On Watershed Cuts and Thinnings

Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie

Université Paris-Est, LABINFO-IGM, UMR CNRS 8049, A2SI-ESIEE, France
{j.cousty,g.bertrand,l.najman,m.couprie}@esiee.fr

Abstract. We recently introduced the watershed cuts, a notion of watershed in edge-weighted graphs. In this paper, we propose a new thinning paradigm to compute them. More precisely, we introduce a new transformation, called border thinning, that lowers the values of edges that match a simple local configuration until idempotence and prove the equivalence between the cuts obtained by this transformation and the watershed cuts of a map. We discuss the possibility of parallel algorithms based on this transformation and give a sequential implementation that runs in linear time whatever the range of the input map.

Introduction

The watershed transform introduced by Beucher and Lantuéjoul [1] for image segmentation is used as a fundamental step in many powerful segmentation procedures. Many approaches [2,3,4,5] have been proposed to define and/or compute the watershed of a vertex-weighted graph corresponding to a grayscale image. The digital image is seen 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 light areas.

In a recent paper [6], we investigate the watersheds in a different framework: we consider a graph whose edges are weighted by a cost function (see, for example, [7] and [8]). A watershed of a topographic surface may be thought of as a separating line-set from which a drop of water can flow down toward several minima. Following this intuitive idea, we introduce the definition of watershed cuts in edge weighted graphs. In [6], we establish the consistency (with respect to characterizations of the catchment basins and dividing lines) of watershed cuts, prove their optimality (in terms of minimum spanning forests) and introduce an algorithm to compute them.

Our main contribution in this paper consists of a new thinning paradigm to compute watershed cuts in linear time. More precisely, we propose a new transformation, called border thinning, that lowers the values of edges which match a simple local configuration until idempotence. The minima of the transformed map constitute a minimum spanning forest relative to the minima of the original one and, hence, induce a watershed cut. Moreover, any such minimum spanning forest can be obtained by this transformation. We discuss the possibility of parallel algorithms based on this transformation and give a sequential implementation

D. Coeurjolly et al. (Eds.): DGCI 2008, LNCS 4992, pp. 434–445, 2008.

which runs in linear time. We emphasize that this algorithm, contrarily to many watershed algorithms (see, for instance, [2,3]), does not require any sorting step, nor the use of any hierarchical queue. Therefore, it runs in linear time whatever the range of the input map.

The proofs of the properties presented in this paper are given in an extended version [9].

1 Basic Notions for Edge Weighted Graphs

We present some basic definitions to handle edge-weighted 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 $\pi = \langle x_0, \ldots, x_l \rangle$ 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 $i \in [1, l]$, x_i is adjacent to x_{i-1}. In this case, we say that *x_0 and x_l are linked for X*. If $l = 0$, 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 $E \neq \emptyset$.*

For applications to image segmentation, we may assume that V is the set of picture elements (pixels) and E is any of the usual adjacency relations.

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)*.

Let $S \subseteq E$. We denote by *$\overline{S}$ the complementary set of S in E*, *i.e.*, $\overline{S} = E \setminus S$. *The graph induced by S* is the graph whose edge set is S and whose vertex set is made of all points that belong to an edge in S, *i.e.*, $(\{x \in V \mid \exists u \in S, x \in u\}, S)$. In the following,the graph induced by S is also denoted by S.

We denote by $\mathcal{F}$ the set of all maps from E to $\mathbb{Z}$.

Let $F \in \mathcal{F}$. If u is an edge of G, $F(u)$ is the *altitude* of u. Let $X \subseteq G$ and $k \in \mathbb{Z}$. A subgraph X of G is a *(regional) minimum of F (at altitude k)* if: *i*) X is connected ; *ii*) k is the altitude of any edge of X; and *iii*) 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: *Throughout this paper, F denotes an element of $\mathcal{F}$.*

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). Thus, we suppose that the salient contours correspond to the highest edges of G.

2 Watersheds

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 domains of attraction of drops of water. In [6], we follow this intuitive idea to define the watersheds in an edge-weighted graph.

2.1 Extensions and Graph Cuts

We recall the notions of extension [5,6] 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.

Definition 1. *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.*

Let $S \subseteq E$. 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 subgraphs in Fig. 1b and c are two extensions of the one in Fig. 1a. The set S of dashed edges in Fig. 1c is a cut for X (Fig. 1a). It can be verified that $\overline{S}$ (bold subgraph in Fig. 1c) is an extension of X and that S is minimal for this property.

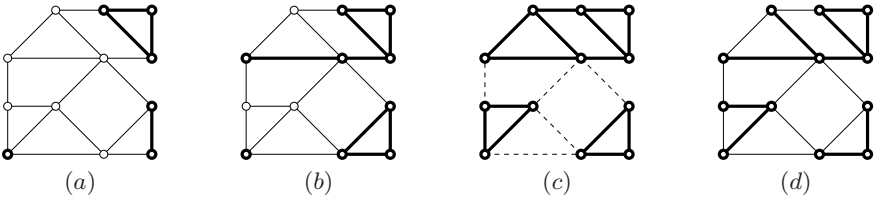

Fig. 1. A graph G. The set of vertices and edges represented in bold is: (a), a subgraph X of G; (b), an extension of X; (c), an extension Y of X which is maximal; and (d), a spanning forest relative to X. In (c), the set of dashed edges is a cut for X.

The notion of graph cut has been studied for many years in graph theory. For applications to image segmentation, a classical problem is to find a cut of minimum weight (a min-cut) for a set of terminal points. The links between these approaches and the one developed in this paper are investigated in [10].

2.2 Watersheds

We now recall the definition of a watershed cut in an edge-weighted graph. 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 $\pi = \langle x_0, \ldots, x_l \rangle$ be a path in G. The path π is *descending (for F)* if, for any $i \in [1, l-1]$, $F(\{x_{i-1}, x_i\}) \geq F(\{x_i, x_{i+1}\})$.

Definition 2 (drop of water principle, Def. 3 in [6]). *Let $S \subseteq E$. We say that S* is a watershed cut (or simply a watershed) of 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.

In order to illustrate the previous definition, it may be seen that the set S of dashed edges in Fig. 2b is a watershed of the corresponding map F. The minima of F are depicted in bold in Fig. 2a.

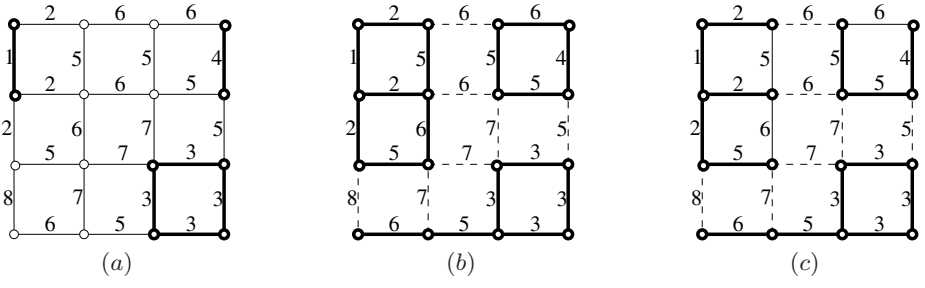

Fig. 2. A graph G and a map F. Edges and vertices in bold depict: (a), the minima of F; (b), an extension of $M(F)$ which is equal to $\overline{S}$; and (c), a MSF relative to $M(F)$. In (b) and (c), the set of dashed edges is a watershed S of F.

Let $S \subseteq E$. We remark that if S is a watershed of F, then S is necessarily a cut for $M(F)$. The converse is in general not true since 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.

A popular alternative to the drop of water principle defines a watershed exclusively by its catchment basins and does not involve any property of the divide.

In the framework of edge-weighted graph, we define a *catchment basin* as a component of the complementary of a watershed.

The following theorem (Th. 3) shows that a watershed cut can be defined equivalently by its divide line or by its catchment basins. For that purpose, we start with some definitions relative to the notion of path with steepest descent.

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

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

Theorem 3 (consistency, Th.1 in [6]). *Let $S \subseteq E$ be a cut for $M(F)$. The set S is a watershed of F if and only if, from each point of V to $M(F)$, there exists a path in the graph induced by $\overline{S}$ which is a path with steepest descent for F.*

Th. 3 establishes the consistency of watershed cuts: 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) that separates them.

3 Minimum Spanning Forests and Watershed Optimality

In this section, we recall the definition of a minimum spanning forest relative to a subgraph of G. We also recall the equivalence between watershed cuts and cuts induced by minimum spanning forests relative to the minima.

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 $V(Y) = V$.

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

Let $X \subseteq G$, *the weight of X (for F)* is the value $F(X) = \sum_{u \in E(X)} F(u)$.

Definition 4. *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.*

For instance, the graph Y (bold edges and vertices) in Figs. 2c is a MSF relative to X (Fig. 2a).

We now have the mathematical tools to state the optimality of watershed cuts (Th. 2 in [6]).

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 and this cut is also a cut for X. We say that this unique cut is the *cut induced by Y*. Furthermore, if Y is a MSF relative to X, we say that that S is a *MSF cut for X*.

Theorem 5 (optimality, Th. 2 in [6]). *Let $S \subseteq E$. The set S is a MSF cut for $M(F)$ if and only if S is a watershed cut of F.*

The minimum spanning tree problem is one of the most typical and well-known problems of combinatorial optimization (see [11]). In [9,6] (see also [7]), we show that the minimum spanning tree problem is equivalent to the problem of finding a MSF relative to a subgraph of G. A direct consequence is that any minimum spanning tree algorithm can be used to compute a relative MSF. Many efficient algorithms (see [11]) exist in the literature for solving the minimum spanning tree problem.

4 Optimal Thinnings

As seen in the previous section, a MSF relative to a subgraph of G 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 [12]. In this section, we introduce a new paradigm to compute MSFs relative to the minima of a map and obtain a linear time algorithm to solve this particular instance of the MSF problem. To this aim, we define a new thinning transformation that iteratively lowers the values of the edges that satisfy a simple local property. The minima of the transformed map constitute precisely a MSF relative to the minima of the original one. More remarkably, any MSF relative to the minima of a map can be obtained by this transformation. We discuss the possibility of parallel algorithms based on this transformation and give a sequential implementation (Algo. 1) which runs in linear time.

4.1 Border Thinnings and Watersheds

We introduce an edge classification based exclusively on local properties, *i.e.*, properties that depend only on the adjacent edges. This classification will be used in the definition of a lowering process (Def. 7) which allows one to extract the watersheds of a map.

Remind that, if x is a vertex of G, $F(x)$ is the minimal altitude of an edge that contains x.

Definition 6. *Let $u = \{x, y\} \in E$.*
We say that u is locally separating (for F) *if $F(u) > \max(F(x), F(y))$.*
We say that u is border (for F) *if $F(u) = \max(F(x), F(y))$ and $F(u) > \min(F(x), F(y))$.*
We say that u is inner (for F) *if $F(x) = F(y) = F(u)$.*

Notice that a notion similar to the one of border edge has been proposed in [13] under the name of *min-contractible edge*.

Fig. 3 illustrates the above definitions. In Fig. 4a, $\{j, n\}$, $\{a, e\}$ and $\{b, c\}$ are examples of border edges; $\{i, m\}$ and $\{k, l\}$ are inner edges and both $\{h, l\}$ and $\{g, k\}$ are locally-separating edges. Note that any edge of G corresponds exactly to one of the types presented in Def. 6.

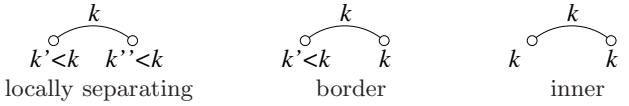

Fig. 3. Illustration of the different local configurations for edges

Let $u \in E$. The *lowering of* F *at* u is the map F' in $\mathcal{F}$ such that:
- $F'(u) = \min_{x \in u}\{F(x)\}$; and
- $F'(v) = F(v)$ for any edge $v \in E \setminus \{u\}$.

Definition 7 (border cut). *Let $H \in \mathcal{F}$. We say that H is a* border thinning of F *if:*
i) $H = F$; or
ii) there exists $J \in \mathcal{F}$ a border thinning of F such that H is the lowering of J at a border edge for J.
If there is no border edge for H, we say that H is a border kernel. *If H is a border thinning of F and if it is a border kernel, then H is a* border kernel of F. *If H is a border kernel of F, any cut for $M(H)$ is called* a border cut for F.

To illustrate the previous definition, we assume that F (resp. H, J) is the map of Fig. 4a (resp. b,c). The maps H and J are border thinnings of F. The map J is a border kernel of both F and H. The function depicted in Fig. 4d is another border kernel of F which is not a border kernel of H. In Fig. 4c and d, the border cuts are represented by dashed edges. Remark that the minima of the two border kernels constitute forests relative to $M(F)$, and that all edges which do not belong to the bold graphs are locally separating.

In the next subsection, we show the equivalence between border cuts and watersheds. Thus, the border thinnings constitute a new approach to compute a watershed. This approach is particularly interesting since it relies only on local conditions and produces a result that is globally optimal. It may be noticed that the optimality of the result does not depend on the order on which the values of the edges are lowered. In particular, if a set of mutually-disjoint border edges is lowered in parallel, then the resulting map is a border thinning. Consequently, this transformation suggests new strategies toward efficient parallel algorithms to compute a watershed.

4.2 Linear-Time Algorithm for Border Cut

On a sequential computer, a naive algorithm to obtain a border kernel could be the following: *i*) for all $u = \{x, y\}$ of G, taken in an arbitrary order, check the values of $F(u)$, $F(x)$ and $F(y)$ and whenever u is border, lower the value of u down to the minimum of $F(x)$ and $F(y)$; *ii*) repeat step *i*) until no border edge remains. Consider the graph G whose vertex set is $\{0, \ldots, n\}$ and whose edge set is made of all the pairs $u_i = \{i, i+1\}$ such that $i \in [0, n-1]$. Let $F(u_i) = n - i$, for all $i \in [0, n-1]$. On this graph, if the edges are processed in the order of their indices, step *i*) will be repeated exactly $|E|$ times. The cost for checking all

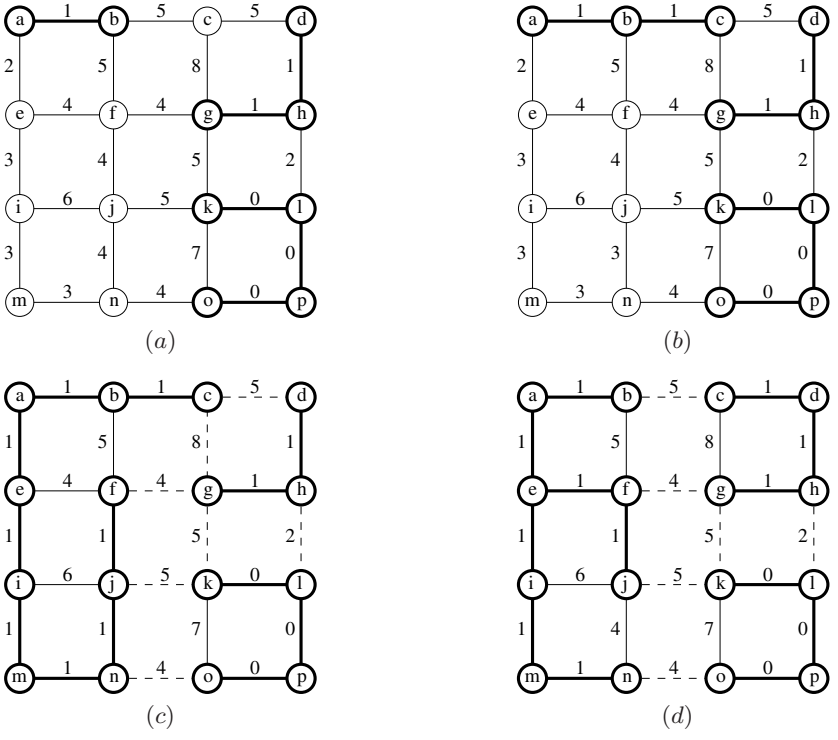

Fig. 4. A graph and some associated functions. The bold graphs superimposed are the minima of the corresponding functions; (*b*), a border thinning of (*a*); (*c*): a border kernel of both (*a*) and (*b*); and (*d*), another border kernel of (*a*). In (*c*) and (*d*), the border cuts are represented by dashed edges.

edges of G is $O(|E|)$. Thus, the worst case time complexity of this naive algorithm is $O(|E|^2)$. In order to reduce the complexity, we introduce a second lowering process in which any edge is lowered at most once. This process is a particular case of border thinning that also produces, when iterated until stability, a border kernel of the original map. Thanks to this second thinning strategy, we derive a linear-time algorithm to compute border kernels and, thus, watersheds.

It may be seen that an edge which is in a minimum at a given step of a border thinning sequence never becomes a border edge later. Thus, lowering first the edges adjacent to the minima seems to be a promising strategy.

Definition 8 (M-border cut). *We say that an edge u in E is* minimum-border (for F), written M-border*, if u is border for F and if exactly one of the vertices in u is a vertex of $M(F)$.*
Let $H \in \mathcal{F}$. We say that H is an M-border thinning of F *if:*
i) $H = F$*; or*
ii) *there exists $J \in \mathcal{F}$ an M-border thinning of F such that H is the lowering of J at an M-border edge for J.*

If there is no M-border edge for H, we say that H is an M-border kernel. *If H is an M-border thinning of F and if it is an M-border kernel, we say that H is an* M-border kernel of F.
If H is an M-border kernel of F, any cut for $M(H)$ is called an M-border cut for F.

In Fig. 4a, the edge $\{c, d\}$ is M-border whereas $\{j, n\}$ is border but not M-border.

On an edge-weighted graph, the classical algorithm by flooding (or immersion) [3] could be described as follows: i) mark the minima with distinct labels; ii) mark the lowest edge containing exactly one labelled vertex with this label; and iii) repeat step ii) until idempotence. In fact, the M-border thinning transformation (which is itself a particular case of border thinning) generalizes immersion algorithms on edges. Indeed, it is proved [9] that any edge of minimal altitude among the edges outgoing from $M(F)$ is an M-border edge.

Definition 9 (I-cut). *If u is an edge with minimal altitude among all the edges outgoing from $M(F)$, then u is an* immersion edge for F.

Let $H \in \mathcal{F}$. We say that H is an I-thinning of F *if $H = F$ or if there exists a I-thinning J of F such that H is the lowering of J at an immersion edge for J. If there is no immersion edge for H, we say that H is an* I-kernel. *If H is an I-thinning of F and if it is an I-kernel, we say that H is an* I-kernel of F. *In this case, any cut for $M(H)$ is called a* I-cut for F.

Let $H \in \mathcal{F}$. We say that $M(H)$ is the *min-graph of H*. This notion will be used in the following property which states that the min-graphs of border, M-border and I- kernels of F are MSFs relative to $M(F)$. More remarkably, any MSF relative to $M(F)$ can be obtained as the min-graph of an M-border kernel, of a flooding kernel and of an I-kernel of F.

Lemma 10. *Let $X \subseteq G$. The four following statements are equivalent:*
(i) X is the min-graph of an I-kernel of F;
(ii) X is the min-graph of an M-border kernel of F;
(iii) X is the min-graph of a border kernel of F; and
(iv) X is a MSF relative to $M(F)$.

Since a relative MSF induces a unique graph cut, from the previous lemma, we immediately deduce that an I-kernel (resp. an M-border kernel, a border kernel) of a map defines a I-cut (resp. M-border cut, border cut). Hence, the following theorem which states the equivalence between watershed cuts, border cuts, M-border cuts and I-cuts can be easily proved.

Theorem 11. *Let $S \subseteq E$. The four following statements are equivalent:*
(i) S is an I-cut for F;
(ii) S is an M-border cut for F;
(iii) S is a border cut for F; and
(iv) S is a watershed cut for F.

Using the notions introduced in this section, we derive Algo. 1, an efficient algorithm to compute M-border kernels, hence watershed cuts. We recall that an

Algorithm 1. M-Border

```
   Data: (V, E, F): an edge-weighted graph;
   Result: F, an M-border kernel of the input map and M its minima.
 1 L ← ∅ ;
 2 Compute M(F) = (V_M, E_M) and F(x) for each x ∈ V;
 3 foreach u ∈ E outgoing from (V_M, E_M) do L ← L ∪ {u} ;
 4 while there exists u ∈ L do
 5     L ← L \ {u} ;
 6     if u is border for F then
 7         x ← the vertex in u such that F(x) < F(u) ;
 8         y ← the vertex in u such that F(y) = F(u) ;
 9         F(u) ← F(x) ; F(y) ← F(u) ;
10         V_M ← V_M ∪ {y} ; E_M ← E_M ∪ {u} ;
11         foreach v = {y', y} ∈ E such that y' ∉ V_M do L ← L ∪ {v} ;
```

edge u is border for F if the altitude of one of its extremities equals the altitude of u and the altitude of the other is strictly less than the altitude of u.

In order to achieve a linear complexity, the graph G can be stored as an array of lists which maps to each point the list of all its adjacent vertices. An additional mapping can be used to access in constant time the two vertices which compose a given edge. Nevertheless, for applications to image processing, and when usual adjacency relations are used, these structures do not need to be explicit.
Furthermore, to achieve a linear complexity, the minima of F must be known at each iteration. To this aim, in a first step (line 2), the minima of F are computed and represented by two Boolean arrays V_M and E_M, the size of which are respectively $|V|$ and $|E|$. This step can be performed in linear time thanks to classical algorithms. Then, in the main loop (line 4), after each lowering of F (line 9), V_M and E_M are updated (line 10). In order to access, in constant time, the edges which are M-border, the (non-already examined) edges outgoing from the minima are stored in a set L (lines 3 and 11). This set can be, for instance, implemented as a queue. Thus, we obtain the following property.

Property 12. *At the end of Algorithm 1, F is an M-border kernel of the input function F. Furthermore, Algorithm 1 terminates in linear time with respect to $|E|$.*

We emphasize that Algo. 1 does not require any sorting step nor the use of any hierarchical queue. Thus, whatever the range of the considered map, it runs in linear time with respect to the size of the input graph. Furthermore, even in the case where the range of the map is rather limited ($[0, 255]$) the proposed algorithm runs about two times faster than the watershed by flooding algorithm [3] applied on vertex-weighted graph (see the experiments in [9]).

It may be also noticed that in Algo. 1, if the set L is implemented as a hierarchical queue [3], then the resulting map is still a border kernel. Furthermore, in this case, the induced cut is "centered" (according to the distance induced

(*a*) (*b*) (*c*)

Fig. 5. Results (superimposed in black to the original image), obtained by applying a watershed on (*a*) an edge-weighted graph and (*b*, *c*) a vertex-weighted graph. In (*a*), we first assign the absolute difference of intensities to each edge and then filter the map before aplying the weatershed cut. In (*b*) (resp. (*c*)), we first assign the result of the Deriche edge detector (resp. morphological gradient) to each vertex of the graph and then filter the map before aplying a watershed. In each case, a filtering step, based on area closing [14], ensures that the watershed defines exactly 12 catchment basins (see [15] for implementation details).

by G) on plateaus. On the other hand, this version of Algo. 1 runs in linear time only if the number of distinct values of F is sufficiently small.

Conclusion and Perspectives

In this paper, we study the watersheds in edge-weighted graphs. Fig. 5 preliminary illustrates that, for some images, these watersheds produce a better delineation (see the man's helmet) than the watersheds in vertex-weighted graphs[1]. In this framework of edge weighted graphs, we introduce new thinning transformations and prove the equivalence between the produced cuts and the watershed cuts. Based on these transformations, we derive a watershed algorithm that runs in linear time whatever the range of the input map. For more details on watershed cuts, we refer to [9]. In particular in [9], we show that a watershed cut is a separation which corresponds to a separation produced by a topological watershed [4, 5] defined on edge-weighted graphs. Furthermore, we study the links with shortest-path forests [8].

Future work will be focused on hierarchical segmentation schemes based on watershed cuts (including *geodesic saliency of watershed contours* [16] and *incremental MSFs*) as well as on watershed in weighted simplicial complexes, an image representation adapted to the study of topological properties. Based on border thinnings, we also expect to propose efficient parallel algorithms for watershed cuts. Furthermore, we intend to show that our watershed algorithm can be used to efficiently compute minimum spanning trees.

[1] We verified, on the image of Fig. 5, that similar results are obtained by different approaches [3, 4] to watersheds in vertex-weighted graphs.

References

1. Beucher, S., Lantuéjoul, C.: Use of watersheds in contour detection. In: Procs. of the International Workshop on Image Processing Real-Time Edge and Motion Detection/Estimation (1979)
2. Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Trans. PAMI 13(6), 583–598 (1991)
3. Meyer, F.: Un algorithme optimal de ligne de partage des eaux. In: Procs. of 8ème Congrès AFCET, Lyon-Villeurbanne, France, pp. 847–859 (1991)
4. Couprie, M., Bertrand, G.: Topological grayscale watershed transform. Procs. of SPIE Vision Geometry V 3168, 136–146 (1997)
5. Bertrand, G.: On topological watersheds. JMIV 22(2-3), 217–230 (2005)
6. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts. In: Procs. ISMM., pp. 301–312 (2007)
7. Meyer, F.: Minimum spanning forests for morphological segmentation. In: Procs. ISMM., pp. 77–84 (1994)
8. Falcão, A.X., Stolfi, J., de Alencar Lotufo, R.: The image foresting transform: theory, algorithm and applications. IEEE Trans. PAMI 26, 19–29 (2004)
9. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watersheds, minimum spanning forest and the drop of water principle, Technical report IGM-2007-01 (Submitted, 2007), `http://igm.univ-mlv.fr/LabInfo/rapportsInternes/2007/01.pdf`
10. Alléne, C., Audibert, J.Y., Couprie, M., Cousty, J., Keriven, R.: Some links between min-cuts, optimal spanning forests and watersheds. In: Procs. ISMM., pp. 253–264 (2007)
11. Cormen, T.H., Leiserson, C., Rivest, R.: Introduction to algorithms, 2nd edn. MIT Press, Cambridge (2001)
12. Chazelle, B.: A minimum spanning tree algorithm with inverse-Ackermann type complexity. Journal of the ACM 47, 1028–1047 (2000)
13. Englert, R., Kropatsch, W.: Image structure from monotonic dual graph. In: Münch, M., Nagl, M. (eds.) AGTIVE 1999. LNCS, vol. 1779, pp. 297–308. Springer, Heidelberg (2000)
14. Serra, J., Vincent, L.: An overview of morphological filtering. Circuits Systems Signal Process 11(1), 48–107 (1992)
15. Najman, L., Couprie, M.: Building the component tree in quasi-linear time. IEEE Trans. Image Processing 15(11), 3531–3539 (2006)
16. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. IEEE Trans. PAMI 18(12), 1163–1173 (1996)