PERGAMON

Pattern Recognition 33 (2000) 907–916

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

An efficient watershed algorithm based on connected components

A. Bieniek*, A. Moga

Institute for Computer Science, Albert-Ludwigs-Universität Freiburg, Chair of Pattern Recognition and Image Processing, Universitätsgelände Flugplatz, D-79085 Freiburg i.Br., Germany

Accepted 27 July 1999

Abstract

In this paper, a formal definition and a new algorithmic technique for the watershed transformation is presented. The novelty of the approach is to adapt the connected component operator to solve the watershed segmentation problem. The resulting algorithm is independent of the number of grey-levels, employs simple data structures, requires less error prone memory management, and issues a lower complexity and a short running time. However, the algorithm does not modify the principle of the watershed segmentation; the output result is the same as that of using any traditional algorithm which does not build watershed lines. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Watersheds; Image segmentation; Connected components

1. Introduction

The watershed transformation is a popular image segmentation algorithm for grey-scale images. The traditional watershed algorithm simulates a flooding process. Thus, an image is identified with a topographical surface, in which the altitude of every point is equal to the grey level of the corresponding pixel. Holes are then pierced in all regional minima of the relief (connected plateaus of constant altitude from which it is impossible to reach a location of lower altitude without having to climb). Sinking the whole surface slowly into a lake, water springs through the holes and progressively immerses the adjacent walls. To prevent streams of water coming from different holes to intermingle, a hinder is set up at the meeting locations. Once the relief is completely covered by water, the set of obstacles depicts the watershed image.

Various definitions of watersheds have been proposed in the literature for both digital and continuous spaces [1–5]. Most algorithms label each pixel with the identifier of its catchment basin and no watershed lines are explicitly constructed. In this paper, we present a new algorithm to perform the watershed transformation which does not construct watershed lines. Let us mention that the algorithm produces the same segmentation result as the techniques in Refs. [1–3], but a simpler algorithmic construction and hence a lower complexity is issued.

The traditional implementation of the watershed segmentation algorithm simulates the flooding process over the image surface. First, regional minima are detected and uniquely labelled with integer values. Then, the algorithm simulates the flooding process using a hierarchical queue [1,2]. Such a queue consists of H first-in–first-out (FIFO) queues, one queue for each of the H grey levels in the image; the size of the hth FIFO queue is given by the number of pixels in the image having the grey-level h. This data structure is used to impose the order of accessing pixels to operate on. Initially, the hierarchical queue contains the seeds for the flooding, i.e. the minima which are at the interface line between the regional minima and

* Corresponding author. Tel.: +49-611-714-6736; fax: +49-611-714-6736.
E-mail address: andreas.bieniek@gmx.de (A. Bieniek)

the non-minima pixels; a pixel of grey-level h is introduced into the hth FIFO queue of the hierarchical queue. The hierarchical queue is then parsed from the lowest grey level to the highest one. A pixel p, removed from the queue, propagates its label to all its neighbours which have not been already reached by flooding. The latter are introduced, at their turn, into the queue of their grey level. The FIFO order of serving the candidate pixels within the same connected plateau ensures the synchronous breadth-first propagation of labels coming from different minima inside a plateau. When all FIFO queues have been emptied, each pixel was appended to a single region and the procedure stops. The image of labelled pixels depicts the segmentation result. For a simple input image, the flooding process, illustrated by arrows, is shown in Fig. 1.

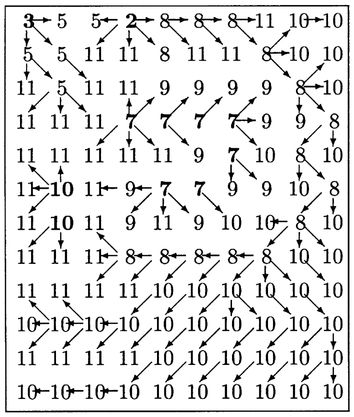
Following the flowing scheme in Fig. 1, we developed a formalism which allows us to determine for every pixel p a neighbouring pixel q from which p will be flooded. As in other watershed formalisms, q may not be unique. In such a case, q is arbitrarily chosen among the potential pixels. Having this local "connectivity" relation, between neighbouring pixels which pertain to the same catchment basin, embedded into the image (technique also known as arrowing [2]), the result is nothing but a directed graph, for which the connected components [6,7], must be computed. The novelty of our approach is to effectively apply the connected component operator [6,7], to compute catchment basins. Preliminary results for this approach have been published in Ref. [8] and a modified version, which constructs watershed pixels according to the definitions of Meyer [2], has been recently found in Ref. [9]. However, the connected component technique has been previously used in Refs. [10–12] for the parallelization of the watershed transformation.

The paper is organized as follows. In Section 2, a formal definition of watersheds in digital space for images without non-minima plateaus is presented. In Section 3, our formalism is compared with Meyer's definition of watersheds [2]. Further on, the proposed definitions lead to a connected component-like watershed algorithm for images without non-minima plateaus in Section 4. The definitions are extended for images with non-minima plateaus in Section 5, whereas the corresponding algorithm follows in Section 6. In Section 7, the complexity analysis of the algorithm and timing results are presented, while conclusions are drawn in Section 8.

2. Segmentation based on local conditions

In this section, we present a definition of the watershed segmentation for images without non-minima plateaus. The reason to consider just such images is that each pixel has at least one lower neighbour, except minima pixels, i.e. the image is lower complete [3,11]. The extension to include images with non-minima plateaus is presented in Section 5.

Let $f(p)$ be a function of grey levels, representing a digital image with the domain $\Omega \subset \mathbb{Z}^2$. Each pixel $p \in \Omega$ has a grey level $f(p)$ and a set of neighbouring pixels $p' \in N(p)$ with a distance function dist (p, p') to each neighbour. In most cases, a 4- or 8-square neighborhood is used with a constant distance of 1 to all neighbouring pixels. Before giving our definition of watershed segmentation and catchment basins, some preliminary definitions are introduced:



a

1	1	2	**2**	2	2	2	2	2	2
1	1	2	2	2	2	2	2	2	2
1	1	1	3	3	3	3	3	2	2
1	1	1	**3**	**3**	**3**	**3**	3	2	2
4	4	3	3	3	3	**3**	3	2	2
4	**4**	3	3	**3**	**3**	3	3	2	2
4	**4**	2	3	3	3	3	2	2	2
4	4	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2

b

Fig. 1. Sequential watershed: (a) flooding the input image (b) output image of labels.

Definition 1 (*Lower slope*). The lower slope of a pixel p is given by the maximal ratio $(f(p)-f(p'))/\mathrm{dist}(p,p')$ to all its neighbouring pixels of lower grey level than itself:

$$\mathrm{LS}(p) = \max_{\forall p' \in N(p)} \left(\frac{f(p)-f(p')}{\mathrm{dist}(p,p')} \middle| f(p') \leqslant f(p) \right)$$

and is not defined for the case $f(p') > f(p), \forall p' \in N(p)$ [2].

The lower slope defines the maximum steepness from a pixel to its lower neighbours. Each pixel in the image, excluding minima, has a steepest neighbourhood:

Definition 2 (*Steepest neighbourhood*). $\forall p \in \Omega$, NLS (p) is the set of pixels $p' \in N(p)$ defined as follows:

$$\mathrm{NLS}(p) = \left\{ p' \in N(p) \middle| \frac{f(p)-f(p')}{\mathrm{dist}(p,p')} = \mathrm{LS}(p), \quad f(p') < f(p) \right\}.$$

For the case $\mathrm{dist}(p,p') = 1, \forall p' \in N(p)$, the set becomes

$$\mathrm{NLS}(p) = \left\{ p' \in N(p) \middle| f(p') = \min_{\forall p'' \in N(p)} f(p''), f(p') < f(p) \right\}.$$

A similar definition exists also in Ref. [2]. Let us note that in an image without non-minima plateaus, $\mathrm{NLS}(p) \neq \emptyset \; \forall p \in \Omega, p$ is not a minimum. In addition, the path of steepest descent from a pixel p down to a regional minimum m_i will pass only pixels of the set $\bigcup_{p'=p}^{m_i} \mathrm{NLS}(p')$.

A different definition than in Ref. [2] for a catchment basin and watershed segmentation based on the steepest neighbourhood is next given.

Definition 3 (*Watershed segmentation for images without non-minima plateaus*). For any image without non-minima plateaus, a segmentation is called watershed segmentation if every regional minimum m_i has an unique label $L(m_i)$ and, for every pixel $p \in \Omega$ with $\mathrm{NLS}(p) \neq \emptyset$, the following condition holds:

$\exists p' \in \mathrm{NLS}(p)$ such that $L(p) = L(p')$.

Definition 4 (*Catchment basin*). For the watershed segmentation defined above, a catchment basin $\mathrm{CBLC}(m_i)$ of the regional minimum m_i is the set of pixels with the label $L(m_i)$:

$$\mathrm{CBLC}(m_i) = \{p \mid L(p) = L(m_i)\}.$$

$\mathrm{CBLC}(p \rightarrow m_i)$ denotes the catchment basin of m_i containing pixel p. The definition of watershed segmentation and catchment basin does not imply uniqueness of the segmentation result; in general, an image may have several valid watershed segmentations.

3. Relation to the traditional definition of the watershed segmentation

In this section, Meyer's formalism [2] is presented and compared with our definitions in Section 2. From functions on continuous space, Meyer derived a formal definition of catchment basins for the digital space [2] as follows:

Definition 5 (*Cost function based on lower slope*). The cost for walking on the topographical surface from position p_{i-1} to $p_i \in N(p_{i-1})$ is given by

$$\mathrm{cost}(p_{i-1}, p_i) = \begin{cases} \mathrm{LS}(p_{i-1})\,\mathrm{dist}(p_{i-1}, p_i), & f(p_{i-1}) > f(p_i), \\ \mathrm{LS}(p_i)\,\mathrm{dist}(p_{i-1}, p_i), & f(p_{i-1}) < f(p_i), \\ \frac{1}{2}(\mathrm{LS}(p_{i-1}) + \mathrm{LS}(p_i))\,\mathrm{dist}(p_{i-1}, p_i), & f(p_{i-1}) = f(p_i). \end{cases}$$

Definition 6 (*Topographical distance*). The topographical distance between two pixels p and q of an image is the minimal π-topographical distance among all paths π between p and q inside the image:

$$\mathrm{TD}_f\,(p,q) = \inf \mathrm{TD}_f^{\pi}\,(p,q).$$

where $\mathrm{TD}_f^{\pi}\,(p,q) = \sum_{i=2}^{n} \mathrm{cost}(p_{i-1}, p_i)$ is the π-topographical distance of a path $\pi = (p = p_1, p_2, \ldots, p_n = q)$, such that $\forall i, p_i \in N(p_{i-1})$ and $p_i \in \Omega$.

Definition 7 (*Catchment basin based on topographical distance*). A catchment basin $\mathrm{CBTD}(m_i)$ of a regional minimum m_i is the set of pixels $p \in \Omega$ closer to m_i than to any other regional minimum m_j, according to the topographical distance and the grey levels of the minima:

$$\mathrm{CBTD}(m_i) = \{p \mid f(m_i) + \mathrm{TD}_f\,(p, m_i) < f(m_j) + \mathrm{TD}_f\,(p, m_j) \; \forall j \neq i\}.$$

Based on these definitions, Meyer presents the following theorem (Proposition 5 in Ref [2]):

Theorem 8. *The topographical distance between a pixel p and the regional minimum m_i in the depth of its catchment basin is minimal and equal to $f(p) - f(m_i)$ and the geodesic line between them is a line of steepest descent.*

The reversal of Theorem 8 states that a path of steepest descent ensures a minimal cost. The construction of the catchment basins is reduced to a problem of finding a shortest path between each pixel and a regional minimum. The relation between Definitions 4 and 7 is stated in the following theorem:

Theorem 9. *A catchment basin based on the topographical distance, as in Definition 7, is a subset of the catchment* *basin in Definition* 4, *based on the local condition given in Definition* 3.

Proof. The formal construction of the catchment basin according to Definition 3 can be described as a recursion. The process starts with the set of pixels belonging to the regional minimum m_i. All these pixels are labeled with $L(m_i)$. At each step, unlabelled pixels, whose neighbours of steepest descent are already in the set, are appended to the set. The recursion ends when no more pixels can be incorporated into the set.

$$\mathrm{CBLC}^0(m_i) = m_i,$$

$$\mathrm{CBLC}^{k+1}(m_i) = \mathrm{CBLC}^k(m_i) \cup \Delta\,\mathrm{CBLC}^k(m_i),$$

$$\Delta\,\mathrm{CBLC}^k(m_i) = \{p \mid \forall j,\, p \notin \mathrm{CBLC}^k(m_j) \text{ and } \exists\, p' \in \mathrm{NLS}(p),\ p' \in \mathrm{CBLC}^k(m_i)\}.$$

Each newly inserted pixel p has a neighbour p' being part of the catchment basin $\mathrm{CBLC}^k(m_i)$. Thus, the local condition of Definition 3 is valid for each p. The proof proceeds as follows:

$$p' \in \mathrm{NLS}(p) \Rightarrow \mathrm{LS}(p) \overset{\text{def } 2}{=} \frac{f(p) - f(p')}{\mathrm{dist}(p, p')}$$

$$\Rightarrow \mathrm{LS}(p) \cdot \mathrm{dist}(p, p')$$

$$= f(p) - f(p') \overset{\text{def } 5}{\Rightarrow} \mathrm{cost}(p, p') = f(p) - f(p').$$

According to Theorem 8, one recursion step adds only those pixels p building paths of steepest descent down to $\mathrm{CBLC}^k(m_i)$ with the minimal cost $f(p) - f(p')$, $p' \in \mathrm{CBLC}^k(m_i)$. After the recursion is finished, all paths between pixels of the catchment basin and its minimum are paths of steepest descent. Therefore, it is not possible to construct a steeper path to a different minimum m_j. However, there might exist another steepest path, of equal cost as to m_i, to a different regional minimum m_j. In this case, the pixel is a watershed pixel according to Definition 7. This proves that $\mathrm{CBTD}(m_i)$ is a subset of $\mathrm{CBLC}(m_i)$. □

The difference between Definitions 7 and 4 is the treatment of pixels which have the steepest paths of equal cost in more than one minimum. According to Definition 7, these pixels are watershed pixels. Following Definition 4, based on the local condition, such a pixel is assigned to one of the minima, m_i, to which it is connected by a steepest path and for which the condition $\exists p' \in \mathrm{NLS}(p), L(p') = L(m_i)$ holds. All possible assignments result in a valid watershed segmentation. In such cases, most watershed algorithms which do not construct watershed lines, including algorithms described in Ref. [2], choose one of the possible assignments given by Definition 4. Therefore, these algorithms are consistent with the definition.

Algorithms which follow Definition 7 may result in thick watershed lines and watershed areas. In other cases, no watershed line is visible between neighbouring regions. Algorithms which avoid thick or zero-width watershed lines are not consistent with Definition 7. According to Definition 4, every pixel belongs to a catchment basin, but the segmentation result is scanning order dependent.

4. A simple algorithm for lower complete images

The idea of the proposed algorithm originates in the connected components problem [13–15]. The goal is to label each pixel with the representative of the region it belongs to. Choosing, for every pixel p, a neighbour from the set $\mathrm{NLS}(p)$ as predecessor, a directed graph results. However, minima pixels do not have a steepest neighbourhood. Therefore, for these pixels, another type of connectivity relation is introduced; all neighbours of a minimum pixel p and having the same grey level as p pertain to the same component. Consequently, they are unified such that the representative of the regional minimum is the pixel with the smallest address value. Once the whole graph is constructed, its connected components have to be computed. Our design solution makes use, apart from the input image f, of an image l, which stores for every pixel its representative, or label. Let us underscore that pixel addresses are used for labeling [14] instead of arbitrary integer values. The algorithm consists of three raster scannings described below. $N_{prev}(p) = \{p' \in N(p) \mid p' < p\}$ represents the already scanned neighbourhood of p, i.e. all neighbours with a smaller address than p in the raster scanning order.

```
  **Watershed Algorithm for lower complete images** {
  *Input*: $f$.
  *Output*: $l$.
(1) Raster scan ($p$) {
  $q \leftarrow p$;
  for each ($p' \in N(p)$ and $f[p'] < f[p]$)
     if ($f[p'] < f[q]$) $q \leftarrow p'$;
  if ($q \neq p$)
     $l[p] \leftarrow q$;
     else $l[p] \leftarrow PLATEAU$;
}

(2) Raster scan ($p$) {
  if ($l[p] = PLATEAU$) {
     $l[p] \leftarrow p$;
     for each ($p' \in N_{prev}(p)$ and $f[p'] = f[p]$) {
        $r \leftarrow \mathrm{FIND}(l, p)$; $r' \leftarrow \mathrm{FIND}(l, p')$;
        $l[r] \leftarrow l[r'] \leftarrow \min(r, r')$;
     }
```

```
    }
}
(3) Raster scan ($p$)
    $l[p] \leftarrow$ FIND($l$, $p$);
}

FIND($l$, $u$) {
    for($r \leftarrow u$; $l[r] \neq r$; $r \leftarrow l[r]$);
    for ($w \leftarrow u$; $w \neq r$;)
            $tmp \leftarrow l[w]$; $l[w] \leftarrow r$; $w \leftarrow tmp$;
        return $r$;
}
```

In the first raster scanning, the label of each pixel p, which has a lower neighbour, is set to $q \in \mathrm{NLS}(p)$. Otherwise, if the pixel has no lower neighbour, it is on a minima plateau and is labelled *PLATEAU*.

A representative label is computed for every minima plateau in the second raster scanning. The connected component operator FIND (l, p) with path compression [6,7] returns the representative of the plateau on which p lies; this representative, in our implementation, is the pixel with the smallest address in the plateau. The path compression itself is performed in the second for-loop of the function FIND(l, p), by shortcutting all labels w on the path from u to the representative r; the latter was found in the first for-loop. Let us remark that performing the two raster scannings (1) and (2) at the same time is also possible.

In the third raster scanning, all pixel labels are replaced by their representative. In this way, the condition in Definition 3 is true for every pixel, and therefore, the presented segmentation algorithm performs a watershed segmentation.

Let us notice that apart from the input and output image, no queue or other data structure is needed. The algorithm is independent of the number of grey levels in the image and of the image histogram, uses only contiguous chunks of memory, avoiding thus memory fragmentation or additional indexing variables.

5. Extension to images with plateaus

Natural images do have non-minima plateaus. Therefore, an extension of Definition 3 and of the previous algorithm is needed. In this section, we will show how to extend the set of lower neighbours NLS on a path of steepest descent to include images with non-minima plateaus.

The basic problem is that the topographical distance (Definition 6) has the same value for any two plateau pixels, which do not have lower neighbours. Therefore, the geodesic distance, or an approximation of it, must be used to ensure that a pixel on a non-minima plateau gets the label from the nearest border pixel of the plateau which has a lower neighbour. The geodesic distance between two pixels p and p' on a plateau is equal to the length of the shortest path within the plateau between p and p' [5].

A plateau PL is a connected set of pixels of the same altitude. Let $\partial_{\mathrm{PL}} = \{p' \in \mathrm{PL} \mid \mathrm{NLS}(p') \neq \emptyset\}$ denote the set of pixels on the border of the plateau PL which have a lower neighbour; furthermore, let $\mathrm{gdist}_{\mathrm{PL}}(p, p')$ denote the geodesic distance, or an approximation of it, between p and p' within the plateau. The minimal distance between any pixel p on the plateau PL and all border pixels $p' \in \partial_{\mathrm{PL}}$ is $\mathrm{gdist}_{min}(p, \partial_{\mathrm{PL}}) = \min_{\forall p' \in \partial_{\mathrm{PL}}} \mathrm{gdist}_{\mathrm{PL}}(p, p')$. The watershed segmentation for images with non-minima plateaus can be defined by extending the steepest neighbourhood given in Definition 2:

Definition 10 (*Extended steepest neighborhood*). The set $\mathrm{NLS}'(p)$ contains the pixels of the sets $\mathrm{NLS}(p')$ of all border pixels $p' \in \partial_{\mathrm{PL}}$ such that

$$\mathrm{NLS}'(p) = \left\{ \bigcup_{p' \in \partial_{\mathrm{PL}} \mid \mathrm{gdist}_{\mathrm{PL}}(p,p') = \mathrm{gdist}_{min}(p,\ \partial_{\mathrm{PL}})} \mathrm{NLS}(p') \right\}.$$

Definition 11 (*Watershed segmentation for images with plateaus*). For any image with non-minima plateaus, a segmentation is called watershed segmentation if every regional minimum m_i has an unique label $L(m_i)$ and, for every pixel $p \in \Omega$ and $\mathrm{NLS}'(p) \neq \emptyset$, the following condition holds:

$\exists p' \in \mathrm{NLS}'(p)$ with $L(p) = L(p')$.

The definition leaves open the metric used for the geodesic distance. In our and most other implementations, an approximation of the geodesic distance based on the 4- or 8-square neighbourhood is used. The case of images without non-minimal plateaus is included, because the equation in Definition 10 possesses the following property: $\mathrm{gdist}_{\mathrm{PL}}(p', p) = 0 \Rightarrow p' = p \Rightarrow \mathrm{NLS}'(p) = \mathrm{NLS}(p)$.

6. The algorithm for images with plateaus

In order to perform a watershed segmentation on any input image and to fulfil the condition in Definition 11, another step has to be added to the algorithm for non-minima plateaus. Let us observe that after step (1), in Section 4, plateaus of minima and non-minima are not distinguishable. For a simple input image illustrated in Fig. 2(a), the result of step (1) is shown in Fig. 2(c), where the label *PLATEAU* has value -1.

An intermediate step, for the treatment of non-minima plateaus, is below described in the frame of the entire general algorithm;

3	5	5	2	8	8	8	11	10	10
5	5	11	11	8	11	11	8	10	10
11	5	11	11	9	9	9	9	8	10
11	11	11	7	7	7	7	9	9	8
11	11	11	11	11	9	7	10	8	10
11	10	11	9	7	7	9	9	10	8
11	10	11	9	11	9	10	10	8	10
11	11	11	8	8	8	8	8	10	10
11	11	11	11	10	10	10	10	10	10
10	10	10	10	10	10	10	10	10	10
11	11	11	11	10	10	10	10	10	10

a)

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	23	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109

b)

-1	0	3	-1	3	-1	-1	6	17	-1
0	0	3	3	3	14	5	-1	17	28
10	-1	21	33	33	34	35	36	-1	28
21	21	21	-1	-1	-1	-1	36	28	-1
51	51	33	33	33	34	-1	46	-1	48
51	-1	53	54	-1	-1	55	46	48	-1
61	-1	73	54	54	54	55	76	-1	68
61	61	73	-1	-1	-1	-1	-1	77	68
90	90	73	73	73	74	75	76	77	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
90	90	91	92	-1	-1	-1	-1	-1	-1

c)

-1	0	3	-1	3	4	5	6	17	8
0	0	3	3	3	14	5	6	17	28
10	10	21	33	33	34	35	36	17	28
21	21	21	-1	-1	-1	-1	36	28	28
51	51	33	33	33	34	-1	46	39	48
51	-1	53	54	-1	-1	55	46	48	48
61	-1	73	54	54	54	55	76	59	68
61	61	73	74	75	76	77	68	77	68
90	90	73	73	73	74	75	76	77	88
91	92	93	84	84	84	85	88	88	88
90	90	91	92	93	94	97	97	97	98

d)

0	0	3	3	3	4	5	6	17	8
0	0	3	3	3	14	5	6	17	28
10	10	21	33	33	34	35	36	17	28
21	21	21	33	33	33	33	36	28	28
51	51	33	33	33	34	33	46	39	48
51	51	53	54	33	33	55	46	48	48
61	51	73	54	54	54	55	76	59	68
61	61	73	74	75	76	77	68	77	68
90	90	73	73	73	74	75	76	77	88
91	92	93	84	84	84	85	88	88	88
90	90	91	92	93	94	97	97	97	98

e)

0	0	3	3	3	3	3	3	3	3
0	0	3	3	3	3	3	3	3	3
0	0	0	33	33	33	33	33	3	3
0	0	0	33	33	33	33	33	3	3
51	51	33	33	33	33	33	33	3	3
51	51	33	33	33	33	33	33	3	3
51	51	3	33	33	33	33	3	3	3
51	51	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3

f)

```
Watershed Algorithm {
   Input: f.
   Output: l.
 (I) Do step (1) of the algorithm of Section 4
(II) Raster Scan (p) {
   if ($l[p] = PLATEAU$)
         for each ($p' \in N(p)$)
         if ($l[p'] \neq PLATEAU$ and $f[p] = f[p']$)
            fifo_put(p'); break;
}
(III) while (fifo_empty( ) = FALSE) {
   $p \leftarrow$ fifo_get( );
   for each($p' \in N(p)$ and $l[p'] = PLATEAU$) {
      $l[p'] \leftarrow p$;
      fifo_put(p');
}
(IV) Do step (2) of the algorithm of Section 4
(V) Do step (3) of the algorithm of Section 4
}
```

Let us stepwise follow what result produces the algorithm above on the image example in Fig. 2(a). As already mentioned, 1D pixel addresses, in the raster scanning order, are used for labelling. Thus, the pixel location (i, j), $0 \leqslant i <$ nrows, $0 \leqslant j <$ ncols, in an image of size nrows $\times$ ncols has the 1D address equal to $i \times$ ncols $+ j$. All pixel addresses are illustrated in Fig. 2(b). In the rest of the paper, the 2D notation and its equivalent 1D value will be used to designate a pixel location.

The result of the first raster scanning can be observed in Fig. 2(c). The label of pixels have lower neighbours is set to the address of the lowest grey-level neighbour; otherwise, to *PLATEAU*, i.e. -1. Thus, pixel $(0, 4) \Leftrightarrow 4$ of grey level 8 has as lowest neighbour pixel $(0, 3) \Leftrightarrow 3$, of grey-level 2. Consequently, $l(4) \leftarrow 3$. Its neighbouring pixel $(0, 5) \Leftrightarrow 5$ has no lower neighbour and therefore it receives label *PLATEAU*, $l(5) \leftarrow -1$. Similarly, pixel $(3, 3) \Leftrightarrow 33$ of grey-level 7 is labelled *PLATEAU*, $l(33) \leftarrow -1$.

At step (II), for every *PLATEAU* pixel p which has a neighbour p' of the same grey level as p, but it also has a lower neighbour $(l(p') \neq PLATEAU), p'$ is introduced into the FIFO queue. Indeed, $p' \in \partial_{\mathrm{PL}}$ and therefore it is a seed for the computation of the extended steepest neighbourhood of pixels within the plateau. In our case, pixel $(0, 5) \Leftrightarrow 5$ will insert pixel $(0, 4) \Leftrightarrow 4$ into the queue and pixel $(2, 1) \Leftrightarrow 21$ will introduce pixel $(1, 0) \Leftrightarrow 10$.

Fig. 2. (a) Input image, (b) pixel addresses, (c) after first scan (I), (d) after flooding non-minima plateaus (III), (e) after connecting minima plateaus (IV), and (f) after replacing each label with its representative (V).

A global wave propagation, starting from the seeds in the queue, is performed at step (III). During this process, each seed pixel, accessed in FIFO order, sets its address as a label to all neighbouring *PLATEAU* pixels of the same altitude as itself. The latter become seeds and, at their turn, are introduced into the queue. The result of this step is depicted in Fig. 2(d). Thus, pixel $(0, 5) \Leftrightarrow 5$ receives label 4 from pixel $(0, 4) \Leftrightarrow 4$ and propagates its address to pixel $(0, 6) \Leftrightarrow 6$; next, the latter sets its label to 5. The propagation continues until the whole plateau of grey level 8 is exhausted. Let us remark that the condition of Definition 10 is fulfilled for non-minima plateaus, using an approximation of the geodesic distance. The latter is given by the time stamp of the wave propagation process, but it is not actually tracked during the algorithm.

After step (III), only minima plateaus are labelled *PLATEAU* (see Fig. 2(d)), because they do not have lower brims. The remaining stages are identical with steps (2) and (3) described in Section 4. Thus, pixels on minima plateaus are connected at step (IV) using the connected component operator. The result of this phase is shown in Fig. 2(e). Minima pixels $(0, 0) \Leftrightarrow 0$ and $(0, 3) \Leftrightarrow 3$ are their own representative and accordingly, $l(0) \leftarrow 0$, $l(3) \leftarrow 3$. The effectiveness of the for-loop within this raster scanning is more evident on the plateau of grey-level 7; the latter is completely labeled with its representative label 33, i.e. the smallest pixel address within the plateau, parsed in raster scanning order. Similarly, the regional minimum of grey-level 10 is labelled 51.

At step (V), for each pixel, its label is replaced by its representative at step (V). The output image can be observed in Fig. 2(f).

Unlike in the algorithm in Section 4, a FIFO queue is here needed, but only the pixels on non-minima plateaus are vehiculated through this queue. Thus, the dimension of this queue is smaller than that of the hierarchical queue (a sufficient size could be computed during the first raster scanning, namely by counting the total number of *PLATEAU* pixels). Additionally, before allocating each of the FIFO queue in the hierarchical queue, the classical algorithm must compute the image histogram; this step disappears entirely in the present algorithm. Finally, the mechanisms for manipulating a FIFO queue are much simpler than those for a hierarchical queue.

7. Complexity analysis and experimental results

Given an image with n pixels, the complexity of the algorithm in Section 6 is now analysed step by step. At Steps (I) and (II) a linear scan with access to a limited neighbourhood is performed. Therefore the complexity of both steps is $O(n)$, or linear with the number of pixels (there exist the constants c_1, c_2 such that the complexity equals $c_1 \times n + c_2$).

Table 1
Timing results

Running time (s)			
Image\Algorithm	Approach I	Approach II	Hierarchical queues
Cermet (256 × 256)	0.07	0.08	0.15
Lenna (512 × 512)	0.34	0.35	0.76
Peppers (512 × 512)	0.35	0.36	0.71
Simple512 (512 × 512)	0.39	0.35	0.71
People (1024 × 1024)	1.47	1.35	3.26

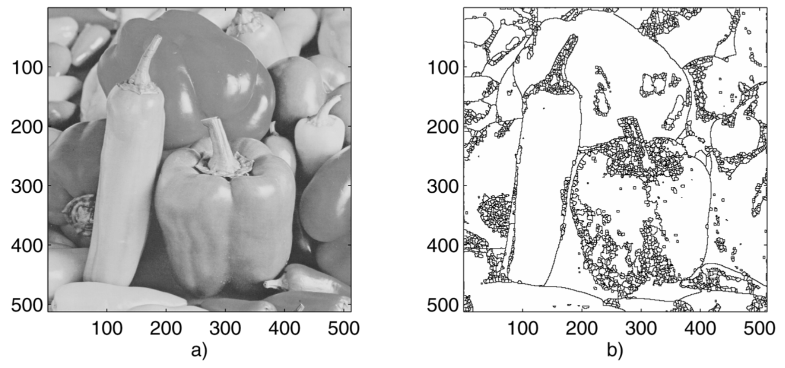


Fig. 3. *Peppers* (a) input image (b) output image.

Each pixel on a non-minimum plateau is inserted into the FIFO queue during Steps (II) and (III) at most once. For each pixel in the FIFO queue, a limited neighbourhood is accessed at step (III). Therefore, the overall worst-case complexity of step (III) is $O(n)$.

Let $n' \leqslant n$ be the number of minima plateau pixels. Since we use path compression in the FIND (l, p) operation in combination with naive linking at step (IV), the worst-case complexity of this step is $O(n' \log n')$ [6,7]. The worst-case complexity can be reduced to linear for practical problem sizes, if linking by rank or size is used [6,7], at the expense of an additional image to store the rank or size. Nevertheless, for the images we tested, the logarithmic factor could not be observed.

At step (V), for each of the n pixels, a FIND (l, p) operation is performed. The pixels in the image can be divided into two sets. Let F be the set of pixels which have not been already accessed by a FIND operation. Initially all pixels are in F. Each FIND operation walks along a path of pixels which are within F. As soon as it hits a pixel $p' \notin F$ the operation finishes, because p' has already been shortcut to its representative. Afterwards, all pixels on the path are shortcut as well and removed from F. Therefore, the total complexity of step (V) is $O(n)$ because $|F| = n$ and the total number of FIND operations is also n.

As a result, the overall worst-case complexity of the algorithm is $O(n + n' \log n')$. With our test images we could not observe the logarithmic factor. Therefore, the algorithm can be treated as $O(n)$ for practical images.

Concerning the memory requirements, the algorithm described in the previous section makes use of an input and an output image, as well as of a FIFO queue. As already mentioned, the size of this queue can be dynamically computed, at the run time; alternatively, the size of the image can be used instead.

In Table 1, the presented algorithm is compared with the traditional hierarchical queue algorithm. The time measurements were performed on a Silicon Graphic™ O2 workstation with an R10000 RISC processor. Approach I is the implementation of the algorithm as presented here, while in Approach II, step (II), slightly modified as below explained, is performed at the same time with step (I). This saves the overhead of a scan through the image, but many unnecessary seeds might be detected, because labels of half of the neighbours are only available at this stage. Therefore, all pixels p' having a lower neighbouring pixel and also neighbours p of the same altitude are stored as seeds; however, not all the pixels p in the neighbourhood not already scanned and having the same altitude as p' will be labeled *PLATEAU* by the test at Scan (I). Hence, p' are useless in the FIFO queue. The results of both implementations show however a significant improvement in the running time compared against the classical algorithm.

One image example is illustrated in Fig. 3(a). Taking the gradient image thresholded with an arbitrary value as input for the watershed algorithm, the output can be observed in Fig. 3(b). Let us notice that only the boundaries of the labelled regions are shown in the latter figure.

8. Conclusion

In this paper, we have presented a definition for the watershed segmentation which is consistent with the behaviour of most implementations of the watershed algorithm, namely, to chose one arbitrary label in the case of competing labels. Different distance metrics to approximate the geodesic distance on plateaus can be incorporated into the definition.

The definition led to a new type of watershed algorithm which is closely related to the connected component algorithm. We have shown that the algorithm has a linear complexity with the number of pixels, except the connection of minima plateau pixels which introduces, in the worst case, an additional logarithmic factor. For the images we tested, the logarithmic factor could be however not observed.

The algorithm has a regular structure (raster scannings comprising simple pixel assignment rules), the memory requirements are minimal (three contiguous chunks of memory accessed by direct indexing techniques) and independent of the image content (image resolution and image histogram), leading to a robust and efficient implementation. Consequently, our timing results show a significant improvement in the running time, compared against the classical watershed algorithm.

Combining our watershed algorithm with an opening by reconstruction [16,17], to find markers for "significant" objects in the image, a marker-based watershed algorithm results, which is thus independent of the number of grey levels. Consequently, the algorithm is very suitable for images of large resolution, for which the hierarchical queue approach is rather expensive.

Finally, the connected component-like formulation of watersheds exhibits a better parallel potential, allowing the design of efficient and scalable parallel watershed algorithms [10–12].

9. Summary

The watershed transformation is a popular image segmentation algorithm for grey-scale images. The traditional watershed algorithm simulates the flooding process with the help of hierarchical queues. In this paper, we develop a formalism for the watershed transformation, which does not build watersheds at the same time with flooding of the basins, based on sets of neighbouring pixels. Our definition is consistent with the behaviour of most implementations of the watershed algorithm, namely, to choose one arbitrary label in the case of competing labels. Moreover, different distance metrics to approximate the geodesic distance on plateaus can be incorporated into the formalism. The relation to the traditional definition of watershed segmentation is proven in the paper.

The formalism leads to a new type of watershed algorithm which is closely related to the connected component algorithm. The algorithm that we here introduce is more simple, with respect to implementation and data structures. Additionally, the memory requirement is small and independent of the number of grey levels in the input image. Furthermore, our timing results show a significant improvement in the running time, compared against the classical watershed algorithm.

References

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

[2] F. Meyer, Topographic distance and watershed lines, Signal Processing 38 (1) (1994) 113–125.

[3] F. Meyer, S. Beucher, Morphological segmentation, J Visual Commun. Image Representation 1 (1) (1990) 21–46.

[4] L. Najman, M. Schmitt, Watershed of a continuous function, Signal Processing 38 (1) (1994) 99–112.

[5] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Trans. Pattern Anal. Mach. Intell. 13 (6) (1991) 583–598.

[6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.

[7] R.E. Tarjan, Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1983.

[8] A. Bieniek, A. Moga, A connected component approach to the watershed segmentation, in: Mathematical Morphology and its Applications to Image and Signal Processing, Computational Imaging and Vision, Vol. 12, Kluwer Academic Publishers, Dordrecht, 1998, pp. 215–222.

[9] A. Meijster, J.B.T.M. Roerdink, A disjoint set algorithm for the watershed transform, Proceedings EUSIPCO'98, IX European Signal Processing Conference, September 8–11, Rhodes, Greece, 1998.

[10] A. Bieniek, H. Burkhardt, H. Marschner, M. Nölle, G. Schreiber, A parallel watershed algorithm, in: Proceedings of the 10th Scandinavian Conf. on Image Analysis (SCIA97), Lappeenranta, Finland, June 1997, pp. 237–244.

[11] Alina Moga, Parallel watershed algorithms for image segmentation, Ph.D. Thesis, Tampere University of Technology, Tampere, Finland, 1997.

[12] A. Moga, M. Gabbouj, Parallel image component labelling with watershed transformation, IEEE Trans. Pattern Anal. Mach. Intell. 19 (5) (1997) 441–450.

[13] R. Lumia, L. Shapiro, O. Zuniga, A new connected components algorithm for virtual memory computers, Comput. Vision Graphics Image Processing 22 (2) (1983) 287–300.

[14] R. Miller, Q.F. Stout, Parallel Algorithms for Regular Architectures: Meshes and Pyramids, MIT Press, Cambridge MA, 1996.

[15] H. Samet, Connected component labeling using quadtrees, J. ACM 28 (3) (1981) 487–501.

[16] Pierre Soille, Morphologische Bildverarbeitung, Springer, Berlin, 1998.

[17] P. Soille, C. Gratin, An efficient algorithm for drainage network extraction on DEMs, J. Visual Commun. Image Representation 5 (1994) 181–189.

About the Author—ALINA NICOLETA MOGA was born in Alba Iulia, Romania, in 1969. She received the MSc degree in computer science from "Politehnica" University of Bucharest, Bucharest, Romania, in 1993 and the Ph.D. degree in parallel image segmentation algorithms at Signal Processing Laboratory, Department of Information Technology, Tampere University of Technology, Tampere, Finland, in 1997. Dr. Moga is currently a research assistant with Albert-Ludwigs-Universitt Freiburg, Institut für Informatik, Freiburg, Germany. Her main research interests include parallel and distributed computing, efficient algorithms, image segmentation, and multiscale adaptive techniques.

About the Author—ANDREAS BIENIEK was born 1966 in Hamburg, Germany. He studied at the Technical University of Hamburg-Harburg until 1993. Toward obtaining the M.Sc. degree in electrical engineering/computer science, Andreas Bieniek worked in 1992 at the University of Melbourne, Australia on "Performance Evaluation of Task Allocation and Scheduling for an Optoelectronic Multicomputer". Currently he is finalizing his Ph.D. thesis on parallel image processing algorithms at the Albert-Ludwigs-Universität, Freiburg, Chair of Pattern Recognition and Image Processing. His research interests include parallel algorithms, image segmentation, and communication networks.