Watershed cuts

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 study the watersheds in edge-weighted graphs. Contrarily to
previous works, we define the watersheds following the intuitive idea
of drops of water flowing on a topographic surface. We establish
the consistency (with respect to characterizations of the catchment
basins and dividing lines) of these watersheds, prove their optimal-
ity (in terms of minimum spanning forests) and derive a linear-time
algorithm. To the best of our knowledge, similar properties are not
verified in other frameworks and the proposed algorithm is the most
efficient existing algorithm.

Keywords: watershed, catchment basin, minimum spanning forest.

Introduction

The watershed transform introduced by Beucher and Lantuéjoul [3] for im-
age segmentation is used as a fundamental step in many powerful segmenta-
tion procedures. Many approaches [2, 6, 9, 14] 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 this paper, we investigate the watershed in a different framework: we
consider a graph whose edges are weighted by a cost function (see, for ex-
ample, [10] and [8]). A watershed of a topographic surface may be thought
of as a separating line-set on which a drop of water can flow down toward
several minima. To formalize this intuitive idea, we introduce the *drop
of water principle* that leads to the definition of *watershed cuts* in edge-
weighted graphs.
We establish 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 catch-
ment basins (through the drop of water principle). As far as we know,
in other frameworks, no similar property has ever been proved and some
counter-examples showing that such a duality does not hold can be found.
We also establish the optimality of watershed cuts. In [10], F. Meyer
shows the link between minimum spanning forests (MSF) and flooding from

marker algorithms. In this paper, we prove the equivalence between water-
shed cuts and separations induced by minimum spanning forests relative to
the minima.

Finally, we propose a linear-time algorithm which does not require any sort-
ing step, nor the use of any hierarchical queue, nor the extraction of the
minima in a preprocessing step. This algorithm therefore runs in linear
time whatever the range of the input map. To the best of our knowledge,
this is the first watershed algorithm with such a property.

The proofs of the properties presented in this paper are given in an
extended version [7].

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 $Z = Y$.

*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 $\{x, y\} \in E$ 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 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 which 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.

In the sequel of 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. Thus, we suppose that salient contours are located on 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 domain of attraction of drops of water. Despite its simplicity, none of the classical definitions handle this intuitive idea. In this paper, contrarily to previous works, we follow the drop of water principle to define a watershed in an edge-weighted graph.

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 [2], G. Bertrand formalizes the notion of extension.

Definition 1 (from Def. 12 in [2]). 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 subgraphs in Figures 1(b) and 1(c) are two extensions of the one in Figure 1(a).

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 cover all the vertices of the graph. The separation which is thus produced is called a graph cut.

Definition 2. 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 $T = S$. □
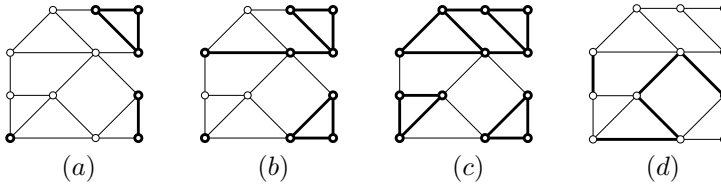
Figure 1. A graph G. The set of vertices and edges represented in bold is: (a), a subgraph X of G; (b), an extension of X; (c): an extension Y of X which is maximal; and (d): a cut S for X such that $\overline{S} = Y$.

The set S depicted in Figure 1(d) is a cut for X (Figure 1(a)). It can be verified that $\overline{S}$ (Figure 1(c)) 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 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 [1].

Watersheds by the drop of water principle

We 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 $\pi = \langle x_0, \ldots, x_l \rangle$ 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 3 (drop of water principle). 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 Figure 2(b) is a watershed of the corresponding map F. The minima of F are depicted in bold in Figure 2(a).

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.
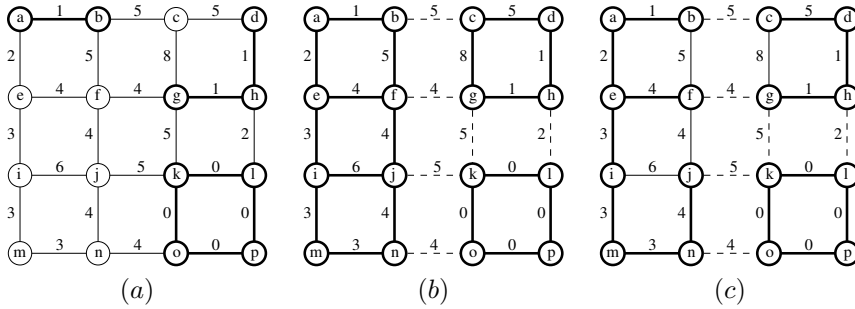
Figure 2. A graph G and a map F. Edges and vertices in bold depict: (a), the minima of F; (b), a maximal extension of $M(F)$; and (c), a MSF relative to $M(F)$. Dashed edges in (b) and (c) depict a watershed of F.

Catchment basins by a steepest descent property

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 graph induced by the complementary set of a watershed.

The following theorem (Theorem 1) shows that a watershed 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.

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

The previous theorem establishes the consistency of watershed cuts: they can be equivalently defined by a steepest descent property on the catchment basins (regions) or by a the drop of water principle on the cut (frontier) which separates them. As far as we know, in the literature about discrete watersheds, no similar property ([11]) has ever been proved and some counter-examples showing that such a duality does not hold in other frameworks can be found. Hence, Theorem 1 emphasizes that edge-weighted graphs are adapted for the definition and study of watersheds.
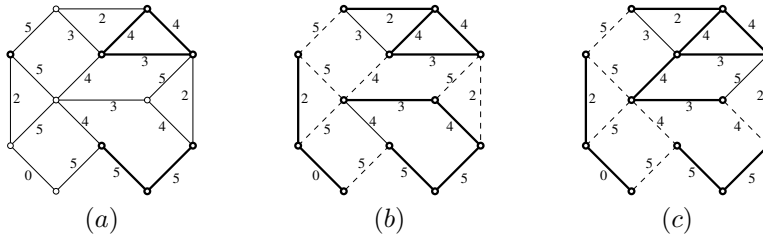
Figure 3. A graph G and a map F. The bold edges and vertices represent: (a), X a subgraph of G; (b) and (c), two MSFs relative to X; their induced cuts are represented by dashed edges.

3. Minimum spanning forests and watersheds

We introduce the minimum spanning forests relative to subgraphs of G and show the equivalence between the watershed cuts of a map and the cuts induced by minimum spanning forests relative to the minima of this map.

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 $V(Y) = V$.

Thanks to the notion of relative forest, the usual notions of a tree and 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 $X \subseteq G$, *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 graphs Y and Z (bold edges and vertices) in Figures 3(b) and 3(c) are two MSFs relative to X (Figure 3(a)).

We now have the mathematical tools to present the main result of this section (Theorem 2) which establishes the optimality of watersheds.

Let X be a subgraph of G and let Y be a spanning forest relative to X. There exists a unique cut for Y and this cut is also a cut for X. We say that this unique cut is the *cut induced by Y*.

Theorem 2 (optimality)**.** *Let $S \subseteq E$. The set S is a cut induced by a MSF relative to $M(F)$ if and only if S is a watershed cut of F.*

A MSF relative to the minima of the map Figure 2 is depicted in bold in Figure 2(c). Observe that the induced cut is indeed a watershed.

As far as we know, this is the first result which establishes watershed optimality in graphs. Furthermore, Theorem 2 suggests that MSF is a method of choice for marker based watershed, an illustration of which is given in [7].

The minimum spanning tree problem is one of the most typical and well-known problems of combinatorial optimization (see [5]). In the next paragraph, we show that the minimum spanning tree problem is equivalent to the problem of finding a 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 a MSF relative to the subgraph $(\{x\}, \emptyset)$, x being any vertex of X.

In order to recover the link between flooding algorithms and minimum spanning tree algorithms, in [10], F. Meyer proposed a construction which shows the equivalence between finding a MSF rooted in a set of vertices (*i.e.*, a MSF relative to a subgraph X of G such that $E(X) = \emptyset$) and finding a minimum spanning tree. This construction can be easily extended for proving the equivalence between finding a minimum spanning tree and a MSF relative 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 graph are linked by multiple edges only the one with minimal value is kept.

A direct consequence is that any minimum spanning tree algorithm can be used to compute a relative MSF. Many efficient algorithms (see a survey in [5]) exist in the literature for solving the minimum spanning tree problem.

4. 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 [4]. In this section, we introduce a linear-time watershed algorithm. This algorithm 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, 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.

We first introduce the mathematical tools which allow us to prove the correctness of the proposed algorithm. In particular, we propose a notion of stream which is crucial to this paradigm. Then, the algorithm is presented, and both its correctness and complexity are analyzed.

Definition 5. 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,

with 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 $y \in L$. □

We 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, with 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 Figure 2. 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, with
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 with
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 $\prec$-*stream* if there is no stream under L.

In Figure 2(a) 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 $\prec$-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.

Proposition 1. *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, $F(\{x, y\}) > F(x)$.

In Figure 2(a) 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.

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 $\mathcal{L} = \{L_1, \ldots, L_n\}$ be a set of n $\prec$-streams. We say that $\mathcal{L}$ is a *flow
family* if: *i*) $\cup\{L_i \mid i \in \{1, \ldots, n\}\} = V$; and *ii*) for any two distinct L
and L' in $\mathcal{L}$, if $L \cap L' \neq \emptyset$, then $L \cap L'$ is the vertex set of a minimum of F.

Let $\mathcal{L}$ be a flow family and let $x \in V$. It may be seen that, either x
belongs to a minimum of F (in this case, it may belong to several elements
of $\mathcal{L}$), or x belongs to a unique $\prec$-stream of $\mathcal{L}$ which itself contains the

vertex set of a unique minimum of F. Thus, thanks to $\mathcal{L}$, we can associate to each vertex x of G a unique minimum of F.

Definition 6. 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 $\{1, \ldots, n\}$ 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 is 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 for F*. □

The next proposed algorithm produces a flow mapping, and hence a flow cut. The following theorem, which states the equivalence between flow cuts and watersheds, is the main tool to establish the correctness of Algorithm 1.

Theorem 3. *Let $S \subseteq E$. The set S is a watershed of F if and only if S is a flow cut for F.*

We now present Algorithm 1 which computes a flow mapping, hence, by Theorem 3, a watershed. It 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' under L and which is already labeled. In this case, the points of L receive the label of L' (Line 9). The function `Stream`, called at Line 4, computes the stream L. Roughly speaking, it performs an intermixed sequence of depth-first and breadth-first exploration of the paths with 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 Proposition 1, 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 with steepest descent. Thus, there is an $\prec$-stream L', under L, included in the set of all vertices labeled *lab*. Thus, by the preceding remarks, the output of Algorithm 1 is a flow mapping of F. Furthermore, using classical data structures to represent the graph G, we obtain a linear complexity.

Proposition 2. *Algorithm 1 outputs a map ψ which is a flow mapping of F. Furthermore, Algorithm 1 runs in linear-time with respect to $|E|$.*

5. Illustration

In order to illustrate the use of watershed cuts in practical applications, we derive, from the classical framework of mathematical morphology, a segmentation scheme which consists in the three following steps: (i), computation

---

Algorithm 1: Flow mapping.

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

1 **foreach** $x \in V$ **do** $\psi(x) := NO\_LABEL$;
2 $nb\_labs := 0$; /* the number of minima already found */
3 **foreach** $x \in V$ *such that* $\psi(x) = NO\_LABEL$ **do**
4 $\quad [L, lab] := \texttt{Stream}(V, E, F, \psi, x)$;
5 $\quad$ **if** $lab = -1$ **then** /* L is an $\prec$-stream */
6 $\quad\quad nb\_labs + +$;
7 $\quad\quad$ **foreach** $y \in L$ **do** $\psi(y) := nb\_labs$;
8 $\quad$ **else**
9 $\quad\quad$ **foreach** $y \in L$ **do** $\psi(y) := lab$;

---

of a simple function that assigns a weight to the edges of the 4-adjacency graph associated to the image; (ii), filtering of this cost function in order to reduce the number of minima; and (iii) computation of a watershed of the filtered cost function.

We assume that the graph G is the one corresponding to the 4-adjacency relation associated to the image I which is to be segmented. We consider also the map F defined for any $\{x, y\} \in E$ by $F(\{x, y\}) = |I(x) - I(y)|$.

A watershed of F would contain too many catchment basins. In order to suppress many of the non-significant minima, a classical approach consists of computing morphological filtering of the function [13]. For this illustration, we implement an adaptation of a classical filter which consists of: (i) remove the connected component of a lower threshold of F with minimal area; (ii), repeat step (i) until F has k (a predefined number) minima. Hence, the watershed of the filtered map contains exactly k catchment basins. The results obtained on the cameraman image ($k = 22$) are presented in Figure 4.

Conclusion and perspectives

In this paper, we introduce watershed cuts, a notion of watershed in edge-weighted graphs. We show the consistency and optimality of watershed cuts. Furthermore, we derive a simple algorithm which runs in linear-time whatever the range of the input map. For more details on watershed cuts, we refer to [7]. In particular in [7], we show that a watershed cut is a separation which corresponds to a separation produced by a topological watershed [2, 6] defined on edge-weighted graphs. We also study the links with shortest-path forests [8].

Further work will be focused on hierarchical segmentation schemes based on watershed cuts (including *geodesic saliency of watershed contours* [12]

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 a 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 $y \in L'$* **do**

4 $L' := L' \setminus \{y\}$;

5 $breadth\_first := TRUE$;

6   **while** *(breadth_first) and (there exists $\{y, z\} \in E$ such that $z \notin L$ and $F(\{y, z\}) = F(y)$)* **do**

7   **if** $\psi(z) \neq NO\_LABEL$ **then**

8   /* there is an $\prec$-stream under L already labelled */

9   return $[L, \psi(z)]$;

10   **else if** $F(z) < F(y)$ **then**

11 $L := L \cup \{z\}$; /* z is now the only bottom of L */

12 $L' := \{z\}$; /* hence, switch to depth-first exploration */

13 $breadth\_first := FALSE$;

14   **else**

15 $L := L \cup \{z\}$; /* $F(z) = F(y)$, thus z is also a bottom of L */

16 $L' := L' \cup \{z\}$; /* continue breadth-first exploration */

17 return $[L, -1]$;

---

and *incremental MSFs*) as well as on watershed in weighted simplicial complexes, an image representation adapted to the study of topological properties. Furthermore, we intend to show that our watershed algorithm can be used to efficiently compute minimum spanning trees.

References

[1] C. Allène, J. Y. Audibert, M. Couprie, J. Cousty, and R. Keriven, *Some links between min cuts, optimal spanning forests and watersheds*, Procs. 8th ISMM, 2007. These proceedings.

[2] G. Bertrand, *On topological watersheds*, JMIV **22** (2005), no. 2-3, 217–230.

[3] S. Beucher and C. Lantuéjoul, *Use of watersheds in contour detection*, Procs. of the international workshop on image processing real-time edge and motion detection/estimation, 1979.

[4] B. Chazelle, *A minimum spanning tree algorithm with inverse-Ackermann type complexity*, Journal of the ACM **47** (2000), 1028–1047.

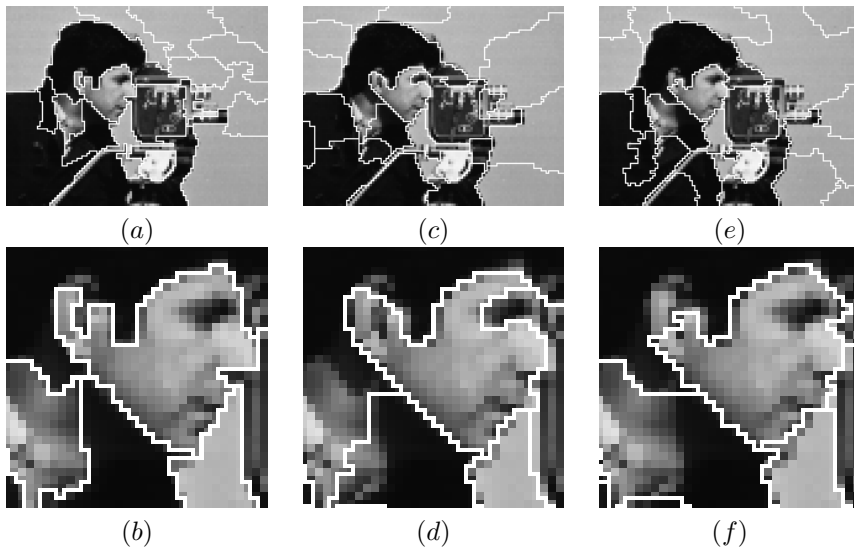[5] T. H. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms, second edition*, MIT Press, 2001.

Figure 4. Results obtained by applying a grayscale watershed on a filtered map [see text]. (a, b) A watershed cut superimposed in white to the original image I; (c, d) a watershed by flooding of the filtered Deriche optimal edge detector; (e, f) a watershed by flooding of a filtered morphological gradient. In the three cases, a similar procedure based on area filtering is used to ensure that the watershed has exactly 22 catchment basins.

[6] M. Couprie and G. Bertrand, *Topological grayscale watershed transform*, Procs. of spie vision geometry v, 1997, pp. 136–146.

[7] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, *Watersheds, minimum spanning forests and the drop of water principle*, Technical report IGM2007-01 (2007).

[8] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo, *The image foresting transform: theory, algorithm and applications*, IEEE Trans. PAMI **26** (2004), 19–29.

[9] F. Meyer, *Un algorithme optimal de ligne de partage des eaux*, Procs. of 8ème congrès afcet, 1991, pp. 847–859.

[10] _____, *Minimum spanning forests for morphological segmentation*, Procs. of the second international conference on mathematical morphology and its applications to image processing, 1994, pp. 77–84.

[11] L. Najman and M. Schmitt, *Watershed of a continuous function*, Signal Processing **38** (1993), no. 1, 68–86.

[12] _____, *Geodesic saliency of watershed contours and hierarchical segmentation*, IEEE Trans. PAMI **18** (1996), no. 12, 1163–1173.

[13] J. Serra and L. Vincent, *An overview of morphological filtering*, Circuits Systems Signal Process **11** (1992), no. 1, 48–107.

[14] L. Vincent and P. Soille, *Watersheds in digital spaces: An efficient algorithm based on immersion simulations*, IEEE Trans. PAMI **13** (1991), no. 6, 583–598.