Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations

Luc Vincent and Pierre Soille

Abstract— **In this paper, a fast and flexible algorithm for computing watersheds in digital grayscale images is introduced. A review of watersheds and related notion is first presented, and the major methods to determine watersheds are discussed. The present algorithm is based on an immersion process analogy, in which the flooding of the water in the picture is efficiently simulated using a queue of pixels. It is described in detail and provided in a pseudo C language. We prove the accuracy of this algorithm is superior to that of the existing implementations. Furthermore, it is shown that its adaptation to any kind of digital grid and its generalization to n-dimensional images and even to graphs are straightforward. In addition, its strongest point is that it is faster than any other watershed algorithm. Applications of this algorithm with regard to picture segmentation are presented for MR imagery and for digital elevation models. An example of 3-D watershed is also provided. Lastly, some ideas are given on how to solve complex segmentation tasks using watersheds on graphs.**

Index Terms—**Algorithm, digital image, FIFO structure, graph, grid, mathematical morphology, picture segmentation, watersheds.**

I. Introduction

WATERSHEDS are one of the classics in the field of topography. Everybody has heard for example about the *great divide*, this particular line which separates the U.S.A. into two regions. A drop of water falling on one side of this line flows down until it reaches the Atlantic Ocean, whereas a drop falling on the other side flows down to the Pacific Ocean. As we shall see in further detail later, this great divide constitutes a typical example of a *watershed line*. The two regions it separates are called the *catchment basins* of the Atlantic and the Pacific Oceans, respectively. The two Oceans are the *minima* associated with these catchment basins.

Now, in the field of image processing and more particularly in Mathematical Morphology (MM) [30], [40], [45], grayscale pictures are often considered as topographic reliefs. In the topographic representation of a given image I, the numerical value (i.e., the gray tone) of each pixel stands for the elevation at this point. Such a representation is extremely useful, since it first allows one to better appreciate the effect of a given transformation on the image under study. We know for example that an opening removes some peaks and crest lines, whereas a closing tends to fill in basins and valleys. Furthermore, thanks to this representation, such notions as minima, catchment basins and watersheds can be well defined for grayscale images. As we shall see throughout this paper, they turn out to be extremely important and useful.

Quite naturally, the first algorithms for computing watersheds are found in the field of topography. Topographic surfaces are numerically handled through *digital elevation models* (DEM's). These are arrays of numbers that represent the spatial distribution of terrain altitudes. The most commonly used data structure for DEM's is the regular square grid in which available elevations are equally spaced in two orthogonal directions. Automated watershed extraction from DEM's has received increasing attention in the past few years [9], [36], [28], [1]. The first step of most published algorithms is a parallel procedure performing local operations defined on a 3×3 window. This allows one to extract potential dividing pixels. In a second step, the extracted pixels are connected into geomorphological networks. However, the very local approach of the first step and the lack of objective rules to perform the second one usually lead to poor results [43].

Meanwhile and apart from these researches in digital topography, the above notions were studied in the field of image processing. The introduction of the *watershed transformation* as a morphological tool is due to H. Digabel and Ch. Lantuéjoul [11]. Their data were piles of binary images representing successive thresholds of a bituminous surface's relief whose drainability was to be studied. Later, a joint work of C. Lantuéjoul and S. Beucher led to the "inversion" of this original algorithm in order to extend it the more general framework of grayscale images [3], [4]. Watersheds were then approached theoretically by F. Maisonneuve [27] and used in numerous grayscale segmentation problems. They are currently being studied from theoretical, practical, and algorithmic points of view. When combined with other morphological tools, the watershed transformation is at the basis of extremely powerful segmentation procedures [49].

Extracting watersheds from digital pictures is far from being an easy task. As available algorithms for computing the watershed transformation are either excessively slow or inaccurate (see Sections II-D and IV-B), the tremendous practical interest of this transformation is not as obvious as it should be. The purpose of the present paper is to introduce an efficient and completely new implementation of watersheds (see also [50]). Roughly speaking, it is based on a sorting of the pixels in the increasing order of their gray values, and on fast breadth-first

Manuscript received February 4, 1990; revised January 24, 1991. Recommended for acceptance by S. L. Tanimoto.

L. Vincent is with the Division of Applied Sciences, Harvard University, Pierce Hall, Cambridge, MA 02138, on leave from the Center for Mathematical Morphology, Ecole des Mines de Paris, 35 Rue Saint-Honore, 77305 Fontainebleau Cedex, France.

P. Soille is with the Center for Mathematical Morphology, Ecole des Mines de Paris, 35 Rue Saint-Honore, 77305 Fontainebleau Cedex, France.

IEEE Log Number 9100083.

0162–8828/91/0600–0583$01.00 © 1991 IEEE

scannings of the plateaus enabled by a *first-in-first-out* type data structure. Our algorithm turns out to be hundreds of times faster than some classical ones on conventional computers (see Section IV). It is also more accurate and behaves well in all the particular pixel configurations where many algorithms produce incorrect results (see Section IV-B). Furthermore, the present algorithm is very general: its adaptation to any kind of underlying grid (4-, 6-, 8-connectivity . . .) is straightforward, and it can be fairly easily extended to n-dimensional images and even to graphs.

For the sake of clarity, Sections II and III only deal with watersheds for two-dimensional pictures. After some definitions concerning the objects which are considered in this paper, some reminders about watersheds and related notions are given in Section II. In particular, two different definitions of catchment basins and watersheds are brought to the fore. The existing algorithms for determining watersheds in digital images are then reviewed and discussed. Section III is devoted to the introduction of our implementation, here decomposed into two steps: an initial sorting and a flooding step. The emphasis is put on the accuracy of the results. The efficiency and advantages of our algorithm are then discussed in Section IV and opposed to the existing implementations. The rest of the paper is concerned with some of its possible applications: its interest with respect to picture segmentation is proved and illustrated on an example taken from the field of MR imagery. The algorithm is then applied to digital elevation models. Lastly, it is extended to other digital spaces: three-dimensional images are first considered and second, the graph version of our algorithm is used for the hierarchical description and segmentation of grayscale images.

II. Definition and Computation of Watersheds

A. Basic Definitions

It may seem easy to define watersheds on digital pictures, since this notion is a quite natural one. However, when looking closer at it, it turns out that there exist many particular cases, so that this definition task must be achieved very carefully.

Let us consider a two-dimensional grayscale picture I whose definition domain is denoted $D_I \subset \mathbb{Z}^2$. I is supposed to take discrete (gray) values in a given range $[0, N]$, N being an arbitrary positive integer:

$$I\begin{pmatrix} D_I \subset \mathbb{Z}^2 & \rightarrow & \{0,1,\cdots,N\} \\ p & \mapsto & I(p). \end{pmatrix} \tag{1}$$

In the following, we equally consider grayscale images as numerical functions or as topographic reliefs.

Let G denote the underlying digital grid, which can be of any type: a square grid in four or eight connectivity, or a hexagonal grid in six connectivity. G is a subset of $\mathbb{Z}^2 \times \mathbb{Z}^2$.

Definition 1: A path P of length l between two pixels p and q in picture I is a $(l+1)$-tuple of pixels $(p_0, p_1, \cdots, p_{l-1}, p_l)$ such that $p_0 = p$, $p_l = q$, and $\forall\, i \in [1, l]$, $(p_{i-1}, p_i) \in G$.

In the following, we denote $l(P)$ the length of a given path P. We also denote $N_G(p)$ the set of the neighbors of a pixel p, with respect to G :, $N_G(p) = \{p' \in \mathbb{Z}^2, (p, p') \in G\}$.

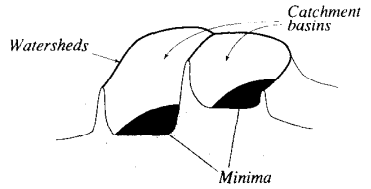


Fig. 1. Minima, catchment basins, and watersheds.

Before introducing watersheds, we need to recall the notion of *minimum* (see Fig. 1).

Definition 2: A minimum M of I at altitude h is a connected plateau of pixels with the value h from which it is impossible to reach a point of lower altitude without having to climb:

$$\begin{aligned} &\forall p \in \quad \forall q \notin M, \quad \text{such that } I_q \leq I(p), \\ &\forall P = (p_0, p_1, \ldots p_l) \quad \text{such that } p_0 = p \text{ and } p_l = q, \\ &\exists i \in [1, l] \quad \text{such that} \quad I(p_i) > I(p_0). \end{aligned} \tag{2}$$

A minimum is thus a connected and iso-intensive area where the gray level is strictly darker than on the neighboring pixels (the darker the pixel, the lower its value or elevation). These extrema are often referred to as *regional* ones [40], as opposed to the *local* ones.

We can now proceed with the definition of *catchment basins* and *watersheds,* which is done below using two different approaches.

B. Definition in Terms of Steepest Slope Lines

Definition 3 (Catchment basin, first definition): Let I be a grayscale image. The catchment basin $C(M)$ associated with a minimum M is the set of pixels p of D_I such that a water drop falling at p flows down along the relief, following a certain descending path called the downstream of p [27], and eventually reaches M.

The lines which separate different catchment basins build what is called the *watersheds* (or *dividing lines* for some authors) of I (see Fig. 1). The problem of the thickness of these lines is discussed in Sections II-C and III-C.

Notice that the catchment basins of an image I correspond to the *influence zones* of its minima. In this sense, we have a close relation between the binary *skeleton by influence zones* [23] and the watersheds. As we shall see in Sections IV-B and V-B, the notion of catchment basin—like that of minimum—is not a local one: in many cases, no local consideration can allow one to decide whether two pixels belong to the same catchment basin or not.

For real, continuous, derivable and *lower-complete* functions (the only possible plateaus of these functions are their minima), the direction of the flow path at any point is defined almost everywhere by the opposite of the gradient's azimuth at this point (Some theoretical problems with this definition are discussed in [40, p. 446]). On the contrary, when dealing with digital functions, there exists no rule to set up the path a drop of water would follow [27]. Therefore, this intuitive approach to watersheds is not well suited to practical implementations:

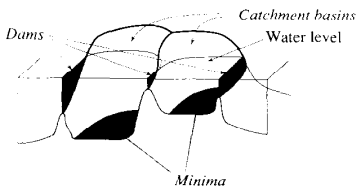


Fig. 2. Building dams at the places where the water coming from two different minima would merge.

in fact, we will see in Section II-D that the algorithms relying on it yield biased results in some cases. The definitions given in the next section are more suited to the formalization of catchment basins and watersheds in digital spaces, and are more oriented towards algorithm design. Thus, from now on, we shall deal only with digital spaces.

C. Definition by "Immersion"

The second approach for introducing watersheds [3] can be considered as an *algorithmic definition,* and is more suited to practical implementations. The algorithm we introduce in Section III relies on the present definition. By analogy, we can figure that we have pierced holes in each regional minimum of I, this picture being regarded as a (topographic) surface. We then slowly immerse our surface into a lake. Starting from the minima of lowest altitude, the water will progressively fill up the different catchment basins of I. Now, at each pixel where the water coming from two different minima would merge, we build a "dam" (see Fig. 2). At the end of this immersion procedure, each minimum is completely surrounded by dams, which delimit its associated catchment basin. The whole set of dams which has been built thus provides a tessellation of I in its different catchment basins. These dams correspond to the watersheds of I.

Let us express this immersion process more formally: I being the grayscale image under study, denote $h_{\min}$ the smallest value taken by I on its domain D_I. Similarly, denote $h_{\max}$ the largest value taken by I on D_I. In the following, $T_h(I)$ stands for the threshold of I at level h:

$$T_h(I) = \{p \in D_I, I(p) \leq h\}. \tag{3}$$

We also denote $C(M)$ the catchment basin associated with a minimum M and $C_h(M)$ the subset of this catchment basin made of the points having an altitude smaller or equal to h:

$$C_h(M) = \{p \in C(M), I(p) \leq h\} = C(M) \cap T_h(I). \tag{4}$$

As concerns the minima of I, $\min_h(I)$ refers to the set of points belonging to the minima at altitude h.

We now need to recall the definitions of the *geodesic distance* [24] and of the *geodesic influence zones.* Let A be a set which is first supposed to be simply connected.

Definition 4: The geodesic distance $d_A(x, y)$ between two pixels x and y in A is the infimum of the length of the paths which join x and y and are totally included in A:

$$d_A(x, y) = \inf\{l(P), P \text{ path between } x \text{ and } y \text{ which is totally included in } A\}. \tag{5}$$

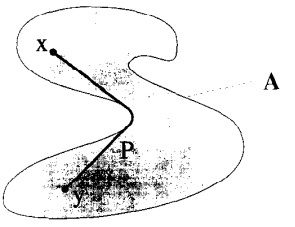


Fig. 3. The geodesic distance between x and y inside A is the infimum of the length of the paths between these two points which are totally included in A.

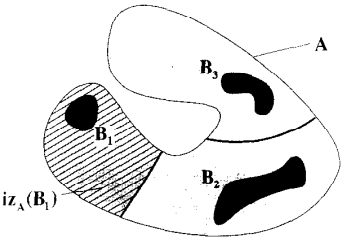


Fig. 4. Geodesic influence zone of connected component B_1 inside set A.

This definition is illustrated in Fig. 3.

Suppose now that A contains a set B made of several connected components $B_1, B_2, \cdots, B_k$.

Definition 5: The geodesic influence zone $iz_A(B_i)$ of a connected component B_i of B in A is the locus of the points of A whose geodesic distance to B_i is smaller than their geodesic distance to any other component of B:

$$iz_A(B_i) = \{p \in A, \forall j \in [1, k]/\{i\}, d_A(p, B_i) < d_A(p, B_j)\}. \tag{6}$$

This concept is illustrated in Fig. 4. Those points of A which do not belong to any geodesic influence zone constitute the *skeleton by influence zones (SKIZ)* of B inside A, denoted $\mathrm{SKIZ}_A(B)$:

$$\mathrm{SKIZ}_A(B) = A/IZ_A(B) \quad \text{with} \quad IZ_A(B) = \bigcup_{i \in [1;k]} iz_A(B_i). \tag{7}$$

According to this digital definition, the geodesic SKIZ of B in A does not necessarily separate the different geodesic influence zones. Indeed, due to parity problems, it is often made of *disconnected* lines. Moreover the digital SKIZ may sometimes be a "thick" one, since the set of the pixels which are equally distant from two connected components may well be very thick, as illustrated by Fig. 9. In Section III-C, we go back to these subtleties, but we do not take these problems into account here: we suppose that the geodesic SKIZ is always made of lines having one pixel thickness and thus separating the different geodesic influence zones. Note that the algorithm introduced in Section III makes use of a labeling of the influence zones and catchment basins, which allows us to avoid parity problems. The above definitions easily extend to the case where A is not simply connected, nor even connected at all.

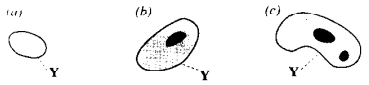


Fig. 5. The three possible inclusion relations between Y and $Y \cap X_{h_{\min}}$.

To simulate the immersion procedure described above, we start from the set $T_{h_{\min}}(I)$, the points of which being those first reached by the water. These points constitute the starting set of our recursion. We thus put

$$X_{h_{\min}} = T_{h_{\min}}(I). \tag{8}$$

$X_{h_{\min}}$ is made of the points of I which belong to the minima of lowest altitude. Let us now consider the threshold of I at level $h_{\min}+1$, i.e., $T_{h_{\min}+1}(I)$. Obviously, $X_{h_{\min}} \subseteq T_{h_{\min}+1}(I)$. Now, Y being one of the connected components of $T_{h_{\min}+1}(I)$, there are three possible relations of inclusion between Y and $Y \cap X_{h_{\min}}$:

1) $Y \cap X_{h_{\min}} = \emptyset$: in this case, Y is obviously a new minimum of I. Indeed, according to the definitions in Section II-A, Y is a plateau at level $h_{\min}+1$, since

$$\forall\, p \in Y, \quad \begin{cases} p \notin X_{h_{\min}} & \Rightarrow\ I(p) \geq h_{\min}+1 \\ p \in Y & \Rightarrow\ I(p) \leq h_{\min}+1. \end{cases}$$

Moreover, all the surrounding pixels do not belong to $T_{h_{\min}+1}(I)$ and have therefore a gray level strictly greater than $h_{\min}+1$. The minimum thus discovered is "pierced," hence its corresponding catchment basin will be progressively filled up with water.

2) $Y \cap X_{h_{\min}} \neq \emptyset$ and is connected: in this case, Y corresponds exactly to the pixels belonging to the catchment basin associated with the minimum $Y \cap X_{h_{\min}}$ and having a gray level lower or equal to $h_{\min}+1$:

$$Y = C_{h_{\min}+1}(Y \cap X_{h_{\min}}). \tag{9}$$

3) $Y \cap X_{h_{\min}} \neq \emptyset$ and is not connected: we therefore notice that Y contains different minima of I. Denote $Z_1, Z_2, \cdots, Z_k$ these minima, and let Z_i be one of them. At this point, the best possible choice for $C_{h_{\min}+1}(Z_i)$ is given by the geodesic influence zone of Z_i inside Y:

$$C_{h_{\min}+1}(Z_i) = iz_Y(Z_i). \tag{10}$$

These inclusion relationships are illustrated in Fig. 5. Since all the possibilities have been discussed, we take as second set of our recursion the following one:

$$X_{h_{\min}+1} = \min_{h_{\min}+1} \cup\, IZ_{T_{h_{\min}+1}(I)}(X_{h_{\min}}). \tag{11}$$

This relation holds of course for all levels h, and finally, we obtain the following definition.

Definition 6 (Catchment basins and watersheds by immersion): The set of the catchment basins of the grayscale image I is equal to the set $X_{h_{\max}}$ obtained after the following recursion:

a) $X_{h_{\min}} = T_{h_{\min}}(I)$,

b) $\forall\, h \in [h_{\min}, h_{\max}-1], X_{h+1} = \min_{h+1} \cup\, IZ_{T_{h+1}(I)} \cdot (X_h)$.

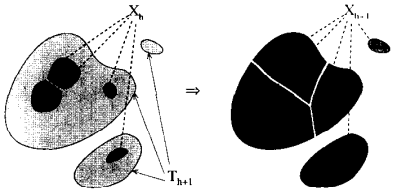


Fig. 6. Recursion relation between X_h and X_{h+1}.

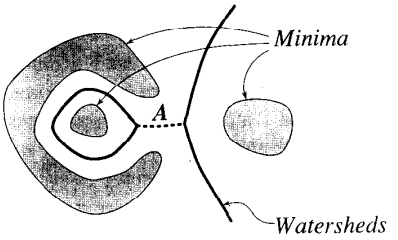


Fig. 7. A classic trap: edge A does not belong to the set of watershed lines.

The watersheds of I correspond to the complement of this set in D_I, i.e., to the set of the points of D_I which do not belong to any catchment basin.

The recursion relation between two successive levels is illustrated in Fig. 6.

D. Review of the Existing Algorithms

We here restrict our attention to the algorithms developed in the field of image processing. (Those coming from the field of digital topography are indeed extensively reviewed in [12], [43].) In Section IV-B, their advantages and drawbacks will be summarized and opposed to those of the algorithm proposed in this paper.

Beucher and Lantuéjoul [3] were the first to propose a watershed algorithm based on a immersion analogy (see Section II-A). According to it, the geodesic influence zones of a level inside the next one are determined via binary thickenings until idempotence with structuring elements M and E (these capital letters refer to *Golay's alphabet* [16]). As illustrated by Fig. 7, watersheds computed this way may, in some special configurations, contain undesirable arcs. This is due to the fact that the involved thickenings are *homotopic* ones, whereas the catchment basin associated with a given minimum does not necessarily have the same homotopy of this minimum. On Fig. 7, arc A is not part of the watersheds, since it does not separate two different minima! Furthermore, as explained in [50, ch. 2], such algorithms are inefficient on nonspecialized architectures, since the whole set of pixels needs to be scanned at each thickening step.

Watersheds can also be determined through graytone skeletons (see [32]). Following this approach, Beucher proved that the watersheds of a function are nothing but the closed arcs of its skeleton [4]. As in the binary case, skeletons of grayscale images can be computed by performing homotopic thinnings until idempotence with the L structuring element. Let us recall that the thinning of a function I by a two-phase structuring element $T = (T_1, T_2)$, denoted $I \bigcirc T$, is given by

$\forall\, p \in D_I$,

$$I(p) \bigcirc T = \begin{cases} I(p) \oplus \check{T}_2 & \text{if } I(p) \oplus \check{T}_2 < I(p) \leq I(p) \ominus \check{T}_1, \\ I(p) & \text{otherwise.} \end{cases} \quad (12)$$

In the above formula, $\oplus$ and $\ominus$ refer to the well-known Minkowski operations [33]. The nonclosed arcs of the skeleton can then be easily removed by thinning it until idempotence with the structuring element E. However, the composition of two idempotent transformation is not necessarily idempotent. Hence, the whole process (skeletonization followed by pruning) must be iterated until idempotence [43]. This results in a highly time consuming algorithm which, like the previous one, falls into such traps as that illustrated by Fig. 7. (Arc A cannot be removed by pruning.)

The algorithm proposed by Friedlander in [14] is a sequential one [38]. Such sequential algorithms are extensively used in the field of MM [25]. They rely on scanning of the images in predefined orders, in which the new value of each pixel is immediately taken into account for the computation of the next pixel values. Here, an initial propagation step yields the "broad catchment basins" of the image I under study. The broad catchment basin associated to a minimum M is the set of pixels that can be reached by following a never descending path starting from M. Every pixel of D_I belongs at least to one broad catchment basin. The zone where two or more broad catchment basins overlap is referred to as a "watershed zone." Its complement constitutes the "restricted catchment basins." Finally, the catchment basins themselves are obtained via the SKIZ of the restricted catchment basins. This procedure is relatively fast, since all steps are performed sequentially. In addition, since a labelling of the different catchment basin is used in the algorithm, such traps as that of Fig. 7 are avoided. However, the propagation step being based on the definition in terms of flow paths (see Section II-C), the resulting watershed lines may be improperly located, i.e., not even on crest-lines of the image.

Beucher describes a sequential algorithm based on an *arrowing* of the image [6, ch. 5]. It requires three major steps: first, complete the function in order to get a lower-complete function (i.e., a function where the only points without neighbors of lower altitude are the points of the minima). Second, "arrow" the completed function: for each pair of adjacent pixels $(p_1, p_2) \in G$, the fact that p_1 is strictly higher than p_2 is indicated by an arrow pointing from p_1 to p_2 (see Fig. 8). Such an arrowing allows one to code the neighborhood configurations of all pixels in a very compact way. Third, label the regional minima and propagate the labels along the image via the arrowing. This algorithm may be implemented sequentially, and is thus faster than the two first ones. However, here again, some errors may occur in the propagation step.

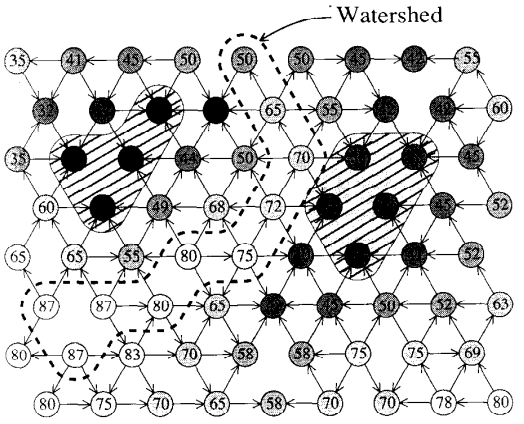


Fig. 8. An example of an *arrowing* on a hexagonal grid. There are here two minima which correspond to the hachured zones.

III. Proposed Algorithm

A. General Description

The algorithms reviewed above share some characteristics: first, they are based on successive complete scannings of the image under processing. This means that at each step, *all* the pixels are scanned one after the other in a predetermined order, generally a video or an antivideo scanning. Second, these algorithms do not run in a fixed number of iterations: the image has to be scanned entirely at each iteration, the number of which being often very large. Some of these algorithms have been implemented on specialized architectures. In this case, their computation time remains acceptable. But on conventional computers, they are far from being efficient: computing a watershed transformation may take hours in some cases.

To speed up the computations, one has to design algorithms taking into account the fact that, at a given step, only the values of a small number of pixels may be modified [50]. Rather than scanning the entire image to modify only two pixels, the algorithm should be designed to have a direct access to these pixels. Therefore, in the following, we suppose that the image pixels are stored in a simple array, and that the following two conditions are satisfied:

1) *Random access* to the pixels of an image.
2) *Direct access to the neighbors of a given pixel* (its 4 neighbors in 4-connectivity, 6 on a hexagonal grid, 8 on a 8-connectivity, etc.).

If these two prerequisites are fulfilled, one is able to design extremely efficient morphological algorithms. The algorithms designed by M. Schmitt [39], based on the propagation of chain codes along the images, rely on these principles and are particularly efficient for geodesic transformations. Similarly, some new algorithms have recently been designed for computing morphological transformations with any kind of structuring elements [51], various kinds of skeletons as well as many different morphological transformations [50].

Our algorithm is based on the definition given in Section II-C.We therefore have to consider the successive thresholds of the image under study, and to compute geodesic influence zones of one threshold inside the next one as fast as possible. In the sequel, for the sake of clarity, the proposed algorithm is decomposed into two steps. Putting them together only allows one to save a little time and memory space. In or-

der to have a direct access to the pixels at a given level, the first step consists in an initial *sorting* of the pixels in the increasing order of their gray values. The method described in Section III-B completes this very efficiently, since it exploits the particular structure of our data. It runs in linear time with respect to the number of pixels to be sorted. In the second step, a fast computation of geodesic influence zones is enabled by a breadth-first scanning of each threshold level. This particular scanning is implemented via the use of a *queue of pixels,* i.e., a first-in-first-out data structure. Notice that many morphological transformations can be efficiently performed by algorithms based on queue structures [52], [50]. This second step, called the *flooding step,* is detailed in Section III-C.

B. The Sorting Step

Among the vast number of available sorting techniques [26], one is particularly suited to the present problem. It is a distributive algorithm [26] which resorts to address calculations. This technique was introduced by E. J. Isaac and R. C. Singleton in [22] and is briefly described in [26, pp. 162–166]. The procedure first determines all the exact frequency distribution of each image gray level. The cumulative frequency distribution is then computed. This induces the direct assignment of each pixel to a unique cell in the sorted array.

Let us denote n the number of image pixels and $h_{\min}$ and $h_{\max}$ the lowest and largest gray levels, respectively. The present sorting technique has the considerable advantage of requiring only $2n$ "look and do" operations—one for determining the frequency distribution and the other for the assignment—plus $h_{\max} - h_{\min} - 1$ additions to get the cumulative frequency distribution. As memory and time requirements to compute frequency distributions are generally negligible compared to those required for images, this sorting procedure constitutes one of the best choices to deal with our data. Together with the cumulative frequency distribution, the sorted array of (pointers to) pixels enables a direct access to the pixels at a given level h. This ability is extensively used in the flooding step described in the next section.

C. The Flooding Step

Once the pixels have been sorted, we proceed to the progressive flooding of the catchment basins of the image. Suppose the flooding has been done up to a given level h. Every catchment basin already discovered—i.e., every catchment basin whose corresponding minimum has an altitude lower or equal to h—is supposed to have a unique label. Thanks to the initial sorting, we now access the pixels of altitude $h+1$ directly and given them a special value, say MASK. Those pixels among them which have an already labeled pixel as one of their neighbors are put into the queue. Starting from these pixels, the queue structure enables to extend the labelled catchment basins inside the mask of pixels having value MASK, by computing geodesic influence zones (see Section II-C). After this step, only the *minima* at level $h + 1$ have not been reached. Indeed, they are not connected to any of the already labelled catchment basins. Therefore, a second scanning of the pixels at level $h + 1$ is necessary to detect the pixels which still have value MASK, and to give a new label to the thus discovered catchment basins.

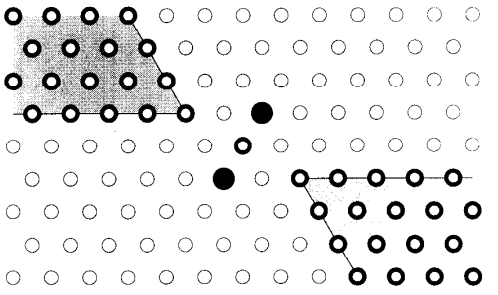


Fig. 9. According to the hexagonal distance, all the bold pixels (gray areas) are equidistant to the two black ones.

The queue which is used is a first-in-first-out data structure: the pixels which are first put into it are those which can first be extracted. In practice a queue is simply a large enough array of pointers to pixels, on which three operations may be performed:

- *fifo_add(p)* Puts the (pointer to) pixel p into the queue.
- *fifo_first()* Returns the (pointer to) pixel which is at the beginning of the queue, and removes it from the queue.
- *fifo_empty()* Returns *true* if the queue is empty and *false* otherwise.

In order to implement such operations, a kind of "circular" queue is one of the most efficient choices: the array representing our FIFO structure is addressed by two indexes, *ptr_first* and *ptr_last.* Each time a new element is put into the queue, it is stored at the address toward which *ptr_last* is pointing. *prt_last* is then incremented. When the limit of the array is reached, this index loops back to the beginning of the array. Similarly, *ptr_first* is a pointer toward the first element which can be removed from the structure, and is incremented after each removal. It may also loop back to the start of the array. To optimize memory requirements, "dynamic" queues may also be considered, but their use generally slows down the whole process.

Not only does the use of a queue of pixels speed up the computations, it also allows us to solve the accuracy problems encountered by most of the algorithms reviewed in Section II-D. First, the labeling of the catchment basins automatically avoids such traps as that of Fig. 7. Now, in order to get perfectly located watershed arcs, the successive geodesic SKIZ involved in the process have to be as good as possible. The first thing to notice is that, according to the discrete distance associated with the underlying grid, the set of pixels which are equidistant to two given connected components may well not be a line, but a very thick area. This is illustrated by Fig. 9. (Recall that the distance between two pixels is equal to the minimal number of grid edges to cross to go from one to the other.) Consequently, some simplistic rules in the computation of the geodesic SKIZ's could result in unwanted thick watershed areas. More precisely, suppose that the plateaus at elevation h are currently being flooded,

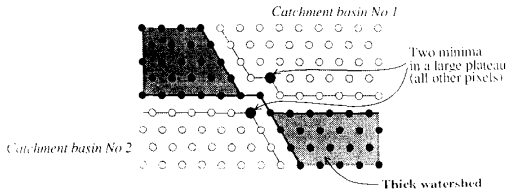


Fig. 10. Using simplistic rules in the breadth-first computation of the geodesic influence zones may result in unwanted thick watershed lines. Here, the basins are not symmetric because the neighbors of the upper minimum were put into the queue before the neighbors of the lower one.

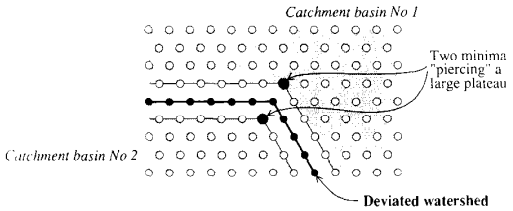


Fig. 11. Some simplistic geodesic SKIZ computations may result in deviated watershed lines (same pixel configuration as in Fig. 10).

and let p be the current pixel. A simplistic rule would be to say that p is necessarily a watershed-pixel (i.e., belongs to the set of the watershed lines) if it has a watershed-pixel in its neighborhood. An example of thick watershed produced with such a rule is illustrated by Fig. 10. Similarly, declaring that any pixel which has two pixels with different labels in its neighborhood is a watershed-pixel may result in deviated watershed lines, as illustrated by Fig. 11. Let us stress that these problems are not specific of the hexagonal grid and also exist with square ones.

To get rid of such difficulties, one may think of resorting to better discrete distances in the geodesic SKIZ computations [7], [35], [46], [37]. It is even possible to make use of actual Euclidean distances by adapting such algorithms as those described in [10] or [50, ch. 3] to the present case. However, this would involve propagating *vectors* rather than distances and would put a considerable burden on the entire flooding step. Therefore, these ideas have not been retained in the present implementation.

Instead, we chose to restrict ourselves to the distance induced by the used discrete grid. The idea is to make use of a work image where the successive geodesic distances are actually stored during the breadth-first propagation. In conjunction with carefully written rules for the propagation of the labels inside the plateaus (see algorithm below, lines 31–43), this results in very well located watershed lines, even in the case of minima embedded in large plateaus, as illustrated by Fig. 12. Note that the algorithm given below is designed to yield a tesselation of the image in its different catchment basins. Only the pixels which are exactly "half-way between" two catchment basins are given a special value, hereafter denoted WSHED.

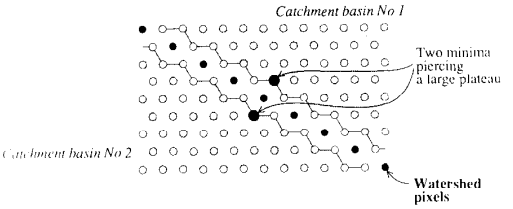


Fig. 12. Exact catchment basins and watershed points obtained with the present algorithm.

Algorithm: Fast Watersheds

```
# define MASK   −2 /⋆ initial value of a threshold level ⋆/
# define WSHED   0 /⋆ value of the pixels belonging to the
                     watersheds ⋆/
# define INIT   −1 /⋆ initial value of im_o ⋆/
• —input:  im_i, decimal image;
  —output: im_o, image of the labeled watersheds;
• Initializations:
  —Value INIT is assigned to each pixel of im_o :
    ∀ p ∈ D_{im_0}, im_0(p) = INIT;
  —current_label ← 0;
  —current_dist: integer variable
  —im_d: work image (of distances), initialized to 0;
• Sort the pixels of im_i in the increasing order of their gray
  values.
  Let h_min and h_max designate the lowest and highest values,
  respectively.
• For h ← h_min to h_max {
        /⋆ geodesic SKIZ of level h − 1 inside level h ⋆/
  For every pixel p such that im_i(p) = h {
        /⋆ These pixels are accessed directly through the
           sorted array. ⋆/
    im_o(p) ← MASK;
    if there exists p' ∈ N_G(p) such that im_o(p') > 0 or
       im_o(p') = WSHED {
       im_d(p) ← 1; fifo_add(p);
    }
  }
  current_dist← 1; fifo_add(fictitious_pixel);
  repeat indefinitely {
    p ←fifo_first();
    if p = fictitious_pixel {
       if fifo_empty() = true then BREAK;
       else { fifo_add(fictitious_pixel);
          current_dist ← current_dist + 1;
          p ← fifo_first();
       }
    }
    For every pixel p' ∈ N_G(p) {
       if im_d(p') < current_dist and (im_o(p') > 0 or
          im_o(p') = WSHED) {
             /⋆ i.e., p' belongs to an already labeled
                basin or to the watersheds ⋆/
          if im_o(p') > 0 {
             if im_o(p) = MASK or im_o(p) = WSHED then
                im_o(p) ← im_o(p');
             else if im_o(p) ≠ im_o(p') then
                im_o(p) ← WSHED;
          }
```

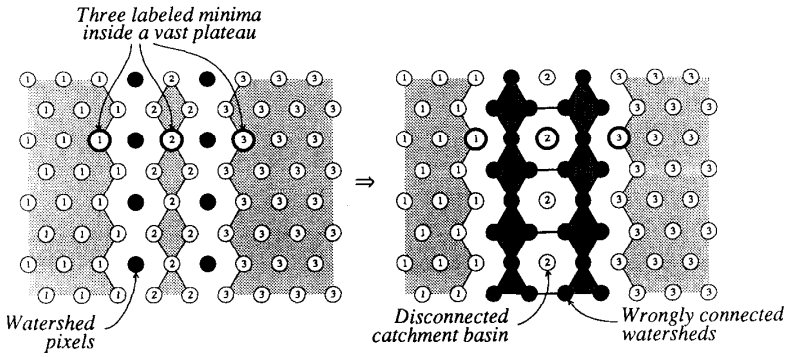


Fig. 13. To separate the different catchment basins, one cannot simply give value WSHED to the pixels having a pixel with another label in their neighborhood. Otherwise, some errors may occur.

```
            else if im_o(p) = MASK then im_o(p) ← WSHED
        }
        else if im_o(p') = MASK and im_d(p') = 0 {
            im_d(p') ← current_dist + 1; fifo_add(p');
        }
    }
}
        /* checks if new minima have been discovered */
For every pixel p such that im_i(p) = h {
    im_d(p) ← 0;        /* the distance associated with p is
                           reset to 0 */
    if im_o(p) = MASK {
        current_label ← current_label + 1;
        fifo_add(p); im_o(p) ← current_label;
        while fifo_empty() = false {
            p' ← fifo_first();
            For every pixel p'' ∈ N_G(p') {
                if im_o(p'') = MASK { fifo_add(p'');
                    im_o(p'') ← current_label; }
            }
        }
    }
}
}
```

At this point, to get rid of the WSHED-pixels (i.e., to obtain a real tesselation of the image in its different catchment basins), it suffices to give them the value of one of their labelled neighbors (in fact, the pixels belonging to thick watershed areas must be processed differently . . .). On the other hand, if we want to separate the different catchment basins, it suffices to give value WSHED to the labeled pixels having in their neighborhood at least one pixel with a *smaller* label. This is not symmetric, but Fig. 13 shows why it is not wise to give label WSHEDS to all the pixels having in their neighborhood a pixel with a *different* label: in some cases, the catchment basins could be disconnected and the watersheds be wrongly connected!

IV. Discussion, Performances

A. Analysis of the Algorithm

This watershed algorithm runs in *linear time* with respect to the number N of pixels in the image which is processed. Indeed:

- In the sorting step, two and only two scannings of the whole image are necessary to construct the sorted array of pointers to pixels. An additional scanning of the frequency array is also required.[1]
- In the flooding step, each pixel is scanned three times on average: at each threshold level h, all the concerned pixels are first assigned value INIT. Then they can be considered a second time during the breadth-first scanning of the plateaus at altitude h. Lastly, all the pixels at altitude h are scanned again in order to see if new minima have appeared.

The two above steps running in linear time, the entire algorithm is linear with respect to N. Furthermore its execution time is practically independent of the number of gray levels in the image. On the SUN Sparc Station 1, the computation of the watersheds of a 256 × 256 image takes approximately 2.5 seconds. This is extremely fast compared to some of the existing algorithm, which may take more than one hour for the same computation (see Table I).

As concerns the memory requirements, they are a little more restricting, since the algorithm uses:

- An output image im_o of the same size as the initial image im_i. The number of catchment basins may be large for practical grayscale images, so that im_o has to be coded on 2 bytes per pixel. We usually use the same image for the input and the output, i.e., we take $im_o = im_i$.
- A sorted array of pointers to pixels. Its size is N (number of pixels in im_i) and a pointer is generally represented on 4 bytes.
- A distance image im_d of the same size as im_i. In fact, im_d is only used for local comparison, so that knowing it modulo 3 is enough. im_d can therefore be coded on 2 bits per pixel.
- An array of pointers to pixels, which must be large enough to contain all the pixels in the queue, at each step. Although this array can be dynamically allocated, it

[1] In most cases, i.e., for images coded on 8 bits or less, the size of this array is negligible compared to that of an image. We thus admit that the execution time of the sorting procedure is independent of the number of gray levels.

TABLE I
COMPARISON OF THE PRESENT WATERSHED ALGORITHM WITH THOSE DESCRIBED IN SECTION II-D. THE EXECUTION TIMES REFER TO IMPLEMENTATIONS ON A SUN SPARC STATION 1, FOR A 512 × 512 DIGITAL ELEVATION MODEL ON 8 BITS.

Algorithm	Execution Time	Precision of the Lines	Correct Result for Fig. 7	Detection of Thick Watershed Areas	Adaptability to Other Grids	Adaptable to Graphs
Homotopic binary thickenings	hours	xx	no	no	xx	no
Grayscale thinnings	hours	xx	no	no	xx	no
Sequential algorithm	68 s	x	yes	no	xxx	no
Arrowing	51 s	xxx	yes	no	x	no
Present algorithm	6.3 s	xxxx	yes	yes	xxxx	yes

is more efficient to use a fixed array of large size: $N/4$ seems to be more than enough in all practical cases.

- A cumulative frequency distribution array, whose size is equal to the number of possible gray levels in im_i. We consider the additional memory it requires as negligible in practice.

To summarize, if the initial image is coded as an array of N pixels, then $7\frac{1}{4} \times N$ bytes are necessary for the watershed computation. This is a lot, but far from being unacceptable in comparison with the random access memory of today's computers.

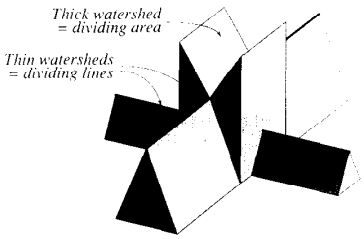


Fig. 14. A case where the watershed lines are no longer thin: we have here a *watershed area*.

B. Advantages, Summary Table

Apart from its computational efficiency, the algorithm introduced in this paper has many other advantages: first, it is very general since it works in 4-, 6-, or 8-connectivity equally well. Once it has been implemented for a given grid, it is straightforward to adapt the program to another grid. To do so, it suffices to change the way the neighbors of a given pixel are generated. Note that on a square grid, if the 4-connectivity is used in the flooding step, the resulting watersheds are only 8-connected. Conversely, if the 8-connectivity is used in the flooding, the resulting watersheds are 4-connected.

Furthermore, the algorithmic definition 6 of watersheds extends to n-dimensional images. The adaptation of the proposed watershed algorithm to n-dimensional images is thus immediate. As said above, it suffices to modify the procedure for generating the neighbors of a given pixel. An example of three dimensional watersheds is shown in Section V-C, and the interest of such a transformation is also discussed. Lastly, Definition 6 also extends to general graphs. Provided the data structure used to represent graphs allows a direct access to the neighbors of a given vertex, our algorithm works fairly well for these objects [48]. Watersheds on graphs are presented in Section V-D and their interest with respect to picture segmentation is shown.

On the other hand, none of the algorithms reviewed in Section II-D is easily adaptable to other digital grids: in particular, the adaptation of the algorithms based on binary thickenings, grayscale thinnings, or arrowing would require cumbersome neighborhood analysis. This is even more true when it comes to extending these procedures to three-dimensional images!

Now, the accuracy of the present algorithm is also remarkable. We have seen in Section III-C that the labeling of the catchment basins allows one to avoid such pitfalls as that illustrated by Fig. 7. We have also shown that the use of a work distance-image results in watershed pixels which are perfectly located, even when large plateaus are involved. Unlike the algorithms reviewed in Section II-D, and in particular, unlike the classic flooding algorithm which uses successive geodesic skeletons by influence zones computed by iterative thickenings [3], the dividing lines are here never deviated. Moreover, the parity problems are avoided, since the algorithm constructs labelled catchment basins rather than watershed lines.

Finally, the present algorithm also gives correct results in presence of configurations which have not been discussed yet: *watershed areas*. These areas are such that one cannot decide towards which minimum a drop falling on them will slide. They correspond to special pixel configurations which are not so rare in practice. An example of a "thick" watershed area is shown in Fig. 14. In such cases, the algorithm introduced in Section III-C will assign value WSHED to all the involved pixels, which is the correct statement. On the contrary, none of the algorithms reviewed in Section II-D is able to detect a thick watershed.

In Table I, we have summarized the qualities of the present algorithm as opposed to those of the algorithms briefly described in Section II-D. The execution times are provided here for information only, since they may vary from one image to another. They refer here to an 8 bits, 512 × 512 digital elevation model with 18 catchment basins. The computations have been achieved in 4-connectivity. Note that algorithms 1 and 2 have also been implemented on a specialized hardware, the Q 570 of Cambridge Instruments. On this machine and for the particular image used in Table I, their common execution

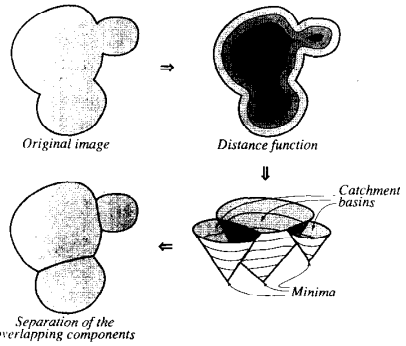


Fig. 15. Binary segmentation by watersheds of the opposite of the distance function.

time was of approximately 200 s.

V. Examples of Application

A. Use of Watersheds in Image Segmentation

The watershed transformation constitutes one of the most powerful segmentation tools provided by mathematical morphology [3], [4], [49]. In this section, the word *segmentation* addresses the extraction of the different objects present in the image under study. As concerns the binary case, this comes down to the separation of the partially overlapping objects—provided there are no artifacts present. This problem is extensively discussed in [49] and [50, Appendix]. Its solution is based on a *marking* of the different components that are to be segmented. A *marking function* is then constructed, whose different catchment basins correspond to the desired objects. Here, the marking function is nothing but the opposite of the *distance function* of our binary image, i.e., the function which associates with every feature point the opposite of its distance to the background. This binary segmentation process is illustrated in Fig. 15. It has been successfully used for many problems, such as the segmentation of rice grains [2] or of coffee bean images [49].

As concerns grayscale images, segmenting them means dividing then into regions: generally, one of them stands for the background whereas each of the others corresponds to one of the objects or areas to be extracted. This segmentation comes down to the extraction of the contours of the desired objects. Now, the problem is to clearly define what is a contour and what is not. Some well-known methods resort to the zero-crossings of the second derivative of the function representing the image I under study [29]. Other edge detectors can be computationally adapted to arbitrary edge profiles [8]. In the field of mathematical morphology, another kind of approach is commonly used: the starting point is to say that the contours of an image correspond to lines where the gray-tone is varying quickly compared to the neighborhood. Suppose now that we have determined an image $\mathrm{grad}(I)$ where the value of each pixel corresponds to the modulus of the gradient at this point (in the following, this image is referred to as a *gradient image*). If we regard it as a relief, the searched contours correspond to some *crest-lines* of this function. At this point, one can consider using on $\mathrm{grad}(I)$ the *grayscale skeleton* [32] as crest-line detector. The problem with this transformation is that it extracts *all* the crest-lines of a given image. This is not what is expected, since the contours that should be extracted are closed ones. Therefore, one has to remove the parasitic dendrites of the skeleton, i.e., to resort to the watershed transformation. According to the gradient which is being used, we define the contours of image I as the *watersheds of its gradient.*

The watersheds of the gradient are thus at the basis of the general morphological approach to segmentation that we briefly present now. A more detailed presentation can be found in [49] or in [50]. The image which will be used to illustrate the present segmentation methodology is denoted I and is displayed in Fig. 16(a)[2] [17]. It is a 256 × 256 image of a vertebral column, digitized according to a hexagonal grid, from which the intervertebral disks have to be extracted. This is a rather difficult problem, since on the one hand, the noise level is relatively high and on the other hand, the desired disks do not have a fixed size and orientation, and can be mistaken for other features. Simple methods based on thresholdings or top-hats [31] do not work, so that one has to make use of more advanced tools.

In fact, the brutal computation of watersheds of the gradient does not constitute a good segmentation method either. Indeed, whatever gradient is used, the simple computation of its watersheds mostly results in an *over-segmentation,* i.e., the correct contours are lost in a mass of irrelevant ones. This is true even if one had taken the precaution of filtering the initial image or its gradient. For example, I' [see Fig. 16(b)] was obtained by performing on I an alternating sequential filtering [41, ch. 10]. A morphological gradient of I' was then determined, which is the supremum of three directional gradients. More precisely, denoting S_1, S_2, and S_3 the three elementary segments of the hexagonal grid, we computed

$$\mathrm{grad}'(I') = \sup_{1 \le i \le 3} [(I' \oplus S_i) - (I' \ominus S_i)]. \qquad (13)$$

This gradient is displayed in Fig. 16(c). Now, in Fig. 16(d), one can see the watersheds of this gradient and appreciate the resulting segmentation! In many cases, the over-segmentation is simply due to noise. But, as in the present example, some irrelevant arcs may also correspond to objects that do not have to be extracted, and whose contours should not appear in the final segmented image. In both cases, so as to get rid of this over-segmentation, one has two possibilities:

- Remove the irrelevant contour elements.
- Modify the gradient function so that the resulting catchment basins only correspond to the desired objects.

The first choice is the most natural and classical one: on the other hand, the watershed image can be regarded as an image of contours, some of which having to be suppressed. On the other hand, one may consider the different catchment basins as regions and merge adjacent regions according to some criteria. These methods are referred to in literature as *region-growing algorithms* [18], [34]. A morphological region growing algorithm relying on watersheds on graphs

[2]This image was provided by Neil Roberts.

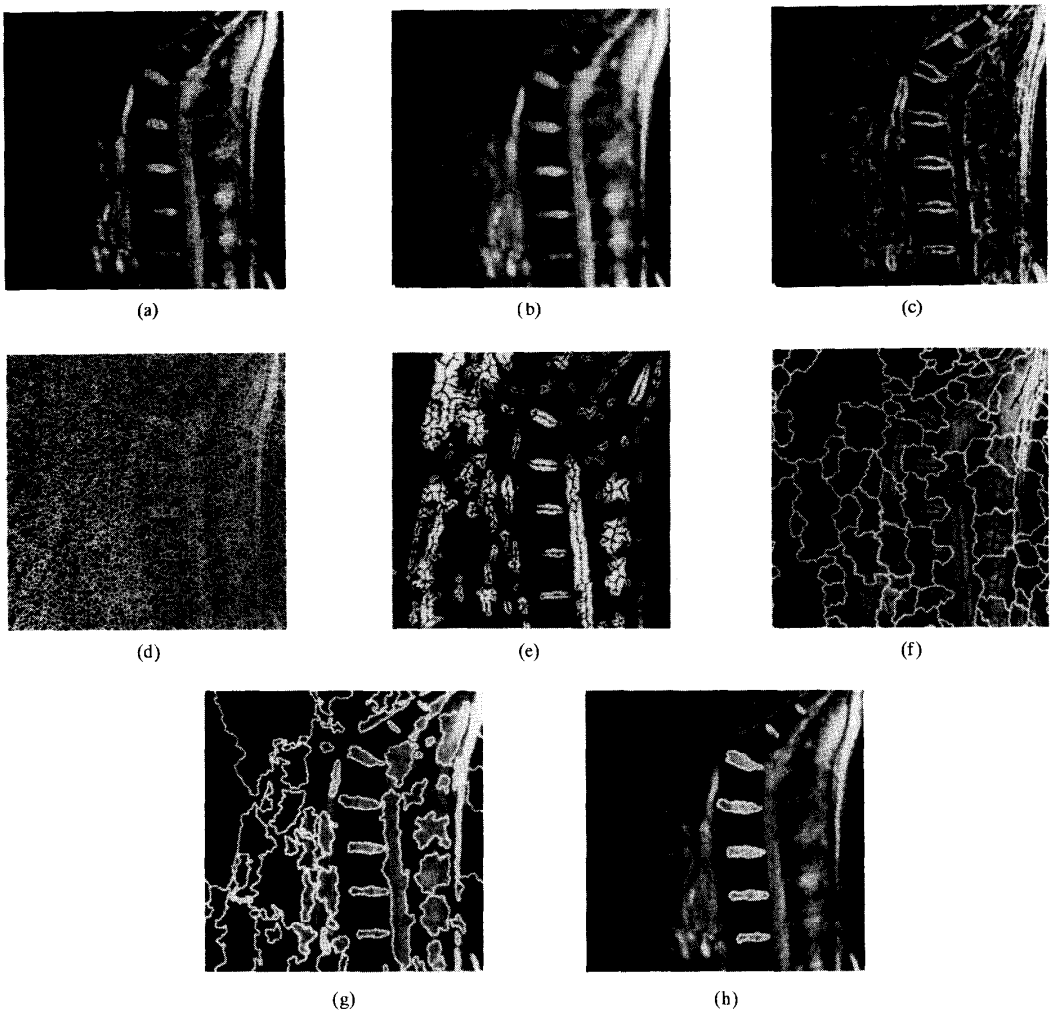


Fig. 16. Segmentation of invertebral disks. (a) Original image I (vertebral column). (b) I' = filtering of I. (c) Gradient image grad$'$ (I'). (d) Watersheds of grad$'$ (I'). (e) Markers: skeletons of the "domes" of I. (f) Inner and outer markers. (g) Watersheds of modified gradient. (h) Final segmentation.

is presented in Section V-D. The second choice makes use of some external knowledge on the collection of images under study, in the sense that it requires and initial *marking* step. One has to use the knowledge available on the problem—shape of the desired objects, noise present on the image, darkness of the background, etc.—to design a robust algorithm for extracting markers of the different regions to be segmented. By marker of a region, we mean a connected set of pixels (or even one single pixel) included in this region. On the example of Fig. 16, using again a topographic analogy, one can see that the intervertebral disks correspond to domes of I having a well defined shape. At this point, the idea is to detect markers of *all* the domes of I, to extract their precise contours and lastly to remove the unwanted domes by using shape information. Hence, a morphological dome extractor described in [17] is utilized. To get more precise markers, the results it yields is then skeletonized. The domes of I as well as their skeletons (actual markers) are displayed in Fig. 16(e). As concerns the background marker, it corresponds to the deepest valley-lines of the original image which separate the previous markers. The algorithm for extracting it is quite similar to the one presented in the next paragraph for selecting the crest-lines of the gradient and is detailed in [49], [50]. This marker is displayed in Fig. 16(f) together with the previously obtained ones.

Once these markers are extracted, a morphological transformation based on grayscale geodesic operations (see [49, Section 4.4], [50, Appendix]) allows us to:

1) impose them as minima of a gradient function grad(I),
2) suppress all the other gradient minima (the insignificant ones) by filling up their catchment basins,
3) preserve the most important crest-lines of grad(I) located between the markers.

This transformation is called *modification of the gradient homotopy,* or simply gradient modification. In some cases, the initial gradient has to be carefully chosen, since its most important crest-lines separating the extracted markers must be

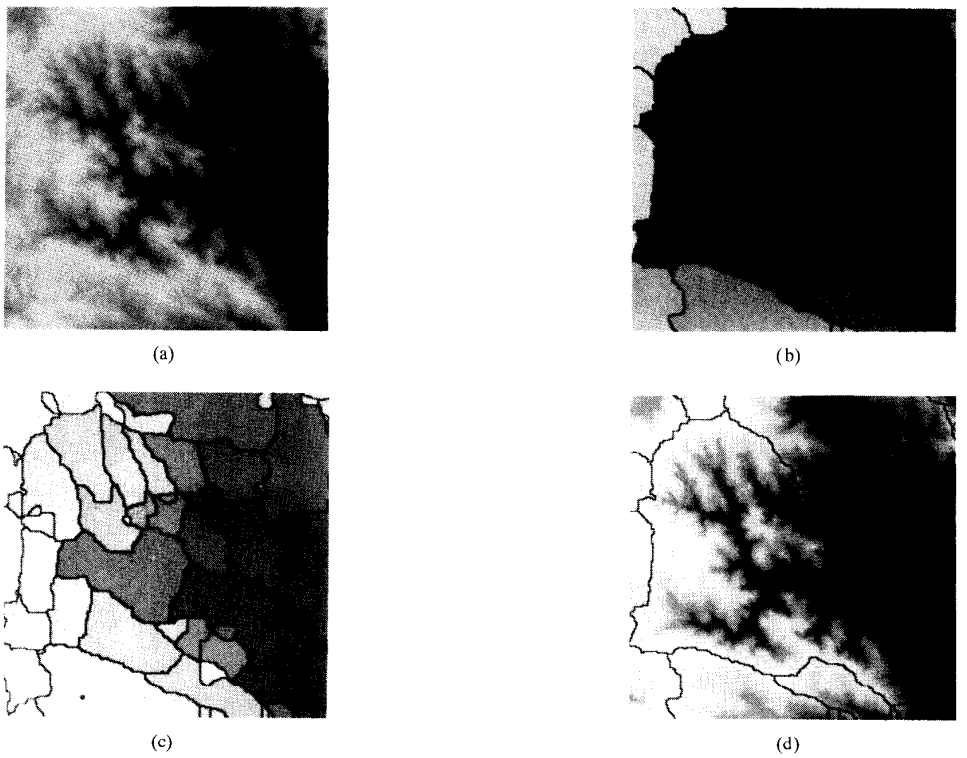


Fig. 17. Topographic watersheds computed on a digital elevation model. (a) Digital elevation model. (b) Catchment basins of original DEM. (c) Catchment basins of modified DEM. (d) Watersheds of modified DEM.

properly located. For the present application, the gradient *grad* introduced in (13) is sufficient.

The computation of the watersheds of this modified gradient provides then the desired segmentation: the catchment basin of the background marker stands for the background itself whereas the boundaries of all other catchment basins—i.e., the watersheds—correspond to the desired objects. In Fig. 16(g), the resulting contours are superimposed on the initial image. Finally, to extract the actual disks from the thus contoured object, one has to use the already mentioned shape information: the disks are the only objects whose skeleton is smooth, elongated in only one direction, and not too long. The final segmentation is displayed in Fig. 16(h). The methodology presented in this section has already been successfully applied to many problems, such as the segmentation of 2-D electrophoresis gels [4], holograms, circuits, cells, etc. Thanks to the algorithm introduced in this paper, the sophisticated segmentation procedure we have presented here runs in less than ten seconds on a SUN Sparc Station.

B. Watersheds on Digital Elevation Models

Watersheds constitute primarily a topographic concept. The present algorithm provides thus an efficient tool for extracting topographic basins from DEM's [43]. The DEM shown in Fig. 17(a) is used to illustrate the methodology.[3] It represents a 256 × 256 matrix of elevations having an equal x-y resolution of 50 meters. We consider it in square grid and 8-connectivity, so that the resulting watersheds will be 4-connected. The major part of the model belongs to the hydrologic basin of the Thyle river (Belgium).

Applying the present watershed algorithm directly to the model leads to Fig. 17(b). Like in the previous section, the disappointing over-segmentation is due to the many minima present inside the DEM. However, fluvial erosion processes do not normally produce any minima at the spatial resolution of usual DEM's. One may thus assume that all minima within the present DEM represent artifacts or data errors. Hence, the only minima to keep are located along the boundaries of the model. The removal of the others by modifying the homotopy of the model is straightforward. The used technique is similar to that presented in the previous section, and is detailed in [43]. Applying the watershed transformation to the modified model leads to the desired catchment basins, which are displayed in Fig. 17(c). In Fig. 17(d) the watersheds are superimposed on the initial DEM.

C. Three-Dimensional Watersheds

As already mentioned, the present algorithm works for n-dimensional images without any modification: it suffices to consider the appropriate set of neighbors of each pixel. For example, Fig. 18(a) is a 32 × 32 × 32 binary image displayed as thirty-two 32 × 32 2-D images (from left to right and top to bottom). It has four voxels set to zero in the lower plane as well as in the upper plane, the rest of the volume being set to

[3] This DEM was produced by Eric Lourtie, Catholic University of Louvain, Belgium.

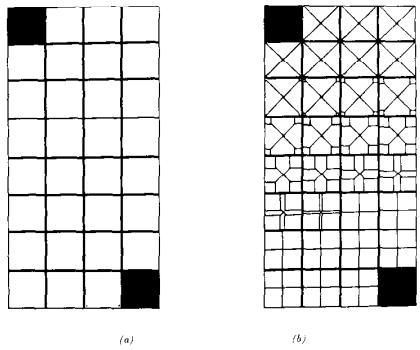


Fig. 18. An example of a three-dimensional watershed.

one. Fig. 18(b) represents the watershed surfaces associated with Fig. 18(a). The cubic grid in six connectivity was used (each voxel is connected to its six nearest neighbors). In fact, the watersheds we have determined are nothing but the SKIZ of the zero voxels present in the 3-D image. n-dimensional watersheds are currently being applied to the segmentation of 3-D medical images and of multispectral images [44].

D. A Morphological Region-Growing Algorithm Based on Watersheds for Graphs

The extension of the present algorithm to graphs[4] is considered here and shown to be at the basis of a powerful region-growing algorithm.

1) Mathematical Morphology on Graphs: Let us first provide some quick reminders on mathematical morphology for graphs: digital images studied through morphological transformations are usually digitized according to square or hexagonal graphs, but one can well imagine using general graphs and applying the same kind of processings [47]. In fact, graphs constitute nothing but a particular kind of lattice and fit very well in the general framework within which MM is defined in [41, ch. 1, 2].

More precisely, let $\mathcal{G} = (V, E)$ be a graph with V its set of vertices and E its set of edges. For the sake of simplicity, $\mathcal{G}$ is here supposed to be a nonoriented and planar 1-graph without loops [19]. It constitutes the underlying structure, just like grids for digital images. The discrete distance induced by E on the set V is denoted d_E and defined as follows:

Definition 7: $\forall (v_1, v_2) \in V^2$, the distance $d_E(v_1, v_2)$ between v_1 and v_2 is equal to the length of the shortest paths between them in E: $d_E(v_1, v_2) = \inf\{l(P), P \text{ path joining } v_1 \text{ and } v_2 \text{ in } E\}$.

The kind of objects we will process can now be defined.

Definition 8: A morphological graph G on $\mathcal{G}$ is a mapping from V into $\mathbb{R}$:

$$G\begin{pmatrix} V & \to & \mathbb{R} \\ v & \mapsto & G(v). \end{pmatrix} \tag{14}$$

Considering a morphological graph on $\mathcal{G}$ comes down to assigning a "gray-tone" to each of its vertices.

[4]Surprisingly, the first implementation of this algorithm was designed for graphs [48].

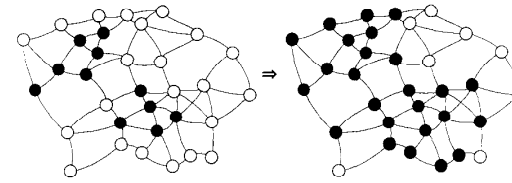


Fig. 19. Dilation of size one of a binary graph (1-vertices in black, 0-vertices in white).

Now, erosions and dilations can well be defined for graphs.

Definition 9: Let G be a morphological graph on $\mathcal{G}$. The dilation of size n of G, denoted $\delta^{(n)}(G)$ is the morphological graph defined on $\mathcal{G}$ by

$$\delta^{(n)}(G)\begin{pmatrix} V & \to & \mathbb{R} \\ v & \mapsto & \sup\{G(v'), v' \in V, d_E(v, v') \le n\}. \end{pmatrix} \tag{15}$$

Similarly, the erosion is defined by taking an inf rather than a sup. An example of a dilation of size one of a binary morphological graph—i.e., a morphological graph taking its values in $\{0, 1\}$—is presented in Fig. 19. Now, all transformations of Euclidean morphology which do not involve any notion of direction can be easily transposed to graphs [47]. In particular, the "immersion" definition of watersheds (see Section II-C) extends to graphs, and the implementation introduced in Section III works very well in this framework [48]. It suffices to use vertices rather than pixels and to code the graphs by means of data structures allowing a direct access to the neighbors of a given vertex [47], [50].

2) Watersheds on Graphs and Picture Segmentation: The watersheds transformation for graphs will now be at the basis of a new segmentation procedure. As explained in Section V-A, one cannot simply use the watersheds of the gradient—*a* gradient—of a grayscale image I to segment it. Indeed, although the correct contours are most of the time present in the watershed image, many contour arcs are irrelevant to the problem. At this point, the solution presented in Section V-A makes use of an external knowledge on the sample under study, in order to find a way for extracting markers of the different regions. These markers are then utilized to modify the homotopy of the gradient on which watersheds will be computed. This method avoids the oversegmentation to appear.

However, for some very complex segmentation problems, and especially when the sample are very different from one another, extracting robust markers is an almost impossible task. The idea is then to remove the insignificant contour arcs of the gradient watersheds. In the present section, rather than using the watershed lines, the dual representation is used: we consider the tessellation provided by the catchment basins of the gradient (here, we implicitly suppose that the watersheds have zero pixel thickness). Suppressing a watershed line comes down to merging two catchment basins.

Let $\{m_k\}_{k\in[1,n]}$ be the minima of the gradient grad(I) of grayscale image I, and denote $\{C(m_k)\}_{k\in[1,n]}$ their associated catchment basins. Each minimum m_k corresponds to an area of I which is "more homogeneous" than the neighborhood.

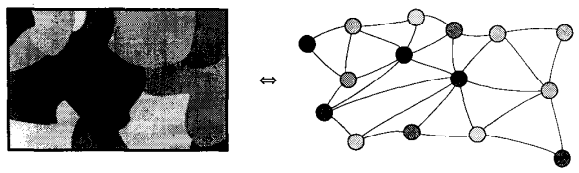


Fig. 20. Mosaic image and associated adjacency graph.

The associated catchment basin $C(m_k)$ simply extends this area until it is bounded by some crest-lines of the gradient function. By assigning to each basin a unique gray level, we decompose image I into homogeneous areas and get a simpler image called *mosaic image.* Of course, the gray level assigned to each catchment basin in the mosaic image has to be carefully chosen, and must be close to the corresponding gray levels of the original image I. One of the best solutions is to assign to the pixels in $C(m_k)$ the infimum (the supremum would work equally well) of $\{I(p), p \in m_k\}$.

Definition 10: Let I be a grayscale image and grad(I) be a gradient image of I. Let $\{m_k\}_{k\in[1,n]}$ be the minima of grad(I) and $\{C(m_k)\}_{k\in[1,n]}$ their associated catchment basins.

The mosaic image of order 1 of I, denoted $I^{(1)}$, is given by

$$\forall\, k \in [1,n], \quad \forall\, p \in C(m_k), \quad I^{(1)}(p) = \inf\{I(p),\, p \in m_k\}. \tag{16}$$

The mosaic image $I^{(1)}$ provides a decomposition of I into roughly homogeneous regions, but as said above, one has to merge regions into larger ones to get rid of the over-segmentation. To do so, we consider the adjacency graph $G^{(1)}(I)$ of $I^{(1)}$: its vertices correspond to the different catchment basins, and there is one and only one edge between two adjacent regions. Moreover, the value associated with each vertex of $G^{(1)}(I)$ is the gray level of the corresponding region. The algorithm for determining the graph is based on a contour tracking of the different regions of $I^{(1)}$. An example of adjacency graph is displayed in Fig. 20. The procedure described in the previous paragraph for I can now well be applied to $G^{(1)}(I)$, by using gradients and watersheds on graphs. This yields a mosaic image of second order $I^{(2)}$ where each region corresponds to a catchment basin of the gradient of $G^{(1)}(I)$. Homogeneous regions of $I^{(1)}$ have been merged into larger ones, thus removing a bit of the over-segmentation. The process is then iterated until the desired merging level is reached, or until a given criterion is fulfilled.

An example of this procedure is shown in Fig. 21. Unfortunately, this example is not extremely significant, since there are not actually regions to segment on image 21(a). But it shows how the described procedure works. Compared to most region-growing algorithms [21], [15], [34], this method has the advantage of being absolutely independent of the order in which the vertices are scanned. Moreover, the merging is not done according to *local* criteria: the watershed being a global transformation, numerous regions may be merged at each step into a single one, whereas other regions remain unchanged. Lastly, contrary to many *split and merge* algorithms, the initial decomposition is not done blindly: the catchment basin tessellation we have presented is meaningful compared to the quadtree decomposition described in [15]. Similar algorithms have already been successfully developed by S. Beucher to segment road images [5]. This method also provides a hierarchical decomposition of images which is an alternative to other morphological decompositions, such as those based on openings and closings [20].

VI. Conclusion and Prospects

The watershed algorithm introduced in this paper is extremely powerful compared to the existing ones. Not only is it often hundreds of times faster on conventional computers, but it also proves to be more accurate. Furthermore, it turns out to be very flexible, since it can be easily adapted to any kind of digital grid and extended to n-dimensional images and graphs.

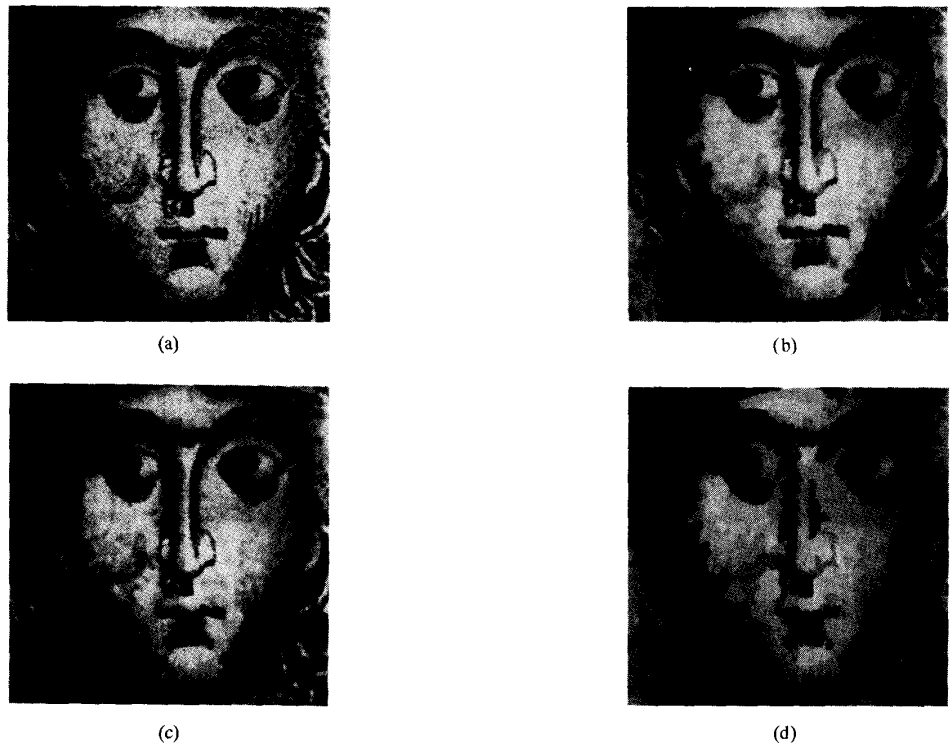
The examples of application which have been provided clearly illustrate the huge interest of the watershed transformation. Until now, its computation was so time consuming on conventional computers that only few people could actually use this transformation in practice. The present algorithm should now allow anyone to resort to watersheds for solving complex segmentation problems. Furthermore, the first steps into the watershed segmentation of 3-D grayscale images have already been enabled by this implementation. It is thus expected to contribute to new insights into the use of watersheds in the field of image analysis. In particular, more experiments are currently being carried on to evaluate the interest of watersheds on graphs with respect to picture segmentation.

Acknowledgment

We are most grateful to M. Grimaud for his useful assistance and for having provided the example of grayscale segmentation described in Section V-A.

References

[1] L. E. Band, "Topographic partition of watersheds with digital elevation models," *Water Resources Res.*, vol. 22, no. 1, pp. 15–24, 1986.
[2] M. Benali, "Du choix des mesures dans les procédures de reconnaissances des formes et d'analyse de texture," Ph.D. dissertation, School of Mines, Paris, 1986.
[3] S. Beucher and C. Lantuéjoul, "Use of watersheds in contour detection," in *Proc. Int. Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation,* Rennes, France, Sept. 17–21, 1979.
[4] S. Beucher, "Watersheds of functions and picture segmentation," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing,* Paris, France, May 1982, pp. 1928–1931.
[5] ——, "Obstacle detection and vehicle trajectories," *Prometheus Image Processing Meeting,* Paris, France, May 18, 1989.
[6] ——, "Segmentation d'images et morphologie mathématique," Ph.D. dissertation, School of Mines, Paris, France, June 1990.
[7] G. Borgefors, "Distance transformations in digital images," *Comput. Vision, Graphics, Image Processing,* vol. 34, pp. 334–371, 1986.
[8] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
[9] S. H. Collins, "Terrain parameters directly from a digital terrain model," *Canadian Surveyor,* vol. 29, no. 5, pp. 507–518, 1975.
[10] P. E. Danielsson, "Euclidean distance mapping," *Comput. Graphics Image Processing,* vol. 14, pp. 227–248, 1980.
[11] H. Digabel and C. Lantuéjoul, "Iterative algorithms," in *Proc. 2nd European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine,* Caen, France, Oct. 1977, J. L. Chermant, Ed. Stuttgart, West Germany: Riederer Verlag, 1978, pp. 85–99.
[12] D. Douglas, "Experiments to locate ridges and channels to create a new type of digital elevation model," *Cartographica,* vol. 23, no. 4, pp. 29–61, 1986.



(a) (b)

(c) (d)

Fig. 21. Hierarchical decomposition of a greek mosaic image: initial image I (a) mosaic decompositions of order 1 (b), 2 (c), and 3 (d).

[13] F. Doyle, "Digital terrain models: An overview," *Photogram. Eng. Remote Sensing,* vol. 44, no. 12, pp. 1481–1485, Dec. 1978.

[14] F. Friedlander, "A sequential algorithm for detecting watersheds on a gray level image," *Acta Stereologica,* vol. 6/III (*Proc 7th Int. Cong. For Stereology*), Caen, France, pp. 663–668, Sept. 1987.

[15] A. Gagalowicz, "A new approach for image segmentation," in *Proc. 8th Int. Conf. Pattern Recognition,* Paris, France, Oct. 1986.

[16] M. Golay, "Hexagonal pattern transforms," in *Proc. IEEE Trans. Comput.,* vol. C-18, no. 8, Aug. 1969.

[17] M. Grimaud, "Intervertebral disk segmentation," School of Mines, Paris, France, Internal Rep. CMM, Jan. 1990.

[18] R. Haralick and L. Shapiro, "Survey: Image segmentation techniques," *Comput. Vision, Graphics, Image Processing,* vol. 29, pp. 100–132, 1985.

[19] F. Harary, *Graph Theory.* Reading, MA: Addision-Wesley, 1969.

[20] H. J. A. M. Heijmans and A. Toet, *Morphological Sampling,* Center Math. Comput. Sci., Amsterdam, The Netherlands, Internal Rep. AM-R8913, Aug. 1989.

[21] J. L. Horowitz and T. Pavlidis, "Picture segmentation by a directed split and merge procedure," in *Proc 2nd Int. Conf. Pattern Recognition,* 1974, pp. 424–433.

[22] E. J. Isaac and R. C. Singleton, "Sorting by address calculation," *J. ACM,* vol. 3, pp. 169–174, 1956.

[23] C. Lantuéjoul, "Skeletonization in quantitative metallography," in *Issues of Digital Image Processing,* R. M. Haralick and J. C. Simon, Eds. Groningen, The Netherlands: Sijthoff and Noordhoff, 1980.

[24] C. Lantuéjoul and F. Maisonneuve, "Geodesic methods in quantitative image analysis," *Pattern Recognition,* vol. 17, pp. 177–187, 1984.

[25] B. Laÿ, "Recursive algorithms in mathematical morphology," *Acta Stereologica,* vol. 6/III (*Proc. 7th Int. Cong. for Stereology*), Caen, France, pp. 691–696, Sept. 1987.

[26] H. Lorin, *Sorting and Sort Systems* (The System Programming Series). Reading, MA: Addison-Wesley, 1975.

[27] F. Maisonneuve, "Sur le partage des eaux," School of Mines, Paris, France, Internal Rep. CMM, 1982.

[28] D. Marks, J. Dozier, and J. Frew, "Automated basin delineation from digital elevation data," *Geoprocessing,* vol. 2, pp. 299–311, 1984.

[29] D. Marr, *Vision.* New York: Freeman, 1982.

[30] M. Matheron, *Random Sets and Integral Geometry.* New York: Wiley, 1975.

[31] F. Meyer, "Cytologie quantitative et morphologie mathématique," Ph.D. dissertation, School of Mines, Paris, France, 1979.

[32] ——, "Skeletons and perceptual graphs," *Signal Processing,* vol. 16, no. 4, pp. 335–363, Apr. 1989.

[33] H. Minkowski, "Allgemein lehrsätze über konvexe polyeder," *Nach. Ges. Wiss. Göttingen,* pp. 198–219, 1897.

[34] O. Monga, "An optimal region growing algorithm for image segmentation," *Int. J. Pattern Recog. Artificial Intell.,* vol. 3, no. 4, Dec. 1987.

[35] J. Piper and E. Granum, "Computing distance transformations in convex and non-convex domains," *Pattern Recog.,* vol. 20, pp. 599–615, 1987.

[36] T. K. Puecker and D. H. Douglas, "Detection of surface-specific points by local parallel processing of discrete terrain elevation data," *Comput. Vision, Graphics, Image Processing,* vol. 4, pp. 375–387, 1975.

[37] I. Ragnemalm, "Contour processing distance transforms," in *Progress in Image Analysis and Processing,* Cantoni *et al.,* Eds. Cleveland, OH: World Scientific, 1990, pp. 204–212.

[38] A. Rosenfeld and J. L. Pfaltz, "Sequential operations in digital picture processing," *J. ACM,* vol. 13, no. 4, pp. 471–494, 1966.

[39] M. Schmitt, "Des algorithmes morphologiques à l'intelligence artificielle," Ph.D. dissertation, School of Mines, Paris, France, Feb. 1989.

[40] J. Serra, *Image Analysis and Mathematical Morphology.* London: Academic, 1982.

[41] J. Serra, Ed., *Image Analysis and Mathematical Morphology, Part II: Theoretical Advances.* London: Academic, 1988.

[42] J. Serra and L. Vincent, *Lecture Notes on Morphological Filtering,* School of Mines, Cahiers du Centre de Morphologie Mathématique, Paris, France, vol. 8, 98 pp., 1989.

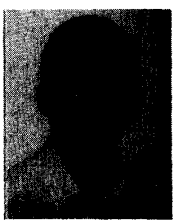
[43] P. Soille and M. Ansoult, "Automated basin delineation from DEMs using mathematical morphology," *Signal Processing,* vol. 20, pp. 171–182, 1990.

[44] P. Soille, "Inversion d'images multispectrales par la morphologie mathématique," School of Mines, Paris, France, Internal Rep. CMM, Jan. 1990.

[45] S. R. Sternberg, "Grayscale morphology," *Comput. Vision, Graphics, Image Processing,* vol. 35, pp. 333–355, 1986.

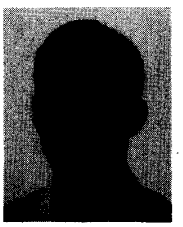
[46] B. J. H. Verwer, P. W. Verbeek and S. T. Dekker, "An efficient uniform cost algorithm applied to distance transforms," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 11, pp. 425–429, 1989.

[47] L. Vincent, "Graphs and mathematical morphology," *Signal Processing*, vol. 16, no. 4, pp. 365–388, Apr. 1989.
[48] ——, "Mathematical morphology for graphs applied to image description and segmentation," in *Proc. Electronic Imaging West*, Pasedena, CA, Apr. 1989, pp. 313–318.
[49] L. Vincent and S. Beucher, "The morphological approach to Segmentation: An introduction," School of Mines, Paris, France, Internal Rep. CMM, July 1989.
[50] L. Vincent, "Algorithmes morphologiques à base de files d'attente et de lacets. Extension aux graphes," Ph.D. dissertation, School of Mines, Paris, France, May 1990.
[51] ——, "Morphological transformations of binary images with arbitrary structuring elements," *Signal Processing*, vol. 22, no. 1, Jan. 1991.
[52] L.J. Van Vliet and B.J.H. Verwer, "A contour processing method for fast binary neighborhood operations," *Pattern Recog. Lett.*, vol. 7, pp. 27–36, Jan. 1988.



Luc Vincent was born in Paris, France, on December 10, 1964. He received the Engineering degree from the Ecole Polytechnique in 1986 and the Master's degree in computer science and artificial intelligence from the Paris XI University in 1987.

From April 1986 to June 1990, he worked at the Center for Mathematical Morphology of the Ecole des Mines de Paris under the guidance of Prof. J. Serra, and defended his PhD in May 1990. Then, after a short research period at the Center for Mathematics of Amsterdam, he joined the Harvard Robotics Laboratory (Division of Applied Sciences) as a Postdoctoral Fellow, sponsored by Dassault Electronique. His research interests range from theoretical studies (morphology on graphs, morphological filtering, etc.) to very practical applications taken from the fields of medical imaging, radar images, and digital elevation models, among others. He has been especially active in image segmentation and is involved in many recent algorithmic developments of mathematical morphology. Over the last several years, he has also taught numerous courses on morphology in Europe and the U.S.A.



Pierre Soille was born in Wavre, Belgium, on June 26, 1966. He received the M.S. degree in agricultural engineering from the University of Louvain, Louvain-la-Neuve, in 1988.

Since 1988 he has been appointed by the Belgian National Fund for Scientific Research as a research assistant. He is presently carrying on his research both at the Center for Mathematical Morphology of the School of Mines of Paris and at the University of Louvain. His research interests include digital image processing and mathematical morphology. Within these areas, he has been particularly involved in image segmentation and has recently developed new methods for classifying and segmenting color images. His results are being applied to remote sensing imagery, digital elevation models, digital maps, and 3-D imagery.