Université de Marne-la-Vallée

Choix de publications pour le
Mémoire d'Habilitation à Diriger des Recherches

Discipline : Informatique

Morphologie mathématique, systèmes dynamiques et applications au traitement des images

Présenté par **Laurent Najman**

Habilitation soutenue le 24 mai 2006 devant le jury :

Jean Serra :	président et rapporteur
Laurent Cohen :	rapporteur
Philippe Salembier :	rapporteur
Gilles Bertrand :	examinateur
Maxime Crochemore :	examinateur
Henri Maitre :	examinateur
Dominique Perrin :	examinateur
Karl Tombre :	examinateur

Table des matières

1 Building the component tree in quasi-linear time

L. Najman and M. Couprie. Building the component tree in quasi-linear time.
A paraître dans *IEEE Transactions on Image Processing*.

Building the component tree in quasi-linear time

L. Najman and M. Couprie
Institut Gaspard-Monge
Groupe ESIEE, Laboratoire A2SI
BP99, 93162 Noisy-le-Grand Cedex France
{l.najman,m.couprie}@esiee.fr

***Abstract*— The level sets of a map are the sets of points with level above a given threshold. The connected components of the level sets, thanks to the inclusion relation, can be organized in a tree structure, that is called the *component tree*. This tree, under several variations, has been used in numerous applications. Various algorithms have been proposed in the literature for computing the component tree. The fastest ones (considering the worst-case complexity) have been proved to run in $\mathbf{O}(n \ln(n))$. In this paper, we propose a simple to implement quasi-linear algorithm for computing the component tree on symmetric graphs, based on Tarjan's union-find procedure. We also propose an algorithm that computes the n most significant lobes of a map.**

***Index Terms*— Component tree, connected operators, mathematical morphology, classification, disjoint sets, union-find, image and signal processing, filtering**

I. Introduction

The level sets of a map are the sets of points with level above a given threshold. The connected components of the level sets, thanks to the inclusion relation, can be organized in a tree structure, that is called the *component tree*. The component tree captures some essential features of the map. It has been used (under several variations) in numerous applications among which we can cite: image filtering and segmentation [12], [11], [7], [14], video segmentation [21], image registration [16], [18], image compression [21] and data visualization [5]. This tree is also fundamental for the efficient computation of the topological watershed introduced by M. Couprie and G. Bertrand [7], [8], [3].

While having been (re)discovered by several authors for image processing applications, the component tree concept was first introduced in statistics [26], [13] for classification and clustering. For image processing, the use of this tree in order to represent the "meaningful" information contained in a numerical function can be found in particular, in a paper by Hanusse and Guillataud [12], [11]; the authors claim that this tree can play a central role in image segmentation, and suggest a way to compute it, based on an immersion simulation. Several authors, such as Vachier [25], Breen and Jones [4], Salembier et al. [21] have used some variations of this structure in order to implement efficiently some morphological operators (e.g. connected operators [22], granulometries, extinction functions, dynamics [2]).

Let us describe informally an "emergence" process that will later help us designing an algorithm for building the component tree. Using topographical references, we see the map as the surface of a relief, with the level of a point corresponding to its altitude. Imagine that the surface is completely covered by water, and that the level of water slowly decreases. Islands (regional maxima) appear. These islands form the leafs of the component tree. As the level of water decreases, islands grow, building the branches of the tree. Sometimes, at a given level, several islands merge into one connected piece. Such pieces are the forks of the tree. We stop when all the water has disappeared. The emerged area forms a unique component: the root of the tree.

Various algorithms have been proposed in the literature for computing the component tree [4], [21], [15], the latter reference also contains a discussion about time complexity of the different algorithms. The fastest ones (considering the worst-case complexity) have been proved to run in $\mathrm{O}(n \ln(n))$, where n denotes the number of pixels of the image. In this paper[1], we propose a quasi-linear algorithm for computing the component tree of functions defined on general symmetric graphs, based on Tarjan's union-find [24] procedure. More precisely, our algorithm runs in $O(N \times \alpha(N))$ where N denotes the size of the graph (number of vertices + number of edges) and α is a very slow-growing "diagonal inverse" of the Ackermann's function (we have $\alpha(10^{80}) \approx 4$). We would like to emphasize that this algorithm is simple to implement.

The paper is organised as follows: we first recall the definitions of some basic graph notions and define the component tree in this framework. We explain the disjoint set problem, together with the solution proposed by Tarjan. Using a disjoint set fomulation, we present our component tree algorithm, and we describe its execution on an example. We then show that the proposed algorithm is quasi-linear with respect to the size of the graph, and compare it to one of the most cited component tree algorithm. We illustrate the use of the component tree for automatic detection of some image features, based on a unique parameter which is the number of features that we expect to find in the image.

II. Vertex-weighted graph and component tree

A. Basic notions for graphs

Let V be a finite set of vertices (or points), and let $\mathcal{P}(V)$ denote the set of all subsets of V. Throughout this paper, E denotes a binary relation on V (that is, a subset of the cartesian product $V \times V$) which is anti-reflexive ($(x, x) \notin E$) and symmetric ($(x, y) \in E \Leftrightarrow (y, x) \in E$). We say that the

[1]A preliminary and reduced version of this paper appeared in conference proceedings as [19]. This work has been partially supported by the CNRS.

pair (V, E) is a *graph*, and the elements of E are called *edges*. We denote by Γ the map from V to $\mathcal{P}(V)$ such that, for all $x \in V$, $\Gamma(x) = \{y \in V | (x, y) \in E\}$. For any point x, the set $\Gamma(x)$ is called the *neighborhood of* x. If $y \in \Gamma(x)$ then we say that y is a *neighbor* of x and that x and y are *adjacent*.

Let $X \subseteq V$. Let $x_0, x_n \in X$. A *path from* x_0 *to* x_n *in* X is a sequence $\pi = (x_0, x_1, \ldots, x_n)$ of points of X such that $x_{i+1} \in \Gamma(x_i)$, with $i = 0 \ldots n-1$. Let $x, y \in X$, we say that x and y are *linked for* X if there exists a path from x to y in X. We say that X is *connected* if any x and y in X are linked for X. We say that $Y \subseteq V$ is a *connected component of* X if $Y \subseteq X$, Y is connected, and Y is maximal for these two properties (*i.e.*, $Y = Z$ whenever $Y \subseteq Z \subseteq X$ and Z is connected).

In the following, we assume that the graph (V, E) is connected, that is, V is made of exactly one connected component.

B. Basic notions for vertex-weighted graphs

We denote by $\mathcal{F}(V, D)$, or simply by $\mathcal{F}$, the set composed of all maps from V to D, where D can be any finite set equipped with a total order (*e.g.,* a finite subset of the set of rational numbers or of the set of integers). For a map $F \in \mathcal{F}$, the triplet (V, E, F) is called a *(vertex-)weighted graph*. For a point $p \in V$, $F(p)$ is called the *weight* or *level* of p.

Let $F \in \mathcal{F}$, we define $F_k = \{x \in V | F(x) \geq k\}$ with $k \in D$; F_k is called a *(cross-)section* of F. A connected component of a section F_k is called a *(level k) component* of F. A level k component of F that does not contain a level $(k+1)$ component of F is called a *(regional) maximum* of F. We define $k_{\min} = \min \{F(x) | x \in V\}$ and $k_{\max} = \max \{F(x) | x \in V\}$, which represent respectively, the minimum and the maximum level in the map F.

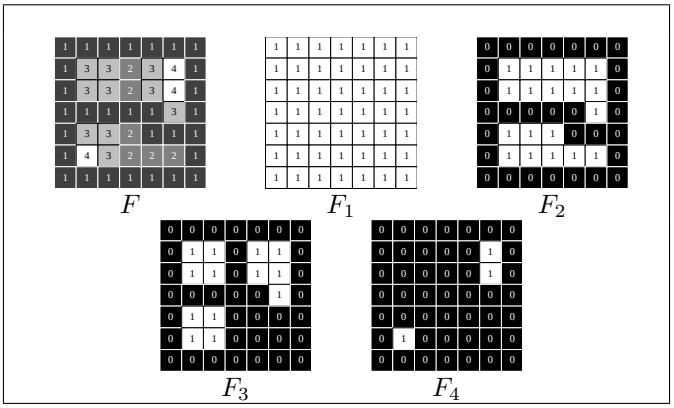
Although the notions we are dealing with in this paper are defined for general graphs, we are going to illustrate our work with the case of 2D images that we model by weighted graphs. Let $\mathbb{Z}$ denote the set of integers. We choose for V a subset of $\mathbb{Z}^2$. A point $x \in V$ is defined by its two coordinates (x_1, x_2). We choose for E the 4-connected adjacency relation defined by $E = \{(x, y) \in V \times V \ ; \ |x_1 - y_1| + |x_2 - y_2| = 1\}$.

Fig. 1.a shows a weighted graph (V, E, F) and four cross-sections of F, between the level $k_{\min} = 1$ and the level $k_{\max} = 4$. The set F_4 is made of two connected components which are regional maxima of F.

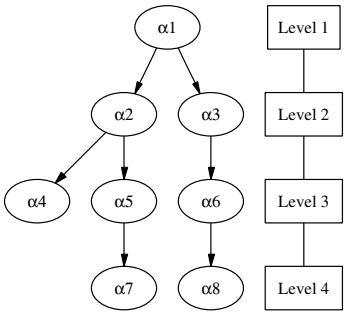
C. Component Tree

From the example of Fig. 1.a, we can see that the connected components of the different cross-sections may be organized, thanks to the inclusion relation, to form a tree structure (see also [2]).

Let $F \in \mathcal{F}$. For any component c of F, we set $h(c) = \max\{k | c \text{ is a level } k \text{ component of } F\}$. Note that $h(c) = \min\{F(x) | x \in c\}$. We define $\mathcal{C}(F)$ as the set composed of all the pairs $[k, c]$, where c is a component of F and $k = h(c)$. We call *altitude of* $[k, c]$ the number k. Remark that $[k_1, c] \in \mathcal{C}(F)$ and $[k_2, c] \in \mathcal{C}(F)$ implies $k_1 = k_2$, in other words, any two distinct elements of $\mathcal{C}(F)$ correspond to distinct sets of points.



(a)



(b)

α_1	α_1	α_1	α_1	α_1	α_1	α_1
α_1	α_4	α_4	α_2	α_5	α_7	α_1
α_1	α_4	α_4	α_2	α_5	α_7	α_1
α_1	α_1	α_1	α_1	α_1	α_5	α_1
α_1	α_6	α_6	α_3	α_1	α_1	α_1
α_1	α_8	α_6	α_3	α_3	α_3	α_1
α_1	α_1	α_1	α_1	α_1	α_1	α_1

(c)

Fig. 1. (a) A vertex-weighted graph (V, E, F) and its cross-sections at levels 1, 2, 3, 4. (b) The component tree of F. (c) The associated component mapping. The component at level 1 is called α_1, the two components at level 2 are called α_2 and α_3 (according to the usual scanning order), and so on.

Let $F \in \mathcal{F}$, let $[k_1, c_1], [k_2, c_2]$ be distinct elements of $\mathcal{C}(F)$. We say that $[k_1, c_1]$ is the *parent of* $[k_2, c_2]$ if $c_2 \subset c_1$ and if there is no other $[k_3, c_3]$ in $\mathcal{C}(F)$ such that $c_1 \subset c_3 \subset c_2$. In this case we also say that $[k_2, c_2]$ is a *child of* $[k_1, c_1]$. With this relation "parent", $\mathcal{C}(F)$ forms a directed tree that we call the *component tree of* F, and that we will also denote by $\mathcal{C}(F)$ by abuse of terminology. Any element of $\mathcal{C}(F)$ is called a *node*. An element of $\mathcal{C}(F)$ which has no child (a maximum of F) is called a *leaf*, the node which has no parent (*i.e.*, $[k_{\min}, V]$) is called the *root*.

We define the *component mapping* M as the map which associates to each point $p \in V$ the node $[k, c]$ of $\mathcal{C}(F)$ such that $p \in c$ and $F(p) = k$. The component mapping is necessary for using the component tree in applications.

Fig. 1.b shows the component tree of the weighted graph depicted in Fig. 1.a, and Fig. 1.c shows the associated component mapping. The component at level 1 is called α_1, the two components at level 2 are called α_2 and α_3 (according to the usual scanning order), and so on.

III. Component tree quasi-linear algorithm

A. Disjoint Sets

The disjoint set problem consists in maintaining a collection $\mathcal{Q}$ of disjoint subsets of a set V under the operation of union. Each set X in $\mathcal{Q}$ is represented by a unique element of X, called the *canonical element*. In the following, x and y denote two distinct elements of V. The collection is managed by three operations:

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

Tarjan [24] proposed a very simple and very efficient algorithm called *union-find* to achieve any intermixed sequence of such operations with a quasi-linear complexity. More precisely, if m denotes the number of operations and n denotes the number of elements, the worst-case complexity is $O(m \times \alpha(m, n))$ where $\alpha(m, n)$ is a function which grows very slowly, for all practical purposes $\alpha(m, n)$ is never greater than four[2].

The implementation of this algorithm is given below in procedure **MakeSet** and functions **Link** and **Find**. Each set of the collection is represented by a rooted tree, where the canonical element of the set is the root of the tree. To each element x is associated a parent $\text{Par}(x)$ (which is an element) and a rank $\text{Rnk}(x)$ (which is an integer). The mappings 'Par' and 'Rnk' are represented by global arrays in memory. One of the two key heuristics to reduce the complexity is a technique called *path compression*, that is aimed at reducing, in the long run, the cost of **Find**. It consists, after finding the root r of the tree which contains x, in considering each element y of the parent path from x to r (including x), and setting the parent of y to be r. The other key technique, called *union by rank*, consists in always choosing the root with the greatest rank to be the representative of the union while performing the **Link** operation. If the two canonical elements x and y have the same rank, then one of the elements, say y, is chosen arbitrarily to be the canonical element of the union: y becomes the parent of x; and the rank of y is incremented by one. The rank $\text{Rnk}(x)$ is a measure of the depth of the tree rooted in x, and is exactly the depth of this tree if the path compression technique is not used jointly with the union by rank technique. Union by rank avoids creating degenerate trees, and helps keeping the depth of the trees as small as possible. For a more detailed explanation and complexity analysis, see Tarjan's paper [24].

Procedure `MakeSet(`*element* x`)`

$\text{Par}(x) := x$; $\text{Rnk}(x) := 0$;

[2]The precise definition of α, a "diagonal inverse" of the Ackermann's function, involves notions which are not in the scope of this paper, it can be found in [24].

Function `element Find(`*element* x`)`

if *(*$\text{Par}(x) \neq x$*)* **then** $\text{Par}(x) := \text{Find}(\text{Par}(x))$;
return $\text{Par}(x)$;

Function `element Link(`*element* x*, element* y`)`

if *(*$\text{Rnk}(x) > \text{Rnk}(y)$*)* **then** exchange($x, y$);
if *(*$\text{Rnk}(x) == \text{Rnk}(y)$*)* **then** $\text{Rnk}(y) := \text{Rnk}(y) + 1$;
$\text{Par}(x) := y$;
return y;

B. Illustration of union-find: labelling the connected components

We can illustrate the use of the union-find algorithm on the classical problem of finding the connected components of a subset X of a graph (V, E). Algorithm 1 (ConnectedComponents) is given below. For a set X, this algorithm returns a map M that gives for each point p, the canonical element $M(p)$ of the connected component of X which contains p.

Algorithm 1: ConnectedComponents

Data: (V, E) - graph
Data: A set $X \subseteq V$
Result: M - map from X to V
1 **foreach** $p \in X$ **do** MakeSet(p);
2 **foreach** $p \in X$ **do**
3 compp := Find(p);
4 **foreach** $q \in \Gamma(p) \cap X$ **do**
5 compq := Find(q);
6 **if** *(compp* $\neq$ *compq)* **then**
7 compp := Link(compq, compp);
8 **foreach** $p \in X$ **do** M(p) := Find(p);

During the first pass (loop 1), for each point p of the set X, the set $\{p\}$ is added to the collection $\mathcal{Q}$ of disjoint subsets. Then, loop 2 processes all points of X in an arbitrary order. For each point p, we first find the canonical element of the set it belongs to (line 3). Then, for each neighbor q of p such that $q \in X$ (line 4), we find the canonical element of the set which contains q (line 5). If p and q are not already in the same set, that is if the two canonical elements differ (line 6), then the corresponding sets are merged (line 7), and one of the two canonical elements is chosen to be the canonical element of the merged set. At the end, a simple pass on all the elements of X (loop 8) builds the map M.
Note that, if the vertices can be processed in some very specific order (as the scanline order), the ConnectedComponents algorithm becomes linear [10], [9]. Unfortunately, such a specific strategy is not applicable for the component tree algorithm, where the scanning order depends on the altitudes of the vertices.

C. Component tree algorithm: high-level description

We are now ready to introduce our quasi-linear algorithm for building the component tree $\mathcal{C}(F)$ from a weighted graph $G = (V, E, F)$.

The algorithm simulates the emergence process described in the introduction, and maintains several data structures. The main one is a forest, which initially consists of a set of mutually disconnected *nodes*, each node being associated (initially) to a single vertex of the graph G. During the emergence process, which is realized by scanning all the vertices of G by decreasing order of altitude, the vertices which belong to a same component and have the same altitude are grouped together thanks to a disjoint set collection called $\mathcal{Q}_{\text{node}}$. The canonical element of such a set is called a *canonical node*. Notice that the disjoint set collection $\mathcal{Q}_{\text{node}}$ has essentially the same function as the disjoint set collection used by algorithm ConnectedComponents (sec. III-B).

Simultaneously, the canonical nodes are progressively linked together to form *partial trees*, each partial tree represents intuitively an emerged island. At the end of the execution, a unique tree groups all the canonical nodes, each one of these nodes represents a component of G, and the whole tree constitutes the component tree of G. To reach a quasi-linear time complexity, we have to maintain another collection $\mathcal{Q}_{\text{tree}}$ of disjoint sets, and an auxiliary map called *lowestNode*. Given an arbitrary node P, the collection $\mathcal{Q}_{\text{tree}}$ allows to find, in quasi-constant time, a node T which "represents" the partial tree which contains P. Due to the particular management of $\mathcal{Q}_{\text{tree}}$, this node T cannot be guaranteed to be precisely the root of the partial tree, this is why we also need to maintain the map *lowestNode* which associates, to each canonical element of $\mathcal{Q}_{\text{tree}}$, the root of the corresponding partial tree.

D. Component tree algorithm: detailed view

Algorithm 2 (BuildComponentTree) is given below. It uses two auxiliary functions **MakeNode** and **MergeNodes**. To represent a node of $\mathcal{C}(F)$, we use a structure called `node` containing the level of the node, and the list of nodes which are children of the current node. For building the component tree, we do not need the reverse link, that is we do not need to know the parent of a given node, but let us note that such information is useful for applications, and can easily be obtained in a linear-time post-processing step. In what follows, we are going to show how to compute some attributes associated to each node of the component tree; we thus need that the structure `node` contains some fields that store those attributes, namely `level`, `area` and `highest`. We defer both the precise definition of the attributes and the explaination on how they are computed until section VI, in order to concentrate on the component tree itself.

Function `node MakeNode(`*int level*`)`

```
Allocate a new node n with an empty list of children;
n→ level := level; n→ area := 1; n→ highest := level;
return n;
```

After a preprocessing (line 1, achievable in linear time for short integers [6]) which sorts the points by decreasing order of level and which prepares the two union-find implementations (line 2), we process the points, starting with the highest ones.

Function `int MergeNodes(`*int* $node1$*, int* $node2$`)`

```
tmpNode := Link_node(node1,node2);
if (tmpNode == node2) then
    Add the list of children of nodes[node1]
        to the list of children of nodes[node2];
    tmpNode2 := node1;
else
    Add the list of children of nodes[node2]
        to the list of children of nodes[node1];
    tmpNode2 := node2;
nodes[tmpNode]→area :=
    nodes[tmpNode]→area + nodes[tmpNode2]→area;
nodes[tmpNode]→highest :=
    max(nodes[tmpNode]→highest,
        nodes[tmpNode2]→highest);
return tmpNode;
```

Let us suppose that we have processed a number of levels. We have built all nodes of the component tree that are above the current level, and we are building the nodes with exactly the current level. For a given point p of the current level (line 3), we know (through the collection $\mathcal{Q}_{\text{tree}}$) the partial tree the node p belongs to (line 4). In each partial tree, there is only one node with the current level, that we can obtain through the auxiliary map $lowestNode$. We then find the associated canonical node (line 5).

We then look at each neighbor q of p with a level greater or equal to the current one (loop 6). Note that, as the graph is symmetric, the "linking operations" between two points are done when one of the two points is processed as a neighbor of the other. Thus, we can use the order of scanning of the points, and we only need to examine the "already processed" neighbors of p. Such a neighbor q satisfies $F(q) \geq F(p)$.

Exactly as we have done for the point p, we search for the canonical node corresponding to the point q (lines 7-8). If the canonical node of p and the canonical node of q differ, that is if the two points are not already in the same node, we have two possible cases:

- either the two canonical nodes have the same level; this means that these two nodes are in fact part of the same component, and we have to merge the two nodes (line 9 and function **MergeNodes**). The merging of nodes of same level is done through the collection $\mathcal{Q}_{\text{node}}$ of disjoint sets. The merging relies on the fact that the $\textbf{Link}_{\textbf{node}}$ function always chooses one of the two canonical elements of the sets that are to be merged as the canonical element of the merged set. This fact is used in the sequel of the function.
  Once the merging has been done, one of the nodes is chosen to be the canonical element of the disjoint set. Observe that the other node is not needed anymore. Indeed, we only have to know to which disjoint set this last node belongs to, and the answer to this question is given by the $\textbf{Find}_{\textbf{node}}$ function.
- or the canonical node of q is strictly above the current level, and thus this node becomes a child of the current

Algorithm 2: BuildComponentTree

Data: (V, E, F) - vertex-weighted graph with N points.
Result: $nodes$ - array $[0 \dots N-1]$ of nodes.
Result: $Root$ - Root of the component tree
Result: M - map from V to $[0 \dots N-1]$ (component mapping).
Local: $lowestNode$ - map from $[0 \dots N-1]$ to $[0 \dots N-1]$.

```
1  Sort the points in decreasing order of level for F;
2  foreach p ∈ V do {MakeSet_tree(p); MakeSet_node(p); nodes[p]:= MakeNode(F(p)); lowestNode[p] := p;};
3  foreach p ∈ V in decreasing order of level for F do
4      curTree := Find_tree(p);
5      curNode := Find_node(lowestNode[curTree]);
6      foreach already processed neighbor q of p with F(q) ≥ F(p) do
7          adjTree := Find_tree(q);
8          adjNode := Find_node(lowestNode[adjTree]);
           if (curNode ≠ adjNode) then
               if (nodes[curNode]→level == nodes[adjNode]→level) then
9                  curNode := MergeNodes(adjNode, curNode);
               else
                   // We have nodes[curNode]→level < nodes[adjNode]→level
10                 nodes[curNode]→addChild(nodes[adjNode]);
11                 nodes[curNode]→area := nodes[curNode]→area + nodes[adjNode]→area;
12                 nodes[curNode]→highest := max(nodes[curNode]→highest, nodes[adjNode]→highest);
13             curTree := Link_tree(adjTree, curTree);
14             lowestNode[curTree] := curNode;
15 Root := lowestNode[Find_tree(Find_node(0))] ;
16 foreach p ∈ V do M(p) := Find_node(p);
```

node (line 10).

In both cases, we have to link the two partial trees, this is done using the collection $\mathcal{Q}_{\text{tree}}$ (line 13). We also have to keep track of the node of lowest level for the union of the two partial trees, that we store in the array $lowestNode$ (line 14).

At the end of the algorithm, we have to do a post-processing to return the desired result. The root of the component tree can easily be found (line 15) using the array $lowestNode$ and the two disjoint set structures $\mathcal{Q}_{\text{tree}}$ and $\mathcal{Q}_{\text{node}}$. The component mapping M can be obtained using the disjoint set $\mathcal{Q}_{\text{node}}$ (loop 16).

IV. Illustration of the algorithm

Let us illustrate the work of the algorithm on an example. Consider the weighted graph of Fig. 2.a. The points are labelled according to their usual lexicographical order (Fig. 2.b).

At the beginning of the sixth step, we have already constructed parts of the component tree (Fig. 3.b). We show in Fig. 3.a the maps Par_{tree}, Par_{node}, and $lowestNode$. For the maps Par_{tree} and Par_{node}, the canonical elements appear in white. It should be noted that the $lowestNode$ mapping is only used for the canonical elements of Par_{tree}: this explains why the values of $lowestNode$ for other elements (in grey) are not updated.

We are going to process nodes at level 50. The first node at level 50 is node 3. Node 0 is a neighbor of node 3. The canonical node corresponding to 0 is node 1, the level of which

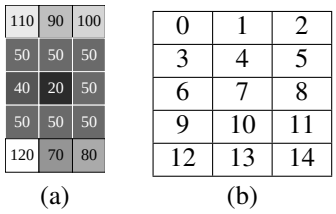


Fig. 2. (a) Original vertex-weighted graph. (b) Points are labelled according to the usual lexicographic order, but they will be processed by decreasing level (that is: $12, 0, 2, 1, 14, 13, 3, 4, 5, 8, 9, 10, 11, 6, 7$).

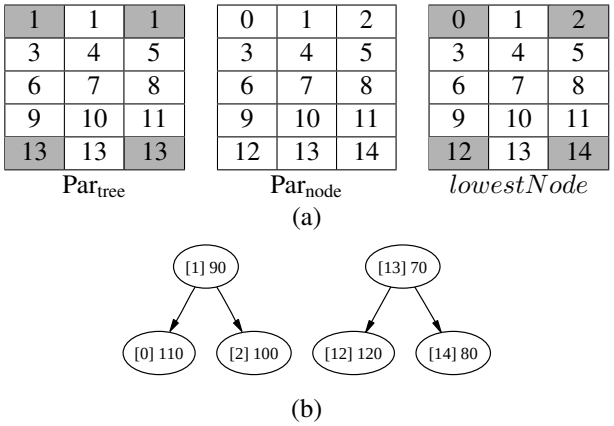


Fig. 3. Beginning of step 6. (a) State of the maps Par_{tree}, Par_{node} and $lowestNode$. (b) Partial trees constructed.

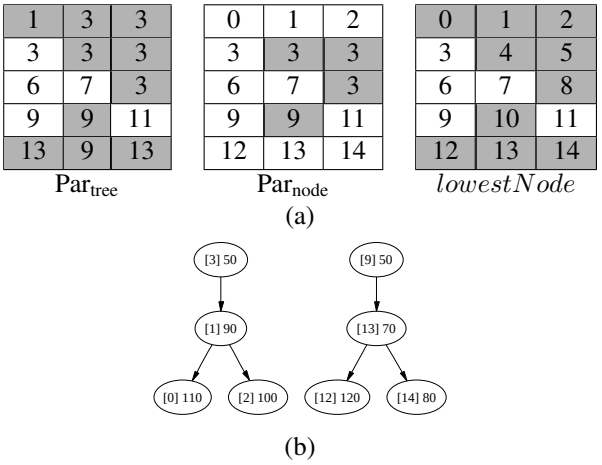


Fig. 4. Beginning of step 11. (a) State of the maps Par_{tree}, Par_{node} and $lowestNode$. (b) Partial trees constructed.

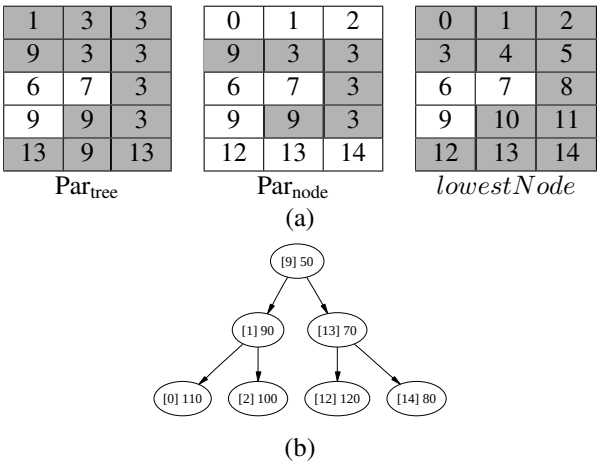


Fig. 5. End of step 11. (a) State of the maps Par_{tree}, Par_{node} and $lowestNode$. (b) Partial trees constructed.

is 90. Thus node 3 becomes the parent of node 1. Then, node 3 is linked for $\mathcal{Q}_{\text{node}}$ succesively with nodes 4, 5 and 8. Then node 9 is examined, and is linked for $\mathcal{Q}_{\text{node}}$ with node 10, the node 9 being chosen as the canonical one. Node 9 is a neighbour of node 12, the canonical element of which is node 13 (level 70). Thus, node 13 becomes a child of node 9. We are then at the beginning of step 11, and this is illustrated on Fig. 4.

Node 11 is a neighbor of both nodes 8 and 10. The canonical node of node 8 is node 3 at level 50. Thus, node 11 and node 3 are linked for $\mathcal{Q}_{\text{node}}$, and node 3 is chosen as the canonical one. The canonical node of node 10 is node 9 at level 50. Thus, nodes 9 and 3 are merged, that is, the corresponding partial trees are merged into a single tree. Node 9 is chosen as the canonical element of the level 50 component, and the children of node 3 are transfered to node 9. We are in the situation depicted in Fig. 5.

We then process node 6 at level 40, which becomes the parent of node 9 at level 50. Node 9 and node 6 are linked for $\mathcal{Q}_{\text{tree}}$, and node 9 is chosen as the canonical element for the partial tree. The lowest node in this partial tree is

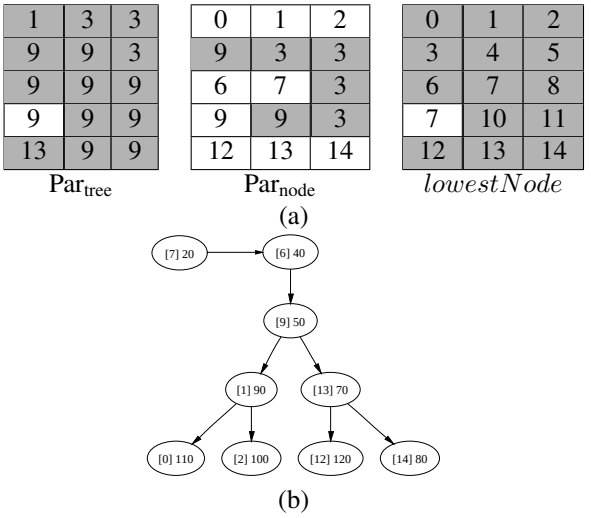


Fig. 6. End of step 14. (a) State of the map Par_{tree}, Par_{node} and $lowestNode$. (b) Component tree.

node 6 at level 40. We use the map $lowestNode$ to store that information, by setting $lowestNode[9] := 6$. Then we process node 7 at level 20, which becomes the parent of node 6. Node 9 is chosen as the canonical element for the partial tree, and thus we have to store the lowest node by setting $lowestNode[9] := 7$. There is no node lower than 20, and thus, the component tree is built. The final situation is depicted in Fig. 6.

The collection $\mathcal{Q}_{\text{tree}}$ of disjoint sets is not useful anymore: indeed, each node of the graph has been examined, and they are all linked for $\mathcal{Q}_{\text{tree}}$, the canonical element being the node 9. The root of the component tree is the node 7. Each of the canonical elements of the collection $\mathcal{Q}_{\text{node}}$ corresponds to a component of F: observe in particular the level 50, whose canonical node is node 9. The collection $\mathcal{Q}_{\text{node}}$ can be used to compute the component mapping M.

V. Complexity analysis

Let n denote the number of points in V, and let m denote the number of edges of the graph (V, E).

The sorting of the points (line 1) can be done in $O(n)$ if the weigths are small integers (counting sort [6]), and in $O(n \log(\log(n)))$ if each weight can be stored in a machine memory word (long integers or floating point numbers [1]).

Loop 2 is the preparation for the union-find algorithm. It is obviously $O(n)$.

In the function **MergeNodes**, the merging of the lists of children can be done in constant time, because we can merge two lists by setting the first member of one list to be the one that follows the last member of the other list. This requires the two lists to be disjoint, which is the case (we are dealing with disjoint sets), and an adequate representation for lists (chained structure with pointers on both first and last element).

The amortized complexity of line 6 is equal to the number m of edges of the graph (V, E). The amortized complexity of all calls to the union-find procedures is quasi-linear (in the sense explained in section III-A) with respect to m. The building of the component mapping M is obviously linear.

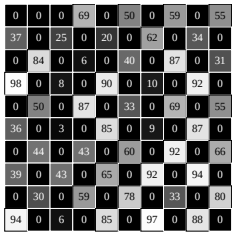


Fig. 7. An example of an artificially generated image of size N*N, where values of pixel (x_1, x_2) with $x_1 + x_2$ odd are uniformly distributed between 0 and $N * N$, and where the other half of the pixels are 0. Using a series of such images, one can verify that the component tree algorithm of Salembier *et al.* is quadratic.

Thus the complexity of the algorithm 2 (BuildComponentTree) is quasi-linear if the sorting step is linear.

Note that the memory for the $lowestNode$ array is not necessary: we can easily modify the code so that we store the content of $lowestNode$ as negative values in Par_{tree} for the canonical element of $\mathcal{Q}_{\text{tree}}$. In this case, for an element $x \in V$, $\text{Find}_{\text{tree}}(x)$ still returns the canonical element c for $\mathcal{Q}_{\text{tree}}$, but $lowestNode(c) = -\text{Par}_{\text{tree}}(c)$. The modifications that have to be made to **MakeSet**, to **Find**, and to **BuildComponentTree** are straightforward and do not change the complexity of the algorithm.

For comparison purpose, one can prove that the most cited component tree algorithm, the Salembier *et al.* algorithm [21] is quadratic. More precisely, although there is no complexity analysis in [21], one can verify that the Salembier *et al.* algorithm has a worst-case time complexity in $O(n \times h + m)$ where h is the number of levels of the image. The worst case can be attained using a series of artificially generated images such that half of the pixels are maxima of the images (an example of an image of the series is provided in Fig. 7). However, this worst case is rare in practice. We observe that, when the level of a point is a short integer (between 0 and 255), the Salembier *et al.* algorithm is generally twice as fast as our algorithm. This can be explained by the fact that, for each point of the image, we have to access the two union-find data structures, while this is not the case for the Salembier *et al.* algorithm.

VI. Attributes

A major use of the component tree is for image filtering: for example, we may want to remove from an image the "lobes" that are not "important enough" or "negligible". Such an operation is easy to do by simply removing the "negligible" components of the component tree. To make such an idea practicable, it is necessary to quantify the relative importance of each node of the component tree. We can do that by computing some attributes for each node.

Among the numerous attributes that can be computed, three are natural: the height, the area, and the volume (Fig. 8).

Let $[k, c] \in \mathcal{C}(F)$. We define

$$\begin{aligned} \text{height}([k,c]) &= \max\{F(x) - k + 1 | x \in c\} \\ \text{area}([k,c]) &= \text{card}(c) \end{aligned}$$

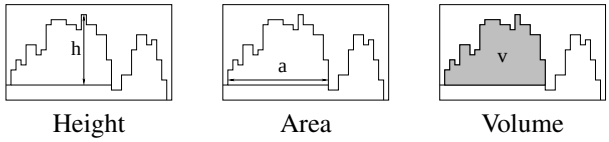


Fig. 8. Illustration of the height, the area and the volume of a component.

$$\text{volume}([k,c]) = \sum_{x \in c} (F(x) - k + 1)$$

The area is easy to compute while building the component tree. Each time two components merge (*i.e.* in the function MergeNodes) or each time a component is declared the parent of another one (*i.e.* line 11 of algorithm 2 BuildComponentTree), we keep as the new area the sum of the areas of the two components.

For computing the highest level in the component, we do as we did for the area, replacing the sum by the maximum (see line 12 of algorithmm 2 BuildComponentTree and the function MergeNodes). From this highest level, the height of a component n can easily be computed by setting $\text{height}(n) = (n \rightarrow \text{highest}) - (n \rightarrow \text{level}) + 1$.

To compute the volume, we first need the area. We then apply the recursive function ComputeVolume on the root of the tree. The complexity of this function is linear with respect to the number of nodes.

Function `int ComputeVolume(`*int n*`)`

```
vol := nodes[n]→area;
foreach c child of nodes[n] do
    vol := vol + ComputeVolume(c) +
        c→area * (c→level - nodes[n]→level);
nodes[n]→volume := vol ;
return vol;
```

VII. Example of application and conclusion

We have mentioned a simple use of the component tree for filtration (removing nodes of the tree whose attribute is below a given threshold). A more advanced use consists in finding the most significant lobes of a given weighted graph F. More precisely, we want to find the N most significant components with respect to either the height, area or volume criterion. By using the tree, this task reduces to the search of the N nodes that have the largest attribute values and are not bound with each other (even transitively) by the inclusion relation. Algorithm 3 (Keep_N_Lobes) performs this task. Its time complexity is in $\text{O}(\text{sort}(n) + m)$, where m is the number of vertices in the graph, n is the number of component tree nodes and $\text{sort}(n)$ is the complexity of the sorting algorithm. At the end of the algorithm, the remaining leaves (more precisely, the pixels which are associated to these leaves) mark the desired significant lobes. For this algorithm, each node must include fields to store its parent and its number of children (but the list of children of a given node is not necessary).

Fig. 9 illustrates this algorithm. Fig. 9.a is an image of cell, in which we want to extract the ten bright lobes. Fig. 9.b shows

Algorithm 3: Keep_N_Lobes

Data: A vertex-weighted graph (V, E, F), its component tree T with attribute value for each node, and the associated component mapping M
Data: The number N of wanted lobes.
Result: The filtered map F

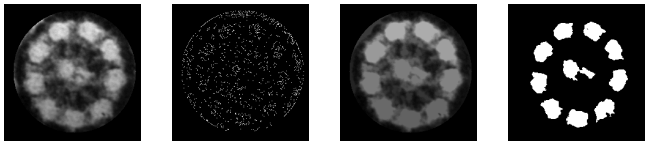
```
Sort the nodes of T by increasing order of attribute value;
Q := ∅; L := number of leaves in T;
forall n do nodes[n]→mark := 0;
while L > N do
    Choose a (leaf) node c in T with smallest attribute value;
    p := nodes[c]→parent ;
    nodes[p]→nbChildren := nodes[p]→nbChildren-1;
    if (nodes[p]→nbChildren > 0) then L := L-1;
    nodes[c]→mark := 1 ; Q := Q ∪ {c};
while ∃c ∈ Q do
    Q := Q \ {c}; RemoveLobe(c);
foreach x ∈ V do F(x) := nodes[M[x]]→level;
```

Function `int RemoveLobe(`*int n*`)`

```
if (nodes[n]→mark == 1) then
    nodes[n] := nodes[RemoveLobe(nodes[n]→parent)];
return n;
```

that the image 9.a contains numerous maxima. Fig. 9.c is the filtered image obtained by using algorithm 3 with the volume attribute and with parameter value 10, and Fig. 9.d shows the maxima of this filtered image. Note that a similar result could be obtained with this image by performing attribute based operations using several volume threshold values, following *e.g.* a dichotomic method, until the desired number of maxima is reached. This latter approach is not only less efficient than the proposed algorithm, but it may also fail to find the precise number of maxima required by the user, in the case of components having precisely the same attribute value. In such cases, the proposed algorithm always makes a choice in order to fulfill the user's requirement.

The component tree allows the efficient implementation of complex image and signal filtering, based for example on the use of criteria such as area, volume or depth, or even the use of non-increasing criteria [21]. Although some of these filters may be computed using specific and sometimes faster algorithm (in particular area filtering [17]), using the component tree is in general the simplest and the most efficient way to compute these filters. Moreover, once the component tree of a function is computed, any of these filters, with any parameter value, can be computed at a very low cost. The component tree is also a key element of an efficient algorithm for the topological watershed [8]. New classes of filters, such as second-order connected operators [23] have been recently introduced to generalize connected operators [22]. Those operators can also be efficiently implemented using the component tree [20]. In this paper, we have proposed a simple-to-implement quasi-linear algorithm for computing the component tree. We hope that such an algorithm will facilitate the extensive practical use of such operators.



(a) (b) (c) (d)

Fig. 9. (a) Original image. (b) Maxima of image (a), in white. (c) Filtered image. (d) Maxima of image (c), which correspond to the ten most significant lobes of the image (a).

REFERENCES

[1] A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in linear time? In *STOC: ACM Symposium on Theory of Computing*, 1995.
[2] G. Bertand. On the dynamics. *Image and Vision Computing*. Submitted.
[3] G. Bertrand. On topological watersheds. *JMIV*, 22(2-3):217–230, 2005.
[4] E.J. Breen and R. Jones. Attribute openings, thinnings and granulometries. *Comp. Vision and Image Und.*, 64(3):377–389, Nov. 1996.
[5] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Comp. Geometry: Theory and Applications*, 30(2):165–195, 2005.
[6] T. H. Cormen, C. L. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
[7] M. Couprie and G. Bertrand. Topological grayscale watershed transform. In *SPIE Vision Geom. V Proc.*, volume 3168, pages 136–146, 1997.
[8] M. Couprie, L. Najman, and G. Bertrand. Quasi-linear algorithms for the topological watershed. *JMIV*, 22(2-3):231–249, 2005.
[9] M.B. Dillencourt, H. Samet and M. Tamminen. *A general approach to connected-component labeling for arbitrary image representations*. Journal of the Assoc. Comput. Mach., 39(2):253–280, 1992.
[10] Ch. Fiorio and J. Gustedt. *Two linear Union-Find strategies for image processing*. *Th. Computer Science* 154:165-181, 1996.
[11] P. Guillataud. *Contribution à l'analyse dendroniques des images*. PhD thesis, Université de Bordeaux I, 1992.
[12] P. Hanusse and P. Guillataud. Sémantique des images par analyse dendronique. In *8th RFIA*, volume 2, pages 577–588, 1992
[13] J.A. Hartigan. Statistical theory in clustering. *Journal of Classification*, 2:63–76, 1985.
[14] R. Jones. Component trees for image filtering and segmentation. In *NSIP'97*, 1997.
[15] J. Mattes and J. Demongeot. Efficient algorithms to implement the confinement tree. In *LNCS*:1953, pages 392–405. Springer, 2000.
[16] J. Mattes, M. Richard, and J. Demongeot. Tree representation for image matching and object recognition. In *LNCS*:1568, pages 298–309, 1999.
[17] A. Meijster and M.H.F. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Trans. on PAMI*, 24(4):484–494, April 2002.
[18] P. Monasse. *Morphological representation of digital images and application to registration*. PhD thesis, Paris-Dauphine Univ., June 2000.
[19] L. Najman and M. Couprie. Quasi-linear algorithm for the component tree. In *SPIE Vision Geometry XII*, Vol. 5300 pages 98–107, 2004
[20] G.K. Ouzounis and M.H.F. Wilkinson. Second-order connected attribute filters using max-trees. In *Mathematical morphology: 40 years on*, pages 65–74. Springer, 2005.
[21] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Trans. on Image Proc.*, 7(4):555–570, April 1998.
[22] P. Salembier and J. Serra. Flat zones filtering, connected operators and filter by reconstruction. *IEEE Tr. on Im. Proc.*, 3(8):1153–1160, 1995.
[23] J. Serra. Connectivity on complete lattices. *JMIV*, 9:231–251, 1998.
[24] R.E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.
[25] C. Vachier. *Extraction de caractéristiques, segmentation d'images et Morphologie Mathématique*. PhD thesis, ENSMP, 1995
[26] D. Wishart. Mode analysis: A generalization of the nearest neighbor which reduces chaining effects. In A.J. Cole, editor, *Numerical Taxonomy*, pages 282–319, London, 1969. Academic Press.

2 Watershed of a Continuous Function

L. Najman and M. Schmitt. Watershed of a Continuous Function. *Signal Processing*, 38, 99-112 (1994).
Special issue on Mathematical Morphology.

ELSEVIER

Signal Processing 38 (1994) 99–112

SIGNAL PROCESSING

Watershed of a continuous function

Laurent Najman*[a,b], Michel Schmitt[a]

[a] *L.C.R, Thomson-CSF, Domaine de Corbeville, 91404 Orsay, France*
[b] *Ceremade, Université Paris Dauphine, 75775 Paris, France*

Received 8 September 1993; revised 1 November 1993

Abstract

The notion of watershed, used in morphological segmentation, has only a digital definition. In this paper, we propose to extend this definition to the continuous plane. Using this continuous definition, we present the watershed differences with classical edge detectors. We then exhibit a metric in the plane for which the watershed is a skeleton by influence zones and show the lower semicontinuous behaviour of the associated skeleton. This theoretical approach suggests an algorithm for solving the eikonal equation: $\|\nabla f\| = g$. Finally, we end with some new watershed algorithms, which present the advantage of allowing the use of markers and/or anchor points, thus opening the way towards grey-tone skeletons.

Zusammenfassung

Der Begriff Wasserscheide, der in der morphologischen Segmentation verwendet wird, hat nur eine diskrete Definition. In diesem Artikel schlagen wir vor, diese Definition auf die kontinuierlich Ebene auszudehnen. Indem wir die kontinuierliche Definition verwenden, stellen wird die Unterschiede zwischen Wasserscheide und klassischen Kantendetektoren vor. Wir zeigen dann eine Metrik in der Ebene, für die die Wasserscheide ein Skelett von Einflußzonen ist und zeigen das untergeordnete halbkontinuierliche Verhalten von damit verbunden Skeletten. Dieser theoretische Ansatz schlägt einen Algorithmus für die Lösung der Eikonal-Gleichung $\|\nabla f\| = g$ vor. Schließlich gelangen wir zu einem neuen Wasserscheiden-Algorithmus, der den Vorteil hat, die Benutzung von Markierungen und/oder Ankerpunkten zu erlauben und daher den Weg zu Graustufenskeletten öffnet.

Résumé

La notion de ligne de partage des eaux, utilisée en segmentation morphologique dispose uniquement d'une définition digitale. Dans cet article, nous proposons d'étendre la définition de la ligne de partage des eaux au plan continu. En utilisant cette définition continue, nous comparons la ligne de partage des eaux avec les extracteurs de contours classiques, et montrons leurs différences. Nous introduisons ensuite une métrique pour laquelle la ligne de partage des eaux est un squelette par zones d'influence, ce qui nous permet de montrer son comportement semi-continu. Cette approche théorique nous suggère un nouvel algorithme pour résoudre l'équation eikonal: trouver f telle que $\|\nabla f\| = g$.

* Corresponding author. Tel: (33-1) 69 33 09 17; Fax: (33-1) 69 33 08 65; E-mail: najman@thomson-lcr.fr.

SSDI 0165-1684(94)00021-Q

Nous terminons enfin sur de nouveaux algorithmes de ligne de partage des eaux, présentant l'avantage de pouvoir inclure des marqueurs et des points d'ancrages, ouvrant ainsi la voie aux squelettes à teintes de gris.

Key words: Mathematical morphology; Watershed; Image distance; Eikonal equation; Edge detection

1. Introduction

One of the most intuitive notions of morphological segmentation is the notion of watershed [2, 3, 13, 22, 28]. The algorithms proposed in the literature have only a formal link with the various definitions of the watershed. Intuitively, the idea is that the watershed is a skeleton by influence zones with respect to a special distance, but all the previous theoretical definitions are in the general case a kind of skeleton, i.e. the previous watersheds do have barbs.

In this paper, we propose a proof of the convergence of the algorithm of Beucher and Lantuéjoul [3]. This proof allows us to give a meaning to a continuous definition of the watershed, and to show the very link between the watershed and the skeleton. Using this continuous definition, we are able to compare the watershed with the classical second-order differential operators used to detect edges. This theoretical work leads us to new algorithms, one to solve the eikonal equation, and the other to compute a watershed, with the advantage of allowing the use of markers and anchor points.

2. The watershed: from discrete to continuum

2.1. Classical digital algorithm

We follow here the presentation of L. Vincent [28].

In mathematical morphology, it is usual to consider that an image is a topographical surface. It is done by considering the grey level (the image intensity) as an altitude. Places of high variation in the intensity are then a good set in which one can search for contour lines. It is then rather straightforward to estimate the variation from the gradient of the image. For the purpose of segmentation, we are then looking for the *crest lines of the gradient image*. A way for doing this operation is to apply the watershed algorithm to the gradient image.

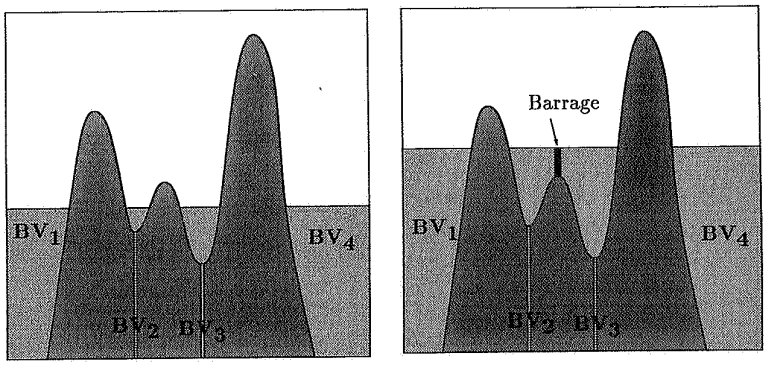
The idea of the watershed is to attribute an *influence zone* to each of the *regional minima* of an image (connected plateau from which it is impossible to reach a point of lower grey level by an always descending path). We then define the watershed as the boundaries of these influence zones.

Numerous techniques have been proposed to compute the watershed. The major ones are reviewed in [28, 30]. The classical idea for building the watershed is simple to describe in one dimension (Fig. 1). We begin by piercing the regional minima of the surface. Then, we slowly immerse the image into a lake. The water progressively floods the basins corresponding to the various minima (Fig. 1(a)). To prevent the merging of two different waters originating from two different minima, we erect a barrage (Fig. 1(b)). Once the surface is totally immersed, the set of the barrages thus built is the watershed of the image. In one dimension, the location of the watershed is straightforward. In two dimensions (which is the case of the classical images) this characterization is not so easy (Fig. 2). One can say in an informal way that the watershed is the *crest lines* of the image.

We give here the classical algorithm allowing the computation of the watershed. The most powerful implantation described in the literature [4, 21, 29, 30] uses FIFO breadth-first scanning techniques for the actual flooding.

Following the ideas we mentioned above, the algorithm consists of flooding the water in the various basins, and to keep as the watershed the set of contact points between two different basins. In the case where this contact is on a plateau, we keep the (geodesic) middle of this plateau. The watershed thus defined is of thickness one on the grid.

To compute the geodesic middle on the contact plateaus, we use the geodesic distance.



(a) At time t, the barrage is not yet constructed.

(b) At time $t+h$, we construct a barrage to separate water from $\mathbf{BV_2}$ and from $\mathbf{BV_3}$.

Fig. 1. Building of the watershed: one-dimensional approach.

Definition 2.1. Let A be a set, a and b two points of A. We call *geodesic distance* in A $d_A(a,b)$ the lower bound of the lengths of the paths γ in A linking a and b.

In the digital case, the distance d_A is deduced from one of the grids [20]. Let $B = \bigcup B_i \subset A$, where B_i are the connected components of B.

Definition 2.2. The *geodesic influence zone* $iz_A(B_i)$ of a connected component B_i of B in A is the set of the points of A for which the geodesic distance to B_i is smaller than the geodesic distance to other connected components of B.

$$iz_A(B_i) = \{p \in A,\ \forall j \in [1,k]\backslash\{i\},\ d_A(p,B_i) < d_A(p,B_j)\}. \tag{1}$$

The points of A which do not belong to any influence zone make up the skeleton by influence zones of B in A, noted $SKIZ_A(B)$:

$$SKIZ_A(B) = A\backslash IZ_A(B), \tag{2}$$

where $IZ_A(B) = \bigcup_{i\in[1,k]} iz_A(B_i)$.

The watershed algorithm on digital images by recurrence on grey level is the following definition [13]:

Definition 2.3. The set of the *catchment basins* of the numerical image I is the set $X_{h_{\max}}$ obtained after the following recurrence:

$$X_{h_{\min}} = T_{h_{\min}}(I), \tag{3a}$$

$$\forall h \in [h_{\min}, h_{\max} - 1],\quad X_{h+1} = \mathrm{Min}_{h+1} \cup IZ_{T_{h+1}(I)}(X_h), \tag{3b}$$

where

- $h_{\min} \in \mathbb{Z}$ (respectively $h_{\max}$) is the lowest (respectively the greatest) grey level of image I.
- $T_h(I)$ is the threshold of the image I at height h: $T_h(I) = \{p \mid I(p) \leqslant h\}$
- Min_h is the set of the regional minima of I at the height h.

The *watershed* of the image I is the complement of this set.

Note that this algorithm works only for step functions.

2.2. Continuous generalization

From now on, image f is supposed to be regular enough ($\mathscr{C}^2$) to allow the use of differential operators. We use classical tools of differential geometry

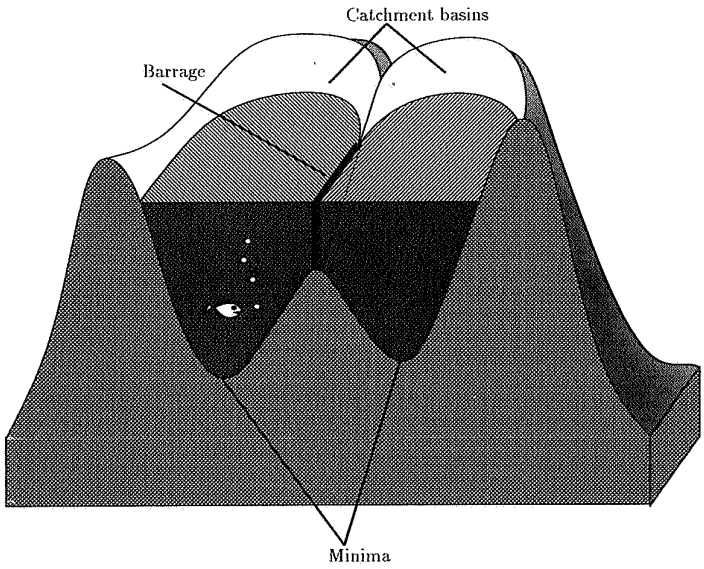


Fig. 2. Building the watershed in 2D.

[24]. The gradient ∇f is the plane vector of the first-order derivatives of f, and the Hessian H_f is the real symmetric matrix of the second-order derivatives.

The definitions of watershed are based on the notion of path of greatest slope. Intuitively, a path of greatest slope is a path parallel to the gradient of f. If we begin on a point a where $\nabla f(a) \neq 0$, we can easily follow the gradient line backward (i.e. the grey levels are decreasing along the line) until we reach a point b where $\nabla f(b) = 0$. But b is not neceassarily a regional minima of f and thus there is an ambiguity to continue the path after b. We propose to formalize this notion by using the notion of maximal line of the gradient. A path of greatest slope will then be a union of maximal line of the gradient.

Definition 2.4. A path $\gamma :]-\infty, +\infty[\to \mathbb{R}^2$ is called a *maximal line of the gradient* if

$$\forall s \in]-\infty, +\infty[, \dot{\gamma}(s) = \pm \nabla f(\gamma(s)) \neq 0 \quad \text{and}$$

$$\lim_{s \to -\infty} \dot{\gamma}(s) = \lim_{s \to +\infty} \dot{\gamma}(s) = 0. \tag{4}$$

We shall say that a maximal line of the gradient is *descending* if

$$\forall s \in]-\infty, +\infty[, \dot{\gamma}(s) = -\nabla f(\gamma(s)). \tag{5}$$

The maximal lines of the gradient are defined on $]-\infty, +\infty[$ for one cannot reach a point with a zero speed. This is due to the parametrization (the speed at which we run on the path) we have chosen. The magnitude of this speed is equal to the gradient modulus. We could have chosen another parametrization of the path, but we use it as it expresses the fact that we cannot *clearly* extend a maximal line of the gradient. Note that the union of all maximal lines of the gradient of a continuous function covers the whole domain of the function (as an example, see Fig. 4).

We recall that a is a critical point if $\nabla f(a) = 0$. We need to link the maximal lines of the gradient if we want them to end in a regional minima. We are going to define a partial ordering relation which will allow us to do so.

Definition 2.5. Let a and b be two critical points of f. We shall say that b is *above* a if there exists

a maximal descending line of the gradient linking b to a. We can extend this notion by saying that b is *above* a if there exists a set (a_i) of critical points such that $a_0 = b$ and $a_n = a$, satisfying a_i is above a_{i+1}. We then obtain the partial ordering relation '*above*'.

This ordering relation allows to distinguish three kinds of critical points:
- regional minima,
- points above an unique minima,
- points above several minima.

The last ones should clearly belong to the definition of the watershed of a continuous function.

Definition 2.6. We denote $\mathscr{P}(f)$ the subset of the critical points a of f which are above several regional minima of f.

Fig. 3 shows some examples of points of $\mathscr{P}(f)$.

We have the following convergence theorem which will be used in the following as our definition of the continuous watershed.

Theorem 2.7. *[17] Let f be a $\mathscr{C}^2$ function, with a compact connected domain. Suppose that f has only isolated critical points, and that, on the critical points, the Hessian has two non-zero eigenvalues. We construct a sequence f_n of step functions which converges pointwise towards f. More precisely, we put $f_n(a) = E(2^n f(a))/2^n$, where $E(x)$ is the integer part of x. Then, the watershed of f, seen as the limit of the watershed of f_n, is the set of the maximal lines of the gradient linking two points of $\mathscr{P}(f)$.*

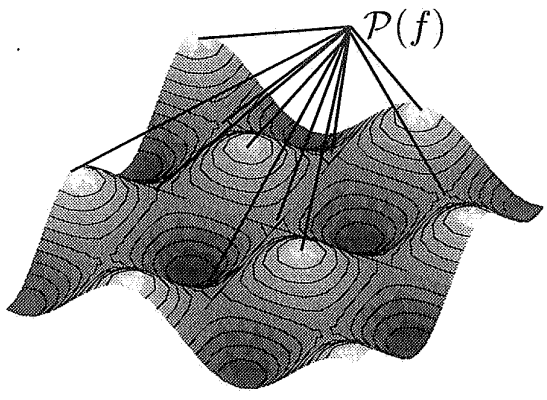


Fig. 3. Some examples of points of $\mathscr{P}(f)$.

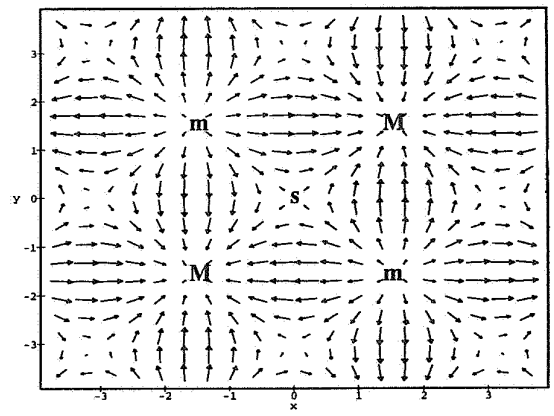


Fig. 4. Gradient field of $\sin(x)\sin(y)$ (M: maxima – m: minima – s: saddle point).

Note that the limit function f does not have plateaus. In fact, we will show in the next section that the choice of a line of watershed on a plateau is arbitrary.

So the watershed lines are parallel to the gradient, which, as far as we know, has not been pointed out by other authors. Note that we can use Theorem 2.7 as a *definition* of the watershed of a continuous function. One should be very careful with the hypothesis we put on f. If Hessian H_f has one zero eigenvalue, the watershed can have a barb (a branch with an endpoint), which is never the case with the algorithm. As an example, one can look at a monkey saddle (Fig. 5). Two branches of the monkey saddle belongs to the contour of a basin. The other one directs itself towards a local maximum of the image which belongs to the interior of a basin. Moreover, the watershed of a continuous function can be thick. As an example, we can look for what Beucher [2] calls a buttonhole (see Fig. 6). Choosing a particular line in the buttonhole is anyway fully arbitrary. Note that, on the buttonhole, $\nabla f(a) = 0$ yields $H_f(a) = 0$. Nevertheless, we can always assume that the images satisfy the

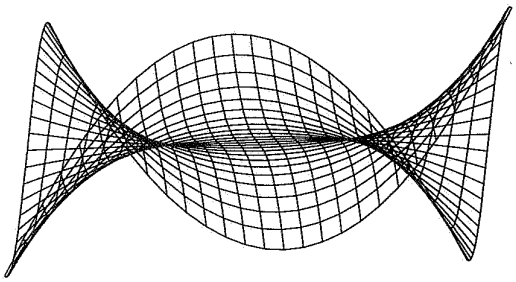


Fig. 5. Monkey saddle.

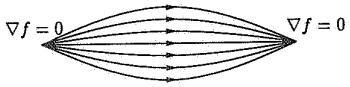


Fig. 6. Gradient field of a buttonhole.

hypothesis of Theorem 2.7: if f does not satisfy it, then an arbitrary small perturbation of f makes f an adequate function (obviously the perturbation must not vanish on the critical points). Such functions are called *Morse functions* [16], and are (uniformly) dense in the set of smooth functions.

The fact that at each point of a $\mathscr{C}^2$ function there exist at least two lines of greatest slope descending towards two minima is true only on points of $\mathscr{P}(f)$ (see Fig. 4).

Theorem 2.7 suggests that we can add some lines to the watershed by adding points to $\mathscr{P}(f)$. If we carefully choose these new points, the result exhibits end points and is a way to introduce the notion of grey-tone skeleton. In the last part of this paper we give an algorithm which allows to do such an operation.

To define the watershed as a subset of the maximal lines of the gradient is a local notion, to which boundary conditions add a global aspect. More precisely, we have the following proposition.

Proposition 2.8. *[17] Let a be a point of the domain of f such that* $\nabla f(a) \neq 0$. *Let* $\mathscr{V}_a$ *be a neighbourhood of a which does not contain any critical point. Let* γ *be a path containing a and parallel to the gradient of f on* $\mathscr{V}_a$. *Then there exists a function* f_0, *equal to f on* $\mathscr{V}_a$, *such as* γ *is in the watershed of* f_0.

In other words, there is no local characterization of the watershed. This is due to the $\mathscr{C}^2$ regularity of f. If f is less regular, there exists in some case a local characterization. The best example is the watershed of $f(a) = d(a, X)$ where X is a binary image. In this case the watershed of f is equal to the skeleton by influence zones of X. As shown by Matheron [23], it is locally characterized by the set of the points of non-differentiability of f; the watershed is included in this set of points but not always equal to.

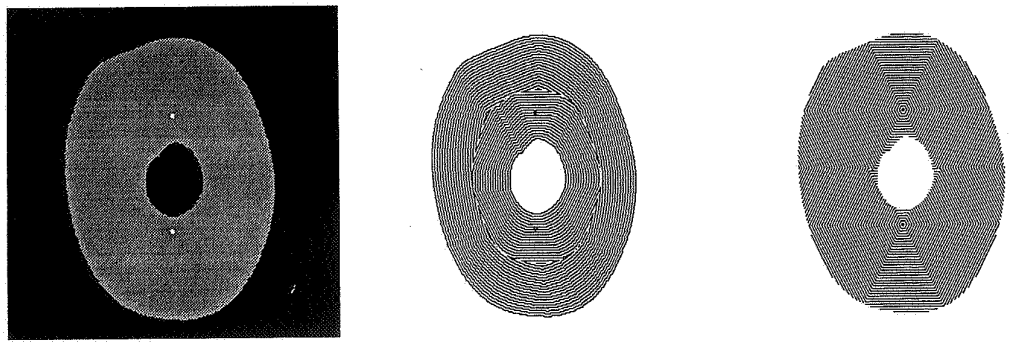
2.3. The problem of the plateaus

Real images often possess plateaus. When they belong to the interior of a catchment basin, there is no problem. From a theoretical point of view, it is easy to slope the plateaus towards crest lines without modifying the watershed.

On the contrary, if we want a thin watershed, we have to make a choice on the plateaus. The immersion algorithm 2.3 chooses a line by computing the 'geodesic middle' of the plateaus. The principle is to compute the geodesic distance to the descending side of each plateau (Fig. 7(b)). There exists another possibility: it is possible to compute the geodesic distance to the ascending side of each plateau (Fig. 7(c)). These two possibilities give different results, and the choice of one rather than the other is arbitrary and depends on the application. We give here some examples illustrating the choice of the distance.

Let us take the image of a ring, corresponding to the contours of an object. The watershed of this image reduces the ring to the middle line, thus creating two equal parts, which seems reasonable. Nevertheless, there are some situations where we want the segmentation to pass through another place.

Let us consider Figs. 8(a) and (d). They are the ring image on which we have added some new contour points through which we want the watershed to pass. In the first case (Fig. 8(a)), these points are near the exterior side, and in the second case (Fig. (8d)), these points are near the interior side. The watershed immersion algorithm floods



(a) Original image

(b) Distance to the descending side

(c) Distance to the ascending side

Fig. 7. Distances to the sides of a ring with two elevated points near the interior side.

the image of the distance to the descending side (Figs. 8(c) and (f)) which do not seem a judicious choice.

If we transform the image by computing the distance to the ascending side (the distance to the added points), the watershed result is then conformed to the intuition in Fig. 8(b). There exists no reasonable criterion which allows a choice between Figs. 8(e) and (f); there are not enough points to place the contour correctly.

These examples are a good illustration that the implicit choice of the distance to the descending side is fully arbitrary, for it can be judicious to choose the distance to the ascending side. The distance to the descending side is nevertheless the right choice in the case where we want the watershed to pass exactly at the middle of the plateaus.

Let us notice that the use of the distance to the descending side is unstable: it can add some new local minima, thus modifying the topology of the watershed. As an example, there exists a tendency to close arcs of circle (Fig. 9). On the other side, the distance to the descending side creates new local maxima, but this does not disturb the watershed.

3. Comparison with the edge detectors

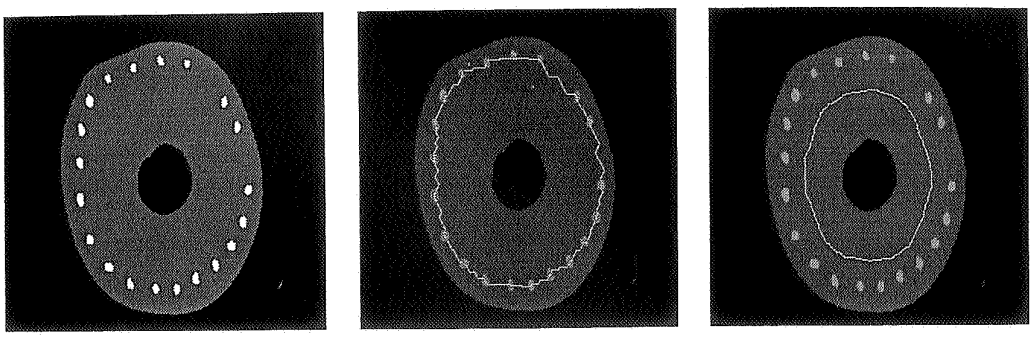
Let f be a smooth function. Two second-order differential operators are commonly used to detect edges. The first one is the Laplacian [14] $\Delta f = (\partial^2 f/\partial x^2) + (\partial^2 f/\partial y^2)$ and the second one is the non-linear Canny's detector [5] which looks for the maximum of the gradient in the direction of the gradient. The Canny's detector, or more exactly the extrema of the gradient in the direction of the gradient, finds the zero crossings of $Q(f) = \langle H_f \nabla_f, \nabla f \rangle$. On the other hand, mathematical morphology uses the watershed of the norm of the gradient of f in order to extract edges. The links between the two differential operators are well known [25] and we focus our attention on the link between Canny's detector and the watershed of the gradient.

In [11] the authors exhibit a characterization of lines extracted by Canny's detector. Their results are useful to point out the differences between the watershed of the gradient and the second-order differential operators. There are two archetypes of structure on which Q vanishes: the step edge and the pitched roof. These two objects are represented n Fig. 10, before the gaussian convolution which make them smooth.

The idea behind edge detection is that an edge is a path where the change in the intensity f is maximum in the direction normal to this path. As the intensity is computed by the modulus of the gradient, we can write

$$\frac{\mathrm{d}}{\mathrm{d}t} \|\nabla f(a + tn)\| = \langle H_f \nabla f, n \rangle = 0, \tag{6}$$

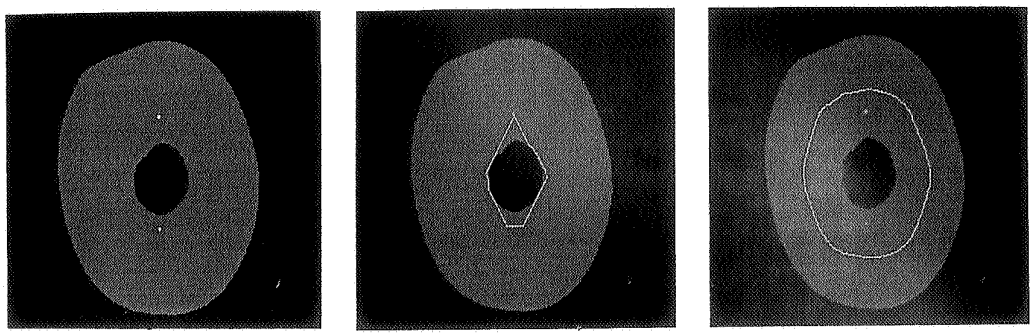
where n is the normal to the path at point a. This



(a) Summits are near the exterior side

(b) Watershed on distance to the ascending side

(c) Watershed on distance to the descending side



(d) Summits are near the interior side

(e) Watershed on distance to the ascending side

(f) Watershed on distance to the descending side

Fig. 8. Plateaus' problem: the choice of a well-placed contour.

equation gives an implicit differential equation for the edge path: $\dot{\gamma} = H_f \nabla f$. As we have seen before, the union of all the paths which are solutions to this differential equation covers the whole domain of f, and we have to make a choice to find the edges. The watershed chooses the paths by imposing boundary conditions. On the other hand, Canny solves the problem by *estimating* the normal n from the gradient direction, i.e. by setting $n = \nabla f$ (which is true on a step edge).

We made a comparison of the action of these operators on the image

$$I(x, y) = \delta_1 \chi_{\{x>0\}}(x, y) + \delta_2 \chi_{\{y>0\}}(x, y), \tag{7}$$

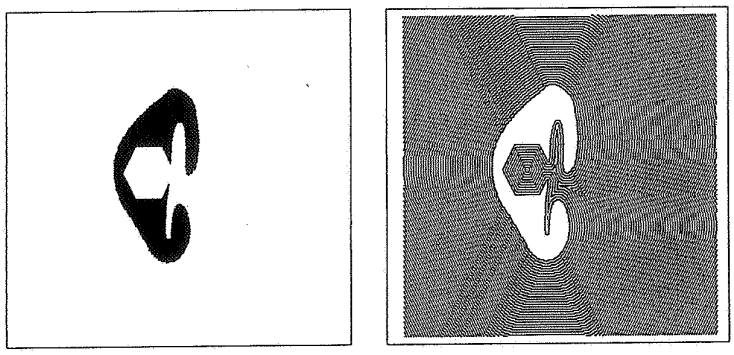
where χ_A is the characteristical function of the set A: $\chi_A(a) = 1$ if $a \in A$ and $\chi_A(a) = 0$ if not. I is regularised by a gaussian kernel G and we obtain

$$f = G * I = \delta_1 \Psi(x) + \delta_2 \Psi(y), \tag{8}$$

where $\Psi(x) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{x} e^{-s^2} ds$.

Fig. 11 shows the comparison of the segmentation algorithms. We see that Canny's detector finds the multiple point only if $\delta_1 = \delta_2$ (Fig. 11(f)).

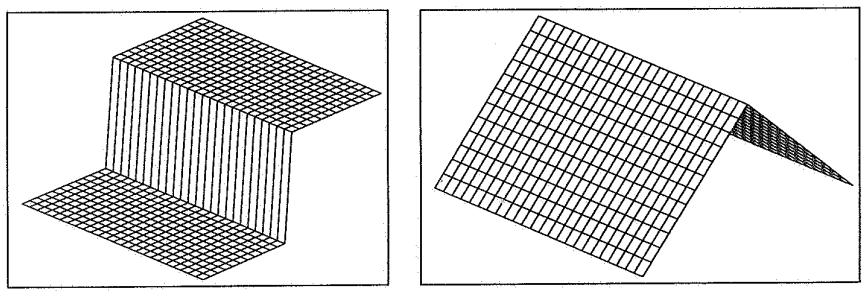
Fig. 12 shows the results: the second-order operators cannot find the multiple point, while the watershed of the gradient modulus can. This is due to the geometric behaviour of Canny's detector and zeros of the Laplacian. Both are the intersection of a function $z = g(x, y)$ with $\{z = 0\}$. So, they have very few multiple points. On the other hand, the watershed has multiple points which are necessarily in $\mathscr{P}(\|\nabla f\|)$.



(a) Example of a non closed arc of circle

(b) The distance to the ascending side creates a new catchment basin

Fig. 9. Example of creation of a new catchment basin through the distance to the ascending side.



(a) Step edge

(b) Pitched roof

Fig. 10. Structural archetype detected by Canny's operator.

Several technics [7, 8] have been recently developed to detect the multiple points. They are based on second-order differential measures and scale-space approach. In fact, we propose a simpler method: if we are interested in finding multiple points, the classical differential crest extractor (local maxima of the modulus of the gradient in the direction of the gradient) has to be replaced by a watershed procedure on the image of gradient modulus.

4. Metrical approach of the watershed

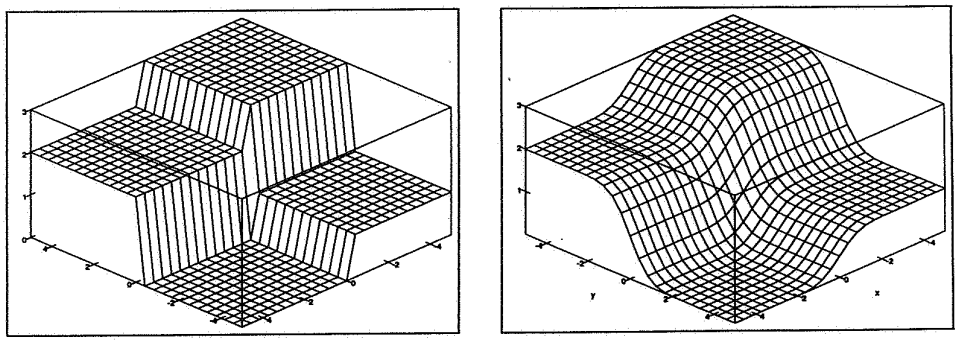
The aim of this section is to exhibit the strong link between the skeleton, one of the notions of the binary mathematical morphology, and the watershed, notion of the grey-level mathematical morphology.

Definition 4.1. The *image distance* on a $\mathscr{C}^1$ function f with a connected domain $\mathrm{Dom}(f)$, is defined by $\forall (a, b) \in \mathrm{Dom}(f)^2$,

$$d_f(a,b) = \inf_{\gamma_{ab}} \left| \int_{\gamma_{ab}} \|\nabla f(\gamma_{ab}(s))\| \, \mathrm{d}s \right|. \tag{9}$$

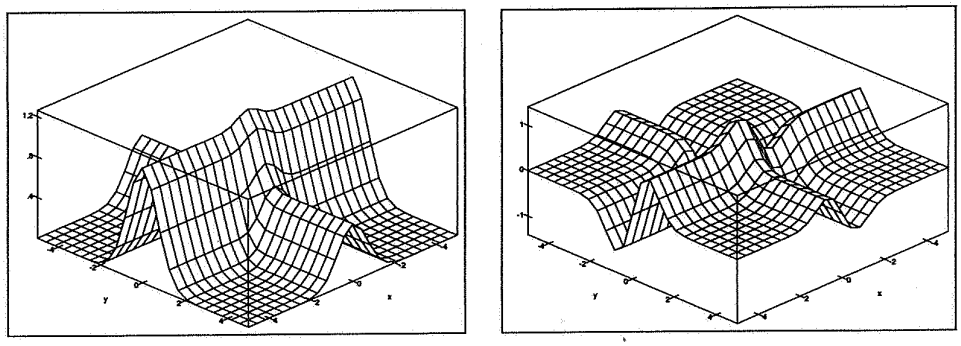
Note that the shortest d_f-path between a and b is a path of greatest slope if it exists.

We restrain the choice of f to the $\mathscr{C}^2$ functions which have only isolated critical points. d_f is then



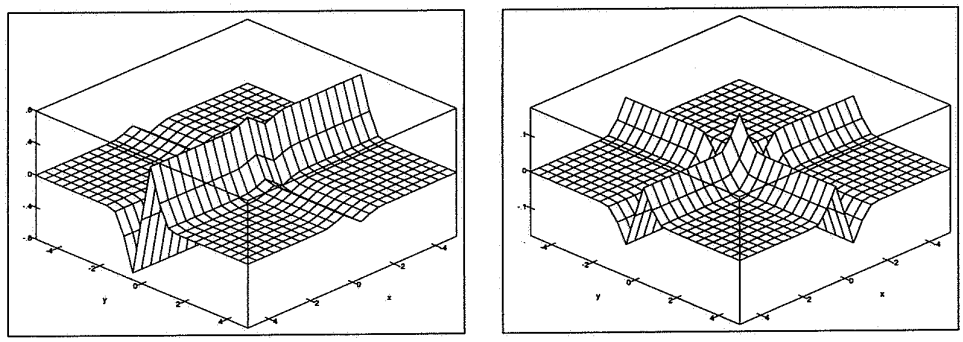
(a) Original image

(b) Convolution of the image with a gaussian



(c) Gradient norm

(d) Laplacian graph

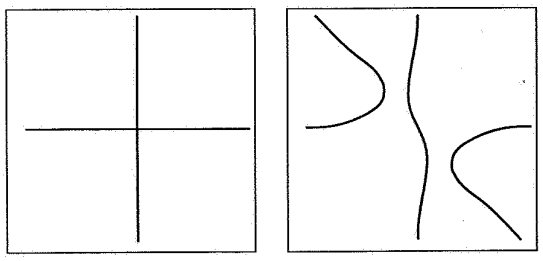


(e) Graph of $\langle H_f \nabla f, \nabla f \rangle$ $(\delta_1 < \delta_2)$

(f) Graph of $\langle H_f \nabla f, \nabla f \rangle$ $(\delta_1 = \delta_2)$

Fig. 11. Comparison of the segmentation algorithms.

a distance. For technical reasons, but without loss of generality, we suppose that the minimum of f are on the same level. Moreover, we suppose that on the critical points, the Hessian has two non-zero eigenvalues. We then have the following result:



(a) Outlines found by watershed (b) Solution of $Q(f) = 0$

Fig. 12. Outlines found by the algorithms.

Theorem 4.2. *[17] The set of points which are at equal d_f-distance of two distinct minima of f is the set of the maximal lines of the gradient linking two points of $\mathscr{P}(f)$, and thus coincides with the watershed of f.*

A similar result has been stated in [18, 19], but with a much more complex metric used as a definition for the continuous watershed, and this complex metric is very far from the usual euclidean distance. Note that the watershed defined by Lantuéjoul and Beucher [3], the one defined by Maisonneuve [13], the one defined by Preteux and Merlet, and the d_f-skeleton by influence zones do have barbs if f does not verify the hypothesis we put on it.

The advantage of our metric is to allow the statement of new results we present hereafter. Moreover, note that if we put $\|\nabla f\| = 1$ a.e., d_f (or f) is the usual euclidean distance function to a set, and the d_f-skeleton by influence zones is the usual one.

With the results of the previous theorem, we can expect that the watershed has properties similar to those of the skeleton. We state one of those properties.

We denote by $\mathscr{M}$ the set of minima of f and by $\mathrm{Dom}(f)$ the domain of f.

Definition 4.3. We define the *skeletal structure* of f as the set of centres of the maximal open d_f-balls contained in $\mathrm{Dom}(f)\backslash\mathscr{M}$.

Obviously, the watershed is contained in the skeletal structure.

Theorem 4.4. *[17] The mapping $f \to \mathscr{S}(f)$, where $\mathscr{S}(f)$ is the skeletal structure of f, is lower semicontinuous, if we use the $\mathscr{C}^2$ convergence on the set of functions and the induced hit or miss topology [15] on the set of watersheds.*

This result shows why the watershed is very sensitive to noise, and justifies in a way the various smoothing [9] and marking [28] techniques.

5. The eikonal equation

As a side effect, the algorithm of the watershed can be adapted to solve an equation widely used in shape from shading, the eikonal equation [10]:

$$\text{finding } f \text{ such as } \|\nabla f\| = g. \tag{10}$$

The idea is that, on each catchment basin of f, we have $f(a) = d_f(a, b) + f(b)$, where b is the minima of the catchment basin. As determining $d_f(a, b)$ only depends on g (see Eq. (9)), we can generalize this result: let $\{b_i\}$ be a set of points with their associated values $f(b_i)$. A continuous solution to the eikonal equation is given by

$$f(a) = \inf_i \left\{ \inf_{\gamma_{ab_i}} \int_{\gamma_{ab_i}} g(\gamma_{ab_i}(s))\,\mathrm{d}s + f(b_i) \right\}. \tag{11}$$

One can show [12] that function f given by Eq. (11) is the unique viscosity solution satisfying $\|\nabla f\| = g$ on an open domain Ω, which is a viscosity supersolution on $\partial\Omega$.

The algorithm proposed by Vincent [28] can be adapted to compute this solution. In fact, Vincent's algorithm splits up in a first stage of sorting the pixels by increasing grey level, and a second stage of flooding propagation threshold by threshold. In our case, we cannot make the first stage, but we can do the sorting during the flooding. This is easy to do if we use an heapsort algorithm [1]. We then obtain a kind of algorithm similar to the one developed by Verwer and Verbeek [27,26].

Note that this technique can also be used to compute the watershed by flooding from selected sources, an efficient tool to prevent the oversegmentation problem.

6. Towards grey-tone skeleton

Theorem 2.7 suggests the possibility of adding some points to $\mathscr{P}(f)$, and thus some lines to the watershed. This lines will then figure the contours inside the basins. We give a new algorithm to compute the watershed which allows to do this.

This algorithm is based on Vincent's one for building skeleton with anchor points [28]. The idea is to do an homotopic thinning of each grey level of the image. We then compute the skeleton by influence zones of the level $h-1$ in the level h by homotopic thinning. This algorithm gives us a watershed.

This implementation has two advantages. The first one is that it is easy to add anchor points; it is enough to prevent the suppression of some points. The second one is that instead of flooding from the minima of the image, we can flood from any marker we want.

Fig. 13 shows an example of application of this new algorithm.

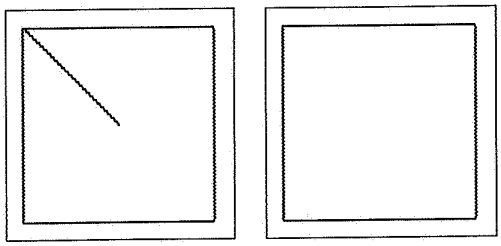
The main problem is the choice of the anchor points. A first idea is to use the segmentation provided by a classical edge detector, as Canny's one. We then have the multiple points which cannot be found by those algorithms, and we keep the edge lines interior to the basins.

Another idea of anchor points is to generalize the notion of binary skeleton. An interesting class of anchor points is the set of the centres of osculating circles to the descending sides of plateaus. On a binary image, these points are in the skeleton of the form. If we transform our image by computing the distance to the descending side of the plateaus, the set of the points of the following (hexagonal) configuration:

```
  <   <
<   .   <
  ?   ?
```

(where $<$ marks a pixel the value of which is lower than the value of the central pixel, and ? means that this value is unimportant) contains these centres, and is thus a good choice for the anchor points. In particular, if we use this procedure on the distance function of a binary image, we obtain the usual skeleton of thickness one.

Fig. 14 shows various possibilities of segmentation on a face sideview image (Fig. 14(b)). The first one (Fig. 14(a)) is the classical Canny–Deriche edge detector [6]. The second one (Fig. 14(b)) is obtained by the classical watershed algorithm applied on the gradient image. The result is very noisy, but contains all the useful information. The third one (Fig. 14(d)) is obtained by the classical watershed algorithm applied on a geodesical reconstruction of gradient image of size 10, which is an useful technique to suppress a lot of unimportant local minima [9]. The fourth and last one (Fig. 14(e)) is obtained by the new watershed algorithm applied on a geodesical reconstruction of gradient image of size 10, using the Canny–Deriche' edges as anchor points.



(a) Original image and result of the new watershed algorithm with the central points as anchor point

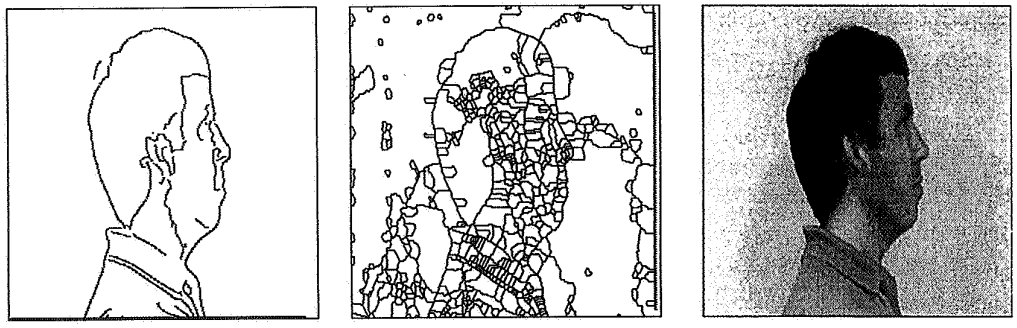
(b) Watershed on the image

Fig. 13. An example of application of the new watershed algorithm.

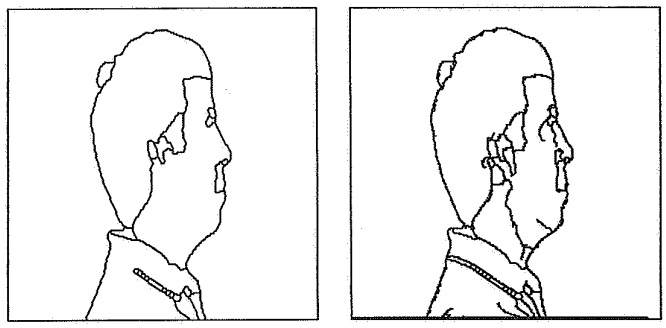
7. Conclusion

This paper is mainly devoted to the convergence and the adaptation to the continuum of an algorithm defined for step functions. This gives us a mathematical tool which links the watershed to the notion of line of greatest slope (maximal line of the gradient), and to the notion of skeleton by influence zones. The associated skeleton is then lower semicontinuous.

The watershed is compared with the classical edge detectors, and we showed that the watershed



(a) Canny-Deriche edges (b) Watershed edge (c) Original image



(d) Edges by watershed on a geodesical reconstruction of the gradient of size 10

(e) Edges by watershed on a geodesical reconstruction of the gradient of size 10, using Canny-Deriche's edges as anchor points

Fig. 14. Comparison of various possibilities of segmentation.

can extract multiple points, an operation that second order differential operators are unable to perform.

On the algorithmic side, the watershed characterization gives an original interpretation of the eikonal equation, and opens the path towards grey-tone skeletons. The powerful watershed-by-flooding algorithm offers efficient adaptations to these various notions.

References

[1] A. Aho, J. Hopcroft and J. Ullmann, *Data Structures and Algorithms*. Addison-Wesley, MA, 1983.

[2] S. Beucher, Segmentation d'images et morphologie mathématique, Thèse École Nationale Supérieure des Mines de Paris, June 1990.

[3] S. Beucher and Ch. Lantuéjoul "Use of watersheds in contour detection", in: *Proc. Internat. Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, September 1979.

[4] S. Beucher and F. Meyer, "The morphological approach to segmentation: The watershed transformation", in: E.R. Dougherty, ed., *Mathematical Morphology in Image Processing*, Optical engineering, Marcel Dekker, New York 1993, pp. 433–482.

[5] J.F. Canny, "A computational approach to edge detection", in: M.A. Fischler and O. Firschein, eds., *Reading in Computer Vision: Issues, Problems, Principles and Paradigms*, Morgan Kaufmann, CA, 1986, pp. 184–203.

[6] R. Deriche, "Using Canny's criteria to derive a recursive implemented optical edge detector", *Internat. J. Comput. Vision*, 1987, pp. 167–187.

[7] G. Giraudon and R. Deriche, Accurate corner detection: An analytical study, Technical Report 1420, INRIA, April 1991.
[8] G. Giraudon and R. Deriche, On corner and vertex detection, Technical Report 1439, INRIA, June 1991.
[9] M. Grimaud, La géodésie numérique en morphologie mathématique: Application à la détection automatique de microcalcifications en mammographie numérique, Thesis, École des Mines de Paris, December 1991.
[10] B.K.P. Horn, *Robot Vision*, MIT Eng. Comput. Sci. Ser., MIT Press, McGraw-Hill, New York, 1986.
[11] W.M. Krueger and K. Phillips, "The geometry of differential operators with application to image processing", *IEEE Pattern Anal. Mach. Intell.*, Vol. 11, December 1989, pp. 1252–1264.
[12] P.L. Lions, E. Rouy and A. Tourin, "Shape-from-shading, viscosity solutions and edges", *Numer. Math.*, Vol. 64, 1993, pp. 323–353.
[13] F. Maisonneuve, Sur le partage des eaux, Technical Report, CGMM, École des Mines de Paris, 1982.
[14] D. Marr and E.C. Hildreth, "Theory of edge detection", in: *Proc. Roy. Soc. Lond. B*, Vol. 207, 1980, pp. 187–217.
[15] G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York, 1975.
[16] J. Milnor, *Morse Theory*, Princeton University Press, 1963.
[17] L. Najman and M. Schmitt, "Quelques caractérisations de la ligne de partage des eaux d'une fonction continue", Technical Report ASRF-92-4, L.C.R., August 1992.
[18] F. Prêteux, "On a distance function approach for gray-level mathematical morphology", in: E.R. Dougherty, ed., *Mathematical Morphology in Image Processing*, Optical engineering, Marcel Dekker, New York, 1993, pp. 323–350.
[19] F. Prêteux and N. Merlet, "New concept in mathematical morphology: The topographical and differential distance functions", in: *Image Algebra and Morphological Image Processing II*, Vol. 1568, San Diego, CA, July 1991, pp. 66–77.
[20] M. Schmitt, "Geodesic arcs in non-Euclidean metrics: Application to the propagation function", *Revue d'Intelligence Artificielle*, Vol. 3, No. 2, 1989, pp. 43–76.
[21] M. Schmitt and L. Vincent, *Morphological Image Analysis: A Practical and Algorithmic Handbook*, Cambridge University Press, 1994, to appear.
[22] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
[23] J. Serra, ed., *Image Analysis and Mathematical Morphology, Volume 2: Theoretical Advances*, Academic Press, London, 1988.
[24] M. Spivak, *A comprehensive Introduction to Differential Geometry*, Vol. 2, Publish or Perish, Houston, TX, 2nd Edition, 1979.
[25] V. Torre and T. Poggio, "On edge detection", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 8, March 1986, pp. 147–163.
[26] B.J.H. Verwer, Distance transforms: metrics, algorithms and applications, Ph.D. thesis, Technical University of Delft, The Netherlands, 1991.
[27] B.J.H. Verwer and P.W. Verbeek, "Shading from shape, the eikonal equation solved by grey-weighted distance transform", *Pattern Recognition Lett.*, Vol. 11, October 1990, pp. 681–690.
[28] L. Vincent, Algorithmes morphologiques à base de files d'attente et de lacets: Extension aux graphes, Thèse École des Mines de Paris, May 1990.
[29] L. Vincent, "Morphological algorithms", in: E.R. Dougherty, ed., *Mathematical Morphology in Image Processing*, Optical engineering, Marcel Dekker, New York, 1993, pp. 255–288.
[30] 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, 1991, pp. 583–598.

3 Watersheds, Mosaics and the Emergence Paradigm

L. Najman, M. Couprie and G. Bertrand. Watersheds, Mosaics and the Emergence Paradigm. *Discrete Applied Mathematics*, Volume 147, Issues 2-3, 15 April 2005, Pages 301-324.
Special Issue on DGCI.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 147 (2005) 301–324
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Watersheds, mosaics, and the emergence paradigm

Laurent Najman[a, b], Michel Couprie[a, b], Gilles Bertrand[a, b]

[a] *Laboratoire A2SI, Groupe ESIEE BP99, 93162 Noisy-le-Grand Cedex, France*
[b] *IGM, Unité Mixte de Recherche CNRS-UMLV-ESIEE UMR 8049*

Received 15 December 2003; received in revised form 1 June 2004; accepted 3 September 2004
Available online 24 December 2004

Abstract

In this paper, we investigate the links between the flooding paradigm and the topological watershed. Guided by the analysis of a classical flooding algorithm, we present several notions that lead us to a better understanding of the watershed: minima extension, mosaic, pass value and separation. We first make a detailed examination of the effectiveness of the divide set produced by watershed algorithms. We introduce the mosaic to retrieve the altitude of points along the divide set. A desirable property is that, when two minima are separated by a crest in the original image, they are still separated by a crest of the same altitude in the mosaic. Our main result states that this is the case if and only if the mosaic is obtained through a topological thinning. We investigate the possibility for a flooding to produce a topological watershed, and conclude that this is not feasible. This leads us to reverse the flooding paradigm, and to propose a notion of *emergence*. An emergence process is a transformation based on a topological criterion, in which points are processed in decreasing altitude order while preserving the number of connected components of lower cross-sections. Our main result states that any emergence watershed is a topological watershed, and more remarkably, that any topological watershed of a given image can be obtained as an emergence watershed of the image.

Keywords: Mathematical morphology; Topology; Watersheds; Mosaic; Topological watershed; Flooding

E-mail addresses: l.najman@esiee.fr (L. Najman), m.couprie@esiee.fr (M. Couprie), g.bertrand@esiee.fr (G. Bertrand).
URL: http://www.esiee.fr/∼ coupriem/Sdi/ (M. Couprie).

0166-218X/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2004.09.017

1. Introduction

The watershed has been extensively studied during the 19th century by Maxwell [17] and Jordan [15] among others. One hundred years later, the watershed transform was introduced by Beucher and Lantuéjoul [4] for image segmentation, and is now used as a fundamental step in many powerful segmentation procedures [20,5]. Image segmentation usually requires several processing steps. For example, a typical morphological segmentation procedure includes a filtering step, a gradient, a marker extraction or a reduction of the number of minima, a watershed step and some post-processing. Most of these steps are often very dependent on the application, only the watershed step is application independent. In this paper, we focus exclusively on watersheds and we study some mathematical properties of several discrete watershed operators.

A popular presentation of the watershed in the morphological community [32,14,12] is based on a flooding paradigm. Let us consider the greyscale image as a topographical relief: the grey level of a pixel becomes the elevation of a point, the basins and valleys of the relief correspond to the dark areas, whereas the mountains and crest lines correspond to the light areas. Let us suppose the surface being immersed in a lake, with holes pierced in local minima. Water fills up basins starting at these local minima, and, at points where waters coming from different basins would meet, dams are built. As a result, the surface is partitioned into regions or basins separated by dams, called watershed divides.

Efficient watershed algorithms based on immersion simulation were proposed by Vincent, Soille [36] and Meyer [18] in the early 90s. Those algorithms build a partition of the space by associating an influence zone to each minimum of the image, and by producing (in their "dividing" variant) a divide set which separates those influence zones; that is to say, they "extend" the minima. The building of the influence zones is based on a flooding paradigm which consists in processing points of the image in increasing grey level order. We can find a presentation of most of the existing morphological watershed algorithms in a paper by Roerdink and Meijster [27]. Nevertheless, to our best knowledge, no attempt has been made to propose comparison criteria. Let us note that a mathematical approach for regular continuous functions has been proposed by Najman and Schmitt [23,24], introducing in particular the equivalence for regular functions between the flooding approach and a distance-based approach to the watershed. Algorithms for computing distance-based watersheds have been proposed in [19]. Such distance-based or cost-based [16] watersheds will not be studied in this paper.

An original approach to the watershed transform, called the topological watershed, has been proposed in [7]. The idea is to define a "topological thinning" that transforms the image while preserving some topological properties, namely the number of connected components of each lower cross-section. Let F be a greyscale image and λ be a grey level, the lower cross-section $\overline{F_\lambda}$ is the set composed of all the points having an altitude strictly lower than λ. A point x is said to be W-destructible for F (where W stands for Watershed) if its altitude can be lowered by one without changing the number of connected components of $\overline{F_k}$, with $k = F(x)$. A map G is called a W-thinning of F if it may be obtained from F by iteratively selecting a W-destructible point and lowering it by one. A topological watershed of F is a W-thinning of F which contains no W-destructible point (see Fig. 1a,c). A major feature of this transform is to produce a greyscale image. A divide set of the original image can

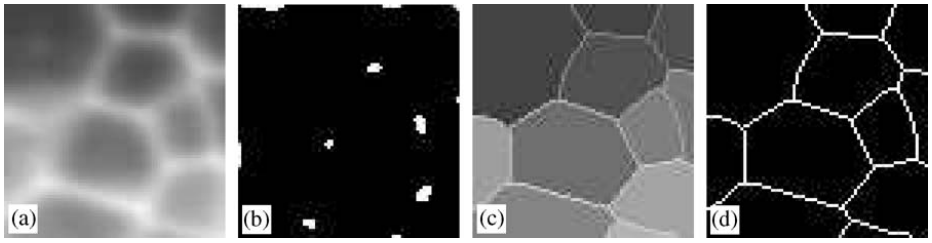


Fig. 1. (a) Original image; (b) regional minima of (a) (in white), (c) a topological watershed of (a), (d) a divide set of (a), obtained by taking the complement of the regional minima of (c).

easily be computed on the transformed image, by taking the complement of the minima of the transformed image (see Fig. 1d). Recently, Bertrand [1] proposed a framework in which fundamental properties of the topological watershed have been derived. Quasi-linear algorithms for computing the topological watershed transform have been obtained and proved using this framework [10].

In this framework, a notion of contrast plays an important role. We will say informally that a transformation "preserves the contrast" if the transformation preserves the altitude of the minima of the image and if, when two minima are separated by a crest in the original image, they are still separated by a crest of the same altitude in the transform. The formal definition relies on the altitude of the lowest pass which separates two minima, named pass value. One of the main results obtained in [1] states that any topological thinning preserves the contrast (in this sense), and that any transformation that preserves the contrast is a topological thinning.

One of the goals of this paper is to examine the links between the flooding paradigm and the topological watershed. In the first part of this paper, guided by the analysis of a classical flooding algorithm, we present some notions that lead us to a better understanding of the watershed: minima extension, mosaic, pass values and separation (see also [21]). A mosaic image is obtained from an image F and a divide set D of F by valuating the points of D with the corresponding values of these points for F. We prove in particular that a mosaic "preserves the contrast" *if and only if* the mosaic is obtained through a topological thinning. We investigate the possibility for a flooding to produce a topological watershed, and we propose a monotone flooding transformation that preserves the number of connected components of each lower cross-section. We show that this monotone flooding does not always produce a topological watershed.

This leads us to the paradigm of emergence: reversing the flooding paradigm, we start with the highest level first. We call emergence watershed a transformation that lowers points in decreasing altitude order while preserving the number of connected components of lower cross-sections. Our main result states that an emergence watershed is a topological watershed, and more remarkably, that any topological watershed of a given image can be obtained as an emergence watershed of the image.

2. Basic notions and notations

Many fundamental notions related to watersheds in discrete spaces can be expressed in the framework of graphs.

Let E be a finite set of vertices (or points), and let $\mathscr{P}(E)$ denote the set of all subsets of E. Throughout this paper, Γ denotes a binary relation on E, which is reflexive ($(x, x) \in \Gamma$) and symmetric ($(x, y) \in \Gamma \Leftrightarrow (y, x) \in \Gamma$). We say that the pair (E, Γ) is a *graph*. We also denote by Γ the map from E to $\mathscr{P}(E)$ such that, for all $x \in E$, $\Gamma(x) = \{y \in E | (x, y) \in \Gamma\}$. For any point x, the set $\Gamma(x)$ is called the *neighborhood of* x. If $y \in \Gamma(x)$ then we say that x and y are *adjacent*.

Let $X \subseteq E$ We denote by $\overline{X}$ the complement of X in E. Let $x_0, x_n \in X$. A *path from* x_0 *to* x_n *in* X is a sequence $\pi = (x_0, x_1, \ldots, x_n)$ of points of X such that $x_{i+1} \in \Gamma(x_i)$, with $i = 0 \ldots n-1$. Let $x, y \in X$, we say that x and y are *linked for* X if there exists a path from x to y in X. We say that X is *connected* if any x and y in X are linked for X. We say that $Y \subseteq E$ is a *connected component of* X if $Y \subseteq X$, Y is connected, and Y is maximal for these two properties (i.e., $Y = Z$ whenever $Y \subseteq Z \subseteq X$ and Z is connected). In the following, we assume that the graph (E, Γ) is connected, that is, E is made of exactly one connected component.

We denote by $\mathscr{F}(E)$ the set composed of all maps from E to $\mathbb{Z}$. A map $F \in \mathscr{F}(E)$ is also called an *image*, and if $x \in E$, $F(x)$ is called the *altitude of* x (*for* F). Let $F \in \mathscr{F}(E)$. We write $F_k = \{x \in E | F(x) \geqslant k\}$ with $k \in \mathbb{Z}$; F_k is called an *upper* (*cross-*) section of F, and $\overline{F_k}$ is called a *lower (cross-) section of* F. A non-empty connected component of a lower section $\overline{F_k}$ is called a (*level* k) lower-component of F. A level k lower-component of F that does not contain a level $(k-1)$ lower-component of F is called a (*regional*) *minimum of* F.

A subset X of E is *flat for* F if any two points x, y of X are such that $F(x) = F(y)$. If X is flat for F, we denote by $F(X)$ the altitude of any point of X for F.

3. The flooding paradigm

The flooding paradigm corresponds to the intuitive idea of immersion described in the second paragraph of the introduction. In mathematical morphology, it was first proposed by Digabel and Lantuéjoul [11] and used for image segmentation by Beucher and Lantuéjoul [4]. Among the numerous morphological algorithms that were developed following this idea, Meyer's algorithm [18] (called *flooding algorithm* in the sequel) is probably the simplest to describe and understand. We are going to use it as a guide that will help us to introduce the questions we are studying in this paper.

3.1. The flooding algorithm

Starting from an image $F \in \mathscr{F}(E)$ and the set M composed of all points belonging to the minima of F, the flooding algorithm expands as much as possible the set M, while preserving the connected components of M. It can be described as follows:

1. Attribute to each minimum a label, two distinct minima having distinct labels; mark each point belonging to a minimum with the label of the corresponding minimum. Initialize two sets Q and V to the empty set.
2. Insert every non-marked neighbor of every marked point in the set Q;

3. Extract from the set Q a point x which has the minimal altitude, that is, a point x such that $F(x) = \min\{F(y) | y \in Q\}$. Insert x in V. If all marked points in $\Gamma(x)$ have the same label, then

   – Mark x with this label; and
   Insert in Q every $y \in \Gamma(x)$ such that $y \notin Q \cup V$;

4. Repeat step 3 until the set Q is empty.

The divide set is the complement of the set of marked points.

3.2. Illustration of the algorithm

In all the examples of the paper, we assume that the graph (E, Γ) corresponds to the 4-adjacency relation on a subset $E \subset \mathbb{Z}^2$, i.e., for all $x = (x_1, x_2) \in E$, $\Gamma(x) = \{(x_1, x_2), (x_1 + 1, x_2), (x_1 - 1, x_2), (x_1, x_2 + 1), (x_1, x_2 - 1)\} \cap E$.

Let us illustrate the behaviour of the algorithm on the example of Fig. 2a which presents an image with three minima at altitudes 0, 1 and 2.

- The minima at altitudes 2, 1, 0 are marked with the labels A, B, C respectively (Fig. 2b). All the non-marked neighbors of the marked points are put into the set Q.
- The first point which is extracted from the set Q is the point x at altitude 10, which has points marked B and C among its neighbors (Fig. 2b). This point cannot be marked.
- The next point to process is one of the points at altitude 20, for instance y (Fig. 2b). The only marked points in the neighborhood of such a point are marked with the label A, and thus y is marked with the label A (Fig. 2c), and the points at altitude 10 which are neighbors of y are put into the set Q.
- The next points to process are points at altitude 10. A few steps latter, all points at altitude 10 but x are processed, and marked with the label A (Fig. 2d).
- Then the other points at altitude 20 are processed. They are marked with the label A (Fig. 2e).
- The next points to process are those at altitude 30, and we finally obtain the set of labeled points shown in Fig. 2f. The divide set is circled in the figure.

Remark 1. We observe that the algorithm is not "monotone", in the following sense: if a point y of altitude $F(y) = k$ is extracted from the set Q, it is sometimes possible to find in the neighborhood of y a point z not already labeled such that $F(z) < k$. This point z will be the next point processed by the algorithm. Thus this algorithm does not always process points according to increasing altitude.

Remark 2. A second observation is related to the contrast of the original image: in the original image, to go from e.g., the minimum at altitude 0 to the minimum at altitude 2, one has to climb to at least an altitude of 20: indeed, there exists a contour at altitude 20 that we have to overcome. We observe that this contour is not present in the divide set produced by

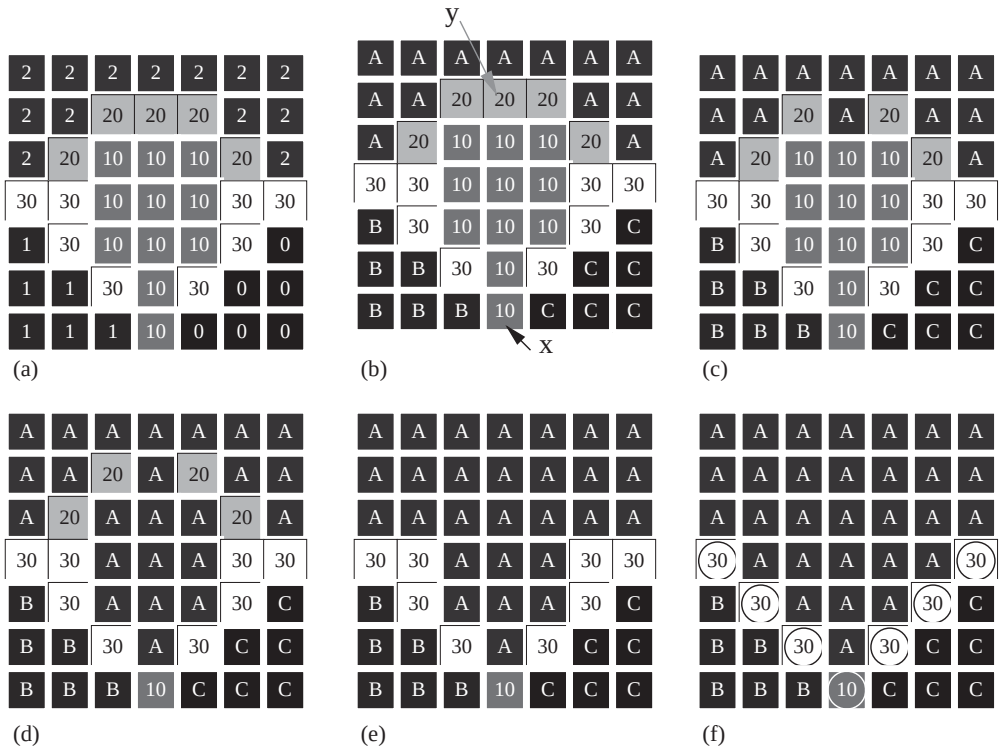


Fig. 2. (a) Original image, (b–f) several steps of the flooding algorithm. One can see that this algorithm is not "monotone": some points at altitude 10 are processed after one of the points at altitude 20. One can also note that the contour at altitude 20 in the original image (a) is not present in the result (f).

the algorithm. Let us emphasize that similar configurations can be found for other adjacency relations, and in particular for the 6- and the 8-adjacency relation. Configurations similar to the examples presented in this paper are found in real-world images.

The following section introduces the formal framework that leads to a better understanding of the previous observations.

4. Minima extensions, mosaics, and pass values

A result of the previous algorithm is to associate an influence zone to each minimum of the image. We formalize this through the definition of a minima extension.

Definition 1. Let $F \in \mathscr{F}(E)$. A *minima extension of F* is a subset X of E such that:

- each connected component of X contains one and only one minimum of F, and
- each minimum of F is included in a connected component of X.

The complementary of a minima extension of F is called a *divide set* (of F).

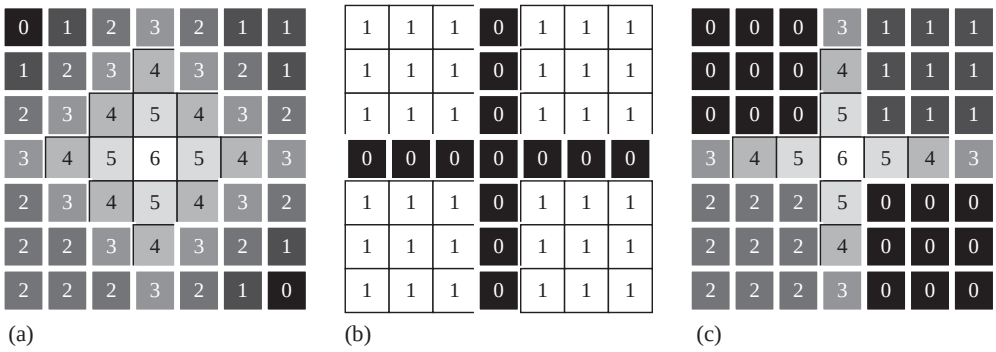


Fig. 3. (a) An image, (b) a minima extension of (a), and (c) the associated mosaic.

It is easy to prove the following result: let $F \in \mathscr{F}(E)$, and let X be the set composed of all the points labeled by the flooding algorithm applied on F; the set X is indeed a minima extension of F. We call any such set X produced by the flooding algorithm a *flooding extension* (of F). Note that, in general, there may exists several flooding extensions of a given map F.

Intuitively, for application to image analysis, the divide set represents the location of points which best separate the dark objects (regional minima), in terms of grey level difference (contrast). In order to evaluate the effectiveness of this separation, we have to consider the values of points along the divide set. This motivates the following definition.

Definition 2. Let $F \in \mathscr{F}(E)$ and let X be a minima extension of F. The *mosaic* of F associated with X is the map $F_X \in \mathscr{F}(E)$ such that

- for any $x \notin X$, $F_X(x) = F(x)$; and
- for any $x \in X$, $F_X(x) = \min\{F(y) | y \in C_x\}$, where C_x denotes the connected component of X that contains x.

The term 'mosaic' for this kind of construction, was coined by Beucher [3].

Fig. 3 shows a simple example of a minima extension and its associated mosaic. The flooding extension of Fig. 3a is the minima extension 3b, and the associated mosaic is Fig. 3c.

Fig. 4 is another illustration of the definitions of minima extension and mosaic, using the flooding algorithm on the image of Fig. 2a.

Let F be a map and let F_X be the mosaic of F associated with a minima extension X of F. It is natural to try to associate any minimum of F_X to a connected component of X and conversely, and to compare the altitude of each minimum of F_X to the altitude of the corresponding minimum of F. We will see with forthcoming properties and examples, that both problems are in fact closely linked.

The following definition extends to maps the minima extension previously defined for sets.

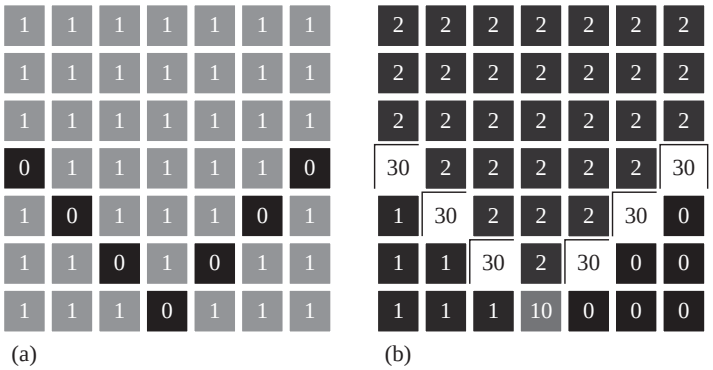


Fig. 4. (a) The flooding extension of Fig. 2a, and (b) the associated mosaic.

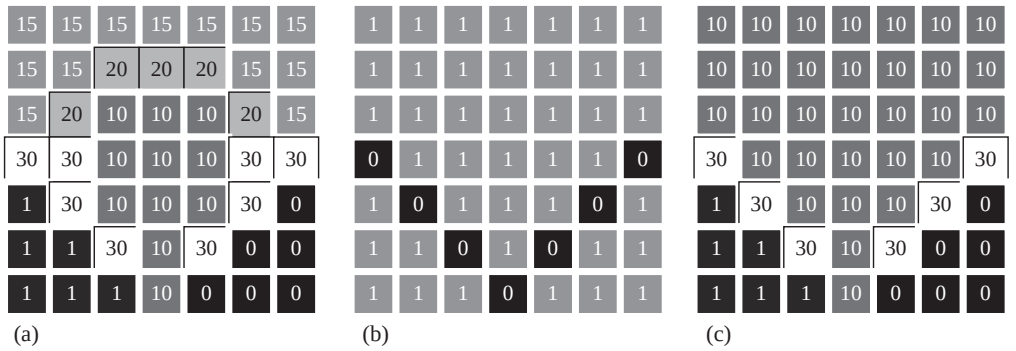


Fig. 5. (a) An image F, and (b) the flooding extension of F, and (c) the associated mosaic.

Definition 3. Let F and G in $\mathscr{F}(E)$ such that $G \leqslant F$. We say that G is a *minima extension* (*of* F) if:

(i) the set composed by the union of all the minima of G is a minima extension of F.
(ii) for any $X \in \mathscr{M}(F)$ and $Y \in \mathscr{M}(G)$ such that $X \subseteq Y$, we have $F(X) = G(Y)$.

The image of Fig. 3c (resp. 4b) is an example of a minima extension of the image of Fig. 3a (resp. 2a).

On the other hand, Fig. 5a shows an image F and Fig. 5c shows the mosaic F_X associated with the flooding extension X (Fig. 5b) of the image F. One can notice that the connected component of X which corresponds to the minimum of altitude 15 for F has an altitude of 10 for F_X, and is not a minimum of F_X. Thus, this mosaic F_X is not a minima extension of F. In other words, Fig. 5 shows that mosaics produced by the flooding algorithm are not always minima extensions of the original map.

We can now turn back to a more precise analysis of Remark 2. To this aim, we present the pass value and the separation. Intuitively, the pass value between two points corresponds to the lowest altitude to which one has to climb to go from one of these points to the other.

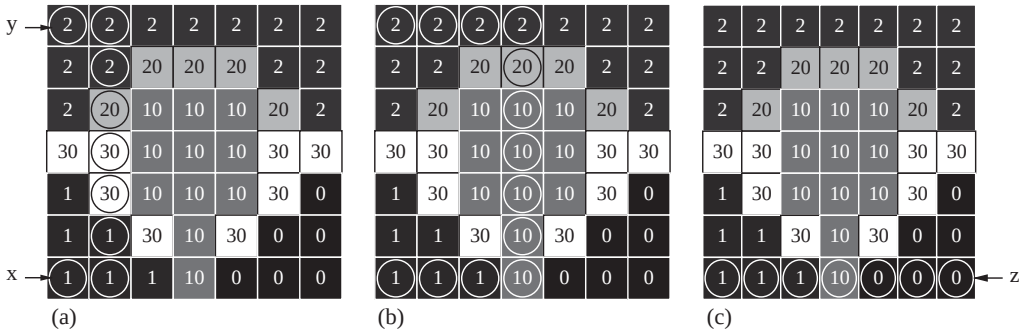


Fig. 6. Illustration of paths and pass values on the image F of Fig. 2a. (a) A path π_1 from the point x to the point y such that $F(\pi_1) = 30$. (b) A path π_2 from the point x to the point y such that $F(\pi_2) = 20$. It is not possible to find a path from x to y with a lower maximal altitude, hence $F(x, y) = 20$. (c) A path π_3 from the point x to the point z such that $F(\pi_3) = 10$, and we can easily check that $F(x, z) = 10$.

Definition 4. Let $F \in \mathscr{F}(E)$. Let $\pi = (x_0, \ldots, x_n)$ be a path in the graph (E, Γ), we set $F(\pi) = \max\{F(x_i) | i = 0, \ldots, n\}$.

Let x, y be two points of E, the *pass value for F between x and y* is defined as $F(x, y) = \min\{F(\pi) | \pi \in \Pi(x, y)\}$, where $\Pi(x, y)$ is the set of all paths from x to y.

Let X, Y be two subsets of E, the pass value for F between X and Y is defined by $F(X, Y) = \min\{F(x, y) | \text{for any } x \in X \text{ and any } y \in Y\}$.

A notion equivalent to the pass value up to an inversion of F (that is, replacing F by $-F$), has been introduced by Rosenfeld [28–30] under the name of *degree of connectivity* for studying connectivity in the framework of fuzzy sets. Fig. 6 illustrates the pass value on the image F of Fig. 2a.

Informally, a transformation "preserves the separation" if, when two points are separated by a crest in the original map, they are still separated by a crest of the same "height" in the transform.

Definition 5 (*Bertrand [1]*). Let $F \in \mathscr{F}(E)$, let $x, y \in E$. We say that x and y are *separated* (*for F*) if $F(x, y) > \max\{F(x), F(y)\}$.

We say that x and y are *linked* (*for F*) if $F(x, y) = \max\{F(x), F(y)\}$.

We say that x and y are *k-separated* (*for F*) if they are separated for F and if $k = F(x, y)$.

Let $G \in \mathscr{F}(E)$, with $G \leqslant F$. We say that G is a *separation of F* if, for all x and y in E, whenever x and y are k-separated for F, x and y are k-separated for G.

We say that G is a *strong separation of F* is G is both a separation of F and a minima extension of F.

Remark 3. We can now restate Remark 2 using the notions we have introduced in this section. Fig. 5 shows that a mosaic produced by the flooding algorithm is not always a minima extension of the original map. Fig. 4 shows that a mosaic produced by the flooding algorithm, even in the case where it is a minima extension, is not necessarily a separation of the original map.

5. Topological watershed

A different approach to the watershed was presented by Couprie and Bertrand [7]. The idea is to transform the image F into an image G while preserving some topological properties of F, namely the number of connected components of the lower cross-sections of F. A minima extension of F can then be obtained easily from G, by extracting the minima of G.

5.1. Definitions

We begin by defining a "simple" point (in a graph), in a sense which is adapted to the watershed, then we extend this notion to weighted graphs through the use of lower sections [7].

Definition 6. Let $X \subseteq E$. The point $x \in X$ is W-*simple* (for X) if x is adjacent to one and only one connected component of $\overline{X}$.

In other words, x is W-simple (for X) if the number of connected components of $\overline{X} \cup \{x\}$ equals the number of connected components of $\overline{X}$.

We can now define the notions of W-destructible point, W-thinning, and topological watershed:

Definition 7. Let $F \in \mathscr{F}(E)$, $x \in E$, and $k = F(x)$.

The point x is *W-destructible* (*for* F) if x is W-simple for F_k.

We say that $G \in \mathscr{F}(E)$ is a *W-thinning of* F if $G = F$ or if G may be derived from F by iteratively lowering W-destructible points by one.

We say that $G \in \mathscr{F}(E)$ is a *topological watershed of* F if G is a W-thinning of F and if there is no W-destructible point for G.

The differences between topological watershed and the notion of *homotopic greyscale skeleton* are discussed in Appendix A.

As a consequence of the definition, a topological watershed G of a map F is a map which has the same number of minima as F. Furthermore, the number of connected components of any lower cross-section is preserved during this transformation.

By the very definition of a W-destructible point, it may easily be proved that, if G is a W-thinning of F, then the union of all minima of G is a minima extension of F (this result is also a consequence of Theorem 10). This allows us to propose the following definition.

Definition 8. Let $F \in \mathscr{F}(E)$ and let G be a W-thinning of F. The *mosaic of* F *associated with* G is the mosaic of F associated with the union of all minima of G.

Notice that in general, there exist different topological watersheds for a given map F. Fig. 7a presents one of the possible topological watersheds of Fig. 2a, and Fig. 7b shows the associated mosaic. One can note that both Fig. 7a and b are separations of Fig. 2a.

An extensive algorithmic study of the topological watershed is made in [10], which proposes in particular a quasi-linear algorithm.

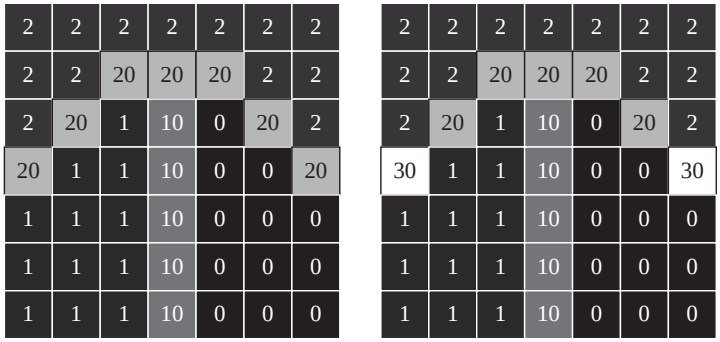


Fig. 7. Example of topological watershed. (a) a topological watershed of Fig. 2a (b) the associated mosaic.

5.2. Topological watershed and separation

Recently, Bertrand [1] showed that a mathematical key underlying the topological watershed is the *separation*. The following theorem asserts that it is sufficient to consider the minima of F for testing if G is a separation of F.

Theorem 9 (*Bertrand [1]*). *Let F and G be two elements of $\mathscr{F}(E)$ such that $G \leqslant F$. The map G is a separation of F if and only if, for all distinct minima X, Y of F, $F(X, Y) = G(X, Y)$.*

The following theorem states the equivalence between the notions of W-thinning and strong separation. The "if" part implies in particular that a topological watershed of an image F preserves the pass values between the minima of F. Furthermore, the "only if" part of the theorem mainly states that if one needs a transformation which is guaranteed to preserve the pass values between the minima of the original map, then this transformation is necessarily a W-thinning.

Theorem 10 (*Bertrand [1]*). *Let F and G be two elements of $\mathscr{F}(E)$. The map G is a W-thinning of F if and only if G is a strong separation of F.*

Let $F \in \mathscr{F}(E)$ and $p \in E$. We denote by $\Gamma^-(p, F)$ the set of (strictly) lower neighbors of p, that is, $\Gamma^-(p, F) = \{q \in \Gamma(p) | F(q) < F(p)\}$. In the sequel, we will need the following characterization of W-destructible points:

Property 11 (*Couprie et al. [10]*). *Let $F \in \mathscr{F}(E)$ and $p \in E$. The point p is W-destructible for F if and only if $\Gamma^-(p, F) \neq \emptyset$ and, for all x and y in $\Gamma^-(p, F)$ with $x \neq y$, we have $F(x, y) < F(p)$.*

5.3. Mosaic and separation

We can now prove that the mosaic associated with any W-thinning of a map F is also a W-thinning of F (and thus, it is a separation of F). Furthermore, we prove that an arbitrary

mosaic F_X of a map F is a separation of F *if and only if* F_X is a W-thinning of F. These strong results can be obtained thanks to the three following properties.

Property 12. *Let $F \in \mathscr{F}(E)$, let X be a minima extension of F, and let F_X be the mosaic of F associated with X. Then, any minimum M of F_X is a connected component of X; furthermore $F_X(M) = F(m)$ where m denotes the unique minimum of F such that $m \subseteq M$.*

A proof of Proposition 12 can be found in Appendix B. The following property follows straightforwardly.

Property 13. *Let $F \in \mathscr{F}(E)$, let X be a minima extension of F, and let F_X be the mosaic of F associated with X. If any connected component of X is a minimum of F_X, then F_X is a minima extension of F.*

Property 14. *Let $F \in \mathscr{F}(E)$, let X be a minima extension of F, and let F_X be the mosaic of F associated with X. If F_X is a separation of F, then F_X is a minima extension of F.*

Proof. As X is a minima extension of F, by Proposition 12, we know that any minimum of F_X is a connected component of X. We have to prove that any connected component of X is a minimum of F_X.

Let M be a connected component of X, and let m be the minimum of F that is included in M. Suppose that M is not a minimum of F_X. Let $k = F_X(M) + 1$, and let C be the connected component of $\overline{(F_X)_k}$ that contains M. Let N be a minimum of F_X that is included in C. By Proposition 12, $N \subseteq X$. Let n the minimum of F that is included in N. We see easily that n and m are such that $F_X(n, m) = F_X(m)$. But F_X is a separation of F, and by Theorem 9, $F_X(n, m) = F(n, m)$. As n and m are minima of F, we have $F(n, m) > \max\{F(n), F(m)\}$, a contradiction. Thus, any connected component of X is a minimum of F_X.

By Proposition 13, F_X is thus a minima extension of F. □

Property 15. *Let $F \in \mathscr{F}(E)$, let G be a W-thinning of F, and let H be the mosaic of F associated with G. Then H is a separation of F.*

Proof. Let M and M' be two distinct minima of F and let $k = F(M, M')$. There exists a path π from a point of M to a point of M' such that $F(\pi) = k$. Since $G \leqslant F$, we have $G(\pi) \leqslant k$, but, by Theorem 9, we must have $G(\pi) \geqslant k$ (otherwise we would have $G(M, M') < k$). Hence $G(\pi) = k$. Since $G \leqslant H$, we have $H(\pi) \geqslant k$. But since $H \leqslant F$, $H(\pi) \leqslant k$. It follows that $H(\pi) = k$ and we may affirm that $H(M, M') \leqslant k$. Now suppose that $H(M, M') < k$. It means that there exists a path π' from a point of M to a point of M' such that $H(\pi') < k$. Since $G \leqslant H$, we would have $G(\pi') < k$ which contradicts $G(M, M') = k$. So $H(M, M') = k$, and, from Theorem 9, we deduce that H is a separation of F. □

Property 16. *Let $F \in \mathscr{F}(E)$, let G be a W-thinning of F, and H be the mosaic of F associated with G. Then H is necessarily a W-thinning of F.*

Proof. By Proposition 15, H is a separation of F. By Proposition 14, H is a minima extension of F. In consequence H is a strong separation of F which, by Theorem 10, implies that H is a W-thinning of F. □

The following theorem is a straightforward consequence of Theorem 10 and Proposition 14.

Theorem 17. *Let $F \in \mathscr{F}(E)$, let X be a minima extension of F, and let F_X be the mosaic of F associated with X. Then F_X is a separation of F if and only if F_X is a W-thinning of F.*

6. Emergence watershed

In this section, we first design a monotone algorithm based on both the flooding paradigm and W-destructible points. We show that such an algorithm does not always produce a topological watershed, more precisely, there may exist points of the divide set that are still W-destructible. This will lead us, in the second part of the section, to reverse the flooding paradigm and to propose the notion of emergence.

To produce a W-thinning, we sequentially lower the altitude of W-destructible points by one. A particular case of this process is obtained if, when a point has been lowered, we immediately check whether the same point is W-destructible or not, and continue until the point is no more W-destructible.

Let $F \in \mathscr{F}(E)$, and let x be a W-destructible point for F.

- We call *W-lowering of x* the action of lowering the altitude of x by one.
- We call *W*-lowering of x* the action of successively W-lowering the altitude of x until it is no more W-destructible for the result.

Let us denote by $\mathscr{F}_0(E)$ the set of all maps $F \in \mathscr{F}(E)$ such that $\min\{F(x)|x \in E\} = 0$. In the sequel, for the sake of simplicity and without loss of generality, we will often restrict ourselves to maps belonging to $\mathscr{F}_0(E)$.

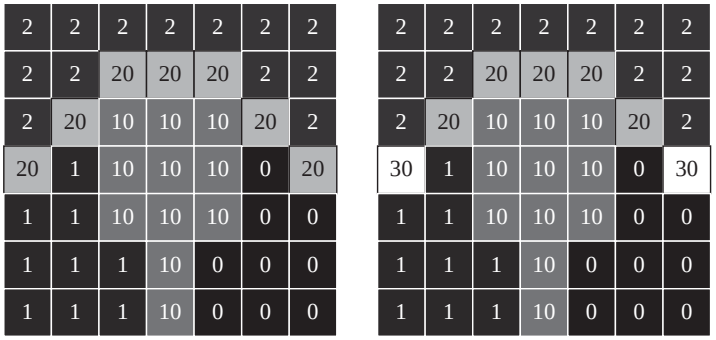
6.1. A monotone W-flooding

Let us design a "monotone" flooding-like algorithm based on the lowering of W-destructible points by increasing order of altitude. By Theorem 10, such an algorithm will always produce separation.

Let $F \in \mathscr{F}(E)$. We say that

- G is a *W*-thinning of F for level k* if $G = F$ or if we can obtain G from F by iteratively W*-lowering some W-destructible points p such that $F(p) = k$.
- G is an *ultimate W*-thinning of F for level k* if G is a W*-thinning of F for level k and if G contains no W-destructible point p such that $F(p) = k$.

The following algorithm builds a W-thinning that is called a *monotone* W-*flooding of F*.



(a) (b)

Fig. 8. Example of monotone W-flooding. (a) a monotone W-flooding of Fig. 2a (b) the associated mosaic.

Definition 18. Let $F \in \mathscr{F}_0(E)$, and let $m = \max\{F(x)|x \in E\}$. Let $G^{(0)} = F$, and for any $k = 0 \ldots m-1$, let $G^{(k+1)}$ be an ultimate W*-thinning of $G^{(k)}$ for level $k+1$. The sequence $(G^{(0)}, \ldots, G^{(m)})$ is called a *monotone* W-*flooding sequence for* F, and $G^{(m)}$ is called a *monotone* W-*flooding* of F.

Let $F \in \mathscr{F}_0(E)$. It is obvious that any monotone W-flooding of F is a W-thinning of F.

Nevertheless, a monotone W-flooding process does not necessarily produces a topological watershed. A monotone W-flooding of the image 2a is depicted in Fig. 8a. It may be seen that, while the monotone W-flooding 8a is a W-thinning of 2a, several points in 8a are W-destructible.

Let us note that a monotone algorithm based on flooding has been proposed by Vincent and Soille [31,34–36]. The application of the dividing variant of this algorithm on an image $F \in \mathscr{F}(E)$ produces a minima extension X of F, but the mosaic F_X of F associated with X is not always a W-thinning of F (see [21]). An illustration is provided in Fig. 11.

6.2. Emergence watershed

We have seen that the flooding paradigm does not lead to a satisfying result, even when we proceed by lowering exclusively W-destructible points. Surprisingly, we will see that reversing the level scanning order leads to an algorithm which possesses good properties. We introduce in this section the emergence watershed, which is based on a process where points are considered in decreasing altitude order, and prove one of the main results of the paper: for any map F, any emergence watershed of F is a topological watershed of F (and thus a separation of F), and more remarkably, any topological watershed of F is an emergence watershed of F. Let us note that a process similar to the emergence has been proposed in [8] in the framework of orders, but no property of this emergence process had been studied in this latter work.

Let $F \in \mathscr{F}(E)$. We say that

- G is a *W-thinning* (*of* F) for level k if $G = F$ or if we can obtain G from F by iteratively W-lowering some W-destructible points p such that $F(p) = k$.

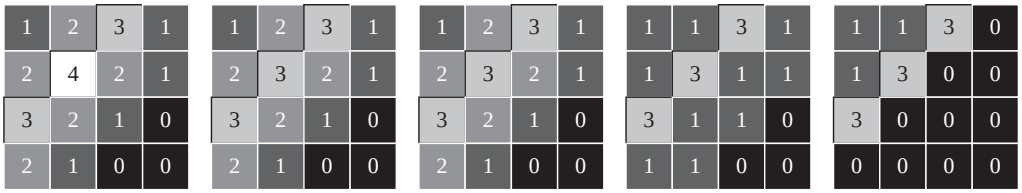


Fig. 9. An image F, and an emergence sequence for F: $(G^{(4)}, G^{(3)}, G^{(2)}, G^{(1)}, G^{(0)})$.

- G is a *ultimate W-thinning (of F) for level k* if G is W-thinning for level k of F and if G contains no W-destructible point p such that $F(p) = k$.

Definition 19. Let $F \in \mathscr{F}_0(E)$, and let $m = \max\{F(x) | x \in E\}$.

Let $G^{(m)} = F$ and, and for any $k = 1 \ldots m$, let $G^{(k-1)}$ be an ultimate W-thinning of $G^{(k)}$ for level k.

The sequence $(G^{(m)} \ldots G^{(0)})$ is called a *emergence sequence for F*, and $G^{(0)}$ is called an *emergence watershed of F*.

Fig. 9 illustrates the emergence process.

Before stating and proving our results, we introduce some notations, definitions and intermediate properties.

Let $F \in \mathscr{F}(E)$. If $x \in E$, we denote by $F \backslash x$ the element of $\mathscr{F}(E)$ such that $(F \backslash x)(y) = F(y)$ for any $y \neq x$ and $(F \backslash x)(x) = F(x) - 1$.

The following two lemmas arise immediately from Property 11 and from the definition of a W-destructible point.

We recall that $\Gamma^-(p, F) = \{q \in \Gamma(p) | F(q) < F(p)\}$.

Lemma 20. *A point p is not W-destructible for F if and only if either $\Gamma^-(p, F) = \emptyset$ or there exist x and y in $\Gamma^-(p, F)$ with $x \neq y$ such that $F(x, y) = F(p)$.*

Proof. It follows from Property 11 and from the fact that the path $\pi = (x, p, y)$ is such that $F(\pi) = F(p)$. □

Lemma 21. *Let $F \in \mathscr{F}(E)$, let p be a point such that $F(p) = k$, and let q be a point such that $F(q) < k$. If p is W-destructible for F, then p is W-destructible for $F \backslash q$.*

Proof. Since the lower cross-section $\overline{F_k}$ is equal to the lower cross-section $\overline{(F \backslash q)_k}$, the property follows from the very definition of a W-destructible point. □

The following notion of stable point is essential for the understanding of the emergence properties.

Definition 22. Let $F \in \mathscr{F}(E)$ and $p \in E$. We say that p is a *stable point (for F)* if p is not W-destructible for any W-thinning of F.

We say that G is a *topological watershed* (*of* F) *above level* k if G is a W-thinning of F and if any p such that $G(p) > k$ is a stable point for G.

Notice that, since any map $F \in \mathscr{F}(E)$ is by definition a W-thinning of F itself, any point which is a stable point for F is not W-destructible for F.

6.3. Emergence and topological watershed

We shall prove that all the points "emerging" from the emergence process (that is, points above the current altitude) are stable points. The proof relies on the following property.

Property 23. *Let $F \in \mathscr{F}(E)$. Let $p \in E$ be a point which is not W-destructible for F and let $q \in E$ be a point W-destructible for F. If p is W-destructible for $F\backslash q$, then $F(q) = F(p)$.*

Proof. Suppose that there exist x and $y \in \Gamma^-(p, F)$ such that $F(x, y) = F(p) = k$. Since q is W-destructible for F, we know that $q \neq p$. Furthermore, since $F\backslash q$ is a W-thinning of F, we know from Theorem 10 that x and y are k-separated for $F\backslash q$, thus p is not W-destructible for $F\backslash q$, a contradiction.

Thus by Lemma 20, we deduce that $\Gamma^-(p, F) = \emptyset$. Since p has no lower neighbor for F and has a lower neighbor for $F\backslash q$, this lower neighbor is q and $F(q) = F(p)$. □

We can now prove that in an emergence sequence, all the points above the current altitude are stable points.

Property 24. *Let $F \in \mathscr{F}_0(E)$. Let $(G^{(m)} \dots G^{(0)})$ be an emergence sequence for F. Let $k \in [0 \dots m]$. Then $G^{(k)}$ is a topological watershed of F above level k.*

Proof. Obviously, $G^{(k)}$ is a W-thinning of F. Thus, in order to prove the property, it is sufficient to show that (1) any point p such that $G^{(k)}(p) > k$ is a stable point for $G^{(k)}$.

The property (1) is true for $k = m$ since there is no point $p \in E$ such that $G^{(m)}(p) > m$. Suppose that the property is true for all $i > k$. We set $h = G^{(k)}(p)$, we have $h > k$.

- Suppose that $h > k + 1$. By the recurrence hypothesis, p is a stable point for $G^{(k+1)}$, thus p is obviously a stable point for $G^{(k)}$ which is a W-thinning of $G^{(k+1)}$.
- Suppose now that $h = k + 1$. Suppose that p is not stable for $G^{(k)}$, i.e., p is W-destructible for a W-thinning G of $G^{(k)}$. By construction of the emergence, the point p is not W-destructible for $G^{(k)}$.

Let us write $G = G^{(k)}\backslash x_0 \backslash \dots \backslash x_n$ where for all $i \in [0 \dots n]$, x_i is W-destructible for $G^{(k)}\backslash x_0 \backslash \dots \backslash x_{i-1}$. Without loss of generality, we can assume that for any $G^{(k)}\backslash x_0 \backslash \dots \backslash x_i$, $i < n$, no point of level h has been lowered (otherwise, we choose the first one among such points instead of p). We can also assume that p is not W-destructible for $G^{(k)}\backslash x_0 \backslash \dots \backslash x_{n-1}$ (otherwise we choose n such that it is the case).

By recurrence hypothesis and by construction, no point of level greater or equal to h has been lowered by this sequence. Thus all the points $x_0, \dots, x_n$ are such that $G^{(k)}(x_i) < h$. On

the other hand, by Property 23, we may affirm that $(G^{(k)}\backslash x_0\backslash\ldots\backslash x_{n-1})(x_n)=G^{(k)}(p)=h$, hence $G^{(k)}(x_n)=h$, a contradiction. □

We shall now prove that any topological watershed of a map can be obtained by an emergence sequence.

Property 25. *Let $F\in\mathscr{F}_0(E)$ and G a topological watershed of F. Then G is an emergence watershed of F.*

Proof. Let us write $G=F\backslash x_1\backslash x_2\backslash\ldots\backslash x_n$, meaning that G is obtained from F by iteratively W-lowering the points $x_1,\ldots,x_n$. For the sake of brevity, we will denote this sequence of W-lowerings by $(x_1,x_2,\ldots x_n)$.

Let $i\in[1\ldots(n-1)]$. Suppose that, at step i, we have $(F\backslash x_1\backslash\ldots\backslash x_{i-1})(x_i)<(F\backslash x_1\backslash\ldots\backslash x_i)(x_{i+1})$.

Let us show that in this case, we can "exchange" the lowerings of points x_i and x_{i+1}, while still proceeding by W-lowerings.

Let us write $F'=F\backslash x_1\backslash\ldots\backslash x_{i-1}$, the hypothesis becomes $F'(x_i)<(F'\backslash x_i)(x_{i+1})$. Notice that we have necessarily $x_{i+1}\neq x_i$, and thus $F'(x_i)<F'(x_{i+1})$.

We need to prove that (a): x_{i+1} is W-destructible for F'; and that (b): x_i is W-destructible for $F'\backslash x_{i+1}$.

(a) Let us write $k=F'(x_{i+1})$. Since $F'(x_i)<k$, we have $\overline{(F'\backslash x_i)_k}=\overline{F'_k}$. Since x_{i+1} is W-destructible for $(F'\backslash x_i)$, x_{i+1} is W-destructible for F'.

(b) Let us write $h=F'(x_i)$. Since $h<F'(x_{i+1})$, we have $\overline{F'_h}=\overline{(F'\backslash x_{i+1})_h}$. Since x_i is W-destructible for F', x_i is W-destructible for $(F'\backslash x_{i+1})$.

Obviously, $F'\backslash x_i\backslash x_{i+1}=F'\backslash x_{i+1}\backslash x_i$. Thus, the new sequence $(x_1,\ldots,x_{i-1},x_{i+1},x_i,x_{i+2},\ldots,x_n)$ of lowerings is indeed composed of W-lowerings and also produces the map G.

By repeating such exchanges until stability, we see that we can obtain a sequence $S=(x'_1,\ldots,x'_n)$ of W-lowerings such that $G=F\backslash x'_1\backslash\ldots\backslash x'_n$ and such that for all $1\leqslant i<n$, $(F\backslash x'_1\backslash\ldots\backslash x'_{i-1})(x'_i)\geqslant(F\backslash x'_1\backslash\ldots\backslash x'_i)(x'_{i+1})$.

We write $F^{(m)}=F$. For any $k\in[1\ldots m]$, we define $F^{(k-1)}=F\backslash x'_1\backslash\ldots\backslash x'_i$, such that x'_i is the last point in the sequence S for which $(F\backslash x'_1\backslash\ldots\backslash x'_{i-1})(x'_i)\geqslant k$. We have $F^{(0)}=G$.

The sequence $(F^{(m)}\ldots F^{(0)})$ is an emergence sequence for F. Indeed, suppose that there exists a point p W-destructible for $F^{(k-1)}$ such that $F(p)=k$. By construction and Lemma 21, this point would be W-destructible for $F^{(0)}$. This is not possible since $F^{(0)}$ is a topological watershed. □

We can now state the main result of this section.

Theorem 26. *Let $F\in\mathscr{F}_0(E)$. A map $G\in\mathscr{F}_0(E)$ is a topological watershed of F if and only if it is an emergence watershed of F.*

Proof. Suppose that $(G^{(m)},\ldots,G^{(0)})$ with $G^{(0)}=G$ is an emergence sequence for F. Obviously, the map $G^{(0)}$ is a W-thinning of F. Property 24 states that for all $k\in[0\ldots m]$,

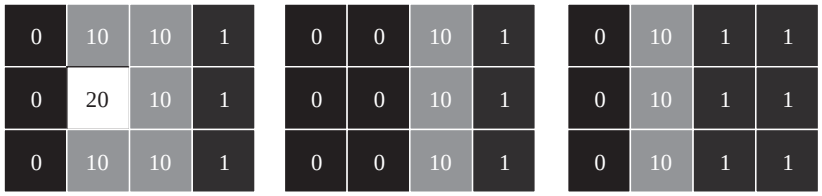


Fig. 10. An image (a), and two associated topological watersheds (b) and (c). Note that, contrary to the topological watershed (b), the topological watershed (c) cannot be obtained through a reverse W-flooding process.

$G^{(k)}$ is a topological watershed above level k of F. In particular $G^{(0)}$ has no W-destructible point such that $G^{(0)}(p) > 0$. Any point p such that $G^{(0)}(p) = 0$ is in a minimum of $G^{(0)}$, hence p is not W-destructible. Thus $G^{(0)}$ is a topological watershed of F. The converse is proved by Property 25. □

6.4. Emergence and reverse W-flooding

We may wonder if we can propose a variant of the emergence process where, instead of lowering the value of points by one (W-lowerings), we lower the value of points until those points are no more W-destructible (W*-lowerings). We are going to see that, although such a process always produces a topological watershed, there exist topological watersheds that cannot be obtained in this way.

The following algorithm, called *reverse W-flooding* is a direct inversion of the monotone W-flooding.

Definition 27. Let $F \in \mathscr{F}_0(E)$, let $m = \max\{F(x) | x \in E\}$.

Let $G^{(m)} = F$ and, for any $k = 1 \ldots m$, let $G^{(k-1)}$ be an ultimate $W^\star$-thinning of $G^{(k)}$ for level k.

The sequence $(G^{(m)}, \ldots, G^{(0)})$ is called a *reverse W-flooding sequence for F*, and $G^{(0)}$ is called a *reverse W-flooding* of F.

A major feature of the reverse W-flooding is that, in opposition to the monotone W-flooding, the result is guaranteed to be a topological watershed. The proof of the following property is very similar to the one of Property 24, and will thus be omitted.

Property 28. *Let $F \in \mathscr{F}_0(E)$, and let G be a reverse W-flooding of F. Then G is a topological watershed of F.*

Fig. 10 shows an image and two associated topological watersheds. It can be easily seen that the topological watershed 10c cannot be obtained through a reverse W-flooding process. The point at altitude 20 is necessarily lowered to 0 by any reverse W-flooding.

7. Conclusion

The watershed transform is more and more used as a low-level operator in complex segmentation chains. Among those segmentation procedures, we can cite *hierarchical seg-*

mentation [6] and *geodesic saliency of watershed contours* [23,25]. Such approaches need to compare several divides, or are based on neighborhood relationship between extended minima. It is thus important to be able to characterize some properties of the divides produced by watershed algorithms. This paper is a step in this direction. We introduced several notions that helped us to understand the watershed: minima extension, mosaic, and we also consider the pass values and separation.

The topic of this paper is to examine the links between the flooding paradigm and the topological watershed. We prove in particular that a mosaic is a separation *if and only if* it is a W-thinning. Inspired by the analysis of the flooding algorithm, we present the monotone W-flooding. A monotone W-flooding does not necessarily produce a topological watershed. This leads us to propose the emergence paradigm. A major result of this paper is that any emergence of a given image is a topological watershed of this image, and more remarkably, that any topological watershed of a given image can be obtained as an emergence of the image.

Future work will build up on those results to revisit the saliency of contours. We also aim at exploring definitions and properties of "watersheds without divides" (Fig. 11).

Appendix A. Topological watershed versus homotopic greyscale skeleton

There exists in the literature an approach called *homotopic greyscale skeleton* [13,2,9,26,33] that can be used for thinning a greyscale image. It can be easily proved that the pass values between the minima of a homotopic greyscale skeleton G of an image $F \in \mathscr{F}(E)$ are the same than the pass values between the minima of F.

Fig. 12 presents a 2D image (Fig. 12a), and both a topological watershed (Fig. 12b) and a homotopic greyscale skeleton (Fig. 12c) of this image.

Let us emphasize the essential difference between the topological watershed and the homotopic greyscale skeleton. With the topological watershed, only the number of connected components of the lower cross-sections of the map are preserved, while the homotopic greyscale skeleton preserves both these components and the components of the upper cross-sections. As a consequence, a homotopic greyscale skeleton may be computed by using a purely local criterion for testing whether a point may be lowered or not, while computing a topological watershed requires the reiteration of global algorithms for computing connected components, or the use of a global data structure called *component tree* [7,22]. Notice that a topological watershed only produces closed contours around the regions of interest (Fig. 12b). One can see in Fig. 12c that this is not the case for a homotopic greyscale skeleton: there is a "skeleton branch" at level 11 which does not separate different minima.

Appendix B. Proof of Property 12

Let $F \in \mathscr{F}(E)$, let x, y be two points of E, recall that "x and y are linked for F" means that $F(x, y) = \max\{F(x), F(y)\}$. Let X, Y be two subsets of E which are flat for F, we say that *x and Y are linked for F* if for any $y \in Y$, x and y are linked for F; and we say that *X and Y are linked for F* if for any $x \in X$, for any $y \in Y$, x and y are linked for F. In the same way, we say that *x and Y are separated for F* if for any $y \in Y$, x and y are separated for F.

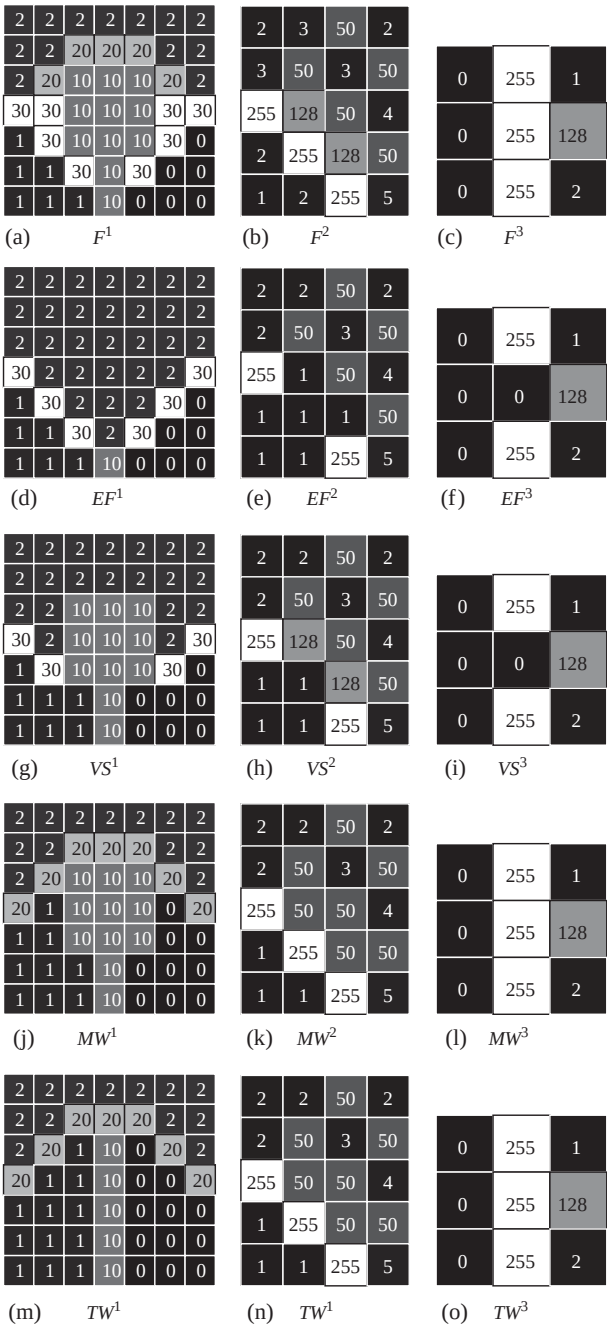


Fig. 11. Examples of the application of the flooding algorithm, the Vincent–Soille algorithm, the monotone W-flooding and the topological watershed on several images F^i ($i = 1, 2$ and 3). The mosaics produced by the flooding algorithm and by Vincent–Soille's algorithm are denoted by EF^i and VS^i, respectively, the monotone W-floodings are denoted by MW^i and the topological watersheds are denoted by TW^i. One can observe that the pass value between the minima (at altitude) 1 and the minima 2 is 20 for F^1, 10 for EF^1 and VS^1, and 20 for MW^1 and TW^1; the pass value between the minima (at altitude) 1 and any other minima is 255 for F^2, 50 for EF^2, 128 for VS^2 and 255 for MW^2 and TW^2; the pass value between the minima (at altitude) 0 and any other minima is 255 for F^3, 128 for EF^3 and VS^3, and 255 for MW^3 and TW^3.

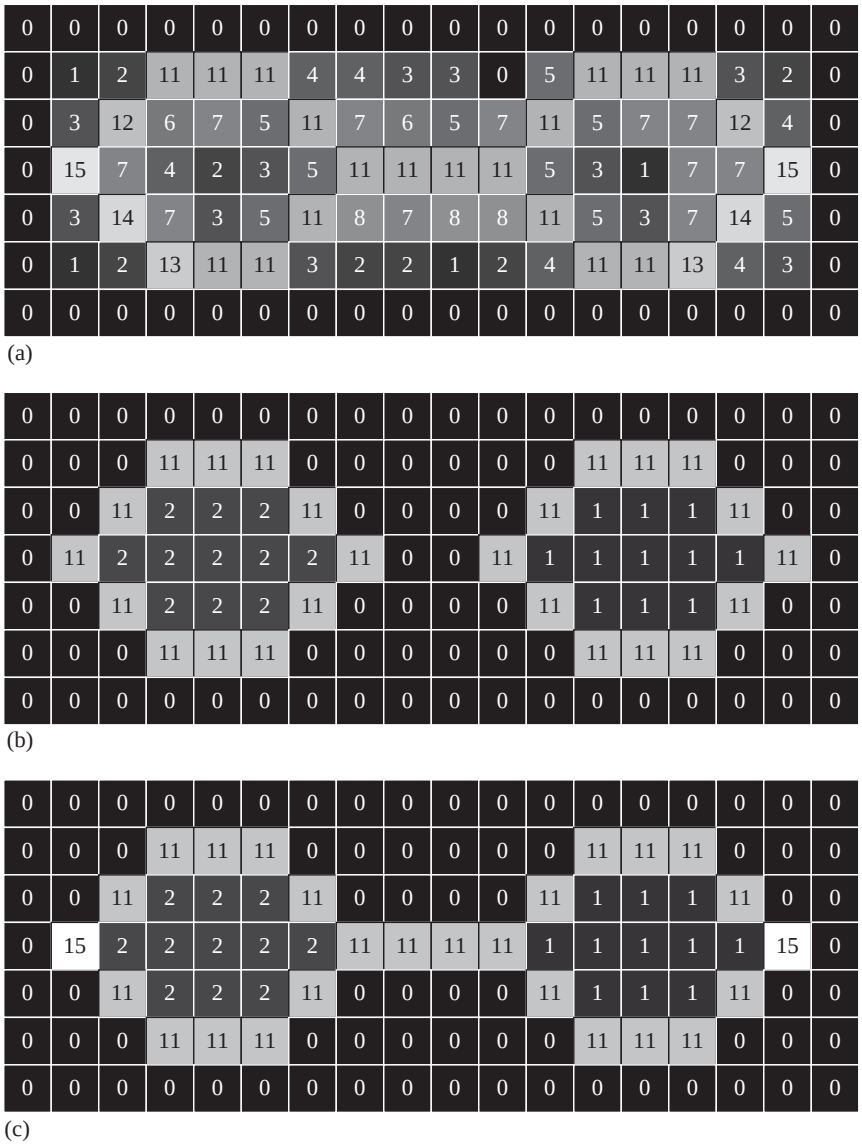


Fig. 12. An image (a), a topological watershed (b) of the image (a) and a homotopic greyscale skeleton (c) of the image (a).

Let us state two basic properties which are fundamental to understand subsequent proofs, and can easily be verified.

Property 29. *Let $F \in \mathscr{F}(E)$, let m be a minimum of F, and let $x \in E$. If x and m are linked for F, then we have*:

$F(x) = F(m)$ *if and only if* $x \in m$, *and*
$F(x) > F(m)$ *if and only if* $x \notin m$

Property 30. *Let $F \in \mathscr{F}(E)$. For any $x \in E$, there exists a minimum m of F such that x and m are linked. Furthermore, $F(x) \geqslant F(m)$.*

Property 31. *Let $F \in \mathscr{F}(E)$, let X be a minima extension of F, let F_X be the mosaic of F associated with X. Let M be a connected component of X, and let m be the unique minimum of F such that $m \subseteq M$.*

If M is a minimum of F_X, then we have $F_X(M) = F(m)$.

Proof. Since $F_X \leqslant F$, we have $F_X(M) \leqslant F(m)$. Suppose that $F_X(M) < F(m)$. By definition of F_X, there exists a point $x \in M$ such that $F(x) = F_X(M) < F(m)$, furthermore x and m must be separated (Proposition 29). By Proposition 30, there exists a minimum m' of F, $m' \neq m$, such that x and m' are linked for F and $F(x) \geqslant F(m')$. Let $\delta(M)$ denote the set of points of $\overline{M}$ which are adjacent to M. Since M is a minimum of F_X we know that for any $y \in \delta(M)$, $F_X(y) > F_X(M) = F(x)$ and thus for any $y \in \delta(M)$, $F(y) > F(x)$ since $y \in \overline{X}$ and thus $F_X(y) = F(y)$. The fact that x and m' are linked for F thus implies that m' is included in M as well as m, a contradiction with the definition of a minima extension. □

Property 32. *Let $F \in \mathscr{F}(E)$, let X be a minima extension of F, let F_X be the mosaic of F associated with X, let $x \in E$ and let m be a minimum of F. If x is linked to m for F and if $F_X(x) = F(x)$, then x is linked to m for F_X.*

Proof. Since m is a minimum for F and x is linked to m for F, by Proposition 29 we have $F(x) \geqslant F(m)$, thus for any point y of m we have $F(x, y) = F(x)$. Thus, there exists a path $\pi = (x_0, \ldots, x_n)$ from x to m, with $x_0 = x$ and $x_n \in m$, such that $F(\pi) \leqslant F(x)$. For any $i = 1 \ldots n$ we have $F_X(x_i) \leqslant F(x_i)$, thus since $F_X(x) = F(x)$ we have $F_X(\pi) = F_X(x)$. □

Proof of Proposition 12. Let m be any minimum of F, we denote by C_m the connected component of X such that $m \subseteq C_m$. We are going to prove that either (a) C_m is a minimum of F_X, and in this case $F_X(C_m) = F(m)$, or (b) C_m is disjoint with any minimum of F_X. We will also prove that (c) no minimum of F_X is included in $\overline{X}$. It may be seen that the property follows from (a), (b), (c).

(a) Let $\delta(C_m)$ denote the set of points $x \in \overline{C_m}$ which are adjacent to C_m. If all the points x of $\delta(C_m)$ are such that $F(x) > F(m)$, then for any x of $\delta(C_m)$ we have $F_X(x) > F(m)$ (since $x \in \overline{X}$, $F_X(x) = F(x)$). Furthermore, from the very definition of F_X, $\forall z \in C_m$, $F_X(z) \leqslant F(m)$; thus C_m is a minimum for F_X. By Proposition 31, we deduce that $F_X(C_m) = F(m)$.

(b) Suppose now that there exists a point $x \in \delta(C_m)$ such that $F(x) \leqslant F(m)$. Then, x and m are separated for F, otherwise if $F(x) = F(m)$ we would have $x \in m$ and thus $x \in C_m$, and if $F(x) < F(m)$, m would not be a minimum of F (Proposition 29).

Thus, there exists a minimum m' of F, $m' \neq m$, such that x is linked to m' for F and $F(x) \geqslant F(m')$ (Proposition 30). Suppose that $F(x) = F(m')$, since x is linked for F with the minimum m' of F, it would imply that $x \in m'$ (Proposition 29), thus C_m and $C_{m'}$ are adjacent, a contradiction with the definition of the minima extension X. Thus we have

$F(x) > F(m')$. On the other hand, since $x \in \overline{X}$ we have $F_X(x) = F(x)$, thus by Proposition 32, x is linked to m' for F_X. Now two cases must be distinguished.

- If $F_X(C_m) = F(m)$, then we have $F_X(C_m) = F(m) \geqslant F(x) > F(m') \geqslant F_X(m')$, thus C_m is linked to m' for F_X with $F_X(C_m) > F_X(m')$. Now suppose that C_m has a non-empty intersection with a minimum M of F_X. Thus both C_m and M are flat for F_X with the same altitude and since M is a minimum, we have $C_m \subseteq M$. The fact that C_m is linked to m', with $F_X(C_m) > F_X(m')$, raises a contradiction with the fact that M is a minimum of F_X.
- If $F_X(C_m) < F(m)$, then there exists a point $y \in C_m$ such that $F(m) > F(y) = F_X(C_m)$, thus $F(y) = F_X(y)$. Since $F(y) < F(m)$ and m is a minimum of F, we know that y does not belong to m, and with the same arguments as above we see that y and m are separated for F. Thus, there exists a minimum m' of F, $m' \neq m$, such that y is linked to m' for F and $F(y) \geqslant F(m')$. As above, we can see that indeed $F(y) > F(m')$, that y is linked to m' for F_X, that $F_X(C_m) = F(y) > F(m') \geqslant F_X(m')$, and finally that C_m cannot have a non-empty intersection with a minimum of F_X.

(c) Let M be any subset of $\overline{X}$ which is flat for F_X (thus M is also flat for F), and let k denote $F_X(M)$ (which is equal to $F(M)$). Since X is a minima extension for F, we know that M is not a minimum of F, thus there exists a point y of $\overline{M}$ adjacent to M such that $F(y) \leqslant k$. Hence, $F_X(y) \leqslant k$ and M is not a minimum of F_X. □

References

[1] G. Bertrand, Some properties of topological greyscale watersheds, in: L.J. Latecki, D.M. Mount, A.Y. Wu (Eds.), Vision Geometry XII. Number 5300 in IS&T/SPIE Symposium on Electronic Imaging, San-Jose, CA, USA, SPIE, 2004, pp. 127–137.

[2] G. Bertrand, J. Everat, M. Couprie, Image segmentation through operators based upon topology, J. Electronic Imaging 6 (1997) 395–405.

[3] S. Beucher, Segmentation d'images et morphologie mathématique, Ph.D. Thesis, École des Mines de Paris, France, 1990.

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

[5] S. Beucher, F. Meyer, The morphological approach to segmentation: the watershed transformation, in: E. Dougherty (Ed.), Mathematical Morphology in Image Processing, Marcel Decker, 1993, pp. 433–481.

[6] S. Beucher, Watershed, hierarchical segmentation and waterfall algorithm, in: J. Serra, P. Soille (Eds.), Proceedings of Mathematical Morphology and its Applications to Image Processing, Fontainebleau, France, Kluwer, 1994, pp. 69–76.

[7] M. Couprie, G. Bertrand, Topological grayscale watershed transform, in: SPIE Vision Geometry V Proceedings, vol. 3168, 1997, pp. 136–146.

[8] M. Couprie, G. Bertrand, Tesselations by connection in orders, in: Discrete geometry for computer imagery, Lecture Notes in Computer Science, vol. 1953, Springer, Berlin, 2000, pp. 15–26.

[9] M. Couprie, F.N. Bezerra, G. Bertrand, Topological operators for grayscale image processing, J. Electronic Imaging 10 (2001) 1003–1015.

[10] M. Couprie, L. Najman, G. Bertrand, Quasi-linear algorithms for topological watershed, Journal of Mathematical Imaging and Vision (2005), Special issue on Mathematical Morphology, to appear.

[11] H. Digabel, C. Lantuéjoul, Iterative algorithms, in: J.L. Chermant (Ed.), Second European Symposium on Qualitative Analysis of Microstructures in Material Science, Biology and Medicine, Stuttgart FRG, Riederer Verlag, 1978, pp. 85–99.
[12] E. Dougherty, R. Lotufo, Hands-on Morphological Image Processing, SPIE Press, 2003.
[13] V. Goetcherian, From binary to grey tone image processing using fuzzy logic concepts, Pattern Recognition 12 (1980) 7–15.
[14] H. Heijmans, Morphological Image Operators, Academic Press, New York, 1994.
[15] C. Jordan, Nouvelles observations sur les lignes de faîtes et de thalweg, Comptes Rendus des Séances de l'Académie des Sciences 75 (1872) 1023–1025.
[16] R.A. Lotufo, A.X. Falcao, F.A. Zampirolli, Ift-watershed from gray-scale marker, in: SIBGRAPI'02, Fortaleza-CE, Brazil, 2002, pp. 146–152.
[17] J. Maxwell, On hills and dales, Philosophical Magazine 4/40 (1870) 421–427.
[18] F. Meyer, Un algorithme optimal de ligne de partage des eaux, in: Actes du 8ème Congrès AFCET, Lyon-Villeurbanne, France, 1991, pp. 847–859.
[19] F. Meyer, Topographic distance and watershed lines, Signal Processing 38 (1994) 113–126 (special issue on Mathematical Morphology).
[20] F. Meyer, S. Beucher, Morphological segmentation, Journal of Visual Communication and Image Representation 1 (1990) 21–46.
[21] L. Najman, M. Couprie, Watershed algorithms and contrast preservation, in: DGCI'03. Lecture Notes in Computer Sciences, vol. 2886, Springer, Berlin, 2003, pp. 62–71.
[22] L. Najman, M. Couprie, Quasi-linear algorithm for the component tree, in: L.J. Latecki, D.M. Mount, A.Y. Wu (Eds.), Vision Geometry XII. Number 5300 in IS&T/SPIE Symposium on Electronic Imaging, San-Jose, CA, USA, SPIE, 2004, pp. 98–107.
[23] L. Najman, Morphologie Mathématique: de la Segmentation d'Images à l'Analyse Multivoque, Ph.D. Thesis, Université Paris-Dauphine, 1994.
[24] L. Najman, M. Schmitt, Watershed of a continuous function, Signal Processing 38 (1994) 99–112 (special issue on Mathematical Morphology).
[25] L. Najman, M. Schmitt, Geodesic saliency of watershed contours and hierarchical segmentation, IEEE Trans. on PAMI 18 (1996) 1163–1173.
[26] V. Ranwez, P. Soille, Order independent homotopic thinning for binary and grey tone anchored skeletons, Pattern Recognition Lett. 23 (2002) 687–702.
[27] J. Roerdink, A. Meijster, The watershed transform: definitions, algorithms and parallelization strategies, Fundamenta Informaticae 41 (2000) 187–228.
[28] A. Rosenfeld, Fuzzy digital topology, Information and Control 40 (1979) 76–87.
[29] A. Rosenfeld, On connectivity properties of grayscale pictures, Pattern Recognition 16 (1983) 47–50.
[30] A. Rosenfeld, The fuzzy geometry of image subsets, Pattern Recognition Lett. 2 (1984) 311–317.
[31] P. Soille, Morphologie mathématique: du relief à la dimensionalité–Algorithmes et méthodes, Ph.D. Thesis, Université catholique de Louvain, Partnership with the Paris's School of Mines, 1992.
[32] P. Soille, Morphological Image Analysis, Springer, Berlin, 1999.
[33] P. Soille, M. Ansoult, Automated basin delineation from digital elevation models using mathematical morphology, Signal Processing 20 (1990) 171–182.
[34] P. Soille, L. Vincent, Determining watersheds in digital pictures via flooding simulations, in: M. Kunt (Ed.), Visual Communications and Image Processing'90, vol. 1360, Bellingham, Society of Photo-Instrumentation Engineers, 1990, pp. 240–250.
[35] L. Vincent, Algorithmes morphologiques à base de files d'attente et de lacets, Extension aux graphes (Morphological Algorithms Based on Queues and Loops, with Extension to Graphs), Ph.D. Thesis, School of Mines, Paris, 1990.
[36] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Transactions on PAMI 13 (1991) 583–598.

4 Quasi-linear algorithms for the topological watershed

M. Couprie, L. Najman and G. Bertrand. Quasi-linear algorithms for the topological watershed. *Journal of Mathematical Imaging and Vision*, Volume 22, Issue 2 - 3, May 2005, Pages 231 - 249.
Special Issue on Mathematical Morphology.

Quasi-linear algorithms for the topological watershed

Michel Couprie[a,b] and Laurent Najman[a,b] and Gilles Bertrand[a,b]

(a) Laboratoire A2SI, Groupe ESIEE BP99, 93162 Noisy-le-Grand Cedex France
(b) IGM, Unité Mixte de Recherche CNRS-UMLV-ESIEE UMR 8049

e-mail: (m.couprie,l.najman,g.bertrand)@esiee.fr;
url: www.esiee.fr/∼coupriem/Sdi_eng

Abstract

The watershed transformation is an efficient tool for segmenting grayscale images. An original approach to the watershed [1,9] consists in modifying the original image by lowering some points while preserving some topological properties, namely, the connectivity of each lower cross-section. Such a transformation (and its result) is called a W-thinning, a topological watershed being an "ultimate" W-thinning. In this paper, we study algorithms to compute topological watersheds. We propose and prove a characterization of the points that can be lowered during a W-thinning, which may be checked locally and efficiently implemented thanks to a data structure called component tree. We introduce the notion of M-watershed of an image F, which is a W-thinning of F in which the minima cannot be extended anymore without changing the connectivity of the lower cross-sections. The set of points in an M-watershed of F which do not belong to any regional minimum corresponds to a binary watershed of F. We propose quasi-linear algorithms for computing M-watersheds and topological watersheds. These algorithms are proved to give correct results with respect to the definitions, and their time complexity is analyzed.

Key words: discrete topology, mathematical morphology, watershed, component tree, segmentation

Introduction

The watershed transformation was introduced as a tool for segmenting grayscale images by S. Beucher and C. Lantuéjoul [3] in the late 70's, and is now used as a fundamental step in many powerful segmentation procedures. A popular presentation of the watershed is based on a flooding paradigm. Let us

consider a grayscale image as a topographical relief: the gray level of a pixel becomes the altitude of a point, the basins and valleys of the relief correspond to the dark areas, whereas the mountains and crest lines correspond to the light areas (Fig. $1a_1, a_2$). Let us imagine the surface of this relief being immersed in still water, with holes pierced in local minima. Water fills up basins starting at these local minima, and dams are built at points where waters coming from different basins would meet. As a result, the surface is partitioned into regions or basins which are separated by dams, called watershed lines.

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

A different approach to watersheds, originally proposed by G. Bertrand and M. Couprie [9], is developed in [1,25]. In this approach, we consider a transformation called topological watershed, which modifies a map (*e.g.*, a grayscale image) while preserving some topological properties, namely, the connectivity [1] of each lower cross-section. The motivation for such a condition will appear a little later, when we will discuss the properties of this transformation. Let F be a map and λ be a number, the lower cross-section $\overline{F}[\lambda]$ is the

[1] The notions of connectivity, path, connected components, etc. will be precisely defined in section 1.

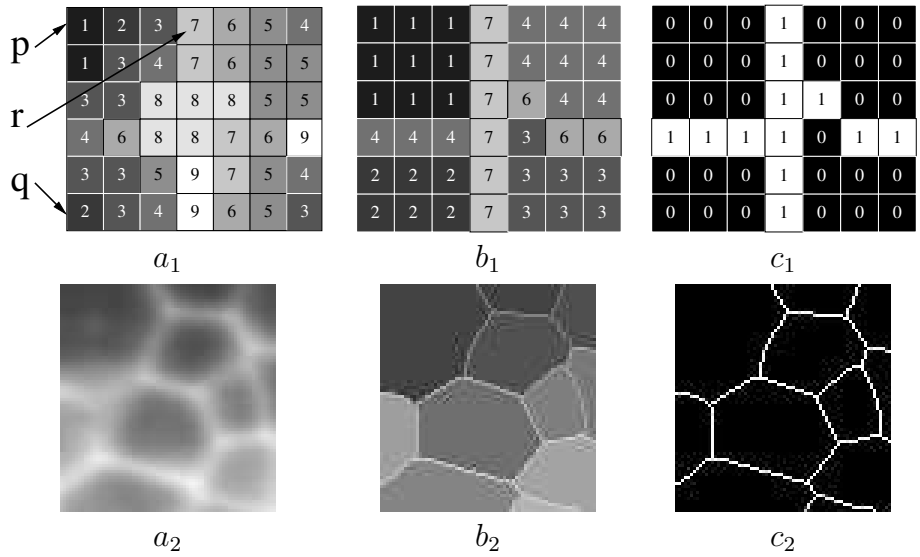


Fig. 1. a_1, a_2: original images; b_1 (resp. b_2): topological watershed of a_1 (resp. a_2); c_1 (resp. c_2): W-crest of a_1 (resp. a_2), in white.

set composed of all the points having an altitude strictly lower than λ (Fig. 3). A point x is said to be W-destructible for F (where W stands for Watershed) if its altitude can be lowered by one without changing the number of connected components of $\overline{F}[k]$, with $k = F(x)$. A map G is called a W-thinning of F if it may be obtained from F by iteratively selecting a W-destructible point and lowering it by one. A topological watershed of F is a W-thinning of F which contains no W-destructible point. This transformation has the effect of spreading the regional minima of the map (see Fig. 1). Let F be a map and let G be a topological watershed of F, the set of points which do not belong to any regional minimum of G is called a W-crest of F. The W-crest of F corresponds to a binary watershed of F (see Fig. $1c_1, c_2$).

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

One of the main theorems proved in [1] (the strong separation theorem) states that G is a W-thinning of F *if and only if* G is a strong separation of F.
The "if" part of the theorem corresponds to a notion of contrast preservation. We will say informally that a transformation "preserves the contrast" if the transformation preserves the altitude of the minima of the image and if, when two minima are separated by a crest in the original image, they are still separated by a crest of the same altitude in the transform. For example in Fig. 1, if we take any two minima which are k-separated in a_i (i = 1,2) for a given k, we know that they are k-separated in b_i since b_i is a W-thinning of a_i. This constrast preservation property is not satisfied in general by the most popular watershed algorithms (see [23,25]).
The "only if" part of the theorem mainly states that, if one needs a transformation which preserves the contrast in this sense, then this transformation is necessarily a W-thinning. This remarkable result shows that the topological watershed is a fundamental tool to obtain a contrast preserving watershed transformation.

In this paper, we study algorithms to compute topological watersheds. A naive

algorithm could be the following: for all p in E (n points), check the number of connected components of the lower cross-section at the level of p which are adjacent to p (cost for each point p: $O(n)$ with a classical connected component labelling algorithm), lower the value of p by one if this number is exactly one. Repeat this whole process until no W-destructible point remains. Consider an image which consists of a single row of $n+2$ points, such that each point has an altitude of g except for the two points at the beginning and at the end of the row, which have an altitude of 0 (with any positive integers n, g). The outer loop will be executed g times. The time complexity of this naive algorithm is thus at least in $O(n^2 \times g)$.
We reduce the complexity by two means.
First, we propose and prove a new characterization of the W-destructible points which may be checked locally and efficiently: the total time for checking the W-destructibleness of all the vertices in a graph with n vertices and m arcs is in $O(n+m)$. We obtain this result thanks to a data structure called component tree, which may be constructed in quasi-linear time [24], that is, in $O(N \times \alpha(N))$ where $N = n+m$ and $\alpha(N)$ is a function which grows extremely slowly with N (we have $\alpha(10^{80}) \approx 4$). This complexity can be reached thanks to a reduction to the disjoint set problem [31].
Second, we propose different strategies to ensure that a point is lowered at most once during the execution of the algorithm. One of these strategies relies on the notions of $\tilde{M}$-point and M-watershed. A point p is an $\tilde{M}$-point if it is adjacent to a regional minimum and if it can be lowered by W-thinning down to the level of this minimum. An M-watershed is obtained by iteratively lowering $\tilde{M}$-points until stability. Recall that a W-crest of a map F is composed by the points which do not belong to any regional minimum of a topological watershed of F. We prove that the set of points which do not belong to any regional minimum of an M-watershed of F is always a W-crest of F, in other words, we can compute a W-crest by only lowering $\tilde{M}$-points. We propose a quasi-linear algorithm for computing an M-watershed — hence a W-crest — of a map.
We also propose a quasi-linear algorithm for the topological watershed transformation. These algorithms are proved to give correct results with respect to the definitions, and their time complexity is analyzed.
In order to ease the reading of the paper, we defer the proofs to the annex.

1 Topological notions for graphs

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

the set $\Gamma(p)$ is called the *neighborhood of* p. If $q \in \Gamma(p)$ then we say that p and q are *adjacent* or that q is a *neighbor of* p. If $X \subseteq E$ and q is adjacent to p for some $p \in X$, we say that q *is adjacent to* X.

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

Let (E, Γ) be a graph, let $X \subseteq E$, and let $p_0, p_k \in X$. A *path from* p_0 *to* p_k *in* X is an ordered family $(p_0, \ldots, p_k)$ of points of X such that $p_{i+1} \in \Gamma(p_i)$, with $i = 0 \ldots k-1$.
Let $p, q \in X$, we say that p *and* q *are linked for* X if there exists a path from p to q in X. We say that X *is connected* if any p and q in X are linked for X. We say that a subset Y of E is a *connected component of* X if $Y \subseteq X$, Y is connected, and Y is maximal for these two properties, *i.e.*, if $Y \subseteq Z \subseteq X$ and if Z is connected, then $Z = Y$. In the sequel of the article, we will assume that E is connected.

We are interested in transformations that preserve the number of connected components of the background. For that purpose, we introduce the notion of W-simple point in a graph. Intuitively, a point of X is W-simple if it may be removed from X while preserving the number of connected components of $\overline{X}$.

Definition 1 *Let* $X \subseteq E$, *let* $p \in X$.
We say that p *is* a border point (for X) *if* p *is adjacent to* $\overline{X}$.
We say that p *is* an inner point (for X) *if* p *is not a border point for* X.
We say that p *is* separating (for X) *if* p *is adjacent to at least two connected components of* $\overline{X}$.
We say that p *is* W-simple (for X) *if* p *is adjacent to exactly one connected component of* $\overline{X}$.

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

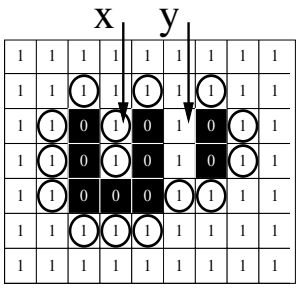


Fig. 2. A set X (the 1's) and its complement $\overline{X}$ (the 0's). The W-simple points are circled.

2 Topological notions for weighted graphs and stacks

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

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

Definition 2 *Let $F = \{F[k]; k \in \mathbb{K}^+\}$ be a family of subsets of E.*
We say that F is an upstack on E *if $F[k_{min}] = E$, $F[k_{max}] = \emptyset$, and $F[j] \subseteq F[i]$ whenever $i \leq j$.*
We say that F is a downstack on E *if $F[k_{min}] = \emptyset$, $F[k_{max}] = E$, and $F[i] \subseteq F[j]$ whenever $i \leq j$.*
We denote by $\mathcal{S}^+$ (resp. $\mathcal{S}^-$) the set of all upstacks (resp. downstacks) on E. Any element of $\mathcal{S}^+ \cup \mathcal{S}^-$ is called a stack on E.
Let F be a stack on E, we define the stack $\overline{F} = \{\overline{F}[k] = \overline{F[k]}; k \in \mathbb{K}\}$ which is called the complement of F. *Let F be a stack on E, any element $F[k]$ of F is called a* section of F (at level k), *or the* k-section of F.
Let $F \in \mathcal{S}^+$ and let $G \in \mathcal{S}^-$. We define the functions induced by F and G, *also denoted by F, G, such that for any $p \in E$:*

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

Important remark: Let $F \in \mathcal{F}$. Clearly, the level sets of F form an upstack (also denoted by F), and the function induced by the upstack F is precisely

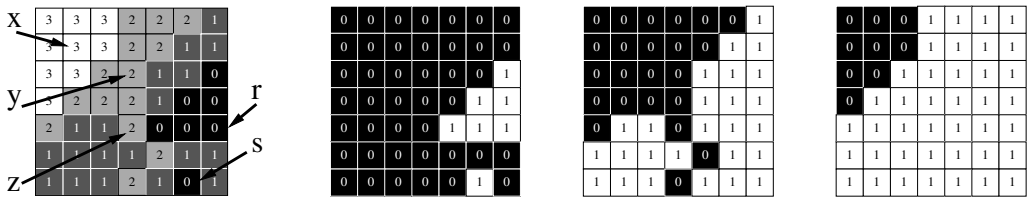


Fig. 3. A grayscale image F and its lower sections $\overline{F}[1]$, $\overline{F}[2]$ and $\overline{F}[3]$ (in white). Notice that $\overline{F}[4] = E$ and $\overline{F}[0] = \emptyset$.

the function F. The complement $\overline{F}$ of the upstack F is a downstack. For any $k \in \mathbb{K}^+$, we have $\overline{F}[k] = \overline{F[k]} = \{p \in E; F(p) < k\} = \{p \in E; \overline{F}(p) \leq k\}$; and for any $p \in E$, we have

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

Definition 3 *Let $F \in \mathcal{F}$, let $k \in \mathbb{K}^+$. A connected component of a non-empty k-section of $\overline{F}$ is called a* component of $\overline{F}$ (at level k)*, or a* k-component of $\overline{F}$*. A component m of $\overline{F}$ is said to be a* minimum of $\overline{F}$ *(and also a* minimum of F*) if there is no other component of $\overline{F}$ which is included in m.*
Let $p \in E$, the component of p in $\overline{F}$*, denoted by $C(p, \overline{F})$ or simply by $C(p)$ when no confusion may occur, is defined as the component of $\overline{F}[k]$ which contains p, with $k = \overline{F}(p)$.*
We denote by $\Gamma^-(p, F)$ the set of lower neighbors of the point p for the function F*, that is, $\Gamma^-(p, F) = \{q \in \Gamma(p); F(q) < F(p)\}$. Notice that $\Gamma^-(p, F) = \Gamma^-(p, \overline{F})$. When no confusion may occur, we write $\Gamma^-(p)$ instead of $\Gamma^-(p, F)$.*

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

Definition 4 *Let $F \in \mathcal{F}$, let $p \in E$, let $k = F(p)$.*
We say that p is a border point (for F) if p is an border point for $F[k]$.
We say that p is an inner point (for F) if p is an inner point for $F[k]$.
We say that p is separating (for F) if p is separating for $F[k]$.
The point p is W-destructible (for F) *if p is W-simple for $F[k]$. Let $v \in \mathbb{K}$, the point p is* W-destructible with lowest value v (for F) *if for any h such that $v < h \leq F(p)$, p is W-simple for $F[h]$, and if p is not W-simple for $F[v]$.*

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

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

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

Definition 5 *Let $F \in \mathcal{F}$. We say that $G \in \mathcal{F}$ is a W-thinning of F if*
i) $G = F$, or if
ii) there exists a function H which is a W-thinning of F and there exists a W-destructible point p for H such that $G = [H \setminus p]$.
We say that G is a (topological) watershed of F if G is a W-thinning of F and if there is no W-destructible point for G.

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

In other words, one can obtain a W-thinning of a function F by iteratively selecting a W-destructible point and lowering it by one. If this process is repeated until stability, one obtains a topological watershed of F. Notice that the choice of the W-destructible point is not necessarily unique at each step, thus, in general, there may exist several topological watersheds for the same function.

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

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

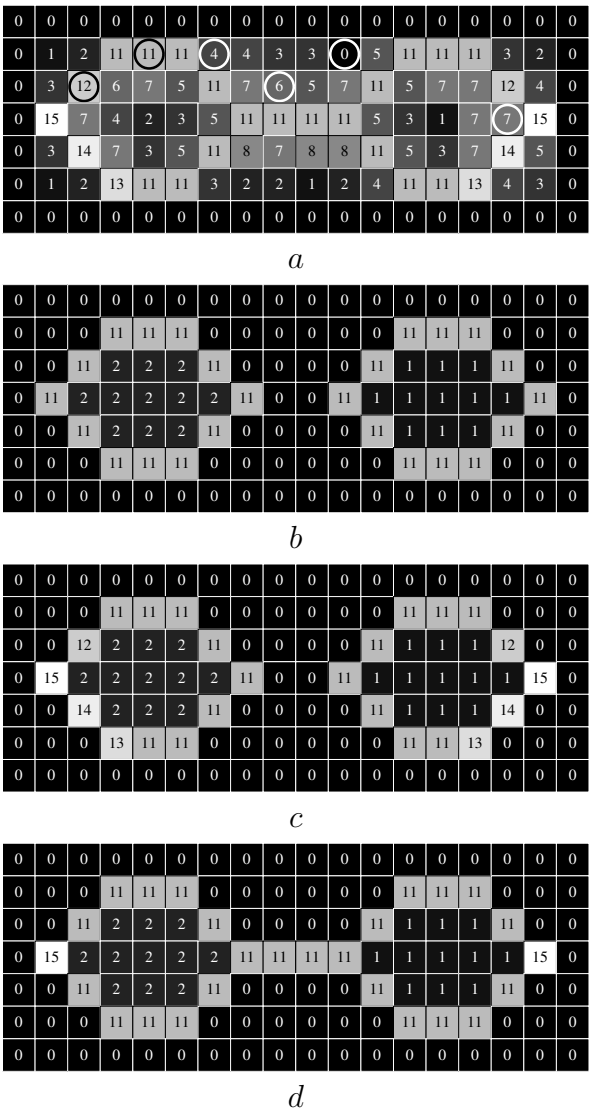


Fig. 4. a: original image; b: a topological watershed of a; c: a W-thinning of a which is also an M-watershed of a (see section 6); d: an homotopic grayscale skeleton of a. In a, we have circled six points which have different types (see Section 3). From left to right: $\tilde{S}$-point (12), S-point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

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

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

Definition 6 *Let $F \in \mathcal{F}$, let p and q be two points of E, and let $k \in \mathbb{K}^+$. We say that* p and q are k-linked (in $\overline{F}$) *if p and q are linked for $\overline{F}[k]$.*

We say that p dominates q (in $\overline{F}$) *if* q *belongs to the component of* p *in* $\overline{F}$.
We say that p and q are linked (in $\overline{F}$) *if* p *dominates* q *in* $\overline{F}$ *or* q *dominates* p *in* $\overline{F}$.
We define the connection value between p and q (for $\overline{F}$) *by:*
$\overline{F}(p,q) = min\{k;\ p$ *and* q *are* k*-linked in* $\overline{F}\}$.
We say that p and q are separated (in $\overline{F}$) *if* p *and* q *are not linked in* $\overline{F}$.
We say that p and q are k-separated (in $\overline{F}$) *if* p *and* q *are separated in* $\overline{F}$ *and if the connection value for* $\overline{F}$ *between* p *and* q *is precisely* k*, i.e., if* $\overline{F}(p,q) = k$.

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

Property 1 ([1]) *Let* $F \in \mathcal{F}$. *Two points* p *and* q *are* k*-separated in* $\overline{F}$, *if and only if:*
i) p *and* q *belong to the same component of* $\overline{F}[k]$, *and*
ii) p *and* q *belong to distinct components of* $\overline{F}[k-1]$.

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

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

3 Classification of points and transitions

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

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

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

M for minimum and P for plateau).

Definition 7 *Let $F \in \mathcal{F}$, let $p \in E$, p not W-destructible for F.*
We say that p is an S-point (for F) *if p is separating for F.*
We say that p is an I-point (for F) *if p is an inner point for F.*
We say that p is an M-point (for F) *if p belongs to a minimum of F.*
We say that p is a P-point (for F) *if p is an inner point for F which does not belong to a minimum of F.*
Let q be a point which is W-destructible for F, and let v be its lowest value.
We say that q is an $\tilde{S}$-point (for F) *(resp. an* $\tilde{I}$-point*, an* $\tilde{M}$-point*, a* $\tilde{P}$-point*) if q is an S-point for $[F \setminus q \downarrow v]$ (resp. an* I-point*, an* M-point*, a* P-point*).*
If p is a T-point, with $T \in \{S, M, P, \tilde{S}, \tilde{M}, \tilde{P}\}$, we say that T is the type of p.

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

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

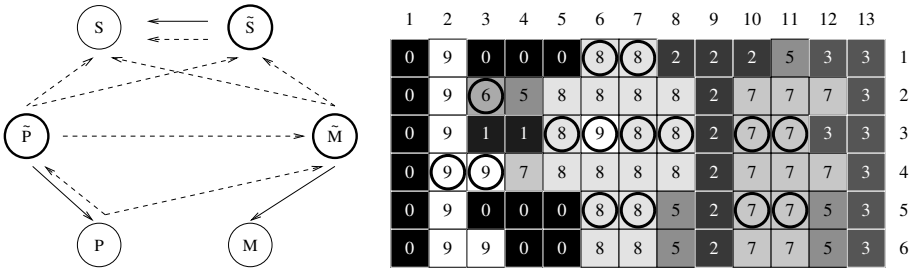
Property 3 *Let $F \in \mathcal{F}$, let $p \in E$, let $v \in \mathbb{K}$.*
The point p is an $\tilde{I}$-point for F with lowest value v if and only if:
i) $\Gamma^-(p) \neq \emptyset$*; and*
ii) any two points q, r in $\Gamma^-(p)$ are linked in $\overline{F}$; and
iii) $v = min\{F(q); q \in \Gamma^-(p)\}$.

Property 4 *Let $F \in \mathcal{F}$, let $p \in E$, let $v \in \mathbb{K}$.*
The point p is an $\tilde{S}$-point for F with lowest value v if and only if:
i) $\Gamma^-(p) \neq \emptyset$*; and*
ii) $\forall q, r \in \Gamma^-(p)$*, if q and r are k-separated in $\overline{F}$ then $k \leq v + 1$; and*
iii) there exist two points q, r in $\Gamma^-(p)$ which are $(v+1)$-separated in $\overline{F}$.

The type of a point p depends on the connected components of the sections of $\overline{F}$ which are adjacent to p, and we know that lowering a W-destructible point preserves the connectivity of all these sections. It may thus be seen that, during a W-thinning process, the type of a point p can only be changed by the modification of either the point p itself or a neighbor of p (this will be proved with the following theorem). By a systematic examination of all the possibilities, we deduce that only certain transitions are possible for the type of a point p during a W-thinning process (all of them are illustrated in Fig. 5).

Theorem 5 *Let $F \in \mathcal{F}$, let $p \in E$. During a W-thinning process, the possible*

transitions for the type of the point p are exactly those depicted in the graph of Fig. 5a, where the solid lines correspond to transitions due to the lowering of the point p itself, and the dashed lines correspond to transitions due to the lowering of a neighbor of p.



a *b*

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

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

4 Component tree

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

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

We see easily that these components can be organized in a tree structure, that we call component tree. This structure has been introduced in the domain of data analysis [35,14], and appears to be a fundamental tool to represent some "meaningful" information contained in a numerical function [13,12]. Several authors, such as Vachier [33], Breen and Jones [7,16], Salembier et al. [30],

Meijster and Wilkinson [21] have used this structure in order to implement efficiently some morphological operators (*e.g.*, connected operators, granulometries, extinction functions). The component tree has also be used as a basis for image matching algorithms [18,19]. Algorithms to compute the component tree for the case of digital images can be found in [7,30,20]; the last reference also contains a discussion about time complexity of the different algorithms. Until recently, the fastest algorithm to compute the component tree was proved to run in $O(n \times \ln(n))$ complexity, where n is the number of image points. L. Najman and M. Couprie have proposed a quasi-linear algorithm [24]. For the sake of completeness, we present this algorithm in Annex 2. Let us now give a formal definition of the component tree and related notions.

Definition 8 *Let $F \in \mathcal{F}$, let $[k,c], [k_1,c_1], [k_2,c_2]$ be elements of $\mathcal{C}(\overline{F})$.*
We say that $[k_1,c_1]$ is the parent *of $[k_2,c_2]$ if $k_1 = k_2 + 1$ and $c_2 \subseteq c_1$, in this case we also say that $[k_2,c_2]$ is a* child *of $[k_1,c_1]$.*
With this relation "parent", $\mathcal{C}(\overline{F})$ forms a directed tree that we call the component tree of $\overline{F}$*, and that we will also denote by $\mathcal{C}(\overline{F})$ by abuse of terminology.*
An element of $\mathcal{C}(\overline{F})$ which has no child is called a leaf*, and an element of $\mathcal{C}(\overline{F})$ which has at least two childs is called a* fork*.*

Fig. 6*b* shows the component tree associated to the function depicted in Fig. 6*a*.

Definition 9 *We say that $[k_1,c_1]$ is an* ancestor of *$[k_2,c_2]$ if $k_2 \leq k_1$ and $c_2 \subseteq c_1$. In this case, we also say that $[k_1,c_1]$ is* over *$[k_2,c_2]$, and that $[k_2,c_2]$ is* under *$[k_1,c_1]$.*
We say that the component $[k,c]$ is a common ancestor of *$[k_1,c_1]$, $[k_2,c_2]$ if $[k,c]$ is an ancestor of both $[k_1,c_1]$ and $[k_2,c_2]$.*
We say that the component $[k,c]$ is the least common ancestor of *$[k_1,c_1]$, $[k_2,c_2]$, and we write $[k,c] = LCA([k_1,c_1], [k_2,c_2])$, if $[k,c]$ is a common ancestor of $[k_1,c_1]$, $[k_2,c_2]$, and if there is no other common ancestor of $[k_1,c_1]$, $[k_2,c_2]$ under $[k,c]$.*
We say that the component $[k,c]$ is the proper least common ancestor of *$[k_1,c_1]$ and $[k_2,c_2]$ if $[k,c]$ is the least common ancestor of $[k_1,c_1]$, $[k_2,c_2]$, and if $[k,c]$ is different from $[k_1,c_1]$ and from $[k_2,c_2]$.*

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

Definition 10 *We say that the components $[k_1,c_1]$, $[k_2,c_2]$ are* separated *if they have a proper least common ancestor, otherwise we say that they are* linked.
Let M be a set $\{[k_1,c_1], [k_2,c_2], \ldots, [k_n,c_n]\}$ of elements of $\mathcal{C}(\overline{F})$. We say that the component $[k,c]$ is the highest fork for M *if the two following conditions*

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

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

We make the following observations:
a) Any two components always have a unique least common ancestor. In particular, if $[k_1, c_1]$ is over $[k_2, c_2]$, then $\mathrm{LCA}([k_1, c_1], [k_2, c_2]) = [k_1, c_1]$. On the other hand, two components which are linked have no proper least common ancestor.
b) Two components are separated if and only if they are disjoint; and two components $[k_1, c_1]$, $[k_2, c_2]$ are linked if and only if either $c_1 \subseteq c_2$ or $c_2 \subseteq c_1$ (see Annex 1, lemma 6.1).
c) A set of components may have no highest fork, and if the highest fork exists, it is indeed a fork, *i.e.*, an element with at least two childs (otherwise it could not be a proper least common ancestor).
d) If a set of components has a highest fork, then this highest fork is unique.

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

Property 6 *Let* $F \in \mathcal{F}$*, let* $p, q \in E$*, let* $k = \overline{F}(p), l = \overline{F}(q)$*, let* $h \in \mathbb{K}$*.*
i) The points p, q *are* h*-separated in* $\overline{F}$ *if and only if* $[k, C(p)]$ *and* $[l, C(q)]$ *are separated and their proper least common ancestor in* $\mathcal{C}(\overline{F})$ *is a component of altitude* h*.*
ii) The points p, q *are linked in* $\overline{F}$ *if and only if the components* $[k, C(p)]$ *and* $[l, C(q)]$ *are linked.*

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

Let X, Y be non-empty subsets of E such that $X \subseteq Y$. We say that Y is an *extension of* X if each connected component of Y contains exactly one connected component of X. We also say that Y is an extension of X if X and Y are both empty.
We denote by $\mathcal{C}(X)$ the set composed of all connected components of X. If Y is an extension of X, the *extension map relative to* (X, Y) is the bijection σ from $\mathcal{C}(X)$ to $\mathcal{C}(Y)$ such that, for any $C \in \mathcal{C}(X)$, $\sigma(C)$ is the connected component of Y which contains C.
Let F, G be two stacks. We say that G is an *extension* of F if, for any $k \in \mathbb{K}^+$, $G[k]$ is an extension of $F[k]$, and we denote by σ_k the extension map relative

to $(F[k], G[k])$.

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

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

5 Characterization of W-destructible points

We saw in section 1 that checking whether a point is W-simple cannot be done locally (*i.e.*, based on the mere knowledge of the status of the point and its neighbors), thus checking whether a point is W-destructible or not cannot be done locally if the only available information is the graph (E, Γ) and the function F. As discussed in the introduction, with a naive approach a connected component search (at least in $O(n)$, with $n = |E|$) is necessary for each tested point, thus the complexity of a naive topological watershed algorithm has a term in n^2; furthermore, a point may be lowered several times until it is no more W-destructible. The following theorem and algorithms make it possible to perform this test on all the vertices of a weighted graph in linear time, and also to check directly how low the W-destructible point may be lowered until it is no more W-destructible (its lowest value), thanks to the component tree which may be built in quasi-linear time. In addition, the proposed algorithm provides the type of the considered point.

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

Theorem 9 *Let $F \in \mathcal{F}$, let $p \in E$.*
We denote by $V(p)$ the set $\{[\overline{F}(q), C(q)], q \in \Gamma^-(p)\}$. Then:
i) The point p is an $\tilde{I}$-point for F if and only if $V(p) \neq \emptyset$ and $V(p)$ has no highest fork in $\mathcal{C}(\overline{F})$; in this case the lowest value of p is $w - 1$, where w denotes the altitude of the lowest element of $V(p)$.
ii) The point p is an $\tilde{S}$-point for F if and only if $V(p) \neq \emptyset$ and $V(p)$ has a highest fork in $\mathcal{C}(\overline{F})$, the altitude of which is $v \leq F(p)$; in this case the lowest value of p is $v - 1$.

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

In Fig. 6, we illustrate the characterization of W-destructible points using theorem 9. The function F (grayscale image) is depicted in (a), and four sections of $\overline{F}$ are shown in the bottom row. Each component of these sections

is identified by a letter. The component tree $\mathcal{C}(\overline{F})$ is shown in (b), and the component mapping Ψ in (c). From top to bottom, let us consider the four circled points p_1, p_2, p_3, p_4. Thanks to the component mapping Ψ, we can build the sets $V(p_1) = \{[1, a]\}$ (no highest fork), $V(p_2) = \{[1, a], [2, c], [3, g]\}$ (no highest fork), $V(p_3) = \{[1, b], [2, e], [3, g]\}$ (highest fork $= [3, g]$), and $V(p_4) = \{[1, b], [2, e]\}$ (highest fork $= [3, g]$). From theorem 9 we conclude that:
• p_1 and p_2 are $\tilde{I}$-points (thus they are W-destructible) and may be lowered down to 0 (they are $\tilde{M}$-points),
• p_3 is an $\tilde{S}$-point (thus p_3 is W-destructible) with lowest value 2,
• p_4 is not W-destructible (p_4 is an S-point).

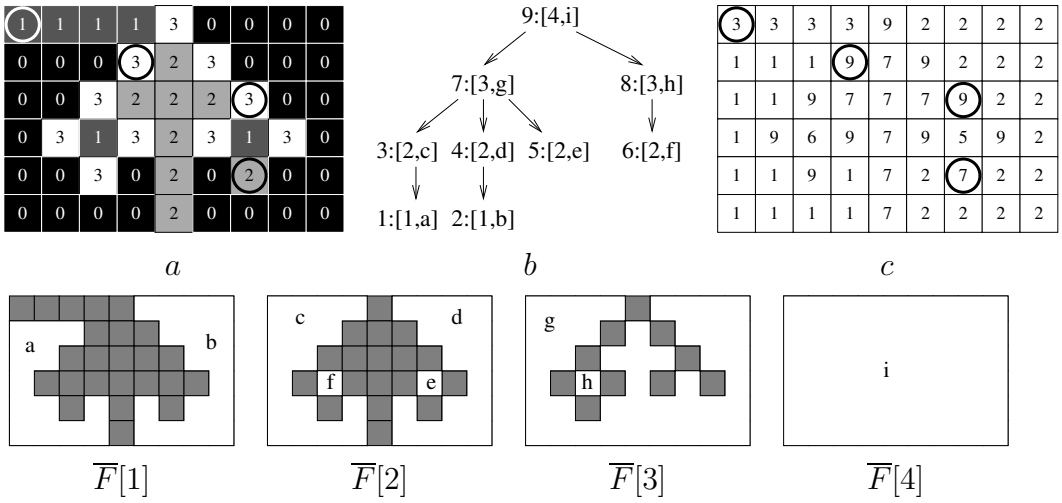


Fig. 6. Illustration of theorem 9. a: original image F; b: component tree $\mathcal{C}(\overline{F})$; c: component mapping Ψ; bottom row: sections $\overline{F}[1]$, $\overline{F}[2]$, $\overline{F}[3]$, $\overline{F}[4]$ (in white, with their components labelled by letters).

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

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

Let $\mathcal{C}$ be a component tree, let V be a set of components of $\mathcal{C}$, we denote by $\min(V)$ an element of V which has the minimal altitude. For this algorithm

and the following ones, we assume that $\mathcal{C}$ is represented in a convenient manner for **BLCA**.

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

Property 10 *Let $\mathcal{C}$ be a component tree, and let V be a non-empty set of components of $\mathcal{C}$.*
i) The algorithm **HighestFork** *returns the highest fork of the set V, or the indicator $[\infty, \emptyset]$ if there is no highest fork.*
ii) This algorithm makes $n - 1$ calls of the **BLCA** *operator, where n is the number of elements in V.*

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

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

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

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

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

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

From the previous properties and observations, we deduce straightforwardly:

Property 11 *Algorithms* **TestType**, **W-Destructible** *and* **M-destructible** *give correct results with regard to the definition of the different types of points (Defs. 4 and 7), and are linear in time complexity with respect to the number of neighbors of p.*

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

6 M-Thinning and binary watershed algorithm

The outline of a topological watershed algorithm is the following:
Repeat Until Stability
Select a W-destructible point p, using a certain criterion
Lower the value of p
It can be seen that, even if a W-destructible point is lowered down to its lowest value, it may again become W-destructible in further steps of the W-thinning process, due to the lowering of some of its neighbors. For example, the point at level 6 circled in white in Fig. 4*a* is W-destructible with lowest value 3. If we lower this point down to 3, we will have to lower it again, after the lowering of its neighbor at level 3 down to 0.

In order to ensure a linear complexity, we must avoid multiple selections of the same point during the execution of the algorithm. The properties of this

section and the following one provide selection criteria which guarantee that a point lowered once will never be W-destructible again during the W-thinning process.

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

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

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

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

In the following algorithm, we introduce a priority function μ which is used to select the next $\tilde{M}$-point. The priority function μ associates to each point p a positive integer $\mu(p)$, called the priority of p. This function is used for the management of a priority queue, a data structure which allows to perform efficiently, on a set of points, an arbitrary sequence of the two following operations (L denotes a priority queue and p a point):
AddPriorityQueue$(L, p, \mu(p))$: store the point p with the priority $\mu(p)$ into the queue L;
ExtractPriorityQueue(L): remove and return a point which has the minimal priority value among those stored in L (if several points fulfill this condition, an arbitrary choice is made).
The choice and the interest of the priority function will be discussed afterwards, but notice that whatever the chosen priority function (for example a constant function), the result will always be an M-watershed of the input.

Procedure M-watershed (**Input** F, $\mathcal{C}(\overline{F})$, Ψ , μ ; **Output** F)
01. $L \leftarrow$ EmptyPriorityQueue
02. **For All** $p \in E$ **Do**
03. **If M-destructible**(F, p, $\mathcal{C}(\overline{F})$, Ψ) $\neq [\infty, \emptyset]$ **Then**

```
04.              AddPriorityQueue(L, p, μ(p)) ; mark p
05.      While  L ≠ EmptyPriorityQueue Do
06.          p ← ExtractPriorityQueue(L) ; unmark p
07.          [i, c] ← M-destructible(F, p, C(F̄), Ψ)
08.          If [i, c] ≠ [∞, ∅] Then
09.              F(p) ← i − 1 ; Ψ(p) ← pointer to [i, c]
10.              For All  q ∈ Γ(p), q ≠ p, q not marked Do
11.                  If M-destructible(F, q, C(F̄), Ψ) ≠ [∞, ∅] Then
12.                      AddPriorityQueue(L, q, μ(q)) ; mark q
```

The following property is a direct consequence of property 7, theorem 8, theorem 9, property 11 and of the fact that, obviously, each point is selected at most once by this algorithm.

Property 13 *Whatever the chosen priority function, the output of Procedure* **M-watershed** *is an M-watershed of the input.*
The time complexity of Procedure **M-watershed** *is in* $O(n+m)+k$*, where* k *is the overall complexity for the management of the priority queue.*

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

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

Let $F \in \mathcal{F}$, let d be a distance on E, let $p \in E$. We denote by $D(p)$ be the minimal distance between p and any point q strictly lower than p, that is, $D(p) = \min\{d(p,q); F(q) < F(p)\}$.
It is easy to build a function μ such that, for any p, q in E:
- if $F(p) < F(q)$ then $\mu(p) > \mu(q)$;
- if $F(p) = F(q)$ and $D(p) \leq D(q)$ then $\mu(p) \geq \mu(q)$.

The efficient management of priority queues is the subject of many articles. Recently, a priority queue algorithm has been proposed by M. Thorup [32], which allows an operation of insertion, extraction of the minimal element or deletion to be performed in $O(\log\log m)$, where m is the number of elements

stored in the structure. This cost can be regarded as constant for practical applications. Furthermore, in most current situations of image analysis, where the number of possible values for the priority function is limited and the number of neighbors of a point is a small constant, specific linear algorithms can be used. An example of such a linear strategy is given in the next section, with algorithm **TopologicalWatershed**.

7 Watershed algorithm

After iteratively lowering $\tilde{M}$-points until stability, we have to process the other W-destructible points in order to get a watershed. Let $F \in \mathcal{F}$, let us call an *MS-watershed of* F a function obtained from F by iteratively lowering $\tilde{M}$-points and $\tilde{S}$-points until stability. We could think that all $\tilde{P}$-points will be eventually changed to $\tilde{M}$-points and then M-lowered in such a process, as it is the case for images like Fig. 4*a*. But the examples of Fig. 7 show that it is not always the case, in other words, an MS-watershed of F is not always a topological watershed of F. Furthermore, there may exist thick regions made of $\tilde{P}$-points in an MS-watershed, and although $\tilde{M}$-points and $\tilde{S}$-points may be lowered directly down to their lowest possible value, we have no such guarantee for the $\tilde{P}$-points (see theorem 5).

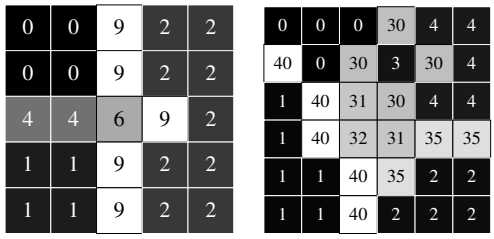


Fig. 7. Examples of W-destructible points in an MS-watershed which are neither $\tilde{M}$-points nor $\tilde{S}$-points: the point at 6 in the image on the left, the points at 31 and 32 in the image on the right.

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

Procedure TopologicalWatershed (**Input** F, $\mathcal{C}(\overline{F})$, Ψ ; **Output** F)
01. **For** k **From** $k_{\min}$ **To** $k_{\max} - 1$ **Do** $L_k \leftarrow \emptyset$
02. **For All** $p \in E$ **Do**
03. $[i, c] \leftarrow$ **W-Destructible**(F, p, $\mathcal{C}(\overline{F})$, Ψ)
04. **If** $i \neq \infty$ **Then**
05. $L_{i-1} \leftarrow L_{i-1} \cup \{p\}$; $K(p) \leftarrow i - 1$; $H(p) \leftarrow$ pointer to $[i, c]$

```
06.      For  k = k_min To  k_max − 1 Do
07.          While  ∃p ∈ L_k Do
08.              L_k = L_k \ {p}
09.              If  K(p) = k Then
10.                  F(p) ←  k ; Ψ(p) ←  H(p)
11.                  For All  q ∈ Γ(p), k < F(q) Do
12.                      [i, c] ← W-Destructible(F, q, C(F̄), Ψ)
13.                      If i = ∞ Then K(q) ← ∞
14.                      Else If K(q) ≠ i − 1 Then
15.                              L_{i−1} ← L_{i−1} ∪ {q} ; K(q) ←  i − 1
16.                              H(q) ←  pointer to [i, c]
```

We have the following guarantees:

Property 14 *In algorithm* **TopologicalWatershed**,
i) at the end of the execution, F is a topological watershed of the input function;
ii) let n and m denote respectively the number of vertices and the number of arcs in the graph (E, Γ). If $k_{max} - k_{min} \leq n$, then the time complexity of the algorithm is in $O(n + m)$.

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

8 Conclusion

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

References

[1] G. Bertrand, "On topological watersheds", *Journal of Mathematical Imaging and Vision*, to appear in this issue, 2005.

[2] G. Bertrand, J. C. Everat, M. Couprie, "Image segmentation through operators based upon topology", *Journal of Electronic Imaging*, Vol. 6, No. 4, pp. 395-405, 1997.

[3] S. Beucher, Ch. Lantuéjoul, "Use of watersheds in contour detection", *Proc. Int. Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, 1979.

[4] S. Beucher, F. Meyer, "The morphological approach to segmentation: the watershed transformation", *Mathematical Morphology in Image Processing*, Chap. 12, pp. 433-481, Dougherty Ed., Marcel Dekker, 1993.

[5] M.A. Bender, M. Farach-Colton, "The LCA problem revisited", *Proc. 4th Latin American Symposium on Theoretical Informatics*, LNCS, Vol. 1776, pp. 88-94, Springer, 2000.

[6] U.M. Braga-Neto, J. Goutsias, "A theoretical tour of connectivity in image processing and analysis", *Journal of Mathematical Imaging and Vision*, Vol. 19, pp. 5-31, 2003.

[7] E.J. Breen, R. Jones, "Attribute openings, thinnings and granulometries", *Computer Vision and Image Understanding*, Vol. 64, No. 3, pp. 377-389, 1996.

[8] T. H. Cormen, C. Leiserson, R. Rivest, *Introduction to algorithms*, McGraw-Hill, 1990.

[9] M. Couprie, G. Bertrand, "Topological grayscale watershed transformation", *Proc. SPIE Vision Geometry VI*, Vol. 3168, pp. 136-146, 1997.

[10] M. Couprie, F.N. Bezerra, G. Bertrand, "Topological operators for grayscale image processing", *Journal of Electronic Imaging*, Vol. 10, No. 4, pp. 1003-1015, 2001.

[11] V. Goetcherian, "From binary to grey tone image processing using fuzzy logic concepts", *Pattern Recognition*, Vol. 12, No. 12, pp. 7-15, 1980.

[12] P. Guillataud, *Contribution à l'analyse dendroniques des images*, PhD thesis of Université de Bordeaux I, 1992.

[13] P. Hanusse, P. Guillataud, "Sémantique des images par analyse dendronique", *8th Conf. Reconnaissance des Formes et Intelligence Artificielle*, Vol. 2, pp. 577-588, AFCET Ed., Lyon, 1992.

[14] J.A. Hartigan, "Statistical theory in clustering", *Journal of classification*, No. 2, pp. 63-76, 1985.

[15] D. Harel, R.E. Tarjan, "Fast algorithms for finding nearest common ancestors", *SIAM J. Comput.*, Vol. 13, No. 2, pp. 338-355, 1984.

[16] Ronald Jones, "Connected filtering and segmentation using component trees", *Computer Vision and Image Understanding*, Vol. 75, No. 3, pp. 215-228, 1999.

[17] T.Y Kong, A. Rosenfeld, "Digital topology: introduction and survey", *Computer Vision, Graphics and Image Processing*, Vol. 48, pp. 357-393, 1989.

[18] J. Mattes, J. Demongeot, "Tree representation and implicit tree matching for a coarse to fine image matching algorithm", *Proc. MICCAI, LNCS*, Springer, Vol. 1679, pp. 646-655, 1999.

[19] J. Mattes, M. Richard, J. Demongeot, "Tree representation for image matching and object recognition", *Proc. DGCI, LNCS*, Springer, Vol. 1568, pp. 298-309, 1999.

[20] J. Mattes, J. Demongeot, "Efficient algorithms to implement the confinement tree", *Proc. DGCI, LNCS*, Springer, Vol. 1953, pp. 392–405, 2000.

[21] A. Meijster and M. Wilkinson, "A comparison of algorithms for connected set openings and closings", *IEEE PAMI*, Vol. 24, pp. 484-494, 2002.

[22] F. Meyer, "Un algorithme optimal de ligne de partage des eaux", *Proc. 8th Conf. Reconnaissance des Formes et Intelligence Artificielle*, Vol. 2, pp. 847-859, AFCET Ed., Lyon, 1991.

[23] L. Najman, M. Couprie, "Watershed algorithms and contrast preservation", *Proc. DGCI, LNCS*, Springer, Vol. 2886, pp. 62–71, 2003.

[24] L. Najman, M. Couprie, "Quasi-linear algorithm for the component tree", *Proc. SPIE Vision Geometry XII*, Vol. 5300, pp. 98-107, 2004.

[25] L. Najman, M. Couprie, G. Bertrand, "Watersheds, mosaics, and the emergence paradigm", to appear in *Discrete Applied Mathematics*, 2004.

[26] L. Najman, M. Schmitt, "Watershed of a continuous function", *Signal Processing*, Vol. 38, pp. 99-112, 1994.

[27] J.B.T.M. Roerdink, A. Meijster, "The watershed transform: definitions, algorithms and parallelization strategies", *Fundamenta Informaticae*, Vol. 41, pp. 187-228, 2000.

[28] A. Rosenfeld, "On connectivity properties of grayscale pictures", *Pattern Recognition*, Vol. 16, pp. 47-50, 1983.

[29] J. Serra, *Image Analysis and Mathematical Morphology, Vol. II: Theoretical Advances*, Academic Press, 1988.

[30] P. Salembier, A. Oliveras, L. Garrido, "Antiextensive connected operators for image and sequence processing", *IEEE Trans. on Image Processing*, Vol. 7, No. 4, pp. 555-570, 1998.

[31] R.E. Tarjan, "Disjoint sets" *Data Structures and Network Algorithms*, Chap. 2, pp. 23-31, SIAM, 1978.

[32] M. Thorup, "On RAM priority queues", *7th ACM-SIAM Symposium on Discrete Algorithms*, pp. 59-67, 1996.

[33] C. Vachier, *Extraction de caractéristiques, segmentation d'images et Morphologie Mathématique*, PhD Thesis, École des Mines, Paris, 1995.

[34] L. Vincent, P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 6, pp. 583-598, 1991.

[35] D. Wishart, "Mode analysis: a generalization of the nearest neighbor which reduces chaining effects", *Numerical Taxonomy*, A.J. Cole Ed, Academic Press, pp. 282-319, 1969.

Annex 1: Proofs

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

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

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

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

Proof: immediate from the definitions. □

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

Proof: Since p is not W-destructible, we have $q \neq p$. We know that p is adjacent to (at least) two distinct components c_1 and c_2 of $\overline{F}[k]$, with $k = F(p)$. Suppose

that p is adjacent to only one component of $\overline{F'}[k]$ (obviously, p is adjacent to at least one component of $\overline{F'}[k]$). This implies that $F(q) \geq k$, that $w < k$ and that q is adjacent to both c_1 and c_2, a contradiction with lemma 5.1 since $[F \setminus q \downarrow w]$ is a W-thinning of F. □

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

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

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

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

Proof of Theorem 5: Let $k = F(p)$, let T_1 denote the type of the point p for F, let q be a W-destructible point for F, let v be an integer such that $[F \setminus q \downarrow v]$ is a W-thinning of F, let F' denote $[F \setminus q \downarrow v]$ and let T_2 denote the type of the point p for F'.
1) Case $T_1 = M$. Since p is not W-destructible, we have $q \neq p$. If q is not adjacent to p then obviously $T_2 = T_1$. Suppose now that q is adjacent to p, and that $F'(q) = v < k$. If $F(q) = k$, then, since p is an M-point for F, we know that q is also an M-point for F, and thus q is not W-destructible

for F, a contradiction. If $F(q) > k$, then consider $[F \setminus q \downarrow k]$ and apply the same argument as above. In conclusion, we have either q not adjacent to p or $F'(q) \geq k$, thus $T_2 = T_1$.
2) Case $T_1 = P$. Since p is not W-destructible, we have $q \neq p$. If q is not adjacent to p then obviously $T_2 = T_1$. Suppose now that q is adjacent to p. If $F'(q) = v \geq k$ then $T_2 = T_1$, otherwise we see that p is W-destructible for F' with lowest value v, and that p has no strictly lower neighbor for $[F' \setminus p \downarrow v]$, thus p is either an $\tilde{M}$-point or a $\tilde{P}$-point, as shown in Fig. 5).
3) Case $T_1 = S$. See lemma 5.2.
4) Case $T_1 = \tilde{S}$. See lemma 5.3.
5) Case $T_1 = \tilde{P}$. Let w be the lowest value of p.
If $q = p$ and $F'(q) > w$ then $T_2 = T_1$.
If $q = p$ and $F'(q) = w$ then, since p is an inner point for F' and not an $\tilde{M}$-point for F, we know that p is adjacent to exactly one component of $\overline{F'}[w+1]$ which is not a minimum, thus $T_2 = P$.
We know that p is adjacent to exactly one component of $\overline{F}[h]$, for all h such that $v < h \leq k$, and that p is not adjacent to any component of $\overline{F}[w]$.
If q is not adjacent to p we see that the same remains true for F', thus $T_2 = T_1$.
Suppose now that q is adjacent to p and $q \neq p$. We see that p must be adjacent to at least one component of $\overline{F'}[k]$, thus T_2 is not an inner type (see examples of the three possibilities other than T_1 in Fig. 5).
6) Case $T_1 = \tilde{M}$. The arguments are the same as for case 5, except that T_2 can be M instead of P, and cannot be $\tilde{P}$ (see Lemma 5.4, and observe that p remains adjacent to a minimum).

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

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

Proof of Prop. 10: If $k_m = k_1$ at line 06, then clearly $[k_1, c_1]$ is under all the other elements of V and there is no highest fork (any two elements of V are linked). Otherwise, one gets easily convinced that:
- the component $[k_m, c_m]$ found at line 06 is indeed the LCA of a given pair of separated components in V, and
- no other pair of separated components in V can have a higher LCA. □

Lemma 12.1 *Let $F, F' \in \mathcal{F}$, let $p, q \in E$ and $v, w \in \mathbb{K}$ such that:*
i) $[F \setminus p \downarrow v]$ is not a W-thinning of F, and
ii) $F' = [F \setminus q \downarrow w]$ is a W-thinning of F, and

iii) $[F' \setminus p \downarrow v]$ *is a W-thinning of* F'.
Then p *and* q *are neighbors,* $F(q) \geq F(p)$, *and* $w \leq v$.

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

Lemma 12.2 *Let* $F \in \mathcal{F}$, *let* $p \in E$ *such that:*
i) p *is not an* $\tilde{M}$*-point for* F, *and*
ii) there exists a point q *and a value* w *such that* $[F \setminus q \downarrow w]$ *is a W-thinning of* F, *and* p *is an* $\tilde{M}$*-point for* $[F \setminus q \downarrow w]$.
Then, q *is an* $\tilde{M}$*-point for* F.

Proof: immediate from lemma 12.1. □

Proof of Theorem 12: Let G' be a W-thinning of G and suppose that G' has a minimum which is strictly larger than the corresponding minimum of G. Consider the sequence of point lowerings which leads from G to G', and let $G = F^0, F^1, \ldots, F^n = G'$ be the successive results of these operations. Let F^k be the first element in the sequence in which a point p is M-lowered. Thus $F^k = [F^{k-1} \setminus p \downarrow v]$ is the result of M-lowering the point p, in other words p is an $\tilde{M}$-point for F^{k-1}. Consider now the last F^i in the sequence $F^0 \ldots F^{k-2}$ such that p is not an $\tilde{M}$-point for F^i. If no such element exists, then we have a contradiction since there is no $\tilde{M}$-point for $F^0 = G$. Otherwise, since p is an $\tilde{M}$-point for F^{i+1} and not for F^i, from lemma 12.2 we deduce that the point q which has been lowered between F^i and F^{i+1} has indeed been M-lowered. This contradicts our definition of F^k. □

Proof of Prop 14:
a) From property 7 and theorem 8, it follows that the initial component tree of $\overline{F}$ remains a component tree for all the modified versions of $\overline{F}$ in this algorithm. We also see that the component mapping Ψ is updated in order to keep correct pointers from the vertices of the graph to the corresponding tree elements.
b) We see easily that ($K(p) = k$ and p in L_k) $\Leftrightarrow$ p is W-destructible for F with lowest value k.
c) Let us prove that in lines 07-16, there is no W-destructible point for F with a lowest value $k' < k$. It is true when $k = k_{\min}$. From lemma 12.1, we

know that a point cannot receive a lowest value v unless one of its neighbors is lowered down to a value $v' \leq v$. All the lowerings are done at line 10, by the statement $F(p) \leftarrow k$. Thus, the property remains true as k increases.
d) From a) and b), we deduce that at each step of the execution, F is a W-thinning of the input function.
e) From c), we deduce that at the end of the execution, F has no W-destructible point.
i) Follows from d) and e).
ii) For any given value of k, a point which is lowered at line 10 will not be lowered again in any step $k' > k$. Thus, each point is lowered at most once. Also, the total number of executions of lines 12-16 will not exceed m. Globally, the sum of the costs of all calls to the function **W-Destructible** is in $O(n + m)$. The calls to list management functions are in constant time. The total number of elements stored in the lists L_i cannot exceed $n + m$. □

Annex 2: Quasi-linear algorithm for the component tree

Let us first describe briefly the disjoint set problem, which consists in maintaining a collection S of disjoint subsets of a set E under the operation of union. Each set X in S is represented by a unique element of X, called the *canonical element*. Three operations allow the management of the collection (in the following x and y denote two distinct elements of E):
MakeSet(x): add the set $\{x\}$ to the collection S, provided that the element x does not already belongs to a set in S. The canonical element of $\{x\}$ is x.
Find(x): return the canonical element of the set in S which contains x.
Link(x, y): let X and Y be the two sets in S whose canonical elements are x and y respectively. Both sets are removed from S, their union $Z = X \cup Y$ is added to S and a canonical element for Z is selected and returned.

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

Procedure MakeSet (element x)
    par(x) $\leftarrow x$; rank(x) $\leftarrow 0$

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

Function Link (element x, y)
    If rank(x) > rank(y) **Then** exchange(x, y)
    If (rank(x) = rank(y)) **Then** rank(y) $\leftarrow$ rank(y) + 1
    par(x) $\leftarrow$ y
    Return y

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

Procedure BuildComponentTree
Input : (E, Γ) - graph; N = number of points in E
Input : F - map from E to $\mathbb{Z}$
Output : N_n - number of nodes (of the component tree) ($\leq N$)
Output : nodes - array $[0 \dots N-1]$ of node
Output : Ψ - map from E to $[0 \dots N-1]$ (component mapping)
Local : subtreeRoot - map from $[0 \dots N-1]$ to $[0 \dots N-1]$

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

5 Fusion graphs : merging properties and watershed

J. Cousty, G. Bertrand, M. Couprie and L. Najman. Fusion graphs : merging properties and watershed.
Soumis à *Computer Vision and Image Understanding*.

Fusion graphs: merging properties and watersheds

J. Cousty, G. Bertrand, M. Couprie and L. Najman

IGM, Unité Mixte de Recherche CNRS-UMLV-ESIEE UMR 8049
Laboratoire A2SI, Groupe ESIEE
Cité Descartes, BP99, 93162 Noisy-le-Grand Cedex France

Abstract

Region merging methods consist of improving an initial segmentation by merging some pairs of neighboring regions. In this paper, we consider a segmentation as a set of connected regions, separated by a frontier. If the frontier set cannot be reduced without merging some regions then we call it a watershed. In a general graph framework, merging two regions is not straightforward. We define four classes of graphs for which we prove, thanks to the notion of watershed, that some of the difficulties for defining merging procedures are avoided. Our main result is that one of these classes is the class of graphs in which any watershed is thin. None of the usual adjacency relations on $\mathbb{Z}^2$ and $\mathbb{Z}^3$ allows a satisfying definition of merging. We introduce the perfect fusion grid on $\mathbb{Z}^n$, a regular graph in which merging two neighboring regions can always be performed by removing from the frontier set all the points adjacent to both regions.

Key words: Graph theory, region merging, watershed, fusion graphs, adjacency relations, connectedness, image segmentation, image processing

This article is dedicated to the memory of Azriel Rosenfeld.

Introduction

Azriel Rosenfeld, by his seminal work, has formalized and explored a number of mathematical notions which are now considered as the most fundamental ones for image analysis. In particular, he pioneered the study of connectivity in both binary and grayscale digital pictures [13,14], through the use of adjacency

Email addresses: j.cousty@esiee.fr (J. Cousty), g.bertrand@esiee.fr (G. Bertrand), m.couprie@esiee.fr (M. Couprie), l.najman@esiee.fr (L. Najman).

relations (*i.e.,* graphs) defined on $\mathbb{Z}^2$ and $\mathbb{Z}^3$. In the important and difficult task of segmenting an image, connectivity often plays an essential role: in many cases, a segmentation can be viewed as a set of connected regions, separated by a background which constitutes the frontiers between regions. A popular approach to image segmentation, called region merging [15,12], consists of progressively merging pairs of regions until a certain criterion is satisfied. The criterion which is used to identify the next pair of regions which will merge, as well as the stopping criterion are specific to each particular method.

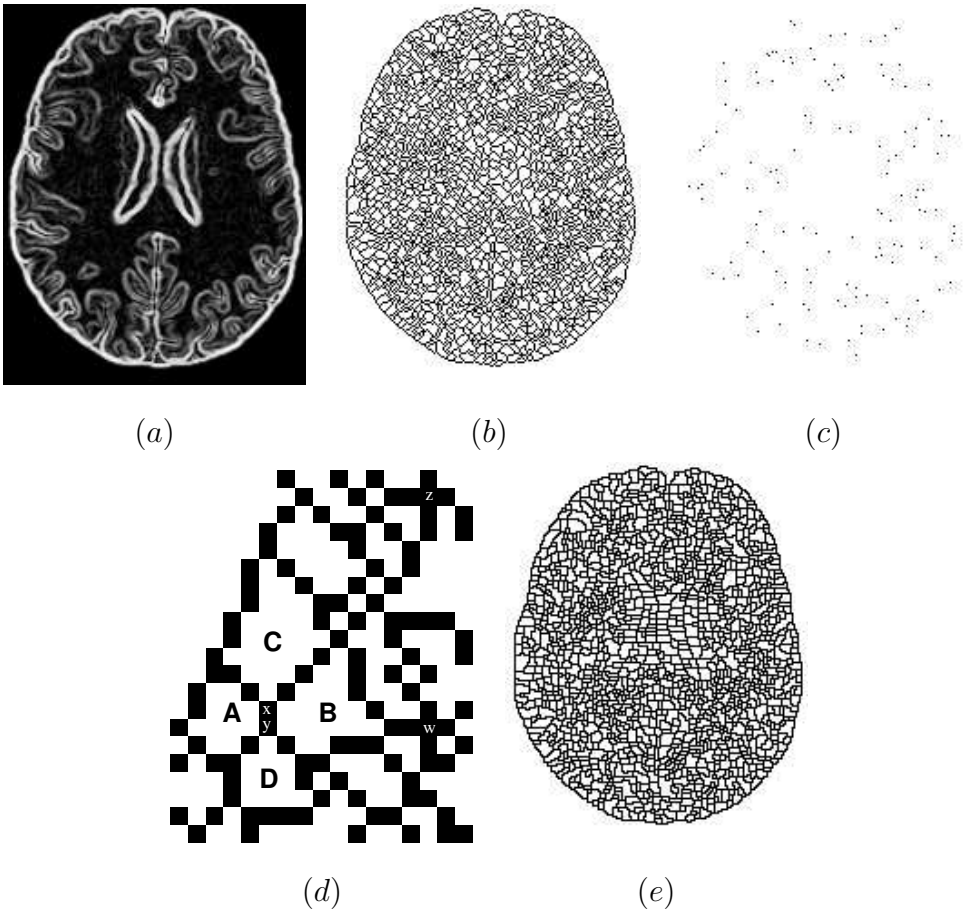


Fig. 1. (a): Original image (cross-section of a brain, after applying a gradient operator). (b): Watershed of (a) with the 4-adjacency (in black). (c): Interior points for the previous image (in black). (d): A zoom on a part of (b). The points z and w are interior points. (e): Watershed of (a) with the 8-adjacency (in black). There are no interior points.

Given a grayscale image, how is it possible to obtain an initial set of regions for a region merging process? The watershed transform [4,10] is a powerful tool for solving this problem. Let us consider a 2D grayscale image as a topographical relief, where the dark pixels correspond to basins and valleys, whereas bright pixels correspond to hills and crests. Suppose that we are interested in segmenting "dark" regions. Intuitively, the watersheds of the image are constituted by the crests which separate the basins corresponding to regional

minima (see Fig. 1a,b). Due to noise and texture, real-world images often have a huge number of regional minima, hence the "mosaic" aspect of Fig. 1b. In [5,3,6,11], the authors developped a framework based on graph theory, in which some important properties of grayscale watersheds are proved, and efficient algorithms to compute them are proposed. In the case of a graph (*e.g.,* an adjacency graph defined on a subset of $\mathbb{Z}^2$), a watershed may be thought of as a "separating set" of vertices which cannot be reduced without merging some connected components of its complementary set.

A first question arises when dealing with watersheds on a graph. Given a subset E of $\mathbb{Z}^2$ and the graph (E, Γ_1) which corresponds to the usual 4-adjacency relation, we observe that a watershed may contain some "interior points", *i.e.,* points which are not adjacent to any point outside the watershed (see Fig. 1c,d). We can say that a watershed on Γ_1 is not necessarily thin. On the other hand, such interior points do not seem to appear in any watershed on Γ_2, which corresponds to the 8-adjacency. Are the watersheds on Γ_2 always thin? We will prove that it is indeed true. More interestingly, we provide in this paper a framework to study the property of thinness of watersheds in any kind of graph, and we identify the class of graphs in which any watershed is necessarily thin. This result is one of the main theorems of the article (Th. 33).

Let us now turn back to the region merging problem. What happens if we want to merge a couple of neighboring regions A and B, and if each pixel adjacent to these two regions is also adjacent to a third one, which is not wanted in the merging? Fig. 1d illustrates such a situation, where x is adjacent to regions A, B, C and y to A, B, D. This problem has been identified in particular by T. Pavlidis (see [12], section 5.6: "When three regions meet"), and has been dealt with in some practical ways, but until now a systematic study of properties related to merging in graphs has not been done. A major contribution of this article is the definition and the study of four classes of graphs, with respect to the possibility of "getting stuck" in a merging process (Sec. 3, Sec. 4). In particular, we say that a graph is a *fusion graph* if any region A in this graph can always be merged with another region B, without problems with other regions. The most striking outcome of this study is that the class of fusion graphs is precisely the class of graphs in which any watershed is thin (Th. 33). We also provide some local characterizations for two of these four classes of graphs, and prove that the two other ones cannot be locally characterized (Sec. 5).

Using this framework, we analyze the status of the graphs which are the most widely used for image analysis, namely the graphs corresponding to the 4- and the 8-adjacency in $\mathbb{Z}^2$ and to the 6- and the 26-adjacency in $\mathbb{Z}^3$ (Sec. 6). In one of the classes of graphs introduced in Sec. 4, that we call the class of *perfect fusion graphs*, any pair of neighboring regions A, B can always be merged, without problems with other regions, by removing all the pixels which are

adjacent to both A and B. We show that none of these classical graphs is a perfect fusion graph. Last, but not least, in Sec. 7 we introduce a graph on $\mathbb{Z}^n$ (for any n) that we call the perfect fusion grid, which is indeed a perfect fusion graph, and which is "between" the direct adjacency graph (which generalizes the 4-adjacency to $\mathbb{Z}^n$) and the indirect adjacency graph (which generalizes the 8-adjacency). Furthermore, in [7], we prove that this n-dimensional grid is the unique grid (up to a translation) that possesses those two properties.

1 Basic notions

Let E be a set, we write $X \subseteq E$ if X is a subset of E, we write $X \subset E$ if X is a proper subset of E, *i.e.,* if X is a subset of E and $X \neq E$. We denote by $\overline{X}$ the complementary set of X in E, *i.e.,* $\overline{X} = E \setminus X$.

Let E be a finite set, we denote by $|E|$ the number of elements of E. We denote by 2^E the set composed of all the subsets of E.

We define a graph as a pair (E, Γ) where E is a finite set and Γ is a binary relation on E (*i.e.,* $\Gamma \subseteq E \times E$), which is reflexive (for all x in E, $(x, x) \in \Gamma$) and symmetric (for all x, y in E, $(y, x) \in \Gamma$ whenever $(x, y) \in \Gamma$). Each element of E is called a *vertex* or a *point*. We will also denote by Γ the map from E to 2^E such that, for all $x \in E$, $\Gamma(x) = \{y \in E \mid (x, y) \in \Gamma\}$. If $y \in \Gamma(x)$, we say that y is *adjacent to* x. We define also the map Γ^* such that for all $x \in E$, $\Gamma^*(x) = \Gamma(x) \setminus \{x\}$. Let $X \subseteq E$, we define $\Gamma(X) = \cup_{x \in X} \Gamma(x)$, and $\Gamma^*(X) = \Gamma(X) \setminus X$. If $y \in \Gamma(X)$, we say that y *is adjacent to* X. If $X, Y \subseteq E$ and $\Gamma(X) \cap Y \neq \emptyset$, we say that Y *is adjacent to* X (or that X is adjacent to Y, since Γ is symmetric). Let $G = (E, \Gamma)$ be a graph and let $X \subseteq E$, we define the *subgraph of* G *induced by* X as the graph $G_X = (X, \Gamma \cap [X \times X])$. In this case, we also say that G_X *is a subgraph of* G. Let $G = (E, \Gamma)$ and $G' = (E', \Gamma')$ be two graphs, we say that G *and* G' *are isomorphic* if there exists a bijection f from E to E' such that, for all $x, y \in E$, y belongs to $\Gamma(x)$ if and only if $f(y)$ belongs to $\Gamma'(f(x))$.

Let (E, Γ) be a graph, let $X \subseteq E$, a *path in* X is a sequence $\pi = \langle x_0, ..., x_l \rangle$ such that $x_i \in X$, $i \in [0, l]$, and $x_i \in \Gamma(x_{i-1})$, $i \in [1, \ldots, l]$. We also say that π is a *path from* x_0 *to* x_l *in* X. Let x, $y \in X$. We say that x and y are *linked for* X if there exists a path from x to y in X. We say that X *is connected* if any x and y in X are linked for X.

Let $Y \subseteq X$. We say that Y *is a connected component of* X, or simply *a component of* X, if Y is connected and if Y is maximal for this property, *i.e.,* if $Z = Y$ whenever $Y \subseteq Z \subseteq X$ and Z connected.
We denote by $\mathcal{C}(X)$ the set of all the connected components of X. Let $S \subseteq E$,

we denote by $\mathcal{C}(X|S)$ the subset of $\mathcal{C}(X)$ composed of the components of X which are adjacent to S.

Notice that the empty set is connected, and that if X is non-empty, then the empty set is not a connected component of X. Notice also that, if Y is a connected component of a set X, then Y is not adjacent to $X \setminus Y$.

Let us consider a subset X of E, and two non-empty subsets A, B of X such that $A \cup B = X$. We can easily see that, if X is connected, then A and B must be adjacent to each other. On the other hand, if X is not connected, then we have two points x and y in X which are not linked for X. Considering the set A of all the points z in X such that x and z are linked for X and considering the set $B = X \setminus A$, we see that X can be partitioned into two non-empty subsets which are not adjacent to each other. These observations lead to the following property which characterizes connected sets (without the need of considering paths).

Property 1. *Let (E, Γ) be a graph, let $X \subseteq E$. The set X is connected if and only if, for any two distinct non-empty subsets A, B of X such that $A \cup B = X$, the subset A is adjacent to B.*

From Prop. 1 we can immediately deduce the following corollary.

Corollary 2. *Let (E, Γ) be a graph, let X be a non-empty subset of E. If E is connected and if $X \neq E$, then $\Gamma^*(X) \neq \emptyset$.*

In this paper, we study in particular some thinness properties of watersheds in graphs. The notions of thinness and interior are closely related.

Definition 3. *Let (E, Γ) be a graph. Let $X \subseteq E$, the* interior of X *is the set $int(X) = \{x \in X \mid \Gamma(x) \subseteq X\}$. We say that the set X* is thin *if $int(X) = \emptyset$.*

Property 4. *Let (E, Γ) be a graph, let $X \subseteq E$ such that $int(X) \neq \emptyset$, let A be a non-empty subset of $int(X)$. We have: $\mathcal{C}(\overline{X \setminus A}) = \mathcal{C}(\overline{X}) \cup \mathcal{C}(A)$. Furthermore, if A is connected, then A is a connected component of $\overline{X \setminus A}$; more precisely we have $\mathcal{C}(\overline{X \setminus A}) = \mathcal{C}(\overline{X}) \cup \{A\}$.*

The proof of Prop. 4 is elementary and thus omitted. To conclude this section, we recall the definition of line graphs. This class of graphs allows to make a strong link between the framework developped in this paper and the approaches of watershed and region merging based on edges rather than vertices.

Definition 5. *Let (E, Γ) be a graph. The* line graph of (E, Γ) *is the graph (E', Γ') such that $E' = \Gamma$ and (u, v) belongs to Γ' whenever $u \in \Gamma$, $v \in \Gamma$, and u, v share a vertex of E.*

We say that a graph (E', Γ') is a line graph *if there exists a graph* (E, Γ) *such that* (E', Γ') *is isomorphic to the line graph of* (E, Γ).

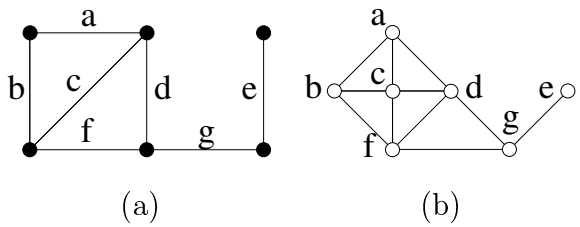


Fig. 2. A graph (a) and its line graph (b).

In Fig. 2, we show a graph and its line graph. All graphs are not line graphs, in other words, there exist some graphs which are not the line graphs of any graph. The following theorem allows to characterize line graphs.

Theorem 6 ([1]). *A graph G is a line graph if and only if none of the graphs of Fig. 3 is a subgraph of G.*

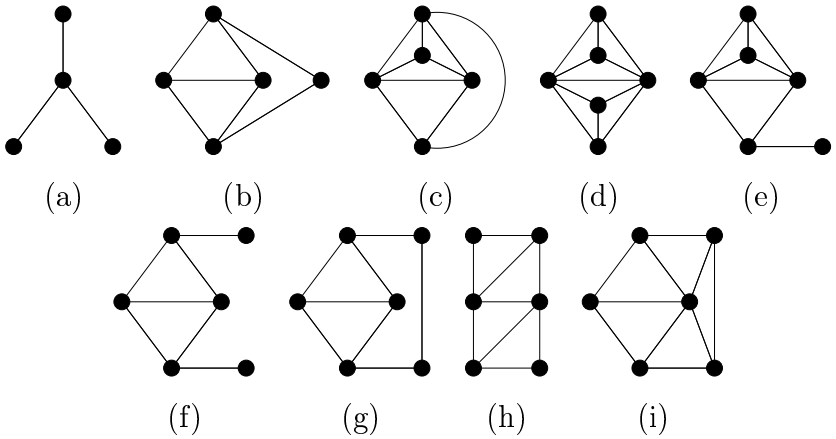


Fig. 3. Graphs for a characterization of line graphs (Th. 6).

As an illustration, we can check that the line graph depicted in Fig. 2b does not contain any graph of Fig. 3 as a subgraph. For example, the subgraph induced by the set $\{d, e, f, g\}$ of the graph shown in Fig. 2b is not the same as the graph of Fig. 3a since it contains one more edge.

2 Watersheds

Informally, in a graph, a watershed may be thought of as a "separating set" of vertices which cannot be reduced without merging some components of its complementary set (see Fig. 4d). We first give formal definitions of these concepts (see [3,5]) and related ones, then we derive some properties which will be used in the sequel.

Important remark. *From now, when speaking about a graph (E, Γ), we will assume for simplicity that E is non-empty and connected.*
Notice that, nevertheless, the subsequent definitions and properties may be easily extended to non-connected graphs.

Definition 7. *Let (E, Γ) be a graph. Let $X \subseteq E$, and let $p \in X$.*
We say that p is a border point (for X) *if p is adjacent to $\overline{X}$.*
We say that p is an inner point (for X) *if p is not a border point for X,* i.e., *if $p \in int(X)$.*
We say that p is W-simple (for X) *if p is adjacent to exactly one connected component of $\overline{X}$.*
We say that p is separating (for X) *if p is adjacent to at least two connected components of $\overline{X}$.*
We say that p is a multiple point (for X) *if p is adjacent to at least three connected components of $\overline{X}$.*

In this definition and the following ones, the prefix "W-" stands for watershed. In Fig. 4a, x is both a border point and a W-simple point for the set X constituted by the black vertices, and y is an inner point. In Fig. 5b, z is a border point and a separating point, and w is a border point, a separating point and a multiple point.

Definition 8. *Let (E, Γ) be a graph. Let $X \subseteq E$, and let $S \subseteq X$.*
We say that S is W-simple (for X) *if there exists $A \in \mathcal{C}(\overline{X})$ such that $A \cup S$ is connected and $\mathcal{C}(\overline{X}|S) = \{A\}$.*

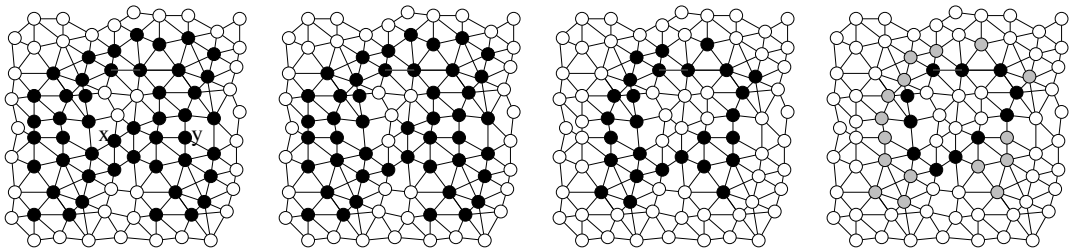
Obviously, a point p is W-simple if and only if the set $\{p\}$ is W-simple. Notice that, in the above definition, S is not necessarily connected. The following property may be proved easily.

Property 9. *Let (E, Γ) be a graph. Let $X \subseteq E$, and let $S \subseteq X$.*
The set S is W-simple (for X) if and only if there exists $A \in \mathcal{C}(\overline{X})$ such that $\mathcal{C}(\overline{X} \cup S) = [\mathcal{C}(\overline{X}) \setminus \{A\}] \cup \{A \cup S\}$.

We are now ready to define the notion of watershed which is central to this section.

Definition 10. *Let $G = (E, \Gamma)$ be a graph. Let $X \subseteq E$, let $Y \subseteq X$.*
We say that Y is a W-thinning of X, *written $X \overset{W}{\searrow} Y$, if*
i) $Y = X$ or if
ii) there exists a set $Z \subseteq X$ which is a W-thinning of X and a point $p \in Z$ which is W-simple for Z, such that $Y = Z \setminus \{p\}$.
A set $Y \subseteq X$ is a watershed (in G) *if $Y \overset{W}{\searrow} Z$ implies $Z = Y$.*
A subset Y of X is a watershed of X *if Y is a W-thinning of X and if Y is a watershed.*
A watershed Y is non-trivial *if $Y \neq \emptyset$ and $Y \neq E$.*

It can be seen that we can obtain a W-thinning of X by iteratively removing W-simple points from X, and that Y is a watershed of X if Y is a W-thinning of X which contains no W-simple point. Fig. 4 shows a set X and some W-thinnings of X, the last one being a watershed of X. Notice that different watersheds may exist for a same set X. It can be also seen that a watershed X is non-trivial if and only if $|\mathcal{C}(\overline{X})| \geq 2$.



(a) (b) (c) (d)

Fig. 4. Illustration of W-thinning and watershed. (a): A graph (E, Γ) and a subset X (black points) of E. The point x is a border point which is W-simple, and y is an inner point. (b): The set $Y = X \setminus \{x\}$ (black points) is a W-thinning of X. (c): The set Z (black points) is a W-thinning of both X and Y. The sets Y and Z are not watersheds: some W-simple points exist in both sets. (d): A watershed of X (black points), which is also a watershed of Y and of Z. The set of gray points will be used to illustrate the notion of annexation (Def. 16).

The following definition and theorem are borrowed from [3] and will play an important role in some subsequent proofs.

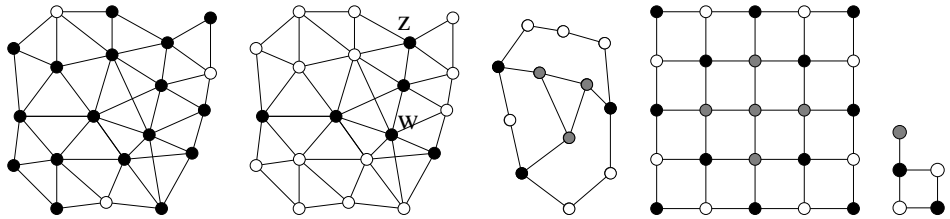
Definition 11. *Let (E, Γ) be a graph. Let X, Y be subsets of E. We say that Y* is an extension of *X if $X \subseteq Y$ and if each connected component of Y contains exactly one connected component of X.*

Theorem 12 ([3])**.** *Let X and Y be subsets of E. The subset Y is a W-thinning of X if and only if $\overline{Y}$ is an extension of $\overline{X}$.*

We can see that if a subset S of X is W-simple for X, then $\overline{X \setminus S}$ is an extension of $\overline{X}$. From this observation and Th. 12, we immediatly deduce the following property.

Corollary 13. *Let $X \subseteq E$ and $S \subseteq X$. If the subset S is W-simple for X, then $X \setminus S$ is a W-thinning of X.*

A watershed is a set which contains no W-simple point, but some of the examples given below show that such a set is not always thin (in the sense of Def. 3). Fig. 4d and Fig. 5b are two examples of watersheds which are thin: in both cases, the set of black points has no W-simple point and no inner point. Fig. 5c,d show two examples of non-thin watersheds. Let us study what happens if we remove from a non-thin watershed X, a connected component



(a) (b) (c) (d) (e)

Fig. 5. Illustration of thin and non-thin watersheds. (a): A graph (E, Γ) and a subset X (black points) of E. (b): A subset Y (black points) of E which is a thin watershed; it is a watershed of the set X shown in (a). The border points z and w are both separating for Y, only w is a multiple point. (c, d, e): The subset X represented by black and gray points is a watershed which is not thin: $\mathrm{int}(X)$ is depicted by the gray points.

of $\mathrm{int}(X)$.

Property 14. *Let (E, Γ) be a graph, let $X \subseteq E$ be a watershed. Let A be a connected component of int(X). Then, $X \setminus A$ is a watershed.*

Proof: The cases where $|\mathcal{C}(\overline{X})| < 2$ or $\mathrm{int}(X) = \emptyset$ are trivial: if $|\mathcal{C}(\overline{X})| = 0$ then $E = X = \mathrm{int}(X) = A$ and $X \setminus A = \emptyset$; if $|\mathcal{C}(\overline{X})| = 1$ then it may be seen that X must be empty since E is connected, thus $X \setminus A = \emptyset$; and if $\mathrm{int}(X) = \emptyset$ then $A = \emptyset$, thus $X \setminus A = X$. Suppose from now that $|\mathcal{C}(\overline{X})| \geq 2$ and $\mathrm{int}(X) \neq \emptyset$. From Prop. 4, $A \in \mathcal{C}(\overline{X \setminus A})$. Let x be a point of $X \setminus A$, we have to prove that x cannot be W-simple for $X \setminus A$. If $x \notin \Gamma^*(A)$, we can easily see that the point x cannot be W-simple for $X \setminus A$, otherwise it would also be W-simple for X. Suppose now that $x \in \Gamma^*(A)$. The point x cannot belong to $\mathrm{int}(X)$ otherwise A would not be a connected component of $\mathrm{int}(X)$. Thus x must be adjacent to a component B of $\mathcal{C}(\overline{X})$, which is also a component of $\mathcal{C}(\overline{X \setminus A})$ (Prop. 4): hence, x is adjacent to both A and B, with $A \neq B$, and is not W-simple for $X \setminus A$. □

The following corrolary follows straightforwardly.

Corollary 15. *Let (E, Γ) be a graph, let $X \subseteq E$. The set $X \setminus$ int(X) is a watershed.*

Let (E, Γ) be a graph. Let $X \subset E$, let $A \in \mathcal{C}(\overline{X})$. Let us consider the family $\mathcal{W}_A$ of all the sets which are W-simple for X and adjacent to A. It may be easily seen that the family $\mathcal{W}_A$ is closed by union, *i.e.,* that $S \cup T$ belongs to $\mathcal{W}_A$ whenever $S \in \mathcal{W}_A$ and $T \in \mathcal{W}_A$. From this observation, we deduce that there exists a unique element of $\mathcal{W}_A$ which is maximal for the inclusion, and this element is the union of all the elements of the family.

Definition 16. *Let (E, Γ) be a graph. Let $X \subset E$, let $A \in \mathcal{C}(\overline{X})$. We define the* annexation of A in X, *denoted by ann(A, X), as the union of all the sets*

which are W-simple for X and adjacent to A. When no confusion may occur, we write $ann(A) = ann(A, X)$.

In Fig. 4c, let A be the (white) component of $\overline{Z}$ which "surrounds" the (black) set Z. The set $\mathrm{ann}(A, Z)$ is depicted in light gray in Fig. 4d.

We have seen that, for any S which is W-simple for X and adjacent to A, the set $\overline{X} \cup S$ is an extension of $\overline{X}$. In particular, the set $\overline{X} \cup \mathrm{ann}(A)$ is an extension of $\overline{X}$.

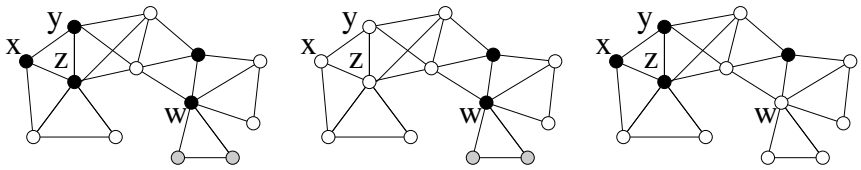
The following properties illustrate the notion of annexation, which will serve us to prove some of the main results of this paper.

Property 17. *Let (E, Γ) be a graph, let $X \subset E$ such that $|\mathcal{C}(\overline{X})| \geq 2$. For any $A \in \mathcal{C}(\overline{X})$, there exists $B \in [\mathcal{C}(\overline{X}) \setminus \{A\}]$ such that $\Gamma^*(A \cup ann(A)) \cap \Gamma^*(B) \neq \emptyset$.*

The proof can be found in the annex. We leave the proof of the following property to the interested reader.

Property 18. *Let (E, Γ) be a graph, let $X \subset E$, let $A \in \mathcal{C}(\overline{X})$. The set $A \cup ann(A, X)$ is equal to the connected component of $int(X \cup A)$ which contains A.*

3 Merging



(a) (b) (c)

Fig. 6. Illustration of merging. (a): A graph (E, Γ) and a subset X of E (black points). (b): The black points represent $X \setminus S$ with $S = \{x, y, z\}$. (c): The black points represent $X \setminus S'$ with $S' = \{w\}$.

Consider the graph (E, Γ) depicted in Fig. 6a, where a subset X of E (black vertices) separates its complementary set $\overline{X}$ into four connected components. If we replace the set X by, for instance, the set $X \setminus S$ where $S = \{x, y, z\}$, we obtain a set which separates its complementary set into three components, see Fig. 6b: we can also say that we "merged two components of $\overline{X}$ through S". This operation may be seen as an "elementary merging" in the sense that only two components of $\overline{X}$ were merged. On the opposite, replacing the set X by the set $X \setminus S'$ where $S' = \{w\}$, see Fig. 6c, would merge three components of $\overline{X}$. We also see that the component of $\overline{X}$ which is below w (in light gray) cannot be merged by an "elementary merging" since any attempt to merge it

must involve the point w, and thus also the three components of $\overline{X}$ adjacent to this point. In this section, we introduce definitions and basic properties related to such merging operations in graphs.

Definition 19. *Let (E, Γ) be a graph and $X \subset E$. Let $p \in X$, let $S \subseteq X$. We say that p* is F-simple (for X) *if p is adjacent to exactly two connected components of $\overline{X}$.*
We say that S is F-simple (for X) *if S is adjacent to exactly two components $A, B \in \mathcal{C}(\overline{X})$ such that $A \cup B \cup S$ is connected.*

In this definition, the prefix "F-" stands for fusion. Observe that the point p is F-simple if and only if the set $\{p\}$ is F-simple. For example, in Fig. 6a, the point z is F-simple while x, y, w are not. Also, the sets $\{z\}$, $\{x, y\}$, $\{x, z\}$, $\{y, z\}$, $\{x, y, z\}$ are F-simple, but the sets $\{x\}$, $\{y\}$ and $\{w\}$ are not.

Notice also that the set S is not necessarily connected. Furthermore, any connected component T of S must be adjacent to either A or B, or both, and cannot be adjacent to any other element of $\mathcal{C}(\overline{X})$. Thus we have the following property.

Property 20. *Let (E, Γ) be a graph, let $X \subset E$, let $S \subseteq X$ such that S is F-simple for X, and let $T \subseteq S$. If $T \in \mathcal{C}(S)$, then T is either W-simple or F-simple for X.*

Definition 21. *Let (E, Γ) be a graph and $X \subset E$. Let A and $B \in \mathcal{C}(\overline{X})$, with $A \neq B$. We say that A* and B can be merged (for X) *if there exists $S \subseteq X$ such that S is F-simple for X and adjacent to both A and B. In this case, we also say that A* and B can be merged through S (for X).
We say that A can be merged (for X) *if there exists $B \in \mathcal{C}(\overline{X})$ such that A and B can be merged for X.*

For example, in Fig. 6a, the component of $\overline{X}$ in light gray cannot be merged, but each of the three white components can be merged for X.

Property 22. *Let (E, Γ) be a graph, let $X \subset E$, let $A, B \in \mathcal{C}(\overline{X})$, $A \neq B$, and let $S \subseteq X$. The components A and B can be merged through S if and only if $A \cup B \cup S$ is a connected component of $\overline{X \setminus S}$. More precisely, A and B can be merged through S if and only if $\mathcal{C}(\overline{X \setminus S}) = [\mathcal{C}(\overline{X}) \setminus \{A, B\}] \cup \{A \cup B \cup S\}$.*

Property 23. *Let (E, Γ) be a graph, let $X \subset E$, let $A, B \in \mathcal{C}(\overline{X})$ with $A \neq B$. The components A and B can be merged for X if and only if there exists $S \subseteq X$ such that S is connected and adjacent to only A and B.*

The proof of Prop. 22 can be found in the annex, and the proof of Prop. 23 is elementary. The following property will be useful for establishing one of the main results of this article, namely Th. 33.

Property 24. *Let (E, Γ) be a graph, let $X \subseteq E$, and let $A \in \mathcal{C}(\overline{X})$. The three following statements are equivalent:*
i) A can be merged for X;
ii) $[A \cup ann(A, X)]$ can be merged for $[X \setminus ann(A, X)]$;
iii) there exists an extension $\overline{Y}$ of $\overline{X}$ and there exists a vertex $x \in \Gamma^*(A')$ which is F-simple, where A' is the connected component of $\overline{Y}$ which contains A.

Proof:
• $[i \Rightarrow ii]$ From i), we know that there exists $B \in \mathcal{C}(\overline{X})$ and $S \subseteq X$ such that S is F-simple for X and adjacent to both A and B. Let $A' = A \cup \text{ann}(A, X)$, and let $Y = X \setminus \text{ann}(A, X)$. From Def. 16 and the observation which follows this definition, $\overline{Y}$ is an extension of $\overline{X}$ and $\mathcal{C}(\overline{Y}) = [\mathcal{C}(\overline{X}) \setminus \{A\}] \cup \{A'\}$. Let $S' = S \cap \overline{A'}$, thus $S' \subseteq Y$. We have: $A' \cup S' \cup B = A \cup S \cup B \cup A'$. We know that A' is connected, that $A \cup S \cup B$ is connected and that $A \subseteq A'$, thus $A \cup S \cup B \cup A'$ is connected, hence so is $A' \cup S' \cup B$. This implies that S' is adjacent to both A' and B. Since the only components of $\overline{X}$ adjacent to S are A and B and since $S' \subseteq S$, we deduce that the only components of $\overline{Y}$ adjacent to S' are precisely A' and B, thus S' is F-simple for Y, hence ii).

• $[ii \Rightarrow iii]$ Let $A' = A \cup \text{ann}(A, X)$, let $Y = X \setminus \text{ann}(A, X)$. We have seen that $\overline{Y}$ is an extension of $\overline{X}$ and that A' is the element of $\mathcal{C}(\overline{Y})$ which contains A. From ii), we know that there exists $B \in \mathcal{C}(\overline{Y})$ and $S \subseteq Y$ such that S is F-simple for Y and adjacent to both A' and B. There must exist some points in S which are adjacent to A', let x be any such point. The point x cannot be W-simple for Y, otherwise the set $\text{ann}(A, X) \cup \{x\}$ would be W-simple for X and adjacent to A, a contradiction with the definition of $\text{ann}(A, X)$. Furthermore, since S is F-simple it cannot contain any multiple point, thus x is F-simple for Y.

• $[iii \Rightarrow i]$ Suppose that x is a point of $\Gamma^*(A')$ which is F-simple. Then, x is adjacent to A' and to B', with $B' \in \mathcal{C}(\overline{Y})$, $B' \neq A'$, and $A' \cup B' \cup \{x\}$ is connected. Let B be the component of $\mathcal{C}(\overline{X})$ such that $B \subseteq B'$. Let us consider $S = [A' \setminus A] \cup [B' \setminus B] \cup \{x\}$. It can be easily seen that $S \subseteq X$ and that S is adjacent to both A and B. Since $\overline{Y}$ is an extension of $\overline{X}$ we know that A' (resp. B') cannot be adjacent to any other connected component of $\overline{X}$ than A (resp. B). Also, x cannot be adjacent to any other connected component of $\overline{X}$ than A and B, otherwise it could not be F-simple for Y. Furthermore, we have $A \cup B \cup S = A' \cup B' \cup \{x\}$, thus $A \cup B \cup S$ is connected. Thus, since S is adjacent to solely A and B, S is F-simple for X, and A can be merged for X. □

From Def. 10 and Th. 12, any extension of a watershed X is equal to X. Thus, the following corollary is an immediate consequence of Prop. 24.

Corollary 25. *Let (E, Γ) be a graph, let $X \subset E$ be a watershed and let*

$A \in \mathcal{C}(\overline{X})$. The subset A can be merged for X if and only if there exists a vertex $x \in \Gamma^*(A)$ which is F-simple for X.

4 Fusion graphs

Region merging [12,15] is a popular approach to image segmentation. Starting with an initial partition of the image pixels into connected regions, which can in some cases be separated by some boundary pixels, the basic idea consists of progressively merging pairs of regions until a certain criterion is satisfied. The criterion which is used to identify the next pair of regions which will merge, as well as the stopping criterion are specific to each particular method. Certain methods do not use graph vertices in order to separate regions, nevertheless even these methods fall in the scope of this study through the use of line graphs [9].

The preceding section and the present one constitute a theoretical basis for the study of such methods. The problems encountered by certain region merging methods (see [12], section 5.6: "When three regions meet") can be avoided by using exclusively the notion of merging introduced in the previous section. In the sequel, we investigate several classes of graphs with respect to the possibility of "getting stuck" in a merging process. The most striking result of this section is a theorem which states the equivalence between one of these classes and the class of graphs in which any watershed is thin.

We begin with the definition of four classes of graphs.

Definition 26. *We say that a graph (E, Γ)* is a weak fusion graph *if for any $X \subset E$ such that $|\mathcal{C}(\overline{X})| \geq 2$, there exist A, $B \in \mathcal{C}(\overline{X})$ which can be merged.*

Definition 27. *We say that a graph (E, Γ)* is a fusion graph *if for any $X \subset E$ such that $|\mathcal{C}(\overline{X})| \geq 2$, each $A \in \mathcal{C}(\overline{X})$ can be merged for X.*

Let $X \subset E$, and let A, $B \in \mathcal{C}(\overline{X})$. We set $\Gamma^*(A, B) = \Gamma^*(A) \cap \Gamma^*(B)$. We say that *$A$ and B are neighbors* if $A \neq B$ and $\Gamma^*(A, B) \neq \emptyset$.

Definition 28. *We say that the graph (E, Γ)* is a strong fusion graph *if, for any $X \subset E$, any A and $B \in \mathcal{C}(\overline{X})$ which are neighbors can be merged.*

Definition 29. *We say that the graph (E, Γ)* is a perfect fusion graph *if, for any $X \subset E$, any A and $B \in \mathcal{C}(\overline{X})$ which are neighbors can be merged through $\Gamma^*(A, B)$.*

Basic examples and counter-examples of weak fusion, fusion, strong fusion and perfect fusion graphs are given in Fig. 7.

These classes are linked by inclusion relations. The following property clarifies these links, and also position our four classes of graphs with respect to general graphs and line graphs. We denote by $\mathcal{G}$ (resp. $\mathcal{G}_L$, $\mathcal{G}_P$, $\mathcal{G}_S$, $\mathcal{G}_F$, and $\mathcal{G}_W$) the set of all graphs (resp. line graphs, perfect fusion graphs, strong fusion graphs, fusion graphs, and weak fusion graphs).

Property 30. *Any line graph is a perfect fusion graph,*
any perfect fusion graph is a strong fusion graph,
any strong fusion graph is a fusion graph,
any fusion graph is a weak fusion graph.
More precisely, we have the following strict inclusion relations:
$\mathcal{G}_L \subset \mathcal{G}_P \subset \mathcal{G}_S \subset \mathcal{G}_F \subset \mathcal{G}_W \subset \mathcal{G}$.

Proof: We prove in the annex (Lem. 58) that any strong fusion graph is a fusion graph. The other inclusions may be proved easily; let us prove that these inclusions are strict. It may be checked from the definitions that the graphs (g), (w), (f) and (s) in Fig. 7 are indeed counter-examples for the corresponding class equalities. It may also be checked that the graph (p) is a perfect fusion graph, while it is not a line graph, a consequence of Th. 6. □

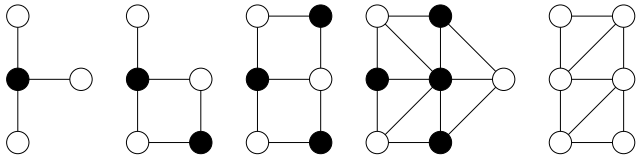


Fig. 7. Examples and counter-examples for different classes of graphs. (g): A graph which is not a weak fusion graph, (w): a weak fusion graph which is not a fusion graph, (f): a fusion graph which is not a strong fusion graph, (s): a strong fusion graph which is not a perfect fusion graph, and (p): a perfect fusion graph which is not a line graph. In the graphs (g, w, f, s), the black vertices constitute a set X which serves to prove that the graph does not belong to the pre-cited class.

The following property is a consequence of Def. 27, Cor. 25 and Prop. 24.

Property 31. *The graph $G = (E, \Gamma)$ is a fusion graph if and only if, for any non-trivial watershed Y in G and for any $A \in \mathcal{C}(\overline{Y})$, there exists $x \in \Gamma^*(A)$ which is F-simple.*

Proof: Let (E, Γ) be a fusion graph, let Y be a non-trivial watershed (thus $|\mathcal{C}(\overline{Y})| \geq 2$), and let $A \in \mathcal{C}(\overline{Y})$. Since (E, Γ) is a fusion graph, we know that A can be merged for Y, thus by Cor. 25, there exists $x \in \Gamma^*(A)$ which is F-simple.

Suppose now that for any non-trivial watershed $Y \subset E$ and for any $A' \in \mathcal{C}(\overline{Y})$, there exists $x \in \Gamma^*(A')$ which is F-simple. Let $X \subset E$ such that $|\mathcal{C}(\overline{X})| \geq 2$, let $A \in \mathcal{C}(\overline{X})$. Let Y be a watershed of X, and let $A' \in \mathcal{C}(\overline{Y})$ such that $A \subseteq A'$. By hypothesis, there exists $x \in \Gamma^*(A')$ which is F-simple for A'. Furthermore,

by Th. 12 we know that Y is an extension of X, thus by Prop. 24, A can be merged for X. $\square$

From Prop. 31, we deduce Prop. 32 which will be used in the proof of Th. 41.

Property 32. *Let $G = (E, \Gamma)$ be a graph. If G is not a fusion graph, then there exist $X \subset E$ and $x \in X$ such that x is a multiple point for X.*

Proof: If G is not a fusion graph, then by Prop. 31, there exists $Y \subset E$ such that $|\mathcal{C}(\overline{Y})| \geq 2$, there exists a watershed X of Y, there exists $A \in \mathcal{C}(\overline{X})$ such that any $x \in \Gamma^*(A)$ is not F-simple. For any such x, since $x \in \Gamma^*(A)$, x is not an inner point; and since X is a watershed, x is not W-simple; thus x must be a multiple point. Furthermore, since $|\mathcal{C}(\overline{Y})| \geq 2$ and thus $|\mathcal{C}(\overline{X})| \geq 2$, we have $A \neq E$, and since E is connected, from Cor. 2 there must exist a point x in $\Gamma^*(A)$. $\square$

Notice that the converse of Prop. 32 is false, as shown by the case of Fig. 7f which is a fusion graph, in which a given subset (black dots) has one multiple point.

Now, we present the main theorem of this section, which establishes that the class of graphs for which any watershed is thin is precisely the class of fusion graphs. As an immediate consequence of this theorem and Prop. 30, we see that all watersheds in fusion graphs, strong fusion graphs, perfect fusion graphs and line graphs are thin.

Theorem 33. *A graph G is a fusion graph if and only if any non-trivial watershed in G is thin.*

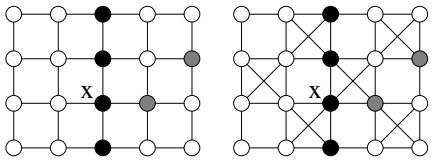
Proof: Let (E, Γ) be a fusion graph, let $Y \subset E$ be a non-trivial watershed. Suppose that $\text{int}(Y) \neq \emptyset$, and let $A \in \mathcal{C}(\text{int}(Y))$. Let $Y' = Y \setminus A$. By Prop. 14, Y' is a watershed. Since (E, Γ) is a fusion graph, from Prop. 31 we deduce that there exists a vertex $x \in \Gamma^*(A)$ which is F-simple for Y', *i.e.,* x is adjacent to exactly two connected components of $\overline{Y'}$. Since $\mathcal{C}(\overline{Y'}) = \mathcal{C}(\overline{Y}) \cup \{A\}$ (Prop. 4), this means that x is only adjacent to one connected component of $\overline{Y}$, *i.e.,* x is W-simple for Y, a contradiction with the fact that Y is a watershed. Thus, Y is thin.

Suppose now that (E, Γ) is not a fusion graph, by Prop. 31 there exists a non-trivial watershed $Y \subset E$, and there exists $A \in \mathcal{C}(\overline{Y})$ such that any $x \in \Gamma^*(A)$ cannot be F-simple. Furthermore, since Y is a watershed we know that any x in $\Gamma^*(A)$ cannot be W-simple for Y, thus any point x in $\Gamma^*(A)$ is a multiple point. Consider now the set $Y' = Y \cup A$, and let y be a point of Y'. Only three cases are possible: 1) if $y \in A$, then we can see that y is an inner point for Y', thus y is not W-simple for Y'; 2) if $y \in \Gamma^*(A)$, then as seen before, y is a multiple point for Y, thus y is adjacent to at least two connected components

of Y' consequently y is not W-simple for Y'; 3) if $y \notin \Gamma(A)$, then y is not W-simple for Y', otherwise Y could not be a watershed. Thus, Y' is a watershed. Furthermore, $A \subseteq \text{int}(Y')$ and $A \neq \emptyset$, thus Y' is not thin. $\square$

Let us look at some examples to illustrate this property. The graphs of Fig. 5c and Fig. 5d are not fusion graphs, in fact they are not even weak fusion graphs; we see that they may indeed contain a non-thin watershed. On the other hand, Fig. 5e is an example of a weak fusion graph which is not a fusion graph (see also Fig. 7w) with a watershed which is not thin.

We conclude this section with a nice property of perfect fusion graphs, which can be useful to design hierarchical segmentation methods based on watersheds. Consider the example of Fig. 8a, where a watershed X (black points) in the graph G separates $\overline{X}$ into two components. Consider now the set Y (gray points) which is a watershed in the subgraph of G induced by one of these components. We can see that the union of the watersheds, $X \cup Y$, is not a watershed, since the point x is W-simple for $X \cup Y$. Property Prop. 34 shows that this problem cannot occur in any perfect fusion graph.



(a) (b)

Fig. 8. Illustrations for Prop. 34. (a): The graph is not a perfect fusion graph (see Sec. 6, Prop. 45), and the union of the watersheds is not a watershed. (b): The graph is a perfect fusion graph (see Sec. 7, Prop. 54), the property holds.

Property 34. *Let $G = (E, \Gamma)$ be a graph. If G is a perfect fusion graph, then for any watershed $X \subset E$ in G and for any watershed $Y \subset A$ in G_A, where $A \in \mathcal{C}(\overline{X})$ and G_A is the subgraph of G induced by A, the set $X \cup Y$ is a watershed in G.*

The proof may be found in the annex. It uses Th. 33 and a local characterization of perfect fusion graphs which will be established in the next section. Fig. 8b illustrates the property with a perfect fusion graph (the set X is depicted in black and the set Y in gray).

5 Local characterizations

The definitions of weak fusion, fusion, strong fusion and perfect fusion graphs are based on conditions that must be verified for all the subsets of the vertex set. This means, if we want to check whether a graph is, for instance, a perfect

fusion graph, then using the straightforward method based on the definition will cost an exponential time with respect to the number of vertices.

On the other hand, we know that certain classes of graphs have local characterizations. For example, line graphs may be recognized thanks to Th. 6, a condition which can be checked independently in a limited neighborhood of each vertex. Do such characterizations exist for the four classes of fusion graphs? We prove in this section that weak fusion graphs and fusion graphs cannot be characterized locally, and we give local conditions for characterizing strong fusion and perfect fusion graphs.

Let (E, Γ) be a graph, let $x \in E$ and $k \in \mathbb{N}$, we denote by $\Gamma^k(x)$ the k^{th} *order neighborhood of* x, that is, $\Gamma^k(x) = \Gamma(\Gamma^{k-1}(x))$, with $\Gamma^0(x) = \{x\}$.

Property 35. *There is no local characterization of weak fusion graphs. More precisely, let k be an arbitrary positive integer. There is no property $\mathcal{P}$ on graphs such that an arbitrary graph $G = (E, \Gamma)$ is a weak fusion graph if and only if, for all $x \in E$, $\mathcal{P}[G(x,k)]$ is true, $G(x,k)$ being the subgraph of G induced by $\Gamma^k(x)$.*

Proof: It can be seen that the graphs of Fig. 9a are weak fusion graphs, while those of Fig. 9b are not. In addition, for any integer k, the same "k-local configurations" may be found in both families, for a sufficiently large graph. □

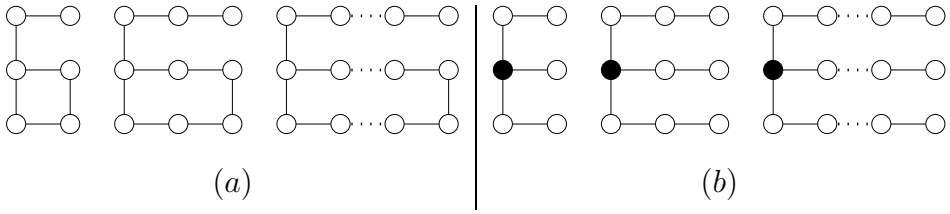


Fig. 9. Graphs for the proof of Prop. 35. In each graph of (b), the black vertices denote a set X such that the condition for a weak fusion graph is not true.

Property 36. *There is no local characterization of fusion graphs. More precisely, let k be an arbitrary positive integer. There is no property $\mathcal{P}$ on graphs such that an arbitrary graph $G = (E, \Gamma)$ is a fusion graph if and only if, for all $x \in E$, $\mathcal{P}[G(x,k)]$ is true, $G(x,k)$ being the subgraph of G induced by $\Gamma^k(x)$.*

Proof: It can be seen that the graphs of Fig. 10a are fusion graphs, while those of Fig. 10b are not. In addition, for any integer k, the same "k-local configurations" may be found in both families, for a sufficiently large graph. □

We are now going to prove that strong fusion graphs can be characterized locally. A few preliminary properties will help us to organize the proof. The following one states that in a strong fusion graph, if two neighboring compo-

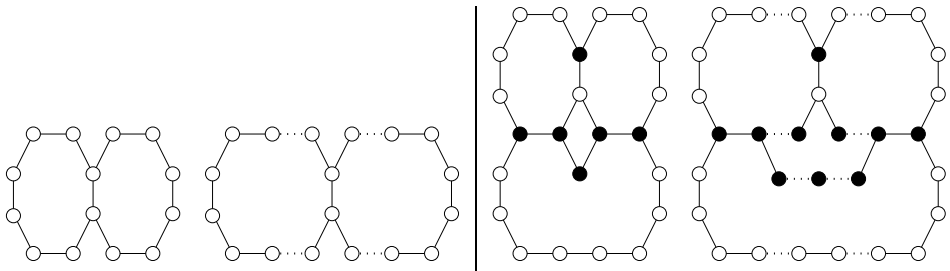


Fig. 10. Graphs for the proof of Prop. 36. In each graph of (b), the black vertices denote a set X such that the condition for a fusion graph is not true.

nents A and B can be merged, then they can be merged through a set S which is "close" to A and B, furthermore (next property), this set S can be reduced to one or two points.

Property 37. *Let $G = (E, \Gamma)$ be a graph. The graph G is a strong fusion graph if and only if for any $X \subseteq E$, for any A and $B \in \mathcal{C}(\overline{X})$ such that A, B are neighbors, there exists $S \subseteq [\Gamma^*(A) \cup \Gamma^*(B)]$ such that A and B can be merged through S.*

Proof: Suppose that G is a strong fusion graph. Let $X \subseteq E$, let A and $B \in \mathcal{C}(\overline{X})$ such that A, B are neighbors. Let $X' = X \setminus \text{int}(X)$. Thus, each point of X' is adjacent to (at least) one component of $\mathcal{C}(\overline{X'})$. Obviously, A, B are also components of $\mathcal{C}(\overline{X'})$, and $\Gamma^*(A) \cap \Gamma^*(B) \neq \emptyset$. Since (E, Γ) is a strong fusion graph, there exists a subset S of X' such that A, B can be merged through S, that is, S is F-simple for X' and adjacent to A and B. Since $\text{int}(X') = \emptyset$ and $S \subseteq X'$, we have $\text{int}(S) = \emptyset$. Thus, it can be easily seen that $S \subseteq \Gamma^*(A) \cup \Gamma^*(B)$. Since $X' \subseteq X$ and $\mathcal{C}(\overline{X}) \subseteq \mathcal{C}(\overline{X'})$ (a consequence of Prop. 4), it follows straightforwardly that S is also F-simple for X. This proves the forward implication, the converse is immediate. □

Property 38. *The graph $G = (E, \Gamma)$ is a strong fusion graph if and only if, for any $X \subseteq E$, for any A and $B \in \mathcal{C}(\overline{X})$ such that A, B are neighbors, there exists $a \in \Gamma^*(A)$ and $b \in \Gamma^*(B)$ such that A and B can be merged through $\{a, b\}$.*

Proof: Suppose that G is a strong fusion graph, let $X \subseteq E$, let A and $B \in \mathcal{C}(\overline{X})$ such that A, B are neighbors. By Prop. 37, there exists $S \subseteq [\Gamma^*(A) \cup \Gamma^*(B)]$ such that A and B can be merged through S. Without loss of generality (by Prop. 23), we may assume that S is connected. If S contains a point $x \in \Gamma^*(A) \cap \Gamma^*(B)$, then the forward implication is proved with $a = b = x$. Otherwise, S may be partitioned into two disjoint sets $A' = S \cap \Gamma^*(A)$ and $B' = S \cap \Gamma^*(B)$. Since S is connected, by Prop. 1 the sets A' and B' must be adjacent, thus there exists $a \in A'$ and $b \in B'$ which are adjacent, and since S is F-simple it can be easily seen that $\{a, b\}$ is also F-simple. This proves the

forward implication, the converse is immediate. □

Notice that in the two previous properties, the merging set S (or $\{a, b\}$) must belong to the union of $\Gamma^*(A)$ and $\Gamma^*(B)$, not to the intersection; more informally it means that A and B cannot necessarily be merged through a subset of their common boundary. To show that it is not necessary that S be included in $\Gamma^*(A) \cap \Gamma^*(B)$ for having a strong fusion graph, it suffices to consider the graph G depicted in Fig. 11. It may be checked that G is indeed a strong fusion graph. Consider the set X of black vertices, $A = \{x\}$ and $B = \{y\}$ (which are neighbors) can only be merged through $S = \{a, b\}$ which is included in $\Gamma^*(A) \cup \Gamma^*(B)$ but not in $\Gamma^*(A) \cap \Gamma^*(B)$.

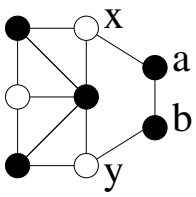


Fig. 11. Illustration of Prop. 37 and Prop. 38.

More generally, if two components A, B of $\overline{X}$ can only be merged through a two-element set $S = \{a, b\}$, it can be seen that necessarily both a and b are W-simple. This means in particular that a configuration like Fig. 11 cannot occur if X is a watershed. From this remark, we can derive a simpler characterization of strong fusion graphs, in which we consider only the subsets X of E which are watersheds.

Property 39. *The graph (E, Γ) is a strong fusion graph if and only if, for any $X \subseteq E$ which is a watershed, for any A and $B \in \mathcal{C}(\overline{X})$ such that A, B are neighbors, there exists $x \in [\Gamma^*(A) \cap \Gamma^*(B)]$ which is F-simple for X.*

We are now ready to prove the local characterization theorem for strong fusion graphs.

Let x and y be two points, we say that *x and y are 2-adjacent* if $y \notin \Gamma(x)$ and $\Gamma^*(x) \cap \Gamma^*(y) \neq \emptyset$.

Theorem 40. *Let $G = (E, \Gamma)$ be a graph. The graph G is a strong fusion graph if and only if, for any two points $x, y \in E$ which are 2-adjacent, there exists $a \in \Gamma^*(x)$ and $b \in \Gamma^*(y)$ such that $b \in \Gamma(a)$ and $\Gamma(\{a, b\}) \subseteq [\Gamma(x) \cup \Gamma(y)]$.*

Proof: Suppose that G is a strong fusion graph. Let $x, y \in E$ which are 2-adjacent, and consider the set $X = \Gamma^*(x) \cup \Gamma^*(y)$. We observe that the sets $A = \{x\}$ and $B = \{y\}$ are two elements of $\mathcal{C}(\overline{X})$. By Prop. 38, there exists $a \in \Gamma^*(x)$ and $b \in \Gamma^*(y)$, $b \in \Gamma(a)$, such that A and B can be merged through $\{a, b\}$ for X. Thus a and b must be mutually adjacent, and $\{a, b\}$ cannot be adjacent to a component of $\overline{X}$ which is neither $\{x\}$ nor $\{y\}$, hence

$\Gamma(\{a,b\}) \subseteq [\Gamma(x) \cup \Gamma(y)]$. Thus the forward implication is proved, and the converse is straightforward. □

We give below seven necessary and sufficient conditions for perfect fusion graphs. Remind that in perfect fusion graphs, any two components A, B of $\mathcal{C}(\overline{X})$ which are neighbors can be merged through $\Gamma^*(A) \cap \Gamma^*(B)$. Thus, perfect fusion graphs constitute an ideal framework for region merging methods. In the sequel, we will use the symbol $G^{\blacktriangle}$ to denote the graph of Fig. 3a.

Theorem 41. *Let (E, Γ) be a graph.*
The eight following statements are equivalent:
i) (E, Γ) is a perfect fusion graph;
ii) for any $x \in E$, any $X \subseteq \Gamma(x)$ contains at most two connected components;
iii) for any non-trivial watershed Y in E, each point x in Y is F-simple;
iv) for any connected subset A of E, the subgraph of (E, Γ) induced by A is a fusion graph;
v) for any subset X of E, there is no multiple point for X;
vi) the graph $G^{\blacktriangle}$ is not a subgraph of G;
vii) any vertices x, y, z which are mutually non-adjacent are such that $\Gamma(x) \cap \Gamma(y) \cap \Gamma(z) = \emptyset$;
viii) for any $x, y \in E$ which are 2-adjacent, for any $z \in \Gamma^*(x) \cap \Gamma^*(y)$, we have $\Gamma(z) \subseteq [\Gamma(x) \cup \Gamma(y)]$.

Proof: We will show that *[not ii]* ⇒ *[not iii]* ⇒ *[not iv]* ⇒ *[not v]* ⇒ *[not vi]* ⇒ *[not vii]* ⇒ *[not viii]* ⇒ *[not i]* ⇒ *[not ii]*, hence the equivalence of the eight statements.
• [*not ii* ⇒ *not iii*] Suppose that there exists $x \in E$ and there exists $X \subseteq \Gamma(x)$ which contains three distinct connected components A, B, C. Let $Y = E \setminus (A \cup B \cup C)$, and let Z be a watershed of Y. Necessarily, $x \in \overline{X}$ and thus $x \in Y$. Furthermore, since x is adjacent to three distinct components of $\overline{Y}$, we know that $x \in Z$ and that x is also adjacent to three distinct components of $\overline{Z}$, and thus is not F-simple for Z.
• [*not iii* ⇒ *not iv*] Suppose that there exist a non-trivial watershed Y and a point $x \in Y$ which is not F-simple for Y. Since Y is a watershed, we know that x is not either a W-simple point. If x is an inner point, by Th. 33 we deduce that (E, Γ) cannot be a fusion graph, and thus condition *iv* does not hold for $A = E$. Otherwise, x is a multiple point for Y. Then, consider the set $A = [\Gamma(x) \setminus Y] \cup \{x\}$. Let (A, Γ_A) be the subgraph of (E, Γ) induced by A, and let $X = \{x\}$. The set A is connected, and since x is a multiple point for Y, $A \setminus X$ must contain at least three connected components for (A, Γ_A), furthermore these components cannot be merged for X since x is the only point separating them. Thus (A, Γ_A) is not a fusion graph.
• [*not iv* ⇒ *not v*] Suppose that there exists a connected subset A of E such that the restriction (A, Γ') of (E, Γ) to A is not a fusion graph. By Prop. 32, there exists $X \subset A$ and $x \in X$ such that x is a multiple point for X in (A, Γ').

Obviously, x is also a multiple point for $[E \setminus A] \cup X$ in (E, Γ).
• [*not v* $\Rightarrow$ *not vi*] Suppose that there exists a subset X of E and a point $x \in X$ which is a multiple point, *i.e.,* x is adjacent to three distinct connected components A, B, C of $\overline{X}$. Let $w \in \Gamma(x) \cap A$, $y \in \Gamma(x) \cap B$, and $z \in \Gamma(x) \cap C$. Since A, B, C are distinct connected components of $\overline{X}$, w, y, z are mutually non-adjacent, thus the subgraph induced by $\{x, y, z, w\}$ is $G^{\blacktriangle}$.
• [*not vi* $\Rightarrow$ *not vii*] Suppose that the subgraph of G induced by some points $\{x, y, z, w\}$ is $G^{\blacktriangle}$, the central point being x. We have $x \in \Gamma(w) \cap \Gamma(y) \cap \Gamma(z)$, and w, y, z are mutually non-adjacent.
• [*not vii* $\Rightarrow$ *not viii*] Let w, y, z be three mutually non-adjacent points of E such that $\Gamma(w) \cap \Gamma(y) \cap \Gamma(z) \neq \emptyset$, and let $x \in \Gamma(w) \cap \Gamma(y) \cap \Gamma(z)$. We have y and z which are 2-adjacent, $x \in \Gamma^*(y) \cap \Gamma^*(z)$, but $\Gamma(x)$ contains w which is not in $\Gamma(y) \cup \Gamma(z)$ by hypothesis.
• [*not viii* $\Rightarrow$ *not i*] Let $y, z \in E$ be two points which are 2-adjacent, and let $x \in \Gamma^*(y) \cap \Gamma^*(z)$ such that there exists $w \in \Gamma(x)$, $w \notin \Gamma(y) \cup \Gamma(z)$. Let $X = E \setminus \{y, z, w\}$. Let $A = \{y\}$, $B = \{z\}$, and $C = \{w\}$. From our hypothesis, we know that A, B and C belong to $\mathcal{C}(\overline{X})$. Let $S = \Gamma^*(A, B) = \Gamma^*(A) \cap \Gamma^*(B)$, clearly $x \in S$. Since x is also adjacent to C, A and B (which are neighbors) cannot be merged through S, and the graph is not a perfect fusion graph.
• [*not i* $\Rightarrow$ *not ii*] We will prove in fact that *ii* $\Rightarrow$ *i*. Suppose that *ii* holds, and let $X \subset E$, let $A, B \in \mathcal{C}(\overline{X})$ such that $\Gamma^*(A, B) \neq \emptyset$. For any x in $\Gamma^*(A, B)$, from the hypothesis (*ii*) we deduce that x is only adjacent to A and B. Furthermore $A \cup B \cup \Gamma^*(A, B)$ is obviously connected, thus $\Gamma^*(A, B)$ is F-simple for X, and A and B can be merged through $\Gamma^*(A, B)$. □

Notice that condition *viii* bears a resemblance with the local characterization of strong fusion graphs (Th. 40).

Remind that any line graph is a perfect fusion graph (Prop. 30). We can see that, thanks to Th. 41 (condition *vi*), perfect fusion graphs can be characterized in a way similar to Th. 6 which characterizes line graphs, but with a much simpler condition.

A consequence of Th. 41 is that all the graphs of Fig. 3 except graph $G^{\blacktriangle}$ are perfect fusion graphs, since none of these graphs contains $G^{\blacktriangle}$ as a subgraph. The reader can also check anyone of the previous eight conditions on these graphs, as an illustration of Th. 41.

Corollary 42. *Let $G = (E, \Gamma)$ be a graph, let X be any connected subset of E. If G is a perfect fusion graph, then the subgraph of G induced by X is also a perfect fusion graph.*

6 Usual grids

The aim of this section and the following one is to answer the question: which are the grids that may be used in order to perform safe merging operations on digital images? In this section, we consider the different grids commonly used in 2-dimensional and 3-dimensional image processing. Our major result is that none of these grids is a perfect fusion graph and several are not even fusion graphs. One of the consequences is that the most natural merging operation, which consists in merging two regions through their common boundary, is not a safe operation in these grids.

We start with some basic definitions which allow to structure the pixels of an image. In this section and the following one, we will assume that n is a strictly positive integer.

Definition 43. *Let E be a set and let E^n be the Cartesian product of n copies of E. An element x of E^n may be seen as a map from $\{1, ..., n\}$ to E, for each $i \in \{1, ..., n\}$, x_i is* the i-th coordinate of x.
Let $\mathbb{Z}$ be the set of integers. We consider the families of sets H_0^1, H_1^1 such that $H_0^1 = \{\{a\} \mid a \in \mathbb{Z}\}$, $H_1^1 = \{\{a, a+1\} \mid a \in \mathbb{Z}\}$. A subset S of $\mathbb{Z}^n$ which is the Cartesian product of exactly $m \leq n$ elements of H_1^1 and $(n-m)$ elements of H_0^1 is called a m-cube.

In order to recover a graph structure for digital images, adjacency relations are defined on $\mathbb{Z}^n$. The following definition allows to retrieve the most frequently used adjacency relations.

Definition 44. *Let $m \leq n$, we say that x and y in $\mathbb{Z}^n$ are* m-adjacent *if there exists a m-cube that contains both x and y. We define Γ_m^n as the binary relation on $\mathbb{Z}^n$ such that for any pair x, y in E, $(x, y) \in \Gamma_m^n$ if and only if x and y are m-adjacent.*

In order to deal with graphs that can be arbitrary large we define a *grid* as a pair (E, Γ) where E is an infinite set and Γ is a binary relation on E. Let $X \subseteq E$ we define the restriction of (E, Γ) to X as the pair (X, Γ_X) where $\Gamma_X = \Gamma \cap (X \times X)$. If X is a finite set (X, Γ_X) is a graph. In the sequel, to simplify the notations, we will write Γ as a shortcut for Γ_X.

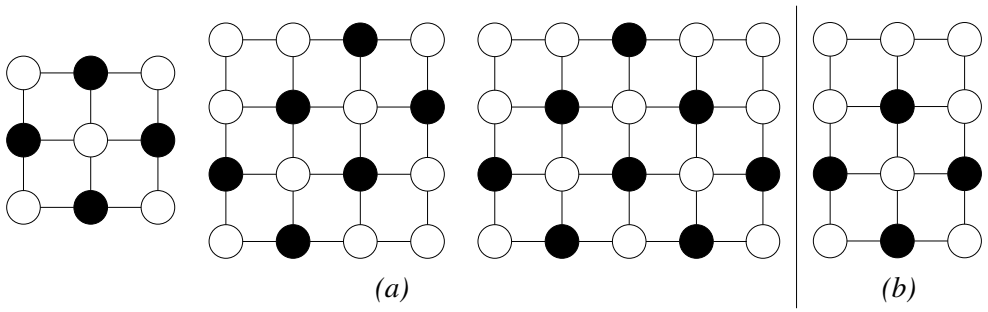


Fig. 12. (a): Counter-examples for the weak fusion property of (E, Γ_1^2); The black points represent a set X; (b): counter-example for the fusion property of (E, Γ_1^2) when $\{w, h\} = \{3, 4\}$.

6.1 2-dimensional usual grids

Let w, h be two integers strictly greater than 1, called respectively *width* and *height*, we set $E = \{x \in \mathbb{Z}^2 \mid 0 \leq x_1 < w \text{ and } 0 \leq x_2 < h\}$. In this section we study the connected graph (E, Γ_1^2) (resp. (E, Γ_2^2)) which is the restriction of $(\mathbb{Z}^2, \Gamma_1^2)$ (resp. $(\mathbb{Z}^2, \Gamma_2^2)$) to E.
Notice that in the literature, the graph (E, Γ_1^2) (resp. (E, Γ_2^2)) corresponds to the 4 (resp. 8)-adjacency.

Property 45. *Let $w > 2$ and $h > 2$. If $\{w, h\} \neq \{3, 4\}$, (E, Γ_1^2) is not a weak fusion graph. If $\{w, h\} = \{3, 4\}$ then (E, Γ_1^2) is a weak fusion graph but not a fusion graph.*

Proof: If $\{w, h\} \neq \{3, 4\}$, let us consider the following set:
(1): if both w and h are odd, $X = \{(i, j) \mid i + j \text{ is odd }\}$;
(2): if only w is odd, $X = \{(i, j) \mid i + j \text{ is odd }\} \setminus \{(0, h-1), (w-1, h-1)\}$;
(3): if only h is odd, $X = \{(i, j) \mid i + j \text{ is odd }\} \setminus \{(w-1, 0), (w-1, h-1)\}$;
(4) if both w and h are even, $X = \{(i, j) \mid i+j \text{ is odd }\} \setminus \{(0, h-1), (w-1, 0)\}$.
Fig. 12a shows the set X for image domains of size 3×3, 4×4 and 5×4.
It may be easily checked that any connected component of $\overline{X}$ cannot be merged for X.
Let $\{w, h\} = \{3, 4\}$. Then (E, Γ_1^2) is a weak fusion graph (exhaustive check). The graph of Fig. 12b shows a set X such that there exists connected components of $\overline{X}$ which cannot be merged, hence (E, Γ_1^2) is not a fusion graph.□

Let $X \subseteq E$, we say that $x \in X$ matches C_1 if the neighborhood of x corresponds to the configuration C_1 depicted in Fig. 13a or to one of its $\pi/2$ rotations. In Fig. 13, points labelled B are in X, points labelled W are in $\overline{X}$, at least one of the points labelled U is in $\overline{X}$ and the point I is either in X or in $\overline{X}$.

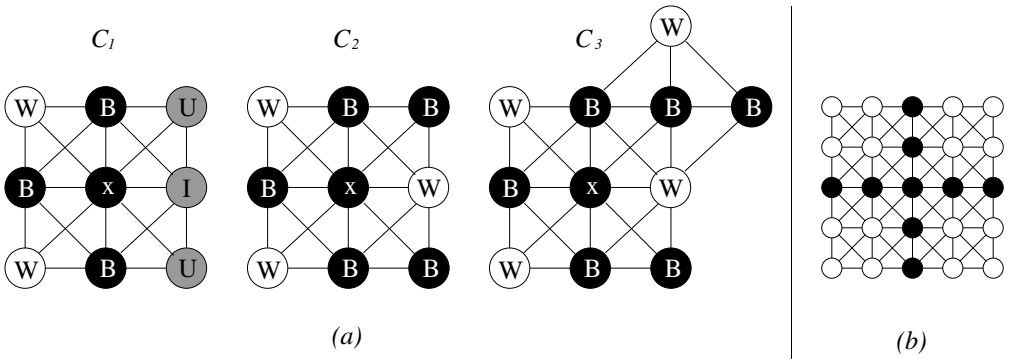


Fig. 13. (a): Local configurations which are used for proving Lem. 47; configurations C_1 and C_2 are the local configurations of multiple points in (E, Γ_2^2); (b): counter-example for the strong fusion property of (E, Γ_2^2).

We say that x matches C_2 if the neighborhood of x corresponds to the configuration C_2 depicted in Fig. 13a or to one of its $\pi/2$ rotations.

Lemma 46. *Let $X \subseteq E$ be a watershed on (E, Γ_2^2). Then any x in X which is multiple matches either C_1 or C_2.*

Proof: Exhaustive check. □

Lemma 47. *Let $X \subseteq E$ be a non-trivial watershed on (E, Γ_2^2). Then any $A \in \mathcal{C}(\overline{X})$ can be merged.*

Proof: Suppose that A cannot be merged, then any $x \in X \cap \Gamma_2^2(A)$ is multiple. As (E, Γ_2^2) is connected and $\mathcal{C}(\overline{X}) > 2$, such x exists. Thus x matches C_1 or C_2. Suppose that x matches C_1. If the two points labelled W in C_1 belong to the same connected component of $\overline{X}$ then the point at the west of x is W-simple, a contradiction with the fact that X is a watershed. Thus necessarily these two points belong to distinct components of $\overline{X}$, and the point at the west of x is F-simple. If A contains one of the these two points, labelled W in C_1, then A is adjacent to an F-simple point and thus can be merged. Otherwise A contains one of the points labelled U. In this case the same arguments can be used to prove that A can be merged, thus x does not match C_1.
Suppose that x matches C_2. For the same reasons, A is the connected component that contains the point at the east of x. As A cannot be merged, necessarily the point which is at the north of x is multiple. Then the only possible configuration is C_3, which is depicted in Fig. 13a. In configuration C_3, it can be verified that the point at the north-east of x is necessarily F-simple. Thus A can be merged, a contradiction. □

Property 48. *Let $h > 2$ and $w > 2$, the graph (E, Γ_2^2) is a fusion graph but is not a strong fusion graph.*

Proof: The fact that (E, Γ_2^2) is a fusion graph is a direct corollary of Lem. 47

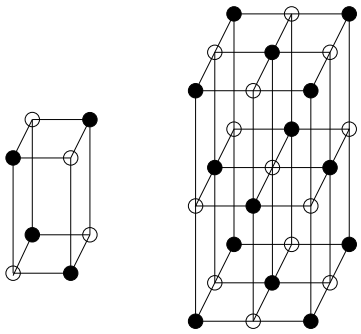


Fig. 14. Counter-examples for the weak fusion property of (E, Γ_1^3). The black points represent a set X.

and Th. 33. Let us consider the set X, composed by the black points in Fig. 13b. It can be seen that this type of "global cross configurations" can be extended whatever the size of E (with $h > 2$ and $w > 2$). In these "cross configurations", the connected components which are diagonally neighbor to each other cannot be merged. Thus the graph (E, Γ_2^2) is not a fusion graph. □

6.2 3-dimensional usual grids

Let w, h and d be three integers strictly greater than 1, called respectively *width*, *height* and *depth*, we set $E = \{x \in \mathbb{Z}^3 \mid 0 \leq x_1 < w, 0 \leq x_2 < h$ and $0 \leq x_3 < d\}$. In the sequel we will consider that $w > 1, h > 1$ and $d > 1$. In this section we study the graphs (E, Γ_1^3) (resp. (E, Γ_3^3)) which is the restriction of $(\mathbb{Z}^3, \Gamma_1^3)$ (resp. $(\mathbb{Z}^3, \Gamma_3^3)$) to E.
Notice that in the literature, the graph (E, Γ_1^3) (resp. (E, Γ_3^3)) corresponds to the 6 (resp. 26)-adjacency.

Property 49. *The graph (E, Γ_1^3) is not a weak fusion graph.*

Proof: Let us consider the set X such that $X = \{x \in E \mid$ the number of odd coordinates of x is equal to 0 or 2 $\}$. This set corresponds to a 3-dimensional chessboard. Samples of such a set are shown in Fig. 14. It may be easily seen that any element of $\overline{X}$ is a connected component that cannot be merged without involving at least two other connected components. Hence the graph is not a weak fusion graph. □

Property 50. *If $w \geq 5$, $h \geq 5$, $d \geq 5$, the graph (E, Γ_3^3) is not a fusion graph.*

Proof: Let us consider the set $\overline{X}$ of white points depicted in Fig. 15. Whatever the size of E and supposing that all points of E outside the figure are in X, it may be seen that the central point x is such that $\{x\}$ is a connected component of $\overline{X}$. Any point 3-adjacent to x (the set of gray points) is adjacent to at least three distinct connected components of $\overline{X}$. Thus any attempt to merge $\{x\}$

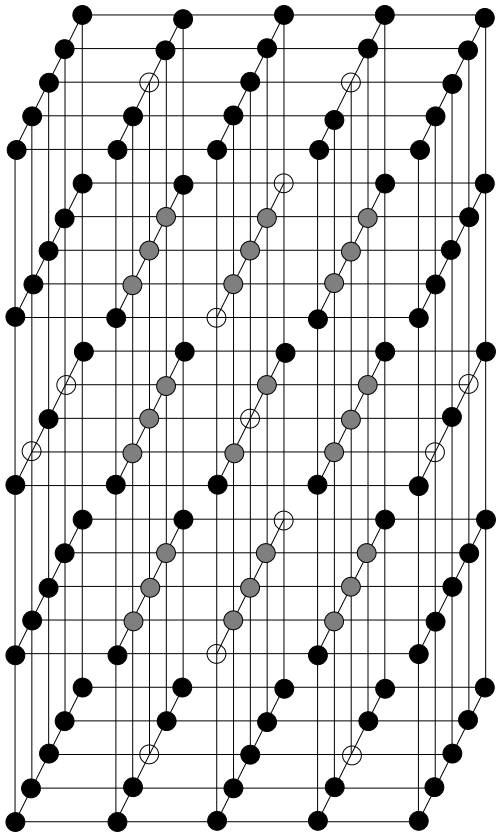
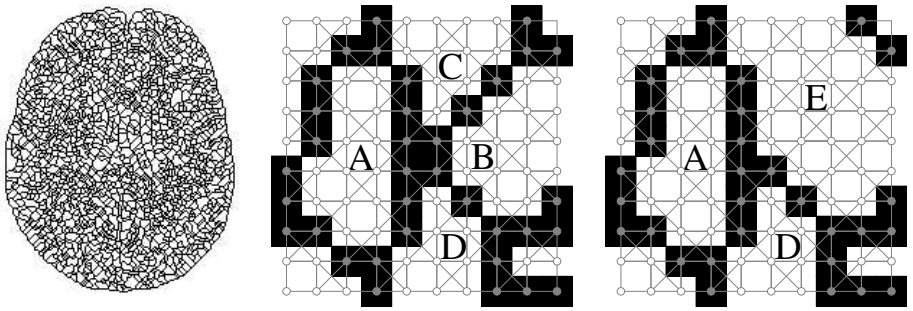


Fig. 15. Counter-example for the fusion property of (E, Γ_3^3). Black and gray points represent a set X.

will involve three connected components of $\overline{X}$, hence $\{x\}$ cannot be merged, (E, Γ_3^3) is not a fusion graph. □

7 Perfect fusion grid

We introduce a grid for structuring n-dimensional digital images and prove that it is a perfect fusion graph, whatever the dimension n. It does thus constitute a structure on which regions can be safely merged.
Let us give an intuitive idea of this grid. Fig. 16a shows a watershed of Fig. 1a obtained on this grid. It can be easily seen that the problems pointed out in the introduction do not exist in this case. The watershed does not contain any inner point. Any pair of neighboring regions can be merged by simply removing from the watershed the points which are adjacent to both regions (see Fig. 16b,c). Furthermore, the resulting set is still a watershed.
It may be seen that this grid is "between" the usual grids. In [7] we prove that this grid is the unique such graph.



(a) (b) (c)

Fig. 16. (a): A watershed of Fig. 1 obtained on the perfect fusion grid; (b): a crop of (a) where the region A, B, C and D coresponds to the region shown in Fig. 1d; in gray, the coresponding perfect fusion grid is superimposed; (c): same as (d) after having merged B and C to form a new region, called E.

Let C^n be the set of all n-cubes of $\mathbb{Z}^n$, we define the map B from C^n to $\mathbb{Z}^n$, such that for any $c \in C^n$, $B(c)_i = \min\{x_i \mid x \in c\}$, where $B(c)_i$ is the i-th coordinate of $B(c)$. It may be seen that c is equal to the Cartesian product: $\{B(c)_1, B(c)_1 + 1\} \times ... \times \{B(c)_n, B(c)_n + 1\}$. Thus clearly B is a bijection.
We set $\mathbb{B} = \{0, 1\}$. We set $\overline{0} = 1$ and $\overline{1} = 0$. A *binary word of length* n is an element of $\mathbb{B}^n$. If u is in $\mathbb{B}^n$, we define *the complement of* u as the binary word $\overline{u}$ such that for any $i \in \{1, ..., n\}$, $(\overline{u})_i = (\overline{u_i})$.

Before defining perfect fusion grids, we first recall the definition of cliques, and a property due to Berge which uses maximal cliques to characterize some line graphs. This property will be used in the proof of Prop. 54.
Let E be a set, let Γ be a binary relation on E and let $X \subseteq E$. We say that X is a *clique (for* (E, Γ)*)* if $X \times X \subseteq \Gamma$. In other words, X is a clique if any two vertices of X are adjacent. We say that X is a *maximal clique* if, for any clique X', $X \subseteq X'$ implies $X' = X$.

Property 51 (Prop. 7 in [2], chapter 17)**.** *Let $G = (E, \Gamma)$ be a graph. If for any $x \in E$, x is in at most two distinct maximal cliques, then G is a line graph.*

Definition 52. *Let f be the map from C^n to $\mathbb{B}^n$ such that for any $c \in C^n$, $f(c)_i$ is equal to $B(c)_i$ mod 2, that is the remainder in the integer division of $B(c)_i$ by 2.*
Let u be an element of $\mathbb{B}^n$, we set $C^n_u = \{c \in C^n \mid f(c) = u\}$ and $C^n_{u/\overline{u}} = C^n_u \cup C^n_{\overline{u}}$.
We define the binary relation $\Gamma^n_{u/\overline{u}} \subseteq \mathbb{Z}^n \times \mathbb{Z}^n$ as the set of pairs $(x, y) \in \mathbb{Z}^n \times \mathbb{Z}^n$ such that there exists $c \in C^n_{u/\overline{u}}$ that contains both x and y.
We define $\mathcal{P}^n$, the family of perfect fusion grids over $\mathbb{Z}^n$*, as the set $\mathcal{P}^n =$*

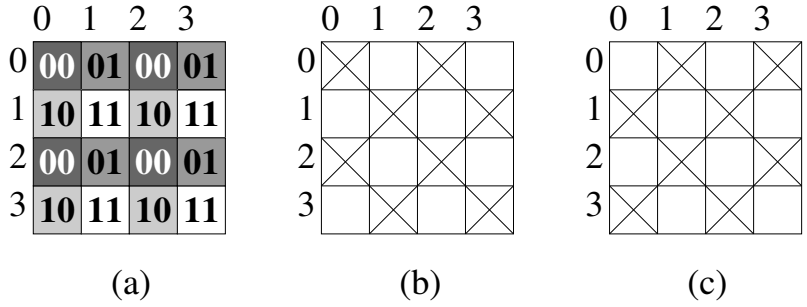


Fig. 17. Illustration of the two perfect fusions grids over $\mathbb{Z}^2$ (restricted to subsets of $\mathbb{Z}^2$). (a): The map f; (b): $(\mathbb{Z}^2, \Gamma^2_{11/00})$; ($c$): $(\mathbb{Z}^2, \Gamma^2_{10/01})$.

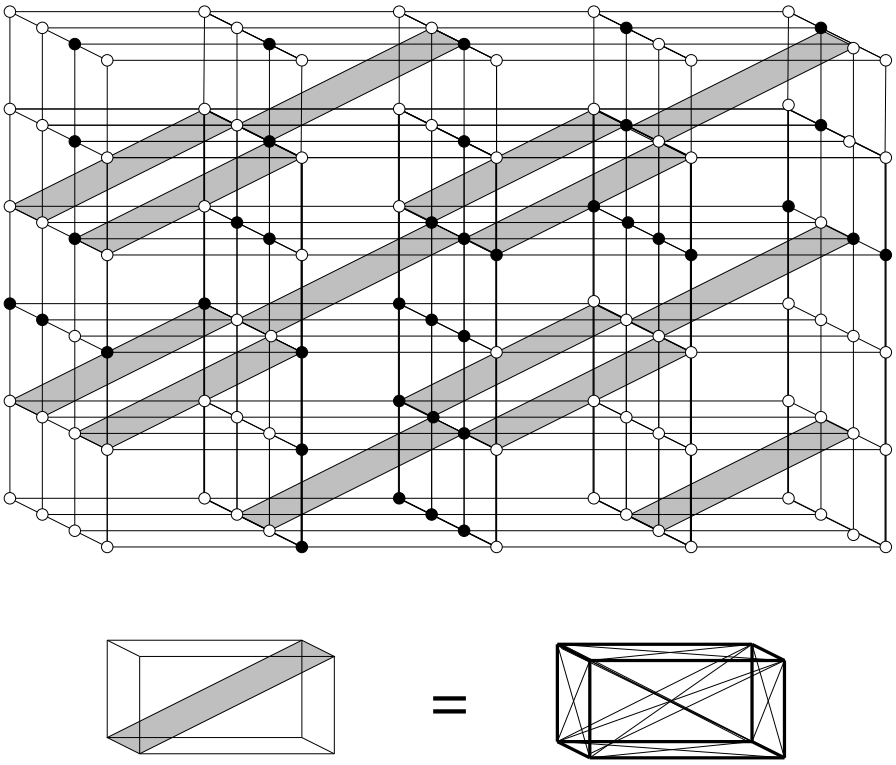


Fig. 18. A 3-dimensional perfect fusion grid. Black points constitute a set which is a watershed.

$\{(\mathbb{Z}^n, \Gamma^n_{u/\overline{u}}) \mid u \in \mathbb{B}^n\}$.

Fig. 17 illustrates the above definitions for the two-dimensional case. Fig. 18 shows a watershed on a 3-dimensional perfect fusion grid. To clarify the figure, we use the following convention: any two points belonging to a same cube marked by a gray stripe are adjacent to each other.

In the sequel, to simplify the notations, we will write c_i as a shortcut for $B(c)_i$.

Lemma 53. *Let $u \in \mathbb{B}^n$ and let $x \in \mathbb{Z}^n$.*

i) There exists a unique c in C^n_u such that $x \in c$.
ii) The point x is in exactly two maximal cliques of $(\mathbb{Z}^n, \Gamma^n_{u/\overline{u}})$.

Proof: It may be easily seen that any element c of C^n which contains x is such that for any $i \in \{1, ..., n\}$, $c_i = x_i - 1$ or $c_i = x_i$, hence i).
We deduce from i) that there are exactly two distinct elements c and c' of $C^n_{u/\overline{u}}$ such that $c \in C^n_u$, $c' \in C^n_{\overline{u}}$ and such that x is in both c and c'. Thus any element adjacent to x is either in c or in c'. From the very definition of $\Gamma^n_{u/\overline{u}}$, any pair of elements of c (resp. c') is in $\Gamma^n_{u/\overline{u}}$. Thus c and c' are cliques of $(\mathbb{Z}^n, \Gamma^n_{u/\overline{u}})$, which both contain x. Since any pair (y, y') with $y \in c \setminus c'$, $y' \in c' \setminus c$ is not in $\Gamma^n_{u/\overline{u}}$, we conclude that x is in exactly two maximal cliques. □

Property 54. *Let $u \in \mathbb{B}^n$ and let X be a finite subset of $\mathbb{Z}^n$ such that $(X, \Gamma^n_{u/\overline{u}})$ is connected. Then $(X, \Gamma^n_{u/\overline{u}})$ is a perfect fusion graph. Furthermore it is a line graph.*

Proof: From Lem. 53, any x in X is in at most two maximal cliques. Thus, as a consequence of Prop. 51, $(X^n, \Gamma^n_{u/\overline{u}})$ is a line graph and from Prop. 30 it is a perfect fusion graph. □

The following property shows that the perfect fusion grid is "between" the usual adjacency relations on $\mathbb{Z}^n$.

Property 55. *Let $u \in \mathbb{B}^n$. We have: $\Gamma^n_1 \subseteq \Gamma^n_{u/\overline{u}} \subseteq \Gamma^n_n$.*

Proof: From Lem. 53, we know that for any $x \in \mathbb{Z}^n$ there exist exactly two maximal cliques $c \in C^n_u$ and $c' \in C^n_{\overline{u}}$ that contain x. Necessarily there exists k such that $B(c) = x - k$ with $k \in \mathbb{B}^n$ and $B(c') = x - \overline{k}$. A point x' is in $\Gamma^n_1(x)$ if there exists a unique $j \in \{1, ..., n\}$ such that $x'_j = x_j + 1$ or $x'_j = x_j - 1$ and for any $i \in [\{1, ..., n\} \setminus \{j\}]$, $x'_i = x_i$. Suppose that $x'_j = x_j - 1$. The case where $x'_j = x_j + 1$ is symmetric to this one and the following arguments hold for both cases. For any $i \in [\{1, ..., n\} \setminus \{j\}]$, either $k_i = 0$ or $k_i = 1$. If $k_i = 0$, then $x'_i = x_i = c_i = c'_i + 1$. If $k_i = 1$, then $x'_i = x_i = c'_i = c_i + 1$. On the other hand, if $k_j = 1$ then $x'_j = x_j - 1 = c_j$, hence $x' \in c$. Otherwise, if $k_j = 0$ then $x'_j = x_j - 1 = c'_j$, hence $x' \in c'$. Whatever the case, $(x, x') \in \Gamma^n_{u/\overline{u}}$, hence $\Gamma^n_1 \subseteq \Gamma^n_{u/\overline{u}}$. The proof of the second inclusion follows straightforwardly from the definition of $\Gamma^n_{u/\overline{u}}$.□

Property 56. *The family $\mathcal{P}^n$ contains 2^{n-1} distinct perfect fusion grids.*

Proof: From the very definition of perfect fusion grids, we have $\Gamma^n_{u/\overline{u}} = \Gamma^n_{\overline{u}/u}$. Furthermore, if $\{u, \overline{u}\} \neq \{v, \overline{v}\}$ then $\Gamma^n_{u/\overline{u}} \neq \Gamma^n_{v/\overline{v}}$. Since the cardinality of $\mathbb{B}^n$ is equal to 2^n, the cardinality of $\mathcal{P}^n$ is equal to $2^n/2 = 2^{n-1}$. □

Let $X \subseteq \mathbb{Z}^n$ and let $t \in \mathbb{B}^n$. We define $X + t = \{x + t \mid x \in X\}$, we say that $X + t$ is a *binary translation of X*. Let m be a positive integer such that

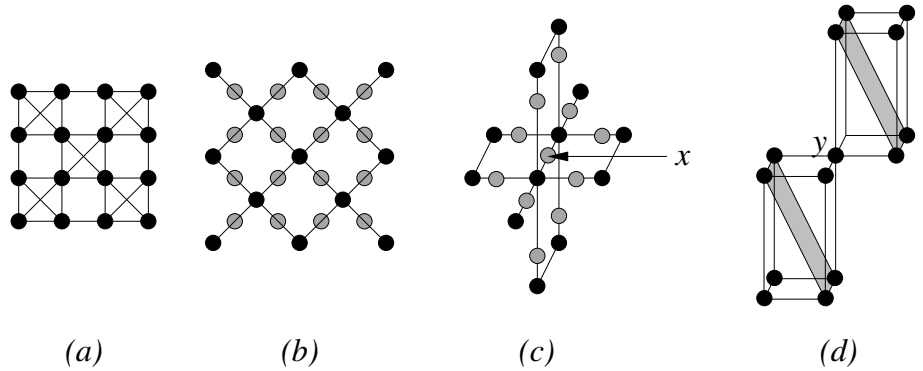


Fig. 19. Illustrations of the relation between line graphs of 1-connected graph and perfect-fusion grids. (a): a restriction of the 2-dimensional perfect fusion grid; (b): a graph (black points and edges) whose line graph is (a); the gray points indicate corresponding vertices of the line graph (a) of (b); (c): black points and edges depict a local configuration of the 3-dimensional 1-connected grid; the gray points indicate corresponding vertices of the line graph of (c) in which any gray point is adjacent to x; (d): a local configuration of the perfect fusion grid, any black point is adjacent to y.

$m \leq n$. Remark that if X is an m-cube then $X + t$ is also an m-cube.
The following property states that any two n-dimensional perfect fusion grids are equivalent up to a binary translation.

Property 57. *Let u and v in $\mathbb{B}^n$. Let $t \in \mathbb{B}^n$ such that for any $i \in \{1, ..., n\}$, if $u_i = \overline{v_i}$ then $t_i = 1$, otherwise $t_i = 0$. Then for any $(x, y) \in \mathbb{Z}^n \times \mathbb{Z}^n$, $(x, y) \in \Gamma^n_{u/\overline{u}}$ if and only if $(x + t, y + t) \in \Gamma^n_{v/\overline{v}}$.*

Proof: It can easily be seen that for any $c \in C^n$, $f(c) = u$ (resp. $f(c) = \overline{u}$) if and only if $f(c + t) = v$ (resp. $(f(c + t) = \overline{v})$). The result follows from this observation and from the definition of the perfect fusion grids.□

Let u in $\mathbb{B}^2$. Let X be a finite subset of $\mathbb{Z}^2$. It can be seen that $(E, \Gamma^2_{u/\overline{u}})$ is the line graph of a graph (E', Γ^2_1), with $E' \subset \mathbb{Z}^2$. For example, Fig. 19a shows a 2-dimensional perfect fusion grid, its associated graph (E', Γ^2_1) is depicted in Fig. 19b.
Remark that a similar statement is not true in dimension 3. Local configurations of $(\mathbb{Z}^3, \Gamma^3_1)$ and of its line graph are depicted in Fig. 19c. A local configuration of $(\mathbb{Z}^3, \Gamma^3_{u/\overline{u}})$ is depicted in Fig. 19d. It can be checked that the point x in Fig. 19c has exactly 10 neighbors whereas the point y in Fig. 19d has 14 neighbors. Thus those two configurations cannot be isomorphic.

Conclusion

This article sets up a theoretical framework for the study of merging properties in graphs. Using this framework, we obtained a necessary and sufficient condition for the thinness of watersheds, we defined four classes of graphs in relation to these merging properties and gave local characterizations of these classes whenever possible. We also analyzed the status of the graphs which are the most widely used for image analysis, and proposed a family of graphs on $\mathbb{Z}^n$ which constitute an ideal support for region merging.

A forthcoming article [8] extends this study to the case of weighted graphs, which constitute a model for grayscale images. In this study, the notion of degree of connectedness for grayscale images, introduced by A. Rosenfeld [14] will play an important role. The notion of topological watershed [3,5] extends the notion of watershed to weighted graphs, and possess interesting properties which are not guaranteed by most popular watershed algorithms [11]. The major outcomes of [8] are:
i) a proof that any topological watershed on any perfect fusion graph is thin;
ii) a new, simple and linear-time algorithm to compute topological watersheds on perfect fusion graphs.

References

[1] L.W. Beineke. On derived graphs and digraphs. In H. Sachs, H.J. Voss, and H. Walther, editors, *Beiträge zur graphen theorie*, pages 17–23. Teubner, 1968.

[2] C. Berge. *Graphes et hypergraphes.* Dunod, 1970.

[3] G. Bertrand. On topological watersheds. *Journal of Mathematical Imaging and Vision*, 22(2-3):217–230, May 2005. Special issue on Mathematical Morphology.

[4] S. Beucher and Ch. Lantuéjoul. Use of watersheds in contour detection. In *procs. Int Workshop on Image Processing Real-Time Edge and Motion Detection/Estimation*, 1979.

[5] M. Couprie and G. Bertrand. Topological grayscale watershed transform. In *SPIE Vision Geometry V Proceedings*, volume 3168, pages 136–146, 1997.

[6] M. Couprie, L. Najman, and G. Bertrand. Quasi-linear algorithms for the topological watershed. *Journal of Mathematical Imaging and Vision*, 22(2-3):231–249, May 2005. Special issue on Mathematical Morphology.

[7] J. Cousty and G. Bertrand. Uniqueness of n-dimensionnal perfect fusion grid. 2006. In preparation.

[8] J. Cousty, M. Couprie, L. Najman, and G. Bertrand. Fusion graphs: grayscale watersheds and merging properties. 2006. In preparation.

[9] J. Cousty, L. Najman, M. Couprie, and G. Bertrand. Watershed on egdes. *Image and Vision Computing*, 2006. To be submitted, Special issue on ISMM05.

[10] F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, 1990.

[11] L. Najman, M. Couprie, and G. Bertrand. Watersheds, mosaics and the emergence paradigm. *Discrete Applied Mathematics*, 147(2-3):301–324, April 2005. Special issue on DGCI.

[12] T. Pavlidis. *Structural Pattern Recognition*, volume 1 of *Springer Series in Electrophysics*, chapter 4–5, pages 90–123. Springer-Verlag, 1977. segmentation techniques.

[13] A. Rosenfeld. Connectivity in digital pictures. *Journal of the Association for Computer Machinery*, 17:146–160, 1970.

[14] A. Rosenfeld. On connectivity properies of grayscale pictures. *Pattern Recognition*, 16:47–50, 1983.

[15] A. Rosenfeld and A.C. Kak. *Digital picture processing*, volume 2, chapter 10. Academic Press, 1982. Section 10.4.2.d (region merging).

Annex

Proof of Prop. 17: Since $|\mathcal{C}(\overline{X})| \geq 2$ we have $A \cup \text{ann}(A) \neq E$, and since E is connected, from Cor. 2 there must exist a point x in $\Gamma^*(A \cup \text{ann}(A))$. Furthermore, x must be adjacent to at least one component B of $\overline{X}$ distinct from A, otherwise $\text{ann}(A) \cup \{x\}$ would be W-simple for $\overline{X}$, a contradiction with the definition of $\text{ann}(A)$; and x cannot belong to B, otherwise $\text{ann}(A)$ would not be W-simple for $\overline{X}$, also a contradiction with the definition of $\text{ann}(A)$. □

Proof of Prop. 22:
Suppose that $A \cup B \cup S \in \mathcal{C}(\overline{X \setminus S})$. Let $C \in \mathcal{C}(\overline{X}|S)$, then $A \cup B \cup S \cup C$ is connected and $A \cup B \cup S \subseteq A \cup B \cup S \cup C \subseteq \overline{X \setminus S}$. Since $\overline{X} \neq \emptyset$, as a connected component of $\overline{X}$ the set C cannot be empty, and since $A \cup B \cup S \in \mathcal{C}(\overline{X \setminus S})$, we must have either $C = A$ or $C = B$.

Suppose now that S is F-simple for X and adjacent to A and B. Thus, $A \cup B \cup S$ is connected, it remains to prove that it is maximal. Let $Z \subset E$ such that $A \cup B \cup S \subseteq Z \subseteq \overline{X \setminus S}$, and Z connected. Let $Y = Z \setminus [A \cup B \cup S]$. Since $Z \subseteq \overline{X \setminus S}$, we have $Y \subseteq \overline{X}$. Since A (resp. B) belongs to $\mathcal{C}(\overline{X})$, Y cannot be adjacent to A (resp. to B), and since $\mathcal{C}(\overline{X}|S) = \{A, B\}$, Y cannot be adjacent to S. Since Z is connected, by Prop. 1 we deduce that Y must be empty, thus $Z = A \cup B \cup S$, and $A \cup B \cup S$ is a component of $\overline{X \setminus S}$. The other components of $\overline{X \setminus S}$ are clearly the components of $\overline{X}$ which differ from A and B. □

Lemma 58. *Any strong fusion graph is a fusion graph.*

Proof: Let $G = (E, \Gamma)$ be a strong fusion graph, let $X \subset E$ such that $|\mathcal{C}(\overline{X})| \geq 2$, and let $A \in \mathcal{C}(\overline{X})$. By Prop. 17, there exists $B \in \mathcal{C}(\overline{X})$, $B \neq A$, such that $A \cup \text{ann}(A)$ and B are neighbors. Since G is a strong fusion graph, there exists $S \subseteq [X \setminus \text{ann}(A)]$ such that $A \cup \text{ann}(A)$ and B can be merged through S for $X \setminus \text{ann}(A)$. Consider $S' = S \cup \text{ann}(A)$, it can easily be seen that S' is adjacent to exactly two components of $\overline{X}$, namely A and B, thus A can be merged for X. □

Lemma 59. *Let (E, Γ) be a graph. Let $X \subseteq E$, let $A \in \mathcal{C}(\overline{X})$, and let $Y \subseteq A$. Then, we have $\mathcal{C}(\overline{X \cup Y}) = [\mathcal{C}(\overline{X}) \setminus \{A\}] \cup \mathcal{C}(A \setminus Y)$.*

The proof is elementary. This lemma is useful in the following proof.

Proof of Prop. 34: We have to prove that any x in $X \cup Y$ cannot be W-simple. If $Y = \emptyset$ then $X \cup Y = X$ which is a watershed. Suppose from now that $Y \neq \emptyset$.

Let $x \in Y$. Since $Y \subset A$ and $Y \neq \emptyset$ and Y is a watershed, there exists $B, C \in \mathcal{C}(\overline{A \setminus Y})$ which are adjacent to x and by Lem. 59, B and C also belong to $\mathcal{C}(\overline{X \cup Y})$, thus x is not W-simple for $X \cup Y$.

Let $x \in X$. Since X is a watershed for E and G is a perfect fusion graph, by Th. 33, X is thin and thus x is adjacent to exactly two elements B, C of $\mathcal{C}(\overline{X})$. If $B \neq A$ and $C \neq A$ then from Lem. 59 we deduce that x is also F-simple for $X \cup Y$, suppose now that $B = A$ (the case $C = A$ is identical). If $\Gamma^*(x) \cap Y = \emptyset$ then x is adjacent to C and to a component of $A \setminus Y$, it is thus not W-simple for $X \cup Y$. Suppose now that there exists $y \in \Gamma^*(x) \cap Y$. Since Y is a watershed for A there exists two points a, b in $\Gamma^*(y)$ which belong to distinct components of $A \setminus Y$ (thus, a and b are not adjacent). Furthermore, $y \in \Gamma(x) \cap \Gamma(a) \cap \Gamma(b)$ and since G is a perfect fusion graph and by the converse of Th. 41(*viii*), x must be adjacent to either a or b. Hence, x is not W-simple. □

6 Geodesic Saliency of Watershed Contours and Hierarchical Segmentation

L. Najman and M. Schmitt. Geodesic Saliency of Watershed Contours and Hierarchical Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18 (12), 1163-1173 (December 1996).

Geodesic Saliency of Watershed Contours and Hierarchical Segmentation

Laurent Najman and Michel Schmitt

Abstract—The watershed is one of the latest segmentation tools developed in mathematical morphology. In order to prevent its oversegmentation, the notion of dynamics of a minimum, based on geodesic reconstruction, has been proposed. In this paper, we extend the notion of dynamics to the contour arcs. This notion acts as a measure of the saliency of the contour. Contrary to the dynamics of minima, our concept reflects the extension and shape of the corresponding object in the image. This representation is also much more natural, because it is expressed in terms of partitions of the plane, i.e., segmentations. A hierarchical segmentation process is then derived, which gives a compact description of the image, containing all the segmentations one can obtain by the notion of dynamics, by means of a simple thresholding. Finally, efficient algorithms for computing the geodesic reconstruction as well as the dynamics of contours are presented.

Index Terms—Morphological segmentation, watershed, dynamics, hierarchical segmentation, geodesic reconstruction.

1 Introduction

Segmentation and contour extraction are key points of image analysis. There are numerous algorithms for doing these operations, which have the drawback of producing an oversegmentation. Several techniques have been developed to diminish this oversegmentation, the most common one being **hysteresis thresholding** by Canny [5]. Combined with the noise reduction induced by the Gaussian convolution, it has largely contributed to the success of Canny's extractor.

Mathematical morphology uses the **watershed** algorithm, introduced for the purpose of segmentation by Lantuéjoul and Beucher [3], and mathematically defined in [14], [16]. See [12] for a definition from a more algorithmical point of view. As the other techniques, the watershed produces an oversegmentation and until now, a procedure similar to "hysteresis thresholding" has not existed. The aim of this paper is to introduce such a procedure.

Watershed is often used in conjunction with **geodesic reconstruction**, a powerful tool developed by mathematical morphology, which simplifies gradient images and prevents oversegmentation. In this paper, we present a new algorithm which aims at computing in one step all the segmentations by watersheds that one can obtain by the use of geodesic reconstruction, or, equivalently, by the concept of **dynamics** [9]. The main advantage of our algorithm is that it directly gives a **hierarchical segmentation** in which all the contour arcs are evaluated by a measure of saliency (and not the catchment basins, as originally proposed by Grimaud), allowing one to choose the desired level of dynamics after the segmentation process. This measure of the saliency of the contour arcs will also be called "dynamics," because it is based on the same morphological tool, namely the geodesic reconstruction. The result can then be used in a similar way to the hysteresis thresholding, but in the context of region based segmentation.

This paper presents the state of the art around ideas of morphological segmentation, while mainly focusing on the geodesic reconstruction, and placing these ideas in a new unifying perspective which leads to the novel concept of dynamics of contours. It is organized as follows: First, we review the basic definition of the watershed. We then present the usual ways to reduce oversegmentation, which all rely on the geodesic reconstruction. Then we introduce the principle of hierarchical segmentation. Under this framework, we present a new concept of dynamics of contours, which allows a valuation of watershed contours relying on the gradient information and on the geodesic reconstruction. Then we discuss the interest of our concept, both from a mathematical and a practical point of view, illustrated by an application to shape recognition. Finally, we propose an algorithm to compute this hierarchical segmentation efficiently, together with a novel algorithm to compute the geodesic reconstruction.

2 The Watershed: A Tool for Segmentation

This section presents the standard definition of the watershed and can be skipped by the reader familiar with mathematical morphology (see for instance [21]).

In mathematical morphology, it is usual to consider that an image is a topographical surface. This is done by considering the gray level (the image intensity) as an altitude. Places of sharp changes in the intensity thus make a good set in which one can search for contour lines. It is then rather straightforward to estimate the variation from the gradient of the image. For the purpose of segmentation, we

- *L. Najman is with ArSciMed, 207 rue de Bercy, 75012 Paris, France. E-mail: lnajman@arscimed.com.*
- *M. Schmitt is with the Centre de Geostatistique, Ecole des Mines de Paris, 35, rue Sant-Honore 77305 Fontainebleau-Cedex, France. E-mail: mschmitt@cg.ensmp.fr.*

Manuscript received Nov. 10, 1994; revised Aug. 10, 1996. Recommended for acceptance by G. Medioni.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96084.

then look for the **crest lines of the gradient image**. A way to characterize these lines is to apply the watershed algorithm to the modulus of the gradient image.

The idea of the watershed [6], [3] is to attribute an **influence zone** to each of the **regional minima** of an image (connected plateau from which it is impossible to reach a point of lower gray level by an always descending path). We then define the watershed as the boundaries of these influence zones.

Numerous techniques have been proposed to compute the watershed. The major ones are reviewed in [23], [25]. The classical idea for building the watershed is illustrated in one dimension (Fig. 1). Using a geographical analogy, we begin by piercing the regional minima of the surface, then slowly immerse the image into a lake. The water progressively floods the basins corresponding to the various minima (Fig. 1a). To prevent the merging of two different waters originating from two different minima, we erect a dam between both lines (Fig. 1b). Once the surface is totally immersed, the set of the dams thus built is the watershed of the image. In one dimension, the location of the watershed is straightforward: it corresponds to the regional maxima of the function. In two dimensions (which is the case for gray-scale images), this characterization is not so easy. The place where two basins meet for the first time is a saddle point in the image. One can say in an informal way that the watershed is the set of **crest lines** of the image, emanating from the saddle points.

We present here the classical algorithm for computing the watershed, in the case of a function defined in $\mathbb{R}^2$ or on a digital grid, with discrete range (step functions). The most powerful implementation described in the literature ([25], [24], [22], [4]) uses FIFO breadth-first scanning techniques for the actual flooding.

Following the ideas mentioned above, the algorithm consists in flooding the various basins, and in keeping as the watershed the set of contact points between two different basins. In the case where this contact is on a plateau, we keep the (geodesic) middle line of this plateau. The watershed thus defined is of thickness one on the grid.

DEFINITION 2.1 *Let A be a set, a and b two points of A. We call* ***geodesic distance*** $d_A(a, b)$ *in A the lower bound of the length of the paths γ in A linking a and b.*

Let B be a set included in A. The geodesic distance $d_A(b, B)$ from a point b to the set B is defined as usual by $d_A(b, B) := \min_{c \in B} d_A(b, c)$.

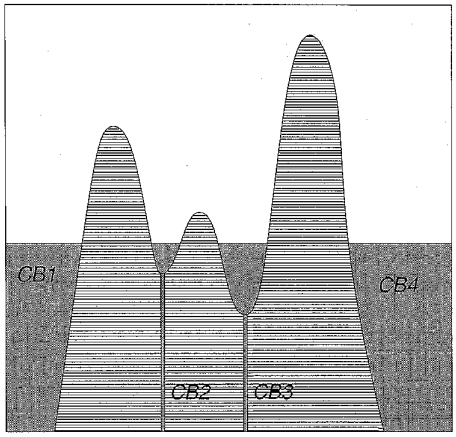
In the digital case, the distance d_A is deduced from the one on the grid [20].

Let $B = \cup B_i \subset A$, where B_i are the connected components of B.

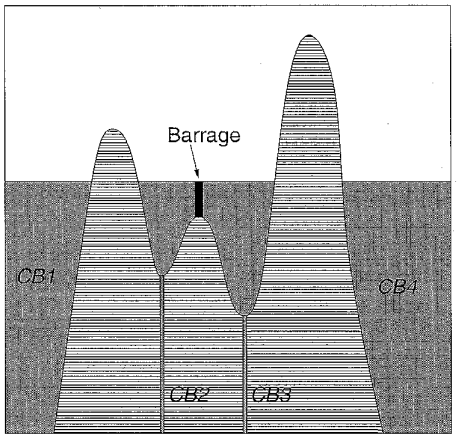
DEFINITION 2.2. *The* ***geodesic influence zone*** $iz_A(B_i)$ *of a connected component B_i of B in A is the set of the points of A for which the geodesic distance to B_i is smaller than the geodesic distance to other connected components of B.*

$$iz_A(B_i) = \{a \in A, \forall j \in [1, k] \backslash \{i\}, d_A(a, B_i) < d_A(a, B_j)\}. \quad (1)$$

The points of A which do not belong to any influence zone make up the skeleton by influence zone of B in A, denoted by $SKIZ_A(B)$:



(a) At time t, the dam is not yet constructed.



(b) At time $t + h$, we construct a dame to separate water from CB_2 and from CB_3

Fig. 1. Building of the watershed: one-dimensional approach.

$$SKIZ_A(B) = A \backslash IZ_A(B) \quad (2)$$

where $IZ_A(B) = \bigcup_{i \in [1,k]} iz_A(B_i)$.

The watershed algorithm on digital images by recurrence on the gray levels is [6]:

DEFINITION 2.3. *Let $f : \mathbb{R}^n \to \mathbb{N}$ be a bounded step function. We note*

1) $h_{min} = \min f$ *and* $h_{max} = \max f$,
2) $[f]^h$ *the upper threshold of f at level h :* $[f]^h := \{a \in \mathbb{R}^n \mid f(a) \leq h\}$,
3) $Reg\_Min_h(f)$ *the set of the regional minima of f at height h.*

The set of ***catchment basins*** *of f is the set $X_{h_{max}}$ obtained after the following recurrence:*

i) $X_{h_{min}} = f^{h_{min}}$

ii) $X_{h+1} = \mathrm{Re}\, g\_Min_{h+1}(f) \cup IZ_{[f]^{h+1}}(X_h)$, $\forall h \in [h_{min}, h_{max} - 1]$.

The ***watershed*** *of f is the complementary of X_{max}* (Fig. 2).

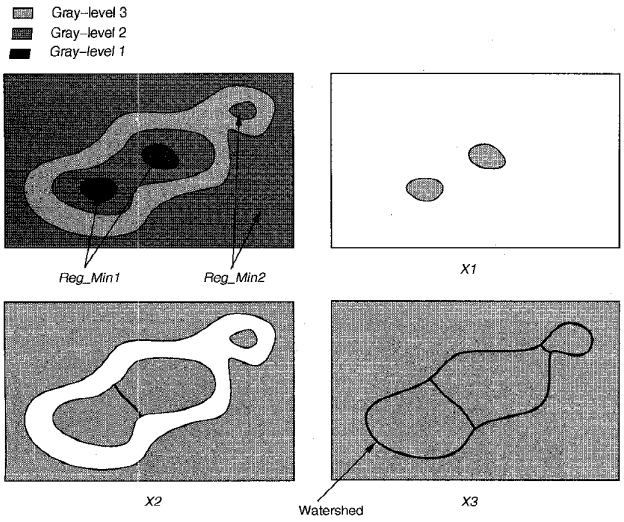


Fig. 2. Illustration of the recurrence immersion process.

3 Suppress the Oversegmentation

The watershed produces an oversegmentation of the images, but always contains contours which appear to be correct. The main problem is to make a choice between all those "right" contours. As in the case of Canny's extractor, the saliency of a contour can be evaluated by the value of the modulus of the gradient. But the step of hysteresis thresholding is not adapted to the watershed for three reasons:

1) Watershed produces a segmentation: Contours are obtained as complementary to the set of regions, and are consequently closed. Hysteresis thresholding is to be applied on edges and usually produces nonclosed contours. In other words, we start with a segmentation and get edges which do not necessarily build a segmentation.
2) Hysteresis thresholding on watershed segmentation produces barbs, which are small edges from adjacent regions (see Fig. 3). Only complicated algorithms could reliably eliminate these barbs.
3) The most important reason is probably that hysteresis thresholding is not a morphological process: It relies on the (local) neighborhood of the pixel, and not on the structure of the image. Hysteresis thresholding is well adapted to a local edge detector like Canny's one, but not to watershed segmentation, as it is the result of a process which is global to the image (one cannot construct the watershed segmentation only from local information).

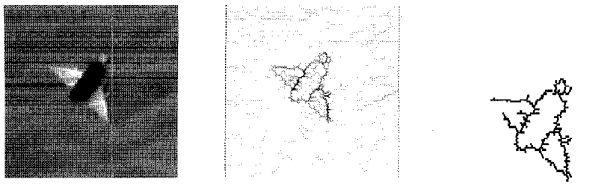


Fig. 3. A hysteresis thresholding on the watershed, valuated by the modulus of the gradient, yields non closed contours and small barbs.

In mathematical morphology, we choose the contours by smoothing the modulus of the image gradient $\| \nabla f \|$ with respect to various criteria. Indeed, we do not work directly on the watershed, but come back to the original information contained in the gradient image. For example, we can suppress some minima while preserving the position of the watershed (this is done by using some **markers**), or we can choose the contour by giving them a value which relies on the values of the gradient and on the watershed (this is what we call **hierarchical segmentation**).

All these morphological methods rely on an arbitrary flooding process, and the theory of morphological simplification of images can be deduced from a powerful tool: the geodesic reconstruction.

3.1 Geodesic Reconstruction

The geodesic reconstruction was originally developed by Beucher [2]. Let M and N be two closed sets of the plane. We denote by d_M the geodesic distance in M, i.e., the lower bound of the lengths of the paths in M linking a to b in M.

DEFINITION 3.1. *We call **geodesic dilation of infinite size** of N in M, or **geodesic reconstruction** of N in M, the (arc) d_M-connected components of M which contain at least one point of N. These components correspond to the points at finite d_M-distance of N. We denote this transformation by $D_M^\infty(N)$. The set N is called the **marker set**.*

On the lattice, the notation $D_M^\infty(N)$ is formally justified by the formula

$$D_M^\infty(N) = \left[(N \oplus B) \cap M\right]^\infty \tag{3}$$

where $\oplus$ denotes the morphological dilation and B the unit ball of the lattice.

With such a definition, the geodesic reconstruction is a binary transform. The simplest way to extend a binary transform to a gray-tone transform is to describe a function f with the help of its lower threshold $[f]\lambda := \{a \mid f(a) \geq \lambda\}$.

Consider two functions $f \leq g$, the **geodesic reconstruction** of f in g is defined through its lower thresholds:

$$\left[D_g^\infty(f)\right]_\lambda := D_{[g]_\lambda}^\infty\left([f]_\lambda\right) \tag{4}$$

As in the binary case, we call $D_g^\infty(f)$ **(gray-scale) reconstruction of f under g by dilation**. The function f is often called the **marker function** or **markers**. We focus our attention on the dual of the geodesic dilation of infinite size, the geodesic erosion of infinite size, or **(gray-scale) reconstruction of f over g by erosion**. The geodesic erosion behaves in a complementary way, that is to say we have to write the functions using their upper threshold $[f]^\lambda := \{x, f(x) \leq \lambda\}$. We then have the following formula:

$$\left[E_g^\infty(f)\right]^\lambda := D_{[g]^\lambda}^\infty\left([f]^\lambda\right) \tag{5}$$

Here, the marker function is g. Using this formula, we have derived a new algorithm for the geodesic erosion, which proceeds by flooding. It is described in the last section devoted to algorithms. This formula is fundamental for the understanding of the watershed. Eroding f over g is done

by flooding progressively the catchment basins of g while the meeting with f is not achieved (for more details, see Section 6.1.3). Thus we understand why the smoothing by geodesic erosion is so helpful for the segmentation by watershed: by calculating a geodesic reconstruction, we are implicitly constructing a watershed.

Before explaining how to use geodesic reconstruction, we need to state a theorem which has never been explicitly written, but which was implicitly used by all the authors.

THEOREM 3.2. *Let f and g be two functions from $\mathbb{R}^n$ to $\mathbb{R}$, $f \geq g$. Each regional minimum of the geodesic erosion $E_g^{\infty}(f)$ contains at least one regional minimum of f.*

That is to say, if $f \geq g$, the geodesic erosion $E_g^{\infty}(f)$ can only suppress or merge regional minima of f. As the main problem in watershed segmentation is to suppress spurious minima, we understand why geodesic erosion can be so helpful.

3.2 The Technique of Markers: A Geodesic Reconstruction by Erosion of the Marker Function Over the Gradient

The oversegmentation produced by the coarse application of the watershed is due to the fact that each regional minimum gives rise to a catchment basin. However, all the catchment basins do not have the same importance. There are important ones, but some of them are induced by the noise, others are minor structures in the image.

The first type of information one can extract is of a geometrical nature. Suppose we know a connected set of points belonging to an object (or a connected set for each object if there is more than one object to segment), and a set of points belonging to the background. We call these connected components **markers**. If we could modify the image on which to compute the watershed by imposing these sets as regional minima, we then obtain a watershed which has a loop around each object, as each catchment basin represents either the background or one unique object.

This is how to impose some regional minima M on an image f. We construct the image:

$$g(x) = \begin{cases} +\infty & \text{if } x \notin M \\ 0 & \text{if } x \in M \end{cases} \tag{6}$$

This image has the regional minima we want. To keep the information of the original image and to put the watershed on the plateau at height ∞ of g, we geodesically erode g on function $f \wedge g$, i.e., we compute $E_{f \wedge g}^{\infty}(g)$. Here, we denote $f \wedge g(a) := \min(f(a), g(a))$. This image has the same minima as g, for the geodesic erosion does not add any new minima (Theorem 3.2). Furthermore, all the pixels which are sufficiently high and not in an unselected regional minimum are the same. We can then apply the watershed algorithm on this new image. Fig. 4 illustrates this method and Fig. 5 shows the results on the image of a cook-stove. Note that we can choose any contrast images: here we have replaced the usual gradient modulus by a top-hat transformation $f - f_B$, where $f_B = (f \ominus B) \oplus B$.

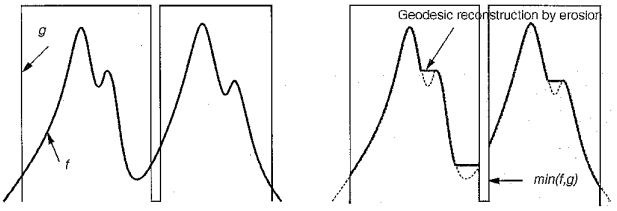


Fig. 4. Illustration of the way to impose regional minima of g on the image f.

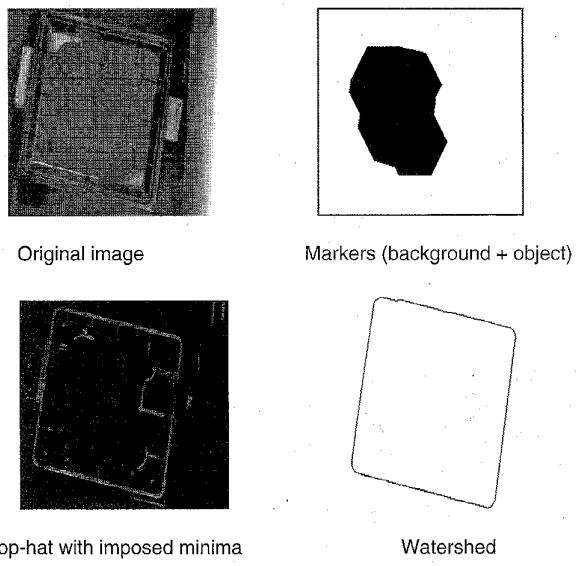


Fig. 5. Constrained watershed: markers are imposed as regional minima of the top-hat transformation.

We did not explain how to choose the markers. It is, in general, the most complex part. The technique of constrained watershed allows us to look for the contour of the objects with less exactitude and guarantees the number of contours found: one around each marked object. All the difficulty lies in determining the markers, i.e., to a rough localization of the objects.

In brief:

Segmentation by constrained watershed

1) Find the markers, i.e., one connected component for each object and one connected component for the background.
2) Compute the image on which the watershed will be constructed (usually a contrast image like the modulus of the gradient).
3) Impose the minima by gray-scale geodesic reconstruction.
4) Compute the watershed.

3.3 The Technique of Minima Dynamics: Geodesic Reconstruction by Erosion of f_t over f

The **dynamics** of a regional minimum is a contrast criterion. Recall that a regional minimum is a connected set from which it is impossible to reach a point with a lower height without climbing. The minimal height of this climbing is the valuation of the contrast of the regional minimum.

DEFINITION 3.3. *Let M be a regional minimum of the function f. The* ***dynamics*** [9], [10] *of M is the number*

$$\min\left\{\max_{s\in[0,1]}\left\{f(\gamma(s)) - f(\gamma(0))\right\} \middle| \gamma : [0, 1] \to \mathbb{R}^2,\ f(\gamma(1)) < f(\gamma(0)), \gamma(0) \in M\right\}$$

where γ is a path linking two points.

One can notice that the dynamics is not defined for the global minimum of the image. In practice, however, the image f has a compact domain of definition, and we can always suppose the global minimum is on the boundary of this image, which allows the valuation of the global minimum inside the domain of definition of f.

The concept of dynamics is illustrated in Fig. 6. It can be used to find relevant markers: the minima with a great dynamics. Let us notice that in practice we do not impose these minima by geodesic reconstruction of a marked function. On the contrary, we suppress the regional minima of f with a dynamics lower than a given contrast value t. The standard algorithm to do this operation is to compute the geodesic reconstruction by erosion $E_f^{\infty}(f_t)$ of f_t over f where $f_t(a) = f(a) + t$ (Fig. 7).

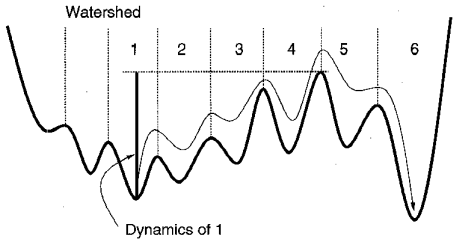


Fig. 6. Illustration of the concept of dynamics.

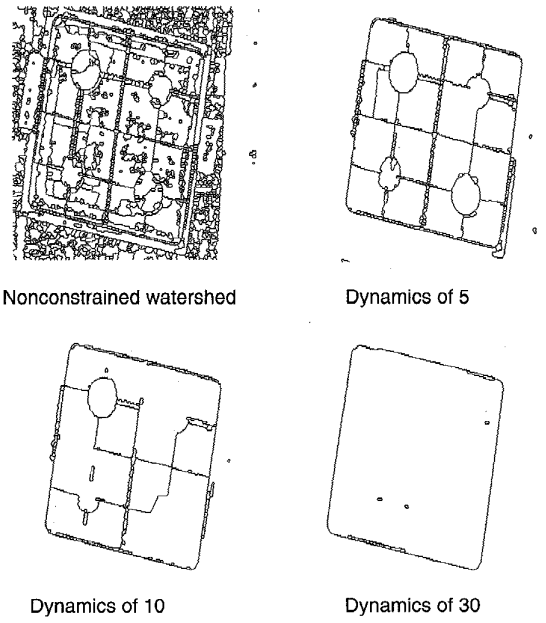


Fig. 7. Watershed constrained by a contrast criterion: the dynamics.

Let us notice that the size and location of the minima with a given dynamics is of little direct use: their catchment basins do not reflect the final segmentation with this level of dynamics (see, later, Fig. 10c). In what follows, we shall explain how to use all the information contained in the map of basins valuated by their dynamics.

The following section is the heart of the paper, in which we create a saliency map from the watershed segmentation.

4 HIERARCHICAL SEGMENTATION

Until now, our aim has been to prevent the oversegmentation by choosing markers and, using homotopy modification, to produce as many catchment basins as there are objects in the image. In this section, we present the notion of **hierarchical segmentation**, originally developed by Beucher [2], which, rather than preventing the oversegmentation, computes the importance of the contours with respect to given criteria.

Let us first mathematically define what we mean by a hierarchy.

DEFINITION 4.1. *Let $\mathcal{P}_{h_i}$ be a sequence of partitions of the plane. The family $(\mathcal{P}_{h_i})_i$ is called a* ***hierarchy*** *if $h_i \geq h_j$ implies $\mathcal{P}_{h_i} \supseteq \mathcal{P}_{h_j}$, i.e., any region of partition $\mathcal{P}_{h_i}$ is a disjoint union of regions of partition $\mathcal{P}_{h_j}$.*

Every hierarchy can be assigned a **saliency map**, by valuating each point of the plane by the highest value h such that it appears in the boundaries of partition $\mathcal{P}_h$. If we interpret these partitions as segmentations, we have a nice way of assigning importance to the contours. The problem is to obtain such a family of segmentations.

4.1 Beucher's Hierarchical Segmentation: The Waterfall Algorithm

In segmentation with the help of markers, the final result strongly depends on the first stage of marker determination. But we point out that marker determination is not an easy process. Images are often noisy, and the objects we want to detect are often complex and varied in shape, size or intensity. Now, when we look at the result of a watershed segmentation, we notice that a lot of apparently homogeneous regions are shattered into small pieces. A natural idea is then to try to merge these regions. Mathematical morphology suggests a solution to make this fusion, **hierarchical segmentation** which was introduced in this context by Beucher [2] and Beucher and Meyer [4].

This fusion is done by automatically selecting some markers, using a procedure called the **waterfall algorithm** which relies on geodesic reconstruction by erosion. Let us build a new function g by setting $g(x) = f(x)$ if x belongs to the watershed and $g(x) = +\infty$ if not. This function g is obviously greater than f. Let us now reconstruct f over g. It is easy to see that the minima of the resulting image (Fig. 8) are significant markers of the original image.

Some remarks should be made here. First of all, even if this procedure allows the construction of a hierarchy by repeating itself until convergence, it does not allow a valuation of the contours thus obtained: The convergence is

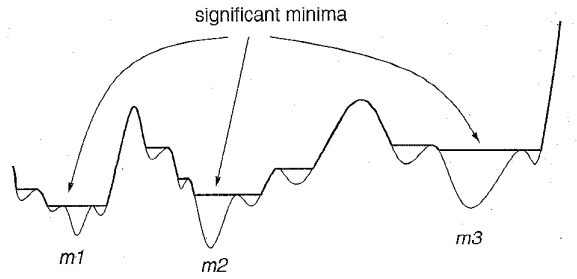


Fig. 8. Reconstruction from watershed lines and detection of the significant markers.

usually very fast, and only a few levels of hierarchy are present in the result (typically half a dozen). Second, even if we value the hierarchy in this way, the final valuation result does not rely on some gradient information: it is only a way of giving a partial order relation on the various minima. A example of such a valuation is given in Fig. 10b.

4.2 Hierarchical Segmentation Using Dynamics

The dynamics of minima notion presented above allows the creation of another hierarchical concept which, by relying on the minima dynamics concept, gives birth to the new concept of dynamics of contour. Let us consider an image f. If we suppose that two different minima and two different saddle points are not on the same gray level (which does not pose any problem in practice), the geodesic reconstruction does not move the contour obtained by watershed. It can only suppress some contours. It is then sufficient to valuate each arc of the watershed with the maximal value of t for which the arc belongs to the watershed of $E_f^{\infty}(f_t)$. It is easy to see that this depends only on the lowest saddle point on the arcs which separate the two basins. Let a be the (saddle) point of lower altitude on these arcs, we define

$$Bas(a) := \{b \mid \exists \gamma, \gamma(0) = a, \gamma(1) = b, f(\gamma(s)) < f(a)\ \forall s \in]0, 1]\} \quad (7)$$

The set $Bas(a)$ is a topological open set, and can be divided in several open connected components B_i ($Bas(a) = \cup_i B_i$). We set

$$dyn(a) := \min_i \max_{a_i \in B_i} \{f(a) - f(a_i)\} \quad (8)$$

We then valuate arcs which separate two basins by the number $dyn(a)$, which we call **dynamics of contour**. The saliency map dyn which associates at each point a its contour dynamics is then given by the formula

$$dyn(a) = \int_0^{+\infty} \chi_{WS(E_f^{\infty}(f_t))}(a)dt \quad (9)$$

where $\chi_{WS(f)}(a)$ is the value at point a of characteristic function[1] of the watershed of f.

This valuation is much more natural than the valuation obtained by the waterfall algorithm, for it relies on the gradient information.

In the last part of this paper, we give an algorithm which directly computes the watershed with this valuation.

1. Which is equal to 1 if a belongs to the watershed and to 0 if a does not belong to the watershed.

It is worth noting that contrary to the noise sensitivity of the dynamics of a basin, the dynamics of a contour is much more stable (yet the contour itself is sensitive to noise). This is illustrated in Fig. 9.

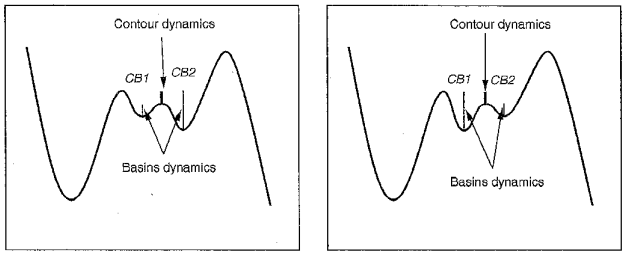
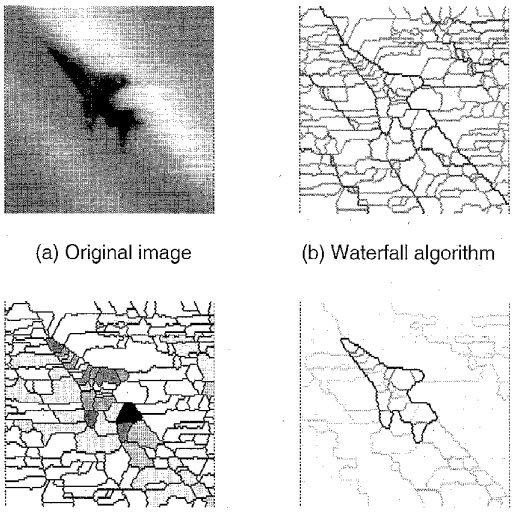


Fig. 9. Between right and left figure, basins CB_1 and CB_2 have exchanged their dynamics. But the dynamics of the contour which separates these basins remains unchanged.

Fig. 10 shows the difference between a simple computation of dynamics (Fig. 10c) and an application of the contour valuation algorithm. The basins of high dynamics do not reflect the extension and shape of the region which could be obtained by keeping only the regional minima of high dynamics. So, the dynamics of basins represent only an intermediate result. Contrary to this, let us notice that the result of our algorithm (Fig. 10d.) gives all the contour information we can extract from the gradient image, that is to say, a threshold of the result image at a given level will give the segmentation which will be obtained by a geodesic reconstruction of the same level. The only way to obtain other contours is to add exterior information, either by the use of markers, either by using other contour extractors (like Canny's one) which we combine to the watershed by use of a watershed algorithm with anchor points [16], [15], [14].



(a) Original image (b) Waterfall algorithm

(c) Map of the basins valuated by their dynamics (white = low, black = high) (d) Computation of the dynamics of contours

Fig. 10. Difference between an application of the waterfall algorithm, a computation of dynamics of basins and a computation of the dynamics of contours.

5 Application to Shape Recognition

In this section, we discuss the interest of our new segmentation process. This section is based first on mathematical criteria used in contour detection, then on the robustness of the arc saliency with respect to noise and finally on its usefulness in image segmentation, especially for the problem of guessing "what are the n most important objects in an image."

5.1 Mathematical Arguments

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a smooth function, and set $f_n(a) := \frac{E(2^n f(a))}{2^n f(a)}$, where $E(x)$ stands for the integer part of x. It has been shown [14], [16] that under adequate hypothesis, the limit of watershed of the f_n is a subset of the maximal integral lines of the gradient of f (lines of steepest slope on f) linking some particular critical points of f (where $\nabla f(a) = 0$), typically a subset of saddle points and of maxima of f. We can apply this result to show why the watershed is a good edge detector.

Let g be the modulus of the gradient of an image f: $g(a) := \|\nabla f(a)\|$. One can say that an edge is a path where the change in the intensity of f is maximum in the direction normal to this path. As the intensity is computed by the modulus of the gradient, we can write

$$\frac{d}{dt} g(a + t n) = \langle H_f(a) \nabla f, n \rangle = 0 \qquad (10)$$

where n is the normal to the path at point a, and where $H_f(a)$ is the Hessian (the matrix of the second derivatives) of f at a. This equation gives an implicitly differential equation for the edge path γ:

$$\dot{\gamma} = H_f \nabla f. \qquad (11)$$

This equation is not sufficient to characterize edges, because on any point in the plane where $\nabla f \neq 0$, there exists a path γ satisfying (11). The union of all the paths which are solutions to this differential equation covers the whole domain of f. It has been proved [16], [14] that the watershed chooses the paths by imposing boundary conditions.

The boundary conditions imposed by the watershed are such that it is possible to join the end points of the path $\dot{\gamma} = H_f \nabla f$ to two different minima by two different always descending paths. So the watershed produces closed contours, and finds exactly multiple points (intersection of contours) which are of great importance in image analysis.

On the other hand, classical edge detectors like Canny's[2] [5] solve the problem by **estimating** the normal n from the gradient direction, i.e., by setting $n = \nabla f$ (which is true on a step edge, but not true on more complicated shapes and especially at locations where many contours meet).

Finally, note that the watershed does not need the complete gradient of f (∇f), but only its modulus ($\| \nabla f \|$). This feature is very important near contour junctions, where the direction of the gradient computed with Canny's method is unreliable, yielding very few triple points in its contour images. Some complex modifications have to be made in order to detect the corner points correctly [7], [8]. On the contrary, the watershed, using only the modulus of the gradient, (which is reliable) gives triple points, accurately positioned. This feature has been successfully applied for corner detection in [17].

5.2 Noise Robustness

One kind of performance evaluation for a segmentation algorithm is its small dependence to noise. Fig. 11 shows a synthetic image with its watershed computed directly on it. Due to the noise free image, all the watershed lines have the same saliency and the four square shaped catchment basins are very large. When adding a small negative Poisson noise, we observe the birth of many small catchment basins created by one pixel large regional minima corresponding to negative noise. The noise has been chosen negative, because at each modified point, it creates a small regional minimum and, consequently, a new catchment basin. However, the new added watershed lines have a very low saliency, whereas the most contrasted watershed lines are very little displaced. The small displacements we observe correspond to noise pixels falling exactly on the watershed line. Also, the junction point at the center of the image, where the four major contours meet, is always preserved. Increasing the noise values does not add new basins, but the saliency of their contours increases. Note however that the most salient

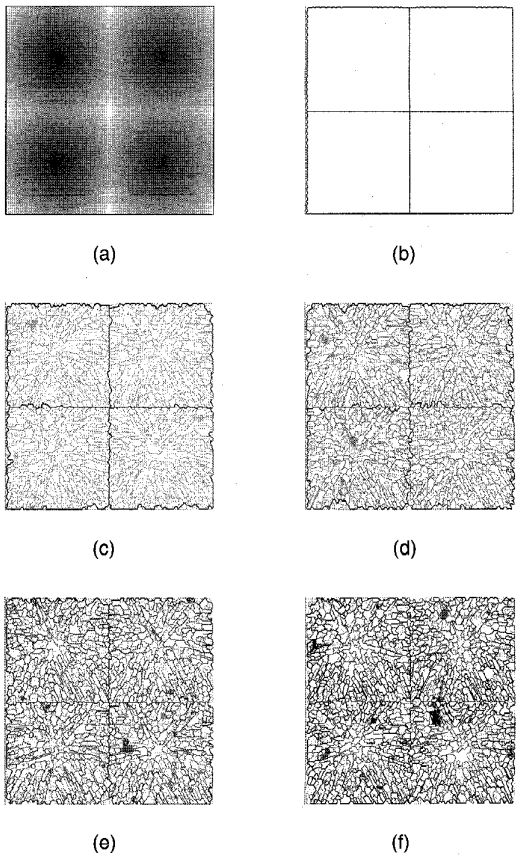


Fig. 11. Robustness of the dynamics of contours with respect to noise. (a) original noise free image, (b) its watershed, (c) to (f) increasing negative Poisson noise.

2. Canny's detector, or more exactly the extrema of the gradient in the direction of the gradient, finds the zero crossings of $Q(f) = \langle H_f \nabla f, \nabla f \rangle$.

contours, namely the desired contours, are still very little affected. Finally, when the noise completely overwhelms the signal (noise values similar to the signal values), the most salient contours no longer really exist. In this case, optimal linear smoothing techniques should be applied prior to watershed extraction.

5.3 Real Image Segmentation

Let us now illustrate the concept of hierarchy on real images. Fig. 12 shows some indoor scene, where some of the structures are very contrasted, whereas others are much less so. These structures can be recognized on the saliency map. The very low salient contours correspond to noise. Note however that some structures like the upper part of the desk exhibit a lower saliency than expected. This is due to the criterion implemented by the watershed: the saliency is governed by the altitude of the saddle point with respect to the two neighboring catchment basins. The altitude of this saddle point corresponds to a minimum on the contour line. So, as soon as some parts of the contour are less contrasted, even for a few pixels (here at the left of the lamp), the lower contrast value is propagated along the whole object contour. In particular, objects which exhibit a low contrasted side are surrounded by a watershed line of low saliency. This is due to the fact that the watershed always closes the contours, yielding a region based segmentation.

Another example is presented in Fig. 13.

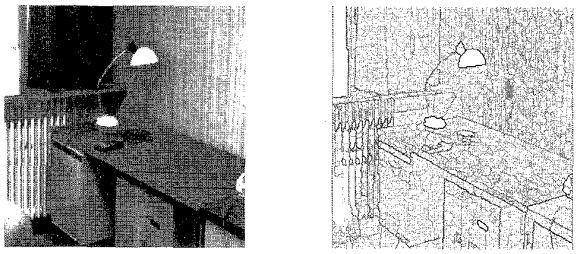


Fig. 12. Indoor scene and the saliency map obtained by the watershed of the modulus of its gradient.

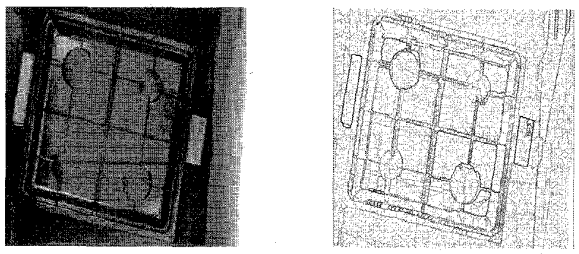


Fig. 13. Another example of the saliency map computed on an image of a cook stove.

5.4 Finding n Objects

In this section, we give an example showing how to use the dynamic hierarchical segmentation algorithm for shape recognition.

Fig. 14 is a snapshot of an airplane. For military systems, one of the key problems is airplane identification and attitude estimation. With this in mind, we may construct a three-dimensional model of the plane, which will be compared to a database of three-dimensional models. More simply, we may compare only the contour of the airplane to a contour database, by extracting a small number of parameters which hopefully reflect the relevant features of the shape [18], [19].

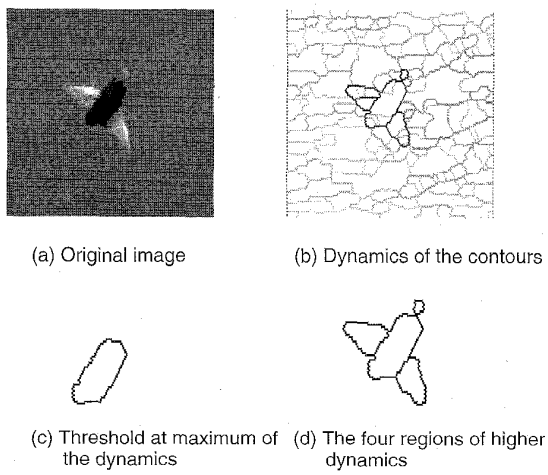


Fig. 14. Some thresholds of the contour dynamics image.

So, the first stage consists of finding the airplane contours. Here the use of markers is very difficult: To obtain the whole plane correctly, including the wings, the marker should have the shape of the plane and, as a marker, it should be included in the plane. In the same way, the use of the highest dynamics will only extract one region in some cases, and we will obtain only the airplane body (Fig. 14c). We propose the use of the dynamic hierarchical segmentation to extract the right contours.

The image of the contour dynamics (Fig. 14b) clearly shows that all the interesting contours are present in the watershed image. One has to introduce additional information for extracting these right contours. For instance, if we choose the object surface, the image of the contour dynamics will be thresholded at a height corresponding to n catchment basins of the expected surface. One can also consider applying criteria for region growing [13], the initial step being a threshold of the image of the contour dynamics.

However, finding the airplane contour directly is futile. On the other hand, one can guarantee that, in a small number of hypotheses, we will find this right segmentation. This is done by successively merging regions to construct the airplane shape. The algorithm is a good procedure for giving these hypotheses (four regions, Fig. 14d) and one can identify the airplane.

In the same way, in the context of interactive segmentation, the hierarchical saliency map can be computed at once. Then, the adjustment of the unique threshold, done manually, allows a human operator to explore the various possible segmentations.

6 Algorithms

6.1 Algorithms for Geodesic Reconstruction

6.1.1 Sequential Reconstruction

There exists a sequential algorithm [11], [23], [24] for geodesic reconstruction which works by propagating information downward in a raster scanning and then upward in an antiraster scanning. These raster and antiraster scannings have to be iterated until stability is reached (usually no more than ten complete image scannings). It is then very fast.

6.1.2 Beucher-Meyer Algorithm

The principle [4] is simple. The regional minima of $g > f$ are given as input. Starting from these minima, g is eroded progressively, while staying above f. The implementation of this algorithm is easy with an ordered queue [4] or with a heap-sort algorithm [1] (these two algorithms create a queue which stays ordered when a new element is introduced): The pixels are examined by increasing gray level of g. Graphically speaking, the function g acts as a film which contracts on a parcel which is the function f. The algorithm is optimal in the sense that each point is processed only once.

6.1.3 New Algorithm

We now present a new algorithm for geodesic reconstruction which "dilates" f under g, and proceeds by flooding. It is optimal in the same way as the Beucher-Meyer algorithm presented above: Each point is processed only once. But more important, it does not need the regional minima as an input. The algorithm computes the regional minima *during* the flooding, as in the classical watershed algorithm [25].

The basic idea is to use the flooding principle developed by Vincent [23]. This principle is adapted to the geodesic reconstruction of f under g, which corresponds to the geodesic erosion of g with respect to f.

In fact, we can reconstruct f by flooding the catchment basins of f until they overflow, or until we meet g. When we flood the catchment basins of f, two cases can appear.

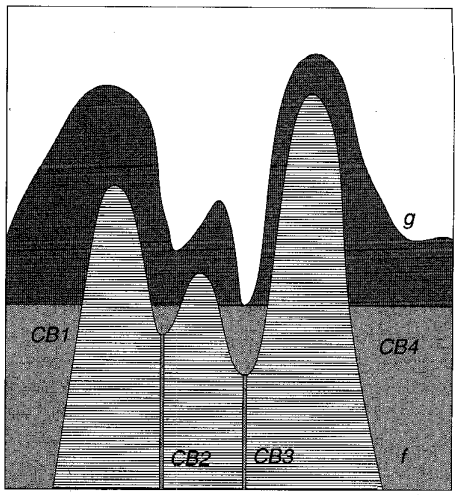
- Either the minimum of g on the basin is equal to the height of flooding we have reached. In this case, we stop flooding this basin (basin CB_3 in Fig. 15a).
- Either we have filled the basin until one of its saddle points (contact point between two basins) is reached. This case is divided into two branches:
  - i) either the other basin has already stopped being flooded. We then stop flooding the considered basin at the height of the flooding (basin CB_2 in Fig. 15b);
  - ii) either the other basin has not yet been stopped. We merge the two basins.

The watershed by flooding algorithm is easy to adapt to this procedure which is fundamental for the computation of constrained watersheds and for the computation of regional maxima under constraints [22]. The advantage of this algorithm with respect to the one of Beucher-Meyer is that the new algorithm computes the regional minima directly.

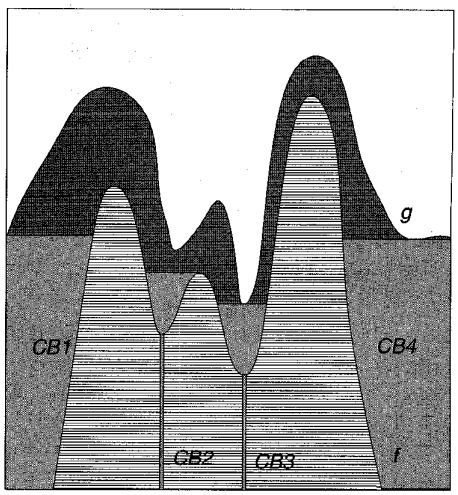
6.2 Algorithm of Contour Valuation

We now give an algorithm which computes the contour valuation. During a first stage, we use Grimaud's algorithm [9] to compute the watershed, the catchment basins and their dynamics. We then compute, for each point of the watershed, a valuation by doing a kind of "gradient descent" on the dynamics value of the catchment basins. It is difficult to compute the valuation of the point during the watershed construction, because when a point of the watershed is created, we do not dispose of the complete list of its neighbors, and in particular triple points can end with a wrong value. In fact, only one point on each arc of the watershed is of interest: the saddle point.

Let us briefly recall Grimaud's dynamics algorithm which is based on Vincent's watershed algorithm. It consists in flooding from the minima, level by level, until water from one minimum meets water from another minimum. The meeting point between two basins is a saddle point, and is the point where we can compute the dynamics of one of the two basins: the basin with the lowest minimum floods the other one, and the dynamics of the basin with the highest minimum is equal to the gray-value of the saddle point minus the gray-value of the minima.

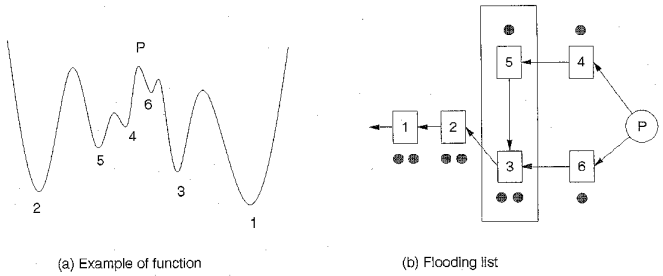


(a) At time t, the catchment basin CB_3 stops to be flooded.



(b) Final stage of the reconstruction.

Fig. 15. Geodesic reconstruction of f under g.



(a) Example of function (b) Flooding list

Fig. 16. Determination of the basin which valuates the contour at which the pixel P belongs.

In our algorithm, the arc of the watershed containing this saddle point is valued by this dynamics. Let us notice that, in Grimaud's algorithm, an arbitrary choice has to be made if the two meeting basins have minima with the same gray-value, but this choice *does not change the value of the arc*. This is why we think that the dynamics of an arc is a better notion than the dynamics of a catchment basin.

We now show how we can value the watershed arc. The best way to explain the algorithm is to look at an example (Fig. 16). Suppose we have applied Grimaud's algorithm on the function of Fig. 16a. We then have a list of watershed pixels, and a list of catchment basins. During the flooding process, we construct a list of catchment basins. Each catchment basin keeps in memory:

- Its dynamics,
- A list of pointers on the catchment basins which have flooded it.

Let's have a look at a pixel p of the watershed (Fig. 16a). This pixel is the connection point between two (or more, three at most on the hexagonal grid) catchment basins. In our example, they are catchment basins 6 and 4.

But basin 6 has been flooded by basin 5, and basin 5 has been flooded by basin 3, which itself has been flooded by basin 2. The basin which has flooded all the other basins is the one with the lowest minimum, that is to say basin 1. In the other way, basin 6 has been flooded by basin 3. This flooding list is represented in Fig. 16b.

The pixel p is in fact flooded when basins 3 and 5 meet and the dynamics of the contour can be computed. Another way to choose the dynamics of the pixel p is to notice that pixel p belongs to the interior of basin 3. So we can say that the dynamics of pixel p is the highest dynamics of the basins which precede basin 3 in the flooding list. So, all the difficulty is to mark out basin 3. We propose a simple procedure for doing that operation. It consists of running through the whole flooding list issued from the pixel p, and in marking the basin by which we pass. The first basin which has been passed over more than once contains the pixel p in its interior, and is basin 3 in our example. So the dynamics of the pixel p is the highest dynamics between the dynamics of basins 5 and 6 (which precede basin 3 in the flooding list), that is to say the dynamics of basin 5.

One could verify that the watershed arc which contains pixel p disappears in a geodesic reconstruction of size equal to the dynamics of basin 5.

7 Conclusion

Image segmentation is not a goal in itself. It should be done by keeping in mind the real purpose of the image processing. We have given an algorithm which is useful for interactive dynamics thresholding. The concept of dynamics of contours allows the valuation of the contours produced by the watershed. This resulting segmentation is *identical* to the one obtained by watershedding the original gradient image after a filtering by reconstruction with the same contrast value. The advantage of the proposed algorithm is this : the dynamic segmentation can be obtained for different contrast values by simply thresholding the valuated watershed image. For instance it enables us to answer the question "which are the n most contrasted objects?" by a simple thresholding of the image (a problem which can be solved in a more complicated way by examining the histogram of dynamics of the image minima).

An example of image processing is **identification**. It is almost impossible to obtain the right segmentation for identification using only segmentation tools based on contrast. But we can guarantee that, in a small number of hypotheses, we will find this right segmentation, which can be extracted by artificial intelligence tools (using for instance a database). We expect our algorithm to be a good procedure for giving these hypotheses.

8 Notations

a, b	points in the $\mathbb{R}^2$ plane
f, g	images, i.e., functions from $\mathbb{R}^2$ to $\mathbb{R}$
∇f	gradient of f, i.e., vector of first derivatives
H_f	Hessian of f, *i.e.*, matrix of second derivatives
$\langle ., . \rangle$	inner product
$f \wedge g$	pointwise minimum of f and g
$f \vee g$	pointwise maximum of f and g
$[f]^h$	the upper threshold of f at level h, i.e., $\{a \in \mathbb{R}^n \mid f(a) \leq h\}$
$[f]_h$	the lower threshold of f at level h, i.e., $\{a \in \mathbb{R}^n \mid f(a) \geq h\}$
$f \oplus B$	dilation, i.e., $\max\{f(y) \mid y \in B_x\}$
$f \ominus B$	erosion, i.e., $\min\{f(y) \mid y \in B_x\}$
$D_g^\infty(f)$	geodesic reconstruction of f into g by dilation ($f \leq g$)
$E_g^\infty(f)$	geodesic reconstruction of f over g by erosion ($f \geq g$)
$d_M(a, b)$	geodesic distance in M (Definition 2.1)
$SKIZ_A(B)$	skeleton by influence zones of B according to the geodesic distance in A (Definition 2.2)
$IZ_A(B)$	influence zones of B according to the geodesic distance in A (Definition 2.2)
$Reg\_Min_h(f)$	regional minima of f *i.e.*, connected plateau from which it is impossible to reach a point of lower gray level by an always descending path

REFERENCES

[1] A. Aho, J. Hopcroft, and J. Ullman, *Data Structures and Algorithms*. Reading, Mass: Addison-Wesley, 1983.

[2] S. Beucher, "Segmentation d'images et morphologie mathématique," thèse, École Nationale Supérieure des Mines de Paris, June 1990.

[3] S. Beucher and C. Lantuéjoul, "Use of Watersheds in Contour Detection," *Proc. Int'l Workshop Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, Sept. 17-21, 1979.

[4] S. Beucher and F. Meyer, "The Morphological Approach to Segmentation: The Watershed Transformation," *Mathematical Morphology in Image Processing*, E.R. Dougherty, ed., Optical engineering, pp. 433-482. New York, Basel, Hong Kong: Marcel Dekker, 1993.

[5] J.F. Canny, "A Computational Approach to Edge Detection," *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, M.A. Fischler and O. Firschein, eds., pp. 184-203. Morgan Kaufmann, 1986.

[6] H. Digabel and C. Lantuéjoul, "Iterative Algorithms," *Proc. Second European Symp. Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, J.-L. Chermant, ed., pp. 85-99. Stuttgart, Germany: Riederer Verlag, 1978.

[7] G. Giraudon and R. Deriche, "Accurate Corner Detection: An Analytical Study," Technical Report 1420, INRIA, Apr. 1991.

[8] G. Giraudon and R. Deriche, "On Corner and Vertex Detection," Technical Report 1439, INRIA, June 1991.

[9] M. Grimau, "La Géodésie Numérique en Morphologie Mathématique: Application à la Détection Automatique de Microcalcifications en Mammographie Numérique," thesis, École des Mines de Paris, Dec. 1991.

[10] M. Grimaud, "A New Measure of Contrast: Dynamics," *SPIE Vol. 1769, Image Algebra and Morphological Processing III*, pp. 292-305, San Diego, July 1992.

[11] B. Läy, "Recursive Algorithms in Mathematical Morphology," *Acta Stereologica*, pp. 691-696, Caen, France, 1987, vol. 6/III, *Proc. Seventh Int'l Congress Stereology*.

[12] F. Meyer, "Topographic Distance and Watershed Lines," *Signal Processing*, special issue on mathematical morphology, vol. 38, no. 1, pp. 113-126, July 1996.

[13] O. Monga, "An Optimal Region Growing Algorithm for Image Segmentation," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 3, no. 4, Dec. 1987.

[14] L. Najman, "Morphologie Mathématique: de la Segmentation d'Images à l'Analyse Multivoque," thèse de doctorat, Université Paris-Dauphine, Paris, France, Apr. 1994.

[15] L. Najman and M. Schmitt, "La Ligne de Partage des eaux: applications d'une approche continue," *Revue Technique Thomson*, vol. 25, no. 2, pp. 261-280, Mar. 1993.

[16] L. Najman and M. Schmitt, "Watershed of a Continuous Function," *Signal Processing*, special issue on mathematical morphology, vol. 38, no. 1, pp. 99-112, July 1994.

[17] L. Najman and R. Vaillant, "Topological and Geometrical Corners by Watershed," *CAIP '95 Proc.*, Hlavác and Sára, eds., LNCS 970. Springer-Verlag, 1995.

[18] L. Najman, R. Vaillant, and É. Pernot, "From Face Sideviews to Identification," *Revue Technique Thomson*, vol. 24, no. 4, pp. 1,037-1,054, Dec. 1992.

[19] L. Najman, R. Vaillant, and É. Pernot, "From Face Sideviews to Identification," *Image Processing: Theory and Applications, An Int'l Conf.*, G. Vemazza, ed., pp. 299-302, San Remo, Italy, June 1993.

[20] M. Schmitt, "Geodesic Arcs in Non-Euclidean Metrics: Application to the Propagation Function," *Revue d'Intelligence Artificielle*, vol. 3, no. 2, pp. 43-76, 1989.

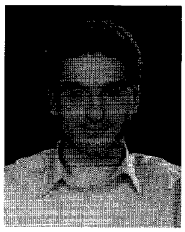
[21] M. Schmitt and J. Matioli, *Morphologie Mathématique*. Logique—Mathématiques—Informatique. Masson, Dec. 1993.

[22] M. Schmitt and L. Vincent, *Morphological Image Analysis: A Practical and Algorithmic Handbook*. Cambridge Univ. Press, to appear in 1996.

[23] L. Vincent, "Algorithmes Morphologiques à Base de Files d'Attente et de Lacets: Extension aux Graphes," thèse, École des Mines, Paris, May 1990.

[24] L. Vincent, "Morphological Algorithms," *Mathematical Mrophology in Image Processing*, E.R. Dougherty, ed., Optical engineering, pp. 255-288. New York,Basel, Hong Kong: Marcel Dekker, 1993.

[25] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583-598, June 1991.



Laurent Najman received his Maîtrise de Mathématiques Appliquées and his Maîtrise d'Ingénierie Mathématique in 1989 from the University of Paris VI, his engineering degree from the Ecole des Mines de Paris in 1991, and the Docteur en Sciences degree from the Paris-Dauphine university. From 1991 to 1994, he was in the Signal Analysis and Pattern Recognition Laboratory at the Central Research Laboratory of Thomson-CSF, where he worked in collaboration with the Viability Team of the CEREMADE at Paris-Dauphine University on some problems of image segmentation using mathematical morphology and set valued analysis. He joined ArSciMed in 1995, a start-up company devoted to the development of particle systems technology, where he is director of research and development. His current research interest is simulation using particle systems, with applications to computer graphics and vision.

Michel Schmitt received his engineer degree from the Ecole Polytechnique in 1982, his engineer of the Corps des Mines degree in 1985. He joined the Mathematical Morpholgical Laboratory of J. Serra at the Ecole des Mines (1985) where he began a thesis on mathematical morphology: theoretical aspects, algorithms, and artificial intelligence. He joined the team of O.D. Faugeras at INRIA in 1985, carrying on his research work, obtained his Docteur en Morphologie Mathématique degree from the Ecole des Mines in 1989 and the Habilitation à diriger des recherches degree in 1991. From 1989 to 1995, he worked at the Central Research Laboratory of Thomson-CSF as head of the Signal Analysis and Pattern Recognition Laboratory. Since 1995, he has been the director of the center for Geostatistics at the Ecole des Mines de Paris. His current research interests are image processing and geostatistics.

7 Euler Method for Mutational Equations

L. Najman. Euler Method for Mutational Equations. *Journal of Mathematical Analysis and Applications* 196, 814-822 (1995)

JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS **196,** 814–822 (1995)

Euler Method for Mutational Equations

LAURENT NAJMAN*

L.C.R, Thomson-CSF, Domaine de Corbeville, 91404 Orsay Cedex, France; and CEREMADE, Université Paris Dauphine, 75775 Paris Cedex, France

Submitted by Hélène Frankowska

Received September 19, 1994

We adapt the Euler theorem for approximating solutions to differential equations to the new mathematical framework of mutational equations, which aims to compute derivatives of shape deformation in metric spaces.

1. INTRODUCTION

Mutational equations [1, 2] are an extension of differential calculus for maps from one metric space to another. Basic theorems of differential equations (such as the Cauchy–Lipschitz and Peano equations) can be adapted to this mathematical framework.

Mutational equations seem to be a relevant tool for describing the dynamics of objects that are not naturally imbeded in a linear space, in particular for describing dynamics of sets. This is important in many areas of applied mathematics, such as *optimal shape* [12, 5], *visual servoing* [6, 4], and *mathematical morphology* [8–11]. We refer to [2] for numerous motivations. Accordingly, the issue of discretization and constructive approximation for mutational equations, which is the aim of this paper, is of particular interest. We adapt the Euler theorem, which allows us to approximate solutions by a sequence of points in metric space. We end this paper by giving an example of an application for computing the evolution of tubes.

2. TRANSITIONS ON METRIC SPACES

Definitions given in this section and those that follow are quoted from [2].

Transitions adapt to metric spaces the concept of half-line $x + hv$ starting from x in the direction v by replacing it by "curved" half-lines $\vartheta(h, x)$.

* E-mail: najman@thomson-lcr.fr.

0022-247X/95 $12.00

Indeed, the "linear" structure of half-lines in vector spaces is not really needed to build a differential calculus.

DEFINITION 1 [2]. Let X be a metric space for a distance d. A map ϑ: $[0, 1] \times X \mapsto$ satisfying

$$\text{(i)} \quad \vartheta(0, x) = x$$

$$\text{(ii)} \quad \|\vartheta(x)\| := \sup_{h \neq k} \frac{d(\vartheta(h, x), \vartheta(k, x))}{|h - k|} < +\infty$$

$$\text{(iii)} \quad \|\vartheta\|_\Lambda := \sup_{h \in [0,1], x \neq y} \frac{d(\vartheta(h, x), \vartheta(h, y))}{d(x, y)} < +\infty$$

$$\text{(iv)} \quad \lim_{h \to 0+} \frac{d(\vartheta(t + h, x), \vartheta(h, \vartheta(h, \vartheta(t, x))}{h} = 0$$

is called a *transition.*

We denote by $\overline{\Theta}(X)$ the space of all transitions on X.

We define an equivalence relation $\sim_x$ between transitions by

$$\vartheta_1 \sim_x \vartheta_2 \qquad \text{if and only if } \lim_{h \to 0+} \frac{d(\vartheta_q(h, x), \vartheta_2(h, x))}{h} = 0.$$

We say that $(X, \Theta(X))$ is a (complete) *mutational space* if X is a (complete) metric space and $\Theta(X) \subset \overline{\Theta}(X)$ is a nontrivial subspace of transitions, closed in $\mathcal{C}([0, 1] \times X, X)$ supplied with the pointwise convergence.

One observes that the transitions $\vartheta(h, \cdot)$ are Lipschitz uniformly with respect to $h \in [0, 1]$ and that for every $x \in X$, the maps $\vartheta(\cdot, x)$ are Lipschitz.

We shall supply a space $\Theta(X)$ of transitions with the distances d_∞ of uniform convergence and Lipschitz semidistance defined, respectively, by

$$d_\infty(\vartheta, \tau) := \sup_{h \in [0,1], z \in X} d(\vartheta(h, z), \tau(h, z))$$

and

$$d_\Lambda(\vartheta, \tau) := \sup_{h \in [0,1], z \in X} \frac{d(\vartheta(h, z), \tau(h, z))}{h}.$$

3. MUTATIONAL EQUATIONS IN METRIC SPACES

Let $x(\cdot)$ be an application from $[0, 1]$ onto X, and ϑ: $[0, 1] \times X \to X$. We say that ϑ is a mutation of $x(t)$ at time t if

$$\lim_{h \to 0+} \frac{d(\vartheta(h, x(t)), x(t + h))}{h} = 0.$$

In this case, we note

$$\mathring{x}(t) \ni \vartheta.$$

Let us consider a mutational space $(X, \Theta(X))$ and a single-valued map f: $[0, \infty[\times X \mapsto \Theta(X)$ from X to its space of transitions. We say that a function $x(\cdot)$ from $[0, T]$ to X is a *solution to the mutational equation* $\mathring{x} \ni f(t, x)$ if for all $t > 0$, $f(t, x(t))$ is a mutation of $x(t)$ at time t, i.e., if

$$\forall t \in [0, T], \qquad \mathring{x}(t) \ni f(t, x(t)), \tag{1}$$

or, equivalently, if

$$\forall t \geq 0, \qquad \lim_{h \to 0+} \frac{d(f(t, x(t))(h, x(t)), x(t+h))}{h} = 0.$$

3.1. *Examples of Mutational Equations*

Let us point out, through two examples, that the uniqueness of the mutation is not ensured, which justifies the notation.

- First, consider the constant tube in $X = \mathbb{R}^2$

$$K(t) = B,$$

where B is the Euclidean unit ball.

It is clear that

$$0 \in \mathring{K}(t), \qquad \forall t.$$

But we can also check that the Lipschitz map defined by

$$\varphi(x, y) = (-y, x)$$

satisfies

$$\varphi \in \mathring{K}(t).$$

- Let F be a set-valued map from a vector space onto itself. The reachable map ϑ associated to the differential inclusion $x'(t) \in F(x(t))$, $x(0) = x_0$, defined by $\vartheta(t, x_0) := \{\{x(s)\} \mid x'(s) \in F(x(s)), x(0) = x_0, s \in [0, t]\}$ is a good choice for being a mutation of $x(\cdot)$ at time $t = 0$. Let us notice that in order for it to be mutation, it is not necessary that ϑ belong to $\Theta(X)$. Let us apply this remark to mathematical morphology.

Let X be the set of compacts of $\mathbb{R}^n$ (which is a metric space for the

Hausdorff distance), K be an element of X, $\vartheta_1(t, K) := \{\{x(s)\} \mid x'(s) \in B, x(0) \in K, s \in [0, t]\}$, where B is the unit Euclidean ball, and $\vartheta_2(t, K) := \{\{x(s)\} \mid x'(s) \in N_K(x(s)) \cap B, x(0) \in K, s \in [0, t]\}$, where $N_K(x)$ is the subnormal cone[1] of the set K at point x for the Euclidean norm.

In [7] it is shown that the morphological tube $K(t) := K \oplus tB = \{k + tb \mid k \in K, b \in B\}$, which corresponds to the dilation (the Minkowski sum) of a compact K with respect to the Euclidean unit ball B, satisfies the two mutational equations

$$\forall t \geq 0, \qquad \mathring{K}(t) \ni \vartheta_1 \tag{2}$$

$$\forall t \geq 0, \qquad \mathring{K}(t) \ni \vartheta_2, \tag{3}$$

with $K(0) = K$. Equation (2) is natural, and Eq. (3) clearly establishes, without any regularity assumptions on the compact set K, the intuitive idea that the dilation transforms the initial set K in the direction of the normal at any point of the set. Indeed, when the set K is a regular manifold, the subnormal cone $N_K(x)$ is reduced to the half-line spanned by the outward normal $n(x)$. But if ϑ_1 is a transition, this is not the case for ϑ_2, since the application $\vartheta_2(t, \cdot)$ is not Lipschitz.

Peano's Theorem, which states the existence of a solution $x(\cdot)$ to Eq. (1), can be adapted to the case of mutational equations.

3.2. *Peano's Theorem for Mutational Equations*

We give here Peano's theorem for mutational equations.

THEOREM 2 [2]. *Let $(X, \Theta(X))$ be a mutational space, and f: $[0, \infty[\times X \mapsto \Theta(X)$ be a uniformly continuous map* bounded *in the sense that*

$$\forall t \geq 0, \forall x \in X, \|f(t,x)\|_\Lambda := \sup_{h \in [0,1], y \neq z} \frac{d(f(t,x)(h,y), f(t,x)(h,z))}{d(y,z)} \leq c_\Lambda$$

[1] The *external circatangent cone* of K at x is

$$C_K(x) := \{v \mid C_\uparrow d(x, K)(x)(v) \leq 0\},$$

and the *external subnormal cone* of K at x is the negative polar cone of $C_K(x)$, i.e.,

$$N_K(x) := C_K(x)^- = \{p \mid \forall v \in C_K(x), \langle p, v \rangle \leq 0\}.$$

See [3] for more details.

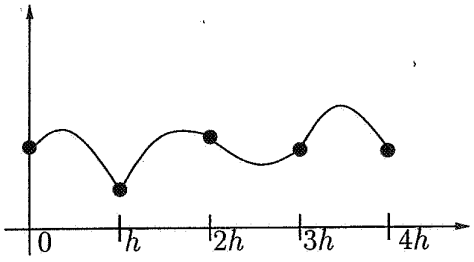


FIG. 1. Illustration of Euler's method for mutational equations.

and that

$$\forall t \geq 0, \forall x \in X, \forall y \in X, \|f(t,x)(y)\| := \sup_{k \neq h} \frac{d(f(t,x)(h,y), f(t,x)(k,y))}{|h-k|} \leq c.$$

Assume that the closed bounded balls of X are compact.

Then, from any initial state $x_0 \in X$ starts one solution to the mutational equation $\overset{\circ}{x} \ni f(t,x)$.

4. Euler Method for Mutational Equations

For the sake of simplicity, we consider a mutational equation with non-explicit time dependence, but the following can be easily adapted to the explicit time-dependent case. We thus consider the mutational equation

$$\overset{\circ}{x}(t) \ni f(x(t)). \tag{4}$$

A natural way for approximating solutions to differential mutations is the difference scheme

$$x_{j+1} := f(x_j)(h, x_j),$$

where $h \in]0, 1]$ is fixed.

We can interpolate the x_j's on the nodes jh by setting (Fig. 1)

$$x_h(t) := f(x_j)(t - hj, x_j), \qquad \forall t \in [jh, (j+1)h[.$$

We shall prove that the function $x_h(\cdot)$ converges in some sense to a solution to the mutational equation (4).

THEOREM 3. *We consider a mutational space* $(X, \Theta(X))$ *and a single-valued map* f: $X \mapsto \Theta(X)$. *We posit the assumptions of Theorem* 2.

For $h \in]0, 1]$, *we set*

$$x_{j+1} := f(x_j)(h, x_j). \tag{5}$$

Let $x_h(\cdot)$ *be a piecewise defined function which interpolates the* x_j's *on the nodes* jh *and on a finite time interval* $[0, T]$:

$$x_h(t) := f(x_j)(t - hj, x_j), \qquad \forall t \in [jh, (j+1)h[, t \leq T. \tag{6}$$

Then, starting from $x_0 \in X$, *the solution to the explicit difference scheme* (5) *converges to a solution to the mutational equation*

$$\overset{\circ}{x}(t) \ni f(x(t)) \tag{7}$$

where $h \to 0$, *in the sense that a subsequence of function* x_h *converges uniformly to a solution* $x(\cdot)$ *to* (7) *with* $x(0) = x_0$, *on* $[0, T]$.

We set

$$d_\infty(\tau, \overset{\circ}{x}(t)) := \inf_{\sigma \in \overset{\circ}{x}(t)} d_\infty(\tau, \sigma)$$

We need the following lemma:

LEMMA 4. *We posit the assumptions of Theorem* 3. *Then, for all* $\varepsilon > 0$, *there exists* $H > 0$ *such that for* $h < H$, *we have, for all* $t > 0$ *and all* $s > 0$,

$$\begin{aligned} &\text{(i)} \quad d_\infty(f(x_h(t)), \overset{\circ}{x}_h(t)) \leq \varepsilon \\ &\text{(ii)} \quad d(x_h(t), x_h(t+s)) \leq cs. \end{aligned} \tag{8}$$

Proof. We first note that

$$\overset{\circ}{x}_h(t) \ni f(x_j), \qquad \forall t \in [jh, (j+1)h[$$

because we have, for $\varepsilon > 0$ and s small,

$$\begin{aligned} d(f(x_j)(s, x_h(t)), x_h(t+s)) &= d(f(x_j)(s, f(x_j)(t - hj, x_j)), f(x_j)(t - hj + s, x_j)) \\ &\leq \varepsilon \end{aligned}$$

by Definition 1(iv).

Then we have

$$d_\infty(f(x_h(t)), \mathring{x}_h(t)) \leq d_\infty(f(x_h(t)), f(x_j)) := \sup_{s\in[0,1], z\in X} d(f(x_h(t))(s,z), f(x_j)(s,z))$$

and, by uniform continuity of f, for all $\varepsilon > 0$, there exists $\eta > 0$ such that

$$\forall t \in [jh, (j+1)h[, d_\infty(f(x_h(t)), f(x_j)) \leq \varepsilon$$

as soon as $d(x_h(t), x_j) \leq \eta$. But for all $t \in [jh, (j+1)h[$,

$$\begin{aligned} d(x_h(t), x_j) &= d(f(x_j)(t-hj, x_j), fx_j)(0, x_j)) \\ &\leq c(t-hj) \\ &\leq ch, \end{aligned}$$

which proves inequality (8(i)).

Now, for $t \in [jh, (j+1)h[$ and $t + s \in [kh, (k+1)h[$, we have

$$\begin{aligned} d(x_h(t+s), x_h(t)) &\leq d(f(x_k)(t+s-hk, x_k), f(x_k)(0, x_k)) \\ &\quad + \sum_{j<i<k} d(f(x_i)(h, x_i), f(x_i)(0, x_i)) \\ &\quad + d(f(x_j)(h, x_j), f(x_j)(t-hj, x_j)) \\ &\leq c(t+s-hk) + \sum_{j<i<k} ch + c(h-t+hj) \\ &\leq c(s + h(-k + (k-j-1) + j + 1)) \\ &\leq cs, \end{aligned}$$

which proves inequality (8(ii)).

Proof of Theorem 3. We continue as in the proof of Theorem 2. For the sake of completeness, we give here the end of the proof.

Since the closed bounded balls of X are compact, and since the solutions remain in such closed bounded balls of X, we deduce that $x_h(t)$ remains in a compact set of X.

Lemma 4, property (ii), implies that the sequence of continuous functions $x_h(\cdot)$ is equicontinuous. Therefore Ascoli's theorem implies that a subsequence (again denoted by $x_h(\cdot)$) converges uniformly to $x(\cdot)$.

This limit is a solution since for any $t \in [jh, (j+1)h[$,

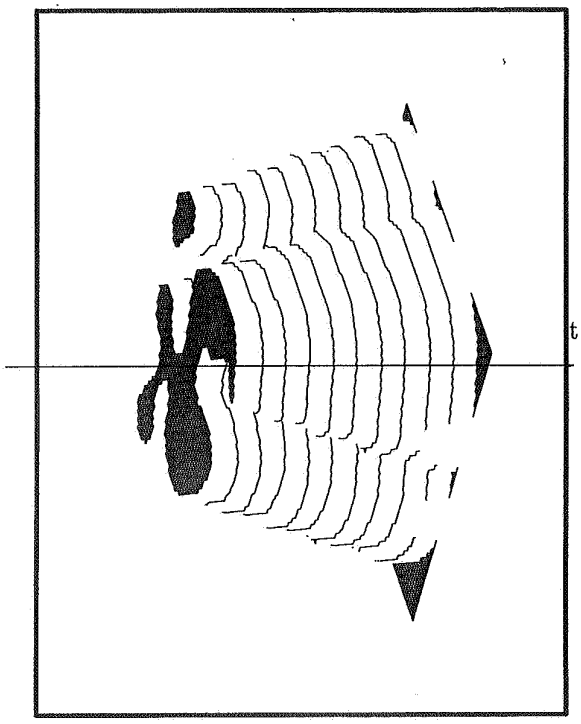


FIG. 2. Trajectory of the dilation by a segment whose orientation relies on time of a compact included in $\mathbb{R}^2$. The initial set $K(0)$ is in grey.

$$d_\infty(f(x(t)), \mathring{x}_h(t)) \leq d_\infty(f(x(t)), f(x_j)) + d_\infty(f(x_j), \mathring{x}_h(t)).$$

But, $f(x_j) \ni \mathring{x}_h(t)$, and f is uniformly continuous. Thus, for all $\varepsilon > 0$, there exists $\eta > 0$ such that

$$d_\infty(f(x(t)), \mathring{x}_h(t)) \leq \varepsilon$$

as soon as $d(x_h(t), x_j) \leq ch \leq \eta$.

Hence the theorem is proved.

5. An Example of Application to Tubes

The main example of application is the evolution of tubes, which are compact-valued maps K: $\mathbb{R} \rightsquigarrow X = \mathbb{R}^n$. Let us look at a morphological example, which is the dilation of a compact by a segment of variable angle.

We set $f(t, K(t))(h, K(t)) = K(t) \oplus h\Theta(t) = \{k + h\theta \mid k \in K(t), \theta \in \Theta(t)\}$, where $\Theta(t)$ is a segment of angle t. Theorem 3 amounts to saying that we can approximate the solution to

$$\mathring{K} \ni f(t, K(t)),$$

with $K(0) = K_0$ by the sequence

$$K_{j+1} = K_j \oplus h\Theta(hj).$$

Figure 2 shows an example of approximation of the solution to this mutational equation.

ACKNOWLEDGMENT

The author thanks Professor Jean-Pierre Aubin for helpfull advices and suggestions, as well as the anonymous reviewer whose comments helped clarify the final version of this paper.

REFERENCES

1. J.-P. AUBIN, "Morphological and Mutational Analysis, Tools for Shape Regulation and Optimization," Commett Matari Programme, CEREMADE, University of Paris-Dauphine, France, 1993.
2. J.-P. AUBIN, Mutational equations in metric spaces, *Set-Valued Anal.* **1** (1993), 3–46.
3. J.-P. AUBIN AND H. FRANKOWSKA, "Set-Valued Analysis," Birkhäuser, Basel, 1990.
4. L. DOYEN, "Évolution, contrôle et optimisation de formes," Thèse de Doctorat, Université Paris-Dauphine, Paris, France, juin 1993.
5. L. DOYEN, Inverse function theorems and shape optimization, *SIAM J. Control*, to appear.
6. L. DOYEN, Shape Lyapunov functions and visual servoing, *J. Math. Anal. Appl.*, to appear.
7. L. DOYEN, L. NAJMAN, AND J. MATTIOLI, Mutational equations of morphological dilation tubes, *J. Math. Imaging Vision*, to appear; preprint, Cahier de Mathématiques de la Décision, No. 9369, CEREMADE, Université Paris, Dauphine, France.
8. G. MATHERON, "Random Sets and Integral Geometry," Wiley, New York, 1975.
9. M. SCHMITT AND J. MATTIOLI, "Morphologie Mathématique. Logique–Mathématiques–Informatique," Masson, Paris, 1993.
10. J. SERRA, "Image Analysis and Mathematical Morphology," Academic Press, London, 1982.
11. J. SERRA, Ed., "Image Analysis and Mathematical Morphology, Vol. 2. Theoretical Advances," Academic Press, London, 1988.
12. J. SOKOLOWSKI AND J.-P. ZOLESIO, "Introduction to Shape Optimization," Springer-Verlag, Berlin, 1992.

8 The Montagnes Russes Algorithm for Global Optimization

J.P. Aubin and L. Najman. The Montagnes Russes Algorithm for Global Optimization. *Mathematical Methods of Operations Research* (1998) 48 :153-168. Special issue on 'Set-valued optimization'.

Math Meth Oper Res (1998) 48:153–168

Mathematical Methods
of Operations Research

The Montagne Russe algorithm for global optimization

Jean-Pierre Aubin[1], **Laurent Najman**[2]

[1] Centre de Recherche Viabilité, Jeux, Contrôle, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, F-75775 Paris Cédex 16, France (e-mail: aubin@dmi.ens.fr)
[2] 20, rue Manin, F-75019 Paris, France (e-mail: lna@mygale.org)

Received June 1997/Revised version December 1997

Abstract. The "Montagnes Russes" algorithm for finding the global minima of a lower semi-continuous function (thus involving state constraints) is a descent algorithm applied to an auxiliary function whose local and global minima are the global minima of the original function. Although this auxiliary function decreases along the trajectory of any of its minimizing sequences, the original function jumps above local maxima, leaves local minima, play "Montagnes Russes" (called "American Mountains" in Russian and "Big Dipper" in American!), but, ultimately, converges to its infimum. This auxiliary function is approximated by an increasing sequence of functions defined recursively at each point of the minimizing sequence.

Key words: Global optimization, viability theory, viability kernel, Lyapunov function

Introduction

Let us consider the minimization problem

$$v_0 := \inf_{x \in X} V(x)$$

where $X := \mathbf{R}^n$ is a finite dimensional vector space and

$$V : X \mapsto \mathbf{R} \cup \{+\infty\}$$

is a nontrivial lower semicontinuous extended function assumed to be bounded from below.

When V is differentiable, it is obviously a Lyapunov function for the continuous gradient method

$$\text{as long as } V'(x(t)) \neq 0, \quad x'(t) = -\frac{V'(x(t))}{\|V'(x(t))\|}$$

so that $V(x(t))$ is not increasing. Since the equilibria of this differential equation are the critical points of the function V, the gradient algorithm, when it converges, may stop at such a critical point, and in particular, at a local minimum.

An obvious way to avoid this difficulty is to ask whether V is an a-exponential Lyapunov function (where $a > 0$) in the sense that it satisfies:

$$\forall t \geq 0, \quad V(x(t)) \leq (V(x_0) - v_0)e^{-at} + v_0 \tag{1}$$

In this case, the limit of the solution, if it exists, would reach the global minimum of V and provide a continuous algorithm to achieve a global minimum. But, naturally, V is not necessarily an exponential Lyapunov function for the gradient equation, except in the rare case when property

$$(*) \quad \forall x, \quad V(x) - v_0 \leq \frac{1}{a}\|V'(x)\|$$

holds true.

A way to overcome the fact that V is not necessarily a exponential Lyapunov function for the gradient equation is to replace the latter by the simple differential inclusion

$$\forall t \geq 0, \quad x'(t) \in B$$

where B denotes the unit ball of the finite dimensional vector space X, leaving open the direction to be chosen by the algorithm, as in the methods of simulated annealing. But instead of choosing the velocities at random and being satisfied by convergence in probability, we shall ask whether V can be an exponential Lyapunov function for the differential inclusion $x' \in B$.

This "viability approach" does not require any regularity properties of the function V, since it is not longer involved in the definition of the dynamics of the gradient, which is replaced by the simplest differential inclusion ($x' \in B$) one can think of. In this approach, the function V is involved only through its epigraph, assumed to be closed, and by the properties of the contingent cone at its points. Observing that the contingent cone to the epigraph of a function at a point of its graph is the epigraph of the contingent epiderivative[1], one

---

[1] Let $V : X \mapsto \mathbf{R} \cup \{\pm\infty\}$ be a nontrivial extended function and x belong to its domain $\mathrm{Dom}(V) := \{x \mid V(x) \neq \pm\infty\}$. The contingent epiderivative $D_\uparrow V(x)(u)$ of V at x in the direction u is equal to

$$D_\uparrow V(x)(u) := \liminf_{h \to 0+, u' \to u} \frac{V(x + hu') - V(x)}{h}$$

If we note $\mathcal{E}p(V) := \{(x, \lambda) \in X \times \mathbf{R} \mid V(x) \leq \lambda\}$ the epigraph of V, we have the following property:

$$\mathcal{E}p(D_\uparrow V(x)) = T_{\mathcal{E}p(V)}(x, V(x))$$

where $T_{\mathcal{E}p(V)}(x, V(x))$ is the contingent cone to $\mathcal{E}p(V)$ at $(x, V(x))$.

See for instance Chapter 6 of Set-valued analysis, [3, Aubin & Frankowska] for a presentation of epidifferentiable calculus.

can characterize property (1) for lower semicontinuous extended functions: Theorem 9.2.2 of Viability theory, [1, Aubin] states that property

$$\forall x \in \mathrm{Dom}(V), \quad \inf_{u \in B} D_\uparrow V(x)(u) + a(V(x) - v_0) \leq 0 \tag{2}$$

is necessary and sufficient for the existence of one solution $x(.)$ to $x'(t) \in B$ satisfying (1) starting from any given initial state $x_0 \in \mathrm{Dom}(V)$.

Observe also that the Fermat rule states that at every local minimum x of V, $0 \leq \inf_{u \in B} D_\uparrow V(x)(u)$. Therefore, an exponential Lyapunov function satisfying

$$\forall x \text{ such that } V(x) > v_0, \quad \inf_{u \in B} D_\uparrow V(x)(u) \leq -a(V(x) - v_0) < 0$$

does not have local minima when $V(x) > v_0$.

Remark: If the function $v \mapsto D_\uparrow V(x)(v)$ is convex, then it is the support function of its subdifferential $\partial_- V(x)$ of V at x, which is the closed convex subset defined by

$$\partial_- V(x) := \{p \in X^* \mid \forall u \in X, \langle p, u \rangle \leq D_\uparrow V(x)(u)\}$$

(see chapter 6 of Set-valued analysis, [3, Aubin & Frankowska], for instance.) In this case, the above property (2) can be written in the form

$$\forall x, \quad V(x) - v_0 \leq \frac{1}{a} d(0, -\partial_- V(x))$$

thanks to the lopsided minimax theorem, because

$$\begin{cases} \inf_{u \in B} D_\uparrow V(x)(u) + a(V(x) - v_0) \\ \quad \leq \inf_{u \in B} \sup_{p \in \partial_- V(x)} \langle p, u \rangle + a(V(x) - v_0) \\ \quad \leq \sup_{p \in \partial_- V(x)} \inf_{u \in B} \langle p, u \rangle + a(V(x) - v_0) \\ \quad \leq \sup_{p \in \partial_- V(x)} (-\|p\|_*) + a(V(x) - v_0) \\ \quad \leq -d(0, -\partial_- V(x)) + a(V(x) - v_0) \end{cases}$$

□

1 The continuous algorithm

Unfortunately, even when V is differentiable, we observe that condition (2) amounts to saying that V is still an exponential Lyapunov function of the gradient equation. However, if this is not the case, Theorem 9.3.1 of Viability

theory, [1, Aubin] implies[2] the existence of the optimal exponential Lyapunov function V_∞, which is the smallest lower semicontinuous exponential Lyapunov function V_∞ larger than or equal to V.

This exponential Lyapunov function can take infinite values.

Consequently, for any initial state $x_0 \in \mathrm{Dom}(V_\infty)$, there exists at least one solution to the differential inclusion $x' \in B$ starting from x_0 satisfying

$$V(x(t)) \leq V_\infty(x(t)) \leq (V_\infty(x_0) - v_0)e^{-at} + v_0$$

Although we know that $V(x(t))$ converges to v_0 when $t \to +\infty$, the function $t \mapsto V(x(t))$ is not necessarily decreasing. Along such a solution, the function V jumps above local maxima, leaves local minima, plays "Montagnes Russes" (called "American Mountains" in Russian and "Big Dipper" in American!), but, ultimately, converges to its infimum.

The domain of this optimal exponential Lyapunov function describes the basin of attraction of the minima, which we define as follows: If $\mathscr{S}(x)$ denotes the set of solutions to be differential inclusion $x' \in B$ starting from x, we set

$$\rho^\flat_{V,a}(x) := \inf_{x(.) \in \mathscr{S}(x)} \sup_{t \geq 0} \frac{V(x(t)) - v_0}{e^{-at}}$$

We thus define the a-basin of attraction of V as

$$\mathrm{Bas}_a(V) := \{x \in X \mid \rho^\flat_{V,a}(x) < +\infty\}$$

In other words, a state x_0 belongs to the a-basin of attraction if and only if there exist a solution $x(.) \in \mathscr{S}(x_0)$ and a constant c such that

$$\forall t \geq 0, \quad V(x(t)) \leq ce^{-at} + v_0$$

One can prove that the domain of V_∞ is actually the basin of exponential attraction of V.

An algorithm yielding solutions $x(.)$ satisfying inequality (*) requires in principle an *a priori* knowledge of the infimum to guarantee the convergence to a global minimum.

---

[2] The epigraph $\mathscr{E}p(V)$ of an exponential Lyapunov function V is the viability domain for the system of differential inclusions

$$\begin{cases} i) & x'(t) \in B \\ ii) & x'(t) = av_0 - aw(t) \end{cases}$$

This means that for any initial state $(x_0, w_0) \in \mathscr{E}p(V)$, there exists a solution to the above solution viable in the epigraph of V, in the sense that

$$\forall t \geq 0, \quad (x(t), w(t)) \in \mathscr{E}p(V)$$

Indeed, to say that $x(.)$ satisfies condition (1) means that $\forall t \geq 0$, $(x(t), w(t)) \in \mathscr{E}p(V)$ where $w(t) := (V(x_0) - v_0)e^{-at} + v_0$ is the solution to differential equation $x'(t) = av_0 - aw(t)$ starting at $V(x_0)$.

The epigraph of the optimal exponential Lyapunov function V_∞ is the viability kernel (the largest closed subset of the epigraph viable under this system) of the epigraph of V for the above system of differential inclusions (see Chapter 9 of Viability theory, [1, Aubin]).

Actually, we shall parametrize the problem by replacing v_0 by any $\lambda \in \mathbf{R}$. We denote by $V_\alpha(\cdot, \lambda)$ the optimal exponential Lyapunov function larger than V, satisfying: for any initial state $x_0 \in \mathrm{Dom}(V_\alpha(\cdot, \lambda))$, there exists at least one solution to the differential inclusion $x' \in B$ starting from x_0 satisfying

$$V(x(t)) \leq V_\alpha(x(t)) \leq (V_\alpha(x_0) - \lambda)e^{-at} + \lambda$$

If $\lambda < \inf_{x \in X} V(x)$, then it is easy to prove that $V_\alpha(x, \lambda) = +\infty$ for every $x \in \mathrm{Dom}(V)$, because in this case, every solution of the algorithm leaves the epigraph of V in finite time[3].

If $\lambda > \inf_{x \in X} V(x)$, then we observe that $V_\alpha(x, \lambda) = V(x)$ for every x in the level set $\{x \mid V(x) \leq \lambda\}$. Indeed, the pair $(x, e^{-at}(V(x) - \lambda) + \lambda)$ is a solution to $(x', w') \in B \times \{a(w - \lambda)\}$ starting from $(x, V(x))$ and which remains in the epigraph of V because $e^{-at}(V(x) - \lambda) + \lambda$ decreases. Therefore, every x in this level set belongs to the viability kernel of the epigraph, which is the epigraph of V_α.

The optimal exponential Lyapunov function being only lower semicontinuous, may take infinite values. But this phenomenon cannot happen when V is Lipschitz, and more generally, Hölder.

Proposition 1.1. *Let $V : \mathrm{Dom}(V) = \mathbf{R}^n \to \mathbf{R}$ be a Hölder[4] function of parameter β, and V_α be the smallest exponential Lyapunov function associated. If V achieves its minimum, then V_α is finite.*

Proof: Assume that $\inf_{x \in X} V(x) = 0$. We have to show that

$$\forall x_0 \in \mathbf{R}^n, \quad V_\alpha(x_0) := \inf_{x(\cdot) \in \mathscr{S}(x_0)} \sup_{t \geq 0} \frac{V(x(t))}{e^{-at}} < +\infty$$

Consider $x_0, x_1, x_2 \in \mathbf{R}^n$ such that $\|x_1 - x_2\| > \beta/a$ and $V(x_2) = 0$. We set

$$l_1 := \|x_1 - x_0\|$$

$$l_2 := \|x_2 - x_1\|$$

We parameterize the path $(x_0, x_1) \cup (x_1, x_2)$ in the following way

$$0 \leq s \leq l_1 \quad x(s) := x_0 + s\frac{x_1 - x_0}{l_1}$$

$$l_1 \leq s \leq l_2 + l_1 - \frac{\beta}{a} \quad x(s) := x_1 + \frac{x_2 - x_1}{l_2}(s - l_1)$$

$$l_2 + l_1 - \frac{\beta}{a} \leq s < +\infty \quad x(s) := x_1 + \frac{x_2 - x_1}{l_2}\left(l_2 - \frac{\beta e^{\frac{a}{\beta}(l_2 + l_1 - s) - 1}}{a}\right)$$

---

[3] Indeed, for any initial state $(x_0, w_0) \in \mathscr{E}p(V)$ outside the viability kernel $\mathscr{E}p(V_\alpha)$, all solutions $(x(t), (w_0 - \lambda)e^{-at} + \lambda)$ leaves the epigraph of V in finite time.

[4] This means that

$$V(x_1) - V(x_2) \leq k\|x_1 - x_2\|^\beta$$

It is clear that $\lim_{s\to+\infty} x(s) = x_2$ and that $x(\,.\,) \in S(x_0)$. In fact, $x(\,.\,)$ is clearly continuous and

$$0 < s < l_1 \quad x'(s) = \frac{x_1 - x_0}{l_1} \in B$$

$$l_1 < s < l_2 + l_1 - \frac{\beta}{a} \quad x'(s) = \frac{x_2 - x_1}{l_2} \in B$$

$$l_2 + l_1 - \frac{\beta}{a} < s < +\infty \quad x'(s) = \frac{x_2 - x_1}{l_2} e^{a/\beta(l_2+l_1-s)-1} \in B$$

Now,

$$\forall s \geq l_2 + l_1 - \frac{\beta}{a}, \quad \frac{V(x(s))}{e^{-as}} \leq k \frac{\|x(s) - x_2\|^\beta}{e^{-as}}$$

$$\leq k e^{as} \left(l_2 - \left(l_2 - \frac{\beta e^{\frac{a}{\beta}(l_2+l_1-s)-1}}{a} \right) \right)^\beta$$

$$\leq k e^{as} \frac{\beta^\beta}{a} e^{a(l_2+l_1-s)} e^{-\beta}$$

$$\leq k \left(\frac{\beta}{a} \right)^\beta e^{a(l_2+l_1)} e^{-\beta}$$

and finally

$$V_\infty(x_0) \leq \max \left(\sup_{0\leq s\leq l_1} V\left(x_0 + s\frac{x_1 - x_0}{l_1} \right) e^{as}, \right.$$

$$\left. \sup_{l_1 \leq s \leq l_1 + l_2 - \frac{\beta}{a}} V\left(x_1 + s\frac{x_2 - x_1}{l_2} \right) e^{as}, k \left(\frac{\beta}{a} \right)^\beta e^{a(l_2+l_1)} e^{-\beta} \right)$$

which proves the result. □

2 The discrete algorithm

Let the exponential rate a be fixed and a discretization step $h \in]0, 1/a[$ be chosen.

Definition 2.1. *We shall say that a lower semicontinuous function $U \geq V$ is a* **discrete descent function** *if from any $x_0 \in \mathrm{Dom}(U)$, there exists a sequence defined by the algorithm*

$$x_0^h = x_0 \ \& \ x_{p+1}^h := x_p^h + hu^h \text{ where } u^h \in B \tag{3}$$

satisfying

$$U(x_p^h) \leq (1 - ah)^p (U(x_0) - v_0) + v_0 \tag{4}$$

The knowledge of such a discrete descent function U larger than or equal to V provides a discrete "Montagnes Russes algorithm", since any cluster point of the sequence x_p^h achieves a global minimum.

We shall provide a constructive formula providing both the smallest discrete descent function larger than V and the algorithm converging to a global minimum.

Let us set $V_0^h := V$ and define recursively the sequence of functions V_n^h by

$$V_n^h(x) := \max\left(V_{n-1}^h(x), \frac{1}{1-ah}\left(\inf_{u \in B} V_{n-1}^h(x + hu) - ahv_0\right)\right) \tag{5}$$

Theorem 2.2. *Let V be a nontrivial lower semicontinuous extended function assumed to be bounded from below. Then there exists a function V_∞^h which is the smallest lower semicontinuous discrete descent function larger than V, called the optimal descent function of V.*

It can be computed by the "Viability Kernel Algorithm" which provides it as a supremum of an increasing sequence of the functions V_n^h:

$$V_\infty^h(x) := \sup_{n \geq 0} V_n^h(x) \tag{6}$$

The set-valued map R^h defined by

$$R^h(x) := \left\{u \in B \mid V_\infty^h(x + hu) = \inf_{w \in B} V_\infty^h(x + hw)\right\} \tag{7}$$

defines the "Montagnes Russes Algorithm": From any $x_0 \in \mathrm{Dom}(V_\infty^h)$, a sequence defined by the Montagnes Russes algorithm

$$x_{p+1}^h := x_p^h + hu^h \quad \text{where } u^h \in R^h(x_p^h) \tag{8}$$

satisfies

$$V(x_p^h) \leq V_\infty^h(x_p^h) \leq (1 - ah)^p (V_\infty^h(x_0) - v_0) + v_0$$

Figure 1 shows some examples of computation of Lyapunov functions for various original functions.

Proof: We consider the discrete set-valued dynamical system $G^h : X \times \mathbf{R} \rightsquigarrow X \times \mathbf{R}$ defined by

$$G^h(x, w) := (x + hB) \times \{w + ah(v_0 - w)\}$$

and the closed subset $\mathscr{K} := \mathscr{E}p(V)$.

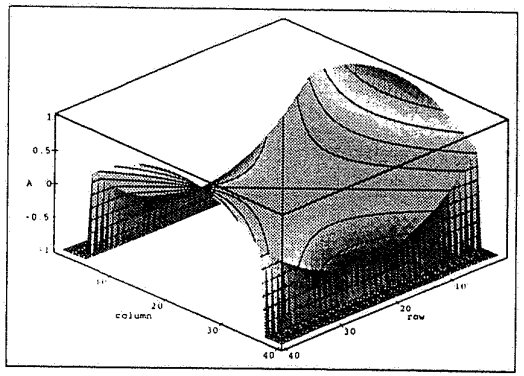
Any solution to the discrete set-valued dynamical system $(x_{p+1}^h, w_{p+1}^h) \in G^h(x_p^h, w_p^h)$ starting at $(x_0, U(x_0))$ viable in $\mathscr{E}p(U)$ is obviously a solution to

(a) Original function

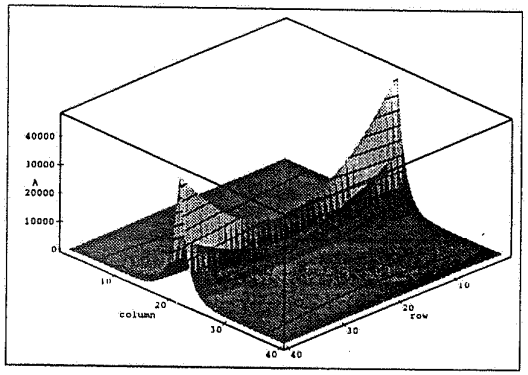
(b) Liapunov function at a different scale

(c) Original function

(d) Liapunov function at a different scale



(e) Original function



(f) Liapunov function at a different scale

Fig. 1. Various examples of computation of Lyapunov function. We show some intermediary steps of computation.

the algorithm (3) satisfying (4). Conversely, if U is a discrete descent function, then its epigraph is a discrete viability domain of G^h: Indeed, if $(x, w) \in \mathrm{Ep}(U)$, then

$$\begin{cases} (x + hu, w - ahw + ahv_0) \\ = (x + hu, (1 - ah)V(x) + ahv_0) + (0, (1 - ah)(w - V(x))) \\ \in \mathscr{E}p(U) + \{0\} \times \mathbb{R}_+ = \mathscr{E}_p(U) \end{cases}$$

Since the unit ball is compact, to say that the epigraph of a lower semi-continuous extended function U is a viability domain of G^h amounts to saying that

$$\inf_{u \in B} U(x + hu) \leq (1 - ah)U(x) + ahv_0$$

The solutions along which the function U decreases geometrically are given by $x^h_{p+1} = x^h_p + hu^h$ with $u^h \in R^h(x^h_p)$ where

$$R^h(x) := \left\{ u \in B \,|\, U(x + hu) = \inf_{w \in B} U(x + hw) \right\}$$

If V is not a discrete descent function, then its epigraph contains its (discrete) viability kernel, which is the largest closed (discrete) viable domain $\mathscr{K}_\alpha$ of G^h contained in this epigraph. This viability kernel is actually the epigraph of the function V^h_α defined by

$$V^h_\alpha(x) := \inf_{(x,\lambda) \in \mathscr{K}_\alpha} \lambda$$

Indeed, since

$$\mathscr{E}p(V^h_\alpha) \subset \mathscr{K}_\alpha + \{0\} \times \mathbb{R}_+$$

it is therefore enough to show that $\mathscr{K}_\alpha + \{0\} \times \mathbb{R}_+ \subset \mathscr{K}_\alpha$. In fact, we prove that *if $\mathscr{M} \subset X \times \mathbb{R}_+$ is a closed viability domain of G^h, then so is the subset*

$$\mathscr{M}_0 := \mathscr{M} + \{0\} \times \mathbb{R}_+$$

Let (x, w) belong to $\mathscr{M}_0$. To see that $G(x, w) \cap \mathscr{M}_0 \neq \varnothing$, let us set

$$U_{\mathscr{M}}(x) := \inf_{(x,\lambda) \in \mathscr{M}} \lambda$$

By assumption, there exists $u \in B$ such that $(x + hu, (1 - ah)U_{\mathscr{M}}(x)) + ahv_0)$ belongs to $\mathscr{M}$ so that

$$\begin{cases} (x + hu, w - ahw + ahv_0) \\ = (x + hu, (1 - ah)U_{\mathscr{M}}(x) + ahv_0) + (0, (1 - ah)(w - U_{\mathscr{M}}(x))) \in \mathscr{M}_0 \end{cases}$$

It remains to prove formula (6), which is the "epigraphical version" of the finite-difference approximation of the viability kernel introduced in [13, Saint-Pierre] (see also section 4.5 of Viability theory, [1, Aubin]). Since V^h_∞ is a discrete descent function larger than V, we see that we can associate with any x an element $u \in B$ such that $(x + hu, (1 - ah)V^h_\infty + ahv_0)$ belongs to the epigraph of V, so that

$$\frac{1}{1-ah}\left(\inf_{u\in B} V(x+hu) - ahv_0\right) \leq V^h_\infty(x)$$

and thus, such that $V^h_1(x) \leq V^h_\infty(x)$. We thus check recursively that $V^h_n(x) \leq V^h_\infty(x)$, so that

$$\sup_{n\geq 0} V^h_n(x) \leq V^h_\infty(x)$$

It remains to prove that $V^\sharp := \sup_{n\geq 0} V^h_n$ is a discrete descent function. If so, it will be larger than or equal to V^h_∞, and thus, equal.

By construction of the functions V^h_n, we can associate with any x an element $u_n \in B$ such that

$$V_n(x + hu_n) \leq (1 - ah)V^\sharp(x) + ahv_0$$

Since the unit ball B is compact, there exists a subsequence (again denoted by) u_n converging to some $\hat{u} \in B$. The sequence of functions V^h_n being nondecreasing, we deduce that

$$\liminf_{n\to\infty} V^h_n(x + hu_n) = \sup_n V^h_n(x + h\hat{u}) = V^\sharp(x + h\hat{u})$$

This implies that

$$\inf_{u\in B} V^\sharp(x + hu) \leq (1 - ah)V^\sharp(x) + ahv_0$$

which proves the claim. □

The second question we may ask is the following: *Is the limits of a sequence of discrete optimal exponential Lyapunov functions V^h_∞ an exponential Lyapunov function larger than V?*

It depends on what we understand as "limit": the appropriate concept is the one of *lower epilimit* defined in the following way:

Definition 2.3. *The epigraph of the lower epilimit*

$$\lim^\sharp_{\uparrow n\to\infty} V_n$$

of a sequence of extended functions $V_n : X \mapsto \mathbf{R} \cup \{+\infty\}$ is the upper limit of the epigraphs:

$$\mathcal{E}p(\lim^\sharp_{\uparrow n\to\infty} V_n) := \mathrm{Limsup}_{n\to\infty} \mathcal{E}p(V_n)$$

One can check that

$$\lim^{\sharp}_{\uparrow n\to\infty} V_n(x_0) := \liminf_{n\to\infty, x\to x_0} V_n(x)$$

and that if the sequence is increasing, that

$$\lim^{\sharp}_{\uparrow n\to\infty} V_n(x_0) = \sup_{n\geq 0} V_n(x_0)$$

We refer to Chapter 7 of Set-valued analysis, [3, Aubin & Frankowska] for further details on *epigraphical convergence.*

Meanwhile, we deduce from Theorem 4.5.2 of Viability theory, [1, Aubin] that

Theorem 2.4. *The lower epilimit of the sequence of optimal discrete descent functions V^h_∞ is an exponential Lyapunov function larger than V.*

3 Implementation

Once the problem is discretized, we know that the algorithm

$$x^h_{p+1} \in x^h_p + hR^h(x^h_p)$$

starting from a state x_0 such that $V^h_\infty(x_0) < +\infty$ converges to the global minimum in the sense that

$$V(x^h_p) \leq V^h_\infty(x^h_p) \leq (1-ah)^p (V^h_\infty(x_0) - v_0) + v_0$$

As in the continuous case, this algorithm requires in principle an *a priori* knowledge of the infimum to guarantee the convergence to a global minimum. If $v_0 < \inf_{x\in X} V(x)$, then it is easy to prove that $V^h_\infty(x) = +\infty$ for every $x \in \mathrm{Dom}(V)$, and if $v_0 > \inf_{x\in X} V(x)$, then we observe that $V^h_\infty(x) = V(x)$ for every x in the level set $\{x \mid V(x) \leq v_0\}$.

In any case, if one does not know the infimum v_0 of the function V in advance, we can replace in the definition of the function V^h_n at step $x^{h,n}_p$ the exact infimum v_0 by the smallest value v_p taken by the original function V on all the points which have been used up this time.

To implement this algorithm, we need also to compute V^h_∞ at least at the points x^h_p. But the computation of the optimal discrete descent function at a point requires eventually the knowledge of every values of the original function V. This is natural because the knowledge of the global minimum requires some global knowledge of the function V.

Hence, we shall approximate further the problem by replacing the optimal discrete descent function V^h_∞ by V^h_n and applying the algorithm

$$x^{h,n}_{p+1} \in x^{h,n}_p + hR^h_n(x^{h,n}_p)$$

where

$$R^h_n(x) := \left\{ u \in B \mid V^h_n(x+hu) = \inf_{w\in B} V^h_n(x+hw) \right\} \tag{9}$$

In this case, the knowledge of the function V_n^h at a given point x requires only the values of the original function V on a neighborhood $x+nhB$ of radius nh around x.

For simplicty, we assume from now on that $v_0 = 0$.

We observe that we always have the estimates

$$V_n^h(x) \leq V_{n+1}^h(x) \leq \frac{1}{1-ah} V_n^h(x)$$

and thus, that x *minimizes* V_n^h *on the ball* $x+hB$ *if and only if* $V_{n+1}^h(x) = \frac{1}{1-ah} V_n^h(x)$.

If the function V_p^h is constant on the ball $x+nhB$, then we deduce that

$$V_{n+p}^h(x) = \frac{1}{(1-ah)^n} V_k^h(x)$$

We deduce by induction that:

$$\sup_{p=0,\ldots,n} \frac{1}{(1-ah)^p} \inf_{u \in pB} V(x+hu) \leq V_n^h(x) \leq \frac{1}{(1-ah)^n} V(x)$$

We shall say that x is an a-local minimum of size nB for the discretization step h if $V_n^h(x) < V_{n+1}^h(x)$, because, in this case,

$$(1-ah) V_n^h(x) < \inf_{u \in B} V_n^h(x+hu)$$

We observe that if

$$\forall u \in nB, \quad V_k^h(x+hu) = V_{k+1}^h(x+hu)$$

then

$$V_k^h(x) = V_{k+n}^h(x)$$

and that

$$\forall u \in nB, \quad V_0^h(x+hu) < V_1^h(x+hu)$$

then we obtain recursively the formula

$$V_n^h(x) = \frac{1}{(1-ah)^n} \inf_{u \in nB} V(x+hu)$$

In this case, x is an a-local minimum of size nB for the discretization step h if

$$(1-ah) \inf_{u \in nB} V(x+hu) < \inf_{u \in (n+1)B} V(x+hu)$$

We can reformulate this remark by saying that if inequality

$$\frac{1}{(1-ah)^n} \inf_{u \in nB} V(x+hu) < V_n^h(x)$$

holds true, then there exists a point $y \in x + nhB$ such that $V_1^h(y) = V(y)$, at which there exists a descent direction u such that $V(y+hu) \leq (1-ah)V(y)$.

If x is an a-local minimum, then we can write

$$V_{n+1}^h(x) - V_n^h(x) = \frac{1}{1-ah}\left(\inf_{u \in B} V_n^h(x+hu) - V_n^h(x)\right) + \frac{ah}{1-ah} V_n^h(x)$$

so that if x minimizes V_n^h on the ball $x + hB$, then

$$V_{n+1}^h(x) - V_n^h(x) = \frac{ah}{1-ah} V_n^h(x)$$

Assume that at the initial state x_0, we have $V_n^h(x_0) = V_{n+1}^h(x_0)$, we know that for any $x_1^h \in R_n^h(x_0)$, we have

$$\frac{1}{1-ah}(V_n^h(x_1^h) - v_0) \leq V_n^h(x_0) - v_0$$

Therefore, as long as $V_n^h(x_p^h) = V_{n+1}^h(x_p^h)$, any $x_{p+1}^h \in R_n^h(x_p^h)$ satisfies

$$V_n^h(x_{p+1}^h) - v_0 \leq (1-ah)(V_n^h(x_p^h) - v_0) \leq (1-ah)^p(V_n^h(x_0) - v_0)$$

If for some q we have $0 < V_n^h(x_q^h) < V_{n+1}^h(x_q^h)$, we know that x_q^h is an a-local minimum of size nB for the discretization step h:

$$\forall u \in B, \quad (1-ah)V_n^h(x_q^h) < V_n^h(x_q^h + hu) + ahv_0$$

The algorithm stops there in this case.

These two modifications – replacing V_∞^h by V_n^h and v_0 by v_p – of the theoretical algorithm do not guarantee anymore the convergence of the algorithm to a global minimum.

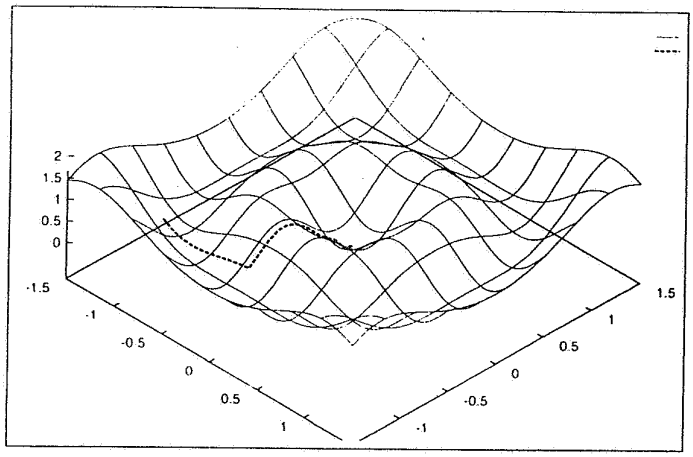
Figure 2 shows an example of a trajectory converging towards the minimum of a function.

Remark: Mathematical morphological tools

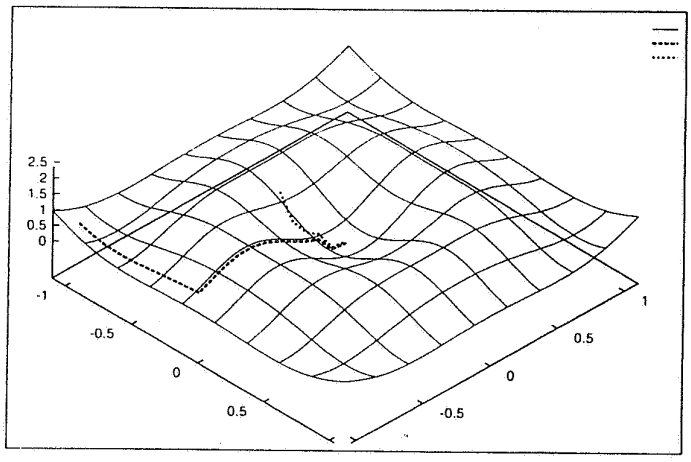
The recursive definition of the functions V_n^h involves the operation

$$\inf_{u \in B} V(x+hu)$$

which is very familiar in mathematical morphology: this is the erosion of the function V by the structuring element B. If B is a digital ball (the cube $[-\alpha, +\alpha]^l$ when $X = \mathbb{R}^l$ for instance), this is an operation which can be easily digitalized.



(a) Trajectory towards minimum



(b) Value of Liapunov function along trajectory

Fig. 2. Example of a trajectory converging towards the minimum of the function V defined by $V(x_1, x_2) := 1 - \cos(2\sqrt{x_1^2 + x_2^2})\cos(3\sqrt{x_1^2 + x_2^2})$. Note that for implementing this algorithm, we need to compute V_n^h at least at the points x_p^h. But the knowledge of the function V_n^h at a given point x requires only the values of the original function V on a neighborhood $x + nhB$ of radius nh around x.

Therefore, the "Montagnes Russes" algorithm uses a sequence of successive erosions, each being multiplied by factor $\eta := 1/(1 - ah)$.

Each erosion "erases" the basins of the local minima. The multiplication by the factor η raises the local minima except the global one. The domain of the optimal discrete descent function is then the basin of exponential attraction of the global minima.

References

[1] Aubin J-P (1991) Viability theory. Birkhäuser
[2] Aubin J-P, Cellina A (1984) Differential inclusions. Grundlehren der math. Wiss. 264, Springer-Verlag
[3] Aubin J-P, Frankowska H (1990) Set-valued analysis. Birkhäuser
[4] Aubin J-P, Najman L (1994) L'algorithme des montagnes russes pour l'optimisation globale. Comptes-Rendus de l'Académie des Sciences, Paris, 319:631–636
[5] Cardaliaguet P, Quincampoix M, Saint-Pierre P (1994) Some algorithms for differential games with two players and one target. MAM 28:441–461
[6] Cardaliaguet P, Quincampoix M, Saint-Pierre P (1994) Temps optimaux pour des problèmes avec contraintes et sans contrôlabilité locale. Comptes-Rendus de l'Académie des Sciences, Série 1, Paris, 318:607–612
[7] Cardaliaguet P, Quincampoix M, Saint-Pierre P (1995) Numerical methods for optimal control and differential games. Cahiers de Mathématiques de la Décision 9510
[8] Frankowska H (to appear) Control of nonlinear systems and differential inclusions. Birkhäuser
[9] Frankowska H, Quincampoix M (1991) L'algorithme de viabilité. Comptes-Rendus de l'Académie des Sciences, Série 1, Paris, 312:31–36
[10] Frankowska H, Quincampoix M (1991) Viability kernels of differential inclusions with constraints: algorithm and applications. J. Math. Systems, Estimation and Control 1:371–388
[11] Gorre A (to appear) The Montagnes Russes algorithm in metric spaces
[12] Quincampoix M, Saint-Pierre P (1995) An algorithm for viability kernels in Hölderian case: Approximation by discrete viability kernels. J. Math. Syst. Est. and Control
[13] Saint-Pierre P (1994) Approximation of the viability kernel. Applied Mathematics & Optimisation 29:187–209

9 From Crowd Simulation to Airbag Deployment : Particle Systems, a New Paradigm of Simulation

E. Bouvier, E. Cohen and L. Najman. From Crowd Simulation to Airbag Deployment : Particle Systems, a New Paradigm of Simulation. *Journal of Electronic Imaging* 6 (1), 94-107 (January 1997).
Special issue on Random Model in Imaging.

Journal of Electronic Imaging 6(1), 94–107 (January 1997).

From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation

Eric Bouvier
Université Paul Sabatier
I.R.I.T.
Toulouse, France
and
ArSciMed
207, rue de Bercy
75012 Paris, France

Eyal Cohen
Laurent Najman
ArSciMed
207, rue de Bercy
75012 Paris, France

Abstract. *Developing generic models to simulate complex behaviors is one of the great challenges of numerical simulation. We present here a generic approach based on particle systems that has proved to be efficient in various kinds of complex situations, such as crowd simulation or airbag deployment. In crowd simulation each particle represents one person of the crowd whose behavior is assigned according to the class it belongs to. In airbag deployment, both the gas mixture and the bag are modeled by particles. In this case, particles don't represent the molecules but abstract blocks chosen to reflect the macroscopic behavior of the system. By describing these two concrete applications, we intend to illustrate how our generic model can be successfully applied to a larger class of problems.* © *1997 SPIE and IS&T.*

1 Introduction

1.1 *Introduction and Objectives*

This paper describes the use of particle systems as a generic model for simulations of dynamic systems. Simulation is one of the key problems in industry. The main objective is to reduce cost, duration and danger generally related to design, tests, and training of people. Starting from a given system S, we wish to obtain a model M such that:

a. the model M is a reduction of the system: by reducing the number of variables, one can achieve faster simulations, and manipulate the equations in an easier way;
b. the model M is an abstraction of the system: by replacing the real world by equations, one can predict the evolution of the system starting from given initial conditions;
c. the model M is simpler than the system itself: the simplification allows us to handle the problem by focusing only on what is really important, leaving out confusing details.

Properties of a good simulation model are:

a. the reduction conserves the essential properties of the system;
b. the simplification does not alter the precision of the results, which should be controllable;
c. the simulation parameters are easy to understand and control;
d. the model is able to give reliable predictions;
e. the model is able to adapt itself to system modifications.

There exist several approaches to modeling. These approaches can be divided in two classes: physically based modeling, and abstract modeling. The main interest for physically based modeling is that the degree of abstraction is low: the model is given in the same language as the system itself.

The same model can be physical for a given application, and abstract for another. The computational fluid dynamics (CFD) can be applied to compute some properties of the flow field (such as velocities or heat transfer). In this case, CFD is a physical model. On the other hand, if applied to simulate car traffic, CFD is an abstract model. A model with various applications is said to be generic.

The interest of generic models can be seen from two points of view. The customer, who is in charge of the sys-

Paper RMI-04 received Oct. 15, 1996; revised manuscript received Nov. 22, 1996; accepted for publication Nov. 22, 1996.
1017-9909/97/$10.00 © 1997 SPIE and IS&T.

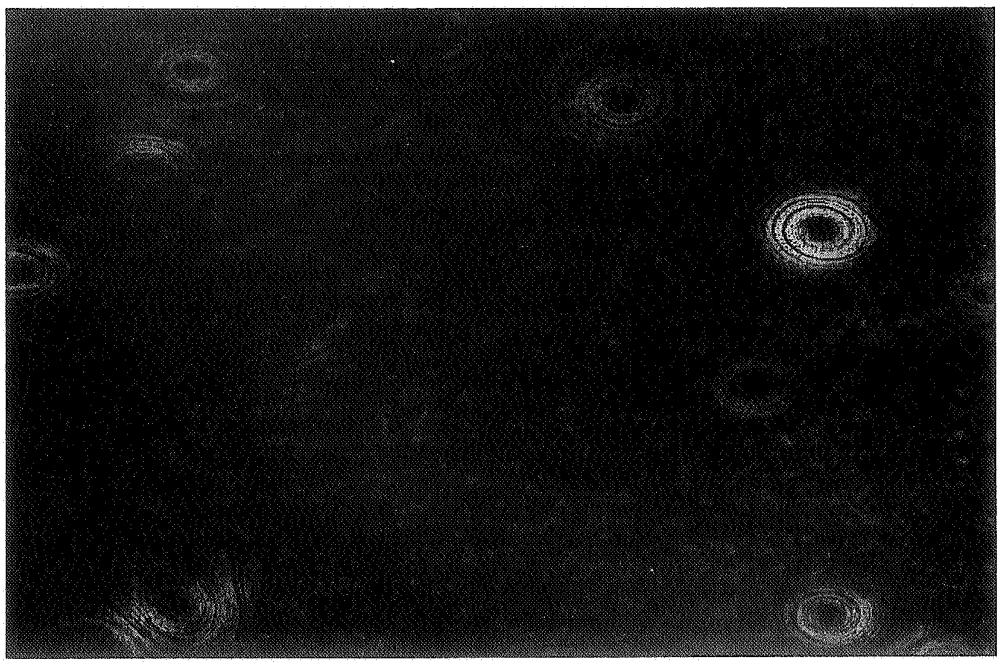


Fig. 1 Particles moving in an external field. The fastest particles are the brightest.

tem S, does not have to learn anything outside of his domain of expertise. He can rely on the services from a specialized software company. The developer, who is in charge of modeling, can amortize the heavy initial investment by recognizing the similarities of mathematical and numerical problems.

Applications of simulations, ranging from physical phenomena to virtual reality, are too numerous to be detailed here. We would like to focus on what we think to be a generic model for simulation, particle systems.

1.2 *Particle Systems*

Particle system technology addresses a growing need to simulate complex systems for predicting real world performance. The complex system is modeled as a collection of a large number (typically millions) of constituents (*particles*). Particles interact among themselves, with other particle sources and with surrounding surfaces or obstacles. They can be subject to external forces and coupled with a surrounding medium (such as a fluid flow). The results can then be output as a 3D rendered visual to view the system simulation results. More or less complex and complete models of particle systems have been developed, mainly for two classes of applications: prediction and visualization of physical phenomena, and computer graphics.

1.2.1 *Prediction and visualization of physical phenomena*

The ability to follow the motion of many particles is crucial to both theoretical and experimental studies. The simplest idea is to use particles as cork tangle by the flow, moving in an external field given by experimental or computed data (Fig. 1).

Applications of particle systems cover a wide range of physical phenomena[1]: plasma, astrophysics, phase transitions are some examples. One can even think of gaining insight to the early universe by visualizing for example the parton distribution inside the proton during the process of hadronization.[2,3]

Particles can be used in order to solve some fluid dynamics problems. To calculate the properties of turbulent reactive flow fields, some authors have developed a class of methods known as pdf (probability density function) which aims to solve the transport equation for the velocity-composition joint pdf.[4] Lattice gas hydrodynamics is a method aiming at solving the Navier-Stokes equation using particles constrained to a lattice.[5] Hybrid models (particles plus fluid) are becoming increasingly attractive.

1.2.2 *Computer graphics*

Particle systems were first introduced in computer graphics by W. T. Reeves in 1983[6] for the modeling and the visualization of natural fuzzy phenomena such as clouds, water, gases and fire. The principle consists of defining objects as clouds of elementary primitives, the particles, assigning to each of them physical attributes that determine their temporal evolution in terms of physical laws. Major difficulties are caused by the fact that modeling complex phenomena implies the ability to treat a great number of particles in interaction for many time steps.

Most of the studies on particle systems were initiated to confer a more realistic aspect to the animation of such fuzzy objects, by enriching the rendering algorithms with statistical techniques,[7] elaborated light models[8] or motion blur.[9] Interesting results were also obtained by borrowing

physical models to sculpt particle clouds.[8] Particle systems are a very competitive field of research in computer graphics.

1.3 Outline of the Paper

We want to introduce a new approach to particle systems, which is described in Section 2. This approach allows us to handle complex systems relying on Newton's law (equation of motion), mixed with interaction between particles. This rather general approach is perfectly suited to model a wide range of applications, abstract or physical. We will present two of these applications: crowd simulation (Section 3) and airbag deployment (Section 4). We will conclude (Section 5) by presenting work under study.

2 Particle Systems

2.1 A New Formalism

With the aim of explaining why we have designed a new formalism, let us describe what is called a collisional system, which are phenomena whose length scales are long with respect to the mean separation of physical particles.

Let Γ be the phase space $dxdv = dx_1 \ldots dx_n dv_1 \ldots dv_n x_i$ being space coordinates and v_i the velocity coordinates. The description of the particle system is formally given through transport equation applied to the probability density f in Γ space:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + v \cdot \frac{\partial f}{\partial x} + \frac{F}{m}\frac{\partial f}{\partial v} = C(f) \tag{1}$$

where $f(x_1, \ldots x_n, v_1, \ldots, v_n, t)dx_1 \ldots dx_n dv_1 \ldots dv_n$ is the probability that the system is in the volume $[(x_1, x_1 + dx_1) \ldots]$ of Γ space at time t, F is the total force (internal and external) applied at position (x,v). Equation (1) is obtained from the definition of a hierarchy of distribution functions starting from Liouville's conservation equation from probability densities in phase space, and integrating over successively fewer particles coordinates. This equation thus relies on an appropriate mathematical description of function f.

The left-hand side of equation (1) simply states that particles evolve according to Newton's Laws (equations of motion):

$$\begin{cases} \dfrac{d}{dt}x = v \\ \dfrac{d}{dt}(mv) = F. \end{cases} \tag{2}$$

The right-hand side of equation (1) ($C(f)$) formally expresses the rate of changes of f due to collisions and simply describes the effect of the graininess of the medium. If $C(f) \equiv 0$, there are no collisions in the system, which then satisfies the equations of motion (2). Differential equation (1) is rather difficult to solve explicitly in the general case. On the one hand, one has to find a stable numerical scheme dealing with $C(f)$, and on the other hand, this numerical scheme can be hard to put into play due to boundary conditions on the mesh.

It is rather difficult to generalize equation (1) to take into account other types of interactions such as for instance chemical transformations. In order to deal with the complexity of interactions, we have developed a formalism based on the transfer matrix, which allows us to solve in a stochastic way the transport equation. The essential idea is to discretize the system in time, and then to construct a Markov matrix (called the transfer matrix), which allows us to pass the state vector from one time step to the following time step. If there is no interaction, this approach amounts to solving the Newtonian equations of motion. As soon as there is some kind of interaction, the system can be solved using realizations of the Markov chain that we construct by our model. For this approach to be valid, the key point is to consider a time step longer than the typical time of interaction.

Note that particles are not merely infinitesimal points in the phase space. Using differential cross section, we can handle complex shapes by the transfer matrix, in a stochastic way. The whole approach is thus very efficient to deal with complex scenes, especially if we remark that for short range interactions, the transfer matrix is quasi diagonal. Using this property, one can implement on computers an extremely fast software, able to manipulate several thousands of objects interacting in real time. ArSciMed has developed such a software, called Kinema/Sim. We will next describe it more precisely.

2.2 Generic Description

Kinema/Sim is a software system allowing the statistical simulation of the dynamic behavior of a generic particle system. It is based on Newtonian mechanics and transfer matrix theory.

A particle system is defined by:

- the description of the particle types,
- the particle sources which generate the particles,
- the 3D geometry, including obstacles,
- the evolution of these particles within the system.

To define each one of the system parameters, Kinema/Sim can use probability distributions or time-dependent functions. Once the system is defined, the simulation consists in computing the evolution of its state over the time steps.

2.2.1 Description of particles

A particle object belongs to a class (or type) defined by the following physical characteristics:

- its mass, leading to a gravitational force, $\vec{F} = m\vec{g}$,
- its lifetime (deterministic or stochastic): leading to either disintegration or disappearance,
- its diffusion properties (random perturbations of position, velocity and/or acceleration),
- its charges, i.e. coefficients with respect to the electric and magnetic fields:
  $\vec{F}_{e-m} = q_e\vec{E}(x_i, y_i, z_i, t) + q_m\vec{v}_i \times \vec{B}(x_i, y_i, z_i, t)$,
- its transformation probabilities for particles closer than a given range,

- its drag with respect to the surrounding medium: $\vec{F}_d = -k(\vec{v}_i - \vec{v}_{medium})$
- its values for interactions with surfaces (stick, bounce, penetrate, transform, etc.),
- its cross section (for self collisions),
- its visualization parameters: color, size, transparency, trail memory, geometry.

All parameters may vary either as a function of absolute time, of the particle's age after leaving the source, or following a population curve. For example, one can introduce a white noise or have particle mass vary as a Gaussian distribution.

2.2.2 *Generation of particles*

Generating particles implies the description of an initial state for the system, by defining particles of different types, with imposed positions and velocities. During the simulation, the interaction of these particles with the system will change these initial values, but the user will have the possibility to create new particles, with defined position and velocities.

The particles are generated by sources. Sources are geometric entities emitting only one type of particle. They are defined by:

- their position in the space and their dimension (0,1,2,3),
- their size and geometry,
- their rate of emission as a function of time,
- their direction of emission: a given vector, a local normal to a surface, or a given trajectory.

Sources can move during time, following user-defined trajectories.

2.2.3 *Evolution of particles*

The evolution of position, velocity or type of a particle can be affected by:

- spontaneous transformations of particles,
- gravitational, electromagnetic fields,
- drag fields defined in the system,
- elastic or inelastic collisions with other particles or with obstacles,
- transformations or reactions for particles closer than a given range.

The inertial mass gives the relation between the acceleration of the particle and the sum of the forces exerted on it, before the introduction of the random effects:

$$\vec{\gamma} = m^{-1} \sum \vec{F} + \text{random}$$

where

$\vec{F}$ are the forces exerted on the particle,
m is the mass of the particle,
$\vec{\gamma}$ is the acceleration of the particle.

It is then possible for the system to compute the position of the particle from the knowledge of its initial position and velocity and the definition of forces at each time step.

$$\text{Velocity: } \vec{v} = \int \vec{\gamma} + \text{random},$$

$$\text{Position: } \vec{x} = \int \vec{v} + \text{random}.$$

Newtonian mechanics is here complemented by the transfer matrix formalism. It adds randomness to the trajectories of particles, by defining states associated with the system, and matrix functions that define the probability to evolve from a state i at time t to a state i' at time $t + dt$. Transitions of state correspond to certain events that can be global (independent of particle position) or associated to local distributions of particles. Particle positions, velocities and accelerations can also be perturbed by noises.

The state (state_i) of particle i at time t:

$$\text{state}_i(t) \equiv [x_i y_i z_i, v_{x_i} v_{y_i} v_{z_i}, \gamma_{x_i} \gamma_{y_i} \gamma_{z_i},$$

$$\text{color}(r,g,b)_i, \alpha_i, \text{mass}_i, \text{charges}_i, \text{size}_i, \text{lifetime}_i, \ldots .]$$

where

$x_i y_i z_i$ is position of particle i,
v is the velocity of the particle,
$\vec{\gamma}$ is the acceleration of the particle,
α is the transparency of the particle.

The key parameter set by the user is the range (R) of forces since particles are assumed to interact only within a certain range. The user also sets the size of the simulation time step Δt where Kinema outputs its data to the user. Kinema divides the simulation space into a 3D grid structure or a set of cubes of size R. If two particles are more than one cube (R) away from each other then interactions are ignored.

The state of particle i at the first time step ($t + \Delta t$) computed by Kinema is then based on the state of all the particles N_p within range R of i to take into account interactions:

$$\text{state}_i(t + \Delta t) = f[\text{state}_{1,2,\ldots,i,\ldots,N_p \text{ within} - R_i}(t)].$$

Therefore, if the simulation has a particle velocity and a simulation time step such that $v\Delta t \geq R$, Kinema may decide to compute internally at a much finer time step. This guarantees that one does not miss the effect of particle interactions between fast moving particles that occur between successive simulation time steps.

The overall state of the system (STATE) at time ($t + \Delta t$) is then based on the collection of all the individual

particle states at time t. In Kinema, local effects in a part of the system can therefore propagate through the entire system.

Kinema is more than just a deterministic model since it can compute the probability of an event happening since there is jitter, noise and other randomness. If there is a non-zero probability, then a random number generator determines the outcome. The user has control over the seed used in the random number generator.

For example:

$$\text{Prob}[\text{state}_i(t+\Delta t)=\text{STATE}]$$
$$=f[\text{state}_{1,2,\ldots,i,\ldots,N_p\ within-R_i}(t),\text{STATE}].$$

Transfer matrices are also used for the modeling of collisions between particles. Collisions concern sets of particles and are defined by:

- classes and numbers of particles that can collide,
- cross sections: function that define the probabilities for a particle to emerge in given directions and speeds.

Kinema/Sim allows the modification of the particles' state when they meet an obstacle. This obstacle object is defined as surfaces or volumes (2D or 3D) that are capable of reacting, as the particles do, to the forces in the system, but that possess, in addition, a rigidity or elasticity aspect. The user controls the outcome of particle-obstacle encounters by first defining the obstacle by:

- its position in the 3D space,
- its geometry,
- its size,
- its orientation.

The effect of an obstacle can either be global (same effect for all the particles) or specific to each type of particle. The user then sets the different available transformation options:

- bounce or rebound (the surface may also rebound due to momentum transfer),
  $v_{\parallel}^{\text{final}}=d_{\parallel}v_{\parallel}^{\text{initial}}$
  $v_{\parallel}^{\text{final}}=-d_{\perp}v_{\perp}^{\text{initial}}$
- adherence or painting where the particle sticks to the surface and follows its motion,
- disappearance with or without decay into other particles,
- ignoring the surface or volume for transmission through the obstacle.

One can also specify equations to define various drag fields, gravitational, and electromagnetic fields. These fields can depend on position and time and operate in a grid basis, or on the actual position of the particle.

When the available information is partly kinetic and partly dynamic, Kinema allows the user to draw the average trajectories. The forces generating those trajectories will be calculated by the computer.

Kinema/Sim can display and store the following data from a simulation:

- individual: data concerning particles characterized by their identity number or their behavior,
- collective: data concerning classes or an ensemble of objects,
- statistical: such as average velocities and the moments of their distributions,
- local: such as the number and velocities of particles traversing the surface of a detector,
- images including volume rendering.

2.2.4 *Parallelism*

Because of the parallel nature of particle systems, they are well suited for highly parallel computation. First studies were proposed on a Connection Machine system,[9] but these studies did not take into account interactions between particles.

The approach that we developed at ArSciMed[10] consists of three steps: distribute the 3D world between a set of processors, then compute the motion of particles on each processor, and finally make the particles migrate from one processor to another. The principal difficulty in this algorithm lay in the dynamic allocation of the 3D world on the set of processors: we want this allocation to preserve on the same processor (or at least on neighbor processor) neighboring particles. We solve this difficulty by a linear configuration of the set of processors, and by a dynamic load balancing.

A PVM (parallel virtual machine) version of this algorithm has been developed allowing a quasi linear acceleration in performance as a function of the number of processors. A multithread version is available for shared memory processor (such as multi-pentium on Windows NT) that leads to linear acceleration.[11]

2.2.5 *Advantages of the Kinema approach to particle systems*

For the needs of scientists and engineers, Kinema/Sim offers powerful capabilities in visual modeling, statistical data collection and time-dependent parameter set ups. In addition, Kinema/Sim offers a number of specific advantages over competing particle systems. The system can:

- handle collisions of particles with objects, surfaces and with other particles,
- handle external forces such as the effect of gravity, electricity or magnetic fields on particles or surfaces,
- handle the influence of surrounding medium such as air, water or viscous fluids, with turbulent dynamics,
- manage the interaction of up to typically hundreds of thousands of particles interactively,
- provide volume rendering, in terms of the spatial density distribution of particle data, from either Kinema or other systems,
- manage the position of sources and emission parameters (rate, direction, speed).

This generic approach can be applied to a wide variety of problems (Fig. 2). We will first describe the crowd simulation.

3 People Flow Simulation

Most methods of people flow simulations are based on a global approach. By calculating the projected surface on the ground occupied by each individual, and the surface of the ground, it is possible to compute the density of the stream:

$$\rho = \frac{\sum_{i=1}^{n} s_i}{\sigma} \tag{3}$$

where

s_i is the projected size on ground of individual i,

σ is the total surface on ground occupied by the stream.

Empirical laws were obtained, linking velocities of pedestrians to density. For very simple situations like streams of pedestrians along a straight way,[12] the average velocity is given by:

$$\|\bar{v}(\rho)\| = 57 - 217\rho + 434\rho^2 - 380\rho^3 + 112\rho^4. \tag{4}$$

It becomes much more difficult, even unrealistic, to adopt such an approach to model complex situations, when the pedestrian flow is constrained by a complex environment and when the flow is composed of heterogeneous individuals. Besides, the microscopic approach allows us to simulate detailed behavior patterns, not just averages. The abundant empirical data is however very useful for the validation and the calibration of the microscopic model.

Another approach consists of adapting hydrodynamics tools by considering pedestrian stream as a fluid.[13] The main problem is that human flow is not a continuous medium (i.e. people in a small area can have very different behaviors) and the corresponding fluid is therefore highly turbulent. Classical fluid dynamics is not very useful in these cases.

3.1 *From a Physical Particle System to the Simulation of Pedestrian Flow*

Particle systems were adapted for studying crowd movements. For low density situations, Henderson[14] used a model based on kinetic theory of gases. For a high density situation, Peschl[15] modeled panic situations by particle streams.

In our approach, human beings are modeled as an interacting set of particles.[16] It was necessary to consider interactions between particles so that the human behavior could be simulated.

Using our three-dimensional particle system, we have developed a "2 1/2-dimensional" (where space is a collection of planes) version called "Kinema/Way" for the simulation of pedestrian flows.

Since the motion of people is based on Newtonian forces, the definition of these forces is critical to our model. Forces can be constant for the whole system (for instance gravity) or caused by the presence of local fields restricted to one plane of the 2 1/2 system:

$$\vec{\mathbf{F}} = q\vec{\mathbf{E}}$$

$\vec{\mathbf{F}}(x,y)$ = Resulting force at point(x,y).

$\vec{\mathbf{E}}(x,y)$ = vector field at point(x,y).

q = charge. (5)

Fields are defined by placing different kinds of objects in the scene. They can be "attracts," or "local lines." If A is an attract positioned at point Pa with a field strength of Sa, a particle at position P will be subject to the field:

$$\vec{\mathbf{E}}_a = Sa\,\frac{\vec{\mathbf{P}}_a - \vec{\mathbf{P}}}{1 + a_f|\vec{\mathbf{P}}_a - \vec{\mathbf{P}}|^2} \tag{6}$$

with

$\vec{\mathbf{E}}_a$ = attraction field,

a_f = attenuation factor (drop rate).

Local lines correspond to geometric segments and define "line" fields by the formula:

$$\vec{\mathbf{E}}_{fl}(x,y) = S_{fl}\,\frac{\vec{\mathbf{T}}(x,y)}{1 + a_f d(x,y)^2} \tag{7}$$

where

$\vec{\mathbf{T}}(x,y)$ is the tangent to the local line $L(x,y)$ at point (x,y),

$d(x,y) = \min_{(xl,yl)\in L(x,y)} (\sqrt{(xl-x)^2 + (yl-y)^2})$,

$L(x,y)$ is the nearest local line from point (x,y).

The application of constant forces to particles would cause constant accelerations, not constant velocities as needed for human displacements. We have introduced, therefore, a parameter to simulate the friction. This is done by decreasing the acceleration of a particle by a vector $\vec{\mathbf{D}}$ which is calculated from the previous velocity of the particle [Eq. (8)]. For a constant acceleration, the velocity of a particle will increase until a constant velocity is reached

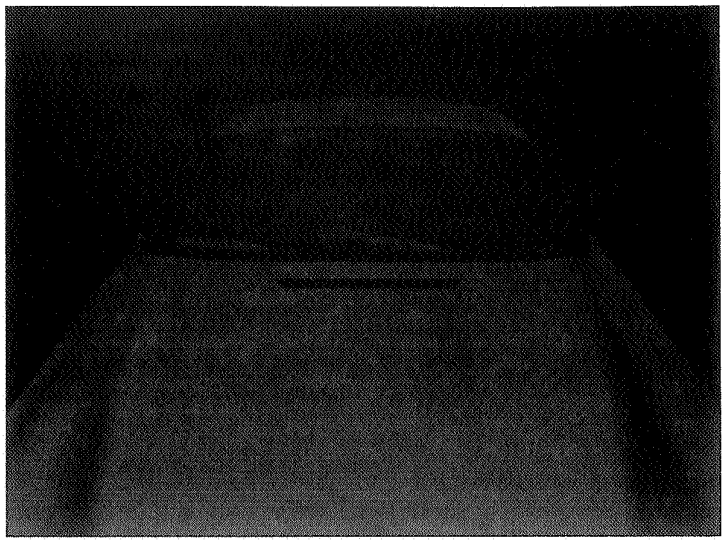
$$\vec{\mathbf{D}}(t+dt) = -k\vec{\mathbf{v}}(t) \tag{8}$$

where

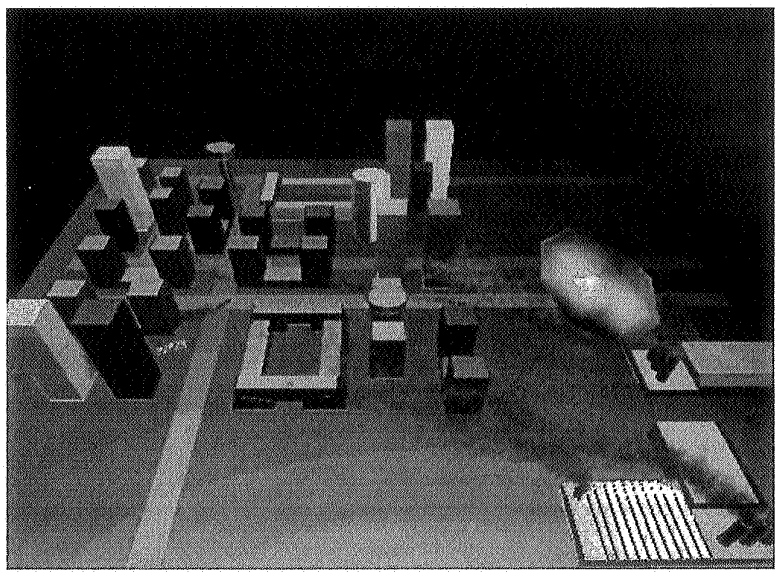
$\vec{\mathbf{v}}(t)$ is the particle velocity,

k is the friction coefficient.

Particles enter the scene by sources that are described by their shapes, positions, directions and rates of emission. The "2 1/2-dimensional" scene is composed of a set of 2-D planes that can be animated in the 3-D system by ani-



(a)



(b)

Fig. 2 Two examples of particle system applications. (a) Simulation of rain on a car with moving windshield wipers. (b) Simulation of urban pollution.

mation of positions and rotations of references. The motion of the particles is restricted to these planes with possible communications between them by predefined regions. This enables the simulation of places with several stairs, moving staircases, vehicles, etc.

3.2 *Human Behavior and External Parameters*

The simulation of pedestrian behavior has proven to be a very useful application of interactive particle systems. We have found many analogies between the displacement of particles in such a model and the behavior of individuals in a crowd. The concept of particle classes is translated to people with a similar behavior pattern, so that defining a set of characteristics enable us to take into account very large numbers of people with minimal effort. Typically, classes of people can be assigned to sex, age, strength, and physical state.

The use of physical parameters for the definition of the different classes makes them easy to adjust because they correspond to real characteristics of people. For instance mass associated to a class of particle will correspond to the real mass of the individual. Another example concerns the typical behavior of drunken fans (encountered in a concert or during sports events), modeled by adding noise to their positions or their velocities.

In a public gathering, people often come in groups, which try to stay together. The cohesion force of such groups may vary from rather weak values (fans of the same

sports team), to a very high one (family). For the sake of the simulation, we treat the latter case as an extended individual, while the visualization takes into account all the individual members. A weakly coupled group may be treated as one unit as long as the separating forces are smaller than a given threshold. Beyond that threshold, the group may break up to a set of smaller groups and individuals.

We divided the modeling of human behavior into:

- reflex (immediate) reactions,
- inference reactions (decisions), resulting from knowledge of the scene.

3.3 *Modeling of Reflex Reactions*

In our model, reflexes concern the ability to avoid other people as well as obstacles detected at the last moment. These reflexes must be interpreted as very local phenomena.

The way people avoid each other depends on the density. For low density, the direction of motion is activated at a certain distance and collisions are rather exceptional; as the density increases, the collisions become more frequent and can involve more than two individuals at a time. Our avoidance model can treat situations where $n1 + n2$ persons of two different classes are confronted. It consists of defining a collision rate and a distance that determine the vicinity for avoidance detection (Fig. 3). The number n of avoidance events during time dt, between two particles of velocities $\vec{\mathbf{v}}_1$ and $\vec{\mathbf{v}}_2$ from classes 1,2 in the area S can be computed as (for instance):

$$n = dt \;\; rate \;\; v_1 v_2 S \|\vec{\mathbf{v}}_1 - \vec{\mathbf{v}}_2\| \tag{9}$$

with

$rate$=collision rate

v_i =number of particle of class i present in S/S.

The velocity directions are modified during avoidance by an angle depending on the relative directions of incoming particles. The speed may be modified by a multiplicative parameter.

Collisions between more than two particles can be taken into account in two ways:

- A sequence of two particle collisions during adjacent time steps.
- A sudden event involving all participants at the same time.

In all cases, the time sampling of the simulation must be adjusted according to the pedestrian velocity in such a way that collisions are not missed because of intervals that are too long. The way people avoid solid obstacles like walls is modeled as a real collision:

$$\vec{v}(t_{after}) = \alpha(-\vec{v}(t_{before}) \cdot \vec{N})\vec{N} + \beta(\vec{v}(t_{before}) \cdot \vec{T})\vec{T}. \tag{10}$$

$\vec{N}$ is the Normal to obstacle at intersection point.

$\vec{T}$ is the Tangent to obstacle at intersection point.

α is the normal damping coefficient.

β is the tangent damping coefficient.

3.4 *Modeling of Decisions*

A more difficult problem is to model human decisions. These decisions depend on a set of parameters: destination, density, intended duration of stay at a particular point, presence of obstacles.

Our approach consists of associating with each person a state that defines his or her reaction as a function of what he or she perceives. Some of the perceived events can be translated to displacements and others can produce a change of state. A state is composed of a set of charges that we call "decision charges" that are influenced by fields that we call "decision fields." These notions can be compared to electric charges and electric fields in the sense that a particle with an electric charge will be influenced by an electric field in the same way as a person with a "decision charge" confronted to a "decision field." The reaction strength is proportional to the charge, so that different behavior may be obtained by adjusting charges. Members of class 1 may react very strongly to the corresponding decision field while those of class 2 may be indifferent to it.

The notion of goal is well represented by decision fields. As we have seen an electric field can be defined by particular objects that we called "attracts." In the same way we can add "decision attracts" to the system that will materialize goals for individuals who have a nonvanishing "decision charge." It is thereby possible to define paths associated with people by specifying key positions (attracts).

Another important point is the way people avoid obstacles. The reflex reaction was defined before. Under normal conditions, we use the concept of "local line" fields. To each obstacle in the system, a set of local lines will be associated, creating a "local decision field" parallel to the tangent to the obstacle (Fig. 4).

The local field is calculated by

$$\vec{\mathbf{E}}_l(x,y) = \frac{\vec{\mathbf{T}}(x,y)}{1 + d_r d^2} \tag{11}$$

where

d_r *is an attenuation factor (drop rate)*

d *is the distance to the nearest obstacle and people*

$\vec{\mathbf{T}}(x,y)$ *is the direction of tangent to nearest obstacle of position* (x,y).

3.5 *Critical Situations*

A situation becomes critical when the decisions of an individual person can no longer influence directly his or her motion. It occurs when the density becomes very high, at least locally. Depending on the particular situation, and on the characteristics of the classes of people present, they will try to avoid the density peaks. The avoidance of a local high density place is simulated by assigning to this place a negative attract field that repulses arriving people.

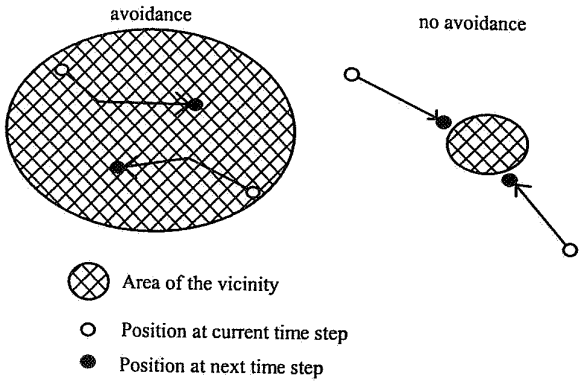


Fig. 3 Effects of decreasing vicinity in collisions.

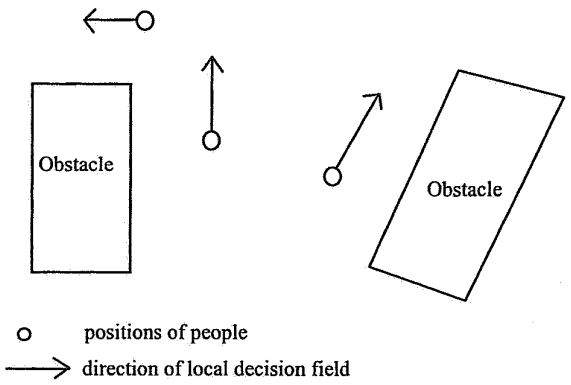


Fig. 4 Direction of local decision fields as a function of the obstacles.

Inside the high density region, people will try to escape in the direction of lower density. This is simulated by applying a force depending on the density gradient:

$$\vec{F}_{cr}=k_{cr}\vec{\nabla}D \tag{12}$$

with

k_{cr}=coefficient of strength of individual
D =density field.

An evacuation planning can be validated by tracing "local lines" corresponding to emergency paths. By disabling all charges, except for the one that corresponds to these "local lines" as evacuation begins, people will act as in real situations trying to reach exits following indicated paths. The evacuation process can thus be observed in detail, including its duration and the presence of dangerous passages.

3.6 *Visualization*

All simulations can be visualized with different quality of rendering. People can be displayed as simple geometric primitives or following much more realistic models with shading. It is also possible to display trajectories by recording previous positions. The point of view can be fixed or in motion and can be associated to a selected person so that the simulation is perceived from inside the crowd.

3.7 *Conclusions*

Section 3 describes a microscopic approach to the simulation of human behavior in the presence of crowds. Each person is a simulation object ("particle"), so that the crowd becomes an interacting particle system. The only elements of that behavior which are directly simulated are those related to motion (position, velocity, acceleration). The formalism used is Newtonian mechanics, enhanced with events to which probability laws are assigned. It is therefore rather similar to a Boltzmann gas.

The main advantage of a microscopic model is that it provides a unified framework for dealing with situations that may differ significantly in the macroscopic (or global) appearance.

We have applied the above model to various types of situations, ranging from leisure shopping to catastrophic overdense crowds. The behavioral parameters are easy to adjust because they correspond to intuitive values for humans. The validation of these parameters is done by applying the model to well-known situations and with iterative adjustment of behavior until empirical results are reproduced.

In the context of emergency, we typically obtain results that are more realistic compared with that which could be calculated with a global approach, which tends to underestimate the risks. This can be explained by the complexity of the system that engenders chaotic events. We have observed phenomena like shock waves and very local high densities that represent extreme danger, in situations judged acceptable by a global approach.

Our software CPU performance has permitted the computation of simulations with up to 45,000 persons during real periods of about one hour, in approximately 7 hours computing time on a Silicon Graphics Indigo R4000 workstation. In addition, we have developed a parallel version of the software that we have applied[17] to the real-time simulation of the Stade de France. It was demonstrated at the Imagina 96 show, using an SP2 with 16 processors R6000 Power 2, and allowed 45,000 persons per second to move in an immersive environment provided by the software 3DIX from IBM (Fig. 5).

Future works will permit the association of the initial conditions of the simulation with a real situation as perceived by sensors and cameras. Computing the evolution of the system faster than real time will allow testing different solutions to prevent catastrophes, or to help alleviate them while they are occurring.

4 Airbag Deployment Simulation

4.1 *Introduction*

ArSciMed is currently applying its particle-based simulation and visualization tools to the airbag deployment problem.

Both the gas mixture and the bag are considered to be composed of a large number of particles. Our simulation software can handle an a priori unlimited number of particles. That number is however bounded by considerations related to the available memory and CPU performance. It is therefore clear that the particles are not to be identified with the molecules of the gas or the bag. They are instead ab-

stract building blocks, whose dynamics are chosen to reflect the macroscopic behavior of the system.

The natural time scales of the molecular system are of the order of 10^{-8} seconds, the typical time interval between collisions. For obvious reasons, the time step of the simulation must be significantly longer. Each of these steps should therefore integrate the effects of many molecular events.

Our approach consists therefore of replacing the molecular content of the system by a different "microscopic" description, whose coarse-grained space-time behavior approximates the macroscopic dynamics. Note that the gas in the bag is a mixture, and contains various molar fractions $r(i)$ of gas type i.

We have developed a hybrid approach in order to solve this problem. We will first see that neither the simple ideal gas approach, nor the particle system approach are sufficient by themselves to give a solution to the bag deployment problem. The first one is too simple to give realistic results, while the second one can be heavily computational time expensive. So we will combine these two approaches to obtain a valid model.

4.2 *Phase Space*

Define the phase space element $d\tau = d^3x d^3v d\tau_{int}$, where x is the positions, v the velocities, and $d\tau_{int}$ corresponds to the internal degrees of freedom (such as rotations, vibrations or orbital excitations).

The spatial distribution of the molecules is given by

$$N(x,t) = \int f(x,\Gamma,t)d\Gamma, \text{ where } d\Gamma = d^3v d\tau_{int}.$$

4.3 *Ideal Gas in Thermal Equilibrium*

Assume a homogeneous gas entirely composed of molecules of type i in thermal equilibrium. The equation of state is $PV = NkT$, where k is the Boltzmann constant, P the pressure, V the volume, N the number of particles, T the temperature.

The velocity distribution of the particles follows the Maxwell-Boltzmann expression:

$$f_0 = n \exp(-m/2kT(v-v_0)^2),$$

where n is the particle density N/V, m the mass of the molecule, v_0 the translational velocity of the gas.

The most probable velocity (relative to the overall translational velocity) is $\bar{v} = v_{mp} = \sqrt{2kT/m}$, and the root mean square velocity is $v_{rms} = \sqrt{3kT/m}$.

4.4 *The Particle System Approach*

We wish to represent the gas mixture by a system containing as many particles as can be reasonably handled by a computer, say a few millions, while keeping the macroscopic quantities approximately unchanged. There is a good theoretical reason to believe that this can be done. The opposite would imply that by purely macroscopic measurements one would be able to derive the real number of molecules in the system, and therefore infer the atomic scale of mass, which is contrary to physical intuition.

Assume that we replace N molecules by N_p particles, and define $z = N/N_p$. By fixing the total mass, which is a macroscopic quantity, we find $m_p = zm$. Apply this transformation to the ideal gas. If we fix P, V, and T, we find that in the particle system world, the Boltzmann constant would be much higher, $k_p = zk$. This is not surprising, since k in the real world is a tiny number when expressed in macroscopic units.

Another, more fundamental, justification for the above relation, and which defines its limitations and extensions, results from the definition of the entropy: $S = k \log(N_s)$, N_s being the number of microscopic states compatible with the macroscopic quantities.

For large N, $\log(N_s)$ is proportional to N, and we confirm the relation $k_p = zk$. Corrections to the simple proportionality rule lead to a systematic expansion of the particle system parameters as a function of z^{-1}.

To leading order, we find $k_p/m_p = k/m$, and the Maxwell–Boltzmann distribution is therefore identical for the gas and its particle system representation.

4.4.1 *Particle collisions*

The mean free path is the average distance a molecule travels between two successive collisions, and τ the average time between collisions. They are given by:

$$\lambda = \frac{1}{4}\sqrt{\frac{\pi}{2}}\frac{1}{n\sigma}, \tau = \frac{\lambda}{\bar{v}}.$$

σ is the cross section for inter-molecular collisions, which is of the order of the surface area of the molecule.

Typical values for these parameters for light molecules in room temperature are:

$$\bar{v} \approx 10^3 \text{ m/sec}, \lambda \approx 10^{-5} \text{ m}, \tau \approx 10^{-8} \text{ sec}.$$

As we have seen, a particle system representation would have the same average velocities. We should now deduce the values for the collision parameters for that system, based on macroscopic quantities related to particle collisions. Those quantities are the thermal conductivity and the viscosity.

Define the energy flux as

$$\mathbf{q} = \int \epsilon \mathbf{v} f d\Gamma,$$

where ϵ is the energy. Assume that a small volume of the gas is moving as a whole with velocity $\mathbf{V}$. We place ourselves in a co-moving frame, $(\,)'$. The velocities of the individual molecules in the original frame are $\mathbf{v} = \mathbf{v}' + \mathbf{V}$.

$\mathbf{v}'$ follows the Maxwell–Boltzmann distribution. The energies in the two frames are related by

$$\epsilon = \epsilon' + m\mathbf{V}\cdot\mathbf{v}' + \tfrac{1}{2}m\mathbf{V}^2.$$

Therefore

$$\mathbf{q} = \mathbf{V}(\tfrac{1}{2}pV^2 + W), \quad W = N\bar{\epsilon}' + P$$

is the heat function per unit volume.

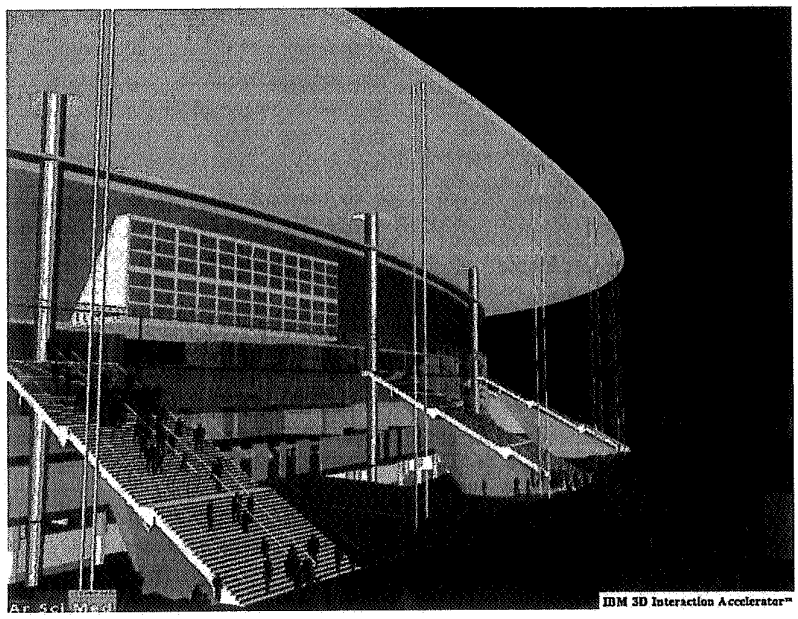


Fig. 5 Simulation of a crowd inside the Stade de France by the software Kinema/Way (Architects: Macary, Zublena, Regembal, Costantini).

The thermal conductivity κ is macroscopically defined implicitly by

$$\mathbf{q} = -\kappa \nabla T.$$

The transport equation implies $\kappa \propto k\bar{v}/\sigma$, which should be equal to the particle system value $\kappa_p \propto k_p \bar{v}_p/\sigma_p$. We find $\sigma_p = z\sigma$.

Similarly, the kinetic theory determines a value for the viscosity $\eta \propto m\bar{v}/\sigma$. From the macroscopic constraint $\eta_p = \eta$, we rederive $\sigma_p = z\sigma$.

The mean free path $\lambda_p \propto 1/(n_p\sigma_p) = 1/(n\sigma)$, and therefore $\lambda_p = \lambda$. Since the average velocities are the same for the molecules and the corresponding particles, we conclude $\tau_p = \tau$.

The above results imply that it is possible to represent the gas as a particle system with a relatively small number of particles, but that the natural length and time scales of the problem would be unchanged. The time interval for the simulation is 30 msec. With a time step of 10^{-8} seconds, we would have about a million steps, which we cannot achieve in a reasonable computing time. We are therefore forced to sum over the individual collisions and develop a time integrated formalism.

Our approach is to consider the gas as a hybrid system containing a fluid in the classical sense (obeying fluid dynamics equations of motion) and particles in interaction with it. Both elements correspond to the same physical system, but they represent different aspects of it. The fluid represents the part which is in thermal equilibrium, while the particles are not necessarily in such a state. This approach has the advantage of dealing both with the turbulent and near-equilibrium aspects of the problem. As our research evolves, we will fine-tune our model so that it describes as closely as possible the gas in the bag.

4.5 *The Hybrid Approach*

Consider each one of the gases in the mixture to be made up of two components:

1. A gas in thermal equilibrium (the "background gas," or BG). Since we will not treat the BG particles individually, we may as well take them to be the constituent molecules.
2. A collection of heavy particles (the "particle gas," or PG) interacting often with the background gas and rarely among themselves. Each one of the heavy particles represents a large number of molecules. The interactions are insufficient for achieving thermal equilibrium for the heavy particles. The definition of the two components is local, so that transitions between the states are allowed.

4.5.1 *Collision in a mixture of gas*

We compute the change of momentum between the particle gas and the background gas, which act as a drag force on the heavy particle. This leads us to define the drag coefficient K implicitly by

$$f_{drag} = -K(\mathbf{v}_P - \mathbf{V}_{BG}),$$

where K can be expressed in terms of microscopic quantities. With typical values for an ideal gas, and $N_P \sim 10^5$, we get $K \sim 1$ (Joule·sec)/(meter)^2.

4.5.2 *The background as an ideal gas*

The simplest imaginable fluid model is an ideal gas at rest. This model may naturally be improved later, but it is already highly nontrivial in the hybrid model context. We therefore assume $\mathbf{v}_{BG} = 0$, and that the temperature, pres-

sures and densities are homogeneous in the bag. These quantities will be determined by the ideal gas equation of state, as well as the energy balance.

The BG molecules are assumed, due to the large Maxwell–Boltzmann velocities, to fill instantaneously the available (unfolded) volume in the bag. In the hybrid gas model, we are not obliged to represent the BG by particles. It is, however, rather convenient to do so for "accounting" purposes (counting the number of particles, the total mass, energy, etc.). We take the ratio z to be the same for BG and PG.

The total internal energy of a gas constituent in BG is therefore

$$E_{BG}(i) = \tfrac{3}{2} z N_{BG}(i) k T_{BG}.$$

From the emission rate function of the gas generator one may in principle (we will return to this question later) infer a partition:

$$R(i) = R_{BG}(i) = R_{PG}(i), R(i) = \frac{\Delta N(i)}{\Delta t}.$$

In the course of the bag inflation, particles of the PG, once their velocities become small enough (say $p\%$ of the most probable Maxwell–Boltzmann velocity) may be identified with the BG. This will not modify the total number of particles. The energy of a PG particle is the sum of its kinetic energy and the internal energy of the gas it represents:

$$E_{PG}(i) = \frac{1}{2} z m(i) \sum v_P^2 + \frac{3}{2} z N_{PG}(i) k T_{PG}.$$

If $p\% \ll 100\%$, the second term on the RHS is much smaller than the first, a necessary condition for the particle representation to be coherent.

4.5.3 *The dynamics of the surface*

The surface of the bag is a collection of strong interacting particles. The force on a given surface element is the sum of four contributions:

$$\mathbf{F} = \mathbf{F}_{BG} + \mathbf{F}_{PG} + \mathbf{F}_{SC} + \mathbf{F}_{EL}.$$

The first three contributions are computed as follows:

- $\mathbf{F}_{BG}$ is the pressure force exerted by the background gas.
- $\mathbf{F}_{PG}$ is computed from the total momentum transferred to the surface by the particle collisions and from the local pressure force exerted by the PG particles.
- $\mathbf{F}_{SC}$ is the force exerted on the surface element due to collisions with other surface elements. It is computed from the momentum transferred during the collision of the two surfaces, assuming damping coefficients for the relative normal and tangential relative velocities.
- $\mathbf{F}_{EL}$ is the elastic force due to local deformations of the surface. We apply the theory of finite element shells as the theoretical framework for the elastic properties of the bag.

4.6 *Conclusion*

We enclose (Fig. 6) one simulation of a hybrid system. Let us notice that the particle approach is very different from the ideal gas one, both numerically and visually.

Very little information is known on the airbag problem. This is one case where most of what we can learn comes from visual experiments (analysis of ultra-high speed films). Indeed, as the airbag deployment is extremely fast, it is rather impossible to measure such parameters as temperature inside the bag for instance. It is then useful to obtain films, both from simulations and from experiments, and to compare them.

The hybrid approach is a framework which allows us to link continuously macroscopic models to microscopic ones. In the airbag problem, the beginning of the inflation is highly unstable, and particle systems are well suited to provide a model for highly unstable problems. On the other hand, the end of the inflation is correctly represented by an ideal gas model. We thus use the model which is the most adapted at a given time during the simulation.

5 Further Simulations under Development

Particle systems can be used to solve a wide variety of problems. Medical nuclear imaging and surface etching are two of the fields we are currently investigating.

Medical nuclear imaging consists of representing the distribution of radioactive sources into a human organism. A weakly radioactive product is injected into the patient's body, in association with a molecular tracer, that has a tendency to concentrate in the studied organ. A scintillation camera can then detect the gamma-ray photons emitted by the radioactive sources, and represent their topology in an image. A simulation of the acquisition process has been developed, using ArSciMed's particle system.[18] Obstacles and sources management facilities enable us to model different kinds of phantoms (scattering environment shapes and radioactive spatial distributions). A photon transport model based on quantum physics has been added. It describes interaction processes between photons and material described by atomical or molecular properties. These effects are photoelectric absorption, incoherent scattering (Compton effect) and coherent scattering. The results we obtain compare well to other published simulation results, due to an explicit collimator simulation that we have developed. These results will soon be extended by experiments with various sources of different radioactivity levels, and of more complex shape.

Simulation of semiconductor device manufacturing, and in particular surface etching, is another field where the same kind of technology can be applied. Numerous processes involved in the deposition and etching of thin films and bulk materials rely on the use of a flux of particles (ions, electrons, photons,...). This is particularly the case in the fields of microelectronics, nanotechnology and optics. A currently interesting case is the etching of a substrate using plasma or ion-beams to erode the substrate (physical erosion) and/or the existence of a chemical reaction (chemi-

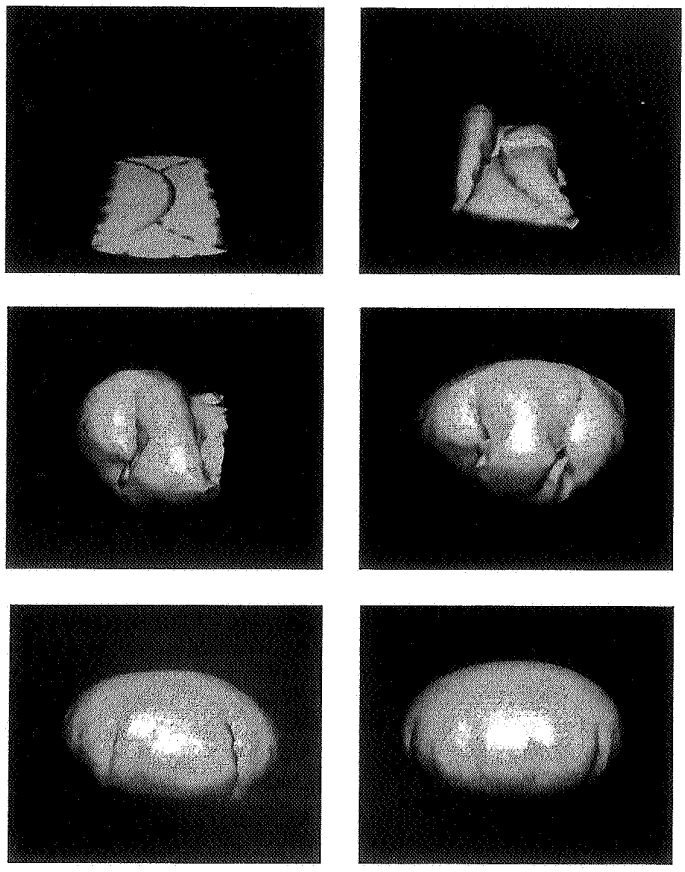


Fig. 6 Simulation of an airbag deployment. The gas inside the bag is a hybrid particle system, and the surface of the bag is a collection of particles with strong interactions.

cal etching). The introduction of a surface reactive species may be used to increase the etch rate and to control the etch anisotropy. The theory of surface erosion due to impact of energetic ions (Sigmund theory) has been confirmed in various cases by experiments and Monte-Carlo simulations. The physical mechanisms involved in reactive gas etching are complex and theoretical modeling remains difficult. This is why existing simulation tools take them into account from an empirical standpoint. A first interesting and practical objective is to be able to simulate the evolution of a surface of arbitrary topography under the action of an ion beam, taking into account characteristics of the beam (energy, energy dispersion, angular dispersion,...).

6 Conclusions

We have developed in this paper a generic method based on particle systems, which aims at simulating complex systems of dynamic interacting objects. We have shown, by two examples of applications, that this method can be applied to a wide class of problems, ranging from physical to abstract modeling. This method is particularly suited to problems where microscopic behavior is well known, but where macroscopic equations are difficult to obtain. Strong results are obtained when the particle system technology is combined in a hybrid approach with for instance a computational fluid dynamics approach: each method can give the best of itself in their own domain of validation, giving birth to a powerful third approach.

It is our opinion that technology based on differential equations (such as, for instance Navier-Stokes equations) are used extensively because these methods are well known; their studies began more than two hundred years ago, when computers did not exist. One can study a differential equation using pen and paper. But it is rather difficult to link such a differential equation to a computer. On the other hand, the human mind is unable to follow three interacting objects, due to their chaotic behavior. But one can easily build programs aiming at doing simulated experiments. As an example, parallel particle systems are natural. Thus, we believe that particle systems can be used extensively, due to the large number of problems they can address, and to their friendship with computers.

Acknowledgment

We thank Christophe Madier de Champvermeil, Didier Delaplace, Hugues Labbé, Philippe Michelon, Eric Tabellion and the ArSciMed's development team for their collaboration on various aspects of this work.

Several films (''Generating a Proton'' and ''Not So Elementary, the Proton''), as well as a virtual reality applica-

tion ("HEUS: Hot Early Universe Soup") have been produced by ArSciMed on the basis of its particle system.

References

1. R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, IOP Publishing Ltd. (1988).
2. E. Cohen, "Computer animations of quantum field theories," *Computer Physics Communications* **70**, 441–446 (1992).
3. E. Cohen, "Computer animations, quantum mechanics and elementary particles," *Europhysics News* **23**, 161–180 (1992).
4. S. B. Pope, "PDF methods for turbulent reactive flows," *Progress in Energy and Combustion Sciences* **11**, 119–192 (1985).
5. U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J. P. Rivet, "Lattice gas hydrodynamics in 2 and 3 dimensions," *Complex System* **1**, 75–136 (1987).
6. W. T. Reeves, "Particle systems—a technique for modelling a class of fuzzy objects," *ACM Transactions on Graphics*, **2**(2) (April 1983); reprinted in *Computer Graphics* **17**(3), 359–376 (1983).
7. W. T. Reeves and R. Blau, "Approximate and probabilistic algorithms for shading and rendering structured particle systems," *Computer Graphics* **19**(3), 313–322 (1985).
8. J. Stam, "Turbulent wind fields for gaseous phenomena," *Computer Graphics* 369–376 (1993).
9. K. Sims, "Particle animation and rendering using data parallel computation," *Computer Graphics* **24**(4), 405–413 (1990).
10. J. M. Pierson, "A dynamic parallel implementation of a physically based particles models," *Eurographics* (1994).
11. W. Thomas, "Sur le chemin de la parallèlisation," Engineer report, Université de Technologie de Compiègne (1996).
12. W. M. Predtechenski and A. I. Milinski, "Personenströme in Gebäuden- Berechnungsmethoden für die Projektierung," *Staatsverlag der Deutscen Demokratischen Republik* (1971).
13. J. Archea, "The evacuation of non-ambulatory patients from hospital and nursing home fires: a framework for a model," *NBSIR* 79–1906 (Nov. 1979).
14. J. T. Henderson, "The statistics of crowd fluids," *Nature* **229**, 381–383 (1971).
15. I. Peschl, "Passage of door openings in panic situations," *BAUN* **26**(2), 62–67 (1971).
16. E. Bouvier and E. Cohen, "Simulation of human flow with particle systems," *Simulators International XII* **27**(3), 349–354 (1995).
17. E. Bouvier and P. Guilloteau, "Crowd simulation in immersive space management," *3rd Eurographics Workshop on Virtual Environments* (1996).
18. E. Tabellion, "Synthèse d'images nucléaires avec simulation d'une caméra par scintillation," DEA report, Université Marne-la-vallée (1996).
19. A. Fournier, D. Fussel, and L. Carpenter, "Computer rendering of stochastic models," *Communication of the ACM* **25**(6), 371–384 (June 1982).

Eric Bouvier is currently completing his PhD thesis at ArSciMed, in collaboration with the I.R.I.T. laboratory of Paul Sabatier University, Toulouse, France. His current research interests include computer animation, crowd simulation, and virtual environments. He received his MS degree in computer science in 1992 from Paul Sabatier University.

Eyal Cohen has distinguished himself as an award-winning theoretical physicist through his work with the Weizmann Institute, Harvard University, CERN, and the Atomic Energy Commission in Saclay, France. He received his PhD in theoretical physics from the Weizmann Institute of Tel Aviv, where he was awarded the John F. Kennedy prize for scientific research. Dr. Cohen's career spans more than 15 years of theoretical and applied physics. Dr. Cohen founded ArSciMed in 1989, a company devoted to the development of software dedicated to simulation and visualization.

Laurent Najman received his doctorate of applied mathematics from Paris Dauphine University and a civil engineering degree from the Ecole Nationale Supérieure des Mines de Paris. Dr. Najman devoted his academic career to solving problems of image processing and mathematical morphology. After earning his diploma in civil engineering, he worked in the research laboratories of Thomson-CSF for three years before joining ArSciMed in 1995. He holds the position of director of research and development. His current research interest is applications of particle system technology to simulation and visualization problems.