# Segmentation by watersheds: definition and parallel implementation

Jos B.T.M. Roerdink and Arnold Meijster
Institute for Mathematics and Computing Science
University of Groningen
P.O. Box 800, 9700 AV Groningen, The Netherlands
Email: roe@cs.rug.nl, arnold@cs.rug.nl

**Abstract.** The watershed algorithm is a method for image segmentation widely used in the area of mathematical morphology. In this paper we first address the problem of how to define watersheds. It is pointed out that various existing definitions are not equivalent. In particular we explain the differences between the recursive definition, a modification of this definition necessary to avoid relabelling of watershed pixels, and the definition based on shortest paths with respect to a certain grey-value distance function. The sequential implementation of both algorithms is discussed. Finally we sketch parallel implementations of the two watershed algorithms on a MIMD ring-architecture, and a Cray J932 shared memory computer, respectively.

## 1 Introduction

In the field of grey scale mathematical morphology the watershed transform, originally proposed by Digabel and Lantuéjoul, is frequently used for image segmentation [1, 9, 11]. It can be classified as a region-based segmentation approach. The intuitive idea underlying this method is that of flooding a landscape or topographic relief with water. Basins will fill up with water starting at local minima, and at points where water coming from different basins would meet, dams are built. When the water level has reached the highest peak in the landscape, the process is stopped. The set of dams thus obtained partitions the landscape into regions or 'catchment basins' separated by dams. These dams are called watershed lines or simply *watersheds*. A sketch is given in Fig. 1.

## 2 Watersheds by immersion

Although a definition for the continuous case is possible [6, 8], we restrict ourselves here to discrete images. First an algorithmic definition of the watershed is presented following Vincent & Soille [11].

Consider a digital grey scale image $f : D \longrightarrow \mathbb{N}$, where $D \subseteq \mathbb{Z}^2$ is the domain of the image and $f(p)$ denotes the grey value of pixel $p \in D$. Let $G$ denote the pixel grid, i.e. G is a subset of $\mathbb{Z}^2 \times \mathbb{Z}^2$. A *path* $P$ of length $l$ between two pixels $p$ and $q$ is an $l+1$-tuple $(p_0, p_1, \ldots, p_{l-1}, p_l)$ such that $p_0 = p$, $p_l = q$ and $\forall i \in [0, l) : (p_i, p_{i+1}) \in G$. A set of pixels $M$ is called *connected* if and only if for every pair of pixels $p, q \in M$ there exists a path between $p$ and $q$ which only passes through pixels of $M$. A *connected component* is a nonempty connected set of pixels of maximal size. A *regional minimum* (minimum, for short) of $f$ at altitude $h$ is a connected

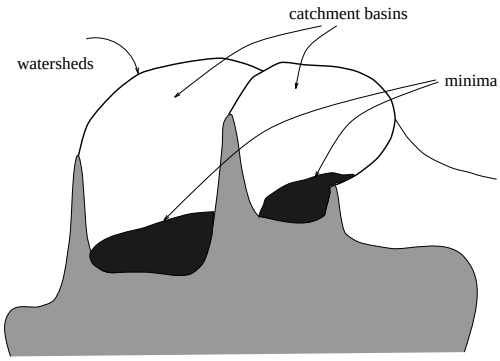


**Figure 1**: *Minima, catchment basins, and watersheds.*

component of pixels $p$ with $f(p) = h$ from which it is impossible to reach a point of lower altitude without having to climb.

Before going to the algorithm for computing watersheds, we need a few more definitions.

**Definition 1** Let $A \subseteq \mathbb{Z}^2$, and $a, b$ two points in $A$. The *geodesic distance* $d_A(a, b)$ within $A$ is the minimum of the lengths of all paths from $a$ to $b$ in $A$. If $B$ is a subset of $A$, define $d_A(a, B) = \text{MIN}_{b \in B}(d_A(a, b))$.

**Definition 2** Let $A \subseteq \mathbb{Z}^2$. Let $B \subseteq A$ be partitioned in $k$ connected components $B_i, i = 1, \ldots, k$. The *geodesic influence zone* of the set $B_i$ within $A$ is defined as

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

**Definition 3** Let $A \subseteq \mathbb{Z}^2$, $B \subseteq A$. The set $IZ_A(B)$ is defined as the union of the geodesic influence zones of the connected components of B, i.e.,

$$IZ_A(B) = \bigcup_{i=1}^{k} iz_A(B_i)$$

The complement of the set $IZ_A(B)$ within $A$ is called the SKIZ ('*skeleton by influence zones*') of $A$:

$$\text{SKIZ}_A(B) = A \backslash IZ_A(B)$$

So the SKIZ consists of all points which are equidistant (in the sense of the geodesic distance) to at least two connected components.

## 2.1 Recursive algorithm

A recursive algorithm for computing the watershed transform was given by Vincent and Soille [10, 11].

The set $T_h = \{p \in D \mid f(p) \leq h\}$ is called the *threshold set* of $f$ at level $h$. Let $h_{min}$ and $h_{max}$ respectively be the minimum and maximum grey level of the digital image. Let $\text{MIN}_h$ denote the union of all regional minima at altitude $h$.

**Definition 4 (Recursive watershed)** Define the following recurrence:

$$X_{h_{min}} = \{p \in D \mid f(p) = h_{min}\}$$

| | | |
|---|---|---|
| 3 | 2 | 2 |
| 3 | 1 | 1 |
| 0 | 1 | 0 |

(a)

| | | |
|---|---|---|
| 3 | 2 | 2 |
| 3 | 1 | 1 |
| A | 1 | B |

(b) $h = 0$

| | | |
|---|---|---|
| 3 | 2 | 2 |
| 3 | W | B |
| A | W | B |

(c) $h = 1$

| | | |
|---|---|---|
| 3 | B | B |
| 3 | W | B |
| A | W | B |

(d) $h = 2$

| | | |
|---|---|---|
| B | B | B |
| A | W | B |
| A | W | B |

(e) $h = 3$

| | | |
|---|---|---|
| 3 | 2 | 2 |
| 3 | 1 | 1 |
| A | 1 | B |

(f) $h = 0$

| | | |
|---|---|---|
| 3 | 2 | 2 |
| 3 | W | B |
| A | W | B |

(g) $h = 1$

| | | |
|---|---|---|
| 3 | B | B |
| 3 | B | B |
| A | W | B |

(h) $h = 2$

| | | |
|---|---|---|
| B | B | B |
| W | B | B |
| A | W | B |

(i) $h = 3$

**Figure 2**: *Watershed on the 4-connected grid. (a): Original image; (b)-(e): labelling steps based on Eq. 1; (f)-(i): labelling steps based on Eq. 2.*

$$X_{h+1} = X_h \cup \text{MIN}_{h+1} \cup (IZ_{T_{h+1}}(X_h) \backslash T_h), \qquad h \in [h_{min}, h_{max}) \tag{1}$$

The *watershed transform* $Wshed(f)$ of $f$ is the complement of $X_{h_{max}}$ in $D$:

$$Wshed(f) = D \cap (X_{h_{max}})^c$$

The recursion (1) is based upon the relation between $X_h$ and $X_{h+1}$. A connected component of $T_{h+1}$ can be either a new minimum, or an extension of the basin $X_h$: in the latter case one computes the geodesic influence zone of $X_h$ within $T_{h+1}$. By adding the '$\backslash T_h$' term in (1), we make sure that at level $h+1$ only pixels with grey value $h+1$ are added to existing basins. It should be noted that the SKIZ is not necessarily connected, and that a set of pixels equally distant from two connected components may be thicker than one pixel. Most algorithms for computing watersheds are direct translations of the recursive relation (1).

**Remark 5** In the original definition of Vincent & Soille [11], Eq. 1 has the form

$$X_{h+1} = X_h \cup \text{MIN}_{h+1} \cup IZ_{T_{h+1}}(X_h), \qquad h \in [h_{min}, h_{max}) \tag{2}$$

This allows that pixels which at earlier levels $h' < h+1$ are equidistant to at least two connected components of the set of basins, and thus are provisionally classified as watershed pixels, are relabelled as belonging to some basin. In (1) this is prevented by the '$\backslash T_h$' term. In fact, the implementation described in [11] based on queue data structures actually corresponds to (1), *not* to (2) (at step $h+1$ only pixels with grey value $h+1$ are put in the queue). A simple example is given in Fig. 2, for a $3 \times 3$ discrete image on the square grid with 4-connectivity. The labelling according to (1) is shown in Fig. 2(b)-(e). There are two local minima (the zeroes), so there

will be two basins whose pixels are labelled $A, B$. Watershed pixels are labelled by $W$. Figure 2(f)-(i) shows the phenomenon of relabelling of watershed pixels when using (2): the pixel in the second row, second column is first labelled $W$, then $B$. When using the modified definition (1) this pixel remains labelled as $W$.

## 2.2 Sequential implementation of the recursive algorithm

Consider the discrete image as a graph $(F, E)$ with nonnegative vertex values, where $F$ is a subset of the square grid, with the set of edges $E$ defined by the connectivity of the grid. The grey value at a node $v$ is denoted by $f(v)$. Also, assume for the moment that all neighbouring pixels in the image have different grey values. The implementation of the recursive definition can be easily formulated on such a graph [4,11]. The algorithm assigns a label *lab* to each minimum and its associated basin by iteratively flooding the grid using a breadth first algorithm. Initially, all nodes with grey level $h$ are given the label MASK. If some node $v$ is adjacent to two or more different basins, it is marked a watershed node by the label WSHED. If the node can only be reached from nodes which have the same label the node is merged with the corresponding basin. Nodes which at the end still have the value MASK are new minima, and get a new label. If the restriction of distinct neighbouring grey values does not hold, additional processing is necessary to partition the plateaus (regions of constant grey value) into regions belonging to different minima. This corresponds to the computation of influence zones during every iteration of the algorithm.

## 2.3 Alternative algorithm

A straightforward parallel implementation of the above algorithm is difficult when plateaus occur. Therefore, an alternative approach was developed, in which the image is first transformed to a graph with distinct neighbour values. Then the graph algorithm described above is directly applicable. This observation leads us to a three-stage watershed algorithm [4].

**Step 1.** Transform the image $f$ to a directed valued graph $f^* = (F, E)$, called the *components graph of* $f$. The vertex set $F$ represents maximal connected sets of pixels with the same grey values, called 'level components' or 'flat zones'. A pair of level components $(v, w)$ is an element of the edge set $E$ if and only if $\exists(p \in v, q \in w : (p, q) \in G \wedge f(p) < f(q))$, cf. Fig. 3.

**Step 2.** Compute the watershed of the directed graph, resulting in a graph with labelled vertices.

**Step 3.** Transform the labelled graph back to an image. Pixels corresponding to a watershed node are coloured white, the other pixels black. This yields a binary image with plateaus representing watersheds of the original image. Thin watersheds can be obtained by computing a skeleton of this image, for which different skeleton algorithms can be used.

## 2.4 Parallelization of the alternative algorithm

The runtime performance of the sequential algorithm proposed in the previous subsection turns out to be approximately the same as that of the algorithm described in [11]. However, since all pixels which are in the same level component are clustered in one single node of the components graph, we can decide locally whether a

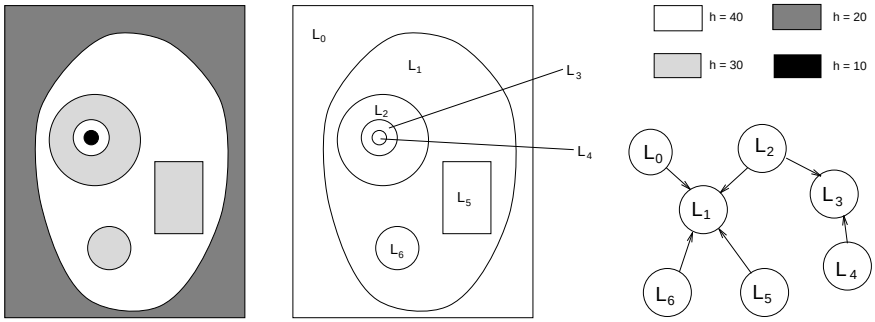


**Figure 3**: *(a) input image. (b) labelled level sets. (c) components graph.*

node is a watershed node by looking at the adjacent nodes. In contrast to the traditional algorithm, the graph algorithm can be parallelized rather easily, see Meijster & Roerdink [3, 4]. Assume a ring network of $N$ processors, where each processor can communicate directly with both neighbouring processors. The programming style we use is called SPMD (single program multiple data), meaning that every processor runs exactly the same program, performing operations on its own data space.

**Level components labelling.** Labelling of level components is performed by a single processor on the entire image. After labelling, this processor distributes the input image and the labelled image over the processors in the network. To each processor is assigned an (approximately) equal slice of consecutive scanlines. Consecutive slices are assigned to neighbouring processors, with one scanline overlap so that it can be decided whether level components are shared with neighbouring processors.

**Parallel watershed of a graph.** After labelling every processor builds a local components graph for its own slice of the image. Since some level components are shared between several processors the graphs on the processors are not disjoint. Next every processor performs an adapted version of the flooding algorithm, taking special care of vertices which are shared between two or more processors. At the end of the flooding process each processor transforms its local components graph back to an image slice, as in the sequential case.

# 3 Watershed definition by shortest paths

Meyer [6] gives a definition of the watershed of a continuous (see also [8]) or digital grey value image in terms of shortest paths with respect to a certain distance function. We confine ourselves here to the digital case.

The *lower slope* of a grey value image $f$, which is the maximal slope linking a pixel $p$ to any of its neighbours of lower altitude, is defined as

$$LS(p) = \underset{q \in \{p\} \cup N_E(p)}{\mathrm{MAX}} (f(p) - f(q)),$$

where $N_E(p)$ is the set of neighbours of pixel $p$ on the grid $E$. Here we restrict ourselves to the case where distances between neighbours all equal 1. This can be generalized, e.g. to chamfer distances [6]. The *cost* for walking from pixel $p$ to a

neighbouring pixel $q$ is defined as

$$cost(p,q) = \begin{cases} LS(p) & \text{if } f(p) > f(q) \\ LS(q) & \text{if } f(p) < f(q) \\ \frac{LS(p)+LS(q)}{2} & \text{if } f(p) = f(q) \end{cases}$$

Denote the set of all paths from $p$ to $q$ by $p \rightsquigarrow q$. The *topographical distance* between $p$ and $q$ *along a path* $P = (p_0, \ldots, p_{l(P)})$ of length $l(P)$ is defined as

$$T_f^P(p,q) = \sum_{i=0}^{l(P)-1} cost(p_i, p_{i+1}).$$

The *topographical distance* between points $p$ and $q$ is the minimum of the topographical distances along all paths between $p$ and $q$:

$$T_f(p,q) = \underset{P \in p \rightsquigarrow q}{\text{MIN}} T_f^P(p,q).$$

The topographical distance between a point $p \in D$ and a set $A \subseteq D$ is defined as $T_f(p,A) = \text{MIN}_{a \in A} T_f(p,a)$. It is assumed that values of pixels in all the local minima of $f$ have been reset to 0.

The set of lower neighbours $p'$ of $p$ (i.e. $f(p) \geq f(p')$), for which the slope $f(p) - f(p')$ is maximal is denoted by $\Gamma(p)$. We call $\pi = (p_0 = p, p_1, \ldots, p_n = q)$ a *path of steepest descent* from $p$ to $q$ if, for each $i = 0, \ldots, n-1$, $p_{i+1} \in \Gamma(p_i)$. The topographical distance has the following property, on which the watershed definition crucially depends.

**Proposition 6** *There exists a path $\pi$ of steepest descent from $p$ to $q$ if and only if $T_f(p,q) = f(p) - f(q)$. In all other cases, $T_f(p,q) > f(p) - f(q)$.*

This proposition implies that lines of steepest descent are the geodesics (shortest paths) of the topographical distance function. In fact, $T_f$ is not exactly a distance, since for pixels $p, q$ in the interior of a plateau $T_f(p,q) = 0$. So an auxiliary order relation is necessary to separate them.

Let $(m_i)_{i \in I}$ be the collection of minima of $f$. The *catchment basin* of a minimum $m_i$, denoted by $CB(m_i)$, is defined as the set of points $p \in D$ that are topographically closer to $m_i$ than to any other minimum $m_j$:

$$CB(m_i) = \{p \in D \mid \forall j \in I \backslash \{i\} : T_f(p,m_i) < T_f(p,m_j)\}.$$

The watershed of a function $f$ is the set of points of its domain which do not belong to any catchment basin:

$$Wshed(f) = D \cap (\cup_{i \in I} CB(m_i))^c$$

## 3.1 Computation of the watershed based on Dijkstra's algorithm

In order to obtain the watershed of an image, the distance of each pixel to each minimum has to be computed. Using the function *cost* as the weight function associated with the edges of the grid, Dijkstra's algorithm [2] for finding shortest paths in a graph can be used to compute the topographical distances.

Given an undirected graph $G = (V, E)$, and a weight function $w : E \to \mathbb{N}$, that assigns a length to each edge of the graph, the goal is to find for each $v \in V$ the length of the shortest path from a source node $s$ to $v$. In Dijkstra's algorithm, one initializes for each node $v \in V \backslash \{s\}$ the distance $d[v]$ between $v$ and $s$ to infinity,

while the distance $d[s]$ between $s$ and itself is set to zero. Next, a wavefront starting in $s$ is propagated through the graph along the edges. During the propagation one keeps track of the distance the wavefront has travelled so far. When a node is reached by the wavefront and the distance travelled is smaller than the current value stored in this node, this value is updated. Propagation stops when all nodes of the graph have been reached.

Instead of applying this algorithm separately for each minimum, one may modify the function $d$ in Dijkstra's algorithm as follows [5]. Store for each $v \in D$ in the first coordinate of $d[v]$ the index of the nearest minimum, and in the second coordinate the distance to this minimum. The range of the function $d$ is $\mathcal{R} = (I \cup \{\text{WSHED}\}) \times \mathbb{N}$. This leads to the implementation in Algorithm 1. In each minimum a wavefront is initiated, labelled with the index of the minimum it started in. If wavefront $i$ reaches a node $v$ after it has propagated over a distance $l$, and $l$ is less then the value of the second coordinate of $d[v]$ (denoted by $snd(d[v])$), the value $l$ is placed in the second coordinate of $d[v]$, while the first coordinate $fst(d[v])$ is set to $i$. If a node $v$ is reached by another wavefront that has propagated over the same distance, the first coordinate of $v$ is set to the artificial value WSHED, designating that $v$ is a watershed pixel.

**Algorithm 1** Sequential watershed algorithm based on shortest paths

```
procedure SeqWshed (E : D × D; cost : E → ℕ; var d : D → ℛ);
var u : D;
begin forall v ∈ D do d[v] := (0, ∞);
      forall i ∈ I do
         forall v ∈ m_i do d[v] := (i, 0);
      while D ≠ ∅ do
      begin u := GetMinDist(D); (* find u ∈ V with smallest d-value *)
            D := D\{u};
            forall v ∈ D with (u, v) ∈ E do
               if snd(d[u]) + cost[u, v] < snd(d[v])
               then d[v] := (fst(d[u]), snd(d[u]) + cost[u, v]);
               else if snd(d[u]) + cost[u, v] = snd(d[v])
                    then d[v] := (WSHED, snd(d[v]));
      end
end;
```

In Fig. 4 an example is given of the computation of the watershed of a digital image via topographical distances. For comparison we also show the result of the Vincent-Soille definition (2), as well as our modification (1). Note that all three results are different.

**Implementation using ordered queues.** The function $GetMinDist$ in Algorithm 1 can be implemented such that it has time complexity which is linear in the number of pixels of the image. This can be realized with a priority queue of fifo-queues, also called a 'hierarchical' or 'ordered' queue [1].

With each fifo-queue is associated the distance that a wavefront still has to travel before it will reach the pixels in this queue. These distances are used as the priority values in the priority queue. Pixels which are located in the interior of a plateau are ordered in this queue according to another distance function which measures how far pixels are away from the boundary of the plateau. It is clear that, using this data structure, *GetMinDist* runs in $O(1)$ time, since it simply returns (and removes) the pixel at the front of the first fifo-queue in the priority queue. Insertion in the

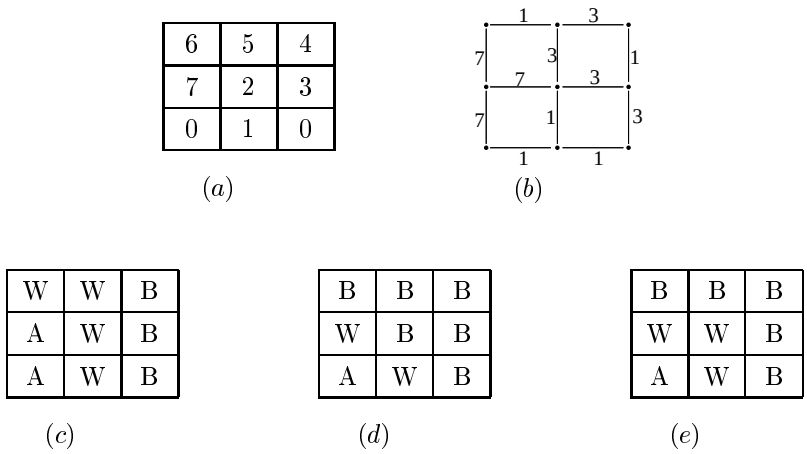


**Figure 4**: *Watershed on the 4-connected grid. (a): original; (b): cost function on the edges; (c): watershed according to topographical distances; (d): watershed according to Eq. 2; (e): watershed according to Eq. 1.*

queues can also be done in $O(1)$ time, if we keep track of the first and last position in each fifo-queue.

### 3.2 Parallelization

The algorithm has been implemented on a Cray J932, a shared memory computer [5]. Computing the lower slope and the cost function in parallel is easy, since the computations are independent for different pixels. Detection of minima is a time-consuming step, since local minima can be huge plateaus, and as a result one cannot decide whether a pixel is located in a regional minimum by just inspecting its value and those of its neighbours. The algorithm for detecting local minima was adapted from [7], but research on faster algorithms is currently going on. Computation of the watershed on the graph can also easily be parallelized. Each processor computes the catchment basins of an (approximately) equal number of minima. Since we use shared memory, concurrent references to the same memory locations have to be synchronized using critical sections.

**Performance Results.** The speedup for computing lower slope and cost function is almost linear in the number $N$ of processors. The same holds for minima detection, although the influence of concurrent references to the same memory locations starts to play a major role if we use many processors. If the number of minima is smaller than $N$, no speed is gained by using more processors. In practice, however, the number of minima is usually much larger than $N$. Load imbalance as a result of different sizes of the catchment basins is a much more serious cause of decrease in speedup, see the timing results in [5].

## 4 Conclusions

We have reviewed various existing definitions of watersheds based on recursive thresholding [11] or shortest paths with respect to a certain distance function [6]. We also made a modification of the definition in [11] to avoid 'leaking watersheds', i.e., relabelling of watershed pixels. Some examples were presented which show that

the various definitions in principle give different answers. Of course, in practical applications the differences may be small. For both watershed definitions, a sequential and a parallel implementation was described. The original watershed algorithm [11] is very hard to parallelize because of its inherently sequential nature. A parallel implementation of this algorithm was based upon splitting the computation in three consecutive stages involving the transformation to a components graph. The watershed on this graph is easy to parallelize because of its local nature [4]. The distance-based definition [6] allows computing watersheds in parallel using a simple adaptation of Dijkstra's shortest path algorithm [5]. The problem of load imbalance due to unequal sizes of catchment basins will be the subject of future study.

# References


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

[2] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959).

[3] Meijster, A., Roerdink, J. B. T. M.: The implementation of a parallel watershed algorithm. In: van Vliet J.C. (ed.): Proc. Computing Science in the Netherlands, 27-28 November, Utrecht. Amsterdam: Stichting Mathematisch Centrum 1995 (pp. 134–142).

[4] Meijster, A., Roerdink, J. B. T. M.: A proposal for the implementation of a parallel watershed algorithm. In: Hlaváč V., Šára R. (eds.): Computer Analysis of Images and Patterns. New York Heidelberg Berlin: Springer-Verlag 1995 (Lecture Notes in Computer Science, vol. 970, pp. 790–795).

[5] Meijster, A., Roerdink, J. B. T. M.: Computation of watersheds based on parallel graph algorithms. In: Maragos P., Shafer R. W., Butt M. A. (eds.): Mathematical Morphology and its Applications to Image and Signal Processing. Dordrecht: Kluwer Acad. Publ. 1996 (pp. 305–312).

[6] Meyer, F.: Topographic distance and watershed lines. Signal Processing 38, 113–125 (1994).

[7] Moga, A.N., Viero, T., Dobrin, B.P., Gabbouj, M.: Implementation of a distributed watershed algorithm. In: Serra J., Soille P. (eds.): Mathematical Morphology and its Applications to Image Processing. Dordrecht: Kluwer Acad. Publ. 1994 (pp. 281–288).

[8] Najman, L., Schmitt, M.: Watershed of a continuous function. Signal Processing 38, 99–112 (1994).

[9] Serra, J.: Image Analysis and Mathematical Morphology. New York: Academic Press 1982.

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

[11] Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(6), 583–598 (1991).