Journal of Mathematical Imaging and Vision 22: 231–249, 2005
© 2005 Springer Science + Business Media, Inc. *Manufactured in The Netherlands.*

Quasi-Linear Algorithms for the Topological Watershed

MICHEL COUPRIE, LAURENT NAJMAN AND GILLES BERTRAND
Laboratoire A2SI, Groupe ESIEE BP99, 93162 Noisy-le-Grand Cedex France; IGM, Unité Mixte de Recherche CNRS-UMLV-ESIEE UMR 8049
m.couprie@esiee.fr
l.najman@esiee.fr
g.bertrand@esiee.fr
url: www.esiee.fr/~info/a2si/gdi.html

Abstract. The watershed transformation is an efficient tool for segmenting grayscale images. An original approach to the watershed (Bertrand, *Journal of Mathematical Imaging and Vision*, Vol. 22, Nos. 2/3, pp. 217–230, 2005.; Couprie and Bertrand, *Proc. SPIE Vision Geometry VI*, Vol. 3168, pp. 136–146, 1997.) consists in modifying the original image by lowering some points while preserving some topological properties, namely, the connectivity of each lower cross-section. Such a transformation (and its result) is called a W-thinning, a topological watershed being an "ultimate" W-thinning. In this paper, we study algorithms to compute topological watersheds. We propose and prove a characterization of the points that can be lowered during a W-thinning, which may be checked locally and efficiently implemented thanks to a data structure called component tree. We introduce the notion of M-watershed of an image F, which is a W-thinning of F in which the minima cannot be extended anymore without changing the connectivity of the lower cross-sections. The set of points in an M-watershed of F which do not belong to any regional minimum corresponds to a binary watershed of F. We propose quasi-linear algorithms for computing M-watersheds and topological watersheds. These algorithms are proved to give correct results with respect to the definitions, and their time complexity is analyzed.

Keywords: discrete topology, mathematical morphology, watershed, component tree, segmentation

Introduction

The watershed transformation was introduced as a tool for segmenting grayscale images by Beucher and Lantuéjoul [3] in the late 70's, and is now used as a fundamental step in many powerful segmentation procedures. A popular presentation of the watershed is based on a flooding paradigm. Let us consider a grayscale image as a topographical relief: the gray level of a pixel becomes the altitude of a point, the basins and valleys of the relief correspond to the dark areas, whereas the mountains and crest lines correspond to the light areas (Fig. 1(a_1) and (a_2)). Let us imagine the surface of this relief being immersed in still water, with holes pierced in local minima. Water fills up basins starting at these local minima, and dams are built at points where waters coming from different basins would meet. As a result, the surface is partitioned into regions or basins which are separated by dams, called watershed lines.

Efficient watershed algorithms based on such immersion simulation were proposed by Vincent and Soille [34], Beucher and Meyer [4] and Meyer [22] in the early 90's. Many different watershed paradigms and algorithms have been proposed until now, see [27] for a review. In the continuous space, a definition and some properties of the watersheds of "regular" functions have been studied by Najman and Schmitt [26]. However, until recently, there was no general framework including a precise definition, strong properties, and algorithms which may be proved to indeed implement the definition.

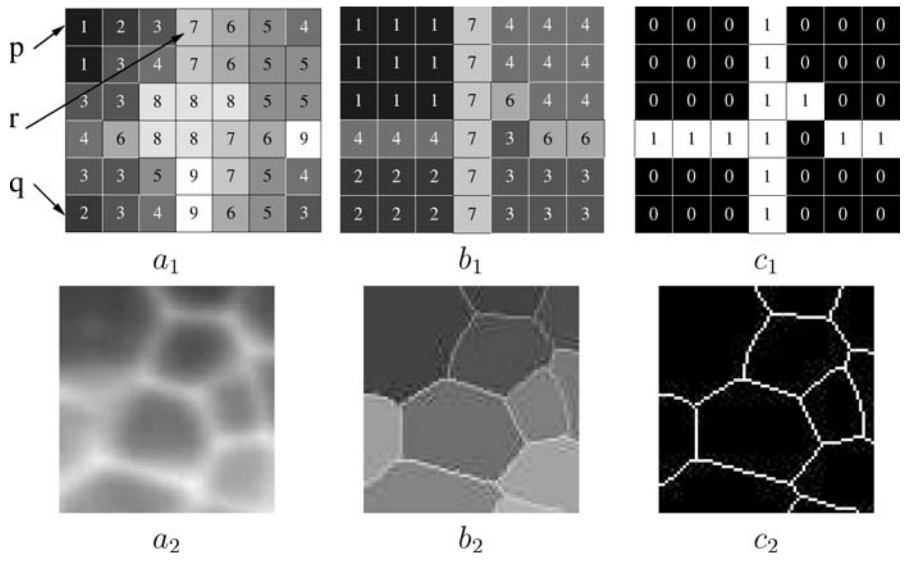

Figure 1. a_1, a_2: original images; b_1 (resp. b_2): topological watershed of a_1 (resp. a_2); c_1 (resp. c_2): W-crest of a_1 (resp. a_2), in white.

A different approach to watersheds, originally proposed by Bertrand and Couprie [9], is developed in [1, 25]. In this approach, we consider a transformation called topological watershed, which modifies a map (e.g., a grayscale image) while preserving some topological properties, namely, the connectivity of each lower cross-section. The motivation for such a condition will appear a little later, when we will discuss the properties of this transformation. Let F be a map and λ be a number, the lower cross-section $\bar{F}[\lambda]$ is the set composed of all the points having an altitude strictly lower than λ (Fig. 3). A point x is said to be W-destructible for F (where W stands for Watershed) if its altitude can be lowered by one without changing the number of connected components of $\bar{F}[k]$, 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. This transformation has the effect of spreading the regional minima of the map (see Fig. 1). Let F be a map and let G be a topological watershed of F, the set of points which do not belong to any regional minimum of G is called a W-crest of F. The W-crest of F corresponds to a binary watershed of F (see Fig. 1(c_1) and (c_2)).

In [1], Bertrand develops a framework in which fundamental properties of topological watersheds are proved, and where the notion of *separation* plays a central role. Consider a map F, we can say that two points p and q are k-separated if there exists a path between p and q, the maximal altitude of which is $k-1 > \max(F(p), F(q))$, and if there is no path between p and q with a maximal altitude strictly less than $k-1$ (notice that this notion of k-separation between two points is closely related to the notion of grayscale connectivity introduced by Rosenfeld [28], see also [6]). For example, in Fig. 1(a_1), the point p and the point q are 5-separated, but the point p and the point r are not separated. We say that a map G, such that $G \leq F$, is a separation of F, if whenever p and q are k-separated for F, p and q are k-separated for G. We say that G is a strong separation of F if G is a separation of F and if the minima of G are "extensions" of the minima of F. In Fig. 1, it can be checked that b_1 is a strong separation of a_1.

One of the main theorems proved in [1] (the strong separation theorem) states that G is a W-thinning of F *if and only if* G is a strong separation of F.

The "if" part of the theorem corresponds to a notion of contrast preservation. We will say informally that a transformation "preserves the contrast" if the transformation preserves the altitude of the minima of the image and if, when two minima are separated by a crest in the original image, they are still separated by a crest of the same altitude in the transform. For example in Fig. 1, if we take any two minima which are k-separated in a_i ($i = 1, 2$) for a given k, we know that they are k-separated in b_i since b_i is a W-thinning of a_i. This constrast preservation property is not satisfied in general by the most popular watershed algorithms (see [23, 25]).

The "only if" part of the theorem mainly states that, if one needs a transformation which preserves the contrast in this sense, then this transformation is necessarily a W-thinning. This remarkable result shows that the topological watershed is a fundamental tool to obtain a contrast preserving watershed transformation.

In this paper, we study algorithms to compute topological watersheds. A naive algorithm could be the following: for all p in E (n points), check the number of connected components of the lower cross-section at the level of p which are adjacent to p (cost for each point p: $O(n)$ with a classical connected component labelling algorithm), lower the value of p by one if this number is exactly one. Repeat this whole process until no W-destructible point remains. Consider an image which consists of a single row of $n + 2$ points, such that each point has an altitude of g except for the two points at the beginning and at the end of the row, which have an altitude of 0 (with any positive integers n, g). The outer loop will be executed g times. The time complexity of this naive algorithm is thus at least in $O(n^2 \times g)$.

We reduce the complexity by two means.

First, we propose and prove a new characterization of the W-destructible points which may be checked locally and efficiently: the total time for checking the W-destructibleness of all the vertices in a graph with n vertices and m arcs is in $O(n + m)$. We obtain this result thanks to a data structure called component tree, which may be constructed in quasi-linear time [24], that is, in $O(N \times \alpha(N))$ where $N = n + m$ and $\alpha(N)$ is a function which grows extremely slowly with N (we have $\alpha(10^{80}) \approx 4$). This complexity can be reached thanks to a reduction to the disjoint set problem [31].

Second, we propose different strategies to ensure that a point is lowered at most once during the execution of the algorithm. One of these strategies relies on the notions of $\tilde{M}$-point and M-watershed. A point p is an $\tilde{M}$-point if it is adjacent to a regional minimum and if it can be lowered by W-thinning down to the level of this minimum. An M-watershed is obtained by iteratively lowering $\tilde{M}$-points until stability. Recall that a W-crest of a map F is composed by the points which do not belong to any regional minimum of a topological watershed of F. We prove that the set of points which do not belong to any regional minimum of an M-watershed of F is always a W-crest of F, in other words, we can compute a W-crest by only lowering $\tilde{M}$-points. We propose a quasi-linear algorithm for computing an M-watershed—hence a W-crest—of a map.

We also propose a quasi-linear algorithm for the topological watershed transformation. These algorithms are proved to give correct results with respect to the definitions, and their time complexity is analyzed.

In order to ease the reading of the paper, we defer the proofs to the annex.

1. Topological Notions for Graphs

Let E be a finite set, we denote by $\mathcal{P}(E)$ the set of all subsets of E. Throughout this paper, Γ will denote a binary relation on E (thus, $\Gamma \subseteq E \times E$), which is reflexive (for all p in E, $(p, p) \in \Gamma$) and symmetric (for all p, q in E, $(q, p) \in \Gamma$ whenever $(p, q) \in \Gamma$). We say that the pair (E, Γ) is a *graph*, each element of E is called a *vertex* or a *point*. We will also denote by Γ the map from E into $\mathcal{P}(E)$ such that, for any p in E, $\Gamma(p) = \{q \in E; (p, q) \in \Gamma\}$. For any point p, the set $\Gamma(p)$ is called the *neighborhood of* p. If $q \in \Gamma(p)$ then we say that p and q are *adjacent* or that q is a *neighbor of* p. If $X \subseteq E$ and q is adjacent to p for some $p \in X$, we say that *q is adjacent to X*.

For applications to digital image processing, assume that E is a finite subset of $\mathbb{Z}^n$ ($n = 2, 3$), where $\mathbb{Z}$ denotes the set of integers. A subset X of E represents the "object", its complementary $\bar{X} = E \setminus X$ represents the "background", and Γ corresponds to an adjacency relation between points of E. In $\mathbb{Z}^2$, Γ may be one of the usual adjacency relations, for example the 4-adjacency or the 8-adjacency in the square grid. Let us recall briefly the usual notions of path and connected component in graphs.

Let (E, Γ) be a graph, let $X \subseteq E$, and let $p_0, p_k \in X$. A *path from p_0 to p_k in X* is an ordered family $(p_0, \dots, p_k)$ of points of X such that $p_{i+1} \in \Gamma(p_i)$, with $i = 0 \dots k - 1$.

Let $p, q \in X$, we say that *p and q are linked for X* if there exists a path from p to q in X. We say that *X is connected* if any p and q in X are linked for X. We say that a subset Y of E is a *connected component of X* if $Y \subseteq X$, Y is connected, and Y is maximal for these two properties, i.e., if $Y \subseteq Z \subseteq X$ and if Z is connected, then $Z = Y$. In the sequel of the article, we will assume that E is connected.

We are interested in transformations that preserve the number of connected components of the background. For that purpose, we introduce the notion of W-simple

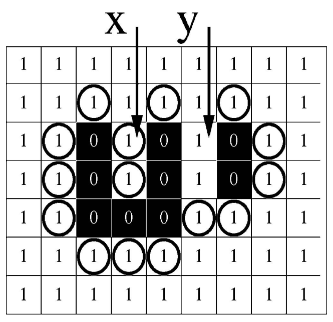

Figure 2. A set X (the 1's) and its complement $\bar{X}$ (the 0's). The W-simple points are circled.

point in a graph. Intuitively, a point of X is W-simple if it may be removed from X while preserving the number of connected components of $\bar{X}$.

Definition 1. Let $X \subseteq E$, let $p \in X$.

- p is *a border point* (*for* X) if p is adjacent to $\bar{X}$.
- p is *an inner point* (*for* X) if p is not a border point for X.
- p is *separating* (*for* X) if p is adjacent to at least two connected components of $\bar{X}$.
- p is W*-simple* (*for* X) if p is adjacent to exactly one connected component of $\bar{X}$.

Notice that a point which is not W-simple, is either an inner point or a separating point. In Fig. 2, the points of the set X are represented by "1"s, and the 4-adjacency is assumed, as for all subsequent examples. The points which are W-simple are circled. It may be easily seen that one cannot locally decide whether a point is W-simple or not. Consider the points x and y in the third row: their neighborhoods are alike, yet x is W-simple (it is adjacent to exactly one connected component of $\bar{X}$), and y is not, since it is adjacent to two different connected components of $\bar{X}$.

2. Topological Notions for Weighted Graphs and Stacks

Now, we extend these notions to a weighted graph (E, Γ, F), where F is a function from E to $\mathbb{Z}$. A weighted graph is a model for a digital grayscale image; for any point $p \in E$, the value $F(p)$ represents the gray level of p. Let $k_{\min}$ and $k_{\max}$ be two elements of $\mathbb{Z}$ such that $k_{\min} < k_{\max}$. We set $\mathbb{K} = \{k \in \mathbb{Z}; k_{\min} \leq k < k_{\max}\}$, and $\mathbb{K}^+ = \mathbb{K} \cup \{k_{\max}\}$. We denote by $\mathcal{F}$ the set composed of all functions from E to $\mathbb{K}$. Let $F \in \mathcal{F}$, let $k \in \mathbb{K}^+$. We denote by $F[k]$ the set $\{p \in E; F(p) \geq k\}$; $F[k]$ is called a *level set* of F. Notice that $F[k_{\min}] = E$ and $F[k_{\max}] = \emptyset$.

Any function in $\mathcal{F}$ can be represented by its different level sets. For a given function, these level sets constitute a "stack": in fact, the datum of a function is equivalent to the datum of a stack. We give here a minimal set of definitions borrowed from [1] for stacks, which is is sufficient for our purpose; the interested reader should refer to [1] for a more complete presentation. Considering the equivalence between a function and its corresponding stack, we will use the same symbol for both of them.

Definition 2. Let $F = \{F[k]; k \in \mathbb{K}^+\}$ be a family of subsets of E.

- F is an *upstack on* E if $F[k_{\min}] = E$, $F[k_{\max}] = \emptyset$, and $F[j] \subseteq F[i]$ whenever $i \leq j$.
- F is a *downstack on* E if $F[k_{\min}] = \emptyset$, $F[k_{\max}] = E$, and $F[i] \subseteq F[j]$ whenever $i \leq j$.
- $\mathcal{S}^+$ (resp. $\mathcal{S}^-$) denotes the set of all upstacks (resp. downstacks) on E. Any element of $\mathcal{S}^+ \cup \mathcal{S}^-$ is called a *stack on* E.

Let F be a stack on E, we define the stack $\bar{F} = \{\bar{F}[k] = \overline{F[k]}; k \in \mathbb{K}\}$ which is called *the complement of* F. Let F be a stack on E, any element $F[k]$ of F is called a *section of* F *(at level* k*)*, or the k*-section of* F.

Let $F \in \mathcal{S}^+$ and let $G \in \mathcal{S}^-$. We define the *functions induced by* F *and* G, also denoted by F, G, such that for any $p \in E$:

$$F(p) = \max\{k \in \mathbb{K}; p \in F[k]\}$$
$$G(p) = \min\{k \in \mathbb{K}; p \in G[k]\}.$$

Important remark. Let $F \in \mathcal{F}$. Clearly, the level sets of F form an upstack (also denoted by F), and the function induced by the upstack F is precisely the function F. The complement $\bar{F}$ of the upstack F is a downstack. For any $k \in \mathbb{K}^+$, we have $\bar{F}[k] = \overline{F[k]} = \{p \in E; F(p) < k\} = \{p \in E; \bar{F}(p) \leq k\}$; and for any $p \in E$, we have

$$\bar{F}(p) = F(p) + 1.$$

Definition 3. Let $F \in \mathcal{F}$, let $k \in \mathbb{K}^+$. A connected component of a non-empty k-section of $\bar{F}$ is called a *component of* $\bar{F}$ (*at level* k), or a k*-component of* $\bar{F}$.

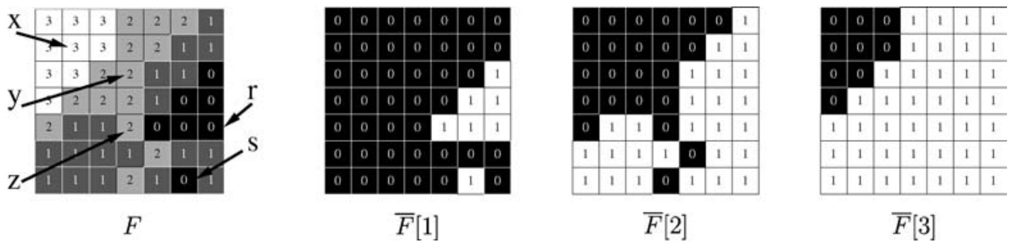

Figure 3. A grayscale image F and its lower sections $\bar{F}[1]$, $\bar{F}[2]$ and $\bar{F}[3]$ (in white). Notice that $\bar{F}[4] = E$ and $\bar{F}[0] = \emptyset$.

A component m of $\bar{F}$ is said to be a *minimum of* $\bar{F}$ (and also a *minimum of* F) if there is no other component of $\bar{F}$ which is included in m.

Let $p \in E$, the *component of* p *in* $\bar{F}$, denoted by $C(p, \bar{F})$ or simply by $C(p)$ when no confusion may occur, is defined as the component of $\bar{F}[k]$ which contains p, with $k = \bar{F}(p)$.

We denote by $\Gamma^-(p, F)$ the *set of lower neighbors of the point* p *for the function* F, that is, $\Gamma^-(p, F) = \{q \in \Gamma(p); F(q) < F(p)\}$. Notice that $\Gamma^-(p, F) = \Gamma^-(p, \bar{F})$. When no confusion may occur, we write $\Gamma^-(p)$ instead of $\Gamma^-(p, F)$.

Figure 3 shows a grayscale image F and three sections of $\bar{F}$. Since we use the 4-adjacency, $\bar{F}[2]$ is made of two components (in white), whereas $\bar{F}[3]$ is made of one component. The set $\bar{F}[1]$ is made of two components which are minima of $\bar{F}$. We have: $C(x, \bar{F}) = E$; $C(r, \bar{F})$ is the component of $\bar{F}[1]$ which contains six points; and $C(y, \bar{F}) = C(z, \bar{F})$: it is the unique component of $\bar{F}[3]$.

Definition 4. Let $F \in \mathcal{F}$, let $p \in E$, let $k = F(p)$.

- p is *a border point* (*for* F) if p is an border point for $F[k]$.
- p is *an inner point* (*for* F) if p is an inner point for $F[k]$.
- p is *separating* (*for* F) if p is separating for $F[k]$.

The point p is W*-destructible* (*for* F) if p is W-simple for $F[k]$. Let $v \in \mathbb{K}$, the point p is W*-destructible with lowest value* v (*for* F) if for any h such that $v < h \leq F(p)$, p is W-simple for $F[h]$, and if p is not W-simple for $F[v]$.

In other words, the point p is W-destructible for F if and only if p is a border point for F (i.e., $\Gamma^-(p) \neq \emptyset$) and all the points in $\Gamma^-(p)$ belong to the same connected component of $\bar{F}[k]$, with $k = F(p)$.

In Fig. 3, the points x, r, s are inner points, y is a W-destructible point (with lowest value 1), and z is a separating point.

Let $F \in \mathcal{F}$, let $p \in E$, let $v \in \mathbb{K}$ such that $v < F(p)$, we denote by $[F \setminus p \downarrow v]$ the element of $\mathcal{F}$ such that $[F \setminus p \downarrow v](p) = v$ and $[F \setminus p \downarrow v](q) = F(q)$ for all $q \in E \setminus \{p\}$. Informally, it means that the only difference between the function F and the function $[F \setminus p \downarrow v]$, is that the point p has been lowered down to the value v. We also write $[F \setminus p] = [F \setminus p \downarrow v]$ when $v = F(p) - 1$. Let p be a W-destructible point for F with lowest value v. If we consider $F' = [F \setminus p \downarrow v]$, it may be easily seen that for all h in $\mathbb{K}^+$, the number of connected components of $\bar{F}'[h]$ equals the number of connected components of $\bar{F}[h]$. That is to say, the value of a W-destructible point may be lowered by one or down to its lowest value without changing the number of connected components of any section of $\bar{F}$.

Definition 5. Let $F \in \mathcal{F}$. We say that $G \in \mathcal{F}$ is *a* W*-thinning* of F if

(i) $G = F$, or if
(ii) there exists a function H which is a W-thinning of F and there exists a W-destructible point p for H such that $G = [H \setminus p]$.

We say that G is a (*topological*) *watershed* of F if G is a W-thinning of F and if there is no W-destructible point for G.

Let $F \in \mathcal{F}$, let $p \in E$, let $v \in \mathbb{K}$. It may be easily seen that, if p is W-destructible with lowest value v, then $[F \setminus p \downarrow v]$ is a W-thinning of F and p is not W-destructible for $[F \setminus p \downarrow v]$; and that the converse is also true.

In other words, one can obtain a W-thinning of a function F by iteratively selecting a W-destructible point and lowering it by one. If this process is repeated until stability, one obtains a topological watershed of F.

Notice that the choice of the W-destructible point is not necessarily unique at each step, thus, in general, there may exist several topological watersheds for the same function.

In Fig. 4, we present an image 4(a) and a topological watershed 4(b) of 4(a). Note that in 4(b), the minima of 4(b) have been spread and are now separated from each other by a "thin line"; nevertheless, their number and values have been preserved. Figure 4(c) shows a W-thinning of 4(a) which is not a topological watershed of 4(a) (there are still some W-destructible points).

Let us emphasize the essential difference between this notion of watershed and the notion of homotopic grayscale skeleton, pioneered by Goetcherian [11] and extensively studied in [2, 10] for the case of 2D digital images. With the topological watershed, only the connected components of the lower cross-sections of the function are preserved, while the homotopic grayscale skeleton preserves both these components and the components of the upper cross-sections. As a consequence, an homotopic grayscale skeleton may be computed by using a purely local criterion for testing whether a point may be lowered or not, while computing a topological watershed requires the use of a global data structure (see Section 5).

Figure 4(d) shows an homotopic grayscale skeleton of 4(a). Notice the difference with 4(b) in the center of the image, a "skeleton branch" at level 11 which does not separate different minima, and also the two peaks (level 15) which have been preserved. In applications where the goal is to find closed contours around the regions of interest, the notion of watershed is a better choice.

Let us quote some definitions and a property of [1] which will be used in the sequel of this article.

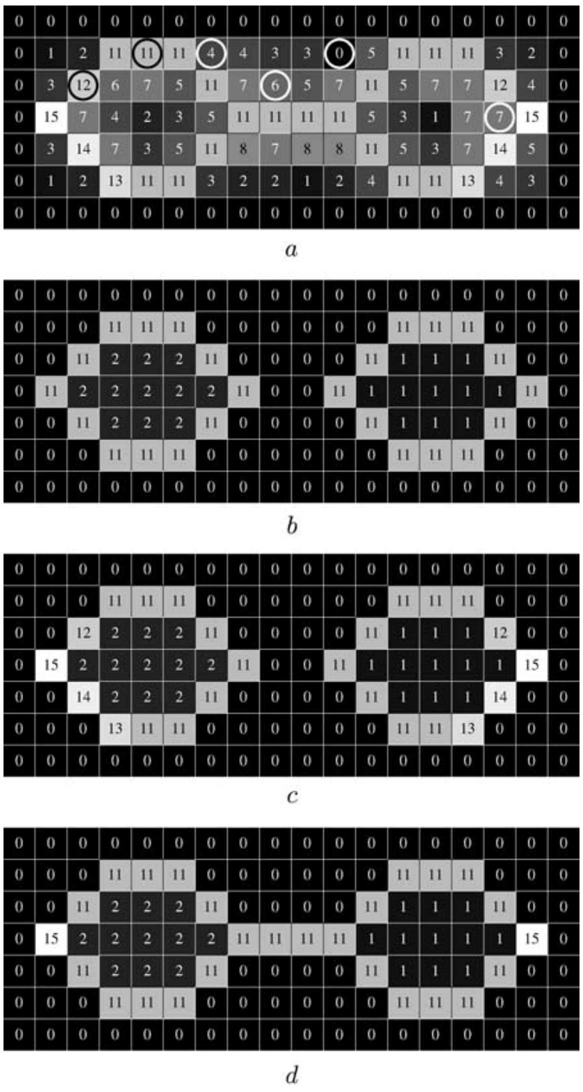

Figure 4. a: original image; b: a topological watershed of a; c: a W-thinning of a which is also an M-watershed of a (see Section 6); d: an homotopic grayscale skeleton of a. In a, we have circled six points which have different types (see Section 3). From left to right: $\tilde{S}$-point (12), S-point (11), $\tilde{M}$-point (4), $\tilde{P}$-point (6), M-point (0), P-point (7).

Definition 6. Let $F \in \mathcal{F}$, let p and q be two points of E, and let $k \in \mathbb{K}^+$.

- p and q are *k-linked* (*in $\bar{F}$*) if p and q are linked for $\bar{F}[k]$.
- *p dominates q* (*in $\bar{F}$*) if q belongs to the component of p in $\bar{F}$.

  We say that p and q are *linked* (*in $\bar{F}$*) if p dominates q in $\bar{F}$ or q dominates p in $\bar{F}$.

  We define the *connection value between p and q* (*for $\bar{F}$*) by:

$$\bar{F}(p,q) = \min\{\text{k};\ p \text{ and } q \text{ are } k\text{-linked in } \bar{F}\}.$$

- p and q are *separated* (*in $\bar{F}$*) if p and q are not linked in $\bar{F}$.
- p and q are *k-separated* (*in $\bar{F}$*) if p and q are separated in $\bar{F}$ and if the connection value for $\bar{F}$ between p and q is precisely k, i.e., if $\bar{F}(p,q) = k$.

The equivalence between this definition of k-separated points and another definition based on paths, stated informally in the introduction, can be easily shown (see [1]). Figure 3 gives some illustrations: the points x and r are linked (x dominates r), and the points r and s are 2-separated, as it can easily be checked using the following property.

Property 1 ([1]). *Let $F \in \mathcal{F}$. Two points p and q are k-separated in $\bar{F}$, if and only if:*

(i) *p and q belong to the same component of $\bar{F}[k]$, and*
(ii) *p and q belong to distinct components of $\bar{F}[k-1]$.*

The next property allows us to characterize a W-destructible point p by considering only the connection values between the lower neighbors of p. It will be used to establish our main characterization theorem (Theorem 9).

Property 2. *Let $F \in \mathcal{F}$, let $p \in E$. The point p is W-destructible for F if and only if $\Gamma^-(p) \neq \emptyset$ and, for all q and r in $\Gamma^-(p)$ with $q \neq r$, we have $\bar{F}(q,r) \leq F(p)$.*

3. Classification of Points and Transitions

As pointed out in the introduction, the time complexity of a naive topological watershed algorithm is $O(n^2 \times g)$, where n denotes the number of points and $g = k_{\max} - k_{\min}$. In order to design a quasi-linear W-thinning algorithm, we need to consider what may happen when we lower the value of a point. The examples of Fig. 4(a) may help the reader to understand the following definitions.

Let us consider a point $p \in E$ which is not W-destructible for $F \in \mathcal{F}$. Several cases may be distinguished. From Definition 4, such a point is either an inner point or a separating point for F. Furthermore, if p is an inner point, then either p belongs to a minimum of F or not.

On the other hand, if p is W-destructible for F, then p is not W-destructible for $[F \setminus p \downarrow v]$ where v is the lowest value of p. Again, we can distinguish the same possibilities for the status of p with respect to $[F \setminus p \downarrow v]$. The following definition formalizes these observations (S stands for separating, I for Inner, M for minimum and P for plateau).

Definition 7. Let $F \in \mathcal{F}$, let $p \in E$, p not W-destructible for F.

- p is an *S-point* (*for F*) if p is separating for F.
- p is an *I-point* (*for F*) if p is an inner point for F.
- p is an *M-point* (*for F*) if p belongs to a minimum of F.
- p is a *P-point* (*for F*) if p is an inner point for F which does not belong to a minimum of F.

Let q be a point which is W-destructible for F, and let v be its lowest value. We say that q is an *$\tilde{S}$-point* (*for F*) (resp. an *$\tilde{I}$-point*, an *$\tilde{M}$-point*, a *$\tilde{P}$-point*) if q is an S-point for $[F \setminus q \downarrow v]$ (resp. an *I-point*, an *M-point*, a *P-point*). If p is a T-point, with $T \in \{S, M, P, \tilde{S}, \tilde{M}, \tilde{P}\}$, we say that T is the *type of p* (*for F*).

Notice that all M-points and all P-points are I-points, and that all $\tilde{M}$-points and all $\tilde{P}$-points are $\tilde{I}$-points. Notice also that any point in E has a unique type, i.e., it is either an S-point, an M-point, a P-point, an $\tilde{S}$-point, an $\tilde{M}$-point, or a $\tilde{P}$-point. In Fig. 4(a), we have circled six points which are representative of each type.

The two following properties characterize respectively $\tilde{I}$-points and $\tilde{S}$-points. They are fundamental to understand and to prove the characterization of destructible points proposed in Section 5.

Property 3. *Let $F \in \mathcal{F}$, let $p \in E$, let $v \in \mathbb{K}$. The point p is an $\tilde{I}$-point for F with lowest value v if and only if:*

(i) *$\Gamma^-(p) \neq \emptyset$; and*
(ii) *any two points q, r in $\Gamma^-(p)$ are linked in $\bar{F}$; and*
(iii) *$v = min\{F(q); q \in \Gamma^-(p)\}$.*

Property 4. *Let $F \in \mathcal{F}$, let $p \in E$, let $v \in \mathbb{K}$. The point p is an $\tilde{S}$-point for F with lowest value v if and only if:*

(i) *$\Gamma^-(p) \neq \emptyset$; and*
(ii) *$\forall q, r \in \Gamma^-(p)$, if q and r are k-separated in $\bar{F}$ then $k \leq v + 1$; and*
(iii) *there exist two points q, r in $\Gamma^-(p)$ which are $(v+1)$-separated in $\bar{F}$.*

The type of a point p depends on the connected components of the sections of $\bar{F}$ which are adjacent to p, and we know that lowering a W-destructible point preserves the connectivity of all these sections. It may thus be seen that, during a W-thinning process, the type of a point p can only be changed by the modification of either the point p itself or a neighbor of p (this will

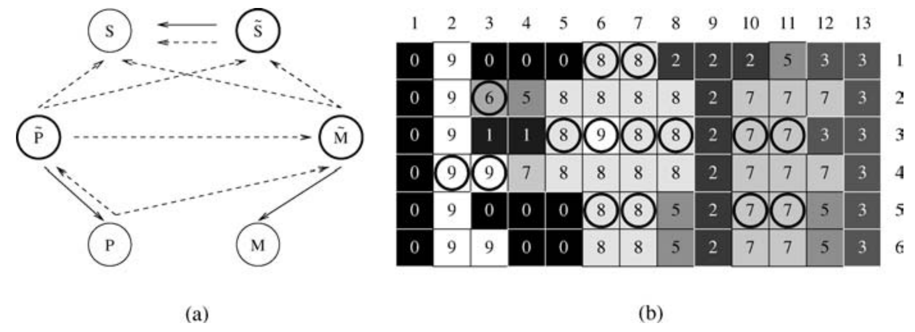

Figure 5. (a) The eleven transition types that may occur in a W-thinning process. (b) Illustration of each possible transition type. Point (3, 2) (value 6) down to 5: $[\tilde{S} \to S]$ for itself. Point (6, 5) (value 8) down to 0: $[\tilde{M} \to M]$ for itself. Point (6, 3) (value 9) down to 8: $[\tilde{P} \to P]$ for itself. Point (2, 4) (value 9) down to 0: $[\tilde{S} \to S]$ for (3, 4). Point (10, 3) (value 7) down to 2: $[\tilde{M} \to \tilde{S}]$ for (11, 3). Point (6, 1) (value 8) down to 0: $[\tilde{M} \to S]$ for (7, 1). Point (5, 3) (value 8) down to 1: $[\tilde{P} \to \tilde{M}]$ for (6, 3). Point (10, 5) (value 7) down to 2: $[\tilde{P} \to \tilde{S}]$ for (11, 5). Point (6, 5) (value 8) down to 0: $[\tilde{P} \to S]$ for (7, 5). Point (8, 3) (value 8) down to 7: $[P \to \tilde{P}]$ for (7, 3). Point (8, 3) (value 8) down to 2: $[P \to \tilde{M}]$ for (7, 3).

be proved with the following theorem). By a systematic examination of all the possibilities, we deduce that only certain transitions are possible for the type of a point p during a W-thinning process (all of them are illustrated in Fig. 5).

Theorem 5. *Let $F \in \mathcal{F}$, let $p \in E$. During a W-thinning process, the possible transitions for the type of the point p are exactly those depicted in the graph of Fig. 5(a), where the solid lines correspond to transitions due to the lowering of the point p itself, and the dashed lines correspond to transitions due to the lowering of a neighbor of p.*

As a corollary of this theorem, we immediately deduce that a point p which is an S-point (resp. an M-point) for F, is also an S-point (resp. an M-point) for any W-thinning of F.

4. Component Tree

Let us present the data structure called component tree, that will allow us to characterize W-destructible points, as well as the other types of points, locally and efficiently (Section 5). We shall see in this section that there is a strong relation between the component tree and the notion of k-separation; this relation will be used to prove the point type characterization (Theorem 9).

Let $F \in \mathcal{F}$, let $\mathcal{C}(\bar{F})$ denote the set of all couples $[k, c]$ where c is a k-component of $\bar{F}$, for all values of k between $k_{\min}$ and $k_{\max}$. We call *altitude of* $[k, c]$ the number k. By abuse of terminology, we will also call *component* an element of $\mathcal{C}(\bar{F})$.

We see easily that these components can be organized in a tree structure, that we call component tree. This structure has been introduced in the domain of data analysis [14, 35], and appears to be a fundamental tool to represent some "meaningful" information contained in a numerical function [12, 13]. Several authors, such as Vachier [33], Breen and Jones [7, 16], Salembier et al. [30], Meijster and Wilkinson [21] have used this structure in order to implement efficiently some morphological operators (e.g., connected operators, granulometries, extinction functions). The component tree has also been used as a basis for image matching algorithms [18, 20]. Algorithms to compute the component tree for the case of digital images can be found in [7, 30, 19]; the last reference also contains a discussion about time complexity of the different algorithms. Until recently, the fastest algorithm to compute the component tree was proved to run in $O(n \times \ln(n))$ complexity, where n is the number of image points. Najman and Couprie have proposed a quasi-linear algorithm [24]. For the sake of completeness, we present this algorithm in Annex 2. Let us now give a formal definition of the component tree and related notions.

Definition 8. Let $F \in \mathcal{F}$, let $[k, c]$, $[k_1, c_1]$, $[k_2, c_2]$ be elements of $\mathcal{C}(\bar{F})$.

We say that $[k_1, c_1]$ is the *parent of* $[k_2, c_2]$ if $k_1 = k_2 + 1$ and $c_2 \subseteq c_1$, in this case we also say that $[k_2, c_2]$ is a *child of* $[k_1, c_1]$.

With this relation "parent", $\mathcal{C}(\bar{F})$ forms a directed tree that we call the *component tree of* $\bar{F}$,

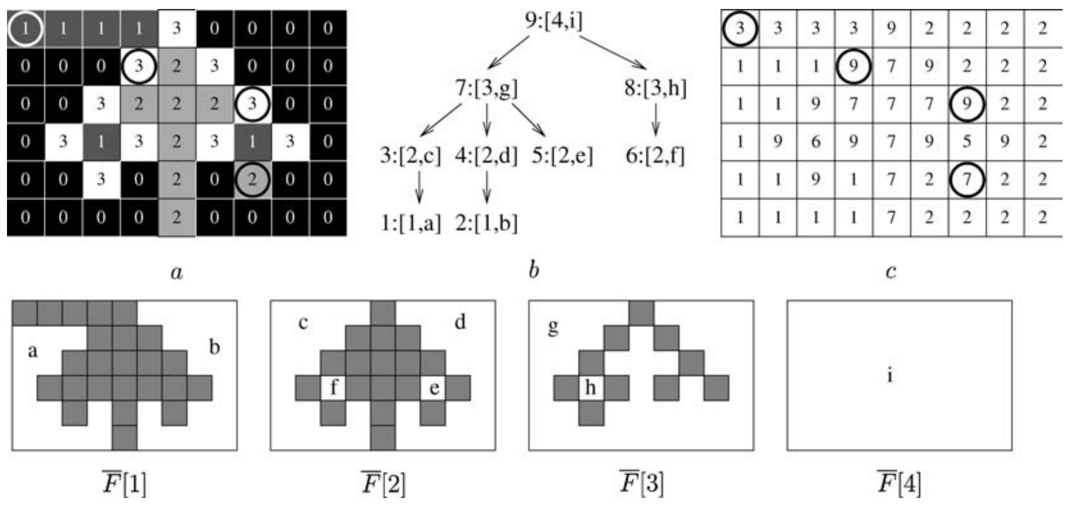

Figure 6. Illustration of Theorem 9. *a*: original image F; *b*: component tree $\mathcal{C}(\bar{F})$; *c*: component mapping Ψ; bottom row: sections $\bar{F}[1]$, $\bar{F}[2]$, $\bar{F}[3]$, $\bar{F}[4]$ (in white, with their components labelled by letters).

and that we will also denote by $\mathcal{C}(\bar{F})$ by abuse of terminology.

An element of $\mathcal{C}(\bar{F})$ which has no child is called a *leaf*, and an element of $\mathcal{C}(\bar{F})$ which has at least two childs is called a *fork*.

Figure 6(b) shows the component tree associated to the function depicted in Fig. 6(a).

Definition 9. We say that $[k_1, c_1]$ is an *ancestor of* $[k_2, c_2]$ if $k_2 \leq k_1$ and $c_2 \subseteq c_1$. In this case, we also say that $[k_1, c_1]$ is *over* $[k_2, c_2]$, and that $[k_2, c_2]$ is *under* $[k_1, c_1]$.

- the component $[k, c]$ is a *common ancestor of* $[k_1, c_1]$, $[k_2, c_2]$ if $[k, c]$ is an ancestor of both $[k_1, c_1]$ and $[k_2, c_2]$.
- the component $[k, c]$ is the *least common ancestor of* $[k_1, c_1]$, $[k_2, c_2]$, and we write $[k, c] = \text{LCA}([k_1, c_1], [k_2, c_2])$, if $[k, c]$ is a common ancestor of $[k_1, c_1]$, $[k_2, c_2]$, and if there is no other common ancestor of $[k_1, c_1]$, $[k_2, c_2]$ under $[k, c]$.
- the component $[k, c]$ is the *proper least common ancestor of* $[k_1, c_1]$ and $[k_2, c_2]$ if $[k, c]$ is the least common ancestor of $[k_1, c_1]$, $[k_2, c_2]$, and if $[k, c]$ is different from $[k_1, c_1]$ and from $[k_2, c_2]$.

For example, in Fig. 6, the fork $[3, g]$ is the proper least common ancestor of the leafs $[1, a]$ and $[2, e]$, and the components $[1, b]$ and $[2, d]$ have no proper least common ancestor.

Definition 10. We say that the components $[k_1, c_1]$, $[k_2, c_2]$ are *separated* if they have a proper least common ancestor, otherwise we say that they are *linked*.

Let M be a set $\{[k_1, c_1], [k_2, c_2], \ldots, [k_n, c_n]\}$ of elements of $\mathcal{C}(\bar{F})$. We say that the component $[k, c]$ is the *highest fork for* M if the two following conditions are satisfied:

(i) for any pair $[k_i, c_i]$, $[k_j, c_j]$ of distinct elements of M, if $[k_i, c_i]$, $[k_j, c_j]$ are separated then the altitude of $\text{LCA}([k_i, c_i], [k_j, c_j])$ is less or equal to k; and
(ii) there exists a pair $[k_i, c_i]$, $[k_j, c_j]$ of separated elements of M such that $[k, c]$ is the proper least common ancestor of $[k_i, c_i]$, $[k_j, c_j]$.

For example, in Fig. 6, the set $\{[1, a], [3, g], [4, i]\}$ has no highest fork, and the component $[3, g]$ is the highest fork of the set $\{[1, a], [3, g], [2, e], [4, i]\}$.

We make the following observations:

(a) Any two components always have a unique least common ancestor. In particular, if $[k_1, c_1]$ is over $[k_2, c_2]$, then $\text{LCA}([k_1, c_1], [k_2, c_2]) = [k_1, c_1]$. On the other hand, two components which are linked have no proper least common ancestor.
(b) Two components are separated if and only if they are disjoint; and two components $[k_1, c_1]$, $[k_2, c_2]$ are linked if and only if either $c_1 \subseteq c_2$ or $c_2 \subseteq c_1$ (see Annex 1, Lemma 6.1).

(c) A set of components may have no highest fork, and if the highest fork exists, it is indeed a fork, i.e., an element with at least two childs (otherwise it could not be a proper least common ancestor).
(d) If a set of components has a highest fork, then this highest fork is unique.

The following property makes a strong link between the component tree and the notion of separation, and justifies the common vocabulary used for both notions. It follows straightforwardly from Property 1 and from b) above.

Property 6. *Let $F \in \mathcal{F}$, let $p, q \in E$, let $k = \bar{F}(p)$, $l = \bar{F}(q)$, let $h \in \mathbb{K}$.*

(i) *The points p, q are h-separated in $\bar{F}$ if and only if $[k, C(p)]$ and $[l, C(q)]$ are separated and their proper least common ancestor in $\mathcal{C}(\bar{F})$ is a component of altitude h.*
(ii) *The points p, q are linked in $\bar{F}$ if and only if the components $[k, C(p)]$ and $[l, C(q)]$ are linked.*

The following property and theorem are from [1]. They show, in particular, that the component tree structure is preserved by any W-thinning.

Let X, Y be non-empty subsets of E such that $X \subseteq Y$. We say that Y is an *extension of* X if each connected component of Y contains exactly one connected component of X. We also say that Y is an extension of X if X and Y are both empty.

We denote by $\mathcal{C}(X)$ the set composed of all connected components of X. If Y is an extension of X, the *extension map relative to* (X, Y) is the bijection σ from $\mathcal{C}(X)$ to $\mathcal{C}(Y)$ such that, for any $C \in \mathcal{C}(X)$, $\sigma(C)$ is the connected component of Y which contains C.

Let F, G be two stacks. We say that G is an *extension* of F if, for any $k \in \mathbb{K}^+$, $G[k]$ is an extension of $F[k]$, and we denote by σ_k the extension map relative to $(F[k], G[k])$.

Property 7 ([1]). *Let F be a stack, let G be an extension of F. Let $k, h \in \mathbb{K}^+$. If $X \in \mathcal{C}(F[k])$ and $Y \in \mathcal{C}(F[h])$ then $Y \subseteq X$ if and only if $\sigma_h(Y) \subseteq \sigma_k(X)$.*

Theorem 8 ([1]). *Let F and G be two elements of $\mathcal{F}$ such that $G \leq F$. The function G is a W-thinning of F if and only if $\bar{G}$ is an extension of $\bar{F}$.*

5. Characterization of W-Destructible Points

We saw in Section 1 that checking whether a point is W-simple cannot be done locally (i.e., based on the mere knowledge of the status of the point and its neighbors), thus 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. As discussed in the introduction, with a naive approach a connected component search (at least in $O(n)$, with $n = |E|$) is necessary for each tested point, thus the complexity of a naive topological watershed algorithm has a term in n^2; furthermore, a point may be lowered several times until it is no more W-destructible. The following theorem and algorithms make it 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 (its lowest value), thanks to the component tree which may be built in quasi-linear time. In addition, the proposed algorithm provides the type of the considered point.

Recall that a W-destructible point is necessarily an $\tilde{I}$-point or an $\tilde{S}$-point (Section 3). We can now introduce the characterization theorem, which translates straightforwardly, thanks to Property 6, the Properties 3 and 4 in terms of relations between elements of the component tree.

Theorem 9. *Let $F \in \mathcal{F}$, let $p \in E$.*

We denote by $V(p)$ the set $\{[\bar{F}(q), C(q)], q \in \Gamma^-(p)\}$. Then:

(i) *The point p is an $\tilde{I}$-point for F if and only if $V(p) \neq \emptyset$ and $V(p)$ has no highest fork in $\mathcal{C}(\bar{F})$; in this case the lowest value of p is $w - 1$, where w denotes the altitude of the lowest element of $V(p)$.*
(ii) *The point p is an $\tilde{S}$-point for F if and only if $V(p) \neq \emptyset$ and $V(p)$ has a highest fork in $\mathcal{C}(\bar{F})$, the altitude of which is $v \leq F(p)$; in this case the lowest value of p is $v - 1$.*

Let $F \in \mathcal{F}$, we define the *component mapping* Ψ which associates, to each point p, a pointer $\Psi(p)$ to the element $[\bar{F}(p), C(p)]$ of the component tree $\mathcal{C}(\bar{F})$.

In Fig. 6, we illustrate the characterization of W-destructible points using Theorem 9. The function F (grayscale image) is depicted in (a), and four sections of $\bar{F}$ are shown in the bottom row. Each component of these sections is identified by a letter. The component tree $\mathcal{C}(\bar{F})$ is shown in (b), and the component mapping Ψ in (c). From top to bottom, let us consider the four circled points p_1, p_2, p_3, p_4. Thanks to the component mapping Ψ, we can build the sets $V(p_1) = \{[1, a]\}$ (no highest fork), $V(p_2) = \{[1, a], [2, c], [3, g]\}$ (no highest fork), $V(p_3) = \{[1, b], [2, e], [3, g]\}$ (highest fork $= [3, g]$), and $V(p_4) = \{[1, b], [2, e]\}$ (highest fork $= [3, g]$). From Theorem 9 we conclude that:

- p_1 and p_2 are $\tilde{I}$-points (thus they are W-destructible) and may be lowered down to 0 (they are $\tilde{M}$-points),
- p_3 is an $\tilde{S}$-point (thus p_3 is W-destructible) with lowest value 2,
- p_4 is not W-destructible (p_4 is an S-point).

The problem of finding the lowest common ancestor of two nodes in a directed tree has been well studied, and efficient algorithms exist: Harel and Tarjan [15] showed that it is possible to build in linear time a representation of a tree, which allows to find the lowest common ancestor of any two nodes in constant time. An algorithm allowing a practical implementation is provided in [5]. We denote by **BLCA** (for Binary LCA) the procedure which implements this algorithm, and which takes as arguments a tree (represented in a convenient manner) and two nodes.

We remark that using Theorem 9 to check whether a point is W-destructible, involves the computation of the highest fork of the elements of the set $V(p)$, and this may require a number of calls to **BLCA** which is quadratic with respect to the cardinality of $V(p)$: every pair of elements of $V(p)$ has to be considered. In fact, we can have a linear complexity with the following algorithm and property.

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

Function HighestFork (**Input** $\mathcal{C}$ a component tree, V a set of components of $\mathcal{C}$)

```
01. $[k_1, c_1] \leftarrow \min(V)$; let $[k_2, c_2] \ldots [k_m, c_m]$
        be the other elements of $V$
02. $k_m \leftarrow k_1$; $c_m \leftarrow c_1$
03. For $i$ From 2 To $n$ Do
04.     $[k, c] \leftarrow$ BLCA($\mathcal{C}$, $[k_i, c_i]$, $[k_m, c_m]$)
05.     If $[k, c] \neq [k_i, c_i]$ Then $k_m \leftarrow k_i$; $c_m$
            $\leftarrow c_i$
06. If $k_m = k_1$ Then Return $[\infty, \emptyset]$ Else Return
        $[k_m, c_m]$
```

Property 10. *Let $\mathcal{C}$ be a component tree, and let V be a non-empty set of components of $\mathcal{C}$.*

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

Based on Theorem 9, we propose the following algorithm for testing the type of a point. In addition, if the point is W-destructible then this algorithm also returns the lowest component to which the point can be added, otherwise the value $[\infty, \emptyset]$ is returned. Notice that, if this component has the finite altitude k, then the lowest value for the point p is $k - 1$.

Function TestType (**Input** F, p, $\mathcal{C}(\bar{F})$, Ψ)

```
01. $V \leftarrow$ set of elements of $\mathcal{C}(\bar{F})$ pointed by $\Psi(q)$
        for all $q$ in $\Gamma^-(p)$
02. If $V = \emptyset$ Then
03.     If $[F(p) + 1, C(p)]$ is a leaf of $\mathcal{C}(\bar{F})$ Then
04.         Return ($M$, $[\infty, \emptyset]$)
05.     Else
06.         Return ($P$, $[\infty, \emptyset]$)
07. Else
08.     $[k_m, c_m] \leftarrow$ HighestFork($\mathcal{C}(\bar{F})$,$V$)
09.     If $[k_m, c_m] = [\infty, \emptyset]$ Then
10.         If $\min(V)$ is a leaf of $\mathcal{C}(\bar{F})$ Then
11.             Return ($\tilde{M}$, $\min(V)$)
12.         Else
13.             Return ($\tilde{P}$, $\min(V)$)
14.     Else
15.         If $k_m \leq F(p)$ Then
16.             Return ($\tilde{S}$, $[k_m, c_m]$)
17.         Else
18.             Return ($S$, $[\infty, \emptyset]$)
```

If we only want to test a particular type, then the previous procedure may be simplified. We give below specialized functions for detecting W-destructible and $\tilde{M}$-points respectively, which will be used in the next sections.

Function W-Destructible (**Input** F, p, $\mathcal{C}(\bar{F})$, Ψ)

```
01. $V \leftarrow$ set of elements of $\mathcal{C}(\bar{F})$ pointed by $\Psi(q)$
        for all $q$ in $\Gamma^-(p)$
02. If $V = \emptyset$ Then Return $[\infty, \emptyset]$
03. $[k_m, c_m] \leftarrow$ HighestFork($\mathcal{C}(\bar{F})$,$V$)
04. If $[k_m, c_m] = [\infty, \emptyset]$ Then Return $\min(V)$
```

```
05.  If $k_m \leq F(p)$ Then Return $[k_m, c_m]$ Else
         Return $[\infty, \emptyset]$
```

Function M**-destructible** (**Input** F, p, $\mathcal{C}(\bar{F})$, Ψ)

```
01.  $V \leftarrow$ set of elements of $\mathcal{C}(\bar{F})$ pointed by $\Psi(q)$
         for all $q$ in $\Gamma^-(p)$
02.  If $V = \emptyset$ Then Return $[\infty, \emptyset]$
03.  If min($V$) is not a leaf of $\mathcal{C}(\bar{F})$ Then Return
         $[\infty, \emptyset]$
04.  $[k_m, c_m] \leftarrow$ HighestFork($\mathcal{C}(\bar{F})$, $V$)
05.  If $[k_m, c_m] = [\infty, \emptyset]$ Then Return min($V$)
         Else Return $[\infty, \emptyset]$
```

From the previous properties and observations, we deduce straightforwardly:

Property 11. *Algorithms* **TestType**, W**-Destructible** *and* M**-destructible** *give correct results with regard to the definition of the different types of points* (*Defs.* 2 *and* 3), *and are linear in time complexity with respect to the number of neighbors of* p.

Notice that, if Γ is a regular grid with a small connectivity degree (such as the graphs of the 4-adjacency or the 8-adjacency on $\mathbb{Z}^2$), then we can regard this complexity as constant. Notice also that even a naive implementation of the LCA operator leads to acceptable performance in practice, since the depth of the tree is usually quite limited. Furthermore, we can remark that the components of the tree which have exactly one child are not useful to characterize the type of a point, since they cannot be lowest common ancestors. It is thus possible to remove all these components from the tree, and update the component mapping accordingly, before using it for point type characterization.

6. M-Thinning and Binary Watershed Algorithm

The outline of a topological watershed algorithm is the following:

```
Repeat Until Stability
    Select a $W$-destructible point $p$, using a certain
        criterion
    Lower the value of $p$
```

It can be seen that, even if a W-destructible point is lowered down to its lowest value, it may again become W-destructible in further steps of the W-thinning process, due to the lowering of some of its neighbors. For example, the point at level 6 circled in white in Fig. 4(a) is W-destructible with lowest value 3. If we lower this point down to 3, we will have to lower it again, after the lowering of its neighbor at level 3 down to 0.

In order to ensure a linear complexity, we must avoid multiple selections of the same point during the execution of the algorithm. The properties of this section and the following one provide selection criteria which guarantee that a point lowered once will never be W-destructible again during the W-thinning process.

The first criterion concerns points which may be lowered by W-thinning down to the value of a neighbor which belongs to a minimum. Such a point is an $\tilde{M}$-point, and such an action is called an *M-lowering*. The aim of Theorem 12 is to show that, if $\tilde{M}$-points are sequentially selected and M-lowered, and if we continue this process until stability, giving a result G, then no W-thinning of G will contain any $\tilde{M}$-point. Since, obviously, a point which has been M-lowered will never be considered again in a W-thinning algorithm, we will obtain a M-thinning algorithm which considers each point at most once, and produces a result in which the minima cannot be extended by further W-thinning.

Definition 11. Let $F, G \in \mathcal{F}$, we say that G is an *M-thinning of F* if $G = F$ or if G can be obtained from F by sequentially M-lowering some $\tilde{M}$-points. We say that G is an *M-watershed of F* if G is a M-thinning of F and has no $\tilde{M}$-point.

Theorem 12. *Let $F \in \mathcal{F}$, let G be an M-watershed of F. Any W-thinning of G has exactly the same minima as G.*

A corollary of this theorem is that the set of points which do not belong to any minimum of an M-watershed of F is always a W-crest of F. Thus, we can compute a W-crest by only lowering $\tilde{M}$-points. In Fig. 4(c), we see an M-watershed of 4(a).

In the following algorithm, we introduce a priority function μ which is used to select the next $\tilde{M}$-point. The priority function μ associates to each point p a positive integer $\mu(p)$, called the priority of p. This function is used for the management of a priority queue, a data structure which allows to perform efficiently, on a set of points, an arbitrary sequence of the two following operations (L denotes a priority queue and p a point):

AddPriorityQueue(L, p, $\mu(p)$): store the point p with the priority $\mu(p)$ into the queue L;

ExtractPriorityQueue(L): remove and return a point which has the minimal priority value among those stored in L (if several points fulfill this condition, an arbitrary choice is made).

The choice and the interest of the priority function will be discussed afterwards, but notice that whatever the chosen priority function (for example a constant function), the result will always be an M-watershed of the input.

Procedure M**-watershed** (**Input** F, $\mathcal{C}(\bar{F})$, Ψ, μ; **Output** F)

```
01. $L \leftarrow$ EmptyPriorityQueue
02. For All $p \in E$ Do
03.     If $M$-destructible($F$, $p$, $\mathcal{C}(\bar{F})$, $\Psi$) $\neq [\infty, \emptyset]$ Then
04.         AddPriorityQueue($L$, $p$, $\mu(p)$); mark $p$
05. While $L \neq$ EmptyPriorityQueue Do
06.     $p \leftarrow$ ExtractPriorityQueue($L$); unmark $p$
07.     $[i, c] \leftarrow$ $M$-destructible($F$, $p$, $\mathcal{C}(\bar{F})$, $\Psi$)
08.     If $[i, c] \neq [\infty, \emptyset]$ Then
09.         $F(p) \leftarrow i - 1$; $\Psi(p) \leftarrow$ pointer to $[i, c]$
10.         For All $q \in \Gamma(p)$, $q \neq p$, $q$ not marked Do
11.             If $M$-destructible($F$, $q$, $\mathcal{C}(\bar{F})$, $\Psi$) $\neq [\infty, \emptyset]$ Then
12.                 AddPriorityQueue($L$, $q$, $\mu(q)$); mark $q$
```

The following property is a direct consequence of Property 7, Theorem 8, Theorem 9, Property 8 and of the fact that, obviously, each point is selected at most once by this algorithm.

Property 13. *Whatever the chosen priority function, the output of Procedure* **M-watershed** *is an M-watershed of the input.*

The time complexity of Procedure M**-watershed** *is in $O(n + m) + k$, where k is the overall complexity for the management of the priority queue.*

This watershed algorithm is the first one which is proved to guarantee a correct placement of the divide set with respect to contrast preservation (see [23, 25] for a comparison with some classical watershed algorithms). More precisely, from the previous property and the strong separation theorem of [1] (see Introduction), we immediately deduce that the result of Procedure **M-watershed** is always a strong separation of the input.

We introduced the priority function and the priority queue in order to take into account some geometrical criteria. For example, with a constant priority function, plateaux or even domes located between basins may be thinned in different ways, depending on the arbitrary choices that are allowed by the calls to **ExtractPriorityQueue** with this particular priority function (line 06). In order to "guide" the watershed set towards the highest locations of the domes and the "center" of the plateaux, we choose a lexicographic priority function μ described below.

Let $F \in \mathcal{F}$, let d be a distance on E, let $p \in E$. We denote by $D(p)$ be the minimal distance between p and any point q strictly lower than p, that is, $D(p) = \min\{d(p, q); F(q) < F(p)\}$.

It is easy to build a function μ such that, for any p, q in E:

- if $F(p) < F(q)$ then $\mu(p) > \mu(q)$;
- if $F(p) = F(q)$ and $D(p) \leq D(q)$ then $\mu(p) \geq \mu(q)$.

The efficient management of priority queues is the subject of many articles. Recently, a priority queue algorithm has been proposed by Thorup [32], which allows an operation of insertion, extraction of the minimal element or deletion to be performed in $O(\log \log m)$, where m is the number of elements stored in the structure. This cost can be regarded as constant for practical applications. Furthermore, 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 a small constant, specific linear algorithms can be used. An example of such a linear strategy is given in the next section, with algorithm **TopologicalWatershed**.

7. Watershed Algorithm

After iteratively lowering $\tilde{M}$-points until stability, we have to process the other W-destructible points in order to get a watershed. Let $F \in \mathcal{F}$, let us call an *MS-watershed of* F a function obtained from F by iteratively lowering $\tilde{M}$-points and $\tilde{S}$-points until stability. We could think that all $\tilde{P}$-points will be eventually changed to $\tilde{M}$-points and then M-lowered in such a process, as it is the case for images like Fig. 4(a). But the examples of Fig. 7 show that it is not always

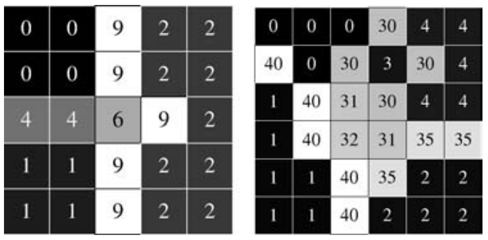

Figure 7. Examples of W-destructible points in an MS-watershed which are neither $\tilde{M}$-points nor $\tilde{S}$-points: the point at 6 in the image on the left, the points at 31 and 32 in the image on the right.

the case, in other words, an MS-watershed of F is not always a topological watershed of F. Furthermore, there may exist thick regions made of $\tilde{P}$-points in an MS-watershed, and although $\tilde{M}$-points and $\tilde{S}$-points may be lowered directly down to their lowest possible value, we have no such guarantee for the $\tilde{P}$-points (see Theorem 5).

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. The idea is to give the greatest priority to a W-destructible point which may be lowered down to the lowest possible value. We prove that an algorithm which uses this strategy never selects the same point twice. A priority queue could be used, as in the previous section, to select W-destructible points in the appropriate order. Here, we propose a specific linear watershed algorithm which may be used when the grayscale range is small.

Procedure TopologicalWatershed (Input $F, \mathcal{C}(\bar{F}), \Psi$; **Output** F)

```
01.  For $k$ From $k_{\min}$ To $k_{\max} - 1$ Do $L_k \leftarrow \emptyset$
02.  For All $p \in E$ Do
03.    $[i, c] \leftarrow$ W-Destructible($F, p, \mathcal{C}(\bar{F}), \Psi$)
04.    If $i \neq \infty$ Then
05.      $L_{i-1} \leftarrow L_{i-1} \cup \{p\}$; $K(p) \leftarrow i - 1$;
         $H(p) \leftarrow$ pointer to $[i, c]$
06.  For $k = k_{\min}$ To $k_{\max} - 1$ Do
07.    While $\exists p \in L_k$ Do
08.      $L_k = L_k \setminus \{p\}$
09.      If $K(p) = k$ Then
10.        $F(p) \leftarrow k$ ; $\Psi(p) \leftarrow H(p)$
11.        For All $q \in \Gamma(p), k < F(q)$ Do
12.          $[i, c] \leftarrow$ W-Destructible($F, q,$
             $\mathcal{C}(\bar{F}), \Psi$)
13.          If $i = \infty$ Then $K(q) \leftarrow \infty$
14.          Else If $K(q) \neq i - 1$ Then
15.            $L_{i-1} \leftarrow L_{i-1} \cup \{q\}$;
               $K(q) \leftarrow i - 1$
16.            $H(q) \leftarrow$ pointer to $[i, c]$
```

We have the following guarantees:

Property 14. *In algorithm* **TopologicalWatershed**,

(i) *at the end of the execution, F is a topological watershed of the input function*;

(ii) *let n and m denote respectively the number of vertices and the number of arcs in the graph (E, Γ). If $k_{max} - k_{min} \leq n$, then the time complexity of the algorithm is in $O(n + m)$.*

As discussed in the previous section, this algorithm provides topological guarantees but does not care about geometrical criteria. If we want to take such criteria into account, we can use first the procedure **M-watershed** with the priority function described at the end of Section 6, and then the procedure **Topological-Watershed**.

8. Conclusion

We presented quasi-linear algorithms for computing W-crests and topological watersheds, which are proved to give correct results with respect to the definitions, and to indeed achieve the claimed complexity. From the purely topological point of view, we consider as equivalent the different possible watersheds of the same function; but other constraints must be taken into account when dealing with certain applications. We provided in Section 6 a criterion which is often considered as a good choice in many practical situations. Filtering methods based on the component tree, like connected operators, can be easily integrated to the presented algorithms. It is also possible to design a variant taking a set of markers as secondary input, following a classical approach based on geodesic reconstruction. Forthcoming publications will develop these points.

Annex 1: Proofs

Proof of Proposition 2: Let $k = F(p)$. Suppose that p is W-destructible for F, thus p is adjacent to exactly one component of $\bar{F}[k]$ and $\Gamma^-(p) \neq \emptyset$. Take any two points q, r in $\Gamma^-(p)$, since they belong to the same component of $\bar{F}[k]$, we deduce that $\bar{F}(q, r) \leq k$. Conversely, suppose that $\Gamma^-(p) \neq \emptyset$ and for all q and r in $\Gamma^-(p)$, we have $\bar{F}(q, r) \leq k$. Since $\Gamma^-(p) \neq \emptyset$ there is at least one component of $\bar{F}[k]$ adjacent to p, and the other condition implies that all the points in $\Gamma^-(p)$ belong to the same component of $\bar{F}[k]$, thus p is W-destructible. □

Proof of Proposition 3: Suppose that the conditions (i), (ii) and (iii) are verified. We see (Proposition 2) that p is destructible. Let $F^{(1)} = [F \setminus p]$, if $F^{(1)}(p) > v$ we see that the conditions (i), (ii) and (iii) are still verified for $F^{(1)}$, and thus p is still destructible. We can repeat this process until the step n such that $F(p) - n = v$, and we easily deduce from (iii) that p is an inner point for $F^{(n)} = [F \setminus p \downarrow v]$. The proof of the converse property is straightforward. □

Proof of Proposition 4: The proof is essentially the same as the proof of Proposition 3. □

Lemma 5.1. *Let $F \in \mathcal{F}$, let $p \in E$, let $k = F(p)$ and let $v \in \mathbb{K}$, $v < k$. The function $[F \setminus p \downarrow v]$ is a W-thinning of F if and only if, for any h such that $v < h \leq k$, p is W-simple for $F[h]$.*

Proof: immediate from the definitions. □

Lemma 5.2. *Let $F \in \mathcal{F}$, let $p \in E$ be an S-point for F, let q be a W-destructible point and let w be an integer such that $[F \setminus q \downarrow w]$ is a W-thinning of F, let F' denote $[F \setminus q \downarrow w]$. Then, p is an S-point for F'.*

Proof: Since p is not W-destructible, we have $q \neq p$. We know that p is adjacent to (at least) two distinct components c_1 and c_2 of $\bar{F}[k]$, with $k = F(p)$. Suppose that p is adjacent to only one component of $\bar{F}'[k]$ (obviously, p is adjacent to at least one component of $\bar{F}'[k]$). This implies that $F(q) \geq k$, that $w < k$ and that q is adjacent to both c_1 and c_2, a contradiction with Lemma 5.1 since $[F \setminus q \downarrow w]$ is a W-thinning of F. □

Lemma 5.3. *Let $F \in \mathcal{F}$, let $p \in E$ be an $\tilde{S}$-point with lowest value v for F, let q be a W-destructible point and let w be an integer such that $[F \setminus q \downarrow w]$ is a W-thinning of F, let F' denote $[F \setminus q \downarrow w]$. Then, p is either an $\tilde{S}$-point for F' with lowest value v, or an $\tilde{S}$-point for F' with lowest value $w > v$, or an S-point for F'. In the two last cases, the point q is necessarily adjacent to p.*

Proof: Let $k = F(p)$. We know that p is adjacent to exactly one component of $\bar{F}[h]$, for all h such that $v < h \leq k$, and that p is adjacent to (at least) two distinct components c_1, c_2 of $\bar{F}[v]$.

- If $q = p$, we see that p remains an $\tilde{S}$-point with lowest value v for $[F \setminus p \downarrow h]$ with $h > v$, and that p becomes an S-point for $[F \setminus p \downarrow v]$.
- If q is not adjacent to p, we see that p is still adjacent to exactly one component of $\bar{F}'[h]$ for all h such that $v < h \leq k$. Suppose that q is not adjacent to p and that p is adjacent to exactly one component of $\bar{F}'[v]$ (obviously p is adjacent to at least one component of $\bar{F}'[v]$). It means that q is adjacent to both c_1 and c_2, that $F(q) \geq v$ and that $F'(q) < v$, a contradiction with Lemma 5.1 since F' is a W-thinning of F.
- Suppose now that q is adjacent to p and that $q \neq p$. If p is W-simple for all $F'[h]$ with $v < h \leq k$, then p is still an $\tilde{S}$-point for F' with lowest value v (same as above). Otherwise, p may be either an $\tilde{S}$-point for F' with lowest value w, with $v < w < k$, or an S-point for F' (see examples in Fig. 5). □

Lemma 5.4. *Let $F \in \mathcal{F}$, let $p \in E$ be an $\tilde{I}$-point for F which is adjacent to a minimum m of F. Then p is an $\tilde{M}$-point with lowest value $F(m)$.*

Proof: let $v = F(m)$, let q be a point of m adjacent to p. Let r be any point in $\Gamma^-(p)$ which is not in m (if there is no such point, the proof is done). By Property 3 we know that r and q are linked in $\bar{F}$, thus, since m is a minimum, the component of r in $\bar{F}$ must contain m, and $F(r) \geq F(m)$. Again by Property 3, we deduce that $F(m)$ is the lowest value of p, which implies that p is an $\tilde{M}$-point. □

Proof of Theorem 5: Let $k = F(p)$, let T_1 denote the type of the point p for F, let q be a W-destructible point for F, let v be an integer such that $[F \setminus q \downarrow v]$ is a W-thinning of F, let F' denote $[F \setminus q \downarrow v]$ and let T_2 denote the type of the point p for F'.

(1) Case $T_1 = M$. Since p is not W-destructible, we have $q \neq p$. If q is not adjacent to p then obviously $T_2 = T_1$. Suppose now that q is adjacent to p, and that $F'(q) = v < k$. If $F(q) = k$, then, since p is an M-point for F, we know that q is also an M-point for F, and thus q is not W-destructible for F, a contradiction. If $F(q) > k$, then consider $[F \setminus q \downarrow k]$ and apply the same argument as above. In conclusion, we have either q not adjacent to p or $F'(q) \geq k$, thus $T_2 = T_1$.

(2) Case $T_1 = P$. Since p is not W-destructible, we have $q \neq p$. If q is not adjacent to p then obviously $T_2 = T_1$. Suppose now that q is adjacent to p. If $F'(q) = v \geq k$ then $T_2 = T_1$, otherwise we see that p is W-destructible for F' with lowest value v, and that p has no strictly lower neighbor for $[F' \setminus p \downarrow v]$, thus p is either an $\tilde{M}$-point or a $\tilde{P}$-point, as shown in Fig. 5).
(3) Case $T_1 = S$. See Lemma 5.2.
(4) Case $T_1 = \tilde{S}$. See Lemma 5.3.
(5) Case $T_1 = \tilde{P}$. Let w be the lowest value of p.

If $q = p$ and $F'(q) > w$ then $T_2 = T_1$.

If $q = p$ and $F'(q) = w$ then, since p is an inner point for F' and not an $\tilde{M}$-point for F, we know that p is adjacent to exactly one component of $\bar{F}'[w+1]$ which is not a minimum, thus $T_2 = P$.

We know that p is adjacent to exactly one component of $\bar{F}[h]$, for all h such that $v < h \leq k$, and that p is not adjacent to any component of $\bar{F}[w]$.

If q is not adjacent to p we see that the same remains true for F', thus $T_2 = T_1$. Suppose now that q is adjacent to p and $q \neq p$. We see that p must be adjacent to at least one component of $\bar{F}'[k]$, thus T_2 is not an inner type (see examples of the three possibilities other than T_1 in Fig. 5).

(6) Case $T_1 = \tilde{M}$. The arguments are the same as for case 5, except that T_2 can be M instead of P, and cannot be $\tilde{P}$ (see Lemma 5.4, and observe that p remains adjacent to a minimum). □

Lemma 6.1. *Let $F \in \mathcal{F}$, let $k_1, k_2 \in \mathbb{K}^+$, and let c_1, c_2 be two components of $\bar{F}[k_1]$, $\bar{F}[k_2]$ respectively. Then c_1 and c_2 are either disjoint or tied by an inclusion relation.*

Proof: If $k_1 = k_2$ then we have either $c_1 \cap c_2 = \emptyset$ or $c_1 = c_2$ (property of connected components). Otherwise, suppose without loss of generality that $k_1 > k_2$. Let c'_2 be the component of $\bar{F}[k_1]$ which contains c_2. We have either $c_1 \cap c'_2 = \emptyset$ or $c_1 = c'_2$ (same as above). If $c_1 \cap c'_2 = \emptyset$ then we have $c_1 \cap c_2 = \emptyset$, otherwise we have $c_2 \subseteq c_1$. □

Proof of Proposition 10: If $k_m = k_1$ at line 06, then clearly $[k_1, c_1]$ is under all the other elements of V and there is no highest fork (any two elements of V are linked). Otherwise, one gets easily convinced that:

- the component $[k_m, c_m]$ found at line 06 is indeed the LCA of a given pair of separated components in V, and
- no other pair of separated components in V can have a higher LCA. □

Lemma 12.1. *Let $F, F' \in \mathcal{F}$, let $p, q \in E$ and $v, w \in \mathbb{K}$ such that:*

(i) *$[F \setminus p \downarrow v]$ is not a W-thinning of F, and*
(ii) *$F' = [F \setminus q \downarrow w]$ is a W-thinning of F, and*
(iii) *$[F' \setminus p \downarrow v]$ is a W-thinning of F'.*
Then p and q are neighbors, $F(q) \geq F(p)$, and $w \leq v$.

Proof: by Lemma 5.2, we deduce that p cannot be an S-point for F, because it could not become a W-destructible point in this case. Suppose now that p is an $\tilde{S}$-point for F with lowest value $h > v$, then by Lemma 5.3 the point p is either an S-point or an $\tilde{S}$-point for F' with a lowest value greater than h, a contradiction with iii) since $h > v$. Thus, p is either an $\tilde{I}$-point with lowest value $h > v$ or a P-point (in this case, we set $h = F(p)$). For any $k \leq h$, no component of $\bar{F}[k]$ is adjacent to p. Since $[F' \setminus p \downarrow v]$ is a W-thinning of F' we know that, for any k such that $v < k \leq h$, there is exactly one component of $\bar{F}'[k]$ adjacent to p (see Lemma 5.1). We deduce that for any such k, this component must contain q which must be a neighbor of p, and that $w \leq v$. □

Lemma 12.2. *Let $F \in \mathcal{F}$, let $p \in E$ such that:*

(i) *p is not an $\tilde{M}$-point for F, and*
(ii) *there exists a point q and a value w such that $[F \setminus q \downarrow w]$ is a W-thinning of F, and p is an $\tilde{M}$-point for $[F \setminus q \downarrow w]$.*

Then, q is an $\tilde{M}$-point for F.

Proof: immediate from Lemma 12.1. □

Proof of Theorem 12: Let G' be a W-thinning of G and suppose that G' has a minimum which is strictly larger than the corresponding minimum of G. Consider the sequence of point lowerings which leads from G to G', and let $G = F^0, F^1, \ldots, F^n = G'$ be the successive results of these operations. Let F^k be the first element in the sequence in which a point p is M-lowered. Thus $F^k = [F^{k-1} \setminus p \downarrow v]$ is the result of M-lowering the point p, in other words p is an $\tilde{M}$-point for F^{k-1}. Consider now the last F^i in the sequence $F^0 \ldots F^{k-2}$ such that p is not an $\tilde{M}$-point for F^i. If no such element exists, then we have a contradiction since there is no

$\tilde{M}$-point for $F^0 = G$. Otherwise, since p is an $\tilde{M}$-point for F^{i+1} and not for F^i, from Lemma 12.2 we deduce that the point q which has been lowered between F^i and F^{i+1} has indeed been M-lowered. This contradicts our definition of F^k. □

Proof of Prop 14:

(a) From Property 7 and Theorem 8, it follows that the initial component tree of $\bar{F}$ remains a component tree for all the modified versions of $\bar{F}$ in this algorithm. We also see that the component mapping Ψ is updated in order to keep correct pointers from the vertices of the graph to the corresponding tree elements.
(b) We see easily that ($K(p) = k$ and p in L_k) $\Leftrightarrow$ p is W-destructible for F with lowest value k.
(c) Let us prove that in lines 07–16, there is no W-destructible point for F with a lowest value $k' < k$. It is true when $k = k_{\min}$. From Lemma 12.1, we know that a point cannot receive a lowest value v unless one of its neighbors is lowered down to a value $v' \leq v$. All the lowerings are done at line 10, by the statement $F(p) \leftarrow k$. Thus, the property remains true as k increases.
(d) From (a) and (b), we deduce that at each step of the execution, F is a W-thinning of the input function.
(e) From (c), we deduce that at the end of the execution, F has no W-destructible point.

  (i) Follows from (d) and (e).
  (ii) For any given value of k, a point which is lowered at line 10 will not be lowered again in any step $k' > k$. Thus, each point is lowered at most once. Also, the total number of executions of lines 12–16 will not exceed m. Globally, the sum of the costs of all calls to the function W-**Destructible** is in $O(n+m)$. The calls to list management functions are in constant time. The total number of elements stored in the lists L_i cannot exceed $n + m$.

□

Annex 2: Quasi-Linear Algorithm for the Component Tree

Let us first describe briefly the disjoint set problem, which consists in maintaining a collection S of disjoint subsets of a set E under the operation of union. Each set X in S is represented by a unique element of X, called the *canonical element*. Three operations allow the management of the collection (in the following x and y denote two distinct elements of E):

MakeSet(x): add the set $\{x\}$ to the collection S, provided that the element x does not already belongs to a set in S. The canonical element of $\{x\}$ is x.
Find(x): return the canonical element of the set in S which contains x.
Link(x, y): let X and Y be the two sets in S whose canonical elements are x and y respectively. Both sets are removed from S, their union $Z = X \cup Y$ is added to S and a canonical element for Z is selected and returned.

Tarjan [31] has proposed a very simple and very efficient algorithm to achieve any intermixed sequence of such operations with a quasi-linear complexity. More precisely, if m denotes the number of operations and n denotes the number of elements, the worst-case complexity is in $O(m \times \alpha(m, n))$ where $\alpha(m, n)$ is a function which grows very slowly, for all practical purposes $\alpha(m, n)$ is never greater than four. The implementation of this algorithm is given below. The maps 'par' (stands for 'parent') and 'rank', which constitute a representation of the disjoint sets in the form of directed trees, are represented by global arrays in memory. For more detailed explanations and complexity analysis, see [31].

```
Procedure MakeSet (element $x$)
  par($x$) ← $x$; rank($x$) ← 0

Function Find (element $x$)
  If par($x$) $\neq x$ Then par($x$) ← Find (par($x$))
  Return par($x$)

Function Link (element $x$, $y$)
  If rank($x$) > rank($y$) Then exchange($x$, $y$)
  If (rank($x$) = rank($y$)) Then rank($y$) ←
        rank($y$) + 1
  par($x$) ← $y$
  Return $y$
```

Now let us give our algorithm to build the component tree. A more detailed explanation, together with a proof of the complexity, can be found in [24].

```
Procedure BuildComponentTree
Input : ($E$, $\Gamma$)-graph; N = number of points in $E$
Input : $F$-map from $E$ to $\mathbb{Z}$
Output : $N_n$-number of nodes (of the component
         tree) ($\leq N$)
```

Output : nodes-array $[0 \ldots N-1]$ of node
Output : Ψ-map from E to $[0 \ldots N-1]$ (component mapping)
Local : subtreeRoot-map from $[0 \ldots N-1]$ to $[0 \ldots N-1]$

```
01. Sort the points in increasing order of value
      for $F$; $N_n \leftarrow N$
02. For All $p \in E$ Do nodes[$p$] ← MakeNode($p$);
      subtreeRoot[$p$] ← $p$;
      MakeSet1($p$); MakeSet2($p$) ← $p$
03. For All $p$ of $E$ in increasing order of value for $F$
      Do
04.   curCanonicalElt ← Find1($p$)
05.   curNode ← Find2(subtreeRoot
        [curCanoni calElt])
06.   For each (already processed) neighbor $q$ of $p$
        with $F(q) \leq F(p)$ Do
07.     adjCanonicalElt ← Find1($q$)
08.     adjNode ← Find2(subtreeRoot
          [adjCanoni calElt])
09.     If curNode ≠ adjNode Then
10.       If nodes[curNode] → height = nodes
            [adjNode] → height Then
11.         tmpNode ← Link2(adjNode,curNode)
12.         If tmpNode =curNode Then
13.           Add the list of childs of
                nodes [adjNode]
14.             to the list of childs of
                  nodes [cur Node]
15.         Else
16.           Add the list of childs of
                nodes[cur Node]
17.             to the list of childs of
                  nodes[adjNode]
18.         delete nodes[adjNode];
              nodes[adjNode] ← nodes[curNode]
19.         curNode ← tmpNode; $N_n \leftarrow N_n - 1$
20.       Else
21.         nodes[curNode] → addChild
              (nodes [adjNode])
22.       curCanonicalElt ← Link1
            (adjCanonicalElt, curCanonicalElt)
23.       subtreeRoot[curCanonicalElt]
            ← curNode
24. For All $p \in E$ Do $\Psi(p)$ ← Find2($p$)
```

References

1. G. Bertrand, "On topological watersheds," *Journal of Mathematical Imaging and Vision*, Vol. 22, Nos. 2/3, pp. ??–??, 2005.
2. G. Bertrand, J.C. Everat, and M. Couprie, "Image segmentation through operators based upon topology," *Journal of Electronic Imaging*, Vol. 6, No. 4, pp. 395-405, 1997.
3. S. Beucher and Ch. Lantuéjoul, "Use of watersheds in contour detection," in *Proc. Int. Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, 1979.
4. S. Beucher and F. Meyer, "The morphological approach to segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*, Dougherty (Ed.), Chap. 12, Marcel Dekker, 1993, pp. 433–481.
5. M.A. Bender and M. Farach-Colton, "The LCA problem revisited," in *Proc. 4th Latin American Symposium on Theoretical Informatics*, LNCS, Springer, Vol. 1776, 2000, pp. 88–94.
6. U.M. Braga-Neto and J. Goutsias, "A theoretical tour of connectivity in image processing and analysis," *Journal of Mathematical Imaging and Vision*, Vol. 19, pp. 5–31, 2003.
7. E.J. Breen and R. Jones, "Attribute openings, thinnings and granulometries," *Computer Vision and Image Understanding*, Vol. 64, No. 3, pp. 377–389, 1996.
8. T. H. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1990.
9. M. Couprie and G. Bertrand, "Topological grayscale watershed transformation," in *Proc. SPIE Vision Geometry VI*, Vol. 3168, 1997, pp. 136–146.
10. M. Couprie, F.N. Bezerra, and G. Bertrand, "Topological operators for grayscale image processing," *Journal of Electronic Imaging*, Vol. 10, No. 4, pp. 1003–1015, 2001.
11. V. Goetcherian, "From binary to grey tone image processing using fuzzy logic concepts," *Pattern Recognition*, Vol. 12, No. 12, pp. 7–15, 1980.
12. P. Guillataud, "Contribution á l'analyse dendroniques des images," PhD thesis of Université de Bordeaux I, 1992.
13. P. Hanusse and P. Guillataud, "Sémantique des images par analyse dendronique," in *8th Conf. Reconnaissance des Formes et Intelligence Artificielle*, AFCET Ed., Lyon, Vol. 2, 1992, pp. 577–588.
14. J.A. Hartigan, "Statistical theory in clustering," *Journal of classification*, No. 2, pp. 63–76, 1985.
15. D. Harel and R.E. Tarjan, "Fast algorithms for finding nearest common ancestors," *SIAM J. Comput.*, Vol. 13, No. 2, pp. 338–355, 1984.
16. R. Jones, "Connected filtering and segmentation using component trees," *Computer Vision and Image Understanding*, Vol. 75, No. 3, pp. 215–228, 1999.
17. T.Y Kong and A. Rosenfeld, "Digital topology: Introduction and survey," *Computer Vision, Graphics and Image Processing*, Vol. 48, pp. 357–393, 1989.
18. J. Mattes and J. Demongeot, "Tree representation and implicit tree matching for a coarse to fine image matching algorithm," in *Proc. MICCAI, LNCS*, Springer, Vol. 1679, 1999, pp. 646–655.
19. J. Mattes and J. Demongeot, "Efficient algorithms to implement the confinement tree," in *Proc. DGCI, LNCS*, Springer, Vol. 1953, 2000, pp. 392–405.
20. J. Mattes, M. Richard, and J. Demongeot, "Tree representation for image matching and object recognition," in *Proc. DGCI, LNCS*, Springer, Vol. 1568, 1999, pp. 298–309.
21. A. Meijster and M. Wilkinson, "A comparison of algorithms for connected set openings and closings," *IEEE PAMI*, Vol. 24, pp. 484–494, 2002.

22. F. Meyer, "Un algorithme optimal de ligne de partage des eaux," in *Proc. 8th Conf. Reconnaissance des Formes et Intelligence Artificielle*, AFCET Ed., Lyon, Vol. 2, 1991, pp. 847–859.
23. L. Najman and M. Couprie, "Watershed algorithms and contrast preservation," in *Proc. DGCI, LNCS*, Springer, Vol. 2886, 2003, pp. 62–71.
24. L. Najman and M. Couprie, "Quasi-linear algorithm for the component tree," in *Proc. SPIE Vision Geometry XII*, Vol. 5300, 2004, pp. 98–107.
25. L. Najman, M. Couprie, and G. Bertrand, "Watersheds, mosaics, and the emergence paradigm," to appear in *Discrete Applied Mathematics*, 2005.
26. L. Najman and M. Schmitt, "Watershed of a continuous function," *Signal Processing*, Vol. 38, pp. 99–112, 1994.
27. J.B.T.M. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta Informaticae*, Vol. 41, pp. 187–228, 2000.
28. A. Rosenfeld, "On connectivity properties of grayscale pictures," *Pattern Recognition*, Vol. 16, pp. 47–50, 1983.
29. J. Serra, *Image Analysis and Mathematical Morphology*, Vol. II: *Theoretical Advances*, Academic Press, 1988.
30. P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *IEEE Trans. on Image Processing*, Vol. 7, No. 4, pp. 555–570, 1998.
31. R.E. Tarjan, "Disjoint sets," *Data Structures and Network Algorithms*, Chap. 2, SIAM, 1978, pp. 23–31.
32. M. Thorup, "On RAM priority queues," in *7th ACM-SIAM Symposium on Discrete Algorithms*, 1996, pp. 59–67.
33. C. Vachier, "Extraction de caractéristiques, segmentation d'images et Morphologie Mathématique," PhD Thesis, École des Mines, Paris, 1995.
34. L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 6, pp. 583–598, 1991.
35. D. Wishart, "Mode analysis: A generalization of the nearest neighbor which reduces chaining effects," in *Numerical Taxonomy*, A.J. Cole (Ed.), Academic Press, 1969, pp. 282–319.

Michel Couprie received his Ingénieur's degree from the École Supérieure d'Ingénieurs en Électrotechnique et Électronique (Paris, France) in 1985 and the Ph.D. degree from the Pierre et Marie Curie University (Paris, France) in 1988. Since 1988 he has been working in ESIEE where he is an Associate Professor. He is a member of the Laboratoire Algorithmique et Architecture des Systèmes Informatiques, ESIEE, Paris, and of the Institut Gaspard Monge, Université de Marne-la-Vallée. His current research interests include image analysis and discrete mathematics.

Laurent Najman received his Ph.D. of applied mathematics from Paris-Dauphine university and an Ingénieur's degree from the Ecole des Mines de Paris. After earning his Ingénieur's degree, he worked in the research laboratories of Thomson-CSF for three years, before joining Animation Science in 1995, as director of research and development. In 1998, he joined OcÈ Print Logic Technolgies, as senior scientist. Since 2002, he is associate professor with the A2SI laboratory of ESIEE, Paris. His current research interest is discrete mathematical morphology.

Gilles Bertrand received his Ingénieur's degree from the École Centrale des Arts et Manufactures in 1976. Until 1983 he was with the Thomson-CSF company, where he designed image processing systems for aeronautical applications. He received his Ph.D. from the École Centrale in 1986. He is currently teaching and doing research with the Laboratoire Algorithmique et Architecture des Systèmes Informatiques, ESIEE, Paris, and with the Institut Gaspard Monge, Université de Marne-la-Vallée. His research interests are image analysis, pattern recognition, mathematical morphology and digital topology.