Analyses of the Watershed Transform

Ramzi Mahmoudi, Mohamed Akil

The Watershed Transform

Ramzi MAHMOUDI *mahmoudr@esiee.fr*
Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI
ESIEE Paris - Cité Descartes, BP99, 93162 Noisy Le Grand, France

Mohamed AKIL *akilm@esiee.fr*
Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI
ESIEE Paris - Cité Descartes, BP99, 93162 Noisy Le Grand, France

Abstract

In the framework of mathematical morphology, watershed transform (WT) represents a key step in image segmentation procedure. In this paper, we present a thorough analysis of some existing watershed approaches in the discrete case: WT based on flooding, WT based on path-cost minimization, watershed based on topology preservation, WT based on local condition and WT based on minimum spanning forest. For each approach, we present detailed description of processing procedure followed by mathematical foundations and algorithm of reference. Recent publications based on some approaches are also presented and discussed. Our study concludes with a classification of different watershed transform algorithms according to solution uniqueness, topology preservation, prerequisites minima computing and linearity.

Keywords: Watershed transform, Flooding, Path-cost minimization, Topology preservation, Local condition, Minimum spanning forest.

1. INTRODUCTION

The watershed concept began with Maxwell [1] who introduces the theory behind representing physical characteristics of a land by means of lines drawn on a map. He highlights relationships between the numbers of hills, dales and passes which can co-exist on a surface. Subsequently, through the work of Beucher and al. [2], watershed transform was introduced to image segmentation and nowadays it represents one of the basic foundations of image processing [3].

In this framework, the most simplified description of the watershed approach is to consider a grayscale image as a topographic surface: the gray level of a pixel becomes the elevation of a point, the basins and valleys correspond to dark areas, whereas the mountains and crest lines correspond to the light areas. If topographic relief is flooded by water, watersheds will be the divide lines of the attraction's domains of rain falling over the region [4] or sources of water springing from reliefs' peaks. Another synopsis has shown consistency is that topographic surface is immersed in a lake with holes pierced in local minima. Catchment basins will fill up with water starting at these local minima, and, at points where water coming from different basins would meet, dams are built. As a result, the topographic surface is partitioned into different basins separated by dams, called watershed lines. Figure 1 gives a very symbolic description of the mentioned approach. In fact, it shows trends in the use of watershed transform for image processing.

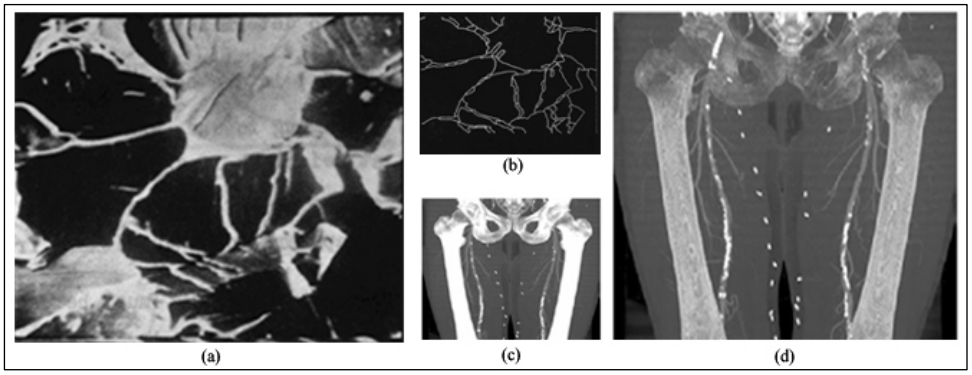

FIGURE 1: (a) Cleavage fractures in steel, (b) contour of (a) obtained truth watershed definition introduced by Beusher and al. [2] in 1979, (c) Maximum intensity projection of original human lower limb (d) Bone tissue removed using mask extended with 3D watershed transform introduced by Straka and al.[5] in2003.

Despite its simplicity, this concept has been formalized in different ways giving rise to several definitions of watershed transform. In the discrete case, which is our main interest in this paper, this problem is amplified since there is no unique definition of the path that the drop of water would follow. This led to a multitude of algorithm to compute watershed transform. Some of these algorithms don't even meet associated watershed definition. We also note that some definitions take the form of algorithm specification which makes the distinction between algorithm specification and implementation very complicated. This problem in literature has been partially resolved in Roerdink and al. [6] ten years ago. Actually authors presented a critical review of several definitions of the watershed transform and the associated sequential algorithms. Even they discuss various issues which often cause confusion in the literature; they don't go further in the classification or comparison of different approaches. They instead focus on parallelization aspect. In other more recent publications, authors tentatively drawn a comparison chart of some watershed transform definition to serve their end goals: showing the relationships that may exist between some discrete definition of watershed [7] or showing that most classical watershed algorithm do not allow the retrieval of some important topological features of the image [8].

The purpose of this paper is to introduce an intensive study of all existing watershed transform in the discrete case: WT based on flooding, WT based on path-cost minimization, watershed based on topology preservation, WT based on local condition and WT based on minimum spanning forest. Indeed, for each approach, we start by giving informal definition, then we present processing procedure followed by mathematical foundations and the algorithm of reference. Recent publications based on some approach are also presented and discussed. Our study concludes with a classification of different algorithms according to criteria of recursion, complexity, basins computing and topology preservation. This paper is organized as follows: in section 2, different approaches to compute watershed are presented. In section 3, we draw a comparison between the various presented algorithms. Finally, we conclude with summary and future work in section 4.

2. WATERSHED TRANSFORMATION

In this section, we propose a comparative study of different approaches to compute the watershed transform in the discrete case. The goal is to identify most suited watershed transform for parallel processing. This study can also be seen as an update of Roerdink work [6] presented ten years ago. Indeed, for each approach, we present processing procedure, mathematical foundations and sequential algorithm. Recent publications based on some approach are also presented and discussed.

2.1 Watershed based on flooding

Based on flooding paradigm [9,10,11], the intuitive idea underlying this method comes from geography. Since grayscale image can be seen as topographic surface, the intensity of a pixel

can be considered as the altitude of a point. Now, let immerge this surface in still water, with holes created in local minima. Water fills up basins starting at these local minima. As described in algorithm 1, the filling of basins is an iterative process that involves gradually raising the water level from Alt_{min} to Alt_{max}. Algorithm must, for each iteration, fill existing basins (extension regions) and possibly create new basins (new regions). We denote by LR the region list. Dams will be built where waters coming from different basins meet.

Algorithm 1: Flooding watershed process

```
1.  for level from Alt_min to Alt_max
2.      // Action 1 : Extend existing region
3.      foreach (R∈ L_R) do Growing [R] until level Alt;
4.      end_for
5.      // Action 2 : Create new region
6.      foreach (Pixel P ∈ level)
7.          if (Pixel P is not associated to any region R) then
8.              Create new region [R] in L_R ;
9.              Add Pixel P to region [R];
10.             Growing [R] until level ;
```

For mathematical formulation of the mentioned process, let $f : D \to \mathrm{N}$ be a digital grey value image, with Alt_{min} and Alt_{max} the minimum and the maximum value of. The threshold set of at level ***Alt*** is: $T_{Alt\,\min} = \{p \in D / f(p) \le Alt\}$ (2.1)

It define a recursion with the gray level ***Alt*** increasing from Alt_{min} to Alt_{max}, the basin associated with the minima of f are successively expanded. Let X_{Alt} denote the union of the set of basins computed at level ***Alt***. A connected component of the threshold set T_{Alt+1} at level $Alt+1$ can be either a new minimum or an extension of the basin in X_{Alt}. The geodesic influence zone (IZ) of X_{Alt} within T_{Alt+1} can be computed resulting in an update X_{Alt+1}. Let MIN_h denote the union of all regional minima at altitude ***Alt***. Then we can introduce the following definition:

Definition 2.1 *(Flooding watershed)* (2.2)

$$\begin{cases} X_{Alt\,\min} = \{p \in D / f(p) = Alt_{\min}\} = T_{Alt\,\min} \\ X_{Alt+1} = MIN_{Alt+1} \cup IZ_{T_{ALT+1}}(X_{Alt}) \end{cases}$$

The watershed $Wshed(f)$ of f is the complement of $X_{Alt_{\max}}$ in D :

$$Wshed(f) = D / X_{Alt\,\max} \quad (2.3)$$

Vincent and Soille [12] presented an original and efficient implementation (algorithm 2) of the flooding watershed. This implementation uses FIFO queue and it needs two steps:

(1) Sorting pixels in increasing order of grey values (Alt_{min} ; Alt_{max})

(2) Flooding process: All nodes with grey level h are first given the initial label. Then those nodes that have labeled neighbors from the previous iteration are inserted in the queue, and from these pixels geodesic influence zones are propagated inside the set of initial pixels.

In their study [6] Roerdink and Meijster have removed two points of inconsistency in the algorithm's recursion. (i) Only pixels with grey value h are masked for flooding (line 13), instead of all non-basin pixels of (level $\le h$), as the definition (2.2) would require. This explains why labels

of wshed-pixels (line 15) are also propagated with labels of catchment basins. (ii) If a pixel is adjacent to two different basins; it is initially labeled 'wshed'. But it is allowed to be overwritten at the current grey level by another neighbor's label, if that neighbor is part of a basin (lines 35-36).

They also propose some modification to implement the recursion (2.3) exactly. In line 13, all pixels with $im[p] \leq h$ have to be masked, the queue has to be initialized with basin pixels only (drop the disjunct lab[q] = wshed in line 15), the resetting of distances (line 50) has to be done in line 14, and the propagation rules in lines 32-47 have to be slightly changed.

Algorithm 2 : Flooding watershed [Vincent & Soille]

Data : Digital grey scale image *G=(D,E,im)*
Result : Labelled watershed image *lab* on *D*

1. #define INIT -1 *//initial value of lab image*
2. #define MASK -2 *//initial value of each level*
3. #define WSHED 0 *//label of the watershed pixels*
4. #define FICTITIOUS (-1,1) *//fictitious pixel $\notin D$*
5. $curlab \leftarrow 0$ *//curlab is the current label*
6. fifo_init(queue)
7. **for all** (P∈ D) **do**
8. $lab[p] \leftarrow INIT;$
9. $dist[p] \leftarrow 0;$ *//dist is a work image of distances*
10. **end_for**
11. **SORT** pixels in increasing order of grey values (h_{min} , h_{max})
    // starting flooding process
12. **for** h = h_{min} **to** h_{max} **do** *//geodesic SKIZ of level h-1 inside level h*
13. **for all** (P∈ D) **with** im[p]=h **do** *// mask all pixels at level h*
    //these are directely accessible because of the sorting step
14. $lab[p] \leftarrow MASK;$
15. **if** (**p** has a neighbour **q**) **with** ((lab[p] > 0 **or** lab[q]=WSHED)) **then**
    //initialize queue with neighbours at level h of current basins or watersheds
16. $dist[p] \leftarrow 1;$
17. $fifo\_add(q, queue);$
18. **end_if**
19. **end_for**
20. $curdist \leftarrow 1;$
21. $fifo\_add(FICTITOUS, queue);$
22. **loop** *//extend basins*
23. $p \leftarrow fifo\_remove(queue);$
24. **if** (p = FICTITIOUS) **then**
25. **if** $(fifo\_empty(queue))$ **then**
26. **BREAK ;**
27. **else** $(fifo\_add(FICTITOUS, queue)$;
28. $curdist \leftarrow curdis + 1;$
29. $p \leftarrow fifo\_remove(queue);$
30. **end_if**
31. **end_if**
32. **for all** (q∈ N_G(p)) **do** *//labelling p by inspecting neighbours*
33. **if** (dist[q] < curdist) **and** (lab[q]>0 **or** lab[q]=WSHED) **then**
    // /q belongs to an existing basin or to watersheds
34. **if** (lab[q]>0) **then**
35. **if ((**lab[p]=MASK) **or** (lab[p]=WSHED)) **then**

```
36.                         lab[p] ← lab[q];
37.                     else if  lab[p] ≠ lab[q] then
38.                         lab[p] ← WSHED;
39.                     end_if
40.             else if (lab[p]=MASK) then
41.                         lab[p] ← WSHED;
42.                 end_if
43.             else if  ((lab[q]=MASK) and (dist[q]= 0)) then //q is plateau pixel
44.                         dist[q] ← curdis + 1;
45.                         p ← fifo_add(q, queue);
46.                     end_if
47.         end_for
48.     end_loop
        //detect and process new minima at level h
49.     for all (p∈ D) with (im[p]=h) do
50.         dist[p] ← 0;  // reset distance to zero
51.         if (lab[p]=MASK) then  // p is inside a new minimum
52.             curlab ← curlab + 1; //create new label
53.             fifo_add(p, queue);
54.             lab[p] ← curlab;
55.             while not (fifo_remove(queue)) do
56.                 q ← fifo_remove(queue);
57.                 for all (r∈ N_G(q)) do // inspect neighbours of q
58.                     if (lab[r]=MASK) then
59.                         fifo_add(r, queue);
60.                         lab[r] ← curlab;
```

From the introduction about immersion simulation above, we can see that the level-by-level method during the flood procedure is uniform. Unfortunately, this can cause over-segmentation in several cases. Based on this original simulating immersion, Shengcai and Lixu [13] propose, in 2005, a novel implementation using multi-degree immersion. To our knowledge, it is the last update that can be found in the literature, the proposed implementation resists to over-segmentation problem effectively. It starts by redefining the threshold set of f at level ***Alt***. Instead of the original formula (2.1), they propose the following one:

$$T_{Alt} = \{p \in D / f(p) - Diff(p) \leq Alt\} \tag{2.4}$$

With $Diff(p)$ refers to the immersion level when the flooding reaches pixel p. Segmentation results are sensitive to this function. In fact, if $Diff(p) = 0$, (2.4) can be seen as a special case of (2.1). Other extreme case, if $Diff(p)$ reaches its maximum values, all pixels susceptible to be dumped, will be. According to the user requirement, $Diff(p)$ can be even a constant function or a function computed according to the local characteristic of p. In their paper, Shengcai and Lixu define it as introduced in (2.5) with $Neighbor(p)$ is the set of all p neighbors. And $conn$ refers to the predefined connectivity: $\{4,8\}$.

$$Diff(p) = \sum_{q \in Neighbor(p)} \frac{|f(p) - f(q)|}{conn} \tag{2.5}$$

Obtained results through two implementations of the original and the multi-degree watershed shows that multi-degree immersion method resists the over-segmentation problem effectively. Indeed, the number of detected region in an image brain (181*217*181 voxel volume), decreases from 10991, using the old method, to only 35 using the new method. Computation time and consumed memory size are practically the same. More information about implementation can be found in [13].

2.2 Watershed based on path-cost minimization

In this class, there are two possible approaches. The first one associates a pixel to a catchment basin when the topographic distance is strictly minimal to the respective regional minimum. While the second one builds a forest of minimum-path trees, each tree representing a basin. In the following we start by introducing watershed by topographic distance [14] before moving to the watershed by image foresting transform [16,17].

Algorithm 3 : Watershed by topographic distance process

1. **Foreach** (marked area ∈ S_M)
2. insert pixels into priority queue Q ;
3. **end_for**
4. **While** ($Q \neq \phi$)
5. Ex_p = extract pixels with highest priority level;
6. **if** (neighbors of $p \in Ex_p$ have the same label Lab) **then**
7. $Lab_p = Lab$;
8. Q = all non-marked neighbors
9. **end_if**
10. **end_while**

Based on the drop of water principle, the intuitive idea behind topographic watershed approach in the steepest descent path principal [14, 15]. A drop of water falling on a topographic relief flows down, as "quickly" as possible, until it reaches a regional minimum. Let $f : D \to \mathrm{N}$ be a digital grey value image. Let S_M be the set of markers, pixels where the flooding shall start, are chosen. Each is given a different a label Lab .

Topographic watershed process can be described by algorithm 3. Let us note that priority level when inserting neighbors (line 2) corresponds to the gray level of the pixel. In line 6, only neighbours that have already been labelled are compared. Finally, only neighbors (in line 8) that are not yet in the priority queue are pushed into the priority queue. The watershed lines set are the complement of the set of labeled points.

For mathematical formulation of the mentioned process, we follow here the presentation in [6] which is based on [14]. For the sake of simplicity, we restrict our self to the minimal set of notion that will be useful for our propos. We start by introducing the topographic distance. Let us consider $N_G(p)$ as the set of neighbors of pixel p , and $d(p,q)$ as the distance associated to edge (p,q) . Then the lower slope $LS(p)$ of f at a pixel p can be defined as follow:

$$LS(p) = \underset{q \in N_G(p) \cup \{p\}}{MAX} \left(\frac{f(p) - f(q)}{d(p,q)} \right) \tag{2.6}$$

And the cost for walking from pixel a p to a neighboring pixel q can be defined as:

$$\cos(p,q) = \begin{cases} LS(p).d(p,q) / f(p) > f(q) \\ LS(q).d(p,q) / f(p) < f(q) \\ \frac{1}{2}\big(LS(p) + LS(q)\big).d(p,q) / f(p) = f(q) \end{cases} \quad (2.7)$$

The topographical distance between p and q is the minimum of the topographical distances $T_f^{\pi}(p,q)$ along all paths between p and q: $T_f(p,q) = \underset{\pi \in [p \to q]}{MIN} T_f^{\pi}(p,q)$ (2.8)

We recall that the topographical distance along a general path $\pi = (p_0,...,q_l)$ is defined as

$$T_f^{\pi}(p,q) = \sum_i^{l-1} d(p_i, p_{i+1}) \cos t(p_i, p_{i+1}) \quad (2.9)$$

Finally we can define the topographic watershed for a grey value image f, with f^* the lower completion of f. Each pixel which is not in a minimum has a neighbor of lower grey value ($f^* = f_{LC}$).

Let $(m_i)_{i \in I}$ be the collection of minima of f. The basin $CB(m_i)$ of f corresponding to a minimum $(m_i)_{i \in I}$ is defined as a basin of the lower completion of f:

$$CB(m_i) = \left\{ p \in D, j \in I / \{i\} : f^*(m_i) + T_{f^*}(p, m_i) < f^*(m_j) + T_{f^*}(p, m_j) \right\} \quad (2.10)$$

And the watershed is the set of points which do not belong to any catchment basin:

$$Wshed(f) = D \cap (\cup_{i \in I} CB(m_i)^c \quad (2.11)$$

Several shortest paths algorithms for the watershed transform with respect to topographical distance can be found in the literature but the reference algorithm is that of Fernand Meyer. In the following we present variant of Meyer algorithm with integrate the lower slope of the input image as introduced Roerdink and al. [6].

Algorithm 4 : Watershed by topographical distance [Meyer]

Data : Lower complete image *im* on a dgital grey scale image G=(D,E) with cost
Result : Labelled watershed image lab on D

```
1.  #define WSHED 0 //label of the watershed pixels
2.  //Uses distance image dist. On output, div[v]=im[v], for all v ∈ D
3.  for all (v∈D) do //Initialize
4.     lab[v] ← 0;
5.     dist[v] ← ∞;
6.  end for
7.  for all (local minima mi) do
8.    for all (v∈mi) do
9.       lab[v] ← i;
10.      dist[v] ← im[v]; //Initialize distance with the values of minima
```

```
11.  end for
12. end for
13. stable ← true; //stable is a Boolean variable
14. repeat
15.   for all pixels u in forward raster scan order do
16.     propagate(u)
17.   end for
18.   for all pixels u in backward raster scan order do
19.     propagate(u)
20.   end for
21. until stable
22. procedure propagate(u)
23.   for all (v∈NG(u)) in the future (w.r.t scan order) for u do
24.     if (dist[u] + cost[u,v] < dist[u]) then
25.       dist[v] ← dist[u] + cost(u,v)
26.       lab[v] ← lab[u]
27.       stable ← false
28.     else if lab[v] ≠ WSHED and (dist[u] + cost[u,v] = dist[v]) then
29.       if (lab[v] ≠ lab[u]) then
30.         lab[v] = WSHED
31.         stable ← false
```

The second approach to compute a watershed based on path-cost minimization, as we introduced in the beginning, consists on building a forest of minimum-path trees where each tree represent a basin. This approach is described in the framework of image foresting transform [16]. The IFT defines a minimum-cost path forest in a graph, whose nodes are the image pixels and whose arcs are defined by an adjacency relation between pixels. The cost of a path in this graph is determined by a specific path-cost function, which usually depends on local image properties along the path, such as color or gradient. The roots of the forest are drawn from a given set of seed pixels. For suitable path-cost functions, the IFT assigns one minimum-cost path from the seed set to each pixel, in such a way that the union of those paths is an oriented forest, spanning the whole image. The IFT outputs three attributes for each pixel: its predecessor in the optimum path, the cost of that path, and the corresponding root. Returned solution is usually obtained in linear time and requires a variant of the Dijkstra [18] , Moore [19] or Dial's shortest-path algorithm [20].

For mathematical formulation of the IFT-watershed, we start by defining some basic notions of image foresting transform as introduced in [16]. Actually, an image ***ImgIn*** can be seen as a pair (J, I) where J refers to a finite set of pixels and I refers to a mapping that assigns to each pixel $(p \in J)$, a pixel value $I(p)$ in some arbitrary value space. Distinct binary relation between pixels of J will define an adjacency relation A. Once the adjacency has been fixed, ***ImgIn*** can be interpreted as a directed graph, whose nodes are image pixels and whose arcs are pixel pairs in A.

Before moving to path cost definition, let's remember that a sequence of pixels $\pi = \langle t_1, t_2, ..., t_k \rangle$ where $(t_i, t_{i+1}) \in A$ for $(1 < i < k-1)$ constitute a path. In the following we will denote by $org(\pi)$ the origin t_i of π and by $dst(\pi)$ the destination t_k of π. Now let assume given a function f that assigns to each path π a path cost $f(\pi)$, in some totally ordered set υ of cost values. We

introduce the max-arc path-cost function $f_{\max}$ that will be used later. Note that $h(t)$ and $w(s,t)$ are fixed.

$$f_{\max}(\langle t \rangle) = h(t)$$
$$f_{\max}(\pi \cdot \langle s,t \rangle) = \max\{f_{\max}(\pi), w(s,t)\} \quad (2.12)$$

For IFT use, a specific function $f^S(\pi)$ can be defined since the search to paths start in a given set $(S \subset J)$ of seed pixels.

$$f^S(\pi)\begin{cases} f(\pi) if\ (org(\pi) \in S) \\ +\infty \quad otherwise \end{cases} \quad (2.13)$$

Now, we can introduce the spanning forest concept. We remember that a predecessor map is a function P that assigns to each pixel $t \in J$ either some other pixel$\in J$ or a distinctive marker $\mathbf{M} \notin J$. Thus, a spanning forest (**SF)** can be seen as a predecessor map which contains no cycles.

Definition 2.2 *(Spanning forest)* (2.14)

For any pixel $t \in J$, a spanning forest P defines a path $P*(t)$ recursively as $\langle t \rangle$ if $P(t) = \mathbf{M}$ and $P^*(s) \cdot \langle s,t \rangle$ if $P(t) = s \neq \mathbf{M}$, we denote by $P^0(t)$ the initial pixel of $P^*(t)$.

Algorithm 5: IFT Algorithm [Falco and Al.]

Input: ***Img*** = (J, I) : Image, $A \subset JxJ$: Adjacency relation, f :path-cost function

Output: P : optimum path forest, α, β : two sets of pixels with $\alpha \cup \beta = J$.

1. Set $\alpha \leftarrow \{\ \}$, $\beta \leftarrow J$ *//Initialize*
2. **for all pixels** $t \in J$ **do** *//Initialize*
3. Set $P(t) \leftarrow \mathbf{M}$
4. **end for**
5. **while** $\beta \neq \phi$ **do** *//Compute*
6. **remove** from β a pixel s such that $f(P^*(s))$ is minimum,
7. **add** s **to** α
8. **for** each pixel t such that $(s,t) \in A$
9. **If** $f(P^*(s) \cdot \langle s,t \rangle) < f(P^*(t))$ **then**
10. **Set** $P(t) \leftarrow s$;

Falcan and al. algorithm describes IFT computing. Its algorithm is based on Dijkstra's procedure for computing minimum-cost path from a single source in a graph and returns an optimum-path forest P for a seed-restricted cost function f^s or any pixel t with finite cost $f^s(P*(t))$. Pixels will belong to a tree whose root is a seed pixel.

The IFT-watershed assumes that seeds pixel correspond to regional minima of the image or to markers that can be considered as imposed minima. The max-arc path-cost function $f_{\max}$ is the same as (2.12). We remember that $h(t)$ is a fixed but arbitrary handicap cost for any paths

starting at pixel t. We remember also that $w(s,t)$ is the weight of arc $\langle s,t \rangle \in A$, ideally, higher on the object boundaries and lower inside the objects.

There are two usual arc weight functions :

a) $w_1(s,t) = |J(s) - J(t)|$, where $J(s)$ refers to the intensity of pixel of s. In that case IFT-Watershed is said by dissimilarity.

b) $w_2(s,t) = G(t)$, where $G(t)$ is the morphological gradient of ***ImgIn*** at pixel t. In that case IFT-Watershed is said on gradient.

Algorithm 6 : IFT-watershed from markers [Lotufo]

Data : I : input image, $wshed$: labeled marker image
Result : $wshed$: watershed catchment basins
Aux : C : cost map, initialized to infinity; $FIFO$:hierarchical FIFO queue

1. **for all pixels** $wshed(p) \neq 0$ **do** *//Initialize*
2. $C(p) \leftarrow I(p)$
3. Insert p in $FIFO$ with cost $C(p)$
4. **end for**
5. **while** $FIFO$ **do** *//Propagation*
6. $p \leftarrow$ remove from $FIFO$
7. **for each** $q \in N(p)$
8. **If** $(C(q) > \max\{C(p), I(q)\})$ **then**
9. $C(q) \leftarrow \max\{C(p), I(q)\}$;
10. Insert q in the $FIFO$ with cost $C(q)$;
11. $wshed(p) \leftarrow wshed(q)$

Lotufo and al. [21] introduce the IFT-watershed from markers, algorithm 6, which can be computed by a single IFT where the labeled markers are root pixels. Proposed algorithm use hierarchical FIFO queue (HFQ).

In this case, the root map can be replaced by the label map which corresponds to the catchment basins and the used path-cost function is given by:

$$f_m(< p_1, p_1, ..., p_1 >) = \begin{cases} \max\{(I(p_1), I(p_2), ..., I(p_n)\} & \text{If } p_1 \text{ is a marker pixel.} \\ +\infty & \text{Otherwise} \end{cases} \quad (2.15)$$

Where $p_{i+1} \in N(p_i)$ and $I(p_i)$ is the value of the pixel $I(p_i)$ is the value of the pixel p_i in the image I .

2.3 Topological watershed

The original concept behind topological watershed [22] is to define a “topological thinning” that transforms the image while preserving some topological properties, namely the number of connected components of each lower cross-section as we will explain in the following.

Before introducing the topological watershed process, algorithm 7, we define some basic notions. Some of these notions will be resumed in next paragraph for mathematical formulation. Let F be

grayscale image and λ be a grey level, the lower cross-section $\overline{F}_\lambda$ is the set composed of all the points having an altitude strictly lower than λ. A point x is said to be W-destructible for F if its altitude can be lowered by one without changing the number of connected components of $\overline{F}_\lambda$, with $k = F(x)$. A map G is called a W-thinning of F if it may be obtained from F by iteratively selecting a W-destructible point and lowering it by one. A topological watershed of F is a W-thinning of F which contains no W-destructible point. The major feature of this transform is to produce a grayscale image.

The following algorithms give a global description of the computing process. Note that this process is repeated on loop until no W-destructible point remains.

Algorithm 7 : Topological watershed process
1. For all p in E, check the number of connected components of the lower cross-section at the level of p which are adjacent to p. 2. Lower the value of p by one if this number is exactly one

For mathematical formulation, we follow description provided in [11]. We start by defining a simple point in a graph, in a sense which is adapted to the watershed, and then we extend this notion to weighted graphs through the use of lower sections [22].

Couprie and al. define a transform that acts directly on the grayscale image, by lowering some points in such a manner that the connectivity of each lower cross-section $\overline{F}_\lambda$ is preserved. The regional minima of the result, which have been spread by this transform, can be interpreted as the catchment basins. The formal definition relies on a the following particular notion of simple point:

Definition 2.3 (Simple point) (2.16)

Let $G = (E, \Gamma)$ be a graph and let $X \subset E$.
The point $x \in X$ is simple (for $\overline{X}$) if the number of connected components of $\overline{X} \cup \{x\}$ equal to the number of connected components of $\overline{X}$. In other words, x is simple (for $\overline{X}$) if x is adjacent to exactly one connected component of $\overline{X}$.

Now, we can define more formerly destructible point, and the topological watershed:

Definition 2.4 (Topological watershed) (2.17)

Let $F \in F(E)$, $x \in E$ and $k = F(x)$. The point x is destructible (for F) if x is simple (for $\overline{F}_k$). We say that $W \in F(E)$ is a topological watershed of F if W may be derived from F by iteratively lowering destructible points by one until stability (that is, until all points of E being non-destructible for W).
Actually checking whether a point is w-destructible, or not, cannot be done locally if the only available information is the graph (E, Γ) and the function F and a point may also be lowered several times until it is no more w-destructible. Couprie and al. [23] propose a new algorithms making possible to perform this test on all the vertices of a weighted graph in linear time, and also to check directly how low the W-destructible point may be lowered until it is no more w-

destructible, thanks to the component tree which may be built in quasi-linear time. In the following, we introduce Couprie's functions to identify W-destructible point.

Algorithm 8 : Function W-destructible

Input : $F, C(\overline{F}), \Psi$;

1. $V \leftarrow$ Set of element of $C(\overline{F})$ pointed by $\Psi(q)$ for all q in $\Gamma^{-1}(p)$;
2. **If** $(V = \phi)$ **then** return $[\infty, \phi]$;
3. $[k_m, c_m] \leftarrow$ **HighestFORK** $(C(\overline{F}), V)$;
4. **If** $[k_m, c_m] = [\infty, \phi]$ **then** return $\min(V)$;
5. **If** $(k_m \leq F(p))$ **then** return $[k_m, c_m]$ **else** return $[\infty, \phi]$;

Previous algorithm gives correct results with regard to the definition (2.17) and is linear in time complexity with respect to the number of neighbors of p .

Checking whether a point is w-destructible or not, involves the computation of the highest fork of different elements of the set V(p),see algorithm 9. This may require a number of calls to BLCA (Binary lowest common Ancestor) which is quadratic with respect to the cardinality of V (p): every pair of elements of V (p) has to be considered.

Algorithm 9 : Function HighestFORK

Input : C : *a component tree,*
V *: a set of components of C*

1. $[k_1, c_1] \leftarrow \min(V)$; *// let [k2,c2]...[kn,cn]be the other elements of V*
2. $k_m \leftarrow k_1$;
3. $c_m \leftarrow c_1$;
4. **for** i **from** 2 **to** n **do**
5. $[k, c] \leftarrow$ BLCA $(C, [k_i, c_i], [k_m, c_m])$;
6. **If** $[k, c] \neq [k_i, c_i]$ **then** $k_m \leftarrow k_1$;
7. $c_m \leftarrow c_1$;
8.
9. **If** $(k_m = k_1)$ **then return** $[\infty, \phi]$ **else return** $[k_m, c_m]$;

The **HighestFork** algorithm returns the highest fork of the set V , or the indicator $[\infty, \phi]$ if there is no highest fork. This algorithm makes (n- 1) calls of the BLCA operator, where n is the number of elements in V .

Let C be a component tree, let V be a set of components of C, we denote by min(V) an element of V which has the minimal altitude. For following algorithm, we assume that C is represented in a convenient manner for BLCA.

Thus, we must propose a criterion for the selection of the remaining W-destructible points, in order to avoid multiple selections of the same point. Couprie and al. introduce the idea to give the greatest priority to a W-destructible point which may be lowered down to the lowest possible value. They prove that an algorithm which uses this strategy never selects the same point twice. A priority queue could be used to select W-destructible points in the appropriate order. Here, we present their specific linear watershed algorithm which may be used when the grayscale range is small.

Algorithm 10 : Topological watershed [Couprie]

Data : $F, C(\overline{F}), \Psi$;

Result : F ;

1. **For** k **from** $k_{\min}$ **to** $(k_{\max}-1)$ **do** $L_k \leftarrow \phi$
2. **For all** $(p \in E)$ **do**
3. $[i,c] \leftarrow W - Destructible(F, p, C(\overline{F}), \Psi)$
4. **If** $(i \neq \infty)$ **then**
5. $L_{i-1} \leftarrow L_{i-1} \cup \{p\}$;
6. $K\{p\} \leftarrow i-1$;
7. $H\{p\} \leftarrow$ pointer to $[i,c]$;
8. **end if**
9. **end for**
10. **For** k **from** $k_{\min}$ **to** $(k_{\max}-1)$ **do**
11. **While** $(\exists p \in L_k)$ **do**
12. $L_k = L_k / \{p\}$;
13. **If** $(K(p) = k)$ **then**
14. $F(p) \leftarrow k$;
15. $\Psi(p) \leftarrow H(p)$;
16. **For all** $(q \in \Gamma(p), k < F(q))$ **do**
17. $[i,c] \leftarrow W - Destructible(F, q, C(\overline{F}), \Psi)$;
18. **If** $(i = \infty)$ **then** $k(p) \leftarrow \infty$;
19. **Else if** $(k(p) \neq (i-1))$ **then**
20. $L_{i-1} \leftarrow L_{i-1} \cup \{q\}$;
21. $k(p) \leftarrow (i+1)$;
22. $H\{q\} \leftarrow$ pointer to $[i,c]$;

2.4 Watershed transform based on local condition

There is a big similarity between this approach and the drop of water one. Actually basin surface increases in a progressive manner. The local condition of label continuity is iteratively applied along the steepest descent path that reaches the basin minimum. The downhill algorithm, the hill climbing algorithm and the toboggan algorithm are based on this approach. More details of the first two algorithms are given by [24][25] and the toboggan algorithm will be detailed later in this section. Differences between these three algorithms lie in the processing strategy and data structure as shown in [6].

For mathematical formulation, we follow description provided in [7]. In witch Audrier and al. start by presenting the following catchment basin formulation $CB_{LC}(m_i) = \{v \in \mathrm{V}, \mathrm{L}(v) = \mathrm{L}(m_i)\}$, since local condition watershed assigns to each pixel the label of some minimum m_i. Thus watershed can be defined as follow. We recall that the condition $\mathbf{P}_{steepest}(v) \neq \{\ \}$ means that v has at least one lower neighbor.

Definition 2.5 (Watershed based on local condition) (2.18)

For any lower complete image L_{CB}, a function L assigning a label to each pixel is called watershed segmentation if:

a) $L(m_i) \neq L(m_j) \forall i \neq j$, With $\{m_k\}$, the set of minima of L_{LC}.

b) For each pixel v with $\mathbf{P}_{steepest}(v) \neq \{\ \}$, $\exists p \in \mathbf{P}_{steepest}(v), \mathrm{L}(v) = \mathrm{L}(p)$.

As we mentioned earlier, we will introduce the toboggan algorithm [24, 26] as a reference of the local condition watershed approach. Actually this algorithm is referred as a drainage analogy. It seeks to identify the steepest descent from each pixel of the gradient magnitude of the input image to a local minimum of the topographic surface. Then pixels that belong to the same minima are merged by assigning them a unique label. Sets of pixels having the same label will define catchment basins. The resulting watershed regions are divided by a boundary path which will build the watershed lines.

Let consider $G : D \rightarrow R^+$ as a gradient magnitude image, where D is the indexing domain of the image. D can be decomposed into a finite number of disjoint level sets since pixels are sorted in the increasing order. Sets can be denoted by: $D_h = \{p \in D \mid G(p) = h\}$. Lin and al [24] define the following pixels classes: Class C_1 refers to all pixels p in D_h with an altitude strictly greater than the altitude of its lowest neighbor. Class C_2 refers to all pixels p in D_h belonging to a connected component with one or more catchment basin and $p \notin C_1$. Finally, class C_3 refers to all pixels p in D_h belonging to a connected component without any catchment basin. Thus we can give a global description of the computing process.

Algorithm 11 : Toboggan watershed process

1. Records the sliding directions for all $(p \in C_1) \cup (p \in C_2)$ in D
   a. Records the lowest neighbours of all $(p \in C_1)$ in D.
   b. Grown region from all $(p \in C_1)$
2. Assign label for all $(p \in C_3)$
3. Assign label to each unlabeled image by first tobogganing then backtracking using best first search.

Based on this process, authors introduce the following algorithm to compute watershed.

Algorithm 12: Toboggan Algorithm [Lin and al.]

Data : Img : a gradient magnitude image;

Result : L : a label image, Q : empty FIFO queue ;

1. **For all** $(p \in D)$ **do** *//Simulation of sliding for all C_1 pixels*
2. $h = G(p)$
3. $h_{MIN} = \min\{G(q), q \in Neighbor(p)\}$
4. **If** $(h > h_{MIN})$ **then**
5. $S = \{(q \mid G(q) = h_{MIN}) \,\&\, \&(q \in Neighbor(p))\}$
6. $SlidingList(p) = S$

```
7.              Q ← p
8.              GrowingDist(p) = 0
9.              Else if :
10.             SlidingList(p) = φ
11.         End if
12.     End for
13.     While Q ≠ φ do //Simulation of keep- sliding for all C2 pixels
14.         p ← Q
15.         d = GrowingDist(p) + 1
16.         h = G(p)
17.         For all (q ∈ Neighbor(p)) and (G(q) = h) do
18.             If (SlidingList(q) = φ) then
19.                 Append (p) to SlidingList(q)
20.                 GrowingDist(q) = d
21.                 Q ← q;
22.             Else If (GrowingDist(q) = d) then
23.                 Append (p) to SlidingList(q)
24.             End if
25.         End while
26.     For all (p0 ∈ D) and (SlidingList(p0) = φ) do // labelling C3 pixels
27.         If L(p0) is not assigned then
28.             L(p0) =new_label
29.             h = G(p0)
30.             While Q ≠ φ do
31.                 p ← Q
32.                 For all (p ∈ Neighbor(p)) and (G(q) = h) do
33.                     If L(q) is not assigned then
34.                         (L(q) = L(p0))
35.                         Q ← p
36.                     End if
37.                 End for
38.             End while
39.         End if
40. End for
41. For all (p ∈ D) do // Tobogganing – Depth first search
42.     Resolve(p)
43. End For
```

Algorithm 13: Resolve function

```
Input : Pixel site p
1.  If L(p) is not assigned then
2.      S = SlidingList(p)
3.      For all (q ∈ S) do
4.          Resolve(q)
```

```
5.      End for
6.      If S has a unique label α then
7.          L(p) = α
8.      Else
9.          L(p) = RIDGE_label
10.     End if
11. End if
```

2.5 Watershed transform based on minimum spanning forest

The original idea is very close to the second case of the path cost minimization based watershed that consist on building a spanning forest from a graph as we introduced in section 2.2. Actually, the beginning was with Meyer [27] who proposes to compute watershed transform from a weighted neighborhood graph whose nodes are the catchment basins corresponding to the minima of the image. Arcs of the graph, that separate neighbor catchment basins, are weighted by the altitude of the pass between these basins. Extracted minimum spanning forests define partitions that are considered solution of watersheds. It's important to mention that returned solutions are multiple. Authors established also the links between the minimum spanning forest and flooding from marker algorithms. Trough Meyer's bases, Cousty and al. [28] introduce the watershed-cuts and establish the optimality of this approach by showing the equivalence between the watershed-cuts and the separations induced by minimum spanning forest relative to the minima.

For mathematical foundations, we will follow the notations in [28] to present some basic definitions to handle with minimum spanning forest cuts and watershed-cuts.

Let G be graph with $G = (V(G), E(G))$. $V(G)$ is a finite set of vertex of G. Unordered pairs of $V(G)$, called also edge of G, constitute the element of $E(G)$ set. Let denote the set of all maps from E to $\mathbb{Z}$ by F and we consider that any maps of F weights the edges of G. Let $\mathrm{F} \subseteq F$ and $u \in E(G)$, $\mathrm{F}(u)$ will refers to the altitude of u and $\mathrm{M}(\mathrm{F})$ will refers the graph whose vertex set and edge set are, respectively, the union of the vertex sets and edge sets of all minima of F.

Let X and Y be two sub-graphs of G. We say that Y is a forest relative to X if Y is an extension of X and for any extension $\mathrm{Z} \subset \mathrm{X}$ of X, we have $\mathrm{Y}=\mathrm{Z}$ whenever $\mathrm{V(Z)}=\mathrm{V(Y)}$.

(i) Y is said a spanning forest relative to X (for G) if Y is a forest relative to X and if $\mathrm{V(Y)}=\mathrm{V}$. In this case, there exists a unique cut S for Y. It is composed by all edges of G whose extremities are in two distinct components of Y. Since Y is an extension of X, it can be seen that this unique cut S (induced by Y) is also a cut for X.

(ii) Y is said a minimum spanning forest relative to X (for F, in G) if Y is a spanning forest relative to X and if the weight of Y is less than or equal to the weight of any other spanning forest relative to X. In this case, S is considered as a minimum spanning forest cut for X.

Trough these equivalences, Cousty demonstrate that the set $\mathrm{S} \subseteq \mathrm{E}$ is a minimum spanning forest cut for $\mathrm{M}(\mathrm{F})$ if and only if S is a watershed cut of F, that can be computed by any minimum spanning tree algorithm. And he proposes a linear algorithm to compute it using a new 'stream' notion that we will not detail in this section. Only the stream algorithm will be introduced.

Now, before presenting the watershed-cuts algorithm we just recall the definition of the minimal altitude of an edge. Let denote by $F^-(x)$ the map from V to $\mathbb{Z}$ such that for any $x \in V$, $F^-(x)$ is the minimal altitude of an edge which contains x. Then a path $\pi = \langle x_0, ..., x_l \rangle$, is considered as a path of steepest decent for F (in G) if for any $i \in [1, l]$, $F(\{x_{i-1}, x_i\}) = F^-(x_{i-1})$.

In the following, we introduce the watershed-cuts computing algorithm and stream function.

Algorithm 14: Watershed-cuts algorithm [Cousty and al.]

```
Data : (V,E,F) : Edge-weighted graphs;
Result: Ψ: a flow mapping of F
1.  Foreach (x ∈ V) do ψ(x) ← NO_LABEL ;
2.  nb_labs ← 0
3.  Foreach (x ∈ V) such that (ψ(x)=NO_Label) do
4.          [L,Lab] ← Stream(V,E,F,ψ,x) ;
5.          If (lab = −1) then
6.              nb_labs++
7.              Foreach (y ∈ L) do ψ(y) ← nb_labs ;
8.          Else
9.              Foreach (y ∈ L) do ψ(y) ← labs ;
```

Algorithm 15: Stream function [Cousty and al.]

```
Data : (V,E,F) : Edge-weighted graphs; Ψ : a label of V; x : point of V;
Result : [L, lab] : L is a flow obtained from x (source of L) ; lab is the associated label to
         an Θflux included in L or (–1).
1.  L ←{x}
2.  L'←{x} // the set of sources not yet explored of L
3.  While there exists (y ∈ L') do
4.      L' ← L' \ {y};
5.      breadth_first ← TRUE ;
6.      While (breadth_first) and (∃ {y,z} ∈ E / z ∉ L and F({y,z})= F(y)) do
7.          If (Ψ(z) # No_label) then
8.                              return [L,Ψ(z)] // exist an Θflow L already labelled
9.          Else if ( F⁻(z) < F⁻(y) ) then
10.                 L ← L ∪ {z}; // z is the only well of L
11.                 L' ← {z}; // switch the in-depth exploration first
12.                 breadth_first ← FALSE
13.             Else
14.                 L ← L ∪ {z}; // therefore z is a well of L
15.                 L' ← L' ∪ {z}; // continue exploration in width first
16. return [L,-1]
```

3. Classification of watersheds transforms

In this section we will learn from different syntheses present in Roerdink [6] and Audigier [7] works. The following table summarizes some characteristic of introduced watershed transform. Selected criteria are justified by our objective to identify the most suitable algorithm for parallel implementation.

	Watershed based on					
	Flooding	***Path-cost minimization***		***Topology***	***Local condition***	***MSF***
		TD	***IFT***			
	Vincent & Soille [15]	Meyer [17]	Lotufo [25]	Couprie [9]	Lin [26]	Cousty [4]
Defined in	Disc. cont. space	Disc. cont. space	Only on discrete space	Disc. cont. space	Disc. cont. space	Only on discrete space
Classified as	Line WT	Line WT	Region WT	Line WT	Region WT	Region WT
Gives unique solution	Yes	Yes	No	No	No	No
Preserve topology	No	No	Yes	Yes	No	Yes
Requires a sorting step	Yes	No	No	No	No	No
Use of hierarchical queue	Yes	Yes	Yes	Yes	Yes	No
Minima computing	Yes	Yes	No	No	-	No
Is linearity	linear	-	linear	Linear*	-	Linear

TABLE 1: Comparison between main watersheds transforms (definitions & algorithms)

The starting point is the definition space; we note that IFT-Watershed and MSF-Watershed definitions are limited to the discrete space while the other watersheds definitions are spread into continue space. IFT-Watershed, MSF-watershed and LC-Watershed form the region based watershed transform family since pixels are assigned to basins. Flooding-Watershed, TD-Watershed and Topological-Watershed form the line based watershed family since some pixels are labeled as watershed. Only Topological-Watershed defines lines that consistently separate basins while Flooding-Watershed and TD-Watershed merely swing between thick and disconnected watershed lines. Through definitions, only Flooding-Watershed and TD-Watershed return unique solution while all other definitions return multiple solutions. Note that set of solutions returned by the IFT-Watershed can be unified by creating litigious zones when solutions differ [7]. All six algorithms, that don't exactly include their definitions, return unique solution but don't preserve the number of connected components of the original input image. Actually, Vincent-Soille, Meyer and Lin's algorithm don't preserve important topological features. Only Lotufo, Couprie and Cousty's algorithm are correct from this point of view.

Regarding computing process, only Flooding-Watershed needs pixel's sorting while others transforms will pass this costly step. But this does not preclude associated algorithms to use hierarchical structures when implementing. Except Cousty's algorithm that doesn't need any hierarchical queue. Vincent-Soille and Meyer's algorithms impose also a prior minima computation, which is not the case of the others. For complexity, observe that Vincent and Soille algorithm runs in linear with respect to the number N of pixels in the image which is processed. In most current situations of image analysis, where the number of possible values for the priority function is limited and the number of neighbors of a point is small constant, Couprie's algorithm

runs also in linear time with O(n + m) complexity. Lotufo and Cousty's algorithm run also in linear time. Cousty's algorithm is executed at most O(|E|) times.

4. Conclusion

In this paper, we have presented an intensive study of all existing watershed transform in the discrete case: WT based on flooding, WT based on path-cost minimization, watershed based on topology preservation, WT based on local condition and WT based on minimum spanning forest.

The first major contribution in this paper is the global nature of the proposed study. In fact, for each approach, we start by giving informal definition, then we present processing procedure followed by mathematical foundations and the algorithm of reference. Recent publications based on some approach are also presented and discussed. The second contribution concerns algorithms' classification according to criteria of recursion, complexity, basins computing and topology preservation.

In our future work we will present a parallel MSF-Watershed algorithm suitable for shared memory parallel machines based on Jean Cousty stream function already presented.

5. REFERENCES

[1] J. Maxwell. "On hills and dales." Philosophical Magazine, vol. 4/40, pp. 421-427, Dec. 1870.

[2] S. Beucher and C. Lantuéjoul. "Use of watersheds in contour detection." International Workshop on Image Processing, Real-Time Edge and Motion Detection/ Estimation, 1979.

[3] S. Beucher and F. Meyer. "The morphological approach to segmentation: the watershed transformation." In Dougherty, E., ed.: Mathematical Morphology in Image Processing, Marcel Decker, pp. 433-481,1993.

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

[5] M. Straka, A. La Cruz, A. Köchl, M. Šrámek, E. Gröller and D. Fleischmann. "3D Watershed Transform Combined with a Probabilistic Atlas for Medical Image Segmentation." Journal of Medical Informatics & Technologies, 2003.

[6] J.B.T.M. Roerdink and A. Meijster. "The watershed transform: Definitions, algorithms and parallelization strategies." Fundamenta Informaticae, Special issue on mathematical morphology vol. 41, pp. 187-228, Jan. 2001.

[7] R. Audigier and R.A. Lotufo. "Watershed by image foresting transform, tie-zone, and theoretical relationship with other watershed definitions." In Mathematical Morphology and its Applications to Signal and Image Processing, pp. 277–288, 2007.

[8] L. Najman and M. Couprie. "Watershed algorithms and contrast preservation." Lecture Notes in Computer Science, vol. 2886, pp. 62-71, 2003.

[9] P. Soille. "Morphological Image Analysis." Springer-Verlag Berlin and Heidelberg GmbH & Co. K, April 1999.

[10] E. Dougherty and R. Lotufo. "Hands-on Morphological Image Processing." SPIE Publications, 2003.

[11] L. Najman, M. Couprie and G. Bertrand. “Watersheds, mosaics and the emergence paradigm.” Discrete Applied Mathematics, vol. 147, pp. 301-324, April 2005.

[12] L. Vincent and P. Soille. “Watersheds in digital spaces - An efficient algorithm based on immersion simulations.” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, pp. 583-598, 1991.

[13] P. Shengcai and G. Lixu. “A Novel Implementation of Watershed Transform Using Multi-Degree Immersion Simulation.” 27th Annual International Conference of the Engineering in Medicine and Biology Society, pp. 1754 – 1757, 2005.

[14] F. Meyer. “Topographic distance and watershed lines.” Signal Processing, vol. 38, pp. 113-125,1993.

[15] L. Najman and M. Schmitt. “Watershed of a continuous function.” Signal Processing, vol. 38, pp. 68-86, 1993.

[16] A. X. Falcão, J. Stolfi, and R. A. Lotufo. “The Image Foresting Transform: Theory, Algorithms, and Applications.” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, pp.19-29, 2004.

[17] R. A. Lotufo and A. X. Falcão. “The Ordered Queue and the Optimality of the Watershed Approaches.” Procs. 5th International Symposium on Mathematical Morphology, pp. 341-350, 2000.

[18] E.W. Dijkstra. “A Note on Two Problems in Connection with Graphs.” Numeriche Mathematik, vol 1, pp. 269-271, 1959.

[19] U. Pape. “Implementation and Efficiency of Moore Algorithms for the Shortest Root Problem.” Mathematical Programming, vol. 7, pp. 212-222, 1974.

[20] R. B. Dial, F. Glover, D. Karney, and D. Klingman. “A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees.” In Networks, vol 9, pp. 215-248,1979.

[21] R.A. Lotufo, A. X. Falcão and F. A. Zampirolli. “IFT-Watershed from Gray-Scale Marker.” 15th Brazilian Symposium on Computer Graphics and Image Processing, Vol 2, 2002.

[22] M. Couprie and G. Bertrand. “Topological grayscale watersheds transform.” SPIE Vision Geometry V Proceedings, vol. 3168, pp. 136-146, 1997.

[23] M. Couprie, L. Najman and G. Bertrand. “Quasi-linear algorithms for the topological watershed.” Journal of Mathematical Imaging and Vision, vol. 22, pp. 231-249, 2005.

[24] Y.C. Lin, Y.P. Tsai, Y.P Hung and Z.C. Shih. “Comparison Between Immersion-Based and Toboggan-Based Watershed Image Segmentation.” 15th IEEE Transactions on Image Processing, vol. 3, pp. 632-640, 2006.

[25] V.O. Ruiz, J.I.G. Llorente, N. S. Lechon and P.G. Vilda. “An improved watershed algorithm based on efficient computation of shortest paths.” Pattern Recognition, vol. 40,pp. 1078-1090, 2007.

[26] E.N. Mortensen and W.A. Barrett. “Toboggan-based intelligent scissors with a four-parameter edge model.” In IEEE Conf. Computer Vision and Pattern Recognition, pp. 452–458, 1999.

[27] F. Meyer. “Minimum Spanning Forests for Morphological Segmentation.” Proc. Second International Conference on Math. Morphology. and Its Applications to Image Processing, pp. 77-84, 1994.

[28] J. Cousty, G. Bertrand, L. Najman and M. Couprie. “Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle.” IEEE Transaction on Pattern Analysis and Machine Intelligence, pp. 1362-1374, 2009.