Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle

Jean Cousty, Gilles Bertrand, Laurent Najman, Michel Couprie

Watershed cuts: minimum spanning forests and the drop of water principle

Jean Cousty[1,2], Gilles Bertrand[1], Laurent Najman[1] and Michel Couprie[1]

***Abstract*— We study the watersheds in edge-weighted graphs. We define the watershed cuts following the intuitive idea of drops of water flowing on a topographic surface. We first establish the consistency of these watersheds: they can be equivalently defined by their "catchment basins" (through a steepest descent property) or by the "dividing lines" separating these catchment basins (through the drop of water principle). Then we prove, through an equivalence theorem, their optimality in terms of minimum spanning forests. Afterward, we introduce a linear-time algorithm to compute them. To the best of our knowledge, similar properties are not verified in other frameworks and the proposed algorithm is the most efficient existing algorithm, both in theory and practice. Finally, the defined concepts are illustrated in image segmentation leading to the conclusion that the proposed approach improves, on the tested images, the quality of watershed-based segmentations.**

***Index Terms*— Watershed, minimum spanning forest, minimum spanning tree, graph, mathematical morphology, image segmentation**

INTRODUCTION

FOR topographic purposes, the watershed has been extensively studied during the 19th century by Maxwell [1] and Jordan [2] among others. One hundred years later, the watershed transform was introduced by Digabel and Lantuéjoul [3] for image segmentation and is now used as a fundamental step in many powerful segmentation procedures.

Let us consider a grayscale image as a topographic surface: the gray level of a pixel becomes the elevation of a point, the basins and valleys of the topographic surface correspond to dark areas, whereas the mountains and crest lines correspond to the light areas. Intuitively, the watershed divide is a set of points which satisfy the "drop of water principle": a separating set of points from which a drop of water can flow down towards at least two regional minima.

In order to compute the watershed of a digital image, several approaches [4]–[14] have been proposed. Many of them consider a grayscale digital image as a vertex-weighted graph. One of the most popular consists of simulating a flooding of the topographic surface from its regional minima [5], [6], [15]. The divide is made of "dams" built at those points where water coming from different minima would meet. Another approach, called topological watershed [10], [16], [17], allows the authors to rigorously define the notion of a watershed in a discrete space and to prove important properties not guaranteed by most watershed algorithms [18]. It consists of lowering the values of a map (*e.g.*, the grayscale image) while preserving some topological properties, namely, the number of connected components of each lower cross-section. In this case, the watershed divide is the set of points which are not in any regional minimum of the transformed map.

In this paper, we investigate the watersheds in a different framework: we consider a graph whose edges are weighted by a cost function (see [19]–[26] for examples of image analysis operators in this framework). We propose a new definition of watershed, called *watershed-cut*, and obtain a set of remarkable properties. Unlike previous approaches in discrete frameworks, the watersheds-cuts are defined thanks to the formalization of the intuitive "drop of water principle".

Our first contribution establishes the consistency of watershed-cuts. In particular, we prove that they can be equivalently defined by their "catchment basins" (through a steepest descent property) or by the "dividing lines" separating these catchment basins (through the drop of water principle). As far as we know, in discrete frameworks, our definition is the first one that satisfies such a property.

Our second contribution establishes the optimality of watershed-cuts. In [19], F. Meyer shows the link between minimum spanning forests (MSF) and flooding from marker algorithms. In this paper, we extend the problem of minimum spanning forests and show that there is indeed an equivalence between the watershed-cuts and the separations induced by minimum spanning forest relative to the minima.

Our third contribution consists of a linear-time algorithm to compute the watershed-cuts of an edge-weighted graph. The proposed algorithm does not require any sorting step, nor the use of any sophisticated data structure such as a hierarchical queue or a representation to maintain unions of disjoint sets. Thus, whatever the range of the edge weights, it runs in linear time with respect to the size (*i.e.*, the number of edges) of the input graph. Furthermore, this algorithm does not need to compute the minima in a preliminary step. To the best of our knowledge, this is the first watershed algorithm satisfying such properties.

Then, we illustrate that, for some situations, the proposed watershed localizes with better accuracy the contours of objects in digital images. To this end, we provide, on some examples, the results of morphological segmentation schemes based on watersheds in vertex-weighted graphs and the results of their adaptation in edge-weighted graphs.

This article[1] is self-contained and, in order to ease the reading, proofs of the properties are deferred to the Appendix.

I. BASIC NOTIONS AND NOTATIONS

This paper is settled in the framework of edge-weighted graphs. Following the notations of [28], we present some basic definitions

[1] Université Paris-Est, LABINFO-IGM, UMR CNRS 8049, A2SI-ESIEE, France

[2] INRIA Sophia Antipolis - ASCLEPIOS Team, France

Email addresses: {j.cousty, g.bertrand, l.najman, m.couprie}@esiee.fr

This work was partially supported by ANR grant SURF-NT05-2_45825

[1]This article extends and completes a previous paper published in a conference [27].

to handle such kind of graphs.

A. Graphs

We define a *graph* as a pair $X = (V(X), E(X))$ where $V(X)$ is a finite set and $E(X)$ is composed of unordered pairs of $V(X)$, *i.e.*, $E(X)$ is a subset of $\{\{x, y\} \subseteq V(X) \mid x \neq y\}$. Each element of $V(X)$ is called a *vertex or a point (of X)*, and each element of $E(X)$ is called an *edge (of X)*. If $V(X) \neq \emptyset$, we say that X is *non-empty*.

Let X be a graph. If $u = \{x, y\}$ is an edge of X, we say that x and y are *adjacent (for X)*. Let $\pi = \langle x_0, \ldots, x_\ell \rangle$ be an ordered sequence of vertices of X, π is *a path from x_0 to x_ℓ in X (or in $V(X)$)* if for any $i \in [1, \ell]$, x_i is adjacent to x_{i-1}. In this case, we say that x_0 *and x_ℓ are linked for X*. If $\ell = 0$, then π *is a trivial path in X*. We say that X *is connected* if any two vertices of X are linked for X.

Let X and Y be two graphs. If $V(Y) \subseteq V(X)$ and $E(Y) \subseteq E(X)$, we say that Y *is a subgraph of* X and we write $Y \subseteq X$. We say that Y is a *connected component of* X, or simply a *component of* X, if Y is a connected subgraph of X which is maximal for this property, *i.e.*, for any connected graph Z, $Y \subseteq Z \subseteq X$ implies $Z = Y$.

Important remark. *Throughout this paper G denotes a connected graph. In order to simplify the notations, this graph will be denoted by $G = (V, E)$ instead of $G = (V(G), E(G))$. We will also assume that $E \neq \emptyset$.*

Typically, in applications to image segmentation, V is the set of picture elements (pixels) and E is any of the usual adjacency relations, *e.g.*, the 4- or 8-adjacency in 2D [29].

Let $X \subseteq G$. An edge $\{x, y\}$ of G is *adjacent to* X if $\{x, y\} \cap V(X) \neq \emptyset$ and if $\{x, y\}$ does not belong to $E(X)$; in this case and if y does not belong to $V(X)$, we say that $\{x, y\}$ is *outgoing from* X and that y *is adjacent to* X. If π is a path from x to y and y is a vertex of X, then π is a *path from x to X (in G)*.

If S is a subset of E, we denote by $\overline{S}$ *the complementary set of S in E, i.e.,* $\overline{S} = E \setminus S$.

Let $S \subseteq E$, *the graph induced by* S is the graph whose edge set is S and whose vertex set is made of all points which belong to an edge in S, *i.e.*, $(\{x \in V \mid \exists u \in S, x \in u\}, S)$. In the following, when no confusion may occur, the graph induced by S is also denoted by S.

B. Edge-weighted graphs

We denote by $\mathcal{F}$ the set of all maps from E to $\mathbb{Z}$ and we say that any map in $\mathcal{F}$ *weights* the edges of G.

Let $F \in \mathcal{F}$. If u is an edge of G, $F(u)$ is the *altitude or weight* of u. Let $X \subseteq G$ and $k \in \mathbb{Z}$. The subgraph X of G is a *minimum of F (at altitude k)* if:

- X is connected; and
- k is the altitude of any edge of X; and
- the altitude of any edge adjacent to X is strictly greater than k.

We denote by $M(F)$ the graph whose vertex set and edge set are, respectively, the union of the vertex sets and edge sets of all minima of F.

Important remark. *In the sequel of this paper, F denotes an element of $\mathcal{F}$ and therefore the pair (G, F) is called an* edge-weighted graph.

For applications to image segmentation, we will assume that the altitude of u, an edge between two pixels x and y, represents the dissimilarity between x and y (*e.g.*, $F(u)$ equals the absolute difference of intensity between x and y; see Sec. V-A for a more complete discussion on different ways to set the map F for image segmentation). Thus, we suppose that the salient contours are located on the highest edges of G.

II. Watershed-cuts

The intuitive idea underlying the notion of a watershed comes from the field of topography: a drop of water falling on a topographic surface follows a descending path and eventually reaches a minimum. The watershed may be thought of as the separating lines of the domain of attraction of drops of water. Despite its simplicity, none of the classical definitions in discrete frameworks handle exactly this intuitive idea. In this paper, contrarily to previous works, we follow explicitly the drop of water principle to define the notion of a watershed in an edge-weighted graph.

A. Extensions and graph cuts

We present the notions of extension and graph cut which play an important role for defining a watershed in an edge-weighted graph.

Intuitively, the regions of a watershed (also called catchment basins) are associated with the regional minima of the map. Each catchment basin contains a unique regional minimum, and conversely, each regional minimum is included in a unique catchment basin: the regions of the watershed "extend" the minima. In [16], G. Bertrand formalizes the notion of extension.

Definition 1 (Extension, from Def. 12 in [16]): Let X and Y be two non-empty subgraphs of G. We say that Y is an *extension of X (in G)* if $X \subseteq Y$ and if any component of Y contains exactly one component of X.

The graphs (drawn in bold) in Fig. 1b, c and d are three extensions of the one depicted in Fig. 1a.

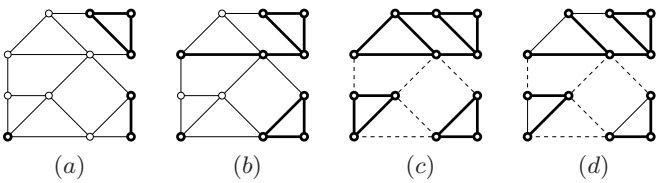

Fig. 1. A graph G. The set of vertices and edges represented in bold is: (a), a subgraph X of G; (b), an extension of X; (c): an extension Y of X which is maximal; and (d): a spanning forest relative to X. In (c) and (d) the set of dashed edges is a cut for X.

The notion of extension is very general. Many segmentation algorithms iteratively extend some seed components in a graph: they produce an extension of the seeds. Most of them terminate once they have reached an extension which covers all the vertices of the graph. The resulting separation is called a graph cut.

Definition 2 (Graph cut, see also [28]): Let $X \subseteq G$ and $S \subseteq E$. We say that S is a *(graph) cut for X* if $\overline{S}$ is an extension of X and if S is minimal for this property, *i.e.*, if $T \subseteq S$ and $\overline{T}$ is an extension of X, then we have $T = S$.

The set S made of the dashed edges in Fig. 1c is a cut for X (Fig. 1a). It can be verified that $\overline{S}$ (in bold Fig. 1c) is an extension of X and that S is minimal for this property.

If X is a subgraph of G and S a cut for X, it may be easily seen that $\overline{S}$ is a maximal extension of X.

The notion of graph cut has been studied for many years and is often defined by means of partitions. In this case, a set $S \subseteq E$ is said to be a graph cut if there exists a partition of V such that S is the set of all edges of G whose extremities are in two distinct sets of the partition. If each set of the partition is connected and contains the vertex set of a unique component of a subgraph of G, then S is a cut for this subgraph. It may be easily seen that this definition is equivalent to Def. 2. One of the most fundamental results in combinatorial optimization involves graph cuts. It states that, given two isolated vertices of an edge-weighted graph (called source and sink), finding a cut of minimal cost that separates these vertices is equivalent to finding a maximum flow (see, for instance, [28], chapter 6.2). There exist polynomial-time algorithms to find the so-called min-cuts. On the other hand, finding a cut of minimal cost among all the cuts for a subgraph which has more than two components is NP-hard [30]. In the forthcoming sections, we introduce the watershed-cuts of an edge-weighted graph and show that these watersheds are graph cuts which also satisfy an optimality property. A major advantage is that they can be computed in linear time.

In image segmentation, a classical application of graph cuts [24] consists of finding a cut of minimum weight (a min-cut) for a set of terminal points in a graph where each vertex is a pixel of an image and each terminal point is included in an object of interest. The links between these approaches and the one developed in this paper are investigated in [31].

B. Watershed-cuts by the drop of water principle

We now introduce the watershed-cuts of an edge-weighted graph. To this end, we formalize the drop of water principle. Intuitively, the catchment basins constitute an extension of the minima and they are separated by "lines" from which a drop of water can flow down towards distinct minima.

Let $\pi = \langle x_0, \ldots, x_\ell \rangle$ be a path in G. The path π is *descending (for F)* if, for any $i \in [1, \ell - 1]$, $F(\{x_{i-1}, x_i\}) \geq F(\{x_i, x_{i+1}\})$.

For instance in Fig. 2a the paths $\langle j, f, b, a \rangle$ and $\langle n, o \rangle$ are descending whereas the path $\langle f, j, n, o, p \rangle$ is not since the altitude of $\{f, j\}$ is strictly less than the one of $\{j, n\}$.

Definition 3 (drop of water principle, watershed cut):
Let $S \subseteq E$. We say that S *satisfies the drop of water principle (for F)* if $\overline{S}$ is an extension of $M(F)$ and if for any $u = \{x_0, y_0\} \in S$, there exist $\pi_1 = \langle x_0, \ldots, x_n \rangle$ and $\pi_2 = \langle y_0, \ldots, y_m \rangle$ which are two descending paths in $\overline{S}$ such that:
- x_n and y_m are vertices of two distinct minima of F; and
- $F(u) \geq F(\{x_0, x_1\})$ (resp. $F(u) \geq F(\{y_0, y_1\})$), whenever π_1 (resp. π_2) is not trivial.
If S satisfies the drop of water principle, we say that S is a *watershed-cut, or simply a watershed, of F*.

We illustrate the previous definition on the function F depicted in Fig. 2. The function F contains three minima (in bold Fig. 2a). We denote by S the set of dashed edges depicted in Fig. 2b. It may be seen that $\overline{S}$ (in bold Fig. 2b) is an extension of $M(F)$. Let us consider the edge $u = \{j, n\} \in S$. There exists $\pi_1 = \langle j, f, b, a \rangle$ (resp. $\pi_2 = \langle n, o \rangle$) a descending path in $\overline{S}$ from j (resp. n) to the minimum at altitude 1 (resp. 3); the altitude of $\{j, f\}$ (resp. $\{n, o\}$), the first edge of π_1 (resp. π_2) is equal to 6 (resp. 5) which is a value lower than 7 the altitude of u. It can be verified that the previous properties hold true for any edge in S. Thus, S is a watershed of F. The next statement follows from the definition of a watershed.

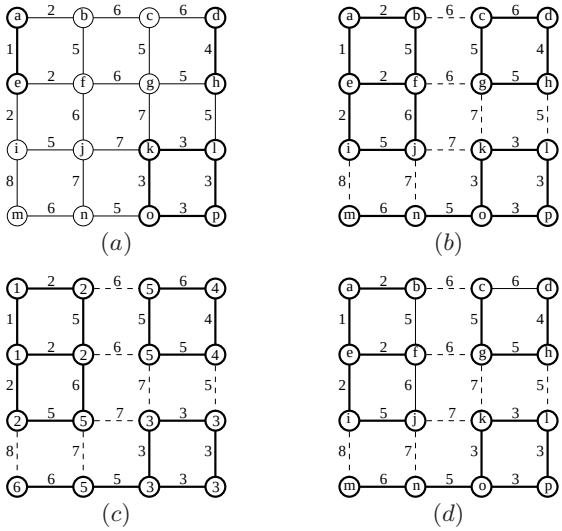

Fig. 2. A graph G and a map F. (a), The minima of F are depicted in bold; (b), the set S of dashed edges is a watershed of F and the graph induced by $\overline{S}$ is depicted in bold; (c), same as (b) but the values of the map $F^{\ominus}$ are associated to the vertices of G; and (d), the subgraph in bold is an MSF relative to $M(F)$ and the induced MSF-cut is composed by the dashed edges.

Property 4: Let $S \subseteq E$. If S is a watershed of F, then S is a cut for $M(F)$.

Notice that a watershed of F is defined thanks to conditions that depend of the altitude of the edges whereas the definition of a cut is solely based on the structure of the graph. Consequently, the converse of Prop. 4 is, in general, not true.

As an illustration of the previous property, it may be verified that the watershed of the map F, depicted in Fig. 2b, is a cut for the minima of F.

To finish this section, we would like to notice that, given an edge-weighted graph, a watershed-cut is not necessarily uniquely defined. There may indeed exist several distinct cuts for $M(F)$ which satisfy the drop of water principle.

C. Catchment basins by a steepest descent property

A popular alternative to Def. 3 consists of defining a watershed exclusively by its catchment basins and the paths of steepest descent (see *e.g.*, [7], [8], [32], [33]). In a vertex-weighted graph, such definitions raise several problems. The catchment basin of a minimum M can be defined as the points from which M can be reached by a path of steepest descent. In this case, several catchment basins may overlap each other. To avoid this problem, some authors define the catchment basin of M as the set of points from which M is the only minimum that can be reached by a path of steepest descent. In this case, some thick sets of points may not belong to any catchment basin (such situations are illustrated in [33]).

The following theorem establishes the consistency of watershed-cuts in edge weighted graphs: they can be equivalently defined by a steepest descent property on the catchment basins (regions) or by the drop of water principle on the cut (border) which separate them. As far as we know, there is no definition of watershed in vertex-weighted graphs that verifies a similar

property. Some counter examples which show that such a duality does not hold in other frameworks can be found in [34]. The following Th. 6 thus emphasizes that the framework considered in this paper is adapted for the definition and study of discrete watersheds.

Before stating Th. 6, we start with some definitions relative to the notion of a path of steepest descent. Then, we derive the definitions of catchment basins and basin-cuts.

Important remark. *From now on, we will denote by $F^{\ominus}$ the map from V to $\mathbb{Z}$ such that for any $x \in V$, $F^{\ominus}(x)$ is the minimal altitude of an edge which contains x,* i.e., *$F^{\ominus}(x) = \min\{F(u) \mid u \in E, x \in u\}$; $F^{\ominus}(x)$ is the* altitude *of x.*

The map $F^{\ominus}$ associated to the map F of Fig. 2a is shown in Fig. 2c.

Let $\pi = \langle x_0, \ldots, x_\ell \rangle$ be a path in G. The path π is *a path of steepest descent for* F if, for any $i \in [1, \ell]$, $F(\{x_{i-1}, x_i\}) = F^{\ominus}(x_{i-1})$.

For instance, in Fig. 2a, $\langle j, i, e \rangle$ and $\langle n, o \rangle$ are paths of steepest descent for the depicted map F. On the contrary, $\langle j, f, b, a \rangle$ and $\langle n, m \rangle$ are not paths of steepest descent for F. Indeed, $F^{\ominus}(j) < F(\{j, f\})$ and $F^{\ominus}(n) < F(\{n, m\})$.

Definition 5 (basin-cut): Let $S \subseteq E$ be a cut for $M(F)$. We say that S is a *basin-cut of* F if, from each point of V to $M(F)$, there exists, in the graph induced by $\overline{S}$, a path of steepest descent for F.

If C is a basin-cut of F, any component of $\overline{C}$ is called a *catchment basin* (of F, for C).

In other words, a cut C for $M(F)$ is a basin-cut of F, if from each point of G to $M(F)$, there exists, in G, a path of steepest descent for F which does not have any edge in the cut C, or said differently all the edges of this path are in a unique component of $\overline{S}$. For instance, it can be verified in Fig. 2b that the set of dashed edges is a basin-cut of the depicted map. The following theorem asserts that any basin-cut of F is a watershed-cut of F and that conversely, any watershed-cut of F is a basin-cut of F.

Theorem 6 (consistency): Let $S \subseteq E$. The set S is a basin-cut of F if and only if S is a watershed-cut of F.

As an illustration of Th. 6, it can be verified that the set of dashed edges in Fig. 2b is both a watershed-cut and a basin-cut of the depicted map.

III. Minimum spanning forests and watershed optimality

In this section, we establish the optimality of watersheds. To this end, we introduce the notion of minimum spanning forests relative to subgraphs of G. We will see that each of these forests induces a unique graph cut. The main result of this section (Th. 10) states that a graph cut is induced by a minimum spanning forest relative to the minima of a map if and only if it is a watershed of this map. In Sec. III-B, we show that the problem of finding a relative minimum spanning forest is equivalent to the classical problem of finding a minimum spanning tree [35]–[37]. In fact, this provides a mean to derive, from any minimum spanning tree algorithm, an algorithm for relative minimum spanning forests, and thus also, for watersheds.

Intuitively, a forest relative to a subgraph X is an extension Y of X such that any cycle (*i.e.*, a simple path whose first and last point are adjacent) in Y is a cycle in X. Formally, the notion of cycle is not necessary to define a forest.

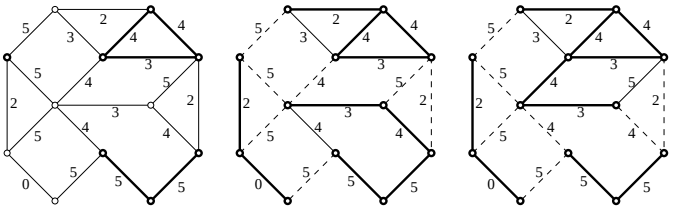

Fig. 3. A graph G and a map F. The bold edges and vertices represent: (a), X a subgraph of G; (b) and (c), two MSFs relative to X; their induced cuts are represented by dashed edges.

Definition 7 (relative forest): Let X and Y be two non-empty subgraphs of G. We say that Y is a *forest relative to* X if:

i) Y is an extension of X; and
ii) for any extension $Z \subseteq Y$ of X, we have $Z = Y$ whenever $V(Z) = V(Y)$.

We say that Y is a *spanning forest relative to X (for G)* if Y is a forest relative to X and if $V(Y) = V$.

Informally speaking, condition ii) imposes that, if Y is a forest, then we cannot remove any edge from Y without affecting its vertex set.

For example, the subgraph depicted in bold in Fig. 1d is a spanning forest relative to the subgraph in Fig. 1a.

Thanks to relative forests, the usual notion of a tree and of a forest can be defined as follows.

Let $X \subseteq G$. We say that X is a *tree* (resp. a *spanning tree*) if X is a forest (resp. spanning forest) relative to the subgraph $(\{x\}, \emptyset)$, x being any vertex of X. We say that X is a *forest* (resp. a *spanning forest*) if X is a forest (resp. a spanning forest) relative to $(S, \emptyset)$, S being a subset of $V(X)$.

Let X be a subgraph of G, *the weight of X (for F)*, denoted by $F(X)$, is the sum of its edge weights: $F(X) = \sum_{u \in E(X)} F(u)$.

Definition 8 (relative minimum spanning forest): Let X and Y be two subgraphs of G. We say that *Y is a minimum spanning forest (MSF) 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, we also say that Y is a *relative MSF*.

Let us consider the graph G depicted in Fig. 3 and the subgraph X depicted in bold in Fig. 3a. The graphs Y and Z (bold edges and vertices) in Figs. 3b and c are two MSFs relative to X.

A. Relative MSFs and watersheds

We now have the mathematical tools to present the main result of this section (Th. 10) which establishes the optimality of watersheds. It shows the equivalence between the cuts which satisfy the drop of water principle and those induced by the MSFs relative to the minima of a map.

We start by the following lemma which gives, thanks to Th. 6, a first intuition of Th. 10.

Lemma 9: Let X be a spanning forest relative to $M(F)$. The graph X is an MSF relative to $M(F)$ if and only if, for any x in V, there exists a path in X from x to $M(F)$ which is a path of steepest descent for F.

Let X be a subgraph of G and let Y be a spanning forest relative to X. 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 is also a cut for X (see, for instance, Fig. 1d). We say that this unique cut is the *cut induced by* Y. Furthermore, if Y is an MSF relative to X, we say that S is an *MSF-cut for* X.

For instance, in Fig. 3b,c, the set of dashed edges are MSF-cuts for the subgraph shown in bold in Fig. 3a.

Theorem 10 (optimality): Let $S \subseteq E$. The set S is an MSF-cut for $M(F)$ if and only if S is a watershed cut of F.

As far as we know, this is the first result which establishes watershed optimality in graphs. As an illustration, the previous theorem can be verified on Fig. 2b,d where the set of dashed edges is both a watershed-cut of the depicted map and an MSF-cut for its minima.

B. Relative MSFs and minimum spanning trees

The minimum spanning tree problem is one of the most typical and well-known problems of combinatorial optimization (see [35]–[38]). It has been applied for many years in image analysis [39]. We show that the minimum spanning tree problem is equivalent to the problem of finding an MSF relative to a subgraph of G.

Let $X \subseteq G$. The graph X *is a minimum spanning tree (for* F*, in* G*)* if X is an MSF relative to the subgraph $(\{x\}, \emptyset)$, x being any vertex of X.

Notice that the notion of a minimum spanning tree presented above corresponds exactly to the usual one.

In order to recover the link between flooding algorithms and minimum spanning trees, in [19], F. Meyer proposed a construction to show the equivalence between finding an MSF rooted in a set of vertices and finding a minimum spanning tree. Here, we extend this construction for proving the equivalence between finding a minimum spanning tree and an MSF relative to a subgraph of G. Let us consider, in a first time, a graph $X \subseteq G$ such that $E(X) = \emptyset$, *i.e.*, a graph composed of isolated vertices. From G and X, we can construct a new graph $G' = (V', E')$ which contains an additional vertex z (*i.e.*, $z \notin V$) linked by an edge to each vertex of X. In other words, $V' = V \cup \{z\}$ and $E' = E \cup E_z$, where $E_z = \{\{x, z\} \mid x \in V(X)\}$. Let us consider the map F' from E' to $\mathbb{Z}$ such that, for any $u \in E$, $F'(u) = F(u)$ and for any $u \in E_z$, $F'(u) = k_{\min} - 1$, $k_{\min}$ being the minimum value of F. Let Y be any subgraph of G and let Y' be the graph whose vertex and edge sets are respectively $V(Y) \cup \{z\}$ and $E(Y) \cup E_z$. It may be seen that Y' is a minimum spanning tree for F' in G' if and only if Y is an MSF relative to X for F in G.

The construction presented above can be easily generalized to any subgraph X of G. To this end, in a preliminary step, each component of X must be contracted into a single vertex and, if two vertices of the contracted graphs must be linked by multiple edges, only the one with minimal value is kept.

A direct consequence of the construction presented above is that any minimum spanning tree algorithm can be used to compute a relative MSF. Many efficient algorithms (see [37]) exist in the literature for solving the minimum spanning tree problem.

IV. Streams and linear-time watershed algorithm

As seen in the previous section, MSFs relative to subgraphs of G, and by the way watershed-cuts, can be computed by any minimum spanning tree algorithm. The best complexity for solving this problem is reached by the quasi-linear algorithm of Chazelle [40]. In this section, we introduce a linear-time watershed algorithm. Contrarily to many watershed algorithms available in the literature (see [5], [6], [11], [12], [17], [25]), the proposed algorithm does not require any sorting step, nor the use of a sophisticated data structure such as a hierarchical queue or a representation to maintain unions of disjoint sets. Whatever the range of the considered map, it runs in linear time with respect to the size of the input graph. Furthermore, this algorithm does not need to compute the minima of the map in a preliminary step. To the best of our knowledge, this is the first watershed algorithm with such properties.

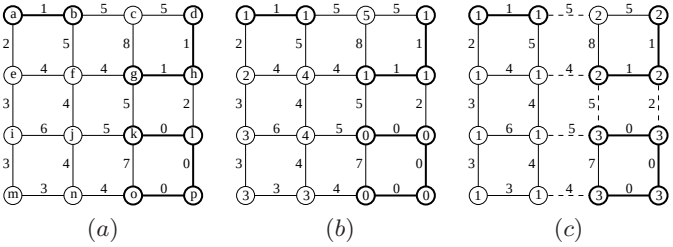

Fig. 4. A graph G and a map F assigned to the edges of G. The bold graphs superimposed are the minima of F; (b), the values of the map $F^{\ominus}$ are associated to the vertices of G; (c): a flow mapping ψ of F is depicted; the index of the top-left (resp top-right and bottom-left) minima is 1 (resp. 2 and 3); the set of dashed edges is the flow-cut of F associated with ψ.

In the first part of the section, the mathematical tools which are used to prove the correctness of the proposed algorithm are introduced. In particular, we propose a new notion of stream which is crucial to this paradigm. Then, the algorithm is presented, and both its correctness and complexity are analyzed.

Definition 11 (stream): Let $L \subseteq V$. We say that L is a *stream* if, for any two points x and y of L, there exists, in L, either a path from x to y or from y to x, of steepest descent for F.
Let L be a stream and let $x \in L$. We say that x *is a top (resp. bottom) of* L if the altitude of x is greater than (resp. less than) or equal to the altitude of any $y \in L$.

Remark that if L is a stream and x is a bottom (resp. a top) of L, then, from any $y \in L$ to x (resp. from x to any $y \in L$), there is a path in L, of steepest descent for F. Notice that, whatever the stream L, there exists a top (resp. bottom) of L. Nevertheless, this top (resp. bottom) is not necessarily unique.

In order to illustrate the previous definitions, let us assume that G and F are the graph and the function depicted in Fig. 4a. The sets $L = \{a, b, e, i\}$ and $\{j, m, n\}$ are two examples of streams. On the contrary, the set $L' = \{i, j, k\}$ is not a stream since there is no path in L', between i and k, of steepest descent for F. The sets $\{a, b\}$ and $\{i\}$ are respectively the set of bottoms and tops of L.

The algorithm which will be proposed in this section is based on the iterative extraction of streams. In order to build such a procedure, we study stream concatenation.

Let L_1 and L_2 be two disjoint streams (*i.e.*, $L_1 \cap L_2 = \emptyset$) and let $L = L_1 \cup L_2$. We say that L_1 *is under* L_2, written $L_1 \prec L_2$, if there exist a top x of L_1, a bottom y of L_2, and there is, from y to x, a path in L of steepest descent for F. Note that, if $L_1 \prec L_2$, then L is also a stream.
We say that a stream L is an *inf-stream, written* $\prec$*-stream*, if there is no stream under L.

In Fig. 4a the stream $\{a, b, e, i\}$ is under the stream $\{j, m, n\}$

and thus $\{a, b, e, i, j, m, n\}$ is also a stream. Furthermore, there is no stream under $\{a, b, e, i\}$ and $\{a, b, e, i, j, m, n\}$. Thus, these are two $\prec$-streams.

The streams extracted by our algorithm are all $\prec$-streams. As said in the introduction, this algorithm does not require minima precomputation. In fact, there is a deep link between $\prec$-streams and minima as assessed by the following property which follows directly from the definitions of a minimum and of an $\prec$-stream.

Property 12: Let L be a stream. The three following statements are equivalent:
(1) L is an $\prec$-stream;
(2) L contains the vertex set of a minimum of F; and
(3) for any $y \in V \setminus L$ adjacent to a bottom x of L, $F(\{x, y\}) > F^{\ominus}(x)$.

In Fig. 4a, the two $\prec$-streams $\{a, b, e, i\}$ and $\{a, b, e, i, j, m, n\}$ contain the set $\{a, b\}$ which is the vertex set of a minimum of F. Remark that any stream L which contains an $\prec$-stream is itself an $\prec$-stream. We also notice that if L is an $\prec$-stream, then the set of all bottoms of L constitutes the vertex set of a minimum of F. Furthermore, a subset L of V is the vertex set of a minimum of F if and only if it is an $\prec$-stream minimal for the inclusion relationship, *i.e.*, no proper subset of L is an $\prec$-stream.

In order to partition the vertex set of G, from the $\prec$-streams of F, the vertices of the graph can be arranged in the following manner.

Let $\mathcal{L}$ be a set of n $\prec$-streams. We say that $\mathcal{L}$ is a *flow family* if:

i) $\cup\{L \mid L \in \mathcal{L}\} = V$; and
ii) for any two distinct L_1 and L_2 in $\mathcal{L}$, if $L_1 \cap L_2 \neq \emptyset$, then there exists a unique minimum of F whose vertex set is included in $L_1 \cap L_2$.

For instance, in Fig. 4, the family composed of the sets $\{a, b, e, f, j\}$, $\{a, b, e, i, m, n\}$, $\{c, d, g, h\}$ and $\{k, l, o, p\}$ is a flow family.

Let $\mathcal{L}$ be a flow family, let $x \in V$ and let $L_1, \ldots L_\ell$ be the elements of $\mathcal{L}$ which contain x. Since the elements of $\mathcal{L}$ are $\prec$-streams, by Prop. 12, any L_i (with $i \in [1, \ell]$) contains the vertex set of exactly one minimum M_i of F. By definition of a flow family, we deduce that, for any i and j in $[1, \ell]$, $M_i = M_j$. Thus, thanks to $\mathcal{L}$, we can associate to each vertex x of G a unique minimum of F.

Definition 13 (flow-cut): Let $\mathcal{L}$ be a flow family. Let us denote by $M_1, \ldots, M_n$ the minima of F. Let ψ be the map from V to $[1, n]$ which associates to each vertex x of V, the index (or label) i such that M_i is the unique minimum of F included in an $\prec$-stream of $\mathcal{L}$ which contains x; we say that ψ *a flow mapping of* F.
If ψ is a flow mapping of F, we say that the set $S = \{\{x, y\} \in E \mid \psi(x) \neq \psi(y)\}$ is a *flow-cut of* F.

Fig. 4c shows the flow mapping associated to the flow family presented above. The dashed edges represent the flow-cut induced by this flow mapping.

The next proposed algorithm produces a flow mapping, hence a flow-cut. The following theorem, which is a straightforward consequence of the definitions of flow families and basin-cuts and of the consistency theorem, states the equivalence between flow-cuts and watersheds. It constitutes the main tool to establish the correctness of Algo. 1.

Theorem 14: Let $S \subseteq E$. The set S is a watershed of F if and only if S is a flow-cut of F.

As an illustration of this theorem, it may be verified that the flow-cut depicted in Fig. 4c is a watershed-cut and that the watershed-cut of Fig. 2b is flow-cut.

We now present Algo. 1 which computes a flow mapping, hence, by Th. 14, a watershed. Algo. 1 makes use of the function Stream introduced hereafter.

Algorithm 1: Watershed

Data: (V, E, F): an edge-weighted graph;
Result: ψ: a flow mapping of F.

```
1  foreach x ∈ V do ψ(x) ← NO_LABEL;
2  nb_labs ← 0; // the number of minima already found
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 /* L is an ≺-stream */
6          nb_labs + + ;
7          foreach y ∈ L do ψ(y) ← nb_labs;
8      else
9          foreach y ∈ L do ψ(y) ← lab;
```

Function `Stream(` V, E, F, ψ, x `)`

Data: (V, E, F): an edge-weighted graph; ψ: a labeling of V; x: a point in V.
Result: $[L, lab]$ where L is a stream such that x is a top of L, and lab is either the label of an $\prec$-stream under L, or -1.

```
1  L ← {x} ;
2  L′ ← {x} ; // the set of non-explored bottoms of L
3  while there exists y ∈ L′ do
4      L′ ← L′ \ {y};
5      breadth_first ← TRUE ;
6      while (breadth_first) and (there exists {y, z} ∈ E
       such that z ∉ L and F({y, z}) = F^⊖(y)) do
7          if ψ(z) ≠ NO_LABEL then
8              /* there is an ≺-stream under L already
               labelled */
9              return [L, ψ(z)] ;
10         else if F^⊖(z) < F^⊖(y) then
11             L ← L ∪ {z} ; /* z is now the only bottom
               of L */
12             L′ ← {z} ; /* hence, switch to depth-first
               exploration */
13             breadth_first ← FALSE ;
14         else
15             L ← L ∪ {z} ; /* F^⊖(z) = F^⊖(y), thus z is
               also a bottom of L */
16             L′ ← L′ ∪ {z} ; /* continue breadth-first
               exploration */
17 return [L, −1] ;
```

The algorithm iteratively assigns a label to each point of the graph. To this end, from each non-labeled point x, a stream L composed of non-labeled points and whose top is x is computed (line 4). If L is an $\prec$-stream (line 5), a new label is assigned to the points of L. Otherwise (line 8), there exists an $\prec$-stream L_1 under L and which is already labeled. In this case, the points of L

receive the label of L_1 (line 9). The function Stream, called at line 4, allows us to compute the stream L. Roughly speaking, it performs an intermixed sequence of depth-first and breadth-first exploration of the paths of steepest descent. The main invariants of the function Stream are: i), the set L is, at each iteration, a stream; and ii), the set L' is made of all non-already explored bottoms of L. The function halts at line 17 when all bottoms of L have been explored or, at line 9, if a point z already labeled is found. In the former case, by Prop. 12, the returned set L is an $\prec$-stream. In the latter case, the label lab of z is also returned and there exists a bottom y of L such that $\langle y, z\rangle$ is a path of steepest descent. Thus, there is an $\prec$-stream L_1, under L, included in the set of all vertices labeled lab. By the preceding remarks, the output of Algo. 1 is a flow mapping of F.

Let us now analyze the complexity of Algo. 1. In order to prove its linearity (with respect to $|E|$), we are going to show that the bottleneck of Algo. 1, which consists of the tests in the While loop (line 3) of function Stream, is executed at most $O(|E|)$ times. Firstly, it is easily seen that, at a each step of Stream, any point y which belongs to L is such that $\psi(y) = NO\_LABEL$. Furthermore, it may be also noticed (lines 1, 2 and 10, 11 and 14, 15) that any point in L' also belongs to L. In Stream the points are never removed from L. Thus, since to be inserted in L' a point z must not be an element of L (test $z \notin L$ line 3 of Stream), we deduce that any point z is inserted at most once in L'. Therefore, the While loop (line 3 of Stream) is executed at most once for each point y in L' (since y is removed from L' just before the execution of the loop). In this loop a set of tests is performed for each neighbor of y. Since the points of L receive a label (line 7 or 9 in Algo. 1) just after the termination of Stream and since Stream only considers non-labeled points, we deduce that the tests in the While loop (line 3 of Stream) are executed at most once for each point of the graph. Thus, an edge being composed of exactly two points, these tests are executed at most $2\times|E|$ times. Furthermore, in order to perform the canonical operations of Algo. 1 in constant time and thus to achieve a linear complexity, the graph (V, E) can be stored as an array of lists which maps to each point the list of all its adjacent vertices (or equivalently the list of all edges which contain this point). Notice that, for applications to image processing, and when usual adjacency relations are used, these structures do not need to be explicit. From the preceding remarks, we can deduce the following property.

Property 15: Algorithm 1 outputs a map ψ which is a flow mapping of F. Furthermore, Algorithm 1 runs in linear-time with respect to $|E|$.

Remark that, in function Stream, the use of breadth-first iterations is required to ensure that the produced set L is always an $\prec$-stream. Otherwise, if only depth-first iterations were used, Stream could be stuck on plateaus (*i.e.*, connected subgraphs of G with constant altitude) since some bottoms of L would never be explored.

Let us note that the two sets L and L' can be efficiently managed by stack, which is a simple and efficient data structure. As far as we know, the watershed algorithms available in the literature (*e.g.*, [5], [6], [11], [12], [17], [25]) all require either a sorting step, a hierarchical queue or a data structure to maintain a collection of disjoint sets under the operation of union. On the one hand, the global complexities of a sorting step and of a (monotone) hierarchical queue (*i.e.*, a structure from which the elements can be removed in the order of their altitude) are equivalent [41]: they both run in linear-time only if the range of the weights is sufficiently small. On the other hand, the best complexity for the disjoint set problem is quasi-linear [42]. Therefore, we emphasize that, to the best of our knowledge, the proposed algorithm (together with the algorithm introduced in [43]) is the first watershed algorithm that runs in linear-time whatever the range of the weight map.

In practice, Algorithm 1 is as fast as a minima computation algorithm. Each catchment basin is associated to a minimum of the original map. For practical applications, one does not always need a basin for each minimum of the image. The following section illustrates how to apply the watershed cuts to image segmentation.

V. Illustrations in image segmentation

In order to illustrate the notions introduced in this paper, we present two segmentation schemes based on watersheds and relative MSFs. After having described (Sec. V-A) how to set up the edge-weighted graph, in Sec. V-B, we derive, from the classical framework of mathematical morphology, a segmentation scheme that permits to automatically segment an image into a predefined number of regions. It consists of the three following steps: (i), computation of a function that assigns a weight to the edges of the 4-adjacency graph associated to the image; (ii), filtering of this weight function in order to reduce the number of minima; and (iii) computation of a watershed of the filtered weight function. The second illustration (Sec. V-C) presents some results of relative MSF, used as a semi-automatic segmentation tool.

A. Graph setup

Even if watersheds are sometimes applied on region adjacency graphs [19], we focus, in this paper, on watershed methods based on pixel adjacency graphs (*i.e.*, graphs whose vertices are the image pixels). Therefore, we assume that the set V is the domain of a 2-dimensional image, more precisely, of a rectangular subset of $\mathbb{Z}^2$. A grayscale image I is a map from the set of pixels V to a subset of the positive integers. For any $x \in V$, the value $I(x)$ is the intensity at pixel x. In order to define a graph over the set of pixels, we consider the 4-adjacency relation [29] defined by: $\forall x, y \in V$, $\{x, y\} \in E$ iff $|x_1 - y_1| + |x_2 - y_2| = 1$, where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. Note that, instead of the 4-adjacency, any other adjacency relation could be used since our work is settled in general graphs. Then, before extracting a watershed-cut from this graph, a map F, which weights the edges of $G = (V, E)$, must be defined. Depending on the application, there are several possibilities to set up the map F.

Let us first consider the "classical" watershed problem, where we want to segment dark regions that are separated by brighter zones (see, for instance, Fig. 5a). In this case, the watershed-cuts can be used, as well as any watershed algorithm settled in vertex-weighted graphs. To this end, the value of F can be defined for each edge $u \in E$, linking two pixels x and y, by the minimum (or maximum) value of the intensities at points x and y: $F(\{x, y\}) = \min\{I(x), I(y)\}$. Fig. 5 illustrates this procedure and also presents the result of a watershed algorithm applied in the vertex-weighted graph associated to the image. It can be observed (see in partiular Fig. 5e and f) that, for this "classical" problem, similar results are obtained in both frameworks of edge-weighted graphs and vertex-weighted graphs.

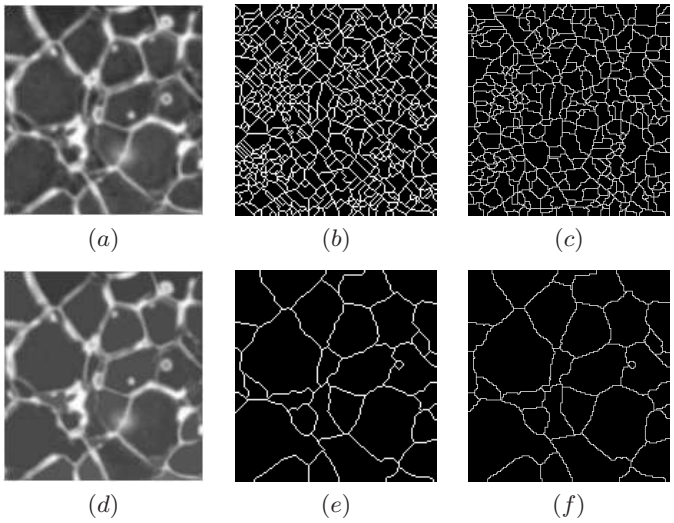

Fig. 5. (a): Original image (microscopic view of a cross-section of a uranium oxyde ceramics); (b): a watershed (white pixel) of (a) considered as a vertex-weighted graph; (c): a watershed-cut of the map F derived from (a) as described in Sec. V-A; (d), a filtration of the original image (a) where the new image is obtained by eliminating the minima whose dynamics [44], [45] is below 25; (e, f): same as (b, c) starting from (d).

Another common issue is to segment a grayscale image into its "homogeneous" zones. To solve this problem in the conventional framework of watersheds, an image I' which has low values in homogeneous zones and high values at the interfaces between the homogeneous zones must be considered. Then, a watershed is extracted from this image I' leading to a segmentation into the homogeneous parts for I. In general, I' is chosen as the gradient magnitude of the original image I. Computing such a gradient magnitude image is not straightforward and several solutions exist (*e.g.*, the Sobel filter [46], the Deriche's optimal edge detector [47] and the morphological gradient [48]). In the framework of edge-weighted graphs, a straightforward gradient function can be used in order to weight the edges of G. In the following, we consider the map F, from E to $\mathbb{Z}$, defined for any $\{x, y\} \in E$ by $F(\{x, y\}) = |I(x) - I(y)|$. For instance, in Fig. 6b, we show an image representation of the map F derived from the image I of Fig. 6a. In the two next sections, we show that this gradient function on the edges leads to satisfactory segmentation results. However, more elaborated formulations (taking into account, for instance, a regularization term) could also be used to define the cost function F (see [49] or an adaptation of [47]). Furthermore, there also exist, in the literature (*e.g.*, [22]), some formulations to define F from multi-channel images, such as color images.

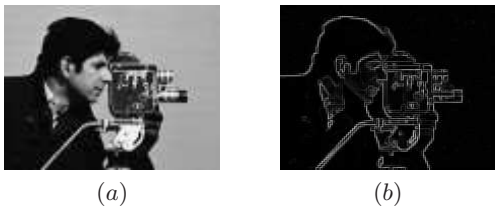

Fig. 6. (a), The cameraman grayscale image and (b), an image representation of the edge-weighted graph derived from (a) as described in Sec. V-A.

The position of the contours produced by watersheds on the plateaus is the subject of many discussions [8], [33], [50]. An usual choice is to place the divide in the "middle" of the plateau. This choice is not always the best one [8], for example it is not adapted for hierarchical schemes [51]. Observe that Algorithm 1 does not include a control of the location of watersheds on plateaus. Such a control can be obtained through a (linear-time) preprocessing [33]; however, note that it is not always needed. For example, in the sequel, we present the result of the algorithm directly on the data, without any preprocessing dedicated to plateaus. In [43], [52], we propose some other algorithms that introduce more flexibility in treatment of plateaus.

B. Segmentation into k regions

In this section, we illustrate the use of watershed-cuts to segment an image into its homogeneous zones. To this end, we consider the cameraman image presented Fig. 6a and adapt a classical scheme of morphological segmentation. Indeed, a watershed of the map F defined above, would contain too many catchment basins. Over-segmentation is a well known feature of all grayscale watersheds due to the huge number of local minima. In order to suppress many of the non-significant minima, a classical approach consists of computing morphological closing of the function [53], [54]. In particular, attribute filters [55] (area, dynamic, volume) have shown to be successful tools. For this illustration, we adapt a classical attribute filter to the case of edge-weighted graphs.

The intuitive idea of this filter is to progressively "fill in" the minima of the map F that are not "important enough". To make such an idea practicable, it is necessary to quantify the relative importance of a minimum. To this end, let us define the *area* of a subgraph of G (*e.g.*, a minimum of F) as the number of its vertices. In order to "fill in" a less significant minimum M of F (according to its area), we consider the transformation that consists of increasing by one the altitude of any edge of M. A common issue in image analysis is to segment an image into k regions (where k is a predefined number). To reach this goal thanks to watershed-cuts, we need a weight function which contains exactly k minima. The map F is thus filtered by iterating the above transformation until F contains k minima (see [56] for an efficient implementation).

In Figs. 7a,b, we present the results which have been obtained on the cameraman image. Here, k is set to 22. In order to evaluate this result, we also use a similar approach settled in the framework of vertex-weighted graph. More precisely, it consists of: (i), computation of a gradient magnitude image: either the Deriche's optimal edge detector [47] in Figs. 7c,d or the morphological gradient (see, for instance, chapter 3.10.1 in [48]) in Figs. 7e,f; (ii), area filtering ($k = 22$) of the gradient; followed by (iii), computation of a watershed by flooding (without dividing line, see [6] or [26]) of the filtered function. Observe, in particular, the quality of the delineation of the man's face in (b) compared to (d) and (f).

C. Image segmentation from markers

Another classical procedure in mathematical morphology consists in selecting (either manually or with an automated process) some markers corresponding to objects that have to be segmented. These markers are indeed some vertices of the underlying graph. Let M be this set of vertices. From the set M the subgraph M^+ whose vertex set is M and whose edge set is made of the

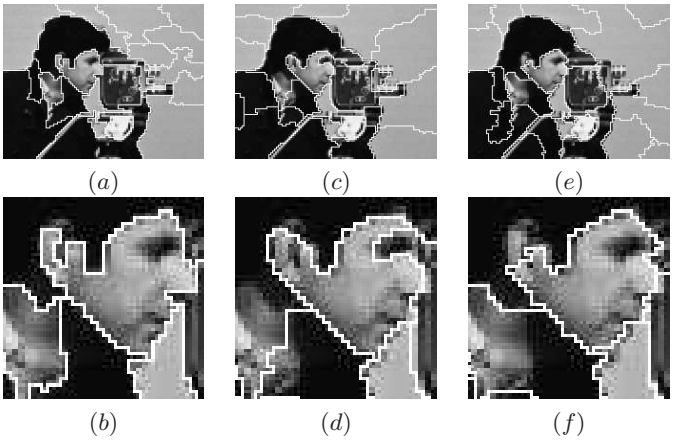

Fig. 7. Results obtained by applying a grayscale watershed on a filtered map [see text]. (a, b) A watershed-cut ($k = 22$) superimposed in white to the original image I; (c, d) a watershed by flooding of the filtered ($k = 22$) Deriche optimal edge detector; and (e, f) a watershed by flooding of a filtered ($k = 22$) morphological gradient. In each image, the image resolution is doubled in order to superimpose the resulting contours.

edges of G which have their extremities in M, (*i.e.*, $M^+ = (M, \{\{x, y\} \in E \text{ with } x \in M, y \in M\})$) is extracted. Then, an MSF relative to M^+ is computed. Here, we use a Prim-like minimum spanning tree algorithm [36]. We note that it is possible to efficiently compute minimum spanning trees by an algorithm which consists of a succession of watersheds [57]. Such an algorithm could be also used to produce relative MSFs. Alternatively, we also could have used a process suggested by [19], that consists in computing a watershed-cut with Algorithm 1, followed by a region-merging scheme on the neighborood graph of the basins; such a process is very efficient and very fast, as it works on a minimum spanning tree of the original cost function [50].

Such an interactive segmentation procedure is illustrated in Figs. 8. For comparison purpose, we also compute the watershed by flooding from markers [6] of the gradient magnitude (the Deriche's optimal edge detector [47] in Fig. 8d and morphological gradient in Fig. 8e). We can observe the quality of the delineation in 8c, compared to (d) and (e). See, in particular, the behavior of our approach in low contrasted zones and in the thin parts of the apple.

CONCLUSION

In this paper, we introduce the watershed-cuts, a notion of watershed in edge-weighted graphs. We prove the consistency and optimality of the watershed-cuts:

- they can be equivalently defined by a steepest descent property on the catchment basins (regions) and by the drop of water principle on the cut (border) which separates them;
- they are equivalent to the separations induced by minimum spanning forests relative to the regional minima.

Then, we propose a linear-time algorithm to compute the watershed-cuts. As far as we know, the proposed algorithm is the most efficient existing watershed algorithm both in theory and practice. Finally, we illustrate the use of watershed-cuts for application to image segmentation and show that, in the considered cases, they are able to improve the quality of the delineation in watershed-based segmentation procedures.

In [43], [52], we introduce a new thinning transformation which equivalently defines the watershed-cuts. On the one hand, this transform permits to introduce flexible sequential algorithms (*e.g.*, for centering the watershed-cuts on plateaus or for watershed-cuts from markers) and opens the way towards efficient parallel watershed strategies. On the other hand, thanks to this new transform, we are able to study the similarities and differences between watershed-cuts and other popular segmentation paradigms such as the Image-Foresting-Transform [25], the fuzzy-connected image segmentation method [21] or the topological watershed [16]. An important result of this study is that any watershed-cut is a *topological-cut* (*i.e.*, a separation obtained by a topological watershed defined in an edge-weighted graph). Thus, the watershed-cuts inherit the properties proved for topological watersheds. In particular, they "preserve the connection value", which is a fundamental property for many hierarchical methods based on watersheds [51], [58], [59].

REFERENCES

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

[2] C. Jordan, "Nouvelles observations sur les lignes de faîtes et de thalweg," *Comptes Rendus des Séances de l'Académie des Sciences*, vol. 75, pp. 1023–1025, 1872.

[3] H. Digabel and C. Lantuéjoul, "Iterative algorithms," in *Proc. 2nd European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, 1978, pp. 85–89.

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

[5] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 583–598, June 1991.

[6] F. Meyer, "Un algorithme optimal de ligne de partage des eaux," in *Procs. of 8ème Congrès AFCET*, Lyon-Villeurbanne, France, 1991, pp. 847–859.

[7] ——, "Topographic distance and watershed lines," *Signal Processing*, vol. 38, no. 1, pp. 113–125, 1993.

[8] L. Najman and M. Schmitt, "Watershed of a continuous function," *Signal Processing*, vol. 38, no. 1, pp. 68–86, 1993.

[9] F. Lemonnier, "Architecture électronique dédiée aux algorithmes rapides de segmentation basés sur la morphologie mathématique," Ph.D. dissertation, CMM, Ecole des Mines de Paris, december 1996.

[10] M. Couprie and G. Bertrand, "Topological grayscale watershed transform," in *Procs. of SPIE Vision Geometry V*, vol. 3168, 1997, pp. 136–146.

[11] A. Meijster and J. Roerdink, "A disjoint set algorithm for the watershed transform," in *Procs. of Eusipco : European signal processing conference*, 1998, pp. 1669–1672.

[12] A. Bieniek and A. Moga, "A connected component approach to the watershed segmentation," in *Mathematical Morphology and its Applications to Image and Signal Processing, procs. ISMM'98*, 1998, pp. 215–222.

[13] S. Stoev, "Rafsi - a fast watershed algorithm based on rainfalling simulation," in *Proceedings of 8-th International Conference on Computer Graphics, Visualization, and Interactive Digital Media (WSCG'2000)*, 2000.

[14] J. Cousty, M. Couprie, L. Najman, and G. Bertrand, "Weighted fusion graphs: merging properties and watersheds," *Discrete Appl. Math.*, 2008, accepted. Also in technical report IGM2007-09. [Online]. Available: http://igm.univ-mlv.fr/LabInfo/rapportsInternes/2007/09.pdf

[15] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," *E. Dougherty (Ed.), Mathematical Morphology in Image Processing, Marcel Decker*, pp. 443–481, 1993.

[16] G. Bertrand, "On topological watersheds," *J. Math. Imaging Vis.*, vol. 22, no. 2-3, pp. 217–230, May 2005.

[17] M. Couprie, L. Najman, and G. Bertrand, "Quasi-linear algorithms for the topological watershed," *J. Math. Imaging Vis.*, vol. 22, no. 2-3, pp. 231–249, 2005.

[18] L. Najman, M. Couprie, and G. Bertrand, "Watersheds, mosaics and the emergence paradigm," *Discrete Appl. Math.*, vol. 147, no. 2-3, pp. 301–324, 2005.

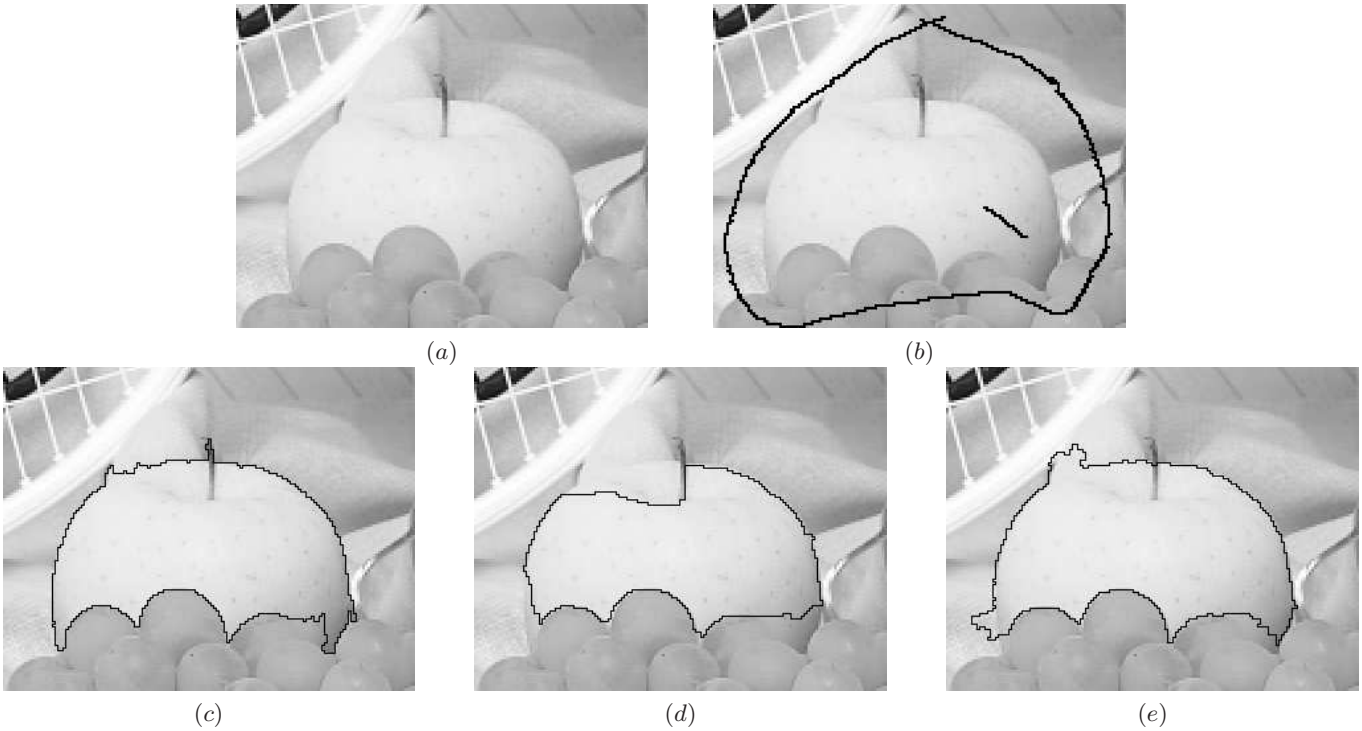

Fig. 8. Comparison of different watersheds from markers. (a) Original image; (b) the markers are superimposed in black. In the second row the resulting watersheds are superimposed in black to the original image. (c): Relative MSF; (d): watershed by flooding of the Deriche optimal edge detector; (e): watershed by flooding of a morphological gradient of the image.

[19] F. Meyer, "Minimum spanning forests for morphological segmentation," in *Procs. of the second international conference on Mathematical Morphology and its Applications to Image Processing*, September 1994, pp. 77–84.

[20] J. K. Udupa and S. Samarsekara, "Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," *Graphical Models and Image Processing*, vol. 58, pp. 246–261, 1996.

[21] J. K. Udupa, P. K. Saha, and R. A. Lotufo, "Relative fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 11, pp. 1485–1500, 2002.

[22] Y. Zhuge, J. K. Udupa, and P. K. Saha, "Vectorial scale-based fuzzy-connected image segmentation," *Comput. Vis. Image Underst.*, vol. 101, pp. 177–193, 2006.

[23] R. Englert and W. Kropatsch, "Image structure from monotonic dual graph," in *AGTIVE'99*, vol. LNCS 1779, 2000, pp. 297–308.

[24] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, 2001.

[25] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo, "The image foresting transform: theory, algorithm and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, pp. 19–29, 2004.

[26] R. Lotufo and A. Falcão, "The ordered queue and the optimality of the watershed approaches," in *Procs. of the 5th International Symposium on Mathematical Morphology*, 2000, pp. 341–350.

[27] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts," in *procs. ISMM*, 2007, pp. 301–312.

[28] R. Diestel, *Graph Theory*, ser. Graduate Texts in Mathematics. Springer, 1997.

[29] T. Kong and A. Rosenfeld, "Digital topology: Introduction and survey," *Comput. Vision Graph. Image Process.*, vol. 48, no. 3, pp. 357–393, 1989.

[30] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, "Complexity of multiway cuts," in *Procs. of the 24th Annual ACM Symposium on the Theory of Computing*, 1992, pp. 241–251.

[31] C. Allène, J.-Y. Audibert, M. Couprie, J. Cousty, and R. Keriven, "Some links between min-cuts, optimal spanning forests and watersheds," in *procs. ISMM*, 2007, pp. 253–264.

[32] P. Soille and C. Gratin, "An efficient algorithm for drainage network extraction on DEMs," *Journal of Visual Communication and Image Representation*, vol. 5, no. 2, pp. 181–189, 1994.

[33] J. B. T. M. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta Informaticae*, vol. 41, no. 1-2, pp. 187–228, 2001.

[34] J. Cousty, "Lignes de partage des eaux discrètes : théorie et application à la segmentation d'images cardiaques," Ph.D. dissertation, Université de Marne-la-Vallée, FRANCE, 2007.

[35] J. Kruskal, "On the shortest spanning tree of a graph and the traveling salesman problem," *procs. Amer. Math. Soc.*, vol. 7, pp. 48–50, 1956.

[36] R. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.*, vol. 36, pp. 1389–1401, 1957.

[37] R. L. Graham and P. Hell, "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, pp. 43–57, 1985.

[38] J. Nesetril, E. Milková, and H. Nesetrilová, "Otakar Boruvka on minimum spanning tree problem. Translation of both 1926 papers, comments, history," *Discrete Mathematics*, vol. 233, pp. 3–36, 2001.

[39] C. Zahn, "Graph-theoretical methods for detecting and descibing gestalt clusters," *IEEE Transactions on Computers*, vol. C-20, no. 1, pp. 99–112, 1971.

[40] B. Chazelle, "A minimum spanning tree algorithm with inverse-Ackermann type complexity," *Journal of the ACM*, vol. 47, pp. 1028–1047, 2000.

[41] M. Thorup, "On ram priority queues," in *Procs. of the 7th ACM-SIAM Symposium on Discrete Algorithms*, 1996, pp. 59–67.

[42] R. Tarjan, "Efficiency of a good but not linear set union algorithm," *Journal of the ACM*, vol. 22, pp. 215–225, 1975.

[43] J.Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: thinnings, topological watersheds and shortest path forests," 2008, submitted.

[44] M. Grimaud, "New measure of contrast: dynamics," in *Image algebra and morphological image processing III*, vol. SPIE-1769, 1992, pp. 292–305.

[45] G. Bertrand, "On the dynamics," *Image Vision Comput.*, vol. 25, no. 4, pp. 447–454, 2007.

[46] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1973.

[47] R. Deriche, "Using Canny's criteria to derive a recursively implemented

optimal edge detector," *The International Journal of Computer Vision*, vol. 1, no. 2, pp. 167–187, May 1987.
[48] P. Soille, *Morphological Image Analysis*. Springer-Verlag, 1999.
[49] P. K. Saha, J. K. Udupa, and D. Odhner, "Scale-based connected image segmentation: Theory, algorithms and validation," *Comput. Vis. Image Underst.*, vol. 77, pp. 145–174, 2000.
[50] F. Meyer and L. Najman, *Morphologie mathématique 1 : approches déterministes*, ser. Traité IC2, série signal et image. Hermès Science Publications, 2008, ch. Segmentation, arbre de poids minimum et hiérarchies, pp. 201–232.
[51] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 12, pp. 1163–1173, December 1996.
[52] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "On watershed cuts and thinnings," in *Discrete Geometry for Computer Imagery*, ser. LNCS, D. Coeurjolly, I. Sivignon, L. Tougne, and F. Dupont, Eds., vol. 4992. Springer, 2008, pp. 434–445.
[53] J. Serra, *Image Analysis and Mathematical Morphology*. Academic Press, 1988, vol. II: Theoretical Advances.
[54] J. Serra and L. Vincent, "An overview of morphological filtering," *Circuits Systems Signal Process*, vol. 11, no. 1, pp. 48–107, 1992.
[55] E. Breen and R. Jones, "Attribute openings, thinnings and granulometries," *Comput. Vis. Image Underst.*, vol. 64, no. 3, pp. 377–389, 1996.
[56] L. Najman and M. Couprie, "Building the component tree in quasi-linear time," *IEEE Trans. Image Processing*, vol. 15, no. 11, pp. 3531–3539, 2006.
[57] J. Cousty, L. Najman, G. Bertrand, and M. Couprie, "Minimum spanning tree by watershed," in preparation.
[58] S. Beucher, "Watershed, hierarchical segmentation and waterfall algorithm," in *Procs. of the second international conference on Mathematical Morphology and its Applications to Image Processing*, 1994, pp. 69–76.
[59] F. Meyer, "The dynamics of minima and contours," in *Mathematical Morphology and its Application to Image and Signal Processing*, 1996, pp. 329–336.

APPENDIX I
PROOFS

This appendix section provides the proofs of the properties given in this article.

A. Proof of Sec. II

The following lemma is a direct consequence of the definition of a minimum.

Lemma 16: Let $P \subseteq V, P \neq \emptyset$. If there is no vertex of $M(F)$ in P, then there exists an edge $u = \{x, y\}$ of G such that $x \in P$, $y \in V \setminus P$, and $F(u)$ is less than or equal to the altitude of any vertex in P.

Proof: [of Th. 6] To prove the theorem, we first show that if S is a basin-cut of F, then S is necessarily a watershed-cut of F. Afterward, we prove that if S is not a basin-cut of F, then S is not a watershed-cut of F. This will complete the proof. (i) Suppose that S is a basin-cut of F. Let $u = \{x_0, y_0\}$ be any edge in S. There exists $\pi_1 = \langle x_0, \ldots, x_\ell \rangle$ (resp. $\pi_2 = \langle y_0, \ldots, y_m \rangle$) a path of steepest descent from x_0 (resp. y_0) to $M(F)$. By definition of a cut, x_0 and y_0 are in two distinct connected components of $\overline{S}$. Thus, since $\overline{S}$ is an extension of $M(F)$, x_ℓ and x_m are necessarily in two distinct minima of F. Whenever π_1 (resp. π_2) is not trivial, by definition of a path of steepest descent, $F(\{x_0, x_1\}) = F^{\ominus}(x_0)$ (resp. $F(\{y_0, y_1\}) = F^{\ominus}(y_0)$). Hence, $F(\{x_0, x_1\}) \leq F(\{x_0, y_0\})$ (resp. $F(\{y_0, y_1\}) \leq F(\{x_0, y_0\})$). Hence, since by definition $\overline{S}$ is an extension of $M(F)$, S is a watershed-cut of F. (ii) Suppose now that S is not a basin-cut of F. If $\overline{S}$ is not an extension of $M(F)$, S is not a watershed of F. Suppose now that $\overline{S}$ is an extension of $M(F)$. Thus, there exists a point $x \in V$ such that there is no path of steepest descent in $\overline{S}$ from x to $M(F)$ (otherwise S would be a basin-cut of S). Let P be the set of all points of G that can be reached from x by a path of steepest descent in $\overline{S}$. By hypothesis, none of the points in P is a vertex of $M(F)$. We denote by T the set of all edges with minimal altitude among the edges $\{y, z\}$ such that $y \in P$, $z \in V \setminus P$. Let $v = \{y, z\} \in T$ such that $y \in P$. Since none of the vertices of P is a vertex of $M(F)$, from Lem. 16, we can deduce that $F^{\ominus}(y) = F(\{y, z\})$. Thus, there is, from x to z, a path in G, of steepest descent for F. Since z is not in P, there is no such path in $\overline{S}$. Thus, $v \in S$ and $T \subseteq S$. Again, let us consider $v = \{y, z\} \in T$. Let $\pi = \langle y_0 = y, \ldots, y_\ell \rangle$ be any descending path in $\overline{S}$ from y to $M(F)$. If such a path does not exist, then S is not a watershed: the proof is done. Suppose now that such a path exists. There exists $k \in [1, \ell]$ such that $y_{k-1} \in P$ and $y_k \in V \setminus P$. Since any edge in T is in S and since $\{y_{k-1}, y_k\}$ is in $\overline{S}$, $F(\{y_{k-1}, y_k\}) > F(v)$. Thus, as π is descending, $F(\{y_0, y_1\}) > F(v)$. Thus, the edge v, which belongs to S, does not satisfy the condition for the edges in a watershed: S is not a watershed. ■

B. Proofs of Sec. III

Before proving the properties of Sec. III, let us state the following propositions whose proofs are elementary.

Thanks to the construction presented in Sec. III-B, we can derive, from classical properties of trees, the following properties.

Let $X \subseteq G$, $u \in E(X)$. We write $X \backslash u$ for $(V(X), E(X) \backslash \{u\})$. Let $v \in E \backslash E(X)$. We write $X \cup v$ for the graph $(V(X) \cup v, E(X) \cup \{v\})$.

Lemma 17: Let X be a subgraph of G and let Y be a spanning forest relative to X. If for any $u \in E(Y) \setminus E(X)$ and $v \in E \setminus E(Y)$ such that $(Y \setminus u) \cup v$ is a spanning forest relative to X, we have $F(u) \leq F(v)$, then Y is an MSF relative to X.

Lemma 18: Let X be a subgraph of G and Y be a spanning forest relative to X. If $u = \{x, y\} \in E(Y) \setminus E(X)$, then there exists a unique component of $Y \setminus u$ which does not contain a component of X. Furthermore, either x or y is a vertex of this component.

Let $\pi = \langle x_0, \ldots, x_\ell \rangle$ be a path in G. We say that π is a *simple path* if for any two distinct i and j in $[0, \ell]$, $x_i \neq x_j$. We say that π is an *$\mathcal{M}$-path (for F)* if π is a *simple path*, if x_ℓ is a vertex of $M(F)$ and if none of $x_0, \ldots, x_{\ell-1}$ is a vertex of $M(F)$. Remark that an $\mathcal{M}$-path does not contain any edge of $M(F)$. Furthermore, it may be seen that if Y is a forest relative $M(F)$, there exists a unique $\mathcal{M}$-path from each vertex of Y.

Proof: [of Lem. 9] (i) Suppose that there exists x_0, a vertex of X such that there is no path from x_0 to $M(F)$, of steepest descent for F. We are going to prove that X is not an MSF relative to $M(F)$. Let $\pi = \langle x_0, \ldots, x_\ell \rangle$ be the unique $\mathcal{M}$-path from x_0 in X. Let $i \in [0, \ell - 1]$ be such that $\langle x_0, \ldots, x_i \rangle$ is a path of steepest descent for F and such that $\langle x_0, \ldots, x_{i+1} \rangle$ is not. We have: $F^{\ominus}(x_i) < F(\{x_i, x_{i+1}\})$. Let $Z = X \setminus \{x_i, x_{i+1}\}$. Since $\{x_i, x_{i+1}\}$ is not an edge of $M(F)$, from Lem. 18, there exists a unique connected component of Z, denoted by C, which does not contain a minimum of F. Furthermore, the vertex set of C does not contain any vertex of $M(F)$. Since π is an $\mathcal{M}$-path , hence a simple path, $\langle x_{i+1}, \ldots, x_\ell \rangle$ is a path in Z and x_ℓ is a vertex of $M(F)$. Thus, x_i is a vertex of C. From Lem. 16, we deduce that there exists $v = \{y, z\} \in E$ such that y is a vertex of C whereas z is not and $F(v) \leq F^{\ominus}(x_i)$. Thus, $F(v) < F(\{x_i, x_{i+1}\})$. By definition, we have $V(Z) = V(X) = V$. Hence, it may be seen that $Z \cup v$ is a spanning forest relative

to $M(F)$ whose weight is strictly less than the weight of X. Thus, X is not an MSF relative to $M(F)$.
(ii) Suppose that X is not an MSF relative to $M(F)$. We are going to prove that there exists $x \in V$ such that there is no path of steepest descent in X from x to $M(F)$. By the converse of Lem. 17, there exists $u \in E(X) \setminus E(M(F))$ and $v \in E \setminus E(X)$ such that $(X \setminus u) \cup v$ is a spanning forest relative to $M(F)$ and $F(v) < F(u)$. Let $X' = X \setminus u$. By Lem. 18, there exists a unique connected component of X', denoted by C, which does not contain any minimum of F. Since $X' \cup v$ is an extension of $M(F)$, there exists a unique vertex x in v which is a vertex of C. As $x \in v$, $F^{\ominus}(x) \leq F(v)$. Thus, $F(v) < F(u)$ implies $F^{\ominus}(x) < F^{\ominus}(u)$. Let π be the unique $\mathcal{M}$-path in X from x to $M(F)$. Since C does not contain any minimum of F, we deduce that π passes through u but $F^{\ominus}(x) < F^{\ominus}(u)$. Hence, π is not a path of steepest descent for F. ∎

The following lemmas will be used in the proof of Th. 10.

Lemma 19: Let $S \subseteq E$ be a watershed of F and $Y \subseteq \overline{S}$ be a forest relative to $M(F)$. If $V(Y) \neq V$, then there exists an edge $\{x, y\}$ in $\overline{S}$ outgoing from Y such that either $\langle x, y\rangle$ or $\langle y, x\rangle$ is a path of steepest descent for F. Furthermore, $Y \cup \{x, y\}$ is a forest relative to $M(F)$.

Proof: Since $V(Y) \neq V$, there exists $x_0 \in V \setminus V(Y)$. Since S is a watershed, by Th. 6, there exists, from x_0 to $M(F)$, a path $\pi = \langle x_0, \ldots, x_\ell\rangle$ in $\overline{S}$ of steepest descent for F. Since $M(F) \subseteq Y$, there exists $i \in [0, \ell - 1]$ such that $x_i \notin V(Y)$ and $x_{i+1} \in V(Y)$. Thus, $\{x_i, x_{i+1}\}$ is outgoing from Y. Furthermore, by the very definition of a path of steepest descent for F, $\langle x_i, x_{i+1}\rangle$ is a path of steepest descent for F.
Since $x_i \notin V(Y)$, any cycle in $Y \cup \{x_i, x_{i+1}\}$ is also a cycle in Y. Thus, by the very definition of a forest, it may be seen that $Y \cup \{x_i, x_{i+1}\}$ is a forest relative to Y, hence a forest relative to $M(F)$. ∎

The following lemma follows straightforwardly from the definition of a path of steepest descent.

Lemma 20: If $\langle x_0, \ldots, x_\ell\rangle$ and $\langle x_\ell, \ldots, x_m\rangle$ are two paths of steepest descent for F, then $\pi = \langle x_0, \ldots, x_m\rangle$ is a path of steepest descent for F.

Proof: [of Th. 10] (i) If S is a cut induced by an MSF relative to $M(F)$, then, by Lem. 9, there exists a path of steepest descent in $\overline{S}$ from each point in V to $M(F)$. Hence, by Th. 6, S is a watershed of F.
(ii) Suppose that S is a watershed of F. Let us consider a sequence of graphs $X_0, \ldots, X_k$ such that:
- $X_0 = M(F)$;
- $X_{i+1} = X_i \cup \{x_i, y_i\}$ where $\{x_i, y_i\}$ is an edge of $\overline{S}$ outgoing from X_i such that $\langle x_i, y_i\rangle$ is a path of steepest descent for F;
- X_k is such that there is no edge $\{x_k, y_k\}$ of $\overline{S}$ outgoing from X_k such that $\langle x_k, y_k\rangle$ is a path of steepest descent for F.
By induction on Lem. 19, X_k is a forest relative to $M(F)$. Furthermore, by the converse of Lem. 19, $V(X_k) = V$. Thus, X_k is a spanning forest relative to $M(F)$. From Lem. 20, it can be deduced by induction that for any $x \in V$ there exists, from x to $M(F)$, a path in X_k of steepest descent for F. Hence, by Lem. 9, X_k is an MSF relative to $M(F)$. Furthermore, since S is a cut and $X_k \subseteq \overline{S}$, it may be seen that S is the cut induced by X_k. ∎

Jean Cousty received his Ingénieur's degree from the École Supérieure d'Ingénieurs en Électrotechnique et Électronique (ESIEE Paris, France) in 2004 and the Ph.D. degree from the Université de Marne-la-Vallée (France) in 2007. After a one-year post-doctoral period in the ASCLEPIOS research team at INRIA (Sophia-Antipolis, France), he is now teaching and doing research with the Informatics Department, ESIEE Paris, and with the Institut Gaspard Monge, Université Paris-Est Marne-la-Vallée. His current research interests include medical image analysis and discrete mathematics.

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 Informatics Department, ESIEE, Paris, and with the Institut Gaspard Monge, Université Paris-Est Marne-la-Vallée. His research interests are image analysis, pattern recognition, mathematical morphology and digital topology.

Laurent Najman received the Habilitation à Diriger les Recherches in 2006 from University the University of Marne-la-Vallée, a Ph.D. of applied mathematics from Paris-Dauphine University in 1994 with the highest honour (Félicitations du Jury) and an "Ingénieur" degree from the Ecole des Mines de Paris in 1991. After earning his engineering degree, he worked in the central research laboratories of Thomson-CSF for three years, working on some problems of infrared image segmentation using mathematical morphology. He then joined a start-up company named Animation Science in 1995, as director of research and development. The technology of particle systems for computer graphics and scientific visualisation, developed by the company under his technical leadership received several awards, including the "European Information Technology Prize 1997" awarded by the European Commission (Esprit programme) and by the European Council for Applied Science and Engineering and the "Hottest Products of the Year 1996" awarded by the Computer Graphics World journal. In 1998, he joined OCÉ Print Logic Technologies, as senior scientist. He worked there on various problem of image analysis dedicated to scanning and printing. In 2002, he joined the Informatics Department of ESIEE, Paris, where he is professor and a member of the Institut Gaspard Monge, Université Paris-Est Marne-la-Vallée. His current research interest is discrete mathematical morphology.

Michel Couprie received his Ingénieur's degree from the Ecole Supérieure d'Ingénieurs en Electrotechnique et Electronique (Paris, France) in 1985, the Ph.D. degree from the Pierre et Marie Curie University (Paris, France) in 1988, and the Habilitation à Diriger des Recherches in 2004 from the University of Marne-la-Vallée (France). Since 1988 he has been working in ESIEE where he is a Professor. He is a member of the Informatics Department, ESIEE, Paris, and of Institut Gaspard Monge, Université Paris-Est Marne-la-Vallée. His current research interests include image analysis and discrete matematics.