Morphologie Mathématique en anglais

Édité par Laurent NAJMAN et Hugues TALBOT

version 3.1
1er février 2010

Table des matières

Première partie

Morceaux choisis

Chapitre 1

Algorithms for mathematical morphology

1.1. Introduction

In this chapter, we deal with the very important implementation problem of various image analysis operators, filters and methods seen in previous chapters.

In general, researchers like to present a novel operator through a mathematical description. However, this may not always be a simple task to translate this description into computer code. Yet, if such an operator is of any interest at all, we should expect to be able to use it in practice. To get there, we need to go beyond pure mathematics into the realm of programming. We need to express the mathematical operator in an applicable algorithm. Note that while a mathematical description is useful for understanding, it is generally not so when an implementation is needed. This usually explains why we often see more than one implementation, with varying characteristics, for most proposed operators. Also, the evolution of computer architectures and programming techniques imply that new implementations have sometimes been proposed decades after the initial definition of some operators.

Rather than attempting to build a comprehensive database of all algorithms and data structures that have ever been used for implementing morphological operators, an undertaking that would be enormous and quite uninteresting to read, in this chapter we concentrate on the purer algorithmic aspects of mathematical morphology. We therefore ignore some specific implementation aspects dealing with both software and hardware.

Chapitre rédigé par Thierry GÉRAUD, Hugues TALBOT et Marc VAN DROOGENBROECK.

Formally, an algorithm is defined, since the Babylonian time till Ada Lovelace [STU 87], as a series of operations sequenced to solve a problem by a calculation. In mathematical morphology, filters or operators usually operate on sets or functions and they are defined in formal mathematical terms.

An algorithm is thus the expression of an efficient solution leading to the same result as the mathematical operator once applied on input data. This translation process in a mathematical algorithm aims to facilitate the implementation of an operator on a computer as a program regardless of the chosen programming language. Consequently, the algorithmic description should be expressed in general and abstract terms so to allow to decline an implementation in any environment (platform, language, toolbox, library, ...).

Computer scientists are familiar with the formalization of the concept of an algorithm and computation on a real computer with the Turing machine [TUR 36]. This formalization makes it possible, although not in a tractable form, to implement any correct algorithm. Rather than to describe algorithms with this formalism, we use a more intuitive notation that, in particular, relies on non trivial *data structures*.

This chapter is organized as follows. In Section 1.2, we first discuss the translation process of data structures and mathematical morphology definitions in computational terms. Then, in Section 1.3, we deal with different aspects related to algorithms in the scope of mathematical morphology. In particular, we propose a taxonomy, discuss possible trade-offs, and present algorithmic classes. These aspects are put into perspective for the particular case of the morphological reconstruction operator in Section 1.4. Finally, historical perspectives and bibliographic notes are presented in Section 1.5.

1.2. Translation of definitions and algorithms

1.2.1. *Data structures*

Before discussing an algorithm, once has to describe the data to be processed and how they materialize once they are not pure mathematical objects anymore.

An image f is a function from a space $\mathcal{E}$ to a space $\mathcal{V}$. Since infinite spaces can nor be stored neither be handled appropriately, these two spaces are always sampled (or discretized) to provide $E \subset \mathcal{E}$ and $V \subset \mathcal{V}$ respectively :

$$f : \left\{ \begin{array}{ccc} E & \longrightarrow & V \\ p & \longmapsto & f(p). \end{array} \right.$$

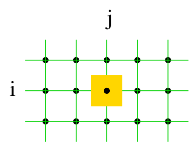

Figure 1.1. *Illustrations of locations defined by a discrete squared grid and a pixel.*

For convenience, a discrete topology with the notion of neighborhood is often associated to E. Let us denote an element of E by p (which stands for "point") and a neighboring point of p by n. In the terms of graph theory, a subset of E is a mesh with nodes (the points p) and a collection of edges that connect points to their neighbors. The most common situation is given by a regular sampling of a subpart of E. The resulting mesh is then regular and points belong to a grid. Such a classical topology for a two-dimensional image defined on a squared grid is shown in Figure 1.1.

A point p of such an image is easily described by two integer indices (i, j), and an image f, that links a point to a value, can be represent in memory by a two-dimensional array of values : $f(p)$ is then equivalent to $f[i, j]$. This common representation of an image has the advantage that point values can be stored and modified independently. In the following, the allocation procedure for a given image f of a value $v \in V$ at point $p \in E$ is denoted in the abstract terms $f(p) := v$ although, from a practical point of view, the underlying mechanism is described by $f[i, j] := v$.

From the implementation perspective, f is not "a mathematical function" but a variable (or a memory) that represents a function at a given time t during the execution process of an algorithm. Formally, an algorithm generates a series of f_t functions, and each allocation changes a function f_t to a next one f_{t+1}. In computational terms, the f variable hides the existence of all the successive functions ; it is similar to a function that evolves over time. The algorithm starts to deviate from mathematics.

1.2.2. *Shape and size of the function domain*

In the field of computer sciences, the representation of an image by an array assumes that the underlying function f is defined on a finite domain. In practice, the function domain is generally a rectangular subset of $\mathbb{Z}^2$ or of $\mathbb{N}^2$ for two-dimensional images. A process that iterates on all points of such an image then accesses all the array elements with two loops, as shown on the left-hand side of the following pseudo-code.

```
for $i$ := 1 to $n_{raws}$
    for $j$ := 1 to $n_{columns}$                    for_all $p \in E$
        ... // use f[i,j]                                ... // use $f(p)$
```

But the abstract right-hand side code is to be preferred as it relates less to a specific implementation and more to the mathematics of the algorithm. In addition, it does not exhibit any restrictive underlying assumption about f : the function domain can have a non-rectangular shape, and its domain does not necessarily have to be two-dimensional. An abstract expression of an algorithm is suitable to hide the details about the representation of the data, and to focus on the functionalities of the algorithm. On the other hand, one has, as well, to provide all the details about the underlying data structure, because performance and complexity issues are closely related to the basic operations on the data. If browsing all the points of a set of N points has a $O(N)$ complexity, the complexity of a random access to the value of a point p has a complexity dependent on the used data structure.

A program accesses values of an image whose data are organized in memory as an array in constant time : reading and writing of image values at a point p have a complexity of $O(1)$, regardless of the used access order. The representation of an image as an array is practical and widely spread. In addition, this representation is compact as it has a negligible overhead to describe the structure itself compared to the data associated to the image. But it might not always be the optimal solution. Consider for example the case of an image that describes the contour of an object. Coding contours of an object by a two-dimensional array guarantees a constant time access to any point of a discrete grid to determine if it belongs to the contour. However, this coding is inappropriate with regards to two other aspects. First of all, the required memory size expands from the size of the contour to the size to the smallest rectangle that encloses the object. Next, the access to any contour point requires searching in an array : one has to search for the first contour point by scanning the image line by line, then in the neighborhood to find the next contour point. For algorithms that operate on a the contour of an object directly, for example a morphological dilation, it might be advisable to use a more appropriate data structure, like a list of contour points.

The conclusion is that any data structure used by an algorithm impacts on the complexity of the algorithm.

1.2.3. *Structure of a set of points*

The classical representation of a function as an array is flexible enough to be able to define a set of points. Indeed, if the destination space of f is the set of booleans ($V = \mathbb{B}$), f can be interpreted as the characteristic function of a set of points : formally, $F = \{\, p \in E \,|\, f(p) = true \,\}$. f is then a binary image. The set $F \subset E$ encodes an object while its complementary $E \setminus F$ denotes the image "background". To scan all the points of F, it is then sufficient to look for points p in the domain E satisfying $f(p) = true$. In the following, we make no notational difference between a set ($F \subseteq E$) and its characteristic function ($F : E \to \mathbb{B}$).

1.2.4. *Notation abbreviations*

For convenience, we use the following notation abbreviations to describe algorithms :

	Extensive notation	Abbreviation
Scans all the points of a set $F \subseteq E$	**for_all** $p \in E$, **if** $F(p) = true$, ...	**for_all** $p \in F$, ...
Allocates a constant c to all the points of an image f	**for_all** $p \in E$, $f(p) := c$	$f := c \,\square$
Copies the content of an image f_1 to an image f_2	**for_all** $p \in E$, $f_2(p) := f_1(p)$	$f_2 := f_1$

As explained before, with this kind of abstract formulation, it is possible to bypass some practical difficulties. For example, there is no need for a two-dimensional image to be rectangular (its domain may be arbitrarily shaped). One problem occurs when dealing with the neighbors of points located on the border of the image. Again, we define the following abstract formulation to operate on all the neighboring points of a point p on turn, regardless of the shape of the neighborhood :

```
for_all $n \in \mathcal{N}(p)$
  ...
```

With this notation, one can focus on the description of the algorithm and ignore useless implementation details.

Finally, we use the symbol $\rhd$ to the denote the conventional video order to scan all the points of an image (from the upper left corner to the bottom right corner, line by line). A reverse video order (from the bottom right corner to the upper left corner, line by line) is denoted by $\lhd$.

1.2.5. *From a definition to an implementation*

As said previously, if morphological operators are often described in mathematical terms, the translation of it in algorithmic terms is not always straightforward. Experience also shows that, in the event that such a translation is possible, its efficiency is often questionable (sometimes, it is even the worst possible implementation). This should not come as a surprise as the purpose of a mathematical definition is to ensure the correctness of an algorithm rather than to provide hints for an appropriate algorithm.

To discuss the suitability of algorithms, let us consider a binary dilation. There are several equivalent ways to define the dilation of a set X by a set B :

$$\begin{aligned} X \oplus B &= \bigcup_{b \in B} X_b && (1.1) \\ &= \{\, x + b \in E \mid x \in X,\, b \in B \,\} && (1.2) \\ &= \{\, p \in E \mid \exists b \in B,\, p - b \in X \,\} && (1.3) \end{aligned}$$

Definition (1.1) leads to a so-called trivial algorithm given as algorithm (1) as shown in Figure 1.2 (from line 1 to line 16). In computational terms, the main procedure is SETDILATION, which uses a function TRANSLATE. In algorithmic terms, it is easy to see that the method derives directly from the definition, which justifies the soundness of the algorithm.

For a processor, accesses to memory account for a significant computational cost both in reading mode but even more in writing mode. For simplicity, let us count only the number of memory allocations (that is, in the writing mode) to $true$. One can see that the trivial algorithm requires $|B| \times |X| \times 2$ allocations, where $|.|$ denotes the cardinality of a set.

A second version, derived from (1.2), reduces the number of allocations by half. It is detailed in Figure 1.2 by the DILDIRECT procedure (from line 33 to 46). But the best approach is given by the algorithm derived from relation (1.3). It improves significantly on the previous algorithms as the number of allocations is lowered to $|X \oplus B|$. Note that the multiplication of the sizes of X and B has been replaced by a summation on their sizes. The corresponding algorithm is detailed in Figure 1.2 by the procedure named DILREVERSE (from line 49 to line 65). This is the “classical” implementation of a set dilation as one can find it in several image processing software packages.

Even with such a simple morphological operator, it appears that there is a major difference between a concise mathematical definition and the transcription of it in an algorithm.

The case of the dilation is representative for the issues raised during the transcription of a definition to an algorithm. Next we consider a more complex algorithm to highlight algorithmic strategies for a single operator.

```
// algorithm (1)

SETDILATION($X$ : Image of $\mathbb{B}$,
              $B$ : Set of Point)
  $\rightarrow$ Image of $\mathbb{B}$
begin
  data $X_b$, $U$ : Image of $\mathbb{B}$
  // initialization to the empty set
  $U$ := false $\square$
  for_all $b \in B$
    // computes $X_b$
    $X_b$ := TRANSLATE($X$, $b$)
    // updates $U$
    $U$ := UNION($U$, $X_b$)
  return $U$
end

TRANSLATE($X$ : Image of $\mathbb{B}$,
            $b$ : Point)
  $\rightarrow$ Image of $\mathbb{B}$
begin
  data $O$ : Image of $\mathbb{B}$
  // initialization to the empty set
  $O$ := false $\square$
  // computes the set
  for_all $p \in X$
    if $p + b \in E$
      $O(p + b)$ := true
  return $O$
end
```

```
// algorithm (2)

DILDIRECT($X$ : Image of $\mathbb{B}$,
           $B$ : Set of Point)
  $\rightarrow$ Image of $\mathbb{B}$
begin
  data $O$ : Image of $\mathbb{B}$
  $O$ := false $\square$ // initialisation
  for_all $p \in E$
    for_all $b \in B$
      if $X(p)$ = true and $p + b \in E$
        $O(p + b)$ := true
  return $O$
end


// algorithm (3)

DILREVERSE($X$ : Image of $\mathbb{B}$,
            $B$ : Set of Point)
  $\rightarrow$ Image of $\mathbb{B}$
begin
  data $O$ : Image of $\mathbb{B}$
  for_all $p \in E$
    for_all $b \in B$
      if $p - b \in E$ and $X(p - b)$ = true
        // existence of a point b
        $O(p)$ := true
        goto next
    $O(p)$ := false // there is no candidate b
    label next
  return $O$
end
```

Figure 1.2. *Dilation of a set X by B.*

Most algorithms in mathematical morphology show a pseudo-polynomial complexity. For example, the trivial algorithm of dilation has a complexity of $\mathcal{O}(N \times M)$ where N and M denotes the number of points of the image and of the structuring element respectively.

1.3. Taxonomy of algorithms

For different reasons, drawing up a taxonomy of algorithms used in mathematical morphology is a hard task. First of all, several pages would not suffice to cite all existing algorithms, because scientists continue to propose new algorithms for operators that are almost forty years old ! Also, one needs a descriptive and complete set of criteria to propose a taxonomy that encompasses a large collection of algorithms. Finally, it has to be noted that there is no universal algorithm valid for any morphological operator. Algorithmic strategies are as diverse as the operators themselves, with specific characteristics, trade-offs, underlying data structures, etc.

1.3.1. *Criteria for a taxonomy*

Taxonomy criteria as applicable to mathematical morphology algorithms are numerous. As an illustration, a non-exhaustive list of criteria runs as follows :

– type of auxiliary or intermediate data structure (file, tree, etc) ;
– order or strategy to browse points in the image ;
– complexity of the algorithm ;
– required memory size ;
– algorithmic properties ;
– operating conditions (and thus limitations) of an algorithm ;
– concerned classes of operators or filters ;
– universality of the algorithm ;
– purpose of the algorithm ;
– processed data types.

Applications constrains lead to additional possible taxonomy criteria : domain of applicability, data range and precision, application objectives, etc.

It would take too long to analyze algorithms with respect to all these criteria. Note however that some criteria lead to a classification of algorithms and that other criteria only permit to distinguish between them. The following tables illustrate some criteria for both cases.

criterion : universality of the algorithm	
restricted	**large**
decomposition of the structuring element (to speed up computations of dilations and erosions) ;	Dijkstra's algorithm (for the computation of a distance function) ; Viterbi's algorithm (for the pruning filter as described in Chapter **??**) ; Prim's or Kruskal's algorithm (for a segmentation process based on the computation of the minimum spanning tree as described in Chapter **??**).

criterion : algorithmic properties	
parallel	**sequential**
classical dilation or erosion algorithm by a structuring element (version (3) as described in Section 1.2.5) ; detection of simple points	chamfer's distance (see Chapter **??**.4.3) ; alternate sequential filters (see Chapter **??**.2.7)

criterion : data range and precision	
small quantization step[1]	**large quantization step**
distribution sort or radix sort ; use of tree representations (see Chapter **??**.2)	fast sorting, heapsort

criterion : browsing order	
propagation of a front	**all points on turn**
distance based dilation ; most algorithms to compute the watershed (see Chapters **??**.5 and 4)	trivial dilation ; hit-and-miss transform (see Chapter **??**.1.3)

The following table elaborates on criteria useful to discriminate between algorithms.

criterion : auxiliary structures
arrays, files, priority queues, trees, graphs, etc.

criterion : purpose
data simplification, resulting transform, computations / estimations on the data, partitioning, etc.

criterion : processed data type
pixels, textures, objects, regions, contours, etc.

1.3.2. *Trade-offs*

An image processing chain is a particular case of the transcription of a scientific calculus by a processing unit. When part of this chain relates to a morphological operator, it is constrained by a general framework. In particular, a typical constraint is the need to find an appropriate balance between three antagonistic notions, as described hereafter.

– Expected computational time or speed. Some applications require to run in real time ; other are less severe on the processing delay. But the absolute execution time is only one part of the discussion related to the implementation of an algorithm. It is also important to analyze the variability of the running time. A hardware implementation will favor a constant execution time, even at the cost of an increased execution time.

– Storage space. Resources proper to the algorithm in terms of disk space or memory usage are generally limited. Likewise, the amount of data handled by an application can be bounded. Therefore available storage capacity plays a crucial role in the choice of an algorithm and adequate auxiliary data structures.

– The results of a computation have a given level of precision. They could be exact or approximated, which then requires to elaborate on the expected level of precision. Many practical situations do not intrinsically require an exact calculation, or at least not for all points.

Practitioners are motivated by application requirements when they implement a morphological operator but they are constrained by the trade-off triangle made of three antagonistic notions, as illustrated on Figure 1.3. If execution speed is favored, then precision is lowered or computational resources are increased. Note however that modern architectures are capable to perform many morphological operations in real time so that questions about precision become meaningless.

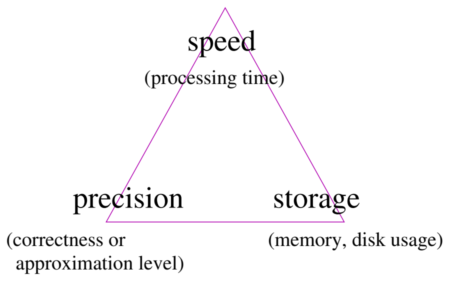

Figure 1.3. *Trade-off triangle.*

1.3.3. *Classes of algorithms and canvases*

Any algorithm running on images usually relies on those following features :

```
POINT_WISE($f$ : Image,
           $h$ : Function)
  $\rightarrow$ $o$ : Image

begin
  for_all $p \in E$
    $o(p) := h(f(p))$
end
```

```
SLIDING_WINDOW($f$ : Image,
               $w$ : Window,
               $h$ : Function)
  $\rightarrow$ $o$ : Image
begin
  for_all $p \in E$
    $o(p) = h(\{ f(q) \mid q \in w(p) \})$
end
```

Figure 1.4. *"Point-wise" canvas (left) and "sliding window" canvas (right).*

– one or several ways of browsing pixels ;

– auxiliary data structures (images and/or any other kind of structures, classical or not) ;

– a processing rationale composed of several steps, most of the time "initialization, loops, and finalization" ;

– and, often, a definition of neighborhood, crucial in mathematical morphology to inspect data around each pixel.

Let us consider the large set of algorithms dedicated to morphological operators. We can observe that it can be split up into several different groups of algorithms sharing the same algorithmic scheme, that is, the same sequence of operations, the same use of control structures (such as loops), and the same auxiliary structures. This leads us to algorithm classes.

In the following we use the term *canvas* to name the description of a class of algorithms. Such a description looks like an algorithm template with some blank parts corresponding to the variability of algorithms belonging in this class. This canvas, or algorithmic scheme, is comparable to use a "pattern" to make clothes where the choice of fabric, color, and ornaments remains to be defined.

Figure 1.4 depicts a couple of canvases that are very common in image processing. The left one is a point-wise operator on image values ; the value at the point p in the output image o is obtained by applying a function h to the value of p in the input image f : $o(p) := h(f(p))$. With $h = C$ (complementation function), this canvas becomes the complementation operator ; with $h = lum$ (luminance function), it is then a conversion of a color image into a gray-scale image.

The canvas in the right part of Figure 1.4 computes $o(p)$ from the set of values belonging to a window w centered at p in the input image. It is therefore the canvas of convolutions $\psi(f) = f * g$ when $h(\{ f(q) \mid q \in w(p) \}) = \sum_q g(p-q)\, f(q)$, and of dilations $\psi(f) = \delta_w(f)$ when $h = \vee$ (supremum).

Algorithmic canvases are interesting for many reasons :

– Firstly, they are general. Each canvas is not specifically dedicated to a single particular operator. To the contrary, a canvas has the ability of being "transposed" in order to handle several operators ; to that aim, one just has to define its variable parts (h in the previous examples).

– Secondly, they are intrinsically abstract. Their expression does not introduce any constraint that would restrict their use to a limited set of input data. For instance, having a double loop over the coordinates of points in the canvases of Figure 1.4 would implicitly state that they are only applicable on bi-dimensional images, which is clearly wrong.

– Lastly, they are helpful for educational purpose. Each canvas translates a meta-algorithm ; understanding it from an algorithmic point of view leads to comprehend any operator that can be derived from this canvas.

1.4. Geodesic reconstruction example

The different classes of algorithms presented in the following did not appear at the same time in the litterature. They belong to an historical perspective which is discussed in Section 1.5.

To illustrate those classes, we will show different algorithms that map the same morphological operator : the geodesic reconstruction by dilation of a function (a gray-scale image for instance) [VIN 93b].

This example, taking the form of an exercise of style, nicely illustrates different algorithmic schemes used by many other operators in mathematical morphology.

1.4.1. *The mathematical version : parallel algorithm*

As a first step, we can start by implementing the geodesic reconstruction by dilation from its mathematical definition :

$$\mathcal{R}^{\delta}_{g}(f) \,=\, \lim_{n\to\infty} \delta^{n}_{g}(f) \,=\, \delta^{\infty}_{g}(f)$$

where :

$$\begin{array}{lcl} \delta^{1}_{g}(f) & = & \delta(f) \bigwedge g, \\ \delta^{n+1}_{g}(f) & = & \delta(\,\delta^{n}_{g}(f)\,) \bigwedge g. \end{array}$$

This reconstruction definition is close to the one given in Chapter **??**.2.2 ; more precisely, it is its generalization to functions. Here δ is the geodesic dilation : $\delta(f) =$

```
RD_PARALLEL($f$ : Image,
            $g$ : Image)
  $\rightarrow$ $o$ : Image
begin

  data
    $o'$ : Image
    $stability$ : $\mathbb{B}$

  // initialization
  $o := f$

  // iterations
  repeat
    $o' := o$ // swap

    // dilation...
    for_all $p \in E$
      $o(p) := \max\{\, o'(q) \mid$
                 $q \in \mathcal{N}(p) \cup \{p\} \,\}$


    // ...under condition
    for_all $p \in E$
      $o(p) := \min\{\, o(p),\ g(p) \,\}$

    $stability := (o = o')$
  until $stability$
  return o

end
```

```
RD_SEQUENTIAL($f$ : Image,
              $g$ : Image)
  $\rightarrow$ $o$ : Image
begin

  data
    $o'$ : Image
    $stability$ : $\mathbb{B}$

  // initialization
  $o := f$

  // iterations
  repeat
    $o' := o$ // memorization

    // first pass (forward)
    for_all $p \in E \rhd$
      $o(p) := \min\{\ \max\{\, o(q) \mid$
                 $q \in \mathcal{N}^-(p) \cup \{p\} \,\},\ g(p) \,\}$

    // second pass (backward)
    for_all $p \in E \lhd$
      $o(p) := \min\{\ max\{\, o(q) \mid$
                 $q \in \mathcal{N}^+(p) \cup \{p\} \,\},\ g(p) \,\}$

    $stability := (o = o')$
  until $stability$
  return o

end
```

Figure 1.5. *Reconstruction canvases (part 1/2) : Parallel algorithm (left) and sequential algorithm (right), described in Sections 1.4.1 and 1.4.2 respectively.*

$f \oplus B$, where $B = \mathcal{N}(0) \cup \{0\}$, with $\mathcal{N}$ the considered neighborhood and 0 the spatial origin.

An implicit but compulsory assumption for this operator to be valid is that $f \leq g$. Put differently, the marker function f to be dilated shall be “under” the mask function g. As in the case of sets, the reconstruction on functions aims at reconstructing some details of g from a simplified image f.

The definition of this reconstruction is itself an algorithm : it is the result of iterations repeated until convergence of a geodesic dilation followed by a point-wise condition.

Such a recursion is implemented with a loop and the algorithm terminates when the result is stable (when no modification has been noticed during the last iteration). The convergence of this algorithm is mathematically ensured, yet it is very slow in practice. Indeed, consecutive passes reconsider parts of the image where local convergence has been reached during previous steps. This algorithm is illustrated by the routine RD_PARALLEL in Figure 1.5.

1.4.1.1. *Similar algorithms*

This kind of algorithm, « repeat modifications until stability, » does not belong exclusively to the area of mathematical morphology ; it is also used, for instance, to compute diffusions.

The complexity of one pass is pseudo-polynomial with respect to the number N of image points and to the connectivity M. In the worst case (as for a Peano curve image), this algorithm has a complexity of $\mathcal{O}(M \times N^2)$. Practically, you should not use this algorithm, except if you are ready to wait for a long time...

A point in favor of this algorithm is that it is easily parallelizable. One just has to observe that the computations performed at every points within both loops only depend on the image obtained at the end of the previous iteration. Those loops thus can be split into several independent tasks, each task running on a single part of an image partition. From that point of view, this algorithm highly contrasts with the alternative ones presented hereafter : with the present parallel version only, there is no dependence between the computation at point p and the computation of its neighbors during the same iteration.

1.4.2. *Sequential algorithm*

Getting an acceptable complexity for the reconstruction can be achieved once one notices that this filter can be expressed in a sequential way. Yet, we will see that it is not a sufficient.

In the parallel version, lines 18 and 20 of Figure 1.5, each dilation is performed independently from the ones of previous iterations, and the current image o is obtained by dilation of the image o' resulting from the previous iteration.

In the sequential version, the auxiliary image o' is not used during the dilation step anymore ; each dilation is performed in-place, see lines 50 to 56 of Figure 1.5. At a given point p, the local dilation value is computed from the neighbor values of p in

o and immediately written in the work image o. Consequently, a modification of o appearing at point p can propagate to other points during the same pass.

In order to ease the comparison between the parallel and the sequential algorithms, we keep a copy o' of o to test the stability condition. Please note that we can get rid of o' if we count the number of modifications in o during the forward-backward pair of passes to test stability.

Despite the propagation mechanism, the complexity of the sequential algorithm is not improved w.r.t. the parallel algorithm ! Again, in the particular unfavorable case of an image representing a Peano curve, the reconstruction requires a number of passes proportional to the number of pixels. However, for convex objects, a forward step and a backward step suffice to obtain the result. In practice, natural images have a lot of such locally convex parts and then those parts are efficiently processed. That explains why one usually observes that this sequential algorithm outperforms the parallel version.

1.4.2.1. *Similar algorithms*

The class corresponding to this sequential algorithm is large. In particular, it includes discrete distance map computation, such as chamfer distances, and the pseudo-Euclidean distance of Danielsson [DAN 80].

1.4.3. ***Queue-based algorithm***

In this algorithm version of the reconstruction, another data structure is used : a queue, First-In-First-Out (FIFO) structure. The general idea is to dilate through a front that propagates into the whole image, while remaining under the condition imposed by g (line 90).

The two main advantages of this approach are its very simple formulation and the one-pass dilation. To the contrary of the previous versions, the need for browsing several times the image pixels is avoided, thus meaning that useless operations (precisely browsing stable regions) are avoided and that complexity is significantly lowered.

Most of queue-based algorithms rely on the same scheme :

– the queue is initialized with some particular points ;

– while the queue is not empty do :
  1) remove the point at the front of the queue file,
  2) perform a given computation at that point,
  3) add some neighbors of that point at the back of the queue.

This scheme is precisely a breath-first traversal of the neighborhood graph of the image starting from the points pushed in the queue during the initialization step. Some

```
RD_QUEUE_BASED($f$ : Image,
               $g$ : Image)
  $\rightarrow$ $o$ : Image
begin
  data
    $q$ : Queue of Point
    $M$ : Image

  // initialization
  $M$ := REGIONAL_MAXIMA($f$)
  for_all $p \in M$
    for_all $n \in \mathcal{N}(p)$
      if $n \notin M$
        $q$.PUSH($p$)
  $o := f$

  // propagation
  while not $q$.EMPTY()
    $p := q$.FIRST()
    for_all $n \in \mathcal{N}(p)$
      if $o(n) < o(p)$ and $o(n) \neq g(n)$
        $o(n) := \min\{o(p),\ g(n)\}$
        $q$.PUSH($n$)

  return $o$
end
```

```
RD_HYBRID($f$ : Image,
          $g$ : Image)
  $\rightarrow$ $o$ : Image
begin
  data
    $q$ : Queue of Point

  // initialization
  $o := f$

  // two−pass sequence...
  for_all $p \in E \rhd$
    $o(p) := \min\{\ \max\{\ o(q)\ |$
              $q \in \mathcal{N}^{-}(p) \cup \{p\}\ \},\ g(p)\ \}$
  for_all $p \in E \lhd$
    $o(p) := \min\{\ max\{\ o(q)\ |$
              $q \in \mathcal{N}^{+}(p) \cup \{p\}\ \},\ g(p)\ \}$
    // ...with enqueuing
    for_all $n \in \mathcal{N}^{+}(p)$
      if $o(n) < o(p)$ and $o(n) < g(n)$
        $q$.PUSH($p$)

  // propagation
  while not $q$.EMPTY()
    $p := q$.FIRST()
    for_all $n \in \mathcal{N}(p)$
      if $o(n) < o(p)$ and $o(n) \neq g(n)$
        $o(n) := \min\{o(p),\ g(n)\}$
        $q$.PUSH($n$)

  return $o$
end
```

Figure 1.6. *Reconstruction canvases (part 2/2) : Queue-based algorithm (left) and hybrid algorithm (right) are respectively described in Sections 1.4.4 and 1.4.3.*

other kinds of browsing are possible ; for instance, we get a depth-first traversal if we switch steps 1 and 3.

In the case of the reconstruction, the initialization starts from the detection of regional maxima and the points of their external contour are pushed in the queue. Those maxima are then propagated by the queue. The operation managed by the queue is not really a “common” dilation, in the sense that the propagation performs neither a dilation with a structuring element, nor a geodesic one. Only maxima values of f are dilated. Yet, it is an algebraic dilation (see Chapter **??**). Since the dilation is only effective under the condition imposed by the mask g, we get the expected reconstruction by dilation, as defined previously [VIN 93b].

The core of the algorithm, given in the left column of Figure 1.6, lies in the propagation process (lines 85 to 91). One can easily notice that every point p is only inspected once, which contrasts with the previous given algorithms. This part of the algorithm thus has a linear complexity.

On the other hand, the initialization step requires to compute the regional maxima of f (line 72) which can be as costly as the propagation step ! Such an operation is equivalent to a connected component labeling, the complexity of which being quasi-linear thanks the Union-Find algorithm [TAR 75].

More generally, using a random-access structure such as a queue or a stack is efficient when the initialization stage extracts adequate information. One can relate that issue with the notion of redundancy of information. In a dilation process relevant information are the localization and the value of local maxima. Algorithms can either ignore such information (those are content blind algorithms), or rely on those information and propagate them. Amongst the fastest (and also the most complex) algorithmic approaches, we have those that detect relevant information *and* propagate them during the *same* browsing of image points. To understand how detection and propagation can merge into a single step, one can look at the algorithm proposed in [Van 05] for the morphological opening. The queue-based reconstruction given in this section does not feature such an elaborate scheme ; in the following, we see that it is possible to get a more efficient reconstruction than the current version.

1.4.3.1. *Remarks*

Similar algorithms : Amongst algorithms similar to the queue-based reconstruction we can cite,

– for the breath-first traversal : distance functions [RAG 93], SKIZ, skeletonization [VIN 91a], ordered dilations [ZAM 80],

– and for the depth-first traversal : the component tree computation given in [SAL 92].

Stack instead of queue : Using a stack, either a simple Last-In-First-Out structure or a pushdown automaton, instead of a queue, can be relevant when combined with the storage of some information related to data. Yet it should often be avoided when the behavior of the algorithm containing it is recursive. Indeed, in such cases, computation calls are stacked so that can be executed later. So, even if the algorithm is theorically correct, it may become inefficient when the size of the stack grows too much (note that its size may be as large as the image).

Priority queue : A simple queue ensures only one property of the contained elements : they are ordered as they were pushed in the queue. This ordering usually corresponds to a particular spatial ordering of the image domain. It is sometimes useful to get a more general property about the element ordering in the queue. For instance, we may want points to be sorted first by their values and then by their introduction order in the queue. To that aim, a particular structure has been proposed by F. Meyer in the context of a watershed transform algorithm : the hierarchical queue [MEY 91]. This is an array of queues whose size as large as the number of values ; it is thus efficient if and only if the values are encoded with a few bits (typically less than 12 bit). Note, however, that it is a particular case of the more general and very common *priority queue* structure, that can be implemented thanks to a heap or a self-balancing binary search tree.

Complexity : Some data structures are more appropriate than others depending on the algorithms, on the nature of input images, and on the operations that rely on those structures. A study of the most classical data structures that we can find in morphology has been realized by E. Breen and D. Monro [BRE 94] ; furthermore they also emphasized the distance existing between theory and practical results, related to the use of those structures. In particular Fibonacci heaps [L. 87], though theoretically efficient, happen to be rather slow when involved in effective algorithms. As a piece of advice, when an efficient priority queue is required, maintaining a stable sort (that is, with the insertion order of elements always preserved in the queue), regardless of the data type used for priorities, then *splay-queues* are often an appropriate choice [SLE 85].

1.4.4. *Hybrid algorithm*

In the hybrid algorithm, given in the right column of Figure 1.6, we can recognize first a sequential part but limited to a couple of passes. During those passes, the reconstruction is thus performed in convex regions of the input image. In the second part of the algorithm, we have the propagation of the queue-based algorithm. This last part completes the reconstruction when the convergence is achieved.

The advantage of this method over the previous one is manifold. The computation of regional maxima is avoided since the queue is initialized with the frontier obtained after the sequential passes. Furthermore, in the case of non-pathological images, most of the reconstruction is actually achieved during this sequential part and the final propagation, much more costly, is only performed on a small part of the image domain.

Please note that hybrid algorithms, where an actual synergy exists between the different approaches they mix, are rather rare in the literature.

1.4.4.1. *Remarks*

We have again the worst cases of the previous algorithms with image representing fractal patterns. In those cases, the queue size remains small but many loops are required for the algorithm to converge.

A way to get better performances from this algorithm is to use a queue implemented by a circular array (more compact in memory and faster when inserting and suppressing elements).

This hybrid version highlights the important relationship between the algorithm in itself and the data structures it rely on. Authors usually study the complexity of the algorithms they propose with respect to the number of manipulation of image data, that is the number of input/output (reading/writing) performed on points. Unfortunately many authors forget to take into account the effective cost of the auxiliary structures involved in those algorithms. The simplest form of such structures is a data buffer in memory, where even a writing operation is not negligible, depending on the buffer size and the amount of RAM available. On the other hand, being able to precisely characterize the cost of an algorithm including its auxiliary structures is a tricky task, in particular because it also highly depends on machine hardware.

1.4.5. *Algorithm based on Union-Find*

The "Union-Find" algorithm is an identification of the equivalence classes of a graph. It is presented in Figure 1.7. This algorithm is relatively complex ; yet, if we do not fully explain it, we will try to describe it roughly. It is composed of three steps : an initialization, a union stage, and a labeling stage (find).

1.4.5.1. *Rough sketch of the algorithm*

The cornerstone of this algorithm lies in a change of representation for images : we move from the notion of pixels, with no connectivity information but local, to a structure of tree, where a node can represent a large connected component of an image. In this tree, the root node maps the whole image domain, whereas leaves relate to local components.

During the initialization, the points of image g are sorted by decreasing value of gray levels and stored in the array S (line 164).

During the union stage, points are browsed as stored in S. The current point can be isolated, in the sense that its neighbors have not yet been inspected ; it thus belongs to a regional maxima. This point then forms a singleton set. If not isolated, it is thus connected to a regional maximum of g ; it is then merged with the corresponding tree and becomes its new root. A key property to understand this algorithm is to realize

```
MAKE_SET($p$ : Point)
begin // creates singleton { p }
  $parent(p) := p$
end

IS_ROOT($p$ : Point) $\rightarrow \mathbb{B}$
begin // tests if p is root
  return $parent(p) = p$
end

FIND_ROOT($p$ : Point) $\rightarrow$ Point
begin // finds the root of p
  if IS_ROOT($p$)
    return $p$
  else
    $parent(p)$ := FIND_ROOT($parent(p)$)
    return $parent(p)$
end

DO_UNION($n$ : Point, $p$ : Point)
begin // merges two trees
  $r$ := FIND_ROOT($n$)
  if $r \neq p$
    if $g(r) = g(p)$ or $g(p) \geq o(r)$
      $parent(r) := p$
      $o(p) := \max(o(r), o(p))$
    else
      $o(p)$ := MAX
end
```

```
RD_UNION_FIND($f$ : Image,
              $g$ : Image)
$\rightarrow$ $o$ : Image
begin
  data
    $parent$ : Image of Point
    $S$ : Array of Point

  // initialization
  $o := f$
  $S$ := SORT(g) // w.r.t. $\rhd$ and $g(p) \downarrow$

  // first pass
  for_all $p \in S$
    MAKE_SET($p$)
    for_all $n \in \mathcal{N}(p)$ if DEJA_VU($n$)
      DO_UNION($n$, $p$)

  // second pass
  for_all $p \in S^{-1}$
    if is_root($p$)
      begin
        if $o(p)$ = MAX, $o(p) := g(p)$
      end
    else
      $o(p) := o(parent(p))$

  return $o$
end
```

Figure 1.7. *Reconstruction by dilation with union-find . This algorithm is described in Section 1.4.5.*

that this regional maximum of g is related to a regional maximum of f. During the process of browsing S, a forest of trees is created that progressively spans the image domain.

In the final phase (find), we differently handle the points such as $g > f$, that receive the maximum value of f in the connected component containing the local regional maximum of g, and the points such as $g = f$, whose value is kept unchanged.

Eventually f has been dilated under the constraint of g and the result is the expected reconstruction.

1.4.5.2. *Details*

During the union stage, the output image o is used as auxiliary data to store the state of all components/trees : either $max(f)$ when dilating is effective ($f < g$ locally) or MAX when the constraint applies. o only takes its final values during the last step of the algorithm.

For every flat zones where it is sure that we will get $o = g$, no tree is computed and the flat zone points are all singletons.

The DEJA_VU function can be evaluated on the fly so this auxiliary structure can be saved.

If g is an image with low-quantized values (for instance an 8-bit image), one can sort points in linear time thanks to a radix sort (a distributed sort based on histogram computation).

1.4.5.3. *Complexity*

In this chapter, a very particular version of the union-find based algorithm is presented (Figure 1.7). It relies on a path compression technique, embedded in the FIND_ROOT routine (line 141), so that the number of recursive calls to this routine is reduced. Yet it is not sufficient to get the best complexity of the union-find algorithm. To minimize the number of recursive calls, it is also necessary to keep all trees as most balanced as possible. For that, one has to add to the version presented here the "union-by-rank" technique. We have voluntarily eluded this technique to make this algorithm more "readable".

Actually, the union-find based reconstruction is quasi-linear in the case of g being a low-quantized image [TAR 75] ; otherwise, for floating data for instance, the complexity is $\mathcal{O}(N\ \log(N))$ due to the sorting step.

1.4.5.4. *Comparison with previous versions*

Although the union-find algorithm is more efficient in theory than the other presented versions, in practice it is not always faster than the hybrid algorithm. However it is emblematic of modern implementations of connected operators : algebraic attribute openings and closings, levellings, watershed transforms [BRE 96, JON 99, MEI 02, GÉR 05]. The representation of image contents as a tree forest allows for a exceptionally rich theoretical description of connected operators [FAL 04, NAJ 06]. See also Chapters **??** and **??**.

With the parallel and sequential approaches, the limiting factor w.r.t. complexity was the number of passes to perform to reach convergence. With the queue-based and the hybrid approaches, the risk came from the queue structure becoming too huge. In the case of the union-find algorithm, the bottleneck is located in the tree root search.

1.4.6. *Algorithm comparison*

In order to compare the five algorithms described earlier, we shall reuse some of the criteria laid out in section 1.3.1. The table below illustrates on the one hand the large diversity of algorithms available to translate a single operator, and on the other hand the difficulty in using the criteria effectively. Indeed, some of these algorithms are not monolithic : they correspond to several criteria at once. The hybrid algorithm, for instance, is only "half-sequential" : it uses both video passes on the image and a propagation front. Regarding the Union-Find algorithm, we could classify it as "sequential" as between the initialization (the sorting pass) and the two other passes, every image pixel is considered exactly three times. However, the order in which they are considered are not the usual video and anti-video sequences.

algorithm name	algorithm class	pixel order	data structures
parallel	parallel	video passes	none extra
sequential	sequential	video passes	none extra
queue-based	queue-based	front	standard queue
hybrid	1/2 sequential	2 passes + front	standard queue
Union-Find	pseudo-sequential	3 passes	array and tree

To compare the performance of these five algorithms for the geodesic reconstruction by dilation, we took as g the standard `lena` 512×512 pixels gray-level image, and for f the pixelwise maximum of g and g rotated 90 degrees clockwise. The neighborhood relation is the 4-connectivity. As we seek a comparison amongst algorithms, we did not perform any compiler-level optimization or used particular techniques such as pointer arithmetic, although they could have resulted in significantly improved running times. The algorithm performance order would have remained the same, however.

algorithm	running time (in sec.)
parallel	25.28
sequential	3.18
queue-based	0.65
hybrid	0.34
Union-Find	0.34

This table illustrates, with performance ranging within a rough factor of 100 to 1, that translating in practice a mathematical operator into actual code can result in widely different results. Producing a "good" algorithm with properties in accordance with what the practitioner expects is indeed an art and a science in itself.

1.5. Historical perspectives and bibliography notes

The history of various algorithms and their links to mathematical morphology as well as other related disciplines would require a book by itself. We lay out in the next section a few, hopefully useful notes.

1.5.1. *Before and around morphology*

Like all scientific endeavors, mathematical morphology was not born, and neither does it evolve in a scientific vacuum. It represents a step toward a better understanding of spatial representations pertaining to physical or virtual objects. Before MATHERON and SERRA named their discipline in 1962 [MAT 02], image analysis already existed and many algorithms had already been developed in comparable disciplines. As well, morphology continued to evolve in a moving context. It is perhaps useful to specify some algorithmic markers in this creative broth.

1.5.1.1. *Graph-based algorithms*

Images are most often represented on regular graphs. As such it is not really surprising that so many mathematical morphology algorithm derive from classical graph algorithms. We may mention Dijkstra's minimal paths [DIJ 59], the minimum spanning tree problem [JOS 56, PRI 57], and the classical Union-Find algorithm [TAR 75]. A good source on many important algorithms is the Cormen *et al.* book [COR 09].

It is more than likely that not all the classical or recent literature on algorithm on graphs has been fully exploited in the context of mathematical morphology. It is probably a very good source of future results.

1.5.1.2. *Discrete geometry and discrete topology algorithms*

Discrete geometry is an active field of research very closely linked to mathematical morphology. The goal of discrete geometry is to redefine and algorithmically exploit the objects and operators of classical geometry, in a purely discrete framework. For instance, lines, planes, intersections, vectors and so on have been partially redefined in such a way [RÉV 91]. Properties of these new objects, although obviously related to the classical ones, are markedly different and usually much more amenable to their use in an algorithmic setting. A recent book on the topic is [GOO 04].

Amongst the most useful algorithms in discrete geometry, also used in mathematical morphology, we can mention distance transforms [ROS 66, BOR 84, SHI 92], that are very useful by themselves, but can also be used to implement binary erosions and dilations [RAG 92].

Discrete topology is a discipline that seek to define topological operators in discrete spaces such as images, but also on arbitrary graphs, on discrete manifolds such as triangulated surfaces, or on complexes [KON 89, BER 07]. The link with morphology is very strong especially in the areas of thinning operators [KON 95] and skeletonization algorithms [MAN 02]. The watershed transform can also be seen as a topological operator [COU 05, COU 10]. The *topological watershed*, of grey-level image I, for instance, is the smallest image, on the lattice of numerical functions, with the same topology as I. The topological watershed operator is also the most efficient known watershed algorithm (with quasi-linear complexity).

1.5.1.3. *Continuous-domain algorithms*

The continuum is not representable exactly on computers, however, some mathematical objects are intrinsically continuous, such as partial derivative equations, and can be used to solve image analysis problems. This approach leads to some interesting algorithms.

Taking as starting point segmentation algorithms such as active contours [KAS 98], it is possible to find links with skeletonization [LEY 92], as well as generalizations of the watershed transform, for instance including some curvature constraints [NGU 03].

Fast marching algorithms [SET 96a] are in essence equivalent to a flexible algorithm for computing the geodesic Euclidean distance transform [SOI 91]. This applies to scalar or tensorial metrics [SET 01]. These algorithms make it possible to propose, in some contexts, a mathematical morphology formulation in the continuous domain [SAP 93]. It is important to note that the original SETHIAN algorithm is only first-order accurate. A fast method to compute the exact geodesic Euclidean distance transform is an open problem at the time of writing.

These methods have been used in morphology for instance in connected filtering [MEY 00], by replacing the dilation operator by a continuous propagation. Formulation of the watershed transform in the continuous domain have been proposed by several authors [NAJ 93, MAR 96], and can be solved using fast marching methods.

The principal benefit derived from a continuous formulation is to abstract away the notion of pixel. Up to an approximation, it is possible to define a dilation of arbitrary, and not only integer radius. It is also possible to propose morphological operators on arbitrary manifolds, for instance on triangulated surfaces, although discrete formulation have been proposed in this context as well.

We will explore other links with the continuous domain through algorithms inspired by linear theories.

1.5.1.4. *Discrete and continuous optimization for segmentation*

Active-contour [KAS 98] or level-set [OSH 88, MER 94, SET 96b] types algorithms are comparable in their approach to segmentation. The idea is to propose a gradient-descent optimization procedure, under some constraints. Common constraints include the necessity of closed contours, the inclusion and/or exclusion of certain zones, topology preservation, etc. These algorithms work in $2D$ or $3D$, and are fairly flexible with respect to the kind of cost function they can optimize. For instance, it is possible to affect costs to region content, motion analysis, regularity, etc. However, the more complex formulations most often cannot be optimized globally.

More recently the image analysis and computer vision communities have found a renewed interest in simpler formulation, but that can be optimized globally, for instance using graph cuts [BOY 04], continuous maximum flows [APP 06], or random walks [GRA 06]. Indeed these formulations are less flexible, but are more reliable and less sensitive to noise. There are some strong links between these techniques and the watershed transform [ALL 07, COU 09].

1.5.1.5. *Linear analysis algorithms*

Here linear analysis means the domain of operators linked to linear integral transforms, such as the FOURIER, RADON and wavelets transforms. These historically were adapted to images from signal processing. In this domain, the basic structure is the classical group, with the addition as base operator. For signals and for some kinds of images (X-Ray, or tomography images) this makes perfect sense as superposition of signals is a reasonable hypothesis. For some kinds of problems this is also a perfectly suited structure. For instance, many sources of noise such as sampling noise is approximated by Gaussian additive white noise, for which there exists an optimal deconvolution in the least square sense.

Mathematical morphology is not linear, and the basic structure is the complete lattice, with infimum and supremum as operators. However there are some links between the two approaches. Of course this is true at the level of applications, but also some tools and approaches are similar. As an example we can cite several works on multi-resolution [HEI 00] and scale-space [JAC 96, VAC 01].

To finish, we present a curiosity : it is possible to define the dilation from a convolution operator :

$$\delta_B[I] = (I \star B) > 0, \tag{1.4}$$

where I is a binary image and B an arbitrary structuring element. Using the FFT, this algorithm can be implemented in constant time with respect to B, which is the only known implementation with this characteristic.

1.5.2. ***Historical perspective on mathematical morphology algorithmic developments***

From the very beginning, the development of mathematical morphology as both a practical and theoretical area was linked to software and hardware developments. The *texture analyzer* was the first machine implementing morphological operators, and was developed at the Ecole des Mines in Paris [MAT 02]. Following this, many advances in this field were the result of a constant synergy between applications, theory, algorithmic and hardware developments.

In the early days, dedicated architectures for image processing were a necessity, due to the relatively weak computing powers of general-purpose architectures. On dedicated hardware, most often access to data can only be realized in an ordered, sequential manner, using video passes on the image. This necessarily also drove the development of corresponding algorithmic techniques.

1.5.2.1. *Parallel algorithms*

The use of video passes on the images and the limited memory of early architectures (typically only three lines could be loaded in main memory at any given time) is a limitation that is still found today, for instance in embedded architectures such as mobile phones. These limitations, amongst others, force the use of *parallel* algorithms. Here this term means a type of processing such that the result on any arbitrary pixel is independent of the result on other pixels. This implies order-independence, and does imply that it is relatively easy to implement such algorithms on massively parallel architectures, but the actual architecture the algorithm is run on does not matter as such. An illustration of this type of algorithm is given on Fig. 1.5 on the left-hand side, and is also described in section 1.4.1. In hardware terms, these algorithms are well-suited to SIMD *Single Instruction Multiple Data* and to the limit case of the so-called *artificial retina*, where each pixel is equipped with its own little processor [MAN 00]. Amongst hardware developments that were known to use parallel morphological algorithms are the Morpho-Pericolor [BIL 92] and the Cambridge Instrument Quantimet 570 [KLE 90] as well as the ASIC (Application-Specific Integrated Circuit) PIMM1 [KLE 89].

The first algorithms implementing the watershed transform, the skeletonization and morphological filters were described and implemented in a parallel fashion. See for instance [BEU 79, MON 68].

1.5.2.2. *Sequential algorithms*

Some (but not all) algorithms can be expressed in a *sequential* manner, which here designates an implementation that uses the current result to derive the next one, most often adopting a particular pixel scanning order. This is illustrated on Fig. 1.5 on the right-hand side, and is also described in section 1.4.2. Sequential algorithms, sometimes also incorrectly called *recursive* algorithms are often more efficient than parallel ones, at least on most general-purpose computers, because they can make use of the local redundancy in many natural images. A typical sequential algorithm is the classical distance transform of ROSENFELD *et al.* [ROS 66], that computes the distance transform in two passes over the image : one in the video scanning order, and the second in the anti-video order. At the hardware level, few sequential algorithms have been implemented, but LEMONNIER [LEM 96], among others, has proposed a sequential watershed transform algorithm.

1.5.2.3. *Breadth first algorithms*

As general-purpose computers became more powerful, the idea of exploring pixels from the border of objects without necessarily following a scanning order imposed by the hardware (in particular the memory wiring) took more hold. This is achieved using a suitable data structure. Among this family of algorithms, we can mention those using boundary paths [SCH 89], queues [VIN 90] and priority queues [MEY 90a]. A classical algorithm belonging to that class is the watershed transform from flooding [VIN 91c, MEY 90b]. This kind of algorithm is on the other hand not well suited to hardware implementations, mostly because the underlying data structure imposes that memory bandwidth be the limit, not computation speed.

1.5.2.4. *Graph-inspired algorithms*

Breadth-first algorithms are a classical approach in graph-based problems. The idea of continuing in this direction and adapt other class of graph algorithms to image data was therefore natural. Among graph-inspired morphological algorithm, we can cite the Image Foresting Transform (IFT) [FAL 04], which is used in segmentation and classification. More recently, the idea of considering an image truly as a graph and to also assign values to edges imposed itself. This makes it possible to define a discrete gradient in a natural way : simply by the numerical difference between two vertices linked by an edge [COU 07]. This had already been proposed earlier by the graph-cut community [BOY 01, BOY 04]. This notion defines a border between regions as a series of edge cuts and not a path of vertices, which solves numerous topological problems. It also paved the way for a unifying framework encompassing many segmentation methods [COU 09].

1.5.2.5. *Topological algorithms*

Beyond the essential notion of simple point, which is the starting point for many efficient topology-preserving algorithms, many works have considered the essential notion of image topology. An important notion is the component tree, used in Section 1.4.5.1 in this chapter, and also described in details in Chapter **??**. The component tree, through its efficient representation of regions and catchment basins, can be used in many interesting algorithms, involving for instance hierarchical segmentation, levelings and other filters [NAJ 06]. This staple of morphological algorithms, the watershed transform, can also be seen as a topological transform [BER 05, COU 05]. For details on the topological watershed, see Chapter **??**.

1.5.2.6. *Morphological filtering algorithms*

Morphological filtering algorithms form an interesting class by themselves. As a starting point, useful literature on morphological filtering includes the two books by SERRA [SER 82, SER 88], an article on the theory of morphological filtering by SERRA and VINCENT [SER 92] and the articles by HEIJMANS and RONSE [HEI 90, RON 91]. A more introductory article by HEIJMANS is [HEI 96]. None of these articles discuss algorithmic aspects, which are nonetheless essential. The following is an incomplete but illustrative list of some problems studied in mathematical morphology.

1) *Fast erosions and dilations*. The topic of fast implementations of basic morphological operator has been studied by many authors. In spite of this, many libraries of mathematical morphology software (including well-known and expensive ones) compute a min or max filter on a window using $O(MN)$ comparisons, with M the number of pixels in the image and N the number of pixels in the window. It is often possible to decompose structuring elements into more readily computable sub-parts [XU 91]. Among the most commonly use SE are the regular convex polygons in $2D$. These can easily be decomposed into operations using line segments. A significant achievement by the community has been to propose increasingly efficient algorithms to compute the basic morphological operators in arbitrary $1D$ segment windows, including arbitrary orientation [HER 92, BRE 93, GIL 02, Van 05]. As a result, the computation of all four basic morphological operators with convex regular polygonal windows can be achieved in constant time with respect to N in $2D$. Note that at the time of writing, an equivalent result in $3D$ or more is still an open problem, except for some particular cases.

Regarding arbitrary structuring elements in nD, there exists an algorithm with complexity $O(\sqrt[\frac{n}{n-1}]{N}M)$ [VAN 96]. A faster algorithm for $2D$ but extensible to more has been proposed [URB 08]. Several algorithms have been proposed in the binary case [JI 89, VIN 91b], with complexity asymptotically linear with respect to M. The FFT-based algorithm mentioned in section **??** has complexity $M \log M$.

2) *Algebraic openings and thinnings* . Filtering in mathematical morphology tends to rely more on openings and closings than erosions and dilations. It is common to define a notion of opening or closing that is not directly related to that of structuring element, but is rather based on the concepts of *attribute* and connectivity [CRE 93, CRE 97, HEI 99]. These ideas are close to the notion of reconstruction seen in this chapter, and were also presented in the introductory chapter of this book **??**.

Thanks to connected and attribute filtering, many very effective operators were proposed in the last decade. Historically, from the algorithmic point of view, the first implementation of a connected filter is due to VINCENT [VIN 92, VIN 93a, VIN 94], with the area filter. The general notion was extended to attributes [BRE 96] that are not necessarily increasing, leading to operators that were no longer openings or closings, but *algebraic thinnings*, however using very similar principles. An efficient implementation was proposed in [MEI 02], followed by a generalization using the component tree and the Union-Find in [GÉR 05]. More recently, the notion of connectivity was extended to *hyper-connectivity* [WIL 06] to account for overlaps.

Path connectivity is also both a topological and a connectivity notion. By adding constraints to acceptable paths, from straight line segments [SOI 01] to more flexible paths [HEI 05, TAL 07], it is possible to enable the filtering of notoriously difficult thin objects in various applications [VAL 09b, VAL 09a].

3) *Spatially variant filtering*. More recently, efficient operators using filtering by non translation-invariant (or *spatially variant*) filters have been proposed. From the theoretical point of view, spatially variant filters have been known at least since SERRA [SER 82], but were recently given some more theoretical treatment [CHA 94, BOU 08a, BOU 08b]. This kind of filtering method is adaptive in the sense that a different structuring element is used at each point, depending on the local content of the image (for instance depending on the orientation, perspective, or texture) [LER 06, VER 08]. These filters can be effective in the context of inverse filtering, for thin feature extrapolation [TAN 09a, TAN 09b].

4) *Extension to nD*. Mathematical morphology is, from the theoretical point of view, largely dimensional-agnostic, meaning most operators can be defined irrespective of the dimension of the underlying space [GES 90, GRA 93]. However there are some practical difficulties as dimension increases. For instance, from the geometrical and topological point of view, while the hexagonal grid is a useful vertices arrangement in $2D$ that is naturally relatively isotropic and self-dual with respect to connectivity, no such arrangement exist in $3D$, and little is known of higher dimensions. From the direction sampling point of view, which is often used in orientation-based filtering, it is possible to sample the $2D$ plane directionally in such a way that is both regular and of arbitrary resolution (say every degree or more or less). This is impossible in $3D$ and more : a result which is known since PLATON and EUCLID [HEA 56]. Of course $3D$ (and more) filtering requires more resources, but thanks to recent advances in sensors, instruments and computers, it has become increasingly common and important. Application fields include medical imaging, materials science, biological and bio-molecular imaging.

1.6. Conclusions

In this chapter, we have sought to express the distance that exists between the mathematical formulation of an operator and its actual algorithmic translation. From a simple but representative example, we have also shown that in general there does not exist a single best way to express the implementation of an operator, but several, for which characteristics can be markedly different.

Algorithmic research, dedicated to mathematical morphology or not, remains a wide open field. As time progresses, increasingly sophisticated operators are being proposed, with correspondingly demanding computation loads. Together with the ever increasing size and complexity of the data itself. This can only mean that it becomes increasingly more important to devote sufficient time and resources to the development of efficient implementations of image analysis operators, whether this implementation be in hardware or software.

We can also anticipate a few fresh challenges on the algorithmic frontier :

First, external and internal observers (for instance users, but also article reviewers), have noticed increasing difficulties in reproducing methods and results presented in the scientific community. From a scientific article, the way leading to an actual piece of working code can be very long. At the understanding of the proposed operators and algorithms can be added the difficult and painful programming and debugging tasks. As a direct consequence, a loss of information capital and of knowledge can be observed. Many, if not most, solutions proposed in the literature are simply abandoned or ignored, and few articles propose comparison with a significant number of solutions. In our opinion, the morphological community should make the effort to endow the public at large with a working library of computer code implementing its efforts. This could take the form of a mutual, open platform for code repository.

A second challenge concerns the implementation of algorithms. Algorithms are by their nature abstract. Indeed in this chapter we have kept an abstract presentation as much as possible, reflecting the fact that they might work just as well on $1D$ signal as on $3D$ volume data, irrespective of sampling, grid and topology issues, unless specified. Unfortunately, the actual translation of algorithms to code is almost inevitably accompanied by a loss of generality : such library of code will only work on $2D$, gray-level, square grid images. Another will be devoted to satellite images, yet another to 3D medical volumes, and so on. Most libraries do not accept arbitrary-shaped structuring elements for instance. Even a “simple” dilation as given by algorithm (3) on Fig. 1.2 become, once implemented, a dilation restricted to a limited number of cases. We note, however, that generic solutions exist, allowing users to apply algorithms to vastly different datasets without necessarily sacrificing efficiency [LEV 09].

A third challenge concerns community effort. It is increasingly understood, that to be acceptable during and after publication, to be reviewed effectively and to be cited, an algorithm description should be accompanied by an implementation, freely accessible to the researcher or individual user. Indeed, re-implementation efforts are usually simply duplicated work. In other communities, like computer vision, discrete geometry or computational geometry, active repository of code exist. This is not yet the case with mathematical morphology, although various attempts have been made, and it is the personal belief of the authors of this chapter that this has hindered the adoption of many effective algorithm in the larger community of researchers and users of image analysis. Of course, making code freely available does not always sit nicely with intellectual property demands of funding agencies and institutions, so this is yet another challenge for which solution have been proposed, such as dual licensing.

Finally, a last challenge is in the evolution of computer architectures. We are now in the era of generalized multi-processors and cheaply available massively parallel co-processors and clusters. This means that the tools of image analysis and in particular algorithms, must yet again adapt themselves to this changing environment.

Bibliographie

[ALL 07] ALLÈNE C., AUDIBERT J.-Y., COUPRIE M., COUSTY J., KERIVEN R., « Some links between min-cuts, optimal spanning forests and watersheds », BANON G. J. F., BARRERA J., BRAGA-NETO U. D. M., HIRATA N. S. T., Eds., *Proceedings*, vol. 1, São José dos Campos, Universidade de São Paulo (USP), Instituto Nacional de Pesquisas Espaciais (INPE), p. 253–264, October 10–13, 2007 2007.

[APP 06] APPLETON B., TALBOT H., « Globally Minimal Surfaces by Continuous Maximal Flows », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, n°1, p. 106-118, 2006.

[BER 05] BERTRAND G., « On topological watersheds », *Journal of Mathematical Imaging and Vision*, vol. 22, n°2-3, p. 217-230, mai 2005.

[BER 07] BERTRAND G., « On critical kernels », *Comptes Rendus de l'Académie des Sciences, Série Math.*, vol. I, n°345, p. 363-367, 2007.

[BEU 79] BEUCHER S., LANTUÉJOUL C., Sur l'utilisation de la ligne de partage des eaux en détection de contours, Rapport n°N-598, Ecole des Mines de Paris, mai 1979.

[BIL 92] BILODEAU M., Architecture logicielle pour processeur de morphologie mathématique, Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1992.

[BOR 84] BORGEFORS G., « Distance transformations in arbitrary dimensions », *Computer Vision, Graphics, and Image Processing*, vol. 27, p. 321-345, 1984.

[BOU 08a] BOUAYNAYA N., CHARIF-CHEFCHAOUNI M., SCHONFELD D., « Theoretical Foundations of Spatially-Variant Mathematical Morphology Part I : Binary Images », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, n°5, p. 823–836, IEEE Computer Society, 2008.

[BOU 08b] BOUAYNAYA N., SCHONFELD D., « Theoretical Foundations of Spatially-Variant Mathematical Morphology Part II : Gray-Level Images », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, n°5, p. 837–850, IEEE Computer Society, 2008.

[BOY 01] BOYKOV Y., VEKSLER O., ZABIH R., « Fast Approximate Energy Minimization via Graph Cuts », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, n°11, p. 1222-1239, 2001.

[BOY 04] BOYKOV Y., KOLMOGOROV V., « An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, n°9, p. 1124-1137, septembre 2004.

[BRE 93] BREEN E., SOILLE P., « Generalization of van Herk recursive erosion/dilation algorithm to lines at arbitrary angles », FUNG K., GINIGE A., Eds., *Proc. DICTA'93 : Digital Image Computing : Techniques and Applications*, Sydney, Australian Pattern Recognition Society, p. 549-555, décembre 1993.

[BRE 94] BREEN E., MONRO D., « An evaluation of priority queues for mathematical morphology », SERRA J., SOILLE P., Eds., *Mathematical Morphology and its Applications to Image Processing*, p. 249–256, Kluwer Academic Publishers, 1994.

[BRE 96] BREEN E., JONES R., « Attribute openings, thinnings, and granulometries », *Computer Vision and Image Understanding*, vol. 64, n°3, p. 377-389, 1996.

[CHA 94] CHARIF-CHEFCHAOUNI M., SCHONFELD D., « Spatially-variant mathematical morphology », *IEEE International Conference on Image Processing (ICIP)*, vol. 2, p. 555-559 vol.2, novembre 1994.

[COR 09] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C., *Introduction to Algorithms*, The MIT Press, 3rd édition, 2009.

[COU 05] COUPRIE M., NAJMAN L., BERTRAND G., « Quasi-linear algorithms for the topological watershed », *Journal of Mathematical Imaging and Vision*, vol. 22, n°2-3, p. 231-249, mai 2005, Special issue on Mathematical Morphology.

[COU 07] COUSTY J., BERTRAND G., NAJMAN L., COUPRIE M., Watersheds, minimum spanning forests, and the drop of water principle, Rapport n°IGM2007-01, Université de Marne-la-Vallée, 2007, submitted.

[COU 09] COUPRIE C., GRADY L., NAJMAN L., TALBOT H., « Power watersheds, A new segmentation framework extending graph cuts, random walker and optimal spanning forest », 2009, submitted.

[COU 10] COUSTY J., BERTRAND G., COUPRIE M., NAJMAN L., « Watersheds and collapses in pseudomanifolds of arbitrary dimension », *In preparation*, 2010.

[CRE 93] CRESPO J., SERRA J., SCHAFER R., « Image segmentation using connected filters », SERRA J., SALEMBIER P., Eds., *Mathematical morphology and its applications to signal processing*, Universitat Politècnica de Catalunya, Barcelone, Espagne, p. 52-57, mai 1993.

[CRE 97] CRESPO J., SCHAFER R. W., « Locality and Adjacency Stability Constraints for Morphological Connected Operators », *Journal of Mathematical Imaging and Vision*, vol. 7, p. 85-102, 1997.

[DAN 80] DANIELSSON P.-E., « Euclidean distance mapping », *Computer Graphics and Image Processing*, vol. 14, p. 227-248, 1980.

[DIJ 59] DIJKSTRA E., « A note on two problems in connexion with graphs », *Numerische Mathematik*, vol. 1, p. 269-271, 1959.

[FAL 04] FALCAO A. X., STOLFI J., DE ALENCAR LOTUFO R., « The Image Foresting Transform : Theory, Algorithms, and Applications », *IEEE Transactions on Pattern Analysis and*

Machine Intelligence, vol. 26, n°1, p. 19–29, IEEE Computer Society, 2004.

[GÉR 05] GÉRAUD T., « Ruminations on Tarjan's Union-Find algorithm and connected operators », *Mathematical Morphology : 40 Years On, Proceedings of the International Symposium (ISMM)*, vol. 30 de *Computational Imaging and Vision*, Paris, France, Kluwer, p. 105–116, April 2005.

[GES 90] GESBERT S., HOWARD V., JEULIN D., MEYER F., « The use of basic morphological operations for 3D biological image analysis », *Trans. Roy. Microsc. Soc.*, vol. 1, London, p. 293-296, juillet 1990.

[GIL 02] GIL J., KIMMEL R., « Efficient dilation, erosion, opening and closing algorithms », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, n°12, p. 1606–1617, 2002.

[GOO 04] GOODMAN J. E., O'ROURKE J., *Handbook of Discrete and Computational Geometry*, Chapman & Hall / CRC, Boca Raton, 2nd édition, 2004.

[GRA 93] GRATIN C., De la reprśentation des images au traitement morphologique d'images tridimensionnelles, PhD thesis, Ecole des Mines de Paris, 1993.

[GRA 06] GRADY L., « Random Walks for Image Segmentation », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, n°11, p. 1768-1783, novembre 2006.

[HEA 56] HEATH T. L., *The thirteen books of Euclid's elements*, vol. (3 volumes), Dover Publications, 2nd édition, 1956, ISBN 0-486-60088-2, 0-486-60089-0, 0-486-60090-4.

[HEI 90] HEIJMANS H., RONSE C., « The algebraic basis of mathematical morphology : I. Dilations and erosions », *Computer Vision, Graphics, and Image Processing*, vol. 50, p. 245–295, 1990.

[HEI 96] HEIJMANS H., « Morphological Filters for Dummies », MARAGOS P., Ed., *Mathematical morphology and its applications to Image and Signal processing*, Atlanta, GA, Kluwer, p. 127–138, May 1996, Proceedings of the 3rd International Symposium on Mathematical Morphology.

[HEI 99] HEIJMANS H., « Connected morphological operators for binary images », *Computer Vision and Image Understanding*, vol. 73, n°1, p. 99-120, 1999.

[HEI 00] HEIJMANS H., GOUTSIAS J., « Nonlinear multiresolution signal decomposition schemes—Part II : Morphological wavelets », *IEEE Transactions on Image Processing*, vol. 9, n°11, p. 1897-1913, novembre 2000.

[HEI 05] HEIJMANS H., BUCKLEY M., TALBOT H., « Path openings and closings », *Journal of Mathematical Imaging and Vision*, vol. 22, p. 107-119, 2005.

[HER 92] VAN HERK M., « A fast algorithm for local minimum and maximum filters on rectangular and octogonal kernels », *Pattern Recognition Letters*, vol. 13, p. 517-521, 1992.

[JAC 96] JACKWAY P., « Gradient watersheds in morphological scale-space », *IEEE Transactions on Image Processing*, vol. 5, n°6, p. 913-921, juin 1996.

[JI 89] JI L., PIPER J., TANG J.-Y., « Erosion and dilation of binary images by arbitrary structuring elements using interval coding », *Pattern Recognition Letters*, vol. 9, p. 201-209, 1989.

[JON 99] JONES R., « Connected filtering and segmentation using component trees », *Computer Vision and Image Understanding*, vol. 75, n˚3, p. 215-228, 1999.

[JOS 56] JOSEPH B. KRUSKAL J., « On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem », *Proceedings of the American Mathematical Society*, vol. 7, n˚1, p. 48–50, February 1956.

[KAS 98] KASS M., WITKIN A., TERZOPOULOS D., « Snakes : Active Contour Models », *International Journal of Computer Vision*, vol. 1, n˚4, p. 321-331, 1998.

[KLE 89] KLEIN J.-C., PEYRARD R., « PIMM1, an image processing ASIC based on Mathematical Morphology », *Proceedings of the Second annual IEEE ASIC Seminar and Exhibit*, p. 1–4, 1989.

[KLE 90] KLEIN J., COLLANGE F., BILODEAU M., « A bit plane architecture for an image processor implemented with P.L.C.A. gate array », SPRINGER, Ed., *Computer Vision, proceedings of ECCV 1990*, vol. 427 de *LNCS*, p. 33–49, 1990.

[KON 89] KONG T., ROSENFELD A., « Digital topology : Introduction and survey », *Computer Vision, Graphics, and Image Processing*, vol. 48, p. 357-393, 1989.

[KON 95] KONG T., « On topology preservation in 2-D and 3-D thinning », *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 9, n˚5, p. 813-844, 1995.

[L. 87] L. F. M., E. T. R., « Fibonacci heaps and their uses in improved network optimization algorithms », *Journal of the ACM*, vol. 34, n˚596-615, 1987.

[LEM 96] LEMONNIER F., Architecture électronique dédiée aux algorithmes rapides de segmentation basés sur la morphologie mathématique, Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1996.

[LER 06] LERALLUT R., DECENCIÈRE E., MEYER F., « Image filtering using morphological amoebas », *IVC*, vol. 25, n˚4, p. 395-404, 2006.

[LEV 09] LEVILLAIN R., GÉRAUD T., NAJMAN L., « Milena : Write Generic Morphological Algorithms Once, Run on Many Kinds of Images », WILKINSON M., ROERDINK J., Eds., *Mathematical Morphology, Proceedings of the 9th International Symposium (ISMM)*, vol. 5720 de *Lecture Notes in Computer Science*, Groningen, The Netherlands, Springer-Verlag, p. 295–306, August 2009.

[LEY 92] LEYMARIE F., LEVINE M., « Simulating the Grassfire Transform Using an Active Contour Model », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, n˚1, p. 56-75, IEEE Computer Society, 1992.

[MAN 00] MANZANERA A., Vision Artificielle Rétinienne, Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, 2000.

[MAN 02] MANZANERA A., BERNARD T., PRÊTEUX F., LONGUET B., « N-dimensional skeletonization : a unified mathematical framework », *Journal of Electronic Imaging*, vol. 11, n˚25, p. 25–37, SPIE, 2002.

[MAR 96] MARAGOS P., « Differential Morphology and Image-Processing », *IEEE Transactions on Image Processing*, vol. 5, n˚6, p. 922–937, 1996.

[MAT 02] MATHERON G., SERRA J., « The birth of mathematical morphology », TALBOT H., BEARE R., Eds., *Proceedings of VIth International Symposium on Mathematical Morphology*, Sydney, Australie, Commonwealth Scientific and Industrial Research Organisation, p. 1-16, avril 2002.

[MEI 02] MEIJSTER A., WILKINSON M., « A comparison of algorithms for connected set openings and closings », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, n°4, p. 484-494, 2002.

[MER 94] MERRIMAN B., BENCE J., OSHER S., « Motion of multiple junctions : a level set approach », *Journal of Computational Physics*, vol. 112, p. 334–363, 1994.

[MEY 90a] MEYER F., Algorithmes à base de files d'attente hiérarchique, Rapport n°NT-46/90/MM, Ecole des Mines de Paris, septembre 1990.

[MEY 90b] MEYER F., BEUCHER S., « Morphological segmentation », *Journal of Visual Communication and Image Representation*, vol. 1, n°1, p. 21-46, septembre 1990.

[MEY 91] MEYER F., « Un algorithme optimal de ligne de partage des eaux », *Reconnaissance des Formes et Intelligence Artificielle, 8e congrès*, Lyon-Villeurbanne, AFCET, p. 847-857, 1991.

[MEY 00] MEYER F., MARAGOS P., « Nonlinear scale-space representation with morphological levelings », *Journal of Visual Communication and Image Representation*, vol. 11, n°3, p. 245–265, 2000.

[MON 68] MONTANARI U., « A method for obtaining skeletons using a quasi-Euclidean distance », *Journal of the ACM*, vol. 15, n°4, p. 600-624, octobre 1968.

[NAJ 93] NAJMAN L., SCHMITT M., « Definitions and some properties of the watershed of a continuous function », SERRA J., SALEMBIER P., Eds., *Mathematical morphology and its applications to signal processing*, p. 76–81, 1993.

[NAJ 06] NAJMAN L., COUPRIE M., « Building the component tree in quasi-linear time », *IEEE Transactions on Image Processing*, vol. 15, n°11, p. 3531-3539, 2006.

[NGU 03] NGUYEN H. T., WORRING M., VAN DEN BOOMGAARD R., « Watersnakes : Energy-Driven Watershed Segmentation », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, n°3, p. 330-342, IEEE Computer Society, 2003.

[OSH 88] OSHER S., SETHIAN J., « Fronts propagating with curvature-dependent speed : Algorithms based on the Hamilton–Jacobi formulation », *J. Comput. Phys.*, vol. 79, n°1, p. 12–49, 1988.

[PRI 57] PRIM R., « Shortest connection networks and some generalisations », *Bell System Technical Journal*, vol. 36, 1957.

[RAG 92] RAGNEMALM I., « Fast erosion and dilation by contour processing and thresholding of distance maps », *Pattern Recognition Letters*, vol. 13, p. 161-166, 1992.

[RAG 93] RAGNEMALM I., « The Euclidean Distance Transform in Arbitrary Dimensions », *Pattern Recognition Letters*, vol. 14, n°11, p. 883 - 888, 1993.

[RÉV 91] RÉVEILLÈ J.-P., Géométrie discrète, calculs en nombres entiers et algorithmique, Thèse d'Etat, Université Louis Pasteur, Strasbourg, 1991.

[RON 91] RONSE C., HEIJMANS H., « The algebraic basis of mathematical morphology : II. Openings and closings », *Computer Vision, Graphics, and Image Processing : Image Understanding*, vol. 54, n˚1, p. 74-97, 1991.

[ROS 66] ROSENFELD A., PFALTZ J., « Sequential operations in digital picture processing », *Journal of the ACM*, vol. 13, n˚4, p. 471-494, 1966.

[SAL 92] SALEMBIER P., SERRA J., « Morphological Multiscale Image Segmentation », MARAGOS P., Ed., *Visual Communications and Image Processing*, vol. SPIE-1818, p. 620-631, 1992.

[SAP 93] SAPIRO G., KIMMEL R., SHAKED D., KIMIA B., BRUCKSTEIN A., « Implementing continuous-scale morphology via curve evolution », *Pattern Recognition*, vol. 26, n˚9, p. 1363-1372, 1993.

[SCH 89] SCHMITT M., Des algorithmes morphologiques à l'intelligence artificielle, Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, février 1989.

[SER 82] SERRA J., *Image analysis and mathematical morphology*, Academic Press, Londres, Royaume-Uni, 1982.

[SER 88] SERRA J., « Examples of structuring functions and their uses », SERRA J., Ed., *Image analysis and mathematical morphology. Volume 2 : Theoretical advances*, Chapitre 4, p. 71-99, Academic Press, 1988.

[SER 92] SERRA J., VINCENT L., « An overview of morphological filtering », *Circuits Systems Signal Process*, vol. 11, n˚1, p. 47-108, 1992.

[SET 96a] SETHIAN J., « A Fast Marching Level Set Method for Monotonically Advancing Fronts », *Proceedings of the National Academy of Sciences*, vol. 93(4), p. 1591–1595, 1996.

[SET 96b] SETHIAN J., *Level Set Methods : Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, 1996.

[SET 01] SETHIAN J., VLADIMIRSKY A., « Ordered upwind methods for static Hamilton-Jacobi equations », *Proceedings of the National Academy of Science U.S.A.*, vol. 98, n˚20, p. 11069-11074, 2001.

[SHI 92] SHIH F. Y., MITCHELL O. R., « A mathematical morphology approach to Euclidean distance transformation », *IEEE Transactions on Image Processing*, vol. 1, p. 197–204, 1992.

[SLE 85] SLEADOR D., TARJAN R., « Self-adjusting Binary Search Trees », *J. Assoc. Comp. Mach.*, vol. 32, p. 652-686, 1985.

[SOI 91] SOILLE P., « Spatial distributions from contour lines : an efficient methodology based on distance transformations », *Journal of Visual Communication and Image Representation*, vol. 2, n˚2, p. 138-150, juin 1991.

[SOI 01] SOILLE P., TALBOT H., « Directional morphological filtering », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, n˚11, p. 1313-1329, novembre 2001.

[STU 87] STUIK D. J., *A concise history of mathematics*, Dover, 4th édition, 1987.

[TAL 07] TALBOT H., APPLETON B., « Efficient complete and incomplete paths openings and closings », *Image and Vision Computing*, vol. 25, n°4, p. 416–425, April 2007.

[TAN 09a] TANKYEVYCH O., TALBOT H., DOKLADAL P., PASSAT N., « Direction-Adaptive Grey-level Morphology. Application to 3D Vascular Brain Imaging », *IEEE International Conference on Image Processing (ICIP)*, Cairo, Egypt, IEEE, p. 2261-2264, 2009.

[TAN 09b] TANKYEVYCH O., TALBOT H., DOKLADAL P., PASSAT N., « Spatially-variant morpho-Hessian filter : efficient implementation and application », *Proceedings of ISMM 2009*, Groningen, The Netherlands, 2009, Accepted.

[TAR 75] TARJAN R., « Efficiency of a good but not linear set union algorithm », *Journal of the ACM*, vol. 22, n°2, p. 215–225, avril 1975.

[TUR 36] TURING A., « On computable numbers, with an application to the Entscheidungsproblem », *Proceedings of the London Mathematical Society*, vol. 42 de 2, November 1936, Reprinted in the Undecidable, pages 115–154.

[URB 08] URBACH E., WILKINSON M., « Efficient 2-D Grayscale Morphological Transformations with Arbitrary Flat Structuring Elements », *IEEE Transactions on Image Processing*, vol. 17, n°1, p. 1-8, January 2008.

[VAC 01] VACHIER C., « Morphological scale-space analysis and feature extraction », *Proceedings of International Conference on Image Processing*, vol. 3, p. 676-679, 2001.

[VAL 09a] VALERO S., CHANUSSOT J., BENEDIKTSSON J., TALBOT H., WASKE B., « Advanced Directional mathematical Morphology for the Detection of the Road Network in very high resolution images », *Pattern Recognition*, 2009, In print.

[VAL 09b] VALERO S., CHANUSSOT J., BENEDIKTSSON J., TALBOT H., WASKE B., « Directional Mathematical Morphology For the detection of the road network in very high resolution remote sensing images », *Proceedings of ICIP 2009*, Cairo, Egypt, 2009, accepted.

[VAN 96] VAN DROOGENBROECK M., TALBOT H., « Fast computation of morphological operations with arbitrary structuring elements », *Pattern Recognition Letters*, vol. 17, n°14, p. 1451–1460, 1996.

[Van 05] VAN DROOGENBROECK M., BUCKLEY M., « Morphological erosions and openings : fast algorithms based on anchors », *Journal of Mathematical Imaging and Vision, Special Issue on Mathematical Morphology after 40 Years*, vol. 22, n°2-3, p. 121-142, May 2005.

[VER 08] VERDÚ-MONEDERO R., ANGULO J., « Spatially-Variant Directional Mathematical Morphology Operators Based on a Diffused Average Squared Gradient Field », *Advanced Concepts for Intelligent Vision Systems*, p. 542–553, 2008.

[VIN 90] VINCENT L., Algorithmes morphologiques à base de files d'attente et de lacets. Extension aux graphes, Thèse de doctorat, Ecole Nationale Supérieure des Mines de Paris, 1990.

[VIN 91a] VINCENT L., « Efficient Computation of Various Types of Skeletons », LOEW M., Ed., *Medical Imaging V*, vol. 1445, San Jose, CA, Society of Photo-Instrumentation Engineers, p. 297-311, 1991.

[VIN 91b] VINCENT L., « Morphological transformations of binary images with arbitrary structuring elements », *Signal Processing*, vol. 22, n°1, p. 3-23, janvier 1991.

[VIN 91c] VINCENT L., SOILLE P., « Watersheds in digital spaces : an efficient algorithm based on immersion simulations », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, n°6, p. 583-598, juin 1991.

[VIN 92] VINCENT L., « Morphological area openings and closings for greyscale images », *Proc. Shape in Picture '92, NATO Workshop*, Driebergen, The Netherlands, Springer-Verlag, septembre 1992.

[VIN 93a] VINCENT L., « Grayscale area openings and closings, their efficient implementation and applications », SERRA J., SALEMBIER P., Eds., *Proc. EURASIP workshop on Mathematical morphology and its applications to signal processing*, Barcelone, Espagne, p. 22-27, mai 1993.

[VIN 93b] VINCENT L., « Morphological grayscale reconstruction in image analysis : applications and efficient algorithms », *IEEE Transactions on Image Processing*, vol. 2, n°2, p. 176–201, avril 1993.

[VIN 94] VINCENT L., « Morphological area openings and closings for greyscale images », O. Y.-L., TOET A., FOSTER D., HEIJMANS H., MEER P., Eds., *Shape in Picture : Mathematical Description of Shape in Grey-Level Images*, vol. 126 de *NATO ASI Series F*, Springer-Verlag, p. 197-208, 1994.

[WIL 06] WILKINSON M. H., « Attribute-space connectivity and connected filters », *IVC*, vol. 25, n°4, p. 426-435, 2006.

[XU 91] XU J., « Decomposition of Convex Polygonal Morphological Structuring Elements into Neighborhood Subsets », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, n°2, p. 153–162, 1991.

[ZAM 80] ZAMPERONI P., « Dilatation und Erosion von konturcodierten Binärbildern », *Microscopica Acta*, vol. Suppl. 4, p. 245-249, 1980.

Index

Fiche pour le service de fabrication

Auteurs :

Édité par Laurent NAJMAN et Hugues TALBOT

Titre du livre :

Morphologie Mathématique en anglais

Titre abrégé :

Morphologie Mathématique Anglais

Date de cette version :

8 septembre 2010

Contact :

– téléphone : 04 92 94 27 48
– télécopie : 04 92 94 28 96
– Mél : rr@unice.fr

Logiciel pour la composition :

– LaTeX, avec la classe `ouvrage-hermes.cls`,
– version 1.3, 2004/10/17.
– traité (option treatise) : Oui *(chapitres avec différents auteurs)*
– livre en anglais (option english) : Non *(par défaut en français)*
– tracé des limites de page (option cropmarks) : Non *(par défaut)*
– suppression des en-têtes de page (option empty) : Non *(par défaut)*
– impression des pages blanches (option allpages) : Oui
– césures actives : voir la coupure du mot signal dans le fichier .log