Determining watersheds in digital pictures via flooding simulations

Pierre Soille, Luc M. Vincent

Pierre Soille, Luc M. Vincent, "Determining watersheds in digital pictures via flooding simulations," Proc. SPIE 1360, Visual Communications and Image Processing '90: Fifth in a Series, (1 September 1990); doi: 10.1117/12.24211

Event: Visual Communications and Image Processing '90, 1990, Lausanne, Switzerland

Determining watersheds in digital pictures via flooding simulations

Pierre Soille and Luc Vincent

Centre de Morphologie Mathématique, Ecole des Mines
35, rue Saint-Honoré, 77305 Fontainebleau Cedex, France

ABSTRACT

The watershed transformation is a very powerful image analysis tool provided by mathematical morphology. However, most existing watershed algorithms are either too time consuming or insufficiently accurate. The purpose of this paper is to introduce a new and flexible implementation of this transformation. It is based on a progressive flooding of the picture and it works for n-dimensional images. Pixels are first sorted in the increasing order of their gray values. Then, the successive gray levels are processed in order to simulate the flooding propagation. A distributive sorting technique combined with breadth-first scannings of each gray level allow an extremely fast computation. Furthermore, the present algorithm is very general since it deals with any kind of digital grid and its extension to general graphs is straightforward. Its interest with respect to image segmentation is illustrated by the extraction of geometrical shapes from a noisy image, the separation of 3-dimensional overlapping particles and by the segmentation of a digital elevation model using watersheds on images and graphs.

1 INTRODUCTION

In the field of image analysis and more particularly of Mathematical Morphology (MM) [20,21], graphs of gray tone pictures are often regarded as topographic surfaces, the gray level of a pixel standing for its elevation. This representation is used to better appreciate the effects of a given transformation on the image under study. Now, let a drop of water fall on such a topographic surface. According to the law of gravitation, it will flow down along the steepest slope path until it reaches a minimum. The whole set of points of the surface whose steepest slope paths reach a given minimum constitutes the catchment basin associated with this minimum. The *watersheds* are the zones dividing adjacent catchment basins.

As might have been expected, the first watershed algorithms were developed to extract terrain divides from digital elevation models (DEMs) [9]. Unfortunately, these algorithms usually led to poor results, as explained in [8,23]. Meanwhile and independently of these researches in digital topography, the watershed transformation was studied in the field of image processing. It was proposed for the first time by Ch. Lantuéjoul [14] who later improved it jointly with S. Beucher [2,3,4]. The watershed transformation constitutes one of the most powerful tools for contour detection and image segmentation provided by MM [6,27,7]. Many algorithms have already been proposed to extract watersheds from digital pictures. They have proved to be either excessively slow (especially on conventional computers) or—even worse—inaccurate [28]. Here, we introduce a new and versatile implementation of the watershed transformation, which works in various kinds of discrete space. In § 2, we show that the definition of watersheds in terms of flooding simulations is better suited to the formalization of watersheds than the one based on steepest slope paths. In § 3, our implementation is introduced and its two major steps, the sorting and the flooding procedures, are detailed. Then, § 4 discusses the performances and the advantages of the present algorithm. Eventually, § 5 is devoted to some applications which illustrate the efficiency of our algorithm in image segmentation.

2 DEFINITION OF WATERSHEDS VIA FLOODING SIMULATIONS

A careful examination of the watershed concept for digital functions reveals that it is rather difficult to provide a non ambiguous definition. For example, what about flow directions on plateaus or when multiple steepest slope paths occur? Furthermore, when dealing with discrete functions, the concept of steepest slope paths is not well suited to practical implementations of watersheds since there exists no rule to decide which path a drop of water would follow on digital functions [17]. In this section, we show that the definition of watersheds via flooding simulations overcomes these difficulties. It can be considered as an *algorithmic definition* since it allows to develop an accurate and versatile algorithm to compute the watershed transformation (see § 3). Before going further, let us present some basic definitions about the objects considered in this paper. All these definitions are formulated for n-dimensional images, but for greater convenience, the figures and some explanations are given in the 2-dimensional case.

Let us consider an n-dimensional grayscale picture I, whose definition domain is denoted D_I. I is supposed to take discrete values in the range $[0, N]$, N being an arbitrary positive integer:

$$I \left(\begin{array}{lll} D_I \subset \mathbb{Z}^n & \longrightarrow & \{0,1,\ldots,N\} \\ p & \longmapsto & I(p) \end{array} \right) \tag{1}$$

Let G denote the underlying digital grid, which is nothing but a subset of $\mathbb{Z}^n \times \mathbb{Z}^n$. We also denote $N_G(p)$ the set of the neighbors of a pixel p, with respect to G:

$$\forall p \in \mathbb{Z}^n, \qquad N_G(p) = \{p' \in \mathbb{Z}^n, (p,p') \in G\}. \tag{2}$$

Definition 1 *A path P of length l between two pixels p and q in D_I is a $(l+1)$-uplet of pixels $(p_0, p_1, \ldots, p_{l-1}, p_l)$ such that*

$$\begin{cases} p_0 = p \quad and \quad p_l = q, \\ \forall i \in [1,l], (p_{i-1}, p_i) \in G. \end{cases} \tag{3}$$

The length of a given path P is denoted $l(P)$.

Definition 2 *A* minimum *M at altitude h of I is a connected plateau of pixels with value h, whose external boundary pixels have a value strictly greater than h.*

Given these preliminary definitions, we will now present watersheds in terms of flooding simulations [4,3]. Suppose that the minima of the function are sorted in the increasing order of their gray values, and start flooding progressively the basin of the minimum at the lowest altitude. When the water level reaches the altitude of the second minimum, we flood simultaneously the two basins until the elevation of the third minimum is reached, and so forth. In addition, dams are erected at the places where the waters coming from two different basins would merge (see Fig. 1). At the end of this flooding procedure, each minimum is completely surrounded by dams, which delimit its associated catchment basin. The set of dams thus obtained corresponds to the watersheds, and provides a tessellation of I in its different catchment basins.

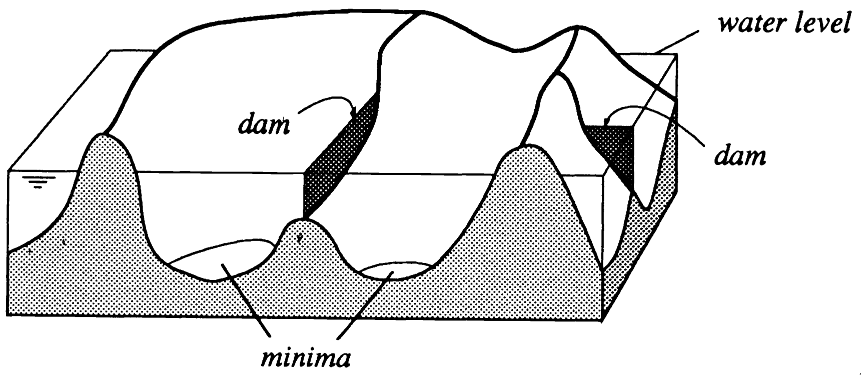


Figure 1: Building dams at the places where the water coming from two different minima would merge

In order to express this flooding simulation in a more formal way, denote $h_{\min}$ the smallest value taken by the grayscale image I on its domain D_I, and $h_{\max}$ the largest. In the following, $T_h(I)$ refers to the threshold of I at level h:

$$T_h(I) = \{p \in D_I, I(p) \leq h\}. \tag{4}$$

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 lower or equal to h:

$$C_h(M) = \{p \in C(M), I(p) \leq h\} = C(M) \cap T_h(I). \tag{5}$$

Lastly, $\text{Min}_h(I)$ refers to the set of points belonging to the minima at altitude h.

We need now to recall the definitions of the *geodesic distance* [15,16] and of the *geodesic influence zones* . Let A be a set that is first supposed to be simply connected.

Definition 3 *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 included in A:*

$$d_A(x,y) = \inf\{l(P), \ \textit{P path between x and y which is included in A}\}. \tag{6}$$

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

Definition 4 *The geodesic influence zone $iz_A(B_i)$ of a connected component B_i of B in A is the set of the points of A whose geodesic distance to B_i is smaller than their geodesic distance to any other component of B:*

$$iz_A(B_i) = \{p \in A, \forall j \in [1,k] \setminus \{i\}, d_A(p,B_i) < d_A(p,B_j)\}. \tag{7}$$

This concept is illustrated in Fig. 2. 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 $\text{SKIZ}_A(B)$:

$$\text{SKIZ}_A(B) = A / \text{IZ}_A(B) \qquad \text{with} \qquad \text{IZ}_A(B) = \bigcup_{i \in [1;k]} \text{iz}_A(B_i). \tag{8}$$

These definitions easily extend to the case where A is not simply connected, nor even connected at all.

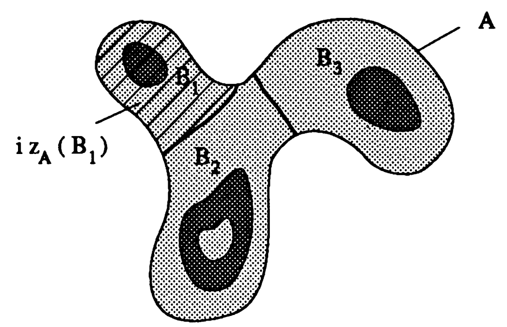


Figure 2: The geodesic influence zone of connected component B_1 inside set A.

To simulate the flooding procedure, we start from the points belonging to the minima at the lowest altitude, i.e., $T_{h_{\min}}(I)$. These points constitute the starting set of our recursion. We thus put:

$$X_{h_{\min}} = T_{h_{\min}}(I). \tag{9}$$

Let us now consider the threshold of I at level $h_{\min} + 1$, i.e., $T_{h_{\min}+1}(I)$. Y being one of the connected components of $T_{h_{\min}+1}(I)$, there are three possible relations of inclusion between Y and $Y \cap X_{h_{\min}}$:

(a) $Y \cap X_{h_{\min}} = \emptyset$: in this case, Y is a new minimum of I. Indeed Y is a plateau at level $h_{\min} + 1$, since

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

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 catchment basin corresponding to the minimum thus discovered will also be progressively filled with water.

(b) $Y \cap X_{h_{\min}} \neq \emptyset$ and is connected: in this case, Y exactly corresponds 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 $h_{\min} + 1$:

$$Y = C_{h_{\min}+1}(Y \cap X_{h_{\min}}). \tag{10}$$

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

$$C_{h_{\min}+1}(Z_i) = \text{iz}_Y(Z_i). \tag{11}$$

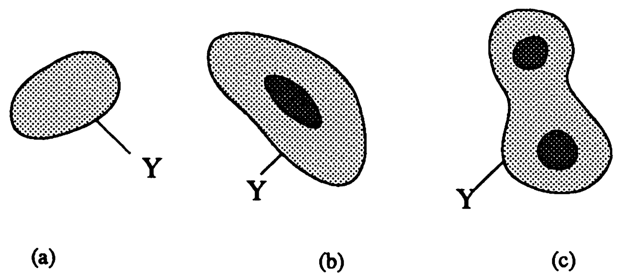


Figure 3: The three possible inclusion relations between Y and $Y \cap X_{h_{\min}}$

These inclusion relationships are illustrated in Fig. 3. Since all the possibilities have been discussed, we take

$$X_{h_{\min}+1} = \text{Min}_{h_{\min}+1} \cup \text{IZ}_{T_{h_{\min}+1}(I)}(X_{h_{\min}}) \tag{12}$$

as the second set of our recursion. This relation holds of course for all levels h, and finally, we obtain the following definition:

Definition 5 (Catchment basins and watersheds by flooding simulations) *The set of the catchment basins of the grayscale image I is equal to the set $X_{h_{\max}}$ obtained after the following recursion:*

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

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

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. 4.

3 PROPOSED ALGORITHM

Our implementation of the watershed algorithm is based on the definition given in § 2. We therefore have to consider the successive thresholds of the image under study, and to compute geodesic influence zones of one threshold inside

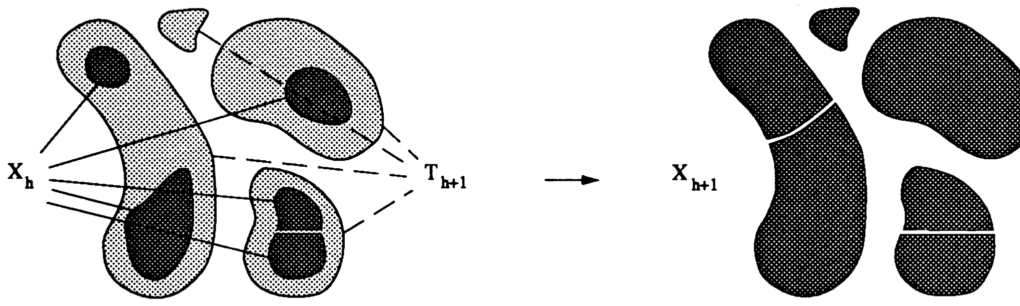


Figure 4: Recursion relation between X_h and X_{h+1}.

the next one as fast as possible. As the image is processed gray level by gray level, only a small number of pixels are effectively processed at each step. Thus, rather than scanning the entire image to modify the value of a few pixels only, one must have a direct access to these pixels. This is achieved by means of a preliminary sorting of the pixels in the increasing order of their gray values. Among the vast number of sorting techniques, the distributive sorting [13] is particularly suited to the present problem. The procedure first determines the frequency distribution of each image gray level. Then, the cumulative frequency distribution is computed. This induces the direct assignment of each pixel to a unique cell in the sorted array. As memory and time requirements to compute frequency distributions are generally negligible compared to those required for images, this sorting technique (linear w.r. to the number of pixels) constitutes one of the best choices to deal with image data.

Once the pixels have been sorted, a fast computation of geodesic influence zones is enabled by a breadth-first scanning of each threshold level. This scanning can be implemented thanks to a queue of pixels. Suppose the flooding has been completed until a given level h. Each catchment basin already discovered—i.e., each catchment basin whose corresponding minimum has an altitude lower or equal to h—is supposed to have a label. Thanks to the prior sorting, the pixels of altitude $h+1$ are accessed directly. These pixels are given a special value, say **mask**. Those among them which have an already labelled 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 § 2). 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. Moreover, a particular value, **wshed**, is assigned to the pixels where two different catchment basins would merge. This exactly corresponds to the construction of dams, as explained in § 2.

The queue which is used is a First-In-First-Out (FIFO) structure: the pixels which are first put into it are those which can be extracted first. Three operations will be performed on this queue :

- *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.
- *fifo_empty():* returns *true* if the queue is empty and *false* otherwise.

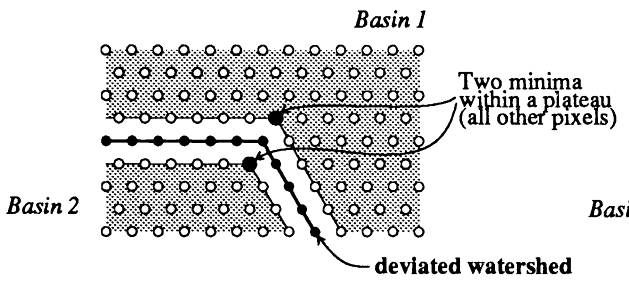
The algorithm itself is detailed below in a pseudo–C language:

```
        # 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 $im_o$ */
        # define inqueue   -3   /* value assigned to the pixels when they are put into the queue */
• – input:    $im_i$, decimal image;
  – output:   $im_o$, image of the labelled watersheds; the labels are 1, 2, 3, etc.
• Initializations:
  – Value init is assigned to each pixel of $im_o$: $\forall p \in D_{im_o}$, $im_o(p)$ = init;
```

```
    – $current\_label \leftarrow 0$;
    – $flag$: boolean variable;

• Sort the pixels of $im_i$ in the increasing order of their gray values (in the range $[h_{\min}, h_{\max}]$).
• For $h \leftarrow h_{\min}$ to $h_{\max}$ {     /* geodesic $SKIZ$ of level $h-1$ inside level $h$ */
    For every pixel $p$ such that $im_i(p) = h$ {     /* thanks to the initial sorting, these pixels are directly accessed */
        $im_o(p) \leftarrow$ mask;
        if there exists $p' \in N_G(p)$ such that $im_o(p') > 0$ or $im_o(p') =$ wshed {     /* ($N_G(p)$ is the set of neighbors of $p$) */
            $im_o(p) \leftarrow$ inqueue;     $fifo\_add(p)$;
        }
    }
    while $fifo\_empty() = false$ {
        $p \leftarrow fifo\_first()$;
        For every pixel $p' \in N_G(p)$ {
            if $im_o(p') > 0$ {     /* i.e., $p'$ belongs to an already labelled basin */
                if ($im_o(p) =$ inqueue or ($im_o(p) =$ wshed and flag = true)) {
                    $im_o(p) \leftarrow im_o(p')$;
                }
                else if ($im_o(p) > 0$ and $im_o(p) \neq im_o(p')$) {
                    $im_o(p) \leftarrow$ wshed;     flag $\leftarrow$ false;
                }
            }
            else if $im_o(p') =$ wshed {
                if $im_o(p) =$ inqueue {
                    $im_o(p) \leftarrow$ wshed;     flag $\leftarrow$ true;
                }
            }
            else if $im_o(p') =$ mask {
                $im_o(p') \leftarrow$ inqueue;     fifo_add($p'$);
            }
        }
    }
    For every pixel $p$ such that $im_i(p) = h$ {     /* check whether new minima have been discovered */
        if $im_o(p) =$ mask {
            $current\_label \leftarrow current\_label + 1$;
            $fifo\_add(p)$;     $im_o(p) \leftarrow current\_label$;
            while $fifo\_empty() = false$ {
                $p' \leftarrow fifo\_first()$;
                For every pixel $p'' \in N_G(p')$ {
                    if $im_o(p'') =$ mask {     $fifo\_add(p'')$;     $im_o(p'') \leftarrow current\_label$;     }
                }
            }
        }
    }
}
```

In the above algorithm, the boolean variable *flag* is used to detect whether the value **wshed**, which is currently assigned to a pixel p, comes from another **wshed**-pixel in the neighborhood of p, rather than from two neighboring pixels with distinct labels. If variable *flag* had not been used, the watersheds would have been either very thick or wrongly located [28,29]. Here, the watershed may still be deviated in such cases as minima embedded in large plateaus (see Fig. 5a). Nevertheless, the configurations yielding important deviations are extremely rare in practice. Furthermore, the algorithm can be adapted to get rid of these deviations: as explained in details in [28] and in [29, chapter 8], this requires the use of to use an additional distance image as a work image. The correct results thus obtained are illustrated by Fig. 5b.



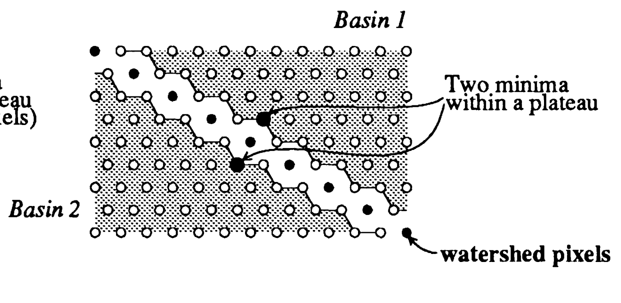


Figure 5: (a) Possible deviation of the watershed lines. In this figure, the neighbors of minimum 1 were put into the queue before those of minimum 2, resulting in a deviation of the watershed line on the side of basin 2. (b) Using an additional distance image allows to obtain exact catchment basins and watersheds.

4 PERFORMANCE EVALUATION

The watershed procedure described above is particularly efficient and, as it often occurs in algorithmics, efficient algorithms have important memory requirements ! Here, the space necessary to hold the cumulative histogram and the queue of pixels is negligible [28]. However, the algorithm also resorts to an array of pointers of pixels, which is of the same size (N) as the initial image. A pointer being generally represented on four bytes, this implies that an array of size $4N$ has to be allocated. This is a lot, but far from being unacceptable in comparison with the random access memory of today computers.

As concerns processing time requirements, the algorithm runs in *linear time* with respect to the number N of pixels in the image under study. Furthermore, each pixel is considered five times at most: two times to construct the sorted array of pointers to pixels, and three times on average during the flooding step. Furthermore, the execution time is nearly independent of the number of gray levels in the image and of the shape of the catchment basins. On a SUN *Sparc Station,* the computation of the watersheds of a 512 x 512 image takes approximately 4.0 seconds. This is extremely fast compared to all existing implementations, which sometimes take more than an hour for the same computation ! Even watershed algorithms implemented on specialized architectures do not run as fast as the one described in this paper.

In addition to its computational efficiency, our algorithm has many other advantages : it works with any kind of digital grid (e.g., 4-, 6-, or 8-connectivity for 2-d images and 6-, 16-, or 26-connectivity for 3-d images...). In order to adapt the program from one grid to the other, it suffices to change the way the neighbors of a given pixel are generated. This allows also one to adapt the procedure to n-dimensional pictures (see for example § 5.2) and to other discrete spaces, like graphs. Watersheds on graphs constitute a really novel tool for image segmentation [26] and are illustrated in § 5.3. Lastly, the present algorithm avoids the classical traps in which most existing implementations fall (namely the plateau problems and the thick "watershed areas" configurations described in [28,29]), so that its accuracy turns out to be remarkable.

5 APPLICATIONS

5.1 Watersheds and image segmentation

When combined with other tools provided by MM, the watershed transformation constitutes a powerful procedure of contour detection and image segmentation [4,6,27,7]. In this section, the methodology is illustrated with the segmentation of a noisy image of geometrical shapes [10] (see Fig. 6). The contours of the four geometrical shapes have to be extracted from this 256 x 256 image. The difficulty is due to its low contrast and its high level of noise. Besides, the simple computation of the watersheds of its morphological gradient [6] leads to a considerable over-segmentation, that is, the searched contours are lost in numerous irrelevant ones. An initial filtering of the original image does not solve the problem. For example, Fig. 6b represents the morphological gradient of the original image processed first with an alternating sequential filter [22], and Fig. 6c shows its corresponding watersheds. To get rid of the over-segmentation, at least two alternatives are available: either remove the irrelevant arcs of the watersheds (post-processing) or modify the gradient function in such a way that its watersheds coincide with the contours of the desired objects (pre-processing). The former alternative generally resorts to *region-growing* algorithms [11,19]. A post-processing technique applied to an over-segmented DEM is presented in § 5.3. As concerns the segmentation of Fig. 6a, we rather resort to the prior modification of the gradient image. It requires some external knowledge in the sense that it necessitates an initial *marking* step. By marker of a region, we mean a connected set of pixels included in this region. Markers are needed for each object and for the background. When the markers have been extracted, a morphological procedure discussed in [18] allows to

1. impose this set of markers as minima of the gradient function while removing all the other minima,
2. preserve the major watershed lines located between the markers.

This transformation is often called *modification of the gradient homotopy.* The computation of the watersheds of this modified gradient then provides the desired segmentation: the catchment basin of the background marker stands for the background itself whereas the watersheds correspond to the contours of the desired objects.

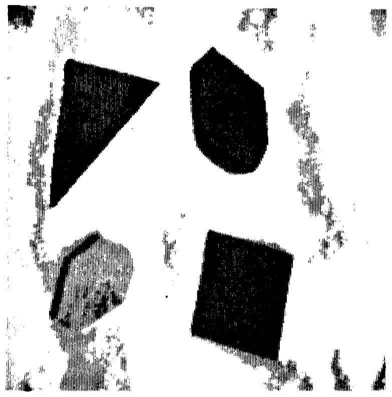
According to this scheme, getting a good segmentation of Fig. 6a involves determining a marker of the background and a marker for each geometrical shape. An immediate solution consists in marking manually these objects, but an automatic extraction of the markers is preferred. In the original image (Fig. 6a) geometrical objects appear darker than the background. Thus, a dilation of this figure followed by a threshold allows to extract the object markers (Fig. 6d). At present, a background marker is still needed. Using the watershed transformation, the highest crest-lines (i.e., white areas) of the original image separating the object markers are determined and serve as background marker. Eventually, the whole set of markers is used to modify the homotopy of the gradient as detailed above. Now, the contours of the geometrical shapes precisely correspond to the watersheds of the modified gradient (Fig. 6f). Thanks to the algorithm introduced in this paper, the whole segmentation procedure described here runs in less than ten seconds on a SUN 4 !

5.2 An application of 3-dimensional watersheds

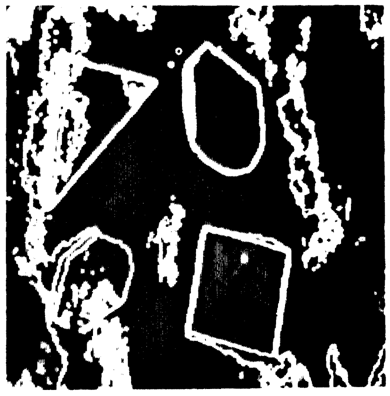
As already mentioned, the present algorithm works for n-dimensional images whatever the connectivity: it suffices to consider the appropriate set of neighbors for each pixel. Its computational efficiency allows to extend the methodology described in § 5.1 for the processing of 3-dimensional images. Applications in the fields of medical imagery and remote sensing are being developed. To illustrate 3-dimensional watersheds, we propose to separate the partially overlapping particles shown in the synthetized image of Fig. 7a. It is a $64 \times 64 \times 64$ binary image displayed as sixty four 64×64 2-dimensional images (from left to right and top to bottom). According to the methodology described in [6], the opposite distance function is first computed, i.e., the function which associates with every feature point the opposite of its distance to the background (see Fig. 7b). The overlapping particles are then separated by the watersheds of the inverse distance function (see Fig. 7c). This binary segmentation procedure has already been applied to practical cases, such as the 2-dimensional segmentation of rice grains [1] or of coffee beans [6].

5.3 Watersheds on graphs: an application to DEMs

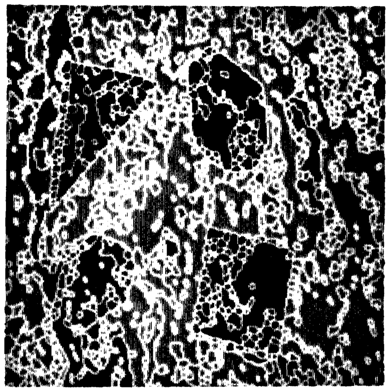
Digital images studied through morphological transformations are usually digitized according to square or hexagonal graphs, but one can well imagine using general graphs and applying the same kind of processings [24,25]. Actually, all the transformations of Euclidean morphology which do not involve any notion of direction easily transpose to these objects [25]. In particular, the definition of watersheds via flooding simulations extends to graphs. In practice, 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 [26].



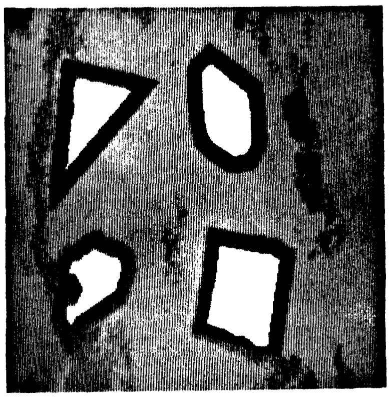
a. Original image I.



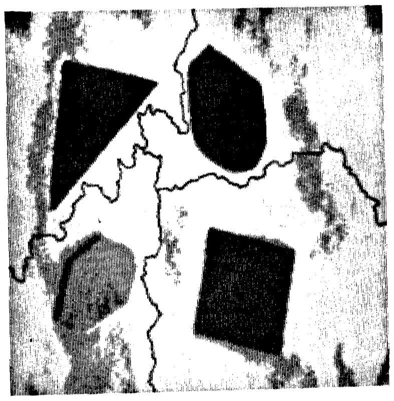
b. Gradient of (filtered) I.



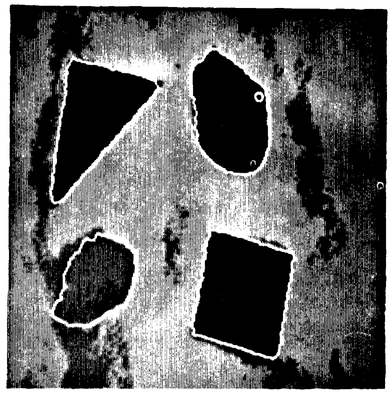
c. Watersheds of Fig. 6b.



d. Object markers.

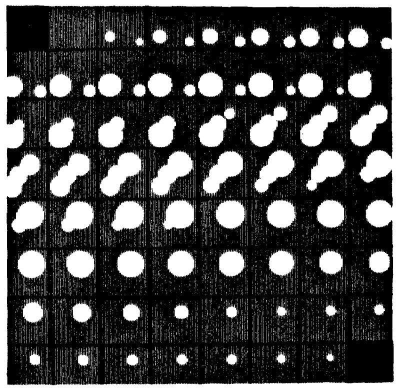


e. Background marker.

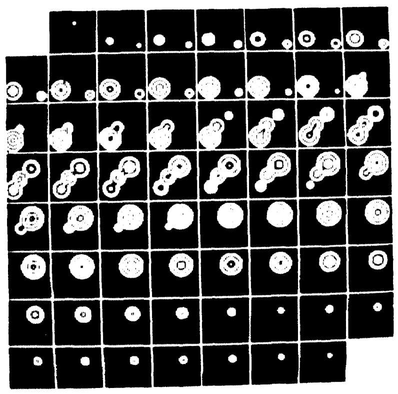


f. Watersheds of modified gradient.

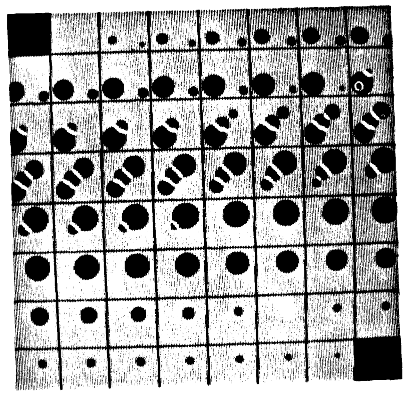
Figure 6: Segmentation of noisy geometrical shapes using the watershed transformation.



a. Original image I.



b. Opposite distance function of I.



c. Resulting segmentation.

Figure 7: Separation of 3-dimensional overlapping particles.

Watersheds on graphs provide a hierarchical decomposition of images [26] which is an alternative to other (morphological) decompositions, such as those based on openings and closings [12]. They seem also to be very promising in morphological region-growing methods. Algorithms based on these ideas have indeed already been successfully used by S. Beucher to segment road images [5,7].

To illustrate watersheds on graphs, the DEM shown in Fig. 8 is processed. It represents a 256 x 256 matrix of elevations having an equal x-y resolution of 50 meters. Applying the watershed transformation to Fig. 8a leads to Fig. 8b. The disappointing over-segmentation observed at this point is due to the numerous minima present *inside* the model. These minima are generally due to artifacts or data errors. In order to get a good partition of the model by using the watershed transformation, pre- or post-processing techniques have to be considered (see § 5.1). Here, rather than modify the homotopy of the DEM to remove its insignificant minima [23], a post-processing technique based on watersheds on graphs is introduced. So as to build a decimal graph from Fig. 8b, a vertex is associated with each catchment basin. The numerical value of a given vertex is determined by the minimum value along the watershed line associated with this vertex. Two vertices are connected by an edge if there exists a minimum calculated previously along their shared watershed lines. By computing the watershed transformation on this graph, we get the segmentation shown in Fig. 8c. It now corresponds to real terrain features.

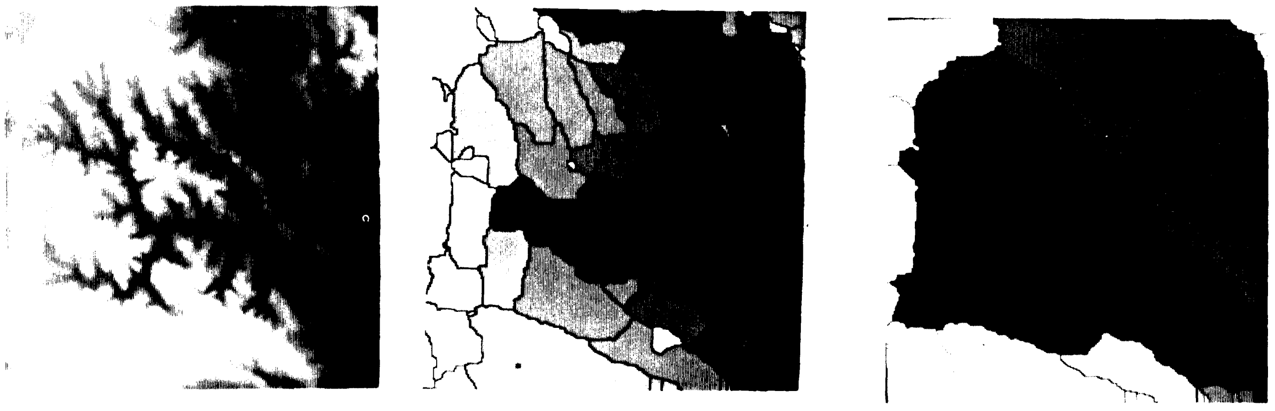

a. Original DEM I. *b. Watersheds of I.* *c. Resulting segmentation.*

Figure 8: Segmentation of a DEM using watersheds on graphs.

6 CONCLUSION

In the field of image processing, every practitioner knows that the computational efficiency of an algorithm is a crucial issue. Up to now, all the existing watershed implementations have been either much too time consuming (even on specialized architecture), or inaccurate. The present algorithm does better than simply getting rid of these drawbacks, since it is also particularly flexible : its adaptation to various kinds of digital spaces is straightforward and the interest of its 2-d, 3-d and graph versions w.r. to picture segmentation have been briefly illustrated in this paper. Beyond the didactic aspect of these examples, our algorithm has already been used for complex segmentation tasks and is expected to contribute to new insights into the use of watersheds. Currently, it is being particularly investigated as a tool for segmenting 3-d and color images.

7 REFERENCES

1. M. Benali, "Du choix des mesures dans les procédures de reconnaissance des formes et d'analyse de texture", Ph.D. Thesis, School of Mines, Paris, 1986.

2. S. Beucher & Ch. Lantuéjoul, "Use of watersheds in contour detection", *Proc. Int. Workshop on image processing, real-time edge and motion detection/estimation*, Rennes, France, September 17–21, 1979.

3. S. Beucher, "La ligne de partage des eaux: comment l'expliquer en termes de transformation fonctionnelle?", *Center for Mathematical Morphology, School of Mines,* Paris, 19 p., May 1981.

4. S. Beucher, "Watersheds of functions and picture segmentation", *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing,* Paris, pp. 1928–1931, May 1982.

5. S. Beucher, "Obstacle detection and vehicle trajectories", *Prometheus Image Processing Meeting,* Paris, May 1989.

6. S. Beucher & L. Vincent, "Introduction aux outils morphologiques de segmentation", *Traitement d'images en microscopie à balayage et en microanalyse par sonde électronique,* ANRT ed., Paris, pp. F1–F43, March 1990.

7. S. Beucher, *Segmentation d'images et morphologie mathématique,* Ph.D. Thesis, School of Mines, Paris, June 1990.

8. 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.

9. F. Doyle, "Digital terrain models: an overview", *Photogrammetric Engineering and Remote Sensing,* Vol. 44, No. 12, pp. 1481–1485, December 1978.

10. M. Grimaud & L. Vincent, "Segmentation de formes géométriques dans une image extrêmement bruitée", *Center for Mathematical Morphology, School of Mines,* Paris, December 1989.

11. R. Haralick & L. Shapiro, "Survey: image segmentation techniques", *Computer Vision, Graphics and Image Processing,* Vol. 29, pp. 100–132, 1985.

12. H.J.A.M. Heijmans & A. Toet, "Morphological sampling", *Int. Report AM-R8913, Center for Mathematics and Computer Science,* Amsterdam, August 1989.

13. Isaac E.J. & R.C. Singleton, "Sorting by address calculation", *Journal ACM* Vol. 3, pp. 169–174, 1956.

14. Ch. Lantuéjoul, *La squelettisation et son application aux mesures topologiques des mosaïques polycristallines,* Ph.D. Thesis, School of Mines, Paris, 1978.

15. Ch. Lantuéjoul & S. Beucher, "On the use of geodesic metric in image analysis", *Journal of Microscopy,* Vol. 121, pp. 39–49, 1981.

16. Ch. Lantuéjoul & F. Maisonneuve, "Geodesic methods in image analysis", *Pattern Recognition,* Vol. 17, pp. 117–187, 1984.

17. F. Maisonneuve, "Sur le partage des eaux", *Center for Mathematical Morphology, School of Mines,* Paris, 1982.

18. F. Meyer & S. Beucher, "Morphological segmentation", Submitted to the *Journal of Visual Communications and Image Representation,* April 1990.

19. O. Monga, "An optimal region growing algorithm for image segmentation", *Int. Journal of Pattern Recognition and Artificial Intelligence,* Vol. 3, No. 4, December 1987.

20. J. Serra, *Image Analysis and Mathematical Morphology,* Academic Press, London, 1982.

21. J. Serra, *Image Analysis and Mathematical Morphology, part II: Theoretical Advances,* edited by J. Serra, Academic Press, London, 1988.

22. J. Serra & L. Vincent, *Lecture notes on morphological filtering,* School of Mines, Cahiers du Centre de Morphologie Mathématique, Vol. 8, 98 p., Paris, 1989.

23. P. Soille & M. Ansoult, "Automated basin delineation from DEMs using mathematical morphology", *Signal Processing,* Vol. 20, pp. 171–182, 1990.

24. L. Vincent,"Mathematical Morphology on Graphs", *Proc. SPIE's Visual Communications and Image Processing 88,* Cambridge, MA, pp. 95–105, November 1988.

25. L. Vincent, "Graphs and Mathematical Morphology", *Signal Processing,* Vol. 16, No. 4, pp. 365–388, April 1989.

26. L. Vincent,"Mathematical morphology for graphs applied to image description and segmentation", *Proc. Electronic Imaging West,* Pasedena, CA, pp. 313–318, April 1989.

27. L. Vincent & S. Beucher, "The morphological approach to segmentation: an introduction", *Center for Mathematical Morphology, School of Mines,* Paris, July 1989.

28. L. Vincent & P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations", Submitted to *IEEE Transactions on Pattern Analysis and Machine Intelligence,* January 1990.

29. L. Vincent, *Algorithmes morphologiques à base de files d'attente et de lacets. Extension aux graphes,* Ph.D. Thesis, School of Mines, Paris, May 1990.