by \( L_n \), denoted \( X \circ L_n \), is the union of the horizontal runs \( R_{X,0}(p) \) whose length is strictly greater than \( n \):

\[
X \circ L_n = \bigcup_{p \in X} \{ R_{X,0}(p) \mid l(R_{X,0}(p)) > n \}.
\]

(12)

Therefore, any horizontal run of length \( n \) is left unchanged by all the openings with \( L_k \), \( k < n \), and is removed by any opening with \( L_k \), \( k \geq n \). Hence, the corresponding pattern spectrum \( PS_0 \) satisfies:

\[
PS_0(X)(n) = \text{card}\{ p \in X \mid l(R_{X,0}(p)) = n \}.
\]

(13)

An extremely efficient 1-scan horizontal granulometry algorithm is easily derived from this formula:

**Algorithm** horizontal binary granulometry

- Initialize pattern spectrum: for each \( n > 0 \), \( PS[n] \leftarrow 0 \)
- Scan each line of image from left to right.
- In this process, each time a run \( R \) is discovered, do:
  \[
  PS[l(R)] \leftarrow PS[l(R)] + l(R);
  \]

In applications where directional information is of interest, this algorithm provides a very useful and efficient way to extract size information characterizing the image under study. Consider, for example, Fig. 5a, which is a binary image of lamellar eutectics; in [95], M. Schmitt proposed a variety of methods for extracting the defect lines present in this image. Different methods used different kind of information about this image, and some required a knowledge of the typical width of the lamellae. This width can be accurately estimated by adapting the previous algorithm to the computation of linear granulometries at +45 degree orientation (direction perpendicular to the lamellae). The resulting pattern spectrum is shown in Fig. 5b, and its peak at 3 indicates that the typical width of the lamellae is of 3 pixels.

### 1.4. Granulometry Functions on Binary Images

For non 1-D granulometries, the direct approach described in the previous section becomes intractable. Consider for example the case of a granulometry \( (\phi_n)_{n \geq 0} \) where \( \phi_n \) is a maximum of openings with the horizontal segment \( L_n \) and its vertical counterpart \( L_n^\perp \). For each pixel, it becomes necessary to know the size of the horizontal run as well as the vertical run it belongs to.

Linear granulometry functions are therefore the required step. Given the horizontal and the vertical granulometry functions of \( X \), the granulometry function of \( X \) corresponding to the \((\phi_n)_{n \geq 0}\) of previous paragraph is simply obtained as a pixel-wise maximum. More generally, the same is true.
Figure 5. Binary image of lamellar eutectics (a) and its granulometric curve using line segments at +45 degrees orientation (b)

for any two granulometry functions, and the following proposition can be stated:

**Proposition 8** Let

\[ \Phi = (\phi_n)_{n \geq 0} \quad \text{and} \quad \Psi = (\psi_n)_{n \geq 0} \]

be two granulometries. Then, \( \max(\Phi, \Psi) = (\max(\phi_n, \psi_n))_{n \geq 0} \)

is also a granulometry and for any set \( X \):

\[ G_{\max(\Phi, \Psi)}(X) = \max(G_\phi(X), G_\psi(X)). \]

Determining the linear granulometry function of a binary image is a relatively straightforward task. Take for example the horizontal case: like in previous section, the principle of the granulometry function algorithm is to locate each horizontal run. But now, in addition, each run \( R \) gets also tagged with its length \( l(R) \). This involves scanning the black pixels of the image twice, and the white pixels only once. The resulting algorithm, hardly more time consuming than the one described in the previous section, is given below.

**Algorithm:** horizontal granulometry function

- Scan each line of original binary image \( I \) from left to right:
  - \( \triangleright p \leftarrow \text{first black pixel of current line, or end-of-current-line if line is empty;} \)
  - \( \triangleright \text{while } p < \text{end-of-current-line, do} \)
  - \( \triangleright p \leftarrow \text{first black pixel} \)

Examples of linear granulometry functions are shown in Fig. 6 and in Fig. 7.

The case of truly 2-D binary granulometry functions is the next level up in complexity. In the rest of this section, we first focus on granulometry functions \( G_S(X) \) based on openings with the homothetics of elementary square \( S \), then we deal with the case of granulometry functions \( G_D(X) \) based on the elementary “diamond” shape \( D \).

\[ S = \bullet \bullet ; \quad D = \bullet \bullet \bullet \bullet \]

Together with the linear case, these granulometries cover 99% of all practical needs.

Like Haralick’s algorithm [44], the first step of the present one consists in computing what some
Figure 6. Linear granulometry function of image 5(a), whose histogram directly provides the curve of Fig. 5(b).

Figure 7. (a) Horizontal granulometry function; (b) vertical granulometry functions. (c) Pointwise maximum of these two images provides the granulometry function corresponding to maxima of openings with vertical and horizontal line segments.
authors have called a generalized distance function \[3, 45\]. Let \( B \) be an arbitrary structuring element containing its center. Let

\[
nB = B \oplus B \oplus \ldots B
\]

\[\text{ntimes}\]

denote the structuring element “of size \( n \)”. Let also \( \varepsilon_B \) denote the erosion by structuring element \( B \) \[100\]:

\[
\varepsilon_B(X) = X \ominus B.
\]

**Definition 9** The generalized distance function \( d_B(X) \) with respect to the family of structuring elements \((nB)_{n \geq 0}\) assigns to each pixel \( p \in X \) the smallest \( k > 0 \) such that \( p \notin \varepsilon_{kB}(X) \):

\[
d_B(X)(p) = \min\{k > 0 \mid p \notin \varepsilon_{kB}(X)\}. \tag{16}
\]

Generalized distance functions are determined using sequential algorithms that are straightforwardly derived from the original algorithm proposed by Rosenfeld \[88, 89\]. When the center of the structuring element is in the bottom-right corner of element \( B \) (last pixel met in a raster-order scan of this element), the distance function \( d_B(X) \) can be computed in one single raster scan.

In the case where \( B = S \) (see Eq. (15)), the following algorithm can be proposed:

**Algorithm:** Generalized dist. func. with square \( S \)
- **Input:** binary image \( I \) of set \( X \)
- **Scan** \( I \) in raster order;
  - \( \Leftrightarrow \) Let \( p \) be the current pixel;
  - \( \Leftrightarrow \) if \( I(p) = 1 \) (\( p \) is in \( X \)):
    - \( I(p) \leftarrow \min\{I(N_4(p)), I(N_3(p)), I(N_2(p))\} + 1; \)

An example of generalized distance function resulting from this algorithm is shown in Figs. 8a–b. A way to interpret the result is to say that, for each pixel \( p \), if one was to translate structuring element \( [d_S(X)(p)]S \) so that its center coincides with \( p \), this translated element—denoted \( p + [d_S(X)(p)]S \)—would be entirely included in \( X \). However, \( p + [d_S(X)(p)] + 1S \notin X \). We can therefore state the following proposition:

**Proposition 10** The granulometry function \( G_S(X) \) is obtained from \( d_S(X) \) as follows:

\[
\forall p \in X, \quad G_S(X)(p) = \max\{d_S(X)(q) \mid p \in (q + d_S(X)(q)S)\}. \tag{17}
\]

In algorithmic terms, we can compute \( G_S(X) \) by propagating the value \( d_S(X)(p) \) of each pixel \( p \) over the square \( p + d_S(X)(p)S \), and then by taking the pixelwise maximum of the values propagated at each pixel.

In the technique proposed by Haralick et al \[44\], this propagation step is achieved via an antiraster scan of the distance function image, in which, at each pixel, a list of propagated values is maintained. In the particular case of square granulometry function \( G_S(X) \), computing the value at pixel \( p \) as well as the list of propagated values at \( p \), requires a merging of the lists of propagated values at pixels \( N_0(p), N_6(p), \) and \( N_7(p) \).

This merging step turns out to be expensive, and in the case of square granulometry function \( G_S(X) \), a less general, but much more efficient technique can be proposed. This technique takes advantage of the fact that square \( S \) can be decomposed into the Minkowski addition of the two elementary line segments \( E_1 \) and \( E_2 \):

\[
S = \odot \odot = \odot \odot = E_1 \oplus E_2 \tag{18}
\]

Therefore, the complex propagation step of the granulometry function algorithm described in \[44\] can in fact be decomposed into two much simpler propagations, with substantial speed gain. The distance function extraction step is followed by two linear propagation steps that are identical, except that one propagates distance values leftward in each line, whereas the other one propagates values upward in each column.

The algorithm for right-to-left propagation of distance values is given below. Its principle is to propagate each pixel value \( I(p) \) to the left \( I(p) \leftarrow \) 1 times, or until a larger value \( v \) is found, in which case the list of propagated values is reset to this value... The algorithm maintains an array propag containing the number of times each value remains to be propagated.

**Algorithm:** Left propag. of dist. values of \( d_S(X) \)
• Input: image \( I \) of the generalized distance function \( d_\varphi(X) \);
• For each line of the image, do:
  \( \Leftrightarrow \) Initializations: \( \text{maxval} \leftarrow 0 \) (current maximal value propagated);
  \( \Leftrightarrow \) Scan line from right to left:
    Let \( p \) be the current pixel;
    If \( I(p) \neq 0 \):
      If \( I(p) > \text{maxval} \):
        \( \text{maxval} \leftarrow I(p) \);
        \( \text{propag}[I(p)] \leftarrow I(p) \);
      \( \forall 0 < i < \text{maxval} \), \( \text{propag}[i] \leftarrow \text{propag}[i] \equiv 1 \);
      \( \text{maxval} \leftarrow \text{largest } i \leq \text{maxval} \) such that \( \text{propag}[i] \equiv 0 \);
    \( I(p) \leftarrow \text{maxval} \)

A few implementation tricks can speed up computation by substantially reducing the number of times the entire array \( \text{propag} \) is scanned per scan-line. Their description would be beyond the scope of this part. The resulting algorithm is quasi-linear with respect to the number of pixels in the image, and is almost independent of object size (see Table 1). Using again the coffee bean image as running example, the result of this propagation step is shown in Figs. 8c–d, and the final granulometry function obtained after upward propagation in each column is shown in Figs. 8e–f.

This algorithm can be adapted for granulometry functions with any structuring element that can be decomposed as a Minkowski addition of the elementary line segments \( E_1, E_2, E_3, \) and \( E_4 \) (see Eqs. (18) and (19)). Furthermore, it extends to the computation of hexagonal opening functions in the hexagonal grid [57].

In square grids however, the elementary “diamond” structuring element \( D \) cannot be decomposed as Minkowski addition of elementary line segments. The closest “approximation” is obtained with:

\[
D = \bullet \oplus \bullet \neq \bullet \oplus \bullet = E_3 \oplus E_4, \quad (19)
\]

and does not contain the central pixel of \( D \)!

Therefore, starting from distance function \( d_\varphi(X) \) and using the propagation algorithm in the Southwest-Northeast (SW-NE) direction, then in the Southeast-Northwest (SE-NW) direction results in an incorrect propagation function, as illustrated by Fig. 9. Correct “diamond” granulometry functions can nevertheless be obtained with this technique if SW-NE and SE-NW propagation steps are followed by a “hole-filling” step in which each pixel \( p \) such that

\[ \forall i \in \{0, 2, 4, 6\}, \quad I(N_i(p)) = I(p) + 1 \]

is given value \( I(p)+1 \). An example of such granulometry function is shown in Fig. 10.

Table 1 summarizes the speed of these granulometry functions on the \( 256 \times 256 \) coffee bean image used as running example. We chose not to compare these timings with those of traditional opening-based algorithms. The speed of the latter algorithms can indeed vary tremendously depending on the quality of the implementation. Note however that for this coffee bean image, which has approximately 30000 black pixels, Haralick’s algorithm [44] takes between 0.5s and 0.6s to compute the square granulometry function shown in Fig. 8e, on a Sparc Station 2. This workstation being between two and three times slower than a Sparc Station 10, we can conclude that the algorithm proposed in the present part is between three and four times faster.

1.5. Grayscale Granulometries

Grayscale granulometries are potentially even more useful than binary ones, because they enable the extraction of information directly from grayscale images. A number of theoretical results have been published on them (see e.g. [52]); however, since until today, no efficient technique was available to compute grayscale granulometries, they have not been used very much in practice. In this section, we remedy this situation and introduce a new algorithm for computing linear grayscale granulometries. A follow-up to this part describes this algorithm in greater detail, and also deals with the case of truly 2-D grayscale granulometries [133].

Like in section 2.2, let us for example deal with the horizontal case. The structuring elements
Figure 8. Computation of granulometry function using square structuring elements. (a) generalized distance function; (b) level lines; (c) propagation of values from right to left; (d) level lines; (e) final granulometry function; (f) level lines of granulometry function.

Figure 9. When using “diamonds” as structuring elements, the two propagation steps of the granulometry function algorithm need to be followed by a “hole-filling” step.
Figure 10. Computation of granulometry function using “diamond” structuring elements. (a) level lines of generalized distance function; (b) level lines of relief obtained after SW to NE propagation; (c) level lines of final granulometry function.

Table 1
Execution time of various granulometry function algorithms on the 256 × 256 coffee bean image, measured on a Sun Sparc Station 10 workstation.

<table>
<thead>
<tr>
<th>Type of granulometry function</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontal</td>
<td>0.018s</td>
</tr>
<tr>
<td>max in 4 directions</td>
<td>0.207s</td>
</tr>
<tr>
<td>square</td>
<td>0.085s</td>
</tr>
<tr>
<td>“diamond”</td>
<td>0.094s</td>
</tr>
</tbody>
</table>
considered are the $l_n$'s of equation (8). Let $I$ be a discrete grayscale image.

**Definition 11** A horizontal maximum $M$ of length $l(M) = n$ in grayscale image $I$ is a set of pixels $\{p, N_0^{(1)}(p), N_0^{(2)}(p), \ldots, N_0^{(n-1)}(p)\}$ such that

$$\forall i, \ 0 < i < n, I(N_0^{(i)}(p)) = I(p) \quad \text{and} \quad I(N_0^{(n)}(p)) < I(p).$$

The study of how such maxima are altered through horizontal openings is at the basis of the algorithm introduced here. The following proposition holds:

**Proposition 12** Let $M$ be a horizontal maximum of $I$. Let $p_l \in M$ and $p_r \in M$ respectively denote the extreme left pixel and the extreme right pixel of $M$. Let $n = l(M)$ be the length of this maximum. Then, for any $p \in M$:

$$\forall k < n, \quad (I \circ L_k)(p) = I(p),$$

for $k = n$, $(I \circ L_n)(p) = \max\{I(N_4(p_l)), I(N_0(p_r))\} < I(p),$ (22)

$$\forall k > n, \quad (I \circ L_k)(p) < I(p).$$

Intuitively, this means that any opening of $I$ by a line segment $L_k$ such that $k < n$ leaves this maximum unchanged, whereas for any $k > n$, all the pixels of $M$ have a lower value in $I \circ L_k$ than in $I$. Furthermore, we can quantify the effect of an opening of size $n$ on the pixels of this maximum: the value of each pixel $p \in M$ is decreased from $I(p)$ to $\max\{I(N_4(p_l)), I(N_0(p_r))\}$. In granulometric terms, the contribution of maximum $M$ to the $n$'s bin of the horizontal pattern spectrum $PS_0(I)$ is:

$$n \times [I(p) \leftrightarrow \max\{I(N_4(p_l)), I(N_0(p_r))\}].$$

This is illustrated by Fig. 11.

In addition, the effect of the horizontal opening of size $n$ on $M$ results in a new “plateau” of pixels being created at altitude $\max\{I(N_4(p_l)), I(N_0(p_r))\}$. This plateau may or may not be itself a maximum of $I \circ L_n$.

Further to these remarks, the principle of the introduced grayscale granulometry algorithm is to scan the lines of $I$ one after the other. Each horizontal maximum $M$ of the current line is identified, and its contribution to $PS_0(I)(l(M))$ is determined. If it turns out that after opening of size $l(M)$, the new plateau formed is still a maximum, the contribution of this maximum to the pattern spectrum is computed as well. The process is iterated until the plateau formed by opening is no longer a maximum, or until it becomes equal to the entire scanline considered. The next maximum of the current line is then considered, etc. This process is illustrated by Fig. 12.

At any time during this process, the left and right pixels of each “maximal region” processed are stored, so that this region does not have to be scanned again later. The resulting algorithm is therefore linear with respect to the number of pixels in the image, and in the worst case, each image pixel is only scanned twice. In practice, the execution time varies only slightly from image to image, depending on the number and complexity of maximal regions found in each scanline.

We compared the speed of this algorithm to the traditional opening-based technique. For this latter, a highly optimized opening algorithm was used, which is linear with respect to the number of pixels in the image, and whose speed is (almost) independent of the length of the line segment used as structuring element. Both original $512 \times 512$ weld images of Fig. 13 were used for this comparison. As illustrated by table 2, the new algorithm introduced in this part is up to three orders of magnitude faster!

The speed of this new algorithm opens up a range of new applications for grayscale granulometries. Traditionally, the practical problems granulometries have been used to address dealt with either texture discrimination, or feature extraction for object recognition. In the first case, either computation time was not an issue, or the discrimination task could be performed off-line. In the second case, granulometries were computed on very small images (e.g. characters), so that computation time could remain reasonable.

With the algorithm introduced in this section, it becomes possible to use grayscale granulometries more systematically. These tools provide indeed an efficient and accurate way to extract...
Figure 11. Horizontal cross section of $I$ with a maximum $M$. The shaded area, of volume $(h' \times h) \times l(M)$ shows the local contribution of this maximum to the $l(M)$-th bin of the horizontal pattern spectrum.

Figure 12. How the contribution of the “maximal region” surrounding a maximum $M$ is determined in the computation of the pattern spectrum.

Table 2
Execution time of traditional opening-based technique and of introduced algorithm for the computation of a grayscale granulometry with horizontal line segments. The $512 \times 512$ images of Fig. 13 were used for this comparison, done on a Sun Sparc Station 10.

<table>
<thead>
<tr>
<th></th>
<th>Traditional (size 1 to 512)</th>
<th>Traditional (size 1 to 30)</th>
<th>New algo (size 1 to 512)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weld image (a)</td>
<td>204s</td>
<td>13.0s</td>
<td>0.248s</td>
</tr>
<tr>
<td>Weld image (b)</td>
<td>204s</td>
<td>12.7s</td>
<td>0.200s</td>
</tr>
</tbody>
</table>
global size information directly from a grayscale image. Extracting this information is sometimes a goal in itself; but this size estimation can also be essential to calibrate the parameters of, e.g., an image segmentation algorithm, thereby greatly enhancing the robustness of this algorithm.

Figs. 13a–b are used to illustrate how grayscale granulometries can be used to estimate size information\(^2\). These figures represent welds at a high magnification. The quality of these welds is related to the size, shape, and organization of the light patterns observed in Figs. 13a–b. To estimate the size of the typical patterns in each image, linear granulometries were used, both in the vertical and in the horizontal direction. The resulting pattern spectra are shown in Figs. 13c–d. First, one can observe that the horizontal granulometric curve is very similar to the vertical one: we conclude that the patterns in images 13a and 13b do not have any preferential orientation. Second, the curves in Fig. 13c exhibit a well-marked peak for size 4, whereas the peak of the curves in Fig. 13d is found for size 12. We conclude that the typical width and height of the patterns in Figs. 13a and in Fig. 13b is of 4 pixels and 12 pixels respectively.

Another example of application is shown in Fig. 14. Images 14a and 14b respectively show X-rays of a healthy lung and of a lung exhibiting signs of the “miner’s disease”. These two images look very different from a texture point of view, thought it would be hard for a segmentation algorithm to specifically extract the “objects” that make image b different from image a. Essentially, the former image shows a very smooth texture while the latter is scattered with white nodules, which makes the texture rougher. Grayscale granulometries are the instrument of choice to discriminate between these two types of textures. To that end, M. Grimaud, author of the original study, had used grayscale granulometries by openings and closings with squares. The resulting pattern spectra for both X-ray images, shown in Fig. 14a, are extremely different from one another. However, in order to effectively discriminate between these two images, expensive grayscale granulometries with squares are not necessary: using the algorithms introduced here to extract horizontal grayscale granulometric curves is just as effective, as shown in Fig. 14d, and is orders of magnitude faster.

1.6. Conclusion

Even though the concept of granulometry was introduced over 25 years ago, the computation time required to extract granulometric curves made it impossible to use these curves for most practical applications. In this part, a comprehensive set of fast algorithms to compute granulometries in binary images was first described. Most of these algorithms use opening functions as an intermediate step, and they are shown to be faster than any previously available method.

A new algorithm for computing granulometries of grayscale images using openings or closings with linear structuring elements is also introduced. This algorithm is so much faster (up to three orders of magnitude) than any previously available technique that it makes it possible to use grayscale granulometries on a “routine” basis. Examples of application show how these granulometries can be used to extract global size information directly from a grayscale image. These granulometries being one-dimensional, they can also be successfully used on 1-D signals. In [133], this granulometry algorithm is generalized to the true 2-D case. It is expected these new algorithms will greatly contribute to popularize the use of grayscale granulometries in the image and signal analysis community.

2. Fast Grayscale Granulometry Algorithms

Abstract

Granulometries constitute an extremely useful set of morphological operators, applicable to a variety of image analysis tasks. Traditional granulometry algorithms involve sequences of openings or closings of increasing size, and are therefore very slow on non-dedicated hardware. Efficient techniques have been proposed to compute granulometries in binary images, based on the concept of opening functions. In the present section,
Figure 13. Using linear grayscale granulometries to estimate object size without prior segmentation. Curve (c) clearly indicates that the typical width and height of the white patterns in image (a) is 4 pixels. Similarly, curve (d) shows that the typical width/height of the patterns in (b) is 12.
Figure 14. Example of texture differentiation using grayscale granulometries by openings and closings. The pattern spectra of (a) and (b) are completely different, whether linear or square granulometries are used (original study due to M. Grimaud).
a class of algorithms for computing granulometries in grayscale images is introduced. The most advanced among them are based on the new concept of opening tree. These algorithms are several orders of magnitude faster than traditional techniques, thereby opening up a range of new applications for grayscale granulometries.

2.1. Introduction, State of the Art

In a variety of image analysis problems, one is interested in extracting the size distribution of the “objects” or “structures” present in an image. In 1967, Matheron formally characterized meaningful size distributions by introducing the concept of granulometry [65, 66]:

**Definition 13** A granulometry is a family of openings \( \Phi = (\phi_\lambda)_{\lambda \geq 0} \) satisfying:
\[
\forall \lambda \geq 0, \mu \geq 0, \quad \lambda \geq \mu \Rightarrow \phi_\lambda \leq \phi_\mu.
\] (24)

Performing the granulometric analysis of an image \( I \) with \( \Phi \) consists in mapping each size \( \lambda \) to a measure of the opened image \( \phi_\lambda(I) \). Similarly, anti-granulometries, or granulometries by closings, can be defined as families of increasing closings. For more details on granulometries, see [66].

In the discrete case, a granulometry is a decreasing family of openings \( \Phi = (\phi_n)_{n \geq 0} \), indexed on an integer parameter \( n \). Denote by \( m(I) \) the measure of a discrete image \( I \): \( m(I) \) is the area of \( I \) in the binary case\(^3\) (number of “on” pixels) and the volume of \( I \) in the grayscale case (sum of pixel values). The granulometric analysis of \( I \) with \( \Phi \) results in a granulometric curve:

**Definition 14** The granulometric curve or pattern spectrum [61] of an image, \( I \), with respect to granulometry \( \Phi = (\phi_n)_{n \geq 0} \) is the mapping \( \text{PS}_\Phi(I) \) given by:
\[
\text{PS}_\Phi(I)(n) = m(\phi_n(I)) \iff m(\phi_{n-1}(I)),
\] (25)
for each \( n > 0 \).

In practice, the most useful granulometries are based on openings (or closings) with the homothetics of a simple convex structuring element \( B \). Typically, \( B \) is a line segment, a square, or a hexagon. In such cases, we talk about “linear granulometries,” “square granulometries”, or “hexagonal granulometries” respectively. Granulometries based on maxima of openings with line segments at different orientations are also commonly used.

Until recently, determining a granulometric curve involved computing a sequence of openings of increasing size, which is very time consuming no matter how efficient the opening algorithm. Even when \( \phi_n(I) \) (n-th opening in the series) can be computed in constant time (for a fixed image size), determining the pattern spectrum using openings of size 1 through \( n \) is still an \( O(n) \) algorithm. For most applications, this is prohibitively costly, unless specialized hardware is used for computing the successive openings.

Over the past few years, a number of algorithms have been proposed for fast granulometries in binary images (see [57, 145, 44] and more recently [134]). These algorithms typically involve as an intermediate step the extraction of the “opening” function of the binary image studied [134]: this function maps each pixel with the size of the first opening that removes it from the image. It can be computed efficiently for all classic structuring element shapes, and the histogram of the resulting grayscale image provides the desired pattern spectrum. An example was shown earlier in Fig. 1.

This section treats the case of grayscale granulometries, which has not been previously addressed in literature. In section 2.2, a fast linear grayscale granulometry algorithm is first described (this algorithm was already briefly mentioned in [134]). Then, in section 2.3, the concept of opening trees is introduced as a way to compactly encode all the successive linear openings of a grayscale image. This concept is at the basis of another set of new algorithms for the fast computation of granulometries with maxima of linear openings, as well as approximations of square granulometric curves. Timing results prove that the introduced algorithms are several orders of magnitude faster than traditional techniques.

---

\(^3\)In the binary case, it may be useful for some granulometries to define \( m(I) \) as the number of connected components in \( I \) [27].
2.2. Linear Granulometries

In this section, the case of granulometries using openings with line segments is considered\(^4\). For simplicity, let us assume that the line segments used as structuring elements are horizontal (the algorithm easily extends to any orientation). The opening \(\phi_n\) in the series is the opening with structuring element \(L_n\):

\[
L_n = \underbrace{\bullet \bullet \cdots \bullet}_{n+1 \text{ pixels}}
\]  

(26)

Let us analyze the effect of an opening by \(L_n\), \(n \geq 0\) on a grayscale image \(I\). Denote by \(N_l(p)\) and \(N_r(p)\) respectively the left and the right neighbors of a pixel \(p\).

**Definition 15** A horizontal line segment \(S\), of length \(l(S)\), is a set of pixels \(\{p_0, p_1, \ldots, p_n\}\) such that for \(0 < i < n\), \(p_i = N_r(p_{i-1})\).

**Definition 16** A horizontal maximum \(M\) of length \(l(M) = n\) in grayscale image \(I\) is a horizontal line segment \(\{p_0, p_1, \ldots, p_n\}\) such that:

\[
\forall i, 0 < i < n, \quad I(p_i) = I(p_0) \quad \text{and} \quad I(N_l(p_0)) < I(p_0), \quad I(N_r(p_{n-1})) < I(p_0) \quad \text{(27)}
\]

This notion is the 1-D equivalent of the classic regional maximum concept \(^{[59]}\). The study of how such maxima are altered through horizontal openings is at the basis of the algorithm introduced in this section. Denote by \(I \circ B\) the opening of image \(I\) by a structuring element \(B\). The following proposition holds:

**Proposition 17** Let \(M = \{p_0, p_1, \ldots, p_n\}\) be a horizontal maximum of \(I\). Let \(p \in M\) be a pixel of this maximum.

\[
\forall k < n, \quad (I \circ L_k)(p) = I(p) \quad \text{(28)}
\]

for \(k = n\),
\[
(I \circ L_n)(p) = \max\{I(N_l(p_0)), \quad I(N_r(p_{n-1}))\} < I(p) \quad \text{(29)}
\]

That is, any opening of \(I\) by a line segment \(L_k\) such that \(k < n\) leaves this maximum unchanged, whereas for any \(k \geq n\), all the pixels of \(M\) have a lower value in \(I \circ L_k\) than in \(I\). Furthermore, we can quantify the effect of an opening of size \(n\) on the pixels of this maximum: the value of each pixel \(p \in M\) is decreased from \(I(p)\) to \(\max\{I(N_l(p_0)), I(N_r(p_{n-1}))\}\). In granulometric terms, the contribution of maximum \(M\) to the \(n\)-th bin of the horizontal pattern spectrum \(PS_h(I)\) is:

\[
n \times [I(p) \Leftrightarrow \max\{I(N_l(p_0)), I(N_r(p_{n-1}))\}] \quad \text{(30)}
\]

This is illustrated by Fig. 11.

Additionally, the local effect of a horizontal opening of size \(n\) on \(M\) is that a new “plateau” of pixels is created at altitude \(\max\{I(N_l(p_0)), I(N_r(p_{n-1}))\}\). This plateau \(P\) contains \(M\), and may be itself a maximum of \(I \circ L_n\). If it is, we say that \(P\) is part of the maximal region \(R(M)\) surrounding maximum \(M\), and we can now compute the contribution of \(P\) to the \(l(P)\)-th bin of the pattern spectrum, etc. Following this principle, the proposed granulometry algorithm works as follows on each image line:

**Algorithm:** horizontal granul. of an image line

- for each maximum \(M\) of this line (in any order) do:

  \[
  \Leftrightarrow \text{Add contribution of this maximum to } I(M)\text{-th bin of pattern spectrum;}
  \]

  \[
  \Leftrightarrow \text{Let } P\text{ be the plateau of pixels formed by opening of size } l(M)\text{ of } M; \text{ if } P\text{ is itself a maximum of } I \circ L(l(M)), \text{ computes its contribution to bin } l(P)\text{ of pattern spectrum;}
  \]

  \[
  \Leftrightarrow \text{Iterate previous step until the new plateau formed is no longer a maximum;}
  \]

  \[
  \Leftrightarrow \text{‘Mark’ the maximal region around } M\text{ as already processed;}
  \]

Note that this algorithm is inherently recursive: once a maximal region \(R\) has been processed, all of its pixels are regarded as having the gray-level they were given by the last opening considered for \(R\). In practice though, there is no need to physically modify the values of all the pixels in

\(^4\)The algorithm described in this section applies to 1-D signals as well.
keeping track of the first and last pixels of \( R \) is sufficient, allowing the algorithm to efficiently skip over already processed maximal regions (see Fig. 15). Thanks to this trick, the algorithm only considers each image pixel twice in the worst case.

This algorithm was compared to the traditional opening-based technique. For the latter, a fast opening algorithm was used, whose speed is proportional to the number of pixels in the image and the length of the line segment used as structuring element. As illustrated by Table 3, the new algorithm described in this section is three orders of magnitude faster.

### 2.3. Granulometries with Maxima of Linear Openings

In order to deal with more complicated cases, we now introduce a technique that can be seen as a generalization of the concept of opening functions for the grayscale case. When performing openings of increasing size of a binary image, each “on” pixel \( p \) is turned “off” for an opening size given by the value of the opening function at pixel \( p \). In other words, the opening function encodes for each pixel the successive values it takes for increasing opening sizes (namely, a series of 1’s followed by a series of 0’s). Similarly, in the grayscale case, as the size of the opening increases, the value of each pixel decreases monotonically. If each pixel was assigned the list of values it takes for every opening size, then the corresponding grayscale granulometry could be extracted straightforwardly.

Unfortunately, even if it were possible to compute such lists of values quickly, assigning one to each image pixel would require far too much memory. A more compact representation needs to be designed, that takes into account the intrinsic “redundancy” of opened images, characterized by their large plateaus of pixels. If we again consider the linear case, an elegant solution can be proposed to both the problem of computing these lists of values, and the problem of storing them compactly:

Let \( M \) be a horizontal maximum of image \( I \), with altitude (grayscale) \( h \). We pointed out in the previous section that a horizontal opening of size \( l(M) \) of \( M \) takes all of its pixels down to a new value \( h' \). Beyond this, the following proposition can easily be proved:

**Proposition 18** Let \( n > 0 \), \( I \) a grayscale image such that \( I = I \circ L_{n-1} \). Then, for every pixel \( p \) in \( I \):

\[
(I \circ L_n)(p) < I(p) \iff \exists M \text{ horiz maximum, } l(M) = n \text{ and } p \in (M)
\]

Therefore, at opening \( n \) in the sequence, the only pixels affected are those which belong to maxima of length \( n \). Furthermore, all the pixels belonging to the same maximum \( M \), \( l(M) = n \), will be affected in the same way for any opening of size greater than or equal to \( n \). The list of decreasing values we wish to associate with each pixel \( p \) in \( M \) can therefore “converge” into one single list for size \( n \). For larger opening sizes, this list may itself be merged with other lists, etc.

Based on this principle, the algorithm introduced here represents each image line as a tree \( T \), which we call its **opening tree**. The leaves of \( T \) are the image pixels, and the nodes are made of pairs \((h,n)\), where \( h \) is a grayscale value and \( n \) is an opening size. Every pixel that can be reached by going upwards in the tree starting from node \((h,n)\) is such that its value for the opening of size \( n \) is \( h \). Conversely, starting from a pixel \( p \), successive pairs \((h_1,n_1),(h_2,v_2),\ldots,(h_i,v_i),\ldots\), are reached by going down towards the root of the tree. By convention, for this pixel, \((h_0,v_0) = (I(p),0)\). These pairs satisfy:

\[
\forall i > 0, \quad h_i > h_{i+1} \quad \text{and} \quad n_i < n_{i+1}.
\]

For \( n \geq 0 \), the value of the opening of size \( n \) of \( I \) at pixel \( p \) is given by:

\[
(I \circ L_n)(p) = h_j,
\]

where \( j \) is such that \( n_j \leq n < n_{j+1} \).

Openings of trees can be computed using an algorithm very similar to the one described in the previous section. An example of opening tree is shown in Fig. 16.

Opening trees provide a hierarchical description that can be used to compactly represent all the horizontal openings of a grayscale image. In this respect, this notion is a grayscale equivalent
of the opening function mentioned earlier (see Fig. 1). One can prove that, in the worst case, the opening tree has one node per image pixel. In practice though, only between 0.3 and 0.9 nodes per pixel are needed depending on the complexity of the image processed.

Any horizontal opening of $I$ can be straightforwardly derived from its opening tree. In addition, the horizontal pattern spectrum of $I$, $PS_h(I)$, can be computed from $T$ as follows:

**Algorithm:** horiz granul of $I$ from its opening tree
- initialize each bin of pattern spectrum $PS_h(I)$ to 0;
- for each pixel $p$ of $I$ do:
  - $v \leftarrow I(p)$; $(h, n) \leftarrow$ node pointed at by $p$;
  - while $(h, n)$ exists, do:
    - $PS_h(I)(n) \leftarrow PS_h(I)(n) + (v \leftarrow h)$;
    - $v \leftarrow h$; $(h, n) \leftarrow$ next node down in tree;

This algorithm is obviously less efficient for horizontal granulometries than the one introduced in the previous section. However, it easily generalizes to the computation of granulometries using maxima of linear openings in several orientations. For example, to determine the granulometric curve corresponding to maxima of horizontal and vertical openings, one first extracts the horizontal opening tree $T_1$ and the vertical opening tree $T_2$. Then, for each pixel $p$, the technique consists in descending the corresponding branches of $T_1$ and $T_2$ simultaneously as follows:

**Algorithm:** granul of $I$ from trees $T_1$ and $T_2$
- initialize each bin of pattern spectrum $PS(I)$ to 0;
• for each pixel $p$ of $I$ do:
  $\therefore v \leftarrow I(p)$;
  $\therefore (h_1, n_1) \leftarrow \text{node of } T_1 \text{ pointed at by } p$;
  $\therefore (h_2, n_2) \leftarrow \text{node of } T_2 \text{ pointed at by } p$;
  $\therefore$ while $(h_1, n_1)$ and $(h_2, n_2)$ exist, do:
    $\quad$ size $\leftarrow \max(n_1, n_2)$;
    $\quad$ while $n_1 \leq \text{size}$ and $(h_1, n_1)$ exists do:
      $\quad\quad$ $(h_1, n_1) \leftarrow \text{next node down in } T_1$;
    $\quad$ while $n_2 \leq \text{size}$ and $(h_2, n_2)$ exists do:
      $\quad\quad$ $(h_2, n_2) \leftarrow \text{next node down in } T_2$;
    $\quad$ PS($f$(size)) $\leftarrow$
    $\quad$ PS($f$(size)) $\leftarrow (v \leftarrow \max(h_1, h_2))$;
    $\quad$ v $\leftarrow \max(h_1, h_2)$;

The same technique extends to any number of opening trees. The whole granulometry algorithm (extraction of trees followed by computation of the pattern spectrum from these trees) is once again orders of magnitude faster than traditional techniques, as illustrated by table 3. In this table, a granulometry by maxima of linear openings at 4 orientations was computed. In terms of memory, the computation of this particular granulometry requires, in the worst case, 1 pointer (4 bytes) and 1 node (8 bytes) per pixel, for each orientation. This comes to a total of 48 bytes/pixel, i.e., a worst case scenario of 12 Megabytes for a $512 \times 512$ image. This is a reasonable tradeoff given the speed of the algorithm, and is not a strain on modern systems.

The algorithm given above can be easily adjusted to the computation of pseudo-granulometries by minima of linear openings (for more details, see [135]). Although minima of openings are not themselves openings [66], minima of openings with line segments of increasing length constitute a decreasing family of image operators. The resulting pseudo-granulometric curves often characterize the same image features as square granulometries.

These pseudo square granulometries have been very successful as one of the feature sets used to characterize plankton in towed video microscopy images [29]. As an illustration, Fig. 17 shows four different imagettes of copepod oithona and their corresponding pattern spectra, for a granulometry using openings with squares. These curves are all relatively flat, exhibiting a single well-marked maximum, between size 5 and 10, which corresponds to the size of the body of the organisms. By contrast, the same granulometric curves for the pteropods shown in Fig. 18 are entirely different, climbing sharply until a strong maximum, reached for a size between 10 and 15. Together with other structural and shape-based features, these pattern spectra are used as input to a neural network classifier. The goal of the project is to be able to classify between 5 and 10 different kinds of organisms with over 90% accuracy.

2.4. Conclusion

In this section, a novel class of algorithms was introduced for efficiently computing granulometries in grayscale images. These algorithms turn out to be several orders of magnitude faster than any previously available technique. One of their key underlying concepts is that of opening tree: such structures are shown to provide a compact representation for the successive openings of a grayscale image by line segments of increasing size. They are at the heart of the algorithm proposed for grayscale granulometries with maxima of linear openings. In addition, opening trees provide a way to extract pseudo-granulometries by minima of openings with line segments at different orientations, which can be used to approximate square granulometries. These algorithms can even be extended to the fast computation of granulometries by area openings and closings [130], as described in [135]. However, the efficient computation of exact grayscale granulometries with square openings remains an open problem (no pun intended).

Over the past few years, grayscale granulometries have proved to be useful in a variety of image analysis tasks, including texture classification [25] and picture segmentation [31]. However, these insights...
Figure 17. Pattern spectra of copepod oithona
Figure 18. Pattern spectra of pteropods
Table 3
Execution time of traditional opening-based techniques and of introduced algorithms for the computation of a horizontal grayscale granulometry (left) and a granulometry by maxima of linear openings in 4 orientations (right). A complex 512 × 512 image was used for this comparison, done on a Sun Sparc Station 10. Granulometries were computed for opening sizes 1 to 512.

<table>
<thead>
<tr>
<th></th>
<th>traditional</th>
<th>new</th>
<th>improvement factor</th>
<th>traditional</th>
<th>new</th>
<th>improvement factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontal</td>
<td>204s</td>
<td>0.206s</td>
<td>990</td>
<td>824s</td>
<td>2.78s</td>
<td>296</td>
</tr>
</tbody>
</table>

Applications have often been handicapped by slow processing speeds. The efficiency of the algorithms introduced in the present part not only makes it now possible to use granulometries on a routine basis, they also open new areas of application for these tools. For example, in [134], it was briefly shown how linear granulometries can be used to directly extract global estimates of object size in grayscale images without the need for any segmentation or binarization. Extracting such information is sometimes a goal in itself, but it can also be essential to calibrate the parameters of subsequent image analysis algorithms to be applied, thereby greatly enhancing their robustness. Undoubtedly, the family of algorithms introduced here will contribute to popularize the use of granulometries for a growing number of image and signal analysis problems.
Part II
Watersheds and Segmentation

This chapter is derived from “Morphological Segmentation for Textures and Particles”, in Digital Image Processing Methods, E. Dougherty Ed., Marcel-Dekker, New York, 1994 (see [137]).

3. Introduction, Morphological Segmentation

The present chapter concerns image segmentation via the methods of morphological image processing. A generally accepted meaning of the word segmentation in the image processing community is the decomposition of the image under study into its different areas of interest. Here, we shall take the perspective that there are essentially two kinds of segmentation: segmentation of images of texture and segmentation of images of particles.

In texture segmentation, an image is partitioned into regions, each of which is defined by a set of features characteristic to the microimage structure within it, this structure typically being viewed in terms of the small texture primitives composing it. Typical applications include segmentation of vegetation types in aerial photographs, segmentation of text and halftones in document pages, medical imaging applications, etc.

The second kind of segmentation is concerned with images of particles (or objects), where textural information is either not present, or cannot be simply used as a discriminating factor. The segmentation task consists of extracting the particles from the image(s) under study. In other words, the goal is to partition the image as many connected components as there are objects or regions to extract, plus some background regions. We distinguish between binary and grayscale particle segmentation: in the binary case, i.e. when the images under study are binary, the segmentation task consists in separating the overlapping particles (e.g., see the coffee beans example of Fig. 19a). In the grayscale case, the segmentation task is equivalent to a contour extraction problem (e.g., in Fig. 19b, the contours of the electrophoresis spots have to be extracted as precisely as possible).

The chapter is primarily concerned with the latter case: particle segmentation. We focus on watershed segmentation, which has the advantages of being very general, usually accurate and fast, and applicable to both binary and grayscale images. A fair amount of morphological machinery is involved: subject to the space constraint of this chapter, we shall introduce the necessary machinery, leaving detailed theoretical descriptions to the literature [100, 102, 103]. Refer to [32] for an introductory account of the fundamentals of morphological image processing. Good introductions to the fields of mathematical morphology can also be found in [27, 98] (note: these references are in French).

4. Binary and Grayscale Particle Segmentation

4.1. Introduction to Morphological Segmentation

As previously mentioned, the purpose of this section is to show how morphology can be successfully applied to a wide range of object segmentation problems. In this section, segmentation of an image refers to the task consisting in extracting from it the objects or particles of interest as precisely as possible. By region-contour duality, this is equivalent to dividing the image in object regions and background regions.

Segmenting an image I is more than simply counting its objects and pointing at them in the image: it also encompasses the extraction of the objects’ contours. However, as we shall repeat throughout this section, the contour extraction step usually requires the prior marking of the objects to extract. By a marker of an object or set X, we simply mean a set M that is included in X. We also generally assume that markers have the same homotopy as the set they are marking. For example, a marker of a connected set is usually
connected itself, although this constraint can in fact be loosened in many practical cases. Moreover, markers are usually located towards the central part of the objects they mark.

This marker extraction step is shown to be the most important step in many complex segmentation applications that are undertaken using morphology. Intuitively, this marking is an algorithmic simulation of human behavior: when asked to show the objects present in an image, a human does not start right away by precisely outlining these objects. On the contrary, he or she will first point at the objects, one after the other, thus marking them. This marking can be seen as a first estimate of the objects; it is then refined by outlining the object’s contours. From an image analysis point of view, it is generally admitted that finding object markers is a less difficult task than directly extracting accurate contours.

In this section, we show that going from markers to actual segmentation can be done automatically using procedures that we will describe, and that are mainly based on the watershed transformation.

In the binary case, segmentation refers to the extraction of the connected components representing objects of interest, as well as the separation of the overlapping objects. For example, the binary image shown in Fig. 19a represents coffee beans. This image will be used throughout this section to illustrate the effect of the morphological transformations we will be dealing with. We can count the beans present in the image by simply extracting a connected marker for each bean and counting the number of markers. However, this is not sufficient if one wishes to perform measurements on each individual bean (area, perimeter, elongation, etc.). For this purpose, the beans need to be segmented, i.e. separated from one another. In the following, binary segmentation means “separation of overlapping objects in a binary image”.

This marker-based methodology will also provide a framework for grayscale segmentation. We will illustrate our approach on the ultraclassic image of 2-D electrophoresis gels shown in Fig. 19b. On this image, we not only want to find all the dark spots, but also correctly outline them in order to measure, e.g., their respective areas, the density of black under them, etc. In both the binary and the grayscale segmentation cases, the watershed transformation (see section 4.4) will be used to extract precise object outlines from markers. We will talk about *marker-driven watershed segmentation*.

Note that in many simpler cases, there is no need to apply this approach. For example, the top-hat transformation originally proposed in [71] provides an excellent tool for extracting light (resp. dark) objects from an uneven background. Its relies on the fact that by grayscale opening, one removes from an image the light areas that cannot hold the structuring element. Subtracting the opened image from the original one yields an image where the objects that have been removed by opening clearly stand out, and that image can then easily be thresholded (see Fig. 20).

Archetypically, the structuring element used in the opening step is a disk or a discrete approximation of a disc. However, infinite variations of this transformation can be derived. We may cite:

- using linear elements, one can specifically extract objects that are elongated in one direction,

- using a closing instead of an opening and subtracting the original image from the closed one allows us to extract dark objects on a lighter background. In this case, we talk about “black top-hat” as opposed to the “white top-hat” (by opening).

- If the background contains a lot of thin and elongated objects, one can use maxima of openings (resp. minima of closings) with linear elements, followed or not by grayscale reconstruction (see section 4.3.4), or area openings (resp. closings) [130].

An application is shown in Fig. 21: Fig. 21a is a scanning electron microscopy image where the balls in the lower right corner are to be extracted. These being compact and light compared to the background around them, they are removed by an opening of size 2 (see Fig. 21b). After subtraction of (b) from (a), i.e. top-hat (see Fig. 21c), these
Figure 19. (a) binary image of coffee beans that need to be separated; (b) grayscale image of 2-D electrophoresis gel whose spots have to be extracted.

Figure 20. Peaks extracted via top-hat followed by thresholding.
small balls stand out and this image can be easily thresholded into Fig. 21d. The desired balls (right side) can now be extracted as the balls contained in the largest connected component of the dilation of Fig. 21d. The dilated image is shown in Fig. 21e, and the resulting segmentation is shown in Fig. 21f.

In the following, we concentrate on more complex segmentation problems, where top-hats cannot provide satisfactory solutions. We first deal with binary segmentation, and the tools and techniques we are lead to use turn out to be even more useful later, for grayscale segmentation. In section 4.2, we present the concepts of maximal balls and skeletons and show how they can be used to robustly mark the centroid of overlapping objects in binary images. To go further and, from these markers, derive the desired binary segmentation, we need to make use of geodesic operators that are defined in section 4.3. The grayscale version of these operators is also very useful for grayscale segmentation. Lastly, in section 4.4, we describe how it unifies both binary and grayscale segmentations. Its use is illustrated on the segmentation of images of 2-D electrophoresis gels.

Some reminders and notations

From this point on, we are only concerned with the discrete case, i.e., our workspace is the discrete plane \( \mathbb{Z}^2 \). In this plane, a grid \( G \) provides the neighborhood relationships between pixels. Commonly used grids are the square grid, for which a pixel \( p \) has either 4 (in 4-connectivity) or 8 neighbors (in 8-connectivity), as well as the hexagonal grid (6-connectivity). Two neighboring pixels \( p \) and \( q \) form an edge of \( G \). The grid \( G \) induces a discrete distance in \( \mathbb{Z}^2 \), the distance between two pixels being the minimal number of edges required to join them.

Discrete images are considered as mappings from \( \mathbb{Z}^2 \) onto \( \mathbb{Z} \); grayscale images take their values in a range \( \{0, 1, \ldots, N\} \) whereas binary images can only take value 0 and 1. The information content of a binary image is contained in its pixels with value 1, and therefore, binary images are often regarded as sets. For this reason, the binary transformations described in the following are often defined as set transformations.

4.2. Maximal Balls, Skeletons, Ultimate Erosions

We have seen earlier that granulometric analyses allow one to extract size information about an image without the need to segment it. For example, consider the coffee bean image \( X \) shown in Fig. 19. Depending on the context, \( X \) shall either refer to the image itself or to the set of its black pixels. The granulometric analysis of this image may be undertaken using squares of increasing sizes, denoted \( S_1, S_2, \) etc: openings with the \( S_i \)'s are thus performed for \( i = 1 \) to the first value \( n \) such that \( X \subseteq S_n = \emptyset \). At each step \( i \), the area of image \( X \ominus S_i \) is determined.

The \( (X \ominus S_i)_{0 \leq i \leq n} \) constitutes a decreasing sequence of sets, so that as shown in [136], one can synthesize all the information contained in this sequence via a single function called the granulometry function.

Definition 19 The granulometry function \( g_X \) associated with a set \( X \) and the family of convex and homothetic elements \( (S_i)_{i \geq 0} \) maps each pixel of \( X \) to the first \( i \) such that \( x \notin X \ominus S_i \):

\[
g_X : p \in X \iff \min\{i \in \mathbb{N}, p \notin X \ominus S_i\}.
\] (34)

Alternatively, one can say that the granulometry function maps each pixel of \( X \) to the maximal \( i \) such that there exists a translation \( t(S_i) \) of \( S_i \) satisfying \( p \in t(S_i) \subseteq X \). The granulometry function of our coffee beans image is shown in Fig. 22.

From these images, it becomes clear that our coffee beans can be described as the areas where the largest balls (in this particular case: squares) can be included. Therefore, to extract markers of our beans, we shall start by looking at the image zones where the “largest” balls can be extracted.

4.2.1. Definitions

In this section and the following, the notion of ball directly stems from the distance being used. For example, in the plane \( \mathbb{R}^2 \) equipped with the usual Euclidean distance, the balls are standard discs. In the discrete plane \( \mathbb{Z}^2 \), the balls are hexagonal if the hexagonal grid is used (6-connectivity) or squares in square grids. The unit size ball \( B \) (ball of radius 1) corresponds to
Figure 2.1. Top-hat segmentation of SEM image.
Figure 22. (a) Granulometry function of Fig. 19a with respect to a family of squares; (b) level lines of this function.
either $S_1$, $H$ or $S_2$ depending on whether 4-, 6- or 8-connectivity is used (see Fig. 23).

Calabi’s definition of the skeleton is based on the following notion of maximal ball:

**Definition 20** A ball $B$ included in $X$ is said to be maximal if and only if there exits no other ball included in $X$ and containing $B$:

$$\forall B' \text{ ball}, \quad B \subseteq B' \subseteq X \implies B' = B. \quad (35)$$

This concept is illustrated by Fig. 24, and the definition of the skeleton follows from it:

**Definition 21** (Skeleton by maximal balls)
The skeleton $S(X)$ of a set $X \subseteq \mathbb{Z}^2$ is the set of the centers of its maximal balls:

$$S(X) = \{ p \in X \mid \exists r \geq 0, \quad B(p,r) \text{ is a maximal ball of } X \}. \quad (36)$$

The skeleton is an intuitive notion: the skeleton of a ball is reduced to its center, that of a band yields a unit thickness line, etc. Examples of skeletons of simple shapes are shown in Fig. 25. One can see why the skeleton is often called the *medial axis transform*: it provides a description of sets in terms of lines of unit pixel thickness.

Unfortunately, things are not as easy as they look: in the continuous case (Euclidean plane $\mathbb{R}^2$), e.g., the skeleton of two tangent discs is reduced to the two centers of these discs (see Fig. 26) instead of being a straight line joining these two points. In other words, the homotopy or connectivity of the original set is not necessarily preserved (see [102, chapters 11–12] for more details).

In the discrete case, let us denote by $nB$ the ball of radius $n$ in the considered connectivity:

$$nB = B \oplus B \oplus \cdots \oplus B,$$

with $B$ being equal to either $S_1$, $H$, or $S_2$. Lantuéjoul proved that the skeleton by maximal balls can be obtained by the following formula:

$$S(X) = \bigcup_{n=0}^{+\infty} [(X \ominus nB)/(X \ominus nB) \circ B]. \quad (37)$$

In words, the skeleton by maximal balls can be obtained as the union of the residues of openings of $X$ at all scales. Unfortunately, once again the skeleton does not behave as one would hope: the direct application of formula (37) yields completely disconnected skeletons, as illustrated by Fig. 27.

Extracting correct homotopic (connected) skeletons from discrete binary images is thus not a straightforward matter. The literature on skeletons is very abundant and we certainly do not intend to cover the extraction of connected skeletons in this chapter. Let us just mention that the method recently proposed in [124] allows the very efficient computation of a connected skeleton which is a minimal superset of the skeleton by maximal balls. An example of a connected skeleton computed in 8-connectivity using this method is shown in Fig. 28a. Since the skeleton by maximal balls is not necessarily of unit-pixel thickness, the connected skeleton of Fig. 28a is not either. For some applications, it can be of interest to use thinning techniques to reduce it to a single pixel thickness, as shown in Fig. 28b.

### 4.2.2. Quench Function

Let us go back to our description of binary sets in terms of maximal balls and see what more can be said about these descriptions. By definition, to every pixel $p$ in the skeleton, there corresponds a maximal ball. Let us denote by $q_\infty(p)$ the radius of this ball. We thus define the *quench function*:

**Definition 22** (Quench function)
The quench function associates with every pixel $p \in S(X)$ the radius of the corresponding maximal ball.

One of the most important results about the quench function is that its datum is sufficient to reconstruct the original set completely:

**Theorem 23** A set $X$ is equal to the union of its maximal balls:

$$X = \bigcup_{p \in S(X)} (p + q_\infty(p)B). \quad (38)$$

The quench function thus allows us to do lossless encoding of binary images, and it has been extensively studied for image compression. Derived versions of this concept equal in performance the famous *CCITT group 4* encoding scheme on some kinds of document images [21].
Figure 23. Unit size ball $B$ in 4-, 6- and 8-connectivity respectively.

Figure 24. Concept of maximal ball in the Euclidean plane.

Figure 25. Skeleton of simple shapes in the Euclidean plane.

Figure 26. In the continuous Euclidean plane, the skeleton of a connected set is not necessarily connected.
Figure 27. Example of skeleton by maximal balls on the coffee beans image, using the balls of the 8-connected distance.

Figure 28. (a) 8-connected skeleton; (b) thinned 8-connected skeleton.
The quench function of our image of coffee beans is presented Fig. 29a. Obviously, since this mapping is defined on the skeleton by maximal balls $S(X)$, its support is not connected. However, using the skeletonization technique briefly mentioned above [124], it is possible to re-connect the skeleton by maximal balls and to extract the radii corresponding to pixels on the connecting arcs. This produces the connected quench function shown in Fig. 29b.

The other major interest of the quench function is the definition of the ultimate erosion. We have now a way to describe a set $X$ as the union of its maximal balls; in order to define markers of our coffee beans, i.e., of the convex blobs of the binary image under study, we look for the largest among these maximal balls. Clearly, for a given connected component $C$ of $X$, (one of) the largest maximal ball is (one of) the largest ball that can be included in $C$, and its center marks an important object. However, if $C$ is made of two overlapping objects, this crude method only allows us to mark one of them.

Let us consider the simple case where $X$ is equal to the union of two overlapping discs. As shown on Fig. 30, the skeleton of $X$ is the line segment joining the centers of these two discs. Now, upon examination of the quench function, one can notice that it exhibits two maxima, located at the exact centers $a$ and $b$ of our discs$^6$. These maxima therefore define markers of our overlapping objects, and their set constitutes the ultimate erosion of $X$:

**Definition 24 (Ultimate erosion)** The ultimate erosion of a set $X$, denoted $	ext{Ult}(X)$, is the set of the (regional) maxima of the quench function $q_X$ of $X$.

At this point, we need to recall the definition of maxima (also called regional maxima) for grayscale images:

**Definition 25 (Regional maximum)** A regional maximum $M$ of a grayscale image $I$ is a connected component of pixels with a given value $h$ (plateau at altitude $h$), such that every pixel in the neighborhood of $M$ has a value strictly lower than $h$.

One should make a clear distinction between regional maxima and local maxima. A local maximum is defined as follows:

**Definition 26 (Local maximum)** A pixel $p$ of a grayscale image $I$ is a local maximum if and only if for every pixel $q$ which is neighbor of $p$, $I(p) \geq I(q)$.

Obviously, if $M$ is a regional maximum of $I$, then $p \in M \implies p$ is a local maximum, but the converse is does not hold, as we shall point out shortly. Similarly, one can define local and regional minima.

The problem with definition 24 is that the discrete quench function is defined on a non-connected support, so that its maxima are not really defined! To get around this problem, we can compute the maxima of the connected quench function presented earlier (see Fig. 29b). The thus extracted ultimate erosion of the coffee-beans image is shown in Fig. 31a.

We can see that this ultimate erosion provides a reasonably good marking of our beans. We are therefore close to reaching our first goal, namely the extraction of one (connected) marker per object. There remain two points to address:

- the marking is still not perfect: some beans are multiply marked. How to reconnect some markers in order to end up with one single marker per bean?
- How to extract ultimate erosions in a more straightforward manner?

These two issues are addressed in the next section with the introduction of the distance function.

### 4.2.3. Ultimate Erosion and Distance Function

In this section, we give a completely different interpretation of the ultimate erosion. Let $X$ be a set made of overlapping components. When performing iterative erosions of this set with respect to the unit size ball $B$, its components are progressively shrunk and separated from the rest of
Figure 29. Quench function (a) and connected quench function (b). Their supports have been dilated for clarity.

Figure 30. Skeleton of a set $X$, associated quench function $q_X$ and its maxima.
the set before they are completely removed by the erosion process. This is illustrated by Fig. 32.

If, throughout this erosion process, we keep aside each connected component just before it is removed, it can be proved that the thus obtained set is exactly the ultimate erosion of $X$. Given two sets $A$ and $B$ such that $B \subseteq A$, let us denote by $\rho_A(B)$ the union of the connected components of $A$ that have a non-empty intersection with $B$. This operator is called reconstruction and is more fully discussed in section 4.3. Our new ultimate erosion algorithm can now be expressed by the following formula:

$$ \text{Ult}(X) = \bigcup_{n \in \mathbb{N}} [(X \ominus nB) \setminus \rho_{X \ominus nB}(X \ominus (n+1)B)] $$

The resulting ultimate erosion of the set of Fig. 32 is shown in Fig. 33.

This method suggests yet another way to determine the ultimate erosion of a $X$. Indeed, there is a morphological transformation which synthesizes all the information contained in the successive erosions of a set $X$. This transformation is called the distance function and associates with each pixel $p$ of $X$ the size of the first erosion of $X$ that does not contain $p$.

**Definition 27 (Distance function)** The distance function $\text{dist}_X$ associated with a set $X$ is given by:

$$ \forall p \in X, \quad \text{dist}_X(p) = \min \{n \in \mathbb{N} \mid p \not\in X \ominus nB\}. $$

For each pixel $p \in X$, $\text{dist}_X(p)$ is the distance between $p$ and the background, i.e., $X^C$. The 8-connected distance function of the coffee beans image is shown in Fig. 34.

By definition, the regional maxima at altitude $h$ of the distance function are the connected components at altitude $h$ of $\text{dist}_X$ such that every neighboring pixel is of altitude strictly smaller than $h$. Any of these regional maxima are removed by a unit-size erosion. Indeed, if this was not true, there would exist pixels $q$ located inside the regional maximum and verifying $\text{dist}_X(q) > h$, which is absurd. These maxima thus belong to the ultimate erosion of $X$ and the following proposition can be derived:

**Proposition 28** The ultimate erosion of a set $X$
Figure 32. Successive erosions of a set. Each component is separated from the rest of the set before it is removed by the erosion process.
Figure 33. Ultimate erosion of a set.

Figure 34. 8-connected distance function of the coffee beans image (a) and level lines of this function (b).
is equal to the union of the regional maxima of the distance function of $X$.

Since distance functions and regional maxima can be computed very efficiently in discrete images (see e.g. [129]), this last proposition provides the best computational method for extracting ultimate erosions. As a matter of curiosity, what happens if we now extract the local maxima of the distance function? The following proposition holds:

**Proposition 29** The skeleton by maximal balls of a set $X$ is equal to the set of local maxima of its distance function.

The distance function is therefore at the basis of a very large number of morphological algorithms. Later in this chapter, it will be used conjunction with the watershed algorithm. For completeness of this section, we shall also mention that after local and regional maxima, the crest-lines of the distance function are of interest: following them allows us to extract the connected skeleton of $X$ (see Fig. 35). This forms the basis of a series of algorithms proposed by F. Meyer [102, chapter 13].

We have now defined the morphological tools that will allow us to obtain perfect bean markers. We saw on Fig. 31 that the ultimate erosion does not quite yield perfect markers of our beans: all beans are marked, but some have multiple markers. The disconnections are caused by our discrete workspace as well as small contour irregularities of the beans. For this particular image, a unit size dilation would be good enough to re-connect the markers and end up with one single marker per bean. However, this might not work in the general case: components of the ultimate erosion marking the same coffee bean may indeed be separated by arbitrarily large distances.

The method used instead relies upon the fact that two components of the ultimate erosion marking the same bean are pretty much on the same “maximal zone” of the distance function. In fact, it is possible to go from one to the other on the distance function by going down no more than one level. Thus, if we subtract 1 from the distance function at the location of all the components of the ultimate erosion, we obtain a modified distance function whose maxima are exactly the desired bean markers, as illustrated in Fig. 36.

### 4.2.4. Skeleton by Influence Zones

The last concept that we need to define in this section is that of skeleton by influence zones, also called SKIZ.

**Definition 30** Let $X$ be a set made of $n$ connected components $(X_i)_{1 \leq i \leq n}$. The influence zone $Z(X_i)$ of $X_i$ is the locus of the points which are closer to it than to any other connected component of $X$:

$$Z(X_i) = \{ p \in \mathbb{Z}^2 \mid \forall j \neq i, d(p, X_i) \leq d(p, X_j) \}.$$  \hfill (39)

The distance $d$ used in this equation is the discrete distance induced by the grid we are using (4-, 6-, or 8-connectivity).

The SKIZ is then defined as follows:

**Definition 31 (SKIZ)** The SKIZ of set $X$, denoted $SKIZ(X)$, is the set of the boundaries of the influence zones $\{Z(X_i)\}_{1 \leq i \leq n}$.

An example of skeleton by influence zones is shown on Fig. 37a. Just as the skeleton follows the crest-lines of the distance function, one can view the SKIZ as following the valley-lines of the inverted distance function of the background (Fig. 37b).

### 4.3. Geodesic Transformations

#### 4.3.1. Introduction, Geodesic Distances

At this point, we have achieved the first step of our segmentation as outlined in section 4.1: the marker extraction. It now remains to make good use of these markers for the extraction of correct set boundaries. The idea is to define each bean as the image region centered around its marker. Our intent is therefore to “grow” these markers back in the mask of the coffee-beans image. For this purpose, we now need the notion of geodesic operators introduced by C. Lantuéjoul [55, 56].

Contrary to classic “Euclidean” morphological operations, geodesic ones do not operate on the entire space, but on a finite set $X$ which is called the *mask*. They are based on the notion of geodesic distance:
Figure 35. The skeleton follows the crest-lines of the distance function.

Figure 36. Final bean markers. For this image as well as for Fig. 31, the exact Euclidean distance function was used.
Figure 37. SKIZ of a binary image; (b) the SKIZ follows the valley lines of the inverted distance function of the background. This example was computed using Euclidean distance.
**Definition 32** The geodesic distance between two points $x$ and $y$ of $X$ is the infimum of the length of the paths between $x$ and $y$ in $X$, if such paths exist:

$$d_X(x,y) = \inf \{ l(C_{x,y}) | C_{x,y} \text{ is a path between } x \text{ and } y \text{ inside } X \}$$

If there are no such paths, we set $d_X(x,y) = +\infty$.

This definition is illustrated by Fig. 38.

We call the geodesic ball of radius $n$ and of center $p \in X$ the set $B_X(p,n)$ defined by:

$$B_X(p,n) = \{ p' \in X, d_X(p',p) \leq n \}.$$  \hfill (41)

**4.3.2. Geodesic Dilations and Erosions**

Suppose now that $X$ is equipped with its associated geodesic distance $d_X$. Given $n \geq 0$, we consider the structuring function $|02|$ mapping each pixel $p \in X$ to the geodesic ball $B_X(p,n)$ of radius $n$ centered at $p$. This leads to the definition of the geodesic dilation of a subset $Y$ of $X$:

**Definition 33** The geodesic dilation $\delta_X^{(n)}(Y)$ of size $n$ of set $Y$ inside set $X$ is given by

$$\delta_X^{(n)}(Y) = \bigcup_{p \in Y} B_X(p,n) = \{ p' \in X, \exists p \in Y, d_X(p',p) \leq n \}. \hfill (42)$$

The dual formulation of the geodesic erosion of size $n$ of $Y$ inside $X$ is the following:

$$\varepsilon_X^{(n)}(Y) = \{ p \in Y | B_X(p,n) \subseteq Y \} = \{ p \in Y \mid \forall p' \in X/Y, d_X(p,p') > n \}. \hfill (43)$$

Examples of geodesic dilation and erosion are shown in Fig. 39.

As already mentioned, the result of a geodesic operation on a set $Y \subseteq X$ is always included in $X$, which is our new workspace. As far as implementation is concerned, an elementary geodesic dilation (of size 1) of a set $Y$ inside $X$ is obtained by intersecting the result of a unit-size dilation of $Y$ (with respect to the unit ball $B$) with the workspace $X$:

$$\delta_X^{(1)}(Y) = (Y \oplus B) \cap X.$$  \hfill (44)

A geodesic dilation of size $n$ is obtained by iterating $n$ elementary geodesic dilations:

$$\delta_X^{(n)}(Y) = \delta_X^{(1)}(\delta_X^{(1)}(\ldots \delta_X^{(1)}(Y)\ldots)). \hfill (45)$$

One can derive similar equations for geodesic erosions.

**4.3.3. Reconstruction and Applications**

One can notice that by performing successive geodesic dilations of a set $Y$ inside a set $X$, it is impossible to intersect a connected component of $X$ which did not initially contain a connected component of $Y$. Moreover, in this successive geodesic dilation process, we progressively "reconstruct" the connected components of $X$ that were initially marked by $Y$. This is shown in Fig. 40.

Now, the sets with which we are concerned are finite ones. Therefore, there exists $n_0$ such that

$$\forall n > n_0, \delta_X^{(n)}(Y) = \delta_X^{(n_0)}(Y).$$

At step $n_0$, we have entirely reconstructed all the connected components of $X$ which were initially marked by $Y$. This operation is naturally called reconstruction:

**Definition 34 (Reconstruction)** The reconstruction $\rho_X(Y)$ of the (finite) set $X$ from set $Y \subseteq X$ is given by the following formula:

$$\rho_X(Y) = \lim_{n \to +\infty} \delta_X^{(n)}(Y). \hfill (46)$$

Fig. 41 illustrates this transformation.

Some applications require that the various markers remain unconnected (this is the case, for instance, of the binary segmentation problem with which we are concerned). In such cases, the geodesic influence zones of the connected components of set $Y$ inside $X$ are used. Indeed, the notions of influence zones and of SKIZ presented in section 4.2.4 easily extend to the geodesic case, as shown by Fig. 42.

**4.3.4. Grayscale Reconstruction**

At present, all the tools required for solving our bean segmentation problem have been defined. However for grayscale segmentation, we
Figure 38. Geodesic distance in a set $X$.

Figure 39. Examples of a geodesic dilation and of a geodesic erosion of set $Y$ inside set $X$. 
Figure 40. Successive geodesic dilations of set $Y$ inside set $X$.

\[
\begin{array}{cc}
\includegraphics[width=0.4\textwidth]{fig40a} & \includegraphics[width=0.4\textwidth]{fig40b}
\end{array}
\]

$\Rightarrow$

Figure 41. Reconstruction of $X$ (light set) from $Y$ (dark set).

\[
\begin{array}{cc}
\includegraphics[width=0.4\textwidth]{fig41a} & \includegraphics[width=0.4\textwidth]{fig41b}
\end{array}
\]

Figure 42. Example of geodesic SKIZ.
will also need to extend the concept of geodesy to grayscale images.

It has been known for several years that—at least in the discrete case—any increasing transformation defined for binary images can be extended to grayscale images [100, 141, 103]. By increasing, we mean a transformation \( \psi \) such that
\[
\forall X, Y \subseteq \mathbb{Z}^2, \quad Y \subseteq X \implies \psi(Y) \subseteq \psi(X). \tag{47}
\]

In order to extend such a transformation \( \psi \) to grayscale images \( I \) taking their values in \( \{0, 1, \ldots, N\} \), it suffices to consider the successive thresholds \( T_k(I) \) of \( I \), for \( k = 0 \) to \( N \):
\[
T_k(I) = \{ p \in D_I \mid I(p) \geq k \}, \tag{48}
\]
where \( D_I \) is the domain of image \( I \). They are said to constitute the threshold decomposition of \( I \) [64]. As illustrated by Fig. 43, these sets obviously satisfy the following inclusion relationship:
\[
\forall k \in [1, N], \quad T_k(I) \subseteq T_{k-1}(I). \tag{49}
\]

When applying the increasing operation \( \psi \) to each of these sets, their inclusion relationships are preserved. Thus, we can now extend \( \psi \) to grayscale images as follows:
\[
\forall p \in D_I, \quad \psi(I)(p) = \max\{k \in [0, N] \mid p \in \psi(T_k(I))\}. \tag{50}
\]

In the present case, binary geodesic reconstruction is an increasing transformation in that it satisfies:
\[
Y_1 \subseteq Y_2, \quad X_1 \subseteq X_2, \quad Y_1 \subseteq X_1, Y_2 \subseteq X_2 \implies \rho_{X_1}(Y_1) \subseteq \rho_{X_2}(Y_2). \tag{51}
\]

Therefore, following the threshold superposition principle of equation (49), we define grayscale reconstruction as follows [132]:

**Definition 35 (Grayscale reconstruction)**

Let \( J \) and \( I \) be two grayscale images defined on the same domain, taking their values in the discrete set \( \{0, 1, \ldots, N\} \) and such that \( J \leq I \) (i.e., for each pixel \( p \in D_I \), \( J(p) \leq I(p) \)). The grayscale reconstruction \( \rho_I(J) \) of \( I \) from \( J \) is given by:
\[
\forall p \in D_I, \quad \rho_I(J)(p) = \max\{k \in [0, N] \mid p \in \rho_{T_k(I)}(T_k(J))\}. \tag{52}
\]

Fig. 44 illustrates this transformation. Just like binary reconstruction extracts those connected components of the mask which are marked, grayscale reconstruction extracts the *peaks* of the mask which are marked by the marker-image.

By duality, we are also able to define the dual grayscale reconstruction, or reconstruction by erosion:

**Definition 36 (Dual grayscale reconstruction)**

Let \( J \) and \( I \) be two grayscale images defined on the same domain, taking their values in the discrete set \( \{0, 1, \ldots, N\} \) and such that \( J \geq I \) (i.e., for each pixel \( p \in D_I \), \( J(p) \geq I(p) \)). The grayscale reconstruction \( \rho_J(I) \) of \( I \) from \( J \) is given by:
\[
\forall p \in D_I, \quad \rho_J(I)(p) = N \Leftrightarrow \rho_{N-I}(I) \Leftrightarrow J. \tag{53}
\]

### 4.3.5. Binary Segmentation

Let us now use all these tools to design a powerful binary segmentation algorithm. Starting from the markers of our objects, i.e., from the ultimate erosion, our goal is to accurately outline these objects. We could consider using the geodesic SKIZ, and defining each object as the geodesic influence zone of its marker inside the initial set. Unfortunately, this is not a satisfactory algorithm. Indeed, as shown in Fig. 45, the separating lines thus defined between objects are poorly located. This is due to the fact that the *altitudes* of the different markers—i.e. the value associated with them by the quench function—is not accounted for by this method.

The way to design a good segmentation procedure—in taking the above altitudes into account—is to use the geodesic SKIZ repeatedly. Let \( n_m \) be the size of the largest nonempty erosion of \( X \):
\[
X \ominus n_mB \neq \emptyset \quad \text{and} \quad X \ominus (n_m + 1)B = \emptyset.
\]

\( X \ominus n_mB \) is obviously a subset of the ultimate erosion of \( X \). Denote this set by \( X_{n_m} \). Now, consider the erosion of size \( n_m \leftrightarrow \) of \( X \), i.e. \( X \ominus (n_m \leftrightarrow)B \). Obviously, the following inclusion relation holds:
\[
X_{n_m} \subseteq X \ominus (n_m \leftrightarrow)B.
\]

Now, let \( Y \) be a connected component of \( X \ominus (n_m \leftrightarrow)B \). There are three possible inclusion relations between \( Y \) and \( Y \cap X_{n_m} \):
Figure 43. Threshold decomposition of a grayscale image.

Figure 44. Grayscale reconstruction of mask $f$ from marker $g$. 
1. \( Y \cap X_{nm} = \emptyset \): in this case, \( Y \) is another connected component of \( \text{Ult}(X) \).

2. \( Y \cap X_{nm} \neq \emptyset \) and is connected: here, \( Y \) is used as a new marker.

3. \( Y \cap X_{nm} \neq \emptyset \) and is not connected: in this last case, the new markers are the geodesic influence zones of \( Y \cap X_{nm} \) inside \( Y \).

These three different cases are shown on Fig. 46. Let \( X_{nm-1} \) be the set of markers produced after this step. To summarize what we have just said, \( X_{nm-1} \) is made of the union of:

- the geodesic influence zones of \( X_{nm} \) inside \( X \ominus (n_m \leftrightarrow 1)B \),
- the connected components of \( \text{Ult}(X) \) whose altitude is \( n_m \leftrightarrow 1 \).

This procedure is then iterated at levels \( n_m \leftrightarrow 2, n_m \leftrightarrow 3, \) etc., until level 0 is reached. In a more formal way, for every \( 0 < n < n_m \), let us introduce the following notations:

(i) \( u_n(X) \) is the set of connected components of \( \text{Ult}(X) \) having altitude \( n \):

\[ p \in u_n(X) \iff p \in \text{Ult}(X) \text{ and } \text{dist}_X(p) = n. \]

(ii) For every set \( Y \subseteq X \), \( z_X(Y) \) designates the set of geodesic influence zones of the connected components of \( Y \) inside \( X \).

The recursion formula between levels \( n \) and \( n \leftrightarrow 1 \) can now be stated:

\[ X_{n-1} = z_{X \ominus (n-1)B}(X_n) \cup u_{n-1}(X). \quad (52) \]

It is illustrated by Fig. 47.

The set \( X_0 \) that is finally obtained after applying this algorithm constitutes a correct segmentation of \( X \). Fig. 48 presents an example of this binary segmentation algorithm. Applying these notions to the bean segmentation problem, we see in Fig. 49 that, whereas a geodesic SKIZ of our markers results in improper separating lines, the segmentation algorithm we just described yields an accurate segmentation of the beans.

4.4. Watersheds and Grayscale Segmentation

4.4.1. Deriving a General Segmentation Approach

As presented in section 4.3.5, our morphological binary segmentation algorithm is rather complicated. In the present section, we give a much more intuitive approach to it. Consider the function (grayscale image) \( \text{dist}_X \), where \( \text{dist}_X \) is the
Figure 46. The three possible inclusion relations between $Y$ and $Y \cap X_{nm}$.

Figure 47. How to obtain $X_{n-1}$ from $X_n$.

distance function introduced in section 4.2.3, and regard it as a topographic surface. The minimum of this topographic surface are located at the different connected components of the ultimate erosion of $X$. Now, if a drop of water falls at a point $p$ of $\approx \text{dist}_X$, it will slide along the topographic surface, following some steepest slope path, until it finally reaches one of its minima. We define the catchment basin $C(m)$ associated with a minimum $m$ of our topographic surface in the following way:

**Definition 37 (Catchment basin)** The catchment basin $C(m)$ associated with a (regional) minimum $m$ of a grayscale image regarded as a topographic surface is the locus of the points $p$ such that a drop falling at $p$ slides along the surface until it reaches $m$.

This definition is not very formal, but has the advantage of being intuitive. In our example, the catchment basins of the function $\approx \text{dist}_X$ exactly correspond to the regions that were extracted by the algorithm presented in section 4.3.5, as illustrated by Fig. 50. The segmentation achieved in section 4.3.5 exactly corresponds to extracting the catchment basins of the opposite of the distance function.

In fact, the notion of catchment basin can be defined for any kind of grayscale image. Moreover, the algorithm of section 4.3.5 can be easily adapted to the determination of the basins of any grayscale image $I$: it suffices to replace the successive erosions $X \ominus nB$—which correspond to the different thresholds of the distance function of $X$—by the successive thresholds of $I$ (for more details, refer to [140]). The crest-lines separating different basins are called watersheds lines or simply watersheds.

**Definition 38 (Watersheds)** The watersheds (lines) of a grayscale image $I$ are the lines that separate the different catchment basins of $I$.

These notions are illustrated by Fig. 51.
Figure 48. Correct binary segmentation algorithm presented in section 4.3.5.
Watersheds stand out as a powerful morphological crest-line extractor. It is therefore most interesting to apply the watershed transformation to gradient images; indeed, the contours of a grayscale image can be viewed as the regions where the gray levels exhibit the fastest variations, i.e., the regions of maximal gradient. These regions are the crest-lines of the gradient. This remark is illustrated by Fig. 52 and is at the basis of the use of watersheds for grayscale segmentation, as described and illustrated in [8, 5, 136, 140].

Note that in morphology, the word gradient refers to an operation associating with each image pixel the modulus of its gradient—in the classical sense of the word. Most of the time, the gradient known as Beucher’s gradient [100] is used, which is obtained as the algebraic difference of a unit-size dilation and a unit-size erosion of I:

\[ \text{grad}(I) = (I \circ B) \leftrightarrow (I \oplus B). \]

Nonetheless, depending on the type of image contours to be extracted, other gradients may be of interest: directional gradients, asymmetric gradients, regularized gradients, etc [84].

The watershed transformation always provides closed contours and constitutes a very general approach to contour detection. However, it can very rarely be used directly on gradient images without resulting in dramatic over-segmentations: the image gets partitioned in far too many regions, i.e., the correct contours are lost in a large number of irrelevant ones! This problem is mainly due to noise in the data: noise in the original image results in noise in its morphological gradient, this in turn causing it to exhibit far too many regional minima. This directly translates into far too many catchment basins, i.e., over-segmentation!

Several approaches have been proposed in literature to overcome this over-segmentation: for example, some techniques remove arcs of the watersheds based on an integration of the gradient’s gray values along them. Others take the dual point of view and merge adjacent regions (i.e., catchment basins here) when the gray level of the original image over them is comparable. None
Figure 50. Interpretation of binary segmentation in terms of catchment basins of the opposite of the distance function.

Figure 51. Regional minima, catchment basins, and watershed lines.
"object", light on a dark background in the present case

Crest-lines of the gradient = contours

$\text{grad}(I)$

Figure 52. Principle of grayscale segmentation via watersheds of the gradient.
of these techniques is satisfactory in that it is very difficult to incorporate to them knowledge specific to the collection of images under study. Besides, they go against the point of view presented at the beginning of this section, claiming that marker extraction should be the first step of every segmentation.

Therefore, the morphological approach to this problem consists in making use of image-specific knowledge (e.g., size, shape, location or brightness of the objects to extract) to design robust object marking procedures [136, 140, 9]. This step of the segmentation can be completely different from one problem to another. Not only must each object be uniquely marked, the background also needs its own marker(s). In a second step, this binary image of markers is used, not to guide region-merging or arc-removal algorithms, but on the contrary, to modify the gradient image on which watersheds are computed.

More precisely, let $I$ denote the original grayscale image, $J = \text{grad}(I)$ its morphological gradient, and let $M$ denote the binary image of markers. The “modification” of $J$ should result in a grayscale image $J'$ with the following characteristics:

- its only regional minima are exactly located on the connected components of $M$ ($M$ is the set of “imposed” minima);
- its only crest-lines are the highest crest-lines of $J$ that are located between the imposed minima.

The watersheds of $J'$ are thus the highest crest-lines of $\text{grad}(I)$ that separate our markers. Hence, they are the optimal contours corresponding to set of markers $M$ and gradient $J$.

The actual computation from $J$ and $M$ of an image $J'$ with these characteristics has been classically achieved using a three-step process [136]:

1. Set to $h_{\min}$ any pixel of $J$ that is located on a marker, $h_{\min}$ being chosen such that $\forall p, h_{\min} < J(p)$. This results in a new image $J^*$:

$$\forall p, \quad J^*(p) = \begin{cases} h_{\min} & \text{if } M(p) = 1 \\ J(p) & \text{otherwise.} \end{cases}$$

2. Create the following grayscale image $M^*$

$$\forall p, \quad M^*(p) = \begin{cases} h_{\min} & \text{if } M(p) = 1 \\ h_{\max} & \text{otherwise,} \end{cases}$$

where $h_{\max}$ is chosen such that $\forall p, J(p) < h_{\max}$.

3. Use $M^*$ to remove all the unwanted minima of $J'$ while preserving its highest crest-lines between markers. This is done using the dual grayscale reconstruction operation $\rho^*$:

$$\forall p, \quad J'(p) = \rho^*_p(M^*).$$  \hfill (53)

This process is illustrated by Fig. 53. The watersheds of the resulting image $J'$ provide the desired segmentation.

The whole procedure presented above is often referred to as marker-driven watershed segmentation. It is extremely powerful in a number of complex segmentation cases, where it mostly reduces the segmentation task to (1) the choice of a gradient and (2) the extraction of object markers (this latter task can itself be very complex in some cases).

4.4.2. The Electrophoresis Example

Let us illustrate this segmentation paradigm on the two-dimensional electrophoresis image of Fig. 54a (see also [4, 136]). The standard morphological gradient of this image is shown in Fig. 54b. As mentioned above, if we simply compute the watersheds of Fig. 54b, the result is clearly disappointing (See Fig. 54c). Indeed, the gradient exhibits a large number of minima, mainly due to the presence of noise in the original image. Nevertheless, one can notice that all the spots are marked by these minima and hence all the correct contours are present in Fig. 54c. The watershed simply produces over-segmentation.

Avoiding over-segmentation requires the prior extraction of correct spot markers. Since the spots constitute the dark part of the image, they should be interpreted as the image minima. Yet, the direct extraction of the initial image’s minima is not a satisfactory solution, as illustrated by Fig. 54d: once again, many of these minima are
Figure 53. Use of dual grayscale reconstruction to “impose” a set $M$ of minima to a grayscale image $J$. 

1. Grayscale image $J$ and marker-image $M$

2. Construction of image $J^*$

3. Construction of image $M^*$

4. First steps of dual reconstruction of $J^*$ from $M^*$

5. Dual reconstruction of $J^*$ from $M^*$ (cont.)

6. Final modified image $J'$
due to acquisition noise in our data. Here however, filtering image 54a (with a morphological filter called an *alternating sequential filter (ASF)*, see [102, chapter 10] or [103]) is sufficient to produce an image whose minima correctly mark the spots (See Figs. 54e-f).

In fact, this marker extraction step is followed by binary watershed segmentation in order to cut markers like the upper-right corner one, which clearly should mark two different spots. The final image of object markers is shown in Fig. 55a. As concerns the background marker, it is extracted as the set of the highest crest-lines of the original image that separate the spot markers. This is the best way to assure that it will be located on the lightest areas of the image and separate all the object markers. Its determination is done in a similar way as gradient modification (see Eq. 53 and Fig. 53). It is shown in Fig. 55b.

Both sets of markers are then combined in a final marker image, shown in Fig. 55c. It is used to modify the gradient of Fig. 54c, this resulting in Fig. 55d. The watersheds of the latter image provide the desired segmentation, as shown in Figs. 55e-f.

The result is in accordance with our expectations: each spot has a unique contour which is located on the inflexion points of the initial luminance function—i.e. the original image. Given the extracted set of markers, we found the optimal gradient crest-lines, i.e. the best possible segmentation for these markers and, to a lesser extent, this gradient.

### 4.4.3. Difficult Segmentations, Recent Developments

**Hierarchical watershed segmentations**

The method described in the previous section can be applied to a wide range of problems, as long as the preliminary marker extraction step can be performed with sufficient accuracy. However, in some difficult segmentation cases, it is impossible to find markers of the regions or of the objects to extract, since these objects or regions are not themselves well defined. This kind of situation often occurs, for instance, with remote sensing images, where the large variety of zones (fields, roads, houses, towns, lakes, etc., under different lighting conditions) makes it almost impossible to design robust marking procedures.

In this latter case, region-growing types of techniques (see beginning of section 4.4.1) may provide an appropriate solution. In fact, a few morphological region-growing techniques based on the watershed transformation have recently been proposed and seem very promising:

- In the first one, “raw” watershed segmentation is applied to the gradient of the original image, thus resulting in an over-segmented image of catchment basins. Each catchment basin \( C \) is assigned a uniform gray-level corresponding, for example, to the mean gray-level of the pixels of the original image over \( C \). In a second step, this “mosaic image” is regarded as a graph (the dual adjacency graph, see Fig. 56), and morphological operations are performed on this graph [120] in order to merge adjacent regions with comparable gray-levels. In fact, graph-gradients and graph-watersheds can themselves be iteratively applied to the original adjacency graph. This results in an image pyramid containing a hierarchy of contours at different resolutions [121].

- Starting again from an over-segmented “mosaic” of catchment basins (resembling Fig. 54c), some methods introduced by S. Beucher process the adjacency graph of *the watershed arcs*. Catchment basins are hierarchically merged via recursive removal of these contour elements in the graph [6].

- Lastly, in [90], Ph. Salembier and J. Serra approach the problem of general image segmentation via a combination of filters and watersheds that is started at coarse scales and progressively refined into more and more detailed segmentations. Their approach is therefore a region-splitting one, as opposed to the previous region-growing approaches.

Obviously, the watersheds are of enormous interest for complex segmentation problems, and the above solutions are barely starting to explore the extraordinary possibilities of this tool.
Figure 54. Segmentation of electrophoresis gels (1)
Figure 55. Segmentation of electrophoresis spots (2)
Segmentation of Intricately Overlapping Particles

In binary segmentation, the marking by ultimate erosion has its limitations [136]: it is only efficient if, on the one hand, the components of the set $X$ under study are “sufficiently blobby”, and on the other hand, if they do not overlap too much. For instance, when $X$ is composed of two overlapping discs, they are both marked by ultimate erosion if and only if their centers are located on either sides of the radical axis (see Fig. 57).

However, there exist more sophisticated marking tools. Among them, let us mention a transformation called conditional bisector (see [72, page 55] or [113]), in which the marking is no longer related to the maxima of the quench function but to the extrema of its derivative. For $\theta \in [0, \pi/2]$, $X \in \mathbb{R}^2$, and assuming that the skeleton $S(X)$ is continuously differentiable, the $\theta$-conditional bisector of $X$ can be defined as follows:

**Definition 39** The $\theta$-conditional bisector of $X$, denoted $CB_\theta(X)$ is the set of points of $S(X)$ where the derivative of the quench function $q_X$ along the skeleton is within $[\tan(\theta), \tan(\theta)]$.

An efficient algorithm for computing discrete approximations of $CB_\theta(X)$ for any $\theta$ has been proposed in [113]. It is extremely useful for marking “sharp” portions of sets.

Similarly, when the set to be segmented is made up of elongated particles, it may be of interest to mark their extremities. This can be achieved either via techniques based on geodesic ultimate erosions or by means of the maxima of the propagation function [60, 95].

Let us illustrate how the conditional bisector can be used to solve a complex binary segmentation application. Fig. 58a is a binary image of a cross section of vitreous fibers. These fibers are overlapping and need to be separated. A very acceptable marking of these fibers was proposed in [111] (Fig. 58b), resulting in the first watershed segmentation shown in Fig. 58c. However, since some intricately overlapping fibers were not marked, they are not separated in Fig. 58c (these particles are pointed at by arrows).

In order to mark them, a $\pi/6$-conditional bisector was used (see Fig. 58d). Not only does it mark these fibers, it also marks some “necks” between fibers. The latter are eliminated as crossing the separating lines of Fig. 58c (see Fig. 58e). Now, since some of the resulting markers are still slightly disconnected, a connection technique described in [41, 113] was used: Fig. 58e is dilated by a unit size disc, resulting in Fig. 58f. The latter image is then skeletonized with the constraint that the pixels of Fig. 58e belong to the resulting skeleton (see [124] for more details on constrained skeletons). This results in Fig. 58g. Adding to this image the markers of Fig. 58b yields the final marker-image shown in Fig. 58h. At this stage, a new watershed segmentation provides the (almost) perfect result of Fig. 58i.

Figure 56. Mosaic image and associated adjacency graph.
Figure 57. These two discs overlap too much to be both marked by ultimate erosion

5. Conclusions

This chapter dealt with the segmentation of images of particles, and showed that adequate combinations of simple morphological operations can lead to powerful segmentation algorithms: we started from elementary operations and progressively constructed a set of more and more elaborate tools that finally lead to watersheds and grayscale reconstruction. As illustrated on numerous examples, they are extremely powerful operations for binary and grayscale object segmentation problems. In addition, throughout this chapter, we attempted to derive a general philosophy of object segmentation using mathematical morphology. The outcome of this can be summarized in just two words: markers and watersheds.

The examples presented have each illustrated a different aspect of this morphological approach to segmentation. However, in no way are these examples full-size applications! There is practically no trivial segmentation problem, and with each new problem, comes a load of new difficulties to overcome: this leads to new ways to combine existing operations, or even to completely new transformations. The above segmentation philosophy should only be considered as a guideline for the image analyst and should not put any constraints on his or her creativity. In this way, each new application will lead to new advances and the field of morphology will keep progressing and being continually enriched with new operations.
Figure 58. Use of conditional bisector and watersheds for the segmentation of binary images of glass fibers.
Part III
Morphological Algorithms

This chapter is derived from “Morphological Algorithms”, in “Mathematical Morphology in Image Processing”, E. Dougherty Ed., Marcel-Dekker, New York, 1992, pages 255-288 (see [129]).

6. Introduction

This chapter is concerned with the efficient implementation of “low-level” morphological transformations [66, 100]. The qualifier “low-level” used here means that we deal with the implementation of transformations which serve as elementary bricks when solving practical image analysis problems. This does not mean that these transformations are simple, or cannot be decomposed into simpler ones: on the contrary, some of the operations considered in this chapter (e.g., skeletons, watersheds, propagation functions) are complex, both to define and to compute! However, from a user’s perspective, these transformations share the characteristics of being easily and intuitively understandable: for example, watersheds extract from a gray-level image the crest-lines which are located between the minima, top-hat transformations extract thin and light (or dark) regions, skeletons reduce binary shapes to their medial axes, etc.

Solving a moderately complex image analysis application by morphological methods often involves the concatenation of several tens or hundreds of low-level transformations [136]. This is the reason why each of these elementary bricks should be implemented as efficiently as possible. This task can be approached via various algorithmic techniques, the majority of which shall be described in this chapter. Each category of techniques is characterized by its advantages and drawbacks, and illustrated using transformations for which it is particularly suited. For more details, see [92, 123, 99].

The present section is first concerned with the notations that will be used throughout the chapter. A particular transformation, the distance function, is also recalled, since it is used as a “leit-motiv” to illustrate how the described families of algorithms work. The characteristics one expects morphological algorithms to be equipped with are then briefly discussed.

Section 7 is devoted to the most classical morphological algorithms, namely the parallel ones. As explained below, these algorithms turn out to be rather inefficient on conventional computers. A first step towards the implementation of fast morphological algorithms was made by introducing the sequential methods. They are presented and illustrated in Section 8.

Although sequential techniques serve very well for the computation of transformations such as distance function, granulometry function or geodesic reconstruction (see Section 8), they remain inefficient in many cases, since they involve numerous scanings of the entire image. To get rid of this problem, new scanning techniques have been introduced: the algorithms relying on them are such that, throughout the computation of a given morphological transform, only those pixels likely to be modified are taken into account. Such algorithms are based on contours and can be divided up into two families: the chains and loops propagation algorithms [92], which constitute the topic of Section 9 and the queue algorithms [123], discussed in Section 10. Both families have been recently introduced in the morphology world and constitute one of the best possible choices for implementing complex transformations on conventional computers. Not only do these methods lead to faster algorithms, they are also extremely flexible and usually produce more accurate results. We shall illustrate their use by the computation of such transformations as propagation functions, Euclidean distance functions, skeletons and watersheds. Lastly, the conclusion summarizes the qualities and drawbacks of these categories of algorithms and provides some guidelines as to what methods should be used for a given purpose.
6.1. Discrete images and grids, notations

In the following, we consider binary and grayscale images $I$ as mappings from a rectangular domain $D_1 \subset \mathbb{Z}^2$ into $\mathbb{Z}$. A binary image may take only values 0 and 1, and is often reduced to the set of its feature pixels, i.e., pixels with value 1. Many of the algorithms described below extend to $n$-dimensional spaces, but for the sake of simplicity, they will always be presented for 2-D images.

The underlying grid $G \subset \mathbb{Z}^2 \times \mathbb{Z}^2$ defines the neighborhood relations between pixels. $G$ is usually a square grid (of 4- or 8-connectivity) or a hexagonal one (see Fig. 59). $N_G(p)$ denotes the set of the neighbors of a pixel $p \in \mathbb{Z}^2$ according to grid $G$:

$$N_G(p) = \{ q \in \mathbb{Z}^2 \mid (p, q) \in G \}.$$

The discrete distance associated with $G$ is denoted $d_G$: $d_G(p, q)$ is the minimal length of the paths of $G$ connecting $p$ to $q$.

In the present chapter, we mostly use the hexagonal grid (6-connectivity). Indeed, the discrete distance it induces, called hexagonal distance and denoted $d_6$, is more isotropic than the city-block distance $d_4$ or the chessboard distance $d_8$, respectively induced by the square grid in 4- and 8-connectivity [20]. More importantly: the hexagonal grid is a triangulation and thus satisfies the digital Jordan property [92, page 61] according to which every nondegenerate simple loop separates the digital plane $\mathbb{Z}^2$ into two different connected components. As illustrated by Fig. 60, this is not true for square grids and causes endless practical difficulties. For example, when dealing with square grids, consistency makes it often necessary to use 8-connectivity for the objects and 4-connectivity for the background (or vice-versa) [123]. For these reasons, morphologists often prefer the hexagonal grid. Its elementary vectors are denoted $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_6$ and are illustrated by Fig. 61. Note however that all the algorithms described below extend to any kind of discrete grid.

The algorithms themselves are described in a pseudo-code which bears similarities to $C$ and $Pascal$. It makes use of a certain number of keywords and symbols which are summarized in Table 4. Some shortcuts like:

\begin{verbatim}
Repeat until stability {
   For every pixel $p'$ in $N_G(p)$ {
      ...
   }
}
\end{verbatim}

will also be used. Instructions specific to the type of image scanning used by the algorithm being discussed will be introduced as needed.

6.2. The distance function

Throughout the chapter, a particular transformation called the distance function [89, 20] is used to illustrate the four families of algorithms described. This transformation is indeed very typical and gives rise to several different implementations. The distance function $dist_X$ of a set $X \subset \mathbb{Z}^2$ associates with each pixel of $X$ its distance to the background:

$$dist_X \left( \begin{array}{ccc} X & \leftrightarrow & \mathbb{Z} \\ p & \leftrightarrow & \min\{d_G(p, q) \mid q \notin X\} \end{array} \right) \quad (54)$$

The distance function $d_I$ of a binary image $I$ is equivalent to that of its set of feature pixels, i.e., pixels with value 1. In addition, we put conventionally: $\forall p \in D_1, I(p) = 0 \Rightarrow dist_I(p) = 0$. An example of distance function is shown in Fig. 62.

6.3. Estimating the quality of a morphological algorithm

The performance of a morphological algorithm may be defined using three main criteria: speed, accuracy and flexibility.

Speed

This is a crucial issue in the field of image analysis. Indeed, on the one hand, an application program is often designed to be used routinely, either on a large amount of data (e.g., in medicine), or daily (e.g., in quality control). It is then unacceptable for the execution time to be larger than a specified upper bound. On the other hand, even during the solution of a given image analysis problem, many different possibilities have to be considered; for each of them, many transformations have to be used, often repeatedly, with parameter adjustments, filter modifications, etc. It is therefore extremely important for the image analyst to have fast algorithms at his or her disposal: it considerably speeds up this development step and even enables to explore ideas which
Figure 59. Square (a) and hexagonal (b) digital grids. The former one can be considered either in 4- or 8-connectivity.

Table 4
Symbols and keywords of the pseudo-code used for algorithm descriptions.

<table>
<thead>
<tr>
<th>Symbol(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=, ≠, &lt;, &gt;, ≤, ≥</td>
<td>Comparisons of values</td>
</tr>
<tr>
<td>←</td>
<td>Assignment</td>
</tr>
<tr>
<td>{, }</td>
<td>Beginning and end of group of instructions</td>
</tr>
<tr>
<td>;</td>
<td>End of instruction</td>
</tr>
<tr>
<td>/*, */</td>
<td>Beginning and end of comments</td>
</tr>
<tr>
<td>If, then, else</td>
<td>Logical tests</td>
</tr>
<tr>
<td>For... to; While; Repeat until</td>
<td>Classical loops</td>
</tr>
<tr>
<td>true, false</td>
<td>Logical values</td>
</tr>
</tbody>
</table>
Figure 60. Square grids of 4- or 8-connectivity do not satisfy the digital Jordan property, in contrast with the hexagonal grid (c). Indeed, the simple nondegenerate loop drawn in (a) (4-connectivity) separates the discrete plane into three connected components whereas that of (b) (8-connectivity) does not separate anything!

Figure 61. The 6 elementary vectors of the hexagonal grid.
could not be considered otherwise. For example, until recently, the use of the watershed transformation [30, 8] was impossible in practice because of its prohibitive computation time. However, the appearance of the most recent specialized architectures (e.g., the Quantimet 570 of Leitz) and algorithms [123, 140] (see Section 10) has moved it to one of the highest ranking of morphological segmentation tools.

**Accuracy**

An algorithm should of course give results that are as accurate as possible. In fact, most of the time, the result is expected to be totally exact. However, the definition of some transformations—like skeletons (see Section 10)—is sometimes not well adapted to the discrete framework. The algorithms for computing such transformations should then be designed to produce results “as close as possible” to the continuous one. Morphological algorithms are also expected to avoid some of the aberrations associated with the use of discrete grids, like the “cone effect” (see Fig. 63.b). Lastly, one often tries to compute morphological transformations in an isotropic fashion. This involves resorting to discrete distances $d$ closer to the Euclidean one than $d_G$ (see Section 9).

**Flexibility**

By flexibility, we mean any of the following:

- the algorithm is adaptable to other grids,
- it works in both the Euclidean and geodesic [56] cases,
- it allows one to produce several transforms close to one another,
- it is adaptable to several metrics.

Flexible algorithms are very interesting in that they spare the energy of the programmer and reduce the implementation costs.

However, most algorithms cannot be extremely fast, accurate and flexible at the same time. Improving one of these characteristics is generally done to the detriment of the two remaining ones. For example, increasing the accuracy of an algorithm mostly requires additional tests and computations which affect its speed... On the other hand, morphological operations are not implemented in the same way on two different computers: on specialized architectures, complex transformations like skeletons and watersheds will be implemented using built-in thinning and thickening capabilities. However, such algorithms would be terribly slow on classical computers, where their execution times could be close to a couple of hours!
6.4. Image structures and how to access them

The most common data structure to represent and process binary and grayscale images is the two-dimensional array of pixels, all of which are either 1, 8 or 16 bits. This type of structure is very simple and enables a fast access to the neighbors of a given pixel, no matter what discrete grid is used. Several attempts have been made in morphology to manipulate images stored using different data structures, e.g., quadtree and octree [91, 11], interval coding [81], structures stemming from computational geometry, like polygons [82], etc. However, as explained in [123, page 22], none of these structures is really adapted to the implementation of morphological transformations.

This is the reason why, in the following, we only consider images stored as arrays of pixels. We always assume that the number of bits per pixel is sufficient and we do not account for edge effects. Indeed, as explained in [123, chapter 2], they are usually easy to cope with by giving to the pixels of the frame a particular value, usually 0, $\pm \infty$ or $\pm 1$. Here, the images under study are considered to be defined in the entire space $\mathbb{Z}^2$ and to take value 0 outside of their definition domain, unless otherwise mentioned. Additionally, special data structures like loops or queues will be used to manipulate these arrays of pixels efficiently.

7. Parallel Algorithms

This category of algorithms is the most common and classical one in the field of morphology. A parallel algorithm typically works as follows: given an input image $I$, the pixels of $I$ are scanned and the new value of the current pixel $p$ is determined from that of the pixels in a given neighborhood $N(p)$ of $p$. In doing this, the following constraint is satisfied:

"The new pixel values are written in an output image $J$ different from $I$."

$J$ is then copied into $I$, and additional image scanings are performed until a given criterion is fulfilled, or until stability is reached.

Since $I$ is different from $J$, its pixels can actually be scanned in an arbitrary order. In particular, one can imagine parallelizing the processing on some image parts, or even on all pixels, as is done by some specialized architectures. Hereafter, a “parallel scanning” is introduced by a sentence like:

"For every pixel $p$ of $D_I$, do { ..."

The parallel algorithm to determine the distance function of a binary image $I$ in grid $G$ is given below in a pseudo-code.

**Algorithm:** parallel distance function
• Input: \( I \), binary image

• Output: \( J \), gray image defined on \( D_I \); \( J \neq I \).

• Repeat until stability 
  
  For every pixel \( p \in D_I \) 

  \( */\) actual parallel scanning 

  If \( I(p) = 1 \) then 

  \[ J(p) \leftarrow \min\{I(q), q \in N_G(p)\} + 1; \]

  \} Copy image \( J \) in \( I \);

This algorithm is illustrated on Fig. 64. One is easily convinced that the number of scannings it requires is proportional to the largest computed distance. More generally, parallel algorithms usually require a large number of complete image scannings, sometimes several hundred! Therefore, although these algorithms are particularly suited to some architectures, they are definitely not adapted to conventional computers.

The basic parallel algorithms (dilations, erosions, distance function, etc) easily extend to any grid and to \( n \)-dimensional images [39], but here again, prohibitive execution times limit their practical interest. Moreover, the parallel computation of some more complex transformations like skeletons [24] and skeletons by influence zones (SKIZ) [53] is usually achieved via iterations of parallel thinnings or thickenings. These operations involve structuring elements [100], i.e., pixel templates used as probes (see Golay’s alphabet [40]). This is the reason why their adaptation to other grids requires cumbersome neighborhood analyses. This remark is even more true when it comes time to extend these algorithms to \( n \)-dimensional data! Furthermore, although some of them can bring Euclidean distances into play [143], in many cases, the very local way parallel algorithms work leads to approximative transforms (e.g., for skeletons).

8. Sequential Algorithms

In an attempt to reduce the number of scannings required for the computation of an image transform, sequential or recursive algorithms have been proposed [88]. They rely on the following two principles:

• the image pixels are scanned in a predefined order, generally raster (left to right and top to bottom) or anti-raster,

• the new value of the current pixel, determined from the values of the pixels in its neighborhood, is written directly in the same image, so that it is taken into account when determining the new values of the as yet unconsidered pixels.

Note that here, unlike for parallel algorithms, the scanning order is essential. A number of transformations which can be obtained sequentially are described in [57]. In the following, a sequential scanning will be introduced by:

Scan \( D_I \) in raster order 

\[ \text{Let } p \text{ be the current pixel; } \ldots \]

To compute a distance function sequentially, a raster scanning followed by an anti-raster one are sufficient [89]: out of the original binary image \( I \), the raster scanning creates an intermediate gray-level image, whose highest values are located in the lower left part of the connected components of \( I \). Each feature pixel \( p \) of \( I \) is assigned the length of (one of) the shortest path \( P \) between \( p \) and the background, with the following constraint: every path element \( xy \) of \( P \) has either a strictly positive vertical component (i.e., \( xy \) is pointing upwards) or a zero vertical component and a positive left component. This is why a second scanning, of anti-raster type, is necessary to get from \( I \) an actual distance function. This algorithm is given below for the hexagonal distance:

Algorithm: sequential distance function

• input: \( I \), binary image; /*

Distance function is computed directly in \( I \) */
Figure 64. Successive steps involved in the parallel computation of a distance function.
Granulometry function, grayscale dilations and erosions

Let $I$ be a discrete binary image and let $(B_n)_{n \geq 0}$ be a family of structuring elements such that the $B_n$’s are the homothetics of a given convex set $B$. The following equations hold:

\[
B_0 = \{ \emptyset \} \\
B_1 = B \\
\forall n \geq 1, B_{n+1} = B_n \oplus B,
\]

$\oplus$ denoting the Minkowski addition. Let also $\gamma_C$ denote the morphological opening with respect to structuring element $C$ [102]. The granulometry function of $I$ with respect to $(B_n)$, denoted here by $g(I)$, associates with each pixel $p \in D_I$ the smallest integer $k$ such that $\gamma_{B_k}(p) = 0$. In other words:

\[
g(I) \begin{cases} 
D_I & \iff \mathbb{Z} \\
 p & \iff \min \{ k \in \mathbb{Z}^+ \mid \gamma_{B_k}(p) = 0 \} 
\end{cases}
\]

Just like the distance function of a binary image $I$ is the “pile” of its successive erosions, the granulometry function is nothing but the pile of its successive openings with respect to the $B_n$’s. This means that by thresholding $g(I)$ at value $k$, one simply gets the binary opening of $I$ with respect to element $B_{k-1}$. The histogram of $g(I)$ provides the granulometric analysis of $I$. Thus, although this transformation is not very exciting from a theoretical point of view, its great interest comes from the fact that it is possible to obtain it very quickly by using sequential methods. For example, the granulometry function shown in Fig. 65 was obtained in three seconds on a Macintosh II. A very similar algorithm can be used to determine grayscale dilations and erosions by the $B_n$’s.

Morphological shadowing

This is another case where sequential methods outperform all other techniques. The shadowing of a grayscale image is realized through dilation by a ray of the 3-D space $\mathbb{Z}^2 \times \mathbb{Z}$. This is often interesting for visualization purposes and can be efficiently implemented thanks to a recursive algorithm described in [122]. An example of morphological shadowing of a distance function is shown in Fig. 66. One can notice in this example that the crest-lines of the distance function have been highlighted, thereby leading to a family of methods for computing skeletons.

Geodesic reconstruction

In geodesic morphology [56], a transformation called reconstruction turns out to be of immense interest. Given two (binary or grayscale) images $f$ and $g$ such that $g \leq f$ (i.e., for every pixel $p$, $g(p) \leq f(p)$), the reconstruction $R_f(g)$ of $f$ from $g$ is obtained by dilating $g$ geodesically under $f$ until stability is reached. $f$ is called the mask image whereas $g$ is the marker. More precisely, denote by $B$ the elementary ball of the grid being used. For example, $B$ is hexagon $H$ in 6-connectivity, 5-pixel square $S_1$ in 4-connectivity or 9-pixel square $S_2$ in 8-connectivity (see Fig. 23). Let $\delta_f$ stand for the dilation with respect to $B$ and $\wedge$ be the pointwise minimum. The reconstruction of $f$ from $g$ is ob-
Figure 65. A binary image and its corresponding granulometry function with respect to a family of squares.

Figure 66. Shadowing of the distance function of a binary shape.
tained by iterating the following operation until stability is reached:

\[ g \leftarrow \delta_B (g) \land f. \]  

(57)

In the binary case, reconstructing \( f \) from \( g \) allows us to extract those connected components of binary image \( f \) which contain at least a pixel of \( g \) [136]. This extends to the grayscale case in terms of peaks: as illustrated by Fig. 67, only those peaks of \( f \) that are marked by \( g \) are preserved through reconstruction. From Eq. (57), it is straightforward to derive a parallel algorithm for binary and grayscale reconstruction, but it is particularly inefficient on standard equipment. Here, it is much preferable to resort to a sequential algorithm [57, 123]: like the sequential distance function algorithm previously described, the present one works by propagating information downwards in a raster scanning and then upwards in an anti-raster scanning. Here however, as explained later, these raster and anti-raster scanings have to be iterated until stability is reached (see Fig. 69). This algorithm is described below for the hexagonal case. It works for both the grayscale and the binary case and usually only requires around 10 complete image scanings:

**Algorithm**: sequential reconstruction

- **Mask**: \( I \), binary or gray image
- **Marker**: \( J \), image defined on domain \( D_I \) /* Reconstruction is computed in marker-image \( J */
  
  /* Note**: we assume that
  \[ \forall p \in D_I, \ J(p) \leq I(p) \] */

- **Repeat until stability** { 
  
  **Scan** \( D_I \) in raster order { 
  
  Let \( p \) be the current pixel;
  
  \[ J(p) \leftarrow (\max\{J(p), J(p + \bar{u}_1), J(p + \bar{u}_2), J(p + \bar{u}_3)\}\)\land I(p); \]

  }
  
  **Scan** \( D_I \) in anti-raster order { 
  
  Let \( p \) be the current pixel;
  
  \[ J(p) \leftarrow (\max\{J(p), J(p + \bar{u}_4), J(p + \bar{u}_5), J(p + \bar{u}_6)\}\)\land I(p); \]

  }

As mentioned earlier, reconstruction is a particularly powerful morphological tool. Its several binary applications (filtering, hole filling, etc) are rather well-known, but it is even more useful in the grayscale case [136, 123]. For example, to extract the maxima of an image \( I \), it suffices to reconstruct \( I \) from \( I \leftrightarrow 1 \). By algebraic difference between \( I \) and the reconstructed function, one gets the desired maxima. Alternatively, the example of Fig. 68 illustrates the use of grayscale reconstruction for picture segmentation: Fig. 68.a is an image of blood vessels in the eye in which microaneurisms have to be detected. They are small compact light spots which are disconnected from the network of the (light) blood vessels. To extract them, the first step is to perform a series of openings of Fig. 68.a with respect to segments of different orientations. These segments are chosen to be longer than any possible aneurism, so that the aneurisms are removed by any such opening. On the other hand, since the blood vessels are elongated and light, there will be at least one orientation at which they are not completely removed by opening. After taking the supremum of these different openings, one gets Fig. 68.b, which is still an algebraic opening of Fig. 68.a [102]. It is used as marker to reconstruct the blood vessels entirely. Fig. 68.c is the result of the grayscale reconstruction of Fig. 68.a from Fig. 68.b. Since the aneurisms are disconnected from the blood vessels, they have not been reconstructed! Thus, by algebraic difference between Fig. 68.a and Fig. 68.c, followed by thresholding, the microaneurisms shown in Fig. 68.d are easily extracted.

The above reconstruction algorithm also underscores some typical drawbacks of sequential algorithms. For example, in the binary case, when the mask is a “rolled-up” particle, the number of image scanings required for its reconstruction may be very important, as illustrated by Fig. 69. However, only the values of a few pixels are actually modified after each scanning!
For this reason, a further step in the design of efficient morphological algorithms consists in considering only the pixels whose value may be modified. A first scanning is used to detect the pixels which are the process initiators and are typically located on the boundaries of the objects or regions of interest. Then, starting from these pixels, information is propagated only in the relevant image parts. The categories of algorithms described in the next two sections rely on these principles. They both require a random access to the image pixels as well as to the neighbors of a given pixel.

9. Loop and Chain Algorithms

These methods were proposed in 1988 by M. Schmitt [92] and are based on the following simple remark:

in a metric space \((E, d)\), the boundary of the dilation \(\delta(X)\) of a set \(X\) by an isotropic structuring element is a curve which is parallel to the boundary of \(X\).

This is illustrated by Fig. 70. Hence, if one is able to determine quickly the curves parallel to a given one, the calculation of isotropic dilations can be efficiently realized. This process can then be used to compute a large number of other morphological transformations which can be defined from isotropic dilations in an incremental fashion. Among others, distance functions, which are nothing but “piles” of erosions, are attainable this way.

The first step of these algorithms therefore consists in a tracking of the contours of the image \(I\) under study and in their encoding as Freeman loops [34]. A loop \(L\) is a data structure made of

1. an origin pixel \(O_{r_L}\),
2. a length \(l(L)\),
3. a list of \(l(L)\) integers of the segment \([0, 5]\), coding the elementary vectors \(\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_5\) of the hexagonal grid.

The two extremities of a loop coincide. An example of a loop and of its encoding is shown in Fig. 71.

Given a loop \(L\) coding the boundary of a set \(X \subset \mathbb{Z}^2\), the dilated loop \(\delta L\)—coding the boundary of \(\delta(X)\)—is determined by means of rewriting rules. These rules allow one to derive from two successive contour elements of \(L\) a certain number (between 0 and 5) of contour elements of \(\delta L\). The dilated loop is thus obtained from \(L\) in linear time with respect to \(l(L)\). In the hexagonal case, there are exactly six rewriting rules (up to the six rotations), which are illustrated by Fig. 72. Rule number 4 may seem useless, but is in fact essential as soon as two successive dilations have been performed [92].

Now, once the dilated loops are determined, they must be written in the image and the corresponding pixels have to be given the appropriate value. For example in the case of a binary dilation, the pixels corresponding to the dilated loops have to be assigned value 1. In fact, while the loops are written in the original image, one can

Figure 67. Grayscale reconstruction of \(f\) from \(g\).
Figure 68. Use of grayscale reconstruction for image segmentation: (a) original image blood vessels, (b) supremum of openings by segments, (c) reconstructed image, (d) microaneurisms obtained by subtracting (c) from (a) and thresholding the result.
Figure 69. The sequential computation of a binary reconstruction in a rolled up mask may involve several complete image scannings; here, the hatched zones represent the pixels which have been modified after each step.

Figure 70. The boundary of the dilated set $\delta(X)$ is parallel to the boundary of $X$. 
Figure 71. Example of a loop and of its encoding.

Figure 72. Rewriting rules allowing to determine $\delta L$ from $L$. 
detect if some of them intersect. For example, it may well happen that two loops coming from two different connected components intersect after a dilation step, as illustrated by Fig. 73. In such cases, the overlapping parts become useless and can be cut. The remaining loop parts are called chains: they are nothing but loops whose extremities do not coincide, and are manipulated exactly as loops by using the rewriting rules of Fig. 72. They can well be cut again at further steps. An example of a chain dilation is shown in Fig. 74. The succession of operations described in this paragraph is referred to as adjustment. More precisely, during this adjustment step, one only keeps the chain or loop parts which are located in a given mask (set of pixels having a certain value) and gives them the appropriate value.

As an illustration, let us consider again the case of the hexagonal distance function. To determine it, we iterate dilations and adjustments of the chains until stability is reached. After each step, the value given to pixels in the “adjusted” chains is incremented by 1.

**Algorithm:** distance function by chains and loops

- **input:** $I$, binary image; /* Distance is directly computed in $I$ */
- **Track the contours of $I^C$** (complement image) and encode them as loops;
- **dist ← 2;** /* variable containing the current distance */
- **Repeat until there remain chains or loops** {
  - Dilate the chains and the loops;
  - Adjust them in the mask
  - $\{p \in D_I \mid I(p) = 1\}$, giving the corresponding pixels value dist;
  - dist ← dist + 1;
}

Like almost all the algorithms relying on this chain propagation principle, the above one only requires two image scannings: one for the contour tracking step plus one scanning of the feature pixels only in the propagation step (in fact here, to avoid an additional scanning, the algorithm is designed to yield dist$_I + 1$).

Chains and loops algorithms are thus extremely fast. Moreover, after the initial contour tracking is achieved, loops and chains may well be propagated inside a given mask: for this reason, the present methods are particularly suited to the computation of binary geodesic transformations [92]. In this framework, they are the fastest techniques available.

**Propagation function**

The previous remark is particularly true for the propagation function of a simply connected set, where chain propagation methods provide the only known efficient algorithm [93, 60]. Recall that the propagation function $p_x$ associates with each pixel of a connected set $X$ its geodesic distance to the farthest pixel of $X$:

$$p_x \left( X \leftrightarrow \mathbb{Z}^+ \right) \leftrightarrow \sup\{d_X(p, q) \mid q \in X\}. \quad (58)$$

The algorithm for computing $p_x$ is detailed in [92] for the hexagonal case. It basically works through the determination of a supremum of geodesic distance functions, each of these being obtained via chain propagations. This transformation is illustrated by Fig. 75 in the case of a 4-connected square grid. It has a very large number of practical applications, ranging from the extraction of extremities and geodesic centers [92] to the determination of anti-skeletons [94].

**Euclidean distance function and applications**

Chains and loops algorithms are flexible in that they can also be adapted to better distances: for example, the dodecagonal one can be obtained by modifying the rewriting rules of Fig. 72 [92, pages 86–89]. But it is even possible to adapt these algorithms to actual Euclidean distance [125]: the idea is to modify the chain and loop structure as well as the rewriting rules of Fig. 72 in such a way that Euclidean distances are conveyed in the image by these structures. Previous algorithms were of sequential type and only
Figure 73. After dilation, certain loop parts must be eliminated: this is the adjustment step, in which *chains* are created. Here, the original (two) loops are located around the gray areas, and the parts of the dilated loops to be cut are drawn in a bold stroke.

Figure 74. Dilation of a chain. Note that a loop has been created here, which will be eliminated during the adjustment step.

Figure 75. Level lines of the 4-connected propagation function of a binary image.
yielded more isotropic distances [20] or approximations of Euclidean distance functions [28]. An example of exact Euclidean distance function is shown in Fig. 76.

The same technique extends to the determination of Euclidean skeletons [75] and skeletons by influence zones [53]. Additionally, Delaunay triangulations [82, 18], Gabriel graphs [38] and relative neighborhood graphs [115] can be derived from these methods, and obtained in arbitrary binary pictures [123, 125]. Examples can be found in the chapter of the present book entitled “Graph Morphology in Image Analysis” [47].

Morphological transformations with arbitrary structuring elements

Other extensions of the present loop based methods include efficient algorithms for computing binary dilations, erosions, openings and closings with structuring elements of arbitrary size and shape [126]. Here, chains and loops are no longer propagated in the image. Instead, the involved structuring element is encoded appropriately and propagated along the loops representing the set to be dilated or eroded. Combining an erosion and a dilation step allows us to determine openings and closings equally well, as illustrated by Fig. 77.

To summarize, chains and loops are particularly efficient for the computation of binary morphological transformations and give rise to the interesting extensions described above. Unfortunately, they are not easy to adapt from one grid to another. Furthermore, they do not extend to multidimensional spaces. In these respects, the algorithms based on queues of pixels which are discussed below are much more general.

10. Algorithms Based on Queues of Pixels

In this section, we again satisfy the principle according to which only the “interesting” image pixels are considered at each step. The image under study is regarded as a graph whose vertices are the pixels, and whose edges are provided by the discrete grid G. Then, instead of loops and chains, we make use of a queue of pixels to perform breadth-first scannings of this graph. This idea has already proved to be particularly interesting in image analysis and morphology [118, 123].

A queue is a First-In-First-Out (FIFO) data structure, which means that the pixels which are first put into it are those which can first be extracted. In other words, each new pixel included in the queue is put on one side whereas a pixel being removed is taken from the other side (see Fig. 78). In practice a queue is simply a large enough array of pointers to pixel, on which three operations may be performed:

- \texttt{fifo-add(p)}: puts the (pointer to) pixel \( p \) into the queue.
- \texttt{fifo-first()}: returns the (pointer to) pixel which is at the beginning of the queue, and removes it.
- \texttt{fifo-empty()}: returns true if the queue is empty and false otherwise.

The implementation of our distance function using this queue and the above operations is accomplished as follows:

\textbf{Algorithm:} distance function using a queue of pixels

\begin{itemize}
  \item input: \( I \), binary image; /*
      Distance function is computed in \( I \)
      directly */
  \item For every pixel \( p \in D_I \), do {
      /* detection of the pixels to be initially put on the queue */
      \begin{itemize}
        \item If \( I(p) = 1 \) and
        \item \( \exists p' \in \mathcal{N}_G(p), I(p') = 0 \) {
          \begin{itemize}
            \item \texttt{fifo-add(p)};
            \item \( I(p) \leftarrow 2; \)
          \end{itemize}
        }
      \end{itemize}
  }
  \item While \texttt{fifo-empty()} = false {
      \( p \leftarrow \texttt{fifo-first}() \);
      For every \( p' \in \mathcal{N}_G(p) \) {
        If \( I(p') = 1 \) {
          \begin{itemize}
            \item \texttt{fifo-add(p')};
            \item \( I(p') \leftarrow 2; \)
          \end{itemize}
        }
      }
    }
\end{itemize}
Figure 76. Comparison between hexagonal and Euclidean distance function.

Figure 77. Binary opening and closing by an arbitrary (and weird!) structuring element.
Here again, this algorithm actually yields $\text{dist} + 1$, a trick which avoids an additional image scanning.

Like the methods described in the previous section, queue based algorithms are extremely efficient, in both the non-geodesic and the geodesic cases. The simplicity of the above distance function procedure is also interesting, and this characteristic is shared by most FIFO algorithms. They are thus more suited than the chain propagation ones to the development of procedures for computing complex transformations like grayscale reconstruction (for an efficient alternative to the sequential algorithm described in Section 8, see [123, chapter 6]), skeletons and watersheds. The latter two are briefly described and illustrated below.

Moreover, contrary to the chain propagation algorithms, the present ones are extremely easy to adapt from one grid to another, since it suffices to modify the function that generates the neighbors of a given pixel. Similarly, their extension to $n$-dimensional images and even to graphs is straightforward. They have been used to implement a large number of morphological transformations on graphs [120], which have already been used in physical applications [138, 49] and are expected to be of great interest for complex picture segmentation tasks [121]. Several ideas and algorithms about mathematical morphology on graphs can be found in [119] or in chapter “Graph Morphology in Image Analysis”, in the present book [47].

**Skeletons**

The skeleton transformation is widely used in morphological image processing. It was introduced by Blum in 1961 as the *medial axis transformation* [15]. The definition he proposed is based on the concept of grassfire: assuming a grassfire starting from the boundary of a set $X \subset \mathbb{Z}^2$ is propagating within it a uniform speed, the skeleton $S(X)$ of $X$ is the set of the pixels where different firefronts meet. This is illustrated by Fig. 79. A more formal definition of the skeleton was then proposed by Calabi [24], based on the notion of maximal ball: the skeleton of $X$ is defined as the locus of its maximal balls for the used metrics. One can show that this skeleton can be obtained as the set of local maxima of the distance function of $X$ [123, 124].

Unfortunately, a well-known result is that the skeleton by maximal balls, sometimes called the *true skeleton*, is not necessarily connected even though the original set is! However, to be useful in practice, skeletonization needs to be a homotopic transformation [100]: roughly speaking, it needs to preserve the number of connected components and the number of holes in the original set, as well as the inclusion relationships between these components and holes. The practical problem with the implementation of skeletons consists therefore in extracting an object of unit-width which would be as close as possible to the skeleton by maximal balls, while preserving the homotopy of the original set. The first class of methods
proposed in the literature are part of what we referred to in this chapter as parallel algorithms (see Section 7). They consist in removing successive peels of the set by means of homotopic thinnings until stability is reached [100]. For example, with hexagonal connectivity one generally performs homotopic thinnings with respect to structuring elements $L$ shown in Fig. 80 [40]. The main drawbacks of these techniques are their inefficiency and the fact that the resulting objects, though homotopic and of unit width, have not much in common with skeletons by maximal balls!

Sequential algorithms based on the crest-lines of the distance function have also been proposed [74, 76]. They are more efficient than the previous ones (they work in a fixed number of images scanings), produce accurate results and can even be extended to Euclidian distances [75]. However, as explained in [124], these algorithms require cumbersome neighborhood analyses and their flexibility is rather poor. Other methods proposed in the literature include computational geometry based algorithms [82] as well as contour-based techniques [2, 142]. They are among the most efficient methods, but are very complex, have little flexibility and only allow the determination of one given type of skeleton (see [123]).

Based on the above remarks, the skeleton algorithm detailed in [124] makes use of homotopic peelings, crest points and contours: more precisely, starting from the boundaries of $X$, successive peelings are realized until stability (i.e., one-pixel thickness) is reached. These peelings— or grassfire propagation process—are efficiently implemented via a queue of pixels. At every step, the current pixel $p$ may be removed (i.e., given value 0) if and only if one of the following conditions is fulfilled:

1. $p$ does not belong to the skeleton by maximal balls. In other words, $p$ is not a crest-point (local maximum) of the distance function.

2. Removing $p$ does not modify the homotopy locally.

The first condition assures the accuracy of the result in that it will be a superset of the skeleton by maximal balls. The second one means that the resulting object is homotopic; the local homotopy checkings are realized via specially designed look-up tables [124]. In this skeletonization process, the pixels belonging to the skeleton by maximal balls play the role of anchor points: the firefronts stemming from the boundaries of $X$ tend to anchor themselves on these particular points. An example of skeletonization based on these principles is shown in Fig. 81.

Like every FIFO algorithm, the present one is particularly efficient since only the feature pixels are considered during the fire propagation step.
Figure 80. The two-phase structuring element $L$ and its 6 rotations.

Figure 81. Construction of the standard skeleton.
For example, the skeletonization process illustrated by Fig. 81 takes less than one second on a Sun Sparc Station 1. The resulting skeletons are also more accurate than with most other methods: they are skeletons by maximal balls to which connecting arcs of unit thickness have been added for homotopy preservation.

The algorithm works in the Euclidean case as well as in the geodesic one. Most interestingly, it allows us to calculate a whole range of different skeleton-like transformations whose computation is hardly possible otherwise: this is simply achieved by using different sets of anchor points. For example, by taking as anchor points the \( \text{regional maxima} \) of the distance function instead of its local maxima, one gets an object referred to in the literature as the \textit{minimal skeleton}. Similarly, an empty set of anchor points results in a “homotopic marking” of the set. Using as anchor points only the maxima of elevation greater than \( n \) yields a smoother skeleton called \textit{skeleton of order} \( n \) \[124\]. Any of these skeletons can then be post-processed via prunings, themselves realized via FIFO algorithms. Fig. 82 shows a sample of these possibilities. To summarize, the present queue-based skeleton algorithm is particularly efficient, accurate and flexible.

**Watersheds**

In the last decade, increasing attention has been put on the watershed transformation as a tool for image segmentation \([30, 8, 6, 140]\). It is defined for grayscale images via the notion of a \textit{catchment basin}: let us regard the image under study as a topographic relief (where the gray-level of a pixel stands for its altitude) on which it is raining on. A drop of water falling at a point \( p \) flows down along a steepest slope path until it is trapped in a minimum \( m \) of the relief. The set \( C(m) \) of the pixels such that a drop falling on them eventually reaches \( m \) is called catchment basin associated with the minimum \( m \). The set of the boundaries of the different catchment basins of an image constitute its \textit{watersheds}. These notions are demonstrated in Fig. 83.

Here again, numerous techniques have been proposed to determine watersheds in digital pictures. The major ones are reviewed in \([123, 140]\).

One of the most interesting algorithms, originally proposed by Beucher, consists of “inverting” the watershed definition. Consider that the minima of the image—regarded here as a 3-D surface—have been pierced and that this image is slowly immersed into a lake. The water progressively floods the different catchment basins, and at some point, water originating from two different minima will merge, thereby connecting the corresponding catchment basins. We prevent this by erecting dams at every place where connection would otherwise occur. Once the surface is totally immersed, the set of dams thus built corresponds to the watersheds of the initial image.

This immersion and dam erection process can now be simulated by an algorithm. The most efficient algorithm described in the literature makes use of FIFO breadth-first scanning techniques for the actual flooding of the catchment basins \([140]\). A labelling of the catchment basins is also used, which automatically prevents the connection of two different basins. It has been shown that the results provided by this technique are more accurate than those of any other method. For example, the cone effect of Fig. 63.b is always avoided. Furthermore, just like almost all other FIFO algorithm, the present one extends to any grid and any dimension in a straightforward manner.

This algorithm dramatically reduces the computation times required for extracting watersheds. On conventional computers, previous approaches typically needed up to a couple of hours! The present one takes around 5 to 10 seconds on a Macintosh II, for a \( 256 \times 256 \) image, thus opening the door to powerful segmentation methods in inexpensive software-based systems.

Let us conclude this section by an example of application of the present algorithm. We consider here Fig. 84.a: it is part of a series of successive images of the same scene, and the problem is to recover the motion of the camera. One of the approaches taken to solve this motion estimation problem consists of decomposing the images into regions and to match these regions over successive time frames \([36]\). Performing this decomposition by means of watershed techniques turns out to provide meaningful regions, which are then easily matched from one image to the next one \([37]\).
Figure 82. Various kinds of skeletons which may be efficiently determined using algorithms based on queues of pixels [124]. This example was produced using the hexagonal grid.

Figure 83. Minima, catchment basins and watersheds.
Following the methodology described in [136], the watershed tool is applied to the morphological gradient of the original image (Fig. 84.c). In fact, to avoid oversegmentation, the watersheds of the gradient are constrained by a marker image. The markers are connected components of pixels belonging to each of the regions to be extracted, and are obtained in this case from the intensity “domes” and “basins” of the original image (see Fig. 84.b). The result of this constrained watershed transformation is the highest watershed lines of the gradient which are located between two markers (Fig. 84.d). The entire segmentation process takes approximately 5 seconds on a Sun Sparc Station 2.

11. Conclusion and Summary

Table 5 summarizes the qualities and drawbacks of the families of algorithms which have been briefly reviewed in this chapter. Now, in practice, what algorithm should be chosen to implement a given transformation in a given environment? Of course, there is no absolute answer to this question. However, the following guidelines can be proposed:

- Parallel algorithms should generally be avoided, unless running on specialized architectures.

- Sequential algorithms constitute one of the best choices to implement grayscale reconstructions, grayscale dilations and erosions with some structuring elements, and granulometry functions.

- Chain and loop methods should be chosen whenever binary transformations are concerned. They are indeed the fastest in this case, particularly for geodesic operations (reconstruction, geodesic distance function, hole filling, labelling, etc.), and provide the only known efficient propagation function algorithm.

- FIFO algorithms will be preferred in all other cases, and in particular in the following ones: geodesic transformations in square grids, n-dimensional or graph morphology, complex transformation like skeletons, SKIZ and watersheds.

Clearly, the last two families are going to become increasingly important in the future. Regardless of accuracy and flexibility considerations, chain or queue algorithms are often faster on conventional computers than parallel algorithms on specialized hardwares! Now, a few years after the introduction of the parallel morphological algorithms, the first specialized hardwares were built on these principles. Between the publication of the sequential distance function algorithm (1968) and its first hardware implementation (1989), more than twenty years have elapsed. Let us hope that we will not have to wait twenty more years to see the first hardware realizing queue based morphological operations. Indeed, this would probably allow us to compute complex morphological transformations in just several hundredths of a second.
Figure 84. Example of a watershed segmentation: (a) original image, (b) marker image, (c) gradient image, (d) final segmentation obtained via watersheds of the gradient controlled by the markers.
Table 5
Respective qualities of the various families of algorithms.

<table>
<thead>
<tr>
<th>Family of algorithms</th>
<th>speed</th>
<th>accuracy (if appropriate)</th>
<th>adaptation to other grids</th>
<th>development ease</th>
<th>hardware implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>parallel</td>
<td>x</td>
<td>xx</td>
<td>xx</td>
<td>xxx</td>
<td>xxx</td>
</tr>
<tr>
<td>sequential</td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
<td>xxx</td>
</tr>
<tr>
<td>loops and chains</td>
<td>xxxx</td>
<td>xxx</td>
<td>x</td>
<td>xxx</td>
<td>x</td>
</tr>
<tr>
<td>queues</td>
<td>xxx</td>
<td>xxx</td>
<td>xxxxxxx</td>
<td>xxxxxxx</td>
<td>x</td>
</tr>
</tbody>
</table>
REFERENCES


93


