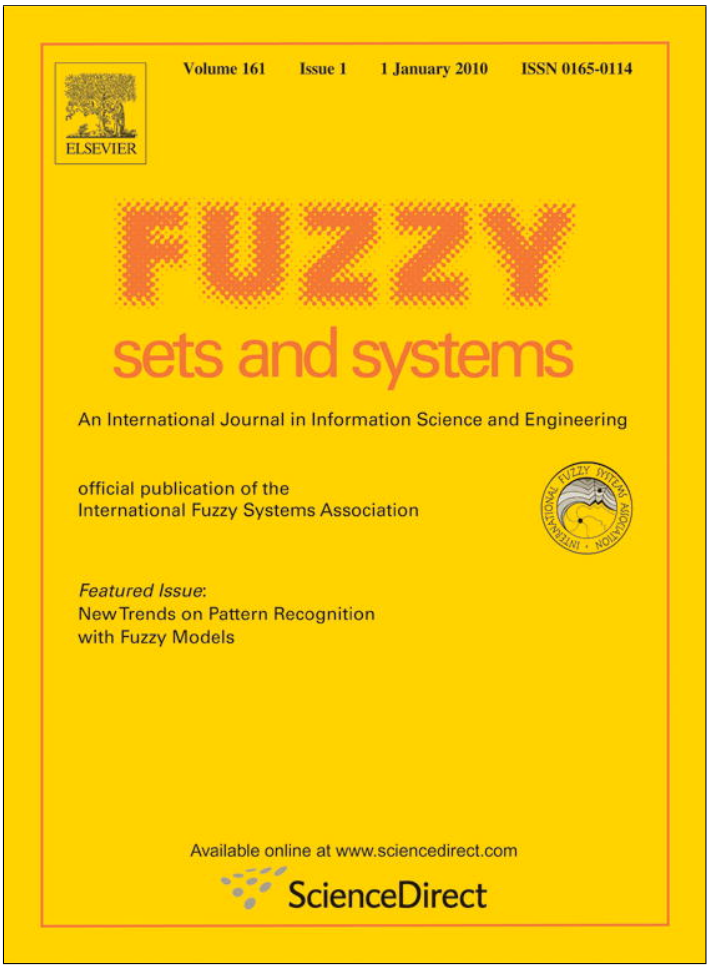


Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Sets and Systems 161 (2010) 118–146
FUZZY sets and systems
www.elsevier.com/locate/fss

Fast fuzzy connected filter implementation using max-tree updates

Giovanni Palma[a, b, *], Isabelle Bloch[b], Serge Muller[a]

[a] *GE Healthcare, 283, rue de la minière, 78530 Buc, France*
[b] *TELECOM ParisTech (ENST), CNRS UMR 5141 LTCI, 46, rue Barrault, 75013 Paris, France*

Received 27 October 2008; received in revised form 27 August 2009; accepted 28 August 2009
Available online 6 September 2009

Abstract

Connected filters are widely used filters in image processing, and their implementation highly benefits from tree representations of images, called max-trees. Extending these filters to fuzzy sets, which may be used to represent imprecision on gray levels in fuzzy gray-level images, requires frequent manipulations of these trees. In this paper we propose efficient algorithms to update tree representations of fuzzy sets according to modifications of the membership values. We show that any modification can be reduced to a series of simple changes, where only one pixel is modified at each step.

Keywords: Fuzzy filtering; Connected filters; Max-tree; Tree merging and updating algorithms; Fuzzy gray-scale images

Contents

1. Introduction

Connected filters [1–5] are widely used in image processing. They allow for example image simplification through levelings [6,7,5] and object or structure detection. They rely on the definition of binary connected components and their behavior depends on how this definition is used to process gray scale images. For instance, connected components

* Corresponding author at: GE Healthcare, 283, rue de la minière, 78530 Buc, France.
E-mail address: giovanni.palma@ge.com (G. Palma).

0165-0114/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2009.08.006

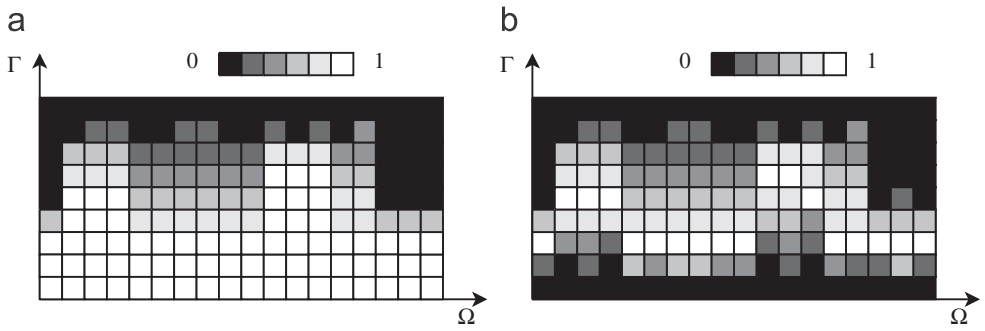


Fig. 1. Example of fuzzy umbra image (a) and fuzzy number image (b) defined on a 1D space Ω.

can be extracted from flat zones of the image [3,8], or from successive thresholdings [1]. Most of the time, this second type of filters is implemented using a max-tree representation of the image [9–11]. Such trees are composed of nodes that represent the connected components of the image obtained for various thresholds while the edges between nodes encode the inclusion order of these connected components. Lots of applications implementing these approaches to image filtering and segmentation have been published, but an exhaustive review is beyond the scope of this paper.

Fuzzy logic has been proposed to cope with various image processing related problems [12] such as segmentation [13–15] and denoising [16,17]. Recently, fuzzy gray scale images [18,19] were introduced in order to model imprecision that may result from the acquisition of images. In fuzzy images, the value of each pixel is a fuzzy quantity while in regular gray scale images it is a crisp gray level. We can define different types of fuzzy images based on the properties of these fuzzy quantities. For instance if the fuzzy quantities decrease with respect to the gray levels (see Fig. 1(a)), we are dealing with fuzzy umbra images [18]. Alternatively, if the fuzzy quantities are fuzzy numbers (see Fig. 1(b)) we are working with fuzzy number images [19]. In both cases, an image is represented as a fuzzy set whose membership function F is defined on $\Omega \times \mathcal{G}$ with Ω the image domain and $\mathcal{G}$ the set of gray levels. Depending on the type of fuzzy images we consider, this membership function has different meanings. For a fuzzy umbra image, with a point $p \in \Omega$ and a gray level g, $F(p, g)$ corresponds to the degree to which the fuzzy image is greater or equal to g at point p. In the case of a fuzzy number image, $F(p, g)$ is the degree to which the image is equal to g at point p. In both cases, extracting fuzzy sets ($F(*, g)$) for each gray level enables to express the fuzzy counterpart of connected filters designed for gray scale images. More specifically, fuzzy connected components [20–22] are extracted from these fuzzy sets and partially or completely removed according to a given criterion [18]. In the case of a fuzzy umbra image, we define an extension of attribute filters or thinnings [23], whereas in the case of fuzzy number images, we extend filters working on flat zones [3]. Note that in the later case, the notion of fuzzy flat zones provided by fuzzy number images is a good alternative to other approaches that relax the notion of flat zones [24–26].

The max-tree representation is not limited to gray scale images. Actually, fuzzy sets can also benefit from it, especially when considering fuzzy connected components. Indeed, they can usually be handled by considering the connected components of the α-cuts of fuzzy sets for which the max-tree representation is appropriate. The problem that arises with fuzzy connected filters in fuzzy gray scale images is that they usually require to have a max-tree representation for each fuzzy set associated to each gray level. Although the construction of such a tree can be done in quasi-linear time [27] or even be parallelized [28], when the number of gray levels is high, building independently one tree for each gray level does not provide a suitable and tractable implementation of fuzzy connected filters. In the case of a linear implementation of the tree construction, the resulting complexity would be $O(GN)$ (with G the number of gray levels and N the number of pixels within the image). Nonetheless, two fuzzy sets extracted from a fuzzy image for two successive gray levels are close to each other in the sense that they only differ on few pixels. This motivates our work about the design of an algorithm aiming at updating a tree that represents a fuzzy set into another tree that represents a second fuzzy set close to the first one. As far as we know, this problem has not been addressed before in the literature. In addition, such an algorithm could potentially be useful in other fields of application where max-trees are used.

In this paper, we propose, as a novel contribution, a set of algorithms to update a tree associated to a fuzzy set when it undergoes changes in the membership values. We focus on definitions, algorithms and related formal results, and do

not detail potential applications to image filtering and segmentation. In Section 2 we introduce notations and definitions used in mathematical developments. Then we propose new operators to grow, respectively deflate, a tree representing a fuzzy set and prove their efficiency in Sections 3 and 4, respectively. In Section 5, resulting algorithms are finally described for the filtering of fuzzy gray scale images.

2. Notations and definitions

The main notations we are using are:

- Ω: image bounded domain endowed with a discrete connectivity.
- $\forall p \in \Omega$, $ngbh(p)$ denotes the neighborhood of p according to the connectivity defined on Ω.
- $\mathcal{G}$: gray scale set.
- $\mathcal{S}$: set of fuzzy sub-sets of Ω ($\Omega \to [0, 1]$).
- 2^{Ω}: set of crisp sub-sets included in Ω ($\Omega \to \{0, 1\}$). Obviously we have: $2^{\Omega} \subset \mathcal{S}$.
- $\mathcal{F}$: set of fuzzy sub-sets defined on $\Omega \times \mathcal{G}$ ($\Omega \times \mathcal{G} \to [0, 1]$).
- $\mathcal{V} = [0, 1] \times 2^{\Omega}$: set of all possible nodes. Every node (α, P) holds a membership degree α and a set P of points included in Ω.
- $\mathcal{E} = \mathcal{V} \times \mathcal{V}$: set of all possible edges between nodes.
- $\mathcal{T}$: set of trees defined as acyclic graphs represented by pairs (V, E) of nodes $V \subseteq \mathcal{V}$ and edges $E \subseteq V^2$. These edges represent the relation between two nodes, and $(n_1, n_2) \in E$ is interpreted as: n_1 is the parent of n_2. Let us remark that without adding any extra-constraint, such trees may not represent fuzzy sub-sets as it will be shown in Definition 2.10.

Definition 2.1. Let $\mathcal{A} : \mathcal{V} \to [0, 1]$ be the operator that returns the membership degree associated with a node:

$$\forall n = (\alpha, P) \in \mathcal{V}, \quad \mathcal{A}(n) = \alpha \tag{1}$$

$\mathcal{A}$ is the first partial mapping from $\mathcal{V}$ into $[0, 1]$. As it will be shown later, if n is a node of a tree representing a fuzzy sub-set, $\mathcal{A}(n)$ will correspond to the membership degree associated with the α-cut represented by n.

Definition 2.2. Let $\mathcal{P} : \mathcal{V} \to 2^{\Omega}$ be the operator that associates with a node the set of points this node contains

$$\forall n = (\alpha, P) \in \mathcal{V}, \quad \mathcal{P}(n) = P \tag{2}$$

$\mathcal{P}$ is the second partial mapping from $\mathcal{V}$ into 2^{Ω}.

Definition 2.3. Let $\mathcal{N} : \mathcal{T} \to 2^{\mathcal{V}}$ be the operator that associates with a tree the set of its nodes:

$$\forall t = (V, E) \in \mathcal{T}, \quad \mathcal{N}(t) = V \tag{3}$$

$\mathcal{N}$ is the first partial mapping from $\mathcal{T}$ into $2^{\mathcal{V}}$.

Definition 2.4. Let $\mathcal{B} : \mathcal{T} \to 2^{\mathcal{E}}$ be the operator that associates with a tree the set of edges between its nodes:

$$\forall t = (V, E) \in \mathcal{T}, \quad \mathcal{B}(t) = E \tag{4}$$

$\mathcal{B}$ is the second partial mapping from $\mathcal{T}$ into $2^{\mathcal{E}}$.

Definition 2.5. Let $\mathcal{W} : \mathcal{T} \times \mathcal{V} \to 2^{\mathcal{V}}$ be the operator that returns the set of children of a node in a tree:

$$\forall t \in \mathcal{T}, \forall n \in \mathcal{N}(t), \quad \mathcal{W}(t, n) = \bigcup_{v \in \mathcal{N}(t)/(n,v) \in \beta(t)} \{v\} \tag{5}$$

Definition 2.6. Let $\mathcal{R} : \mathcal{T} \to \mathcal{V}$ be the operator that returns the root of a tree:

$$\forall t \in \mathcal{T}, \quad \mathcal{R}(t) = n \in \mathcal{N}(t)/\forall v \in \mathcal{N}(t), \ (v, n) \notin \mathcal{B}(t) \tag{6}$$

Definition 2.7. Let $\mathcal{D} : \mathcal{T} \times \mathcal{V} \to 2^{\mathcal{V}}$ be the operator that returns the descendants (children, children's children, etc.) of a node in a tree, which is recursively defined as

$$\forall t \in \mathcal{T}, \forall n \in \mathcal{N}(t), \quad \mathcal{D}(t, n) = \bigcup_{v \in \mathcal{W}(t,n)} (\{v\} \cup \mathcal{D}(t, v)) \tag{7}$$

Let us note that $n \notin \mathcal{D}(t, n)$ and that if n is a leaf, $\mathcal{W}(t, n) = \emptyset$ and consequently $\mathcal{D}(t, n) = \emptyset$.

Definition 2.8. $\mathcal{D}' : \mathcal{T} \times \mathcal{V} \to 2^{\mathcal{V}}$ is the operator defined as $\mathcal{D}$ but whose result also contains the node n:

$$\forall t \in \mathcal{T}, \forall n \in \mathcal{N}(t), \quad \mathcal{D}'(t, n) = \mathcal{D}(t, n) \cup \{n\} \tag{8}$$

Definition 2.9. The operator $st : \mathcal{T} \times \mathcal{V} \to 2^{\mathcal{T}}$, which associates with a node of a tree, the set of sub-trees whose root is a child of this node, is defined as follows:

$$\forall t = (V, E) \in \mathcal{T}, \forall n \in V$$

$$st(t, n) = \bigcup_{v \in \mathcal{W}(t,n)} \left\{ \left(\mathcal{D}'(t, v), \bigcup_{(v_1,v_2) \in \mathcal{D}'(t,v)^2 / (v_1,v_2) \in E} \{(v_1, v_2)\} \right) \right\} \tag{9}$$

Let us remark that n does not belong to any of the resulting sub-trees since their roots are children of n.

Definition 2.10. Let $t \in \mathcal{T}$, t is a nested tree if:

$$\forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n), \quad \mathcal{A}(s) > \mathcal{A}(n) \tag{10}$$

$$\wedge \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n), \quad \mathcal{P}(s) \subseteq \mathcal{P}(n) \tag{11}$$

$$\wedge \forall n \in \mathcal{N}(t), \quad \bigcap_{s \in \mathcal{W}(t,n)} \mathcal{P}(s) = \emptyset \tag{12}$$

A partial order is introduced on the nodes with respect to their membership degrees (Eq. (10)). The sub-sets of points are also nested according to the same order (Eq. (11)). Finally overlapping is forbidden between the descendants of a given node (Eq. (12)).

Definition 2.11. $\forall t \in \mathcal{T}, \forall f \in \mathcal{S}$, t is a representation of f if:

$$t \text{ is nested} \tag{13}$$

$$\wedge \forall p \in \Omega, \forall \alpha \in [0, 1], \quad p \in f_\alpha \Rightarrow \exists n \in \mathcal{N}(t) / \mathcal{P}(n) = \Gamma^p_{f_\alpha} \tag{14}$$

$$\wedge \forall n \in \mathcal{N}(t), \forall p \in \Omega, \quad p \in \mathcal{P}(n) \Rightarrow \Gamma^p_{f_{\mathcal{A}(n)}} = \mathcal{P}(n) \tag{15}$$

with Γ^p_N the connected component of N that contains p, and f_α the α-cut of the fuzzy set f.

With this definition, we introduce a new constraint regarding the mapping between the nodes of the nested tree and the connected components contained in the α-cuts. Each connected component of each α-cut of f has to be represented by a node (Eq. (14)) and each node of t has to represent an existing connected component contained in an α-cut of f (Eq. (15)). Fig. 2 shows that, with this definition, a tree is a more or less compact representation of a fuzzy set. A connected component of an α-cut (here α_0) can be indirectly represented by a node n that may not have the same membership degree (here $\mathcal{A}(n) = \alpha_1$). Actually, some quantization steps may not be explicitly represented by a level in the tree.

The set of trees $t \in \mathcal{T}$ that are nested is denoted by $\mathcal{T}^e$ ($\mathcal{T}^e \subset \mathcal{T}$).

Definition 2.12. The set $Q_p(T)$ of trees included in a set of nested trees T and whose root contains a given point p is defined as follows:

$$\forall p \in \Omega, \forall T \in 2^{\mathcal{T}^e}, \quad Q_p(T) = \{t \in T / p \in \mathcal{P}(\mathcal{R}(t))\} \tag{16}$$

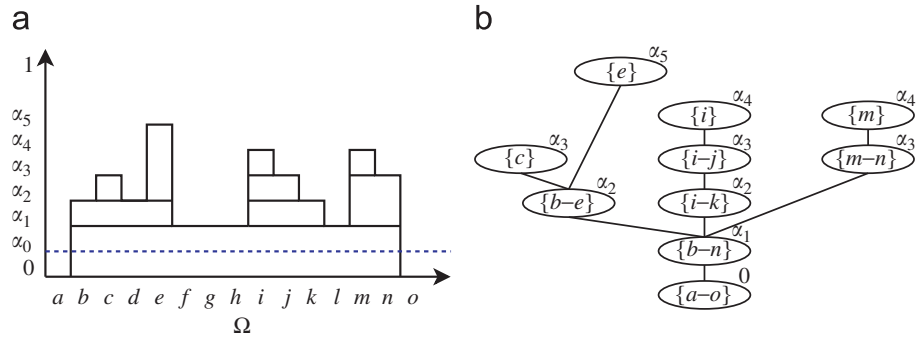


Fig. 2. Fuzzy set representation using a tree. The only connected component contained in the α-cut α_0 is indirectly represented by the node that contains points b to n and that has a membership degree equal to α_1.

Theorem 2.1. *Let t be a nested tree. If we consider a set of sibling nodes within this tree (set of the children $st(t, n)$ of a given node n), and a point $p \in \Omega$, then p is included in the root of at most one of the sub-trees associated to these brothers. This means that information about a fuzzy set represented by t at point p is only contained in a single branch of t.*

$$\forall t \in \mathcal{T}^e, \forall n \in \mathcal{N}(t), \forall p \in \Omega, \quad (|Q_p(st(t, n))| = 0) \vee (|Q_p(st(t, n))| = 1) \tag{17}$$

with $|\cdot|$ the cardinal of a set.

Proof. Using Definition 2.10, we have: $\bigcap_{v \in \mathcal{W}(n)} \mathcal{P}(v) = \emptyset$. Thus for a given p, there is at most only one $v \in \mathcal{W}(t, n)$ such that $p \in \mathcal{P}(v)$. □

3. Growing of a tree representing a fuzzy set

We are now going to introduce a new operator that aims at merging two trees t_1 and t_2 around two points p_1 and p_2, which are contained in the root of t_1 and the root of t_2, respectively (see Fig. 3). Concretely, one of the points will correspond to the place where the fuzzy set is increasing (i.e. the membership of this point is modified to a higher value), and the other point will correspond successively to each point that lies in its neighborhood. This will allow reconnecting the sub-trees, and thus the associated connected components that are marked by these points. This operator will have a similar behavior to the one recently introduced in [28] to reconnect two max-trees representing a gray-level image on neighboring sub-domains. Here we use a higher level formalism in order to use our operator in a process aiming at updating/growing a tree. This also enables us to have a more generic expression with respect to the data structure used during the implementation of our algorithm as it will be shown in Section 5.

Depending on its input (i.e. sub-trees and markers), the operator we are now introducing may produce a tree representing a fuzzy set. A process to restrict input data to ensure that this property is verified will be described later in this section (see Definition 3.4).

Definition 3.1. Let $\overline{M} : \mathcal{T}^e \times \mathcal{T}^e \times \Omega \times \Omega \to \mathcal{T}$ be a merging operator defined as follows:

$$\forall t_1 = (V_1, E_1) \in \mathcal{T}^e, \quad \forall t_2 = (V_2, E_2) \in \mathcal{T}^e, \quad \forall (p_1, p_2) \in \Omega^2$$

Case 1: $(\mathcal{A}(\mathcal{R}(t_1)) = \mathcal{A}(\mathcal{R}(t_2))) \wedge ((|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 0) \vee (|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))| = 0))$
In this case, $\overline{M}(t_1, t_2, p_1, p_2) = (V_3, E_3)$ with

$$V_3 = \{(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2)))\} \cup \bigcup_{t \in st(t_1, \mathcal{R}(t_1)) \cup st(t_2, \mathcal{R}(t_2))} (\mathcal{N}(t))$$

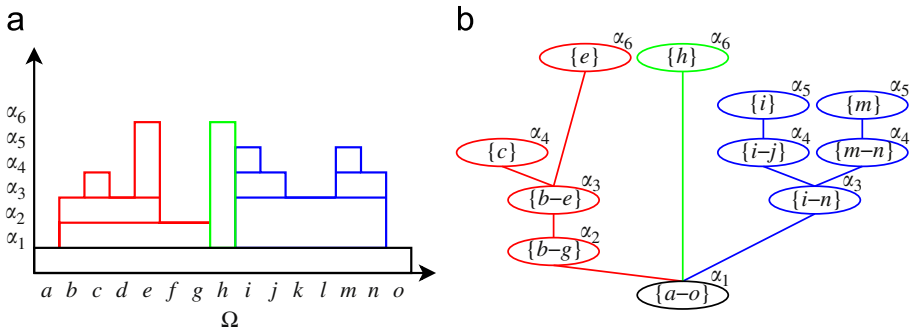


Fig. 3. Overview of the growing process. (a) Fuzzy set to be represented after the merging processes. The value at point h is increased from α_1 to α_6. (b) Tree, which is not a representation of a fuzzy set, that contains sub-trees (in red, green and blue) to be merged. To restore the connected components of the different α-cuts, the tree in red and the tree in green have to be merged using marker points g and h, respectively, and the resulting tree has to be merged with the blue tree using marker points h and i, respectively. Marker points g, h and i are used to select which nodes have to be merged, they represent locations where the connectivity may have changed (neighborhood of the point where the value has increased). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

which corresponds to the new root node and to all the nodes of t_1 and t_2 except their respective roots.

$$\begin{aligned} E_3 = \quad & \mathcal{B}(t_1)\setminus \bigcup_{v\in\mathcal{W}(t_1,\mathcal{R}(t_1))} \{(\mathcal{R}(t_1), v)\} \\ \cup\ & \mathcal{B}(t_2)\setminus \bigcup_{v\in\mathcal{W}(t_2,\mathcal{R}(t_2))} \{(\mathcal{R}(t_2), v)\} \\ \cup\ & \bigcup_{v\in\mathcal{W}(t_1,\mathcal{R}(t_1))\cup\mathcal{W}(t_2,\mathcal{R}(t_2))} \{((\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1)\cup\mathcal{P}(t_2)), v)\} \end{aligned}$$

where the two first terms correspond to the edges of t_1 and t_2 that do not reach $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$, respectively. The third term enables to introduce edges involving the new root.

Case 2: $(\mathcal{A}(\mathcal{R}(t_1)) = \mathcal{A}(\mathcal{R}(t_2))) \wedge (|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 1) \wedge (|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))| = 1)$.

In this case, $\overline{M}(t_1, t_2, p_1, p_2) = (V_3, E_3)$ with

$$\begin{aligned} V_3 = \quad & \{(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(\mathcal{R}(t_1))\cup\mathcal{P}(\mathcal{R}(t_2)))\} \\ \cup\ & \bigcup_{t\in st(t_1,\mathcal{R}(t_1))/p_1\notin\mathcal{P}(\mathcal{R}(t))} \mathcal{N}(t) \\ \cup\ & \bigcup_{t\in st(t_2,\mathcal{R}(t_2))/p_2\notin\mathcal{P}(\mathcal{R}(t))} \mathcal{N}(t) \\ \cup\ & \mathcal{N}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2)) \end{aligned}$$

where the first part corresponds to the creation of the root node. The second and third parts correspond to the insertion of sub-trees whose root does not contain p_1 or p_2. The fourth part corresponds to the nodes resulting from the merging of the two sub-trees that contain one of the former points.

$$\begin{aligned} E_3 = \quad & \bigcup_{t\in st(t_1,\mathcal{R}(t_1))/p_1\notin\mathcal{P}(\mathcal{R}(t))} \mathcal{B}(t) \\ \cup\ & \bigcup_{t\in st(t_2,\mathcal{R}(t_2))/p_2\notin\mathcal{P}(\mathcal{R}(t))} \mathcal{B}(t) \\ \cup\ & \bigcup_{v\in\mathcal{W}(t_1,\mathcal{R}(t_1))\cup\mathcal{W}(t_2,\mathcal{R}(t_2))/p_1\notin\mathcal{P}(v)\wedge p_2\notin\mathcal{P}(v)} \{((\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1)\cup\mathcal{P}(t_2)), v)\} \\ \cup\ & \{(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1)\cup\mathcal{P}(t_2)), \\ & \mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2))\} \\ \cup\ & \mathcal{B}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2)) \end{aligned}$$

where the two first parts correspond to edges of sub-trees whose root does not contain p_1 or p_2. The third part corresponds to edges between the root of these sub-trees and the new root. The fourth part is the link between this last root and the tree resulting from the merging of the two sub-trees that contain p_1 or p_2. The last part corresponds to edges resulting from this merging.

Case 3: $(\mathcal{A}(\mathcal{R}(t_1)) < \mathcal{A}(\mathcal{R}(t_2))) \wedge (|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| \neq 0))$.

In this case, $\overline{M}(t_1, t_2, p_1, p_2) = (V_3, E_3)$ with

$$\begin{aligned} V_3 = \quad & \{(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2)))\} \\ \cup \quad & \bigcup_{t \in st(t_1, \mathcal{R}(t_1))/\, p_1 \notin \mathcal{P}(\mathcal{R}(t))} \mathcal{N}(t) \\ \cup \quad & \mathcal{N}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2)) \end{aligned}$$

where the first part corresponds to the new root node, the second part to the nodes of sub-trees of t_1 whose root does not contain p_1, the third part to the merging of the remaining sub-tree with t_2.

$$\begin{aligned} E_3 = \quad & \bigcup_{t \in st(t_1, \mathcal{R}(t_1))/\, p_1 \notin \mathcal{P}(\mathcal{R}(t))} \mathcal{B}(t) \\ \cup \quad & \bigcup_{v \in \mathcal{W}(t_1, \mathcal{R}(t_1))/\, p_1 \notin \mathcal{P}(v)} \{((\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1) \cup \mathcal{P}(t_2)), v)\} \\ \cup \quad & \{((\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1) \cup \mathcal{P}(t_2)), \mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2)))\} \\ \cup \quad & \mathcal{B}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2)) \end{aligned}$$

where the first part corresponds to the edges taken from the sub-trees of t_1 whose root does not contain p_1, the second part to the edges between the root of these sub-trees and the new root, the third part to the edge between the new root and the root of the merged tree. The last part corresponds to the edges of this merged tree.

Case 4: $(\mathcal{A}(\mathcal{R}(t_1)) < \mathcal{A}(\mathcal{R}(t_2))) \wedge (|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 0))$.

In that case, $\overline{M}(t_1, t_2, p_1, p_2) = (V_3, E_3)$ with

$$\begin{aligned} V_3 = \quad & \{(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2)))\} \\ \cup \quad & \mathcal{N}(st(t_1, \mathcal{R}(t_1))) \\ \cup \quad & \mathcal{N}(t_2) \end{aligned}$$

with, in order of appearance, the new root node, the nodes of t_1 without its root, and all nodes from t_2.

$$\begin{aligned} E_3 = \quad & \bigcup_{t \in st(t_1, \mathcal{R}(t_1)) \cup \{t_2\}} \mathcal{B}(t) \\ \cup \quad & \bigcup_{v \in \mathcal{W}(t_1, \mathcal{R}(t_1))} \{((\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1) \cup \mathcal{P}(t_2)), v)\} \\ \cup \quad & \{((\mathcal{A}(\mathcal{R}(t_1)), \mathcal{P}(t_1) \cup \mathcal{P}(t_2)), \mathcal{R}(t_2))\} \end{aligned}$$

where the first part corresponds to the edges of t_2 and the edges of the sub-trees of t_1. The second part corresponds to edges between the new node and the roots of the sub-trees of t_1. The last part is the edge between the new root and the root of t_2.

Case 5: $\mathcal{A}(\mathcal{R}(t_1)) > \mathcal{A}(\mathcal{R}(t_2))$.

In this last configuration, we have $\overline{M}(t_1, t_2, p_1, p_2) = \overline{M}(t_2, t_1, p_2, p_1)$, and we come back to cases 3 and 4.

Remark 3.1. Let us note that no other configurations are possible for operator $\overline{M}$ since the definition covers all possible combinations of values for $|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))|$ and $|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))|$ for all possible equalities/inequalities between $\mathcal{A}(\mathcal{R}(t_1))$ and $\mathcal{A}(\mathcal{R}(t_2))$.

Let us illustrate Definition 3.1 on simple examples corresponding to the first four cases of input data.

Fig. 4 illustrates the first case for $p_1 = h$ and $p_2 = i$ with t_1 the tree shown in Fig. 4(b) and t_2 the one in Fig. 4(c). In this case, we have $|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 0$ because only the root of t_1 contains the point h (p_1), and $|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))| = 1$

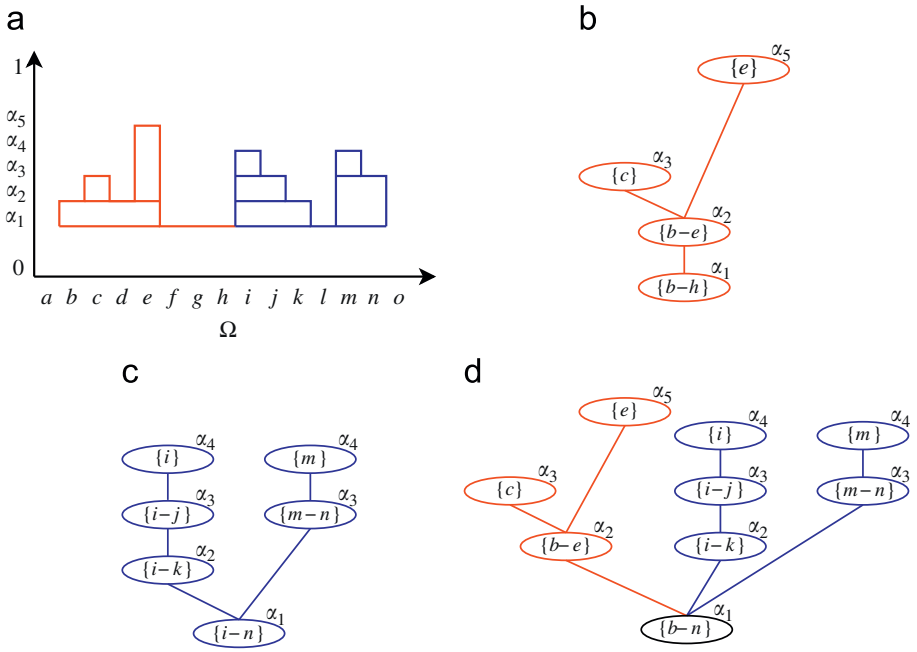


Fig. 4. Operator $\overline{M}$ in case 1. (a) Fuzzy sets represented by t_1 (in red) and by t_2 (in blue), (b) t_1, (c) t_2, (d) $\overline{M}(t_1, t_2, h, i)$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

because there is one sub-tree of t_2 that contains i (p_2) and whose root is a child of the root of t_2. In order to merge these two trees, which both have a root with the same membership degree, we only need to merge the roots of t_1 and t_2 (black node in Fig. 4(d)) and to reconnect all the sub-trees coming from the children of the former roots (nodes in red and blue in Fig. 4(d)).

The second case is illustrated in Fig. 5, again with $p_1 = h$ and $p_2 = i$. The trees t_1 and t_2 are presented in Fig. 5(b) and (c), respectively. For both trees, we have p_1 and p_2 included in the root nodes $\mathcal{R}(t_1)$ and $\mathcal{R}(t_2)$, respectively, but also in one child of both roots. For this reason, $|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 1$ and $|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))| = 1$. Since the root nodes have the same membership degree, they can be merged in the same way as in the former case (black node in Fig. 5(d)). Nonetheless, the remaining sub-trees have to be processed differently: the sub-trees whose root does not contain p_1 or p_2 can be directly linked to the root node (e.g. the blue tree outside the gray area in Fig. 5(d)). Finally, both sub-trees whose root does contain p_1 and p_2, respectively, need to be merged (gray area in Fig. 5(d)).

The third configuration is presented in Fig. 6 again with $p_1 = h$ and $p_2 = i$. The trees t_1 and t_2 are presented in Fig. 6(b) and (c), respectively. This case is similar to the former one, the main difference is the non-equality between the membership degrees of the root nodes of t_1 and t_2. Thus, the new root node (in black in Fig. 6(d)) is the union of points contained in the root of t_1 and t_2 and has the same membership degree as the root of t_1, which is smaller than the one of the root of t_2. The sub-trees of t_1 whose root does not contain p_1 are directly linked to the new root while the sub-tree that contains it is merged with t_2 (in gray in Fig. 6(d)).

The fourth case is illustrated in Fig. 7. It is very similar to the former one except that there is no sub-tree of t_1 whose root node contains p_1. Thus, this is a terminal case where no more merging is necessary.

Theorem 3.1. *Under specific conditions, $\overline{M}$ gives a result in $\mathcal{T}^e$:*

$$
\begin{aligned}
&\forall (t_1, t_2) \in \mathcal{T}^{e^2}, \forall (p_1, p_2) \in \Omega \\
&\forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2) \quad \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset \\
&\Rightarrow \begin{cases} \overline{M}(t_1, t_2, p_1, p_2) \in \mathcal{T}^e \\ \quad \wedge (\mathcal{A}(\mathcal{R}(\overline{M}(t_1, t_2, p_1, p_2))) = \min(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{A}(\mathcal{R}(t_2)))) \\ \quad \wedge (\mathcal{P}(\mathcal{R}(\overline{M}(t_1, t_2, p_1, p_2))) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2))) \end{cases}
\end{aligned}
$$

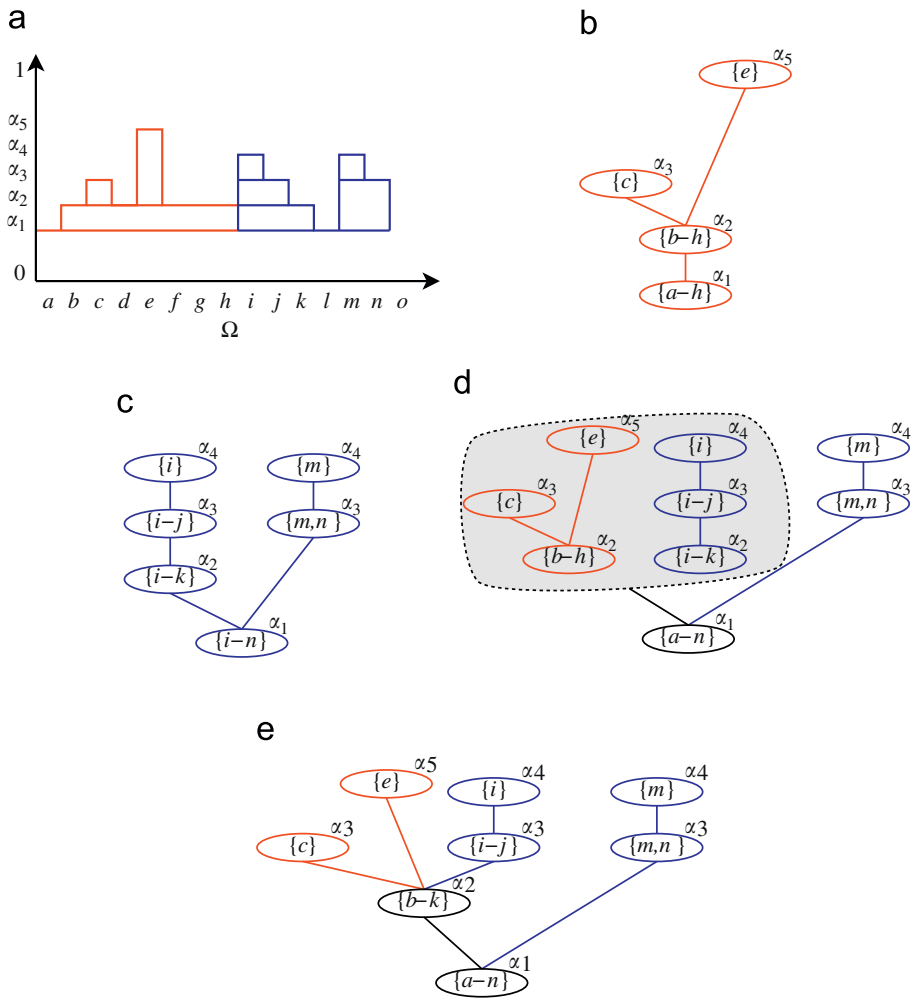


Fig. 5. Operator $\overline{M}$ in case 2. (a) Fuzzy sets represented by t_1 (in red) and t_2 (in blue), (b) t_1, (c) t_2, (d) $\overline{M}(t_1, t_2, h, i)$ where the trees in the gray area have to be merged with a new call to the merging operator $\overline{M}$, (e) final tree after all recursive calls to $\overline{M}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This strong hypothesis corresponds to the idea that t_1 and t_2 partially represent a fuzzy set on distinct sub-domains of Ω. In other words, this means that we are dealing with cases similar to the ones illustrated in Figs. 4–7.

Proof. See Appendix A.

Definition 3.2. Two nodes $(n_1, n_2) \in \mathcal{V}^2$ are said compatible in $(t_1, t_2) \in \mathcal{T}^{e^2}$ if

$$(n_1, n_2) \in \mathcal{N}(t_1) \times \mathcal{N}(t_2)$$

$$\wedge\ \mathcal{P}(n_1) \cup \mathcal{P}(n_2)\ \ \textit{connected}$$

$$\wedge\ \mathcal{A}(n_1) \leq \mathcal{A}(n_2) \Rightarrow \nexists n_2' \in \mathcal{N}(t_2) \,/\, \big((\mathcal{A}(n_1) \leq \mathcal{A}(n_2') < \mathcal{A}(n_2)) \wedge\ (\mathcal{P}(n_1) \cup \mathcal{P}(n_2')\ \ \textit{connected})\big)$$

$$\wedge\ \mathcal{A}(n_2) \leq \mathcal{A}(n_1) \Rightarrow \nexists n_1' \in \mathcal{N}(t_1) \,/\, \big((\mathcal{A}(n_2) \leq \mathcal{A}(n_1') < \mathcal{A}(n_1)) \wedge\ (\mathcal{P}(n_2) \cup \mathcal{P}(n_1')\ \ \textit{connected})\big)$$

This definition enables to tell whether two nodes can be merged in order to represent partially or completely a connected component of a given α-cut as illustrated in Fig. 8.

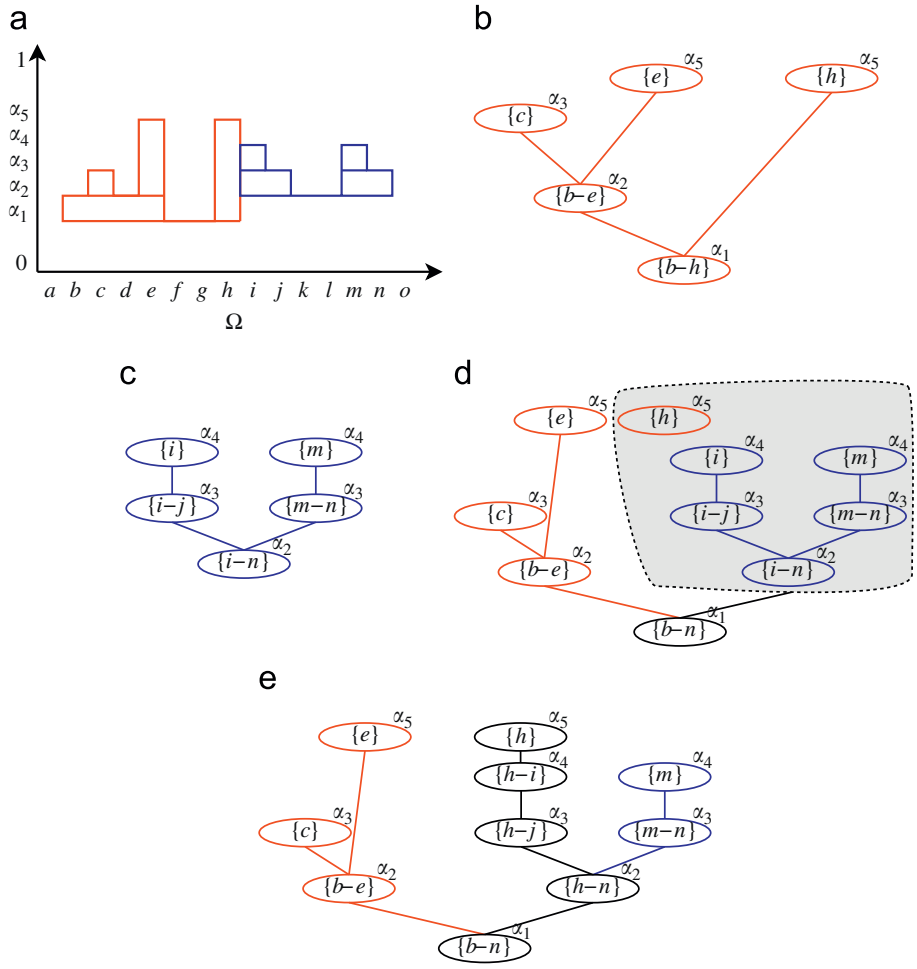


Fig. 6. Operator $\overline{M}$ in case 3. (a) Fuzzy sets represented by t_1 (in red) and t_2 (in blue), (b) t_1, (c) t_2, (d) $\overline{M}(t_1, t_2, h, i)$ where the trees in the gray area have to be merged by a new call to the operator $\overline{M}$, (e) final tree after all recursive calls to $\overline{M}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Theorem 3.2. *Let* $(t_1, t_2) \in \mathcal{T}^{e^2}$ *verifying hypotheses of Theorem* 3.1. *Then* $\forall (p_1, p_2) \in \Omega^2$

$$\forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2)(\mathcal{A}(n_1) \leq \mathcal{A}(n_2)) \wedge (p_1 \in \mathcal{P}(n_1)) \wedge (p_2 \in \mathcal{P}(n_2))$$

$$\wedge\ (n_1 \text{ and } n_2 \text{ compatible in } t_1 \text{ and } t_2)$$

$$\Rightarrow \exists n_3 \in \mathcal{N}(\overline{M}(t_1, t_2, p_1, p_2)) \,/\, ((\mathcal{A}(n_3) = \mathcal{A}(n_1)) \wedge\ (\mathcal{P}(n_3) = \mathcal{P}(n_1) \cup \mathcal{P}(n_2)))$$

Remark 3.2. This theorem can be interpreted as follows: two trees that are nested trees on distinct sub-domains produce a tree that contains the nodes that represent connected components corresponding to the union of points of compatible nodes that contain p_1 and p_2, respectively. In other words, the operator $\overline{M}$ locally reconnects the connected components for various α-cuts. Finally, let us remark that only compatible nodes induce new nodes after merging the trees, thus original nodes are either kept or increased.

Proof. Using the definition of $\overline{M}$, and Eqs. (11) and (12), we directly get Theorem 3.2. □

Definition 3.3. Let $subst : \mathcal{T} \times \mathcal{T} \times \mathcal{T} \to \mathcal{T}$ be the operator that associates to $(t_1, t_2, t_3) \in \mathcal{T}^3$ the tree $subst(t_1, t_2, t_3)$, which corresponds to t_1 where t_2 was replaced by t_3. If t_2 is not a maximal sub-tree of t_1 then $subst(t_1, t_2, t_3) = t_1$.

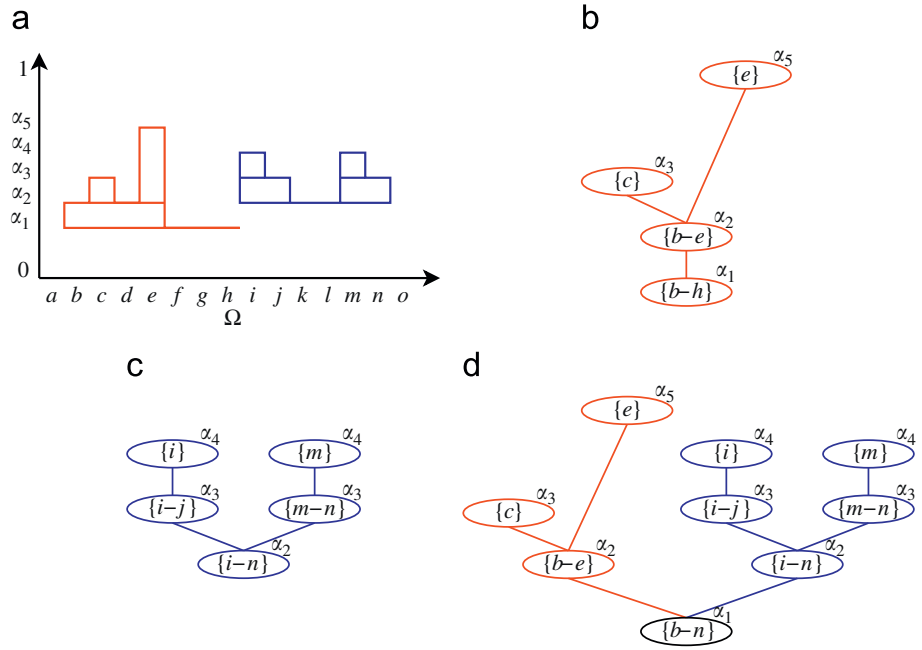


Fig. 7. Operator $\overline{M}$ in case 4. (a) fuzzy sets represented by t_1 (in red) and by t_2 (in blue), (b) t_1, (c) t_2, (d) $\overline{M}(t_1, t_2, h, i)$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

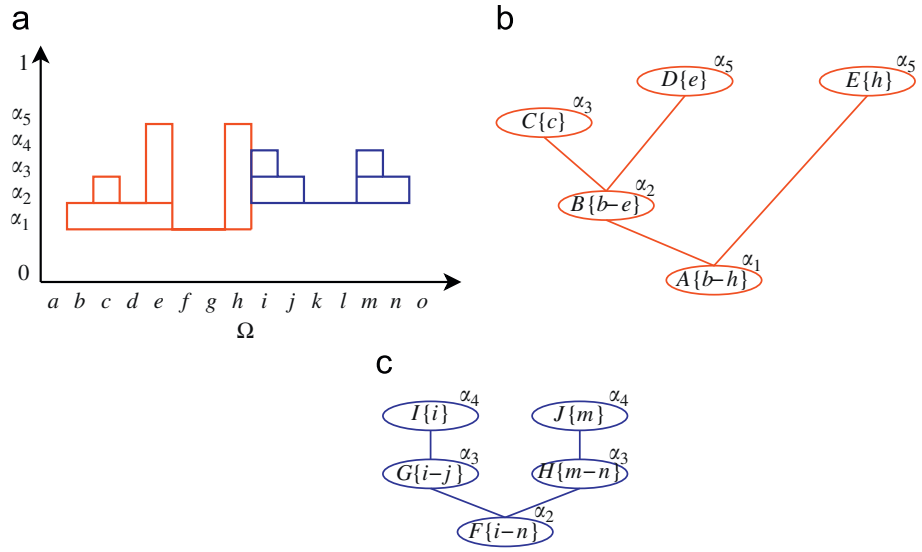


Fig. 8. Example of compatible nodes in trees t_1 (b) and t_2 (c) that represent a fuzzy set (a) on distinct parts of the domain Ω. Only the pairs of nodes (A, F), (E, I), (E, F) and (E, G) are compatible. Non-compatibility is illustrated by nodes A and G since $\mathcal{A}(A) \leq \mathcal{A}(F) \leq \mathcal{A}(G)$.

Fig. 9 illustrates this definition in the case t_2 (cf. Fig. 9(b)) is a maximal sub-tree of t_1 (cf. Fig. 9(a)).

Let us introduce a method to update a tree t, which represents a fuzzy set f, in order to obtain a tree t^c, which represents f' (modified version of f at point $\tilde{p}$). Here the goal is to add a new child, which contains $\tilde{p}$ and has a membership degree of $f'(\tilde{p})$, to the node that contains the same point and that has a membership degree of $f(\tilde{p})$, in order to call the merging operator on this node and nodes that contain a neighbor of $\tilde{p}$. This process is illustrated in Fig. 10.

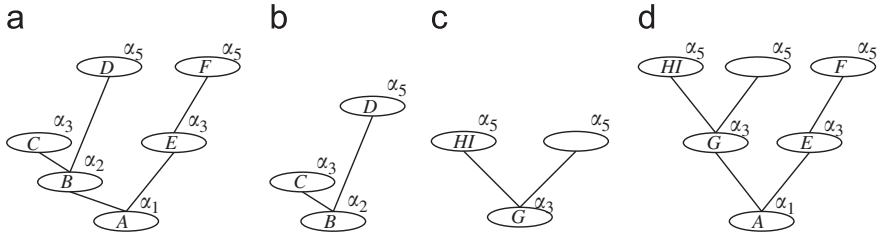


Fig. 9. Example of result of operator *subst*. (a), (b), (c) and (d) represent t_1, t_2, t_3 and $subst(t_1, t_2, t_3)$, respectively.

Definition 3.4. Let $M : \Omega \times 2^{\Omega} \times \mathcal{T}^e \to \mathcal{T}^e$ be the merging operator of several nodes of a tree. M is recursively defined as:

Case 1: $\forall \tilde{p} \in \Omega, \forall \tilde{t} \in \mathcal{T}^e, M(\tilde{p}, \{\}, \tilde{t}) = \tilde{t}$.

Case 2: $\forall \tilde{p} \in \Omega, \forall P = \{p_1..p_k\} \in 2^{\Omega}, \forall t \in \mathcal{T}^e$.

$M(\tilde{p}, P, \tilde{t}) = subst(M(\tilde{p}, P\backslash\{p_k\}, t), t', t'')$ where t' is a sub-tree of $t^0 = M(\tilde{p}, P\backslash\{p_k\}, \tilde{t})$ such that:

$$\mathcal{R}(t') = \underset{r \in \mathcal{N}(t^0)/(\tilde{p} \in \mathcal{P}(r)) \wedge (p_k \in \mathcal{P}(r))}{\arg\max} \mathcal{A}(r) \tag{18}$$

$$\wedge\ \mathcal{N}(t') = \mathcal{D}'(t^0, \mathcal{R}(t')) \tag{19}$$

and where t'' is defined, using the notations $T = st(t', \mathcal{R}(t'))$, $T_Q = Q_{\tilde{p}}(T) \cup Q_{p_k}(T)$ and $T_{\overline{Q}} = T\backslash T_Q$, as follows:

- $t'' = t'$ if $|Q_{\tilde{p}}(T)| = 0 \vee |Q_{p_k}(T)| = 0$,
- otherwise: $\mathcal{N}(t'') = \bigcup_{t''' \in T_{\overline{Q}}} \mathcal{N}(t''') \cup \mathcal{N}(\overline{M}(Q_{\tilde{p}}(T), Q_{p_k}(T), \tilde{p}, p_k)) \cup \{\mathcal{R}(t')\}$ where the first term corresponds to nodes of unchanged trees, the second to nodes of trees that were merged, and the third to the root that remains unchanged ($\mathcal{R}(t'') = \mathcal{R}(t')$); and

$$\mathcal{B}(t'') = \bigcup_{t''' \in T_{\overline{Q}}} (\mathcal{R}(t'), \mathcal{R}(t''')) \cup \mathcal{B}(t''') \cup \mathcal{B}(\overline{M}(Q_{\tilde{p}}(T), Q_{p_k}(T), \tilde{p}, p_k)) \cup (\mathcal{R}(t'), \mathcal{R}(\overline{M}(Q_{\tilde{p}}(T), Q_{p_k}(T), \tilde{p}, p_k)))$$

where the first term corresponds to edges of the unchanged sub-trees (with the edge between their root and the root of t'), and the other ones to edges associated with nodes of the merged sub-tree.

The tree t' used in case 2 corresponds to the sub-tree of t^0, resulting from the first recursive calls (Eq. (19)), whose root contains the two points to be used as markers in the merging process and whose remaining nodes do not contain these two points at the same time. This last property means that the root of t' is the node that contains p_k and $\tilde{p}$ and that maximizes its membership degree (see Eq. (18)) since the descendants of a node n in a nested tree have a membership degree greater than $\mathcal{A}(n)$.

Fig. 10 illustrates the different recursive calls to M on a concrete example. Here the tree $\tilde{t}$ is presented in Fig. 10(b), with $P = \{p_1, p_2.p_3, p_4\}$ the neighbors of $\tilde{p}$. The merging at points $\tilde{p}$ and p_1 (Fig. 10(c)) of $\tilde{t}$ is merged at points $\tilde{p}$ and p_2 (Fig. 10(e)) in order to be merged again at points $\tilde{p}$ and p_3 (Fig. 10(g)) and eventually at points $\tilde{p}$ and p_4. The result of these mergings corresponds to $M(\tilde{p}, P, \tilde{t})$ as shown in Fig. 10(h). In order to identify $t^0, t', t'', \ldots$ at a given recursion step, we can refer, for instance, to Figs. 10(d) and (e) where t^0 is the tree of Fig. 10(d) and t' the tree in red dots. The sub-trees $Q_{p_k}(T) = Q_{p_2}(T)$ and $Q_{\tilde{p}}(T)$ also appear in this last figure. Finally, Fig. 10(e) corresponds to $subst(t^0, t', t'')$ with t'' the tree in blue dots.

Theorem 3.3. $\forall t_1, t_2, t_3 \in \mathcal{T}^e, \mathcal{R}(t_2) = \mathcal{R}(t_3) \Rightarrow subst(t_1, t_2, t_3) \in \mathcal{T}^e$.

Proof. Let us prove Eqs. (10)–(12).

Let $t_1, t_2, t_3 \in \mathcal{T}^e$ such that $\mathcal{R}(t_1) = \mathcal{R}(t_2)$

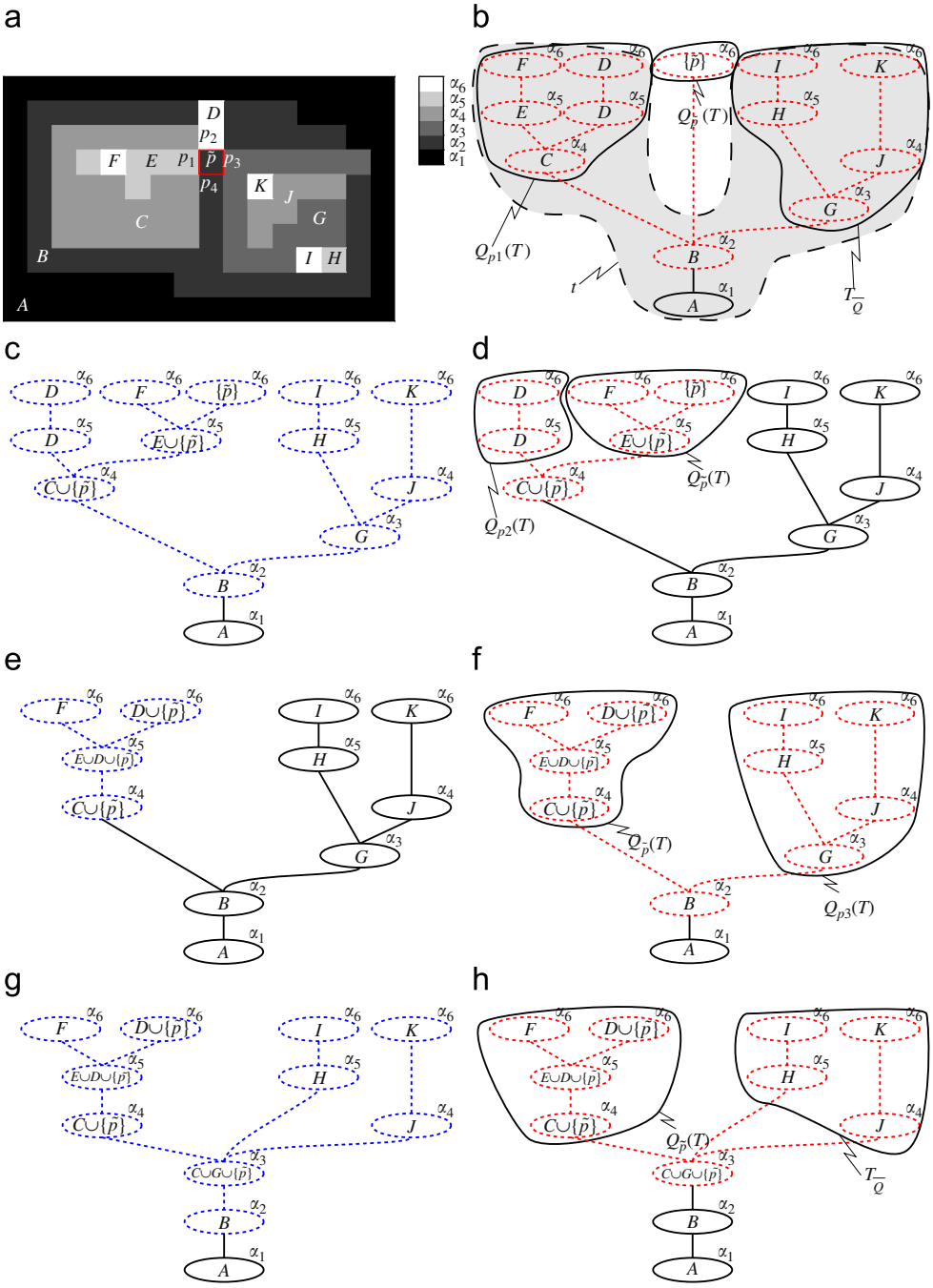


Fig. 10. Application of operator M on the fuzzy set f (a), where the value at $\tilde{p}$ becomes α_6. (b) Tree $\tilde{t}$, which is the representation of f with a new node containing $\tilde{p}$, the both sub-trees $Q_{\tilde{p}}(T)$ and $Q_{p_1}(T)$ are selected for the first merging stage and t' appears in red dots. (c) Result of the first merging stage ($M(\tilde{p}, \{p1\}, \tilde{t})$) with t'' in blue dots. (d) Extraction of t' (in red dots), $Q_{\tilde{p}}(T)$ and $Q_{p_1}(T)$ for the second merging stage. (e) Result of the second merging stage ($M(\tilde{t}, \{p1, p2\}, \tilde{t})$) with t'' in blue dots. (f) Third stage of merging with t' in red dots. (g) Result of the third merging stage ($M(\tilde{p}, \{p1, p2, p3\}, \tilde{t})$) with t'' in blue dots. (h) Last stage of merging with t' and t'' in. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Eq. (10) $(\forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{A}(s) > \mathcal{A}(n))$

- $t_3 \in \mathcal{T}^e \Rightarrow \forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(t_3, n)\ \mathcal{A}(s) > \mathcal{A}(n)$;
- $\mathcal{A}(\mathcal{R}(t_2)) = \mathcal{A}(\mathcal{R}(t_3))$ because $\mathcal{R}(t_2) = \mathcal{R}(t_3)$;
- thus $subst(t_1, t_2, t_3)$ verifies Eq. (10).

Eq. (11) $(\forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t,n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n))$

- $t_3 \in \mathcal{T}^e \Rightarrow \forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(t_3,n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$;
- but $\mathcal{P}(\mathcal{R}(t_2)) = \mathcal{P}(\mathcal{R}(t_3))$ because $\mathcal{R}(t_2) = \mathcal{R}(t_3)$;
- thus $subst(t_1, t_2, t_3)$ verifies Eq. (11).

Eq. (12) $(\forall n \in \mathcal{N}(t)\ \bigcap_{s\in\mathcal{W}(t,n)} \mathcal{P}(s) = \emptyset)$

- $t_3 \in \mathcal{T}^e \Rightarrow \forall n \in \mathcal{N}(t_3)\ \bigcup_{s\in\mathcal{W}(t_3,n)} \mathcal{P}(s) = \emptyset$;
- $\mathcal{P}(\mathcal{R}(t_2)) = \mathcal{P}(\mathcal{R}(t_3))$ because $\mathcal{R}(t_2) = \mathcal{R}(t_3)$;
- $subst(t_1, t_2, t_3)$ verifies Eq. (12).

Consequently, Theorem 3.3 is verified. □

Theorem 3.4. $\forall \tilde{p} \in \Omega, \forall P \in 2^{\Omega}, \forall \tilde{t} \in \mathcal{T}^e\ M(\tilde{p}, P, \tilde{t}) \in \mathcal{T}^e$.

Proof. See Appendix B.

Theorem 3.5. *Let $\tilde{p} \in \Omega$ and $(f, f') \in \mathcal{S}^2$ such that*

$$\begin{cases} f(\tilde{p}) < f'(\tilde{p}) \\ \forall\, p' \in \Omega, \quad f(p') = f'(p')\ \ if\ \ p' \neq \tilde{p} \end{cases}$$

Let $t \in \mathcal{T}^e$ be a representation of f and $\tilde{t}$ the tree such that

$$\mathcal{N}(\tilde{t}) = \mathcal{N}(t) \cup S_{\tilde{p}}^{f'(\tilde{p})} \tag{20}$$

$$\mathcal{B}(\tilde{t}) = \mathcal{B}(t) \cup \left(\underset{n\in\mathcal{N}(t)/\,\tilde{p}\in\mathcal{P}(n)}{\arg\max} \mathcal{A}(n), S_{\tilde{p}}^{f'(\tilde{p})} \right) \tag{21}$$

with S_p^{α} the node with a membership degree equal to α and that contains only p.

Under these conditions, $M(\tilde{p}, ngbh(\tilde{p}), \tilde{t})$ is a representation of f'.

Proof. See Appendix C.

This last theorem enables us to use the operator M to update a tree representing a set f into a tree representing a set f' that contains f. Its proof relies on Theorems 3.2 and 3.4.

4. Deflating a tree representing a fuzzy set

We now introduce a method to decrease the membership value to a fuzzy set at a given point. The proposed idea in order to update the tree representing this set is to remove all nodes that may not represent connected components of α-cuts of the new set anymore. When this first step is completed, the fuzzy set represented by the pruned tree can be inflated using the approach detailed in the previous section.

Definition 4.1. For a given tree representing a fuzzy set, and a location where the membership value of the set decreases, the operator $\mathcal{Y} : \mathcal{T}^e \times \Omega \times [0,1] \to \mathcal{T}^e$ returns a pruned version of the input tree:

$$\forall\, t \in \mathcal{T}^e, \forall \tilde{p} \in \Omega, \forall \alpha \in [0,1]$$

$$\mathcal{N}(\mathcal{Y}(t, \tilde{p}, \alpha)) = \mathcal{N}(t) \backslash Z^{\tilde{p}}_{f'(\tilde{p}),\alpha}(\mathcal{N}(t)) \tag{22}$$

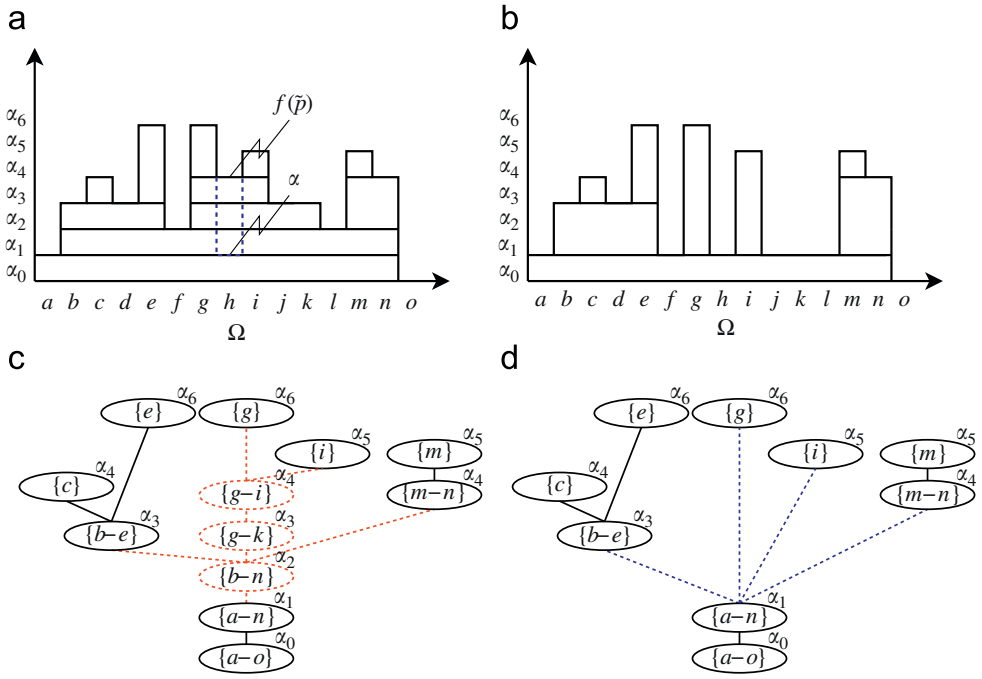


Fig. 11. Example of operator $\mathcal{Y}$ on set (a) at point h. The tree t presented in (c) is the representation of this set. The set (b) corresponds to the result of $\mathcal{Y}(t, h, \alpha)$ with its tree representation in (d). The nodes and the edges in red doted lines in (c) correspond to deleted elements in Eqs. (22) and (23), respectively. The edges in blue dots in (d) correspond to edges added in the last part of Eq. (23). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\mathcal{B}(\mathcal{Y}(t,\tilde{p},\alpha)) = \mathcal{B}(t)\backslash\{(n_1,n_2)\in\mathcal{B}(t)/(n_1\in Z^{\tilde{p}}_{f^t(\tilde{p}),\alpha}(\mathcal{N}(t)))\vee(n_2\in Z^{\tilde{p}}_{f^t(\tilde{p}),\alpha}(\mathcal{N}(t)))\}$$
$$\cup\left\{(n_1,n_2)\in\mathcal{N}(t)^2/\begin{array}{l}(n_2\notin Z^{\tilde{p}}_{f^t(\tilde{p}),\alpha}(\mathcal{N}(t)))\\ \wedge\ (\exists n_3\in Z^{\tilde{p}}_{f^t(\tilde{p}),\alpha}(\mathcal{N}(t))/n_2\in\mathcal{W}(n_3))\\ \wedge\ \left(n_1=\underset{n\in\mathcal{N}(t)/(n_2\in\mathcal{D}(n))\wedge(n\notin Z^{\tilde{p}}_{f^t(\tilde{p}),\alpha}(\mathcal{N}(t)))}{\arg\max}\ \mathcal{A}(n)\right)\end{array}\right\} \tag{23}$$

with f^t the fuzzy set represented by t and $\forall p\in\Omega, \forall\alpha_1\in[0,1], \forall\alpha_2\in[0,1], \forall N\in\mathcal{V}\ Z^p_{\alpha_1,\alpha_2}(N)=\{n\in N/(p\in\mathcal{P}(n))\wedge(\mathcal{A}(n)>\alpha_2)\wedge(\mathcal{A}(n)\leq\alpha_1)\}$.

This operator returns a tree whose nodes (Eq. (22)) are the ones of the input tree except those that contain $\tilde{p}$ and that have a membership degree that ranges between the old and the new membership degrees of the fuzzy set at point $\tilde{p}$ (in red dotted line in Fig. 11(c)). Similarly, the edges of this tree are the ones of the input tree without the ones that refer to the removed nodes (second part of Eq. (23), the removed elements appear in red dots in Fig. 11(c)). In order to make the result of $\mathcal{Y}$ still a tree, the edges between remaining isolated nodes and their direct ascendants in the original tree that are kept, are added (third part of Eq. (23), corresponding to the edges in blue dots in Fig. 11(d)).

Theorem 4.1. $\forall t\in\mathcal{T}^e, \forall n\in\mathcal{N}(t), \forall s\in\mathcal{D}(t,n)\ (\mathcal{P}(s)\subseteq\mathcal{P}(n))\wedge(\mathcal{A}(s)>\mathcal{A}(n))$.

Proof. Using recursively Eq. (10) on t leads to this property. □

Theorem 4.2. *The operator $\mathcal{Y}(t,\tilde{p},\alpha)$ returns a tree representing a fuzzy set included in the fuzzy set (f^t) represented by the input tree (t) if the degree α is less than $f^t(\tilde{p})$:*

$$\forall t\in\mathcal{T}^e, \forall\tilde{p}\in\Omega, \forall\alpha\in[0,1],\quad f^t(\tilde{p})>\alpha\Rightarrow f^{\mathcal{Y}(t,\tilde{p},\alpha)}\subset f^t$$

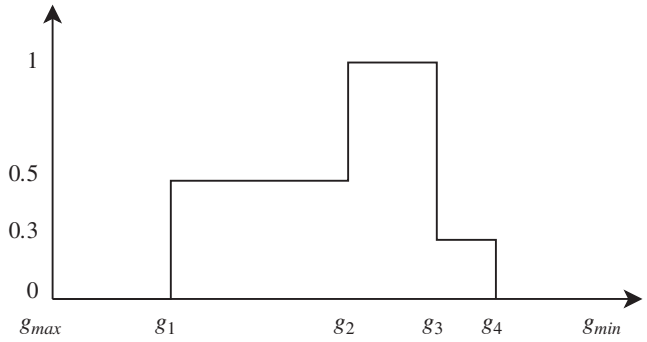


Fig. 12. Fuzzy number corresponding to the gray level at a point of a fuzzy number image. Higher gray levels appear on the left. This number is encoded by enumerating the following transitions: $(0; g_{max})$, $(0, 5; g_1)$, $(1; g_2)$, $(0.3; g_3)$, $(0; g_4)$.

Proof. See proof in Appendix D, which relies on Theorem 4.1.

Let us now introduce a last theorem that enables to implement the removal of nodes and edges, which is needed by the operator $\mathcal{Y}$, using the neighbors of points contained in nodes affected by the update of f at point $\tilde{p}$.

Theorem 4.3. *$\forall t \in \mathcal{T}^e$ representation of a fuzzy set $f, \forall (n_1, n_2) \in \mathcal{B}(t)$*

$$\mathcal{P}(n_1) \neq \mathcal{P}(n_2) \Rightarrow \exists (p_1 \in \mathcal{P}(n_1)) \wedge (p_2 \in \mathcal{P}(n_2)) \,/\, (f(p_1) = \mathcal{A}(n_1)) \wedge (p_1 \in ngbh(p_2))$$

Remark 4.1. This theorem means that for any pair of parent/child nodes that do not represent the same connected component, there exists a pair of neighboring points such that one is contained in both nodes and the other is only contained in the parent node and in no other node corresponding to a higher α-cut level. Thus, if we look at all the neighbors of all points p of a node n that verify $\mathcal{A}(n) = f(p)$, we can list all the nodes whose parent is n.

Proof. Let $t \in \mathcal{T}^e$ be a representation of a fuzzy set f. Let $(n_1, n_2) \in \mathcal{B}(t) / \mathcal{P}(n_1) \neq \mathcal{P}(n_2)$

- Eq. (11) ensures that $\mathcal{P}(n_2) \subset \mathcal{P}(n_1)$ because t is nested,
- Eq. (15) ensures also that $\mathcal{P}(n_1)$ is connected and that $\mathcal{P}(n_2)$ is connected since t is a representation,
- there exists at least one pair of neighboring points $(p_1, p_2) \in \mathcal{P}(n_1) \times \mathcal{P}(n_2)$ such that $p_1 \notin \mathcal{P}(n_2)$ since complete inclusion of $\mathcal{P}(n_2)$ in $\mathcal{P}(n_1)$ is forbidden,
- furthermore, $\nexists n_3 \in \mathcal{W}(t, n_1)/p_1 \in \mathcal{P}(n_3)$ because otherwise n_2 would not be a connected component of f and consequently t would not be a representation. This leads to $f(p_1) = \mathcal{A}(n_1)$.

To conclude, Theorem 4.3 is verified. □

Using operator $\mathcal{Y}$, we are able to produce a tree t' that represents a fuzzy set $f^{t'}$ included in the one represented by the input tree (t). Using the results of Section 3, we can produce from the tree t' a tree t'' representing a fuzzy sub-set $f^{t''}$ that differs from f^t only at one point. We can also imagine to decrease f^t at several locations before making the resulting set increase as it will be shown in the next section.

5. Application to filtering of fuzzy quantity images

In this section we present algorithms resulting from the former developments. Even if these algorithms can be used in a wide range of applications, we will illustrate them in the context of fuzzy image filtering [18,19]. First, we recall the definitions of fuzzy images and introduce a way to encode the gray levels of such images. Then algorithms for inflating and deflating fuzzy sets are proposed, and finally, an algorithm template is proposed to filter fuzzy images.

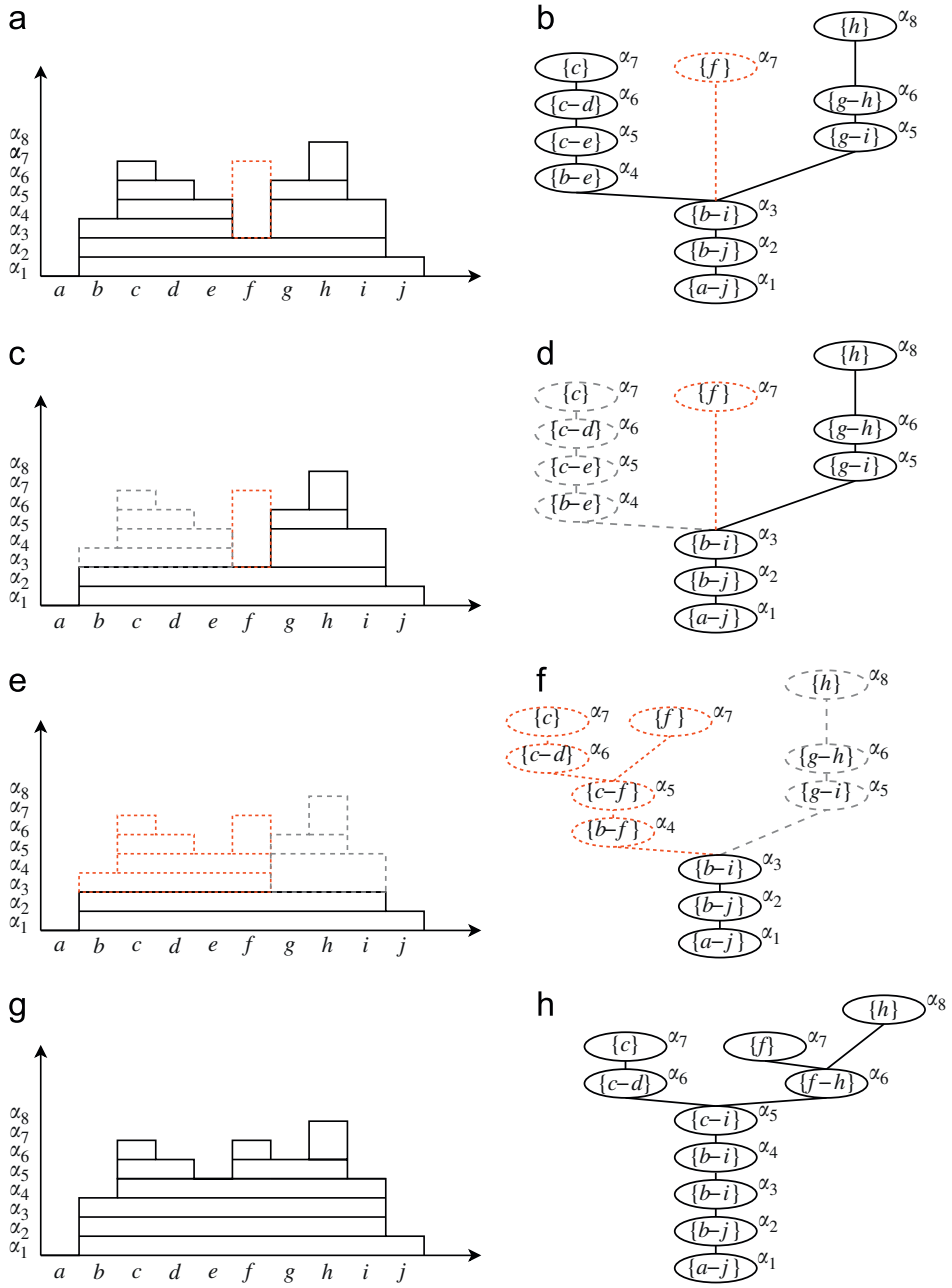


Fig. 13. Increase of a fuzzy set (a) value at point *f*. (c), (d) Selection of the sub-tree that contains *f* (in dotted lines) and selection of the sub-tree that contains its first neighbor (in dashed lines). (e), (f) Second stage of the merging. (g) Updated set. (h) Updated tree.

5.1. Fuzzy images and gray level representation

Fuzzy quantity images are images where pixels values are fuzzy quantities (fuzzy sets defined on the gray levels domain $\mathcal{G}$). They can be represented by fuzzy sets defined on $\Omega \times \mathcal{G}$ with Ω the image domain (see Figs. 1(a), (b)). $\mathcal{F}$ denotes the set of fuzzy images.

Definition 5.1. A fuzzy image $F \in \mathcal{F}$ is a fuzzy umbra image if:

$$\forall p \in \Omega, \forall (g_1, g_2) \in \mathcal{G}^2, \quad g_1 \leq g_2 \Rightarrow F(p, g_1) \geq F(p, g_2)$$

Definition 5.2. A fuzzy image $F \in \mathcal{F}$ is a fuzzy number image if:

$$\forall p \in \Omega, \quad F(p, *) \text{ is a fuzzy number}$$

In both cases, a fuzzy connected filter $\delta_\Psi : \mathcal{F} \to \mathcal{F}$ for a fuzzy gray-level image is expressed using a filter $\psi : \mathcal{S} \to \mathcal{S}$, which may depend on the gray level, applied to fuzzy sets defined on Ω extracted from the fuzzy image F for every gray level g: $\forall p \in \Omega, \forall g \in \mathcal{G}\ \delta_\Psi(F)(p, g) = \psi(F^g)(p)$, where $F^g = F(*, g)$. To be considered as a connected filter, ψ has to behave as a binary connected filter for each α-cut of the input fuzzy set (see [19] for more details on definitions and properties of fuzzy connected filters). From a concrete point of view, the max-tree representation is usually suitable to implement ψ.

Pixel values of fuzzy images can be represented in a compact way by encoding only their transitions [19]. In this section, we will use such a representation from highest gray levels to lowest ones as illustrated in Fig. 12.

5.2. Inflating a fuzzy set

In this subsection, we describe an algorithm to be used to transform a tree t, which represents a fuzzy set f, into a tree t', which represents a fuzzy set f' with $f < f'$. For the sake of clarity, we consider that f and f' only differ at a single point $\tilde{p}$. Fig. 13 shows such a case as well as the method we propose to update the tree. It is a simple illustrative example that may help understanding the main steps of the algorithm. The first step is to add to t a new node that only contains the point where f is modified (in dotted lines in Figs. 13(a) and (b)) with its degree set to the new one (see Eqs. (20) and (21)). When this is done, the idea is to merge this node with its neighbors in order to restore the property for every node to represent a connected component of the α-cuts of the new set. This merging step is done using the proposed operator M (see Definition 3.4) as follows: for each point p' that is a neighbor of $\tilde{p}$, we are looking for all the nodes of t that contain p' and that have a degree higher than $f(\tilde{p})$. When doing so, we select two sub-trees ($Q_{\tilde{p}}(T)$ and $Q_{p'}(T)$ as defined in Definition 3.4) associated to $\tilde{p}$ and p', respectively (in dotted and dashed lines in Figs. 13(c)–(f)). Then the two selected trees just have to be merged node by node starting from their roots (see Definition 3.1 of $\overline{M}$). This merging process is just done by adding points coming from both sub-trees for a given α: thus we partially restore the connected components of α-cuts as it can be seen in Figs. 13(g) and (h) (see Remark 3.2). Here partially, means that only connectivity induced by the neighborhood relationship between $\tilde{p}$ and p' is restored. Actually, this is because of the iteration of this process on all $p' \in ngbh(\tilde{p})$ that we are ensured to completely restore the connectivity.

Algorithm 1 summarizes the process we just described. Theorem 3.5 ensures that this algorithm provides a tree t' that represents f'. This algorithm is one important outcome of the proposed approach.

Algorithm 1. Update of a tree to reflect the increase of a fuzzy set.
let f be the set to modify;
let t be the tree that represents the fuzzy set f;
let $R^+ = \{(p_i, \alpha_i)\}$ be the set of points p_i where the function f is inflating;
while $R^+ \neq \emptyset$ **do**
  take a pair $(p, \alpha) \in R^+$;
  $R^+ \leftarrow R^+ \backslash \{(p, \alpha)\}$;
  create a node $n_0 | (\mathcal{A}(n_0) = \alpha) \wedge (\mathcal{P}(n_0) = \{p\})$;
  add n_0 to the list of children of the node n' that verify
  $(\mathcal{A}(n') = f(p)) \wedge (p \in \mathcal{P}(n'))$;
  forall $p_i \in ngbh(p)$ **do**
    let $n_i \in \mathcal{N}(t) | (p_i \in \mathcal{P}(n_i)) \wedge (\mathcal{A}(n_i) = \mu(p_i))$;
    let $S = \emptyset$ be a stack of nodes;
    while $n_i \neq n_0$ **do**
      if $\mathcal{A}(n_i) > \mathcal{A}(n_0)$ **then**
        pile up n_i in S;

```
        n_i ← parent(n_i);
      end
      else if A(n_i) < A(n_0) then
        pile up n_0 in S;
        n_0 ← parent(n_0);
      end
      else
        pile up n_i and n_0 in S;
        n_i ← parent(n_i);
        n_0 ← parent(n_0);
      end
    end
    let r ← n_0 be the common root;
    while S ≠ ∅ do
      n ← top of the stack S;
      P(r) ← P(r) ∪ P(n);
      if A(r) < A(n) then
        parent(n) ← r;
        r ← n
      end
    end
  end
end
```

Let us remark that the problem of merging sub-trees solved in this algorithm in order to increase the value of a fuzzy set is similar to the problem of merging two max-tree representations of a gray scale image on different sub-domains in the context of parallelization of max-trees construction [28]. Here the main difference is that the nodes are stored in a stack before being merged in order to fit the definition of $\overline{M}$, while in [28] nodes are merged directly. Nonetheless, even if both approaches solve the same tree-merging problem, the algorithm in [28] corresponds only to a sub-part of what we want to do here: correcting the topology of a tree to reflect the changes corresponding to the update of a membership value of the represented fuzzy set.

5.3. Deflating a fuzzy set

In this section, we present an algorithm that aims at transforming a tree t, which represents a fuzzy set f, into a tree t', which represents a fuzzy set f', when $f' < f$. Again, for the sake of clarity, we assume that f' and f differ only at one point $\tilde{p}$.

Deflating a fuzzy set is far more complicated than inflating it since removing a point from some nodes can break connected components leading to the split of nodes as it can be seen in Fig. 14(a).

The proposed approach consists in transforming the problem in the previous one where we want to increase the value of a set. In order to do that, all nodes that contain $\tilde{p}$ and whose degree ranges between $f(\tilde{p})$ and $f'(\tilde{p})$ are removed (operator $\mathcal{Y}$ of Definition 4.1). Since nodes that do not satisfy this last criterion cannot be impacted by the variation of $f(\tilde{p})$, it is useless to remove them.

In order to illustrate this process, we can refer to the example of Fig. 14 where the set f decreases at point g (Fig. 14(a)). The first step consists in removing all points whose membership degree is equal to α_5 and that belong to a node n that also contains g. In our example, it corresponds to d, g and i as illustrated in Fig. 14(c). This suppression is done implicitly by correcting edges that involve n and a node containing a neighbor of these points (see Theorem 4.3). Once this is done, the node n can safely be removed since it is no more the parent of any node. By iterating this process on the other nodes that contain g and whose degree is higher than $f'(g)$, we get a tree that represents a fuzzy set f'' included in f' (see Theorem 4.2). It is then possible to add points that were removed using Algorithm 1.

These steps are summarized in Algorithm 2, which constitutes another concrete result of our approach.

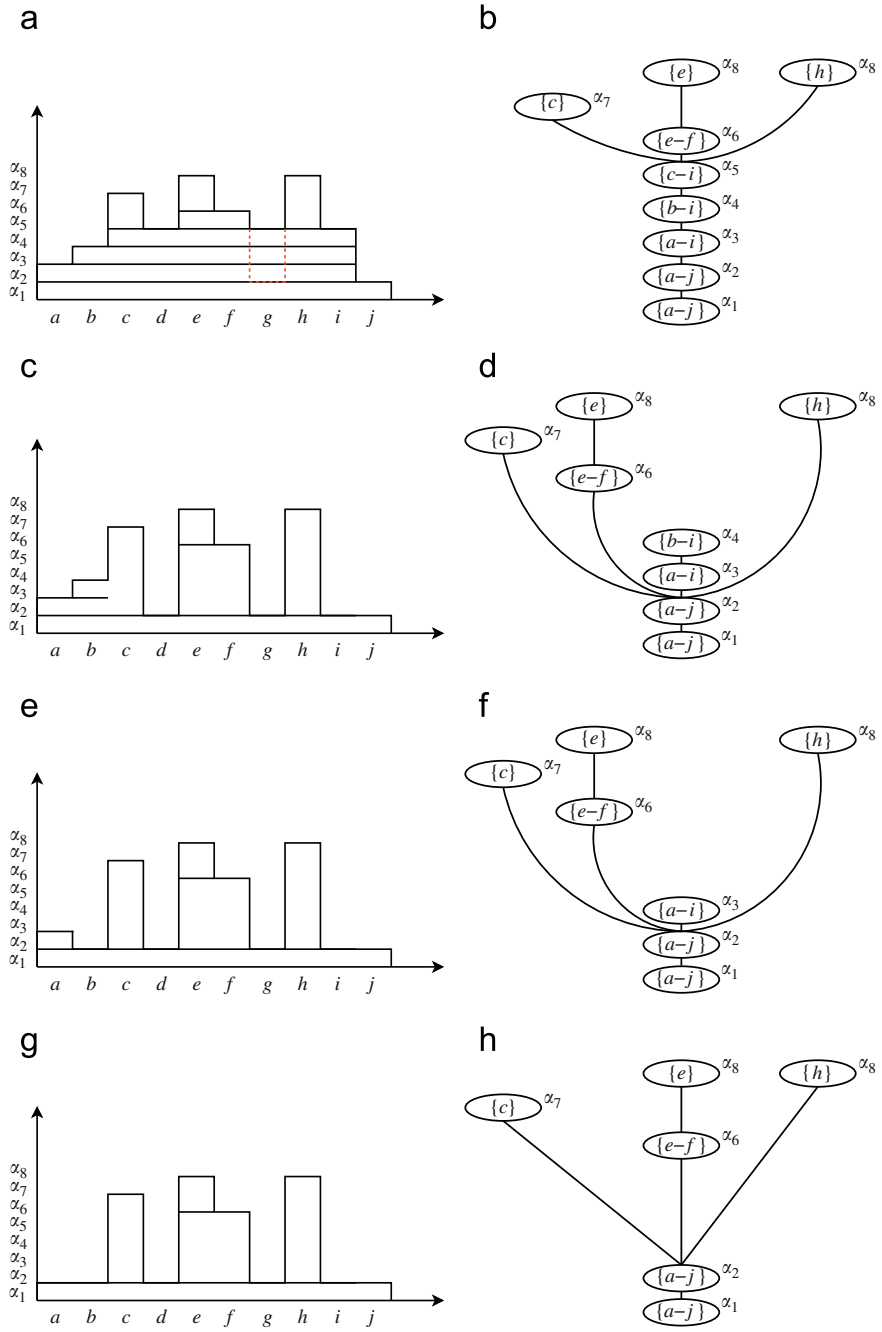


Fig. 14. Decrease of a fuzzy set (a) at point g. (c), (d) First stage: deletion of the node $\langle \alpha_5, c-i \rangle$. (e), (f) Second stage: deletion of the node $\langle \alpha_4, b-i \rangle$. (g), (h) Third stage: deletion of node $\langle \alpha_3, a-i \rangle$. In order to finalize the process, the final set (g) has to be increased at points a, b, d and i.

Algorithm 2. Update of a tree corresponding to the decrease of the value of a set.
let f be the fuzzy set to modify;
let t be the tree representing the fuzzy set f;
let $R^- = \{(p_i, \alpha_i)\}$ be the set of points p_i where f is supposed to deflate;
while $R \neq \emptyset$ **do**
  take a pair $(p, \alpha) \in R$;
  let $n_0 = \underset{n \in \mathcal{N}(n) | p \in \mathcal{P}(n)}{\arg\max} \mathcal{A}(n)$;
  if *si* $\mathcal{A}(n_0) > \alpha$ **then**

```
    while A(n0) > α, n0 ← parent(n0) do
      forall n|(p ∈ P(n)) ∧ (A(n) > α), do
        forall p' ∈ P(n)|f(p') = A(n) do
          R+ ← R+ ∪ {(p', f(p'))};
          forall p'' ∈ ngbh(p') do
            if ∃n'|(p'' ∈ P(n)) ∧ (parent(n') = n) then
              parent(n') ← n0;
            end
          end
          remove node n;
        end
      end
      R− = R−\(p, α);
    end
  end
  use Algorithm 1 on R+;
end
```

5.4. Filtering and discussion

The former algorithms enable to write a meta-algorithm (Algorithm 3) for fuzzy image filtering [18,19]. Thus to filter a fuzzy image F, we can start with a tree composed of only one node whose degree is 0 and that contains all points of Ω (top left tree in Fig. 15), in order to update it by introducing transitions between fuzzy sets F^g and F^{g-1} associated to gray levels g and $g-1$, respectively (left trees in Fig. 15). Using the obtained trees, a fuzzy connected operator δ_ψ can be applied (transformation of trees on the left into trees on the right in Fig. 15). It usually corresponds to extracting branches or sub-trees from the original trees with respect to the fuzzy connected component definition used [20–22] in order to keep or suppress them [18].

Algorithm 3. Fuzzy connected filter template algorithm.

```
let f ← 0_F;
let t ← ⟨0, X⟩ be the tree that represents the fuzzy set f;
let R = {(p_i, μ_i, g_i)} be the set of transitions of the fuzzy image (see Fig. 12);
sort R increasingly with respect to g_i;
while R ≠ ∅ do
  let g be the gray level associated to the first element of R;
  R+ ← {(p_i, α_i)|(p_i, α_i, g) ∈ R) ∧ (max{A(n)|p_i ∈ P(n)} > α_i)};
  R− ← {(p_i, α_i)|(p_i, α_i, g) ∈ R) ∧ (max{A(n)|p_i ∈ P(n)} < α_i)};
  R ← R\R+ ∪ R−;
  run Algorithm 2 using R−;
  run Algorithm 1 using R+;
  filter the updated tree t;
end
```

The gain of this update strategy in comparison to the direct computation of a tree for each gray level will depend on the number of points to be updated between two fuzzy sets. Actually, for a given point p of a fuzzy image F, the number of updates is linked to what the fuzzy quantity $F(p, *)$ looks like. Thus a fuzzy number with a large support will introduce more transitions, while a small quantization of the membership degrees will reduce the number of transitions. In the worst case, we have to update the tree for all the points of the image domain. Nonetheless, this implies a very large amount of fuzziness within the image, which is usually not desirable because of the difficulty to decide something meaningful when everything becomes possible.

Let us also note that calls to Algorithm 2 in Algorithm 3 are useless when the fuzzy image to filter is a fuzzy umbra image because of the decreasingness with respect to the gray levels (see Definition 5.1). Actually because of this

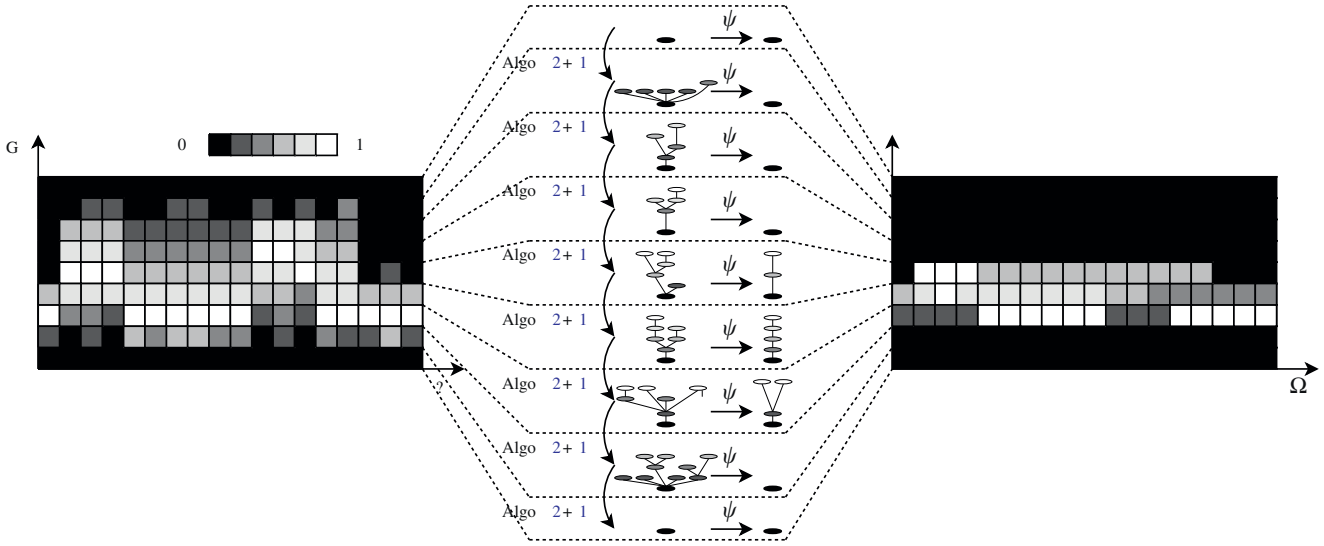
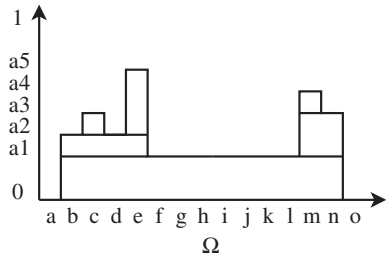


Fig. 15. Fuzzy number image filtering (original image on the left, filtered image on the right). The filtering of each gray level using an arbitrary fuzzy connected operator ψ is done using max-tree representations of the corresponding fuzzy sets. The trees (on the left) are iteratively updated using Algorithms 2 and 1 with respect to the differences between successive fuzzy sets in the original image, starting from a tree containing a single node representing Ω (tree on the top).

a



b

image point	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
node ID	0	1	2	3	4	5	6	6	6	7	8	9	10	11	12

c

node ID	0	1	2	3	4	5	6	7	8	9	10	11	12
level root	0	3	2	3	4	7	6	6	7	6	10	11	0

d

node ID	0	1	2	3	4	5	6	7	8	9	10	11	12
parent	0	∅	3	6	3	∅	6	∅	∅	∅	11	6	∅

Fig. 16. Data structure to represent a max-tree. (a) Fuzzy set to represent. (b) Node IDs associated to points in the image. (c) Representatives of nodes ($level\ \ root[n] = n \Rightarrow n$ is a representative). (d) Parents of nodes (∅ means information is useless since the node is not a representative node).

property, if we encode gray levels using the proposed approach (from higher to lower gray level), we are ensured that membership values of the fuzzy sets extracted from the fuzzy umbra image can only increase or stay unchanged when the gray level decreases.

From an implementation point of view, the way these algorithms are written does not impose a particular data representation. Of course, standard representations of max-trees can easily be adapted to be used by these algorithms. In our experiments, nodes were labeled by integers, and the merging of two nodes was done using a node chaining process (nodes may point to another node representing the same α-cut). Parenthood was expressed using an array between representatives of nodes. This data structure is illustrated in Fig. 16. Finally, we should note that although

nodes could be represented by image points instead of integers [11], in our case we cannot use such a trick because after the update of a tree, two nodes may contain exactly the same points. To avoid that, we should only use compact tree representations, which forbid such configurations.

Finally, the proposed algorithms suggest to filter a tree after the update process. This allows us to propose a generic approach to express a filter, but it may not be always efficient. Of course for real implementations, the attributes used to decide what to prune in the tree to filter can be computed/updated during the update of the tree.

6. Conclusion

Max-tree is a widely used structure to represent an image. While it is mainly used to implement connected filters on gray scale images, it can also represent fuzzy sets allowing to easily manipulate fuzzy connected components according to various definitions of fuzzy connectivity [20–22]. A new kind of filters was recently introduced, they extend the notion of connected filters to the fuzzy sets framework and are called fuzzy connected filters [18,19]. Since they are defined to work on fuzzy images (gray levels are fuzzy quantities), their implementation requires a max-tree representation of a fuzzy set for each gray level.

Using the idea that for successive gray levels these sets are very similar, we introduced efficient algorithms to reflect the changes between two sets into their tree representations. Thus with our approach, if we have a max-tree representation of a fuzzy set, we can update it to make it represent a modified version of the original set without recomputing the whole tree. This update process mainly relies on two distinct steps. The first one is about updating a tree to reflect an increase of the membership value of a fuzzy set at a given point. This is done by adding a new node containing this point with the new membership degree in order to restore the connectivity of α-cuts by merging this new node with nodes containing one of the neighbors of the considered point. The second step consists in updating a tree regarding the decrease of the membership value at a given point of the represented fuzzy set. This is done by deleting all the nodes that contain this point and that have a membership degree higher to the new degree of the fuzzy set at this point. This leads to a tree representing a fuzzy set included in the target fuzzy set which can be further processed using the first step. Combining these two methods, it is possible to update a tree to reflect any change between two fuzzy sets.

Even if this work was developed to filter fuzzy images, other applications may be considered. For instance recent works used similar results to what we propose to merge two sub-trees in order to implement a concurrent computation of a max-tree representation [28]. Nonetheless, let us note that this merging problem corresponds only to a part of our tree-update scheme (sub-trees selection, pruning, etc.). Finally, the formalism we used in this paper enabled to prove the validity of our whole approach.

In summary, the main contributions of this paper consist of a formal representation of fuzzy sets and fuzzy images using trees, and efficient algorithms for modifying membership values, relying on sound mathematical bases. These algorithms can be applied on concrete image filtering and segmentation problems, while coping with the additional complexity induced by the fuzziness. Therefore, filtering fuzzy images becomes now tractable. Applications of these filtering methods relying on the proposed implementation are illustrated in related works [19,29], which focus on assessing the impact of the proposed approach on real imaging problems.

Appendix A. Proof of Theorem 3.1

In order to prove Theorem 3.1, we will consider all possible configurations recursively as illustrated in Fig. 17. The idea is to show that Theorem 3.1 is true for terminal cases (i.e. cases that do not rely on extra calls to $\overline{M}$) and then to show that the other configurations only need the assumption that their recursive calls to $\overline{M}$ verify the theorem.

Proof. Let us prove this theorem recursively:

Terminal cases: Let $(t_1, t_2) \in \mathcal{T}^{e^2} / \forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2)\, \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset$ and $(p_1, p_2) \in \Omega^2$ and $t_3 = \overline{M}(t_1, t_2, p_1, p_2)$. Two terminal cases exist: cases 1 and 4:

Case 1: $(\mathcal{A}(\mathcal{R}(t_1)) = \mathcal{A}(\mathcal{R}(t_2))) \wedge ((|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 0) \vee (|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))| = 0))$.

In this case, we have $\forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(n)\ \mathcal{A}(s) > \mathcal{A}(n)$ (Eq. (10)) because:

- $\forall t \in st(t_1, \mathcal{R}(t_1))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{A}(s) > \mathcal{A}(n)$ since $t_1 \in \mathcal{T}^e$;
- $\forall t \in st(t_2, \mathcal{R}(t_2))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{A}(s) > \mathcal{A}(n)$ since $t_2 \in \mathcal{T}^e$;

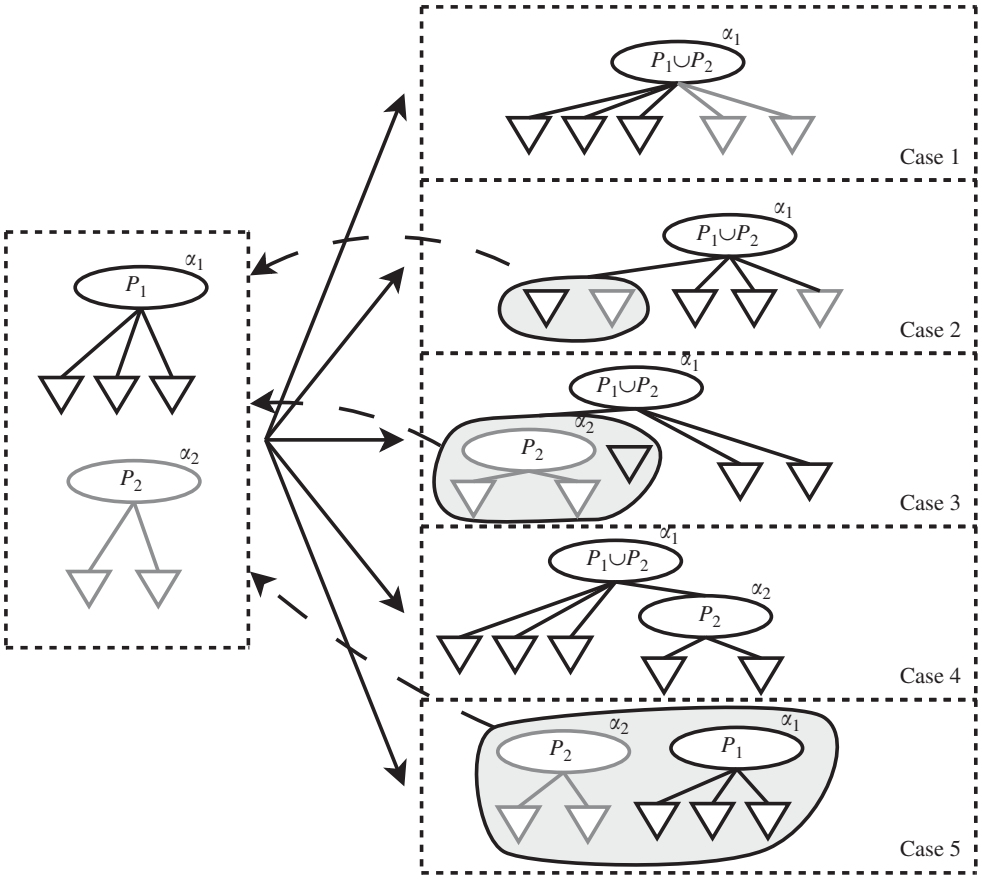


Fig. 17. Proof of Theorem 3.1. Cases 1 and 4 are terminal ones (the proof does not rely on any assumption about the two input sub-trees) and they verify the theorem. Cases 2, 3, 5 are true if the theorem is true for the merging of the sub-trees in the gray area. We recursively prove this (dashed arrows) using all new possible configurations (the five cases) till we get to a terminal case.

- $\forall n \in \mathcal{W}(t_1, \mathcal{R}(t_1)) \cup \mathcal{W}(t_2, \mathcal{R}(t_2))\ \mathcal{A}(\mathcal{R}(t_3)) < \mathcal{A}(n)$ since $\mathcal{A}(\mathcal{R}(t_3)) = \mathcal{A}(\mathcal{R}(t_1)) = \mathcal{A}(\mathcal{R}(t_2))$.

On the other hand, we also have $\forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(t_3, n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$ (Eq. (11)) because:

- $\forall t \in st(t_1, \mathcal{R}(t_1))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$ since $t_1 \in \mathcal{T}^e$;
- $\forall t \in st(t_2, \mathcal{R}(t_2))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$ since $t_2 \in \mathcal{T}^e$;
- $\forall n \in \mathcal{W}(t_1, \mathcal{R}(t_1)) \cup \mathcal{W}(t_2, \mathcal{R}(t_2))\ \mathcal{P}(n) \subseteq \mathcal{P}(\mathcal{R}(t_3))$ since $\mathcal{P}(\mathcal{R}(t_3)) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2))$.

Eventually, we have $\forall n \in \mathcal{N}(t_3) \bigcap_{s \in \mathcal{W}(t_3, n)} \mathcal{P}(s) = \emptyset$ (Eq. (12)) because:

- $\forall t \in st(t_1, \mathcal{R}(t_1))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n) \bigcap_{s \in \mathcal{W}(t,n)} \mathcal{P}(s) = \emptyset$ since $t_1 \in \mathcal{T}^e$;
- $\forall t \in st(t_2, \mathcal{R}(t_2))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n) \bigcap_{s \in \mathcal{W}(t,n)} \mathcal{P}(s) = \emptyset$ since $t_2 \in \mathcal{T}^e$;
- $\bigcap_{s \in \mathcal{W}(t_1, \mathcal{R}(t_1)) \cup \mathcal{W}(t_2, \mathcal{R}(t_2))} \mathcal{P}(s) = \emptyset$ since $\forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2)\ \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset$.

Thus $t_3 \in \mathcal{T}^e$. Furthermore, using the construction of t_3, we derive $\mathcal{A}(\mathcal{R}(\overline{M}(t_1, t_2, p_1, p_2))) = \min(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{A}(\mathcal{R}(t_2))) = \mathcal{A}(\mathcal{R}(t_1)) = \mathcal{A}(\mathcal{R}(t_2))$ and $\mathcal{P}(\mathcal{R}(\overline{M}(t_1, t_2, p_1, p_2))) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2))$, which means that Theorem 3.1 is verified.

Case 4: $(\mathcal{A}(\mathcal{R}(t_1)) < \mathcal{A}(\mathcal{R}(t_2))) \wedge (|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 0)$.

In that case, we can perform the same reasoning as the former one by substituting $\mathcal{W}(t_2)$ by t_2.

In both terminal cases, Theorem 3.1 is verified.

Non-terminal cases: Let $(t_1, t_2) \in \mathcal{T}^{e^2} / \forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2)\ \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset$ and $(p_1, p_2) \in \Omega^2$ and $t_3 = \overline{M}(t_1, t_2, p_1, p_2)$.

Case 2: $(\mathcal{A}(\mathcal{R}(t_1)) = \mathcal{A}(\mathcal{R}(t_2))) \wedge (|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| = 1) \wedge (|Q_{p_2}(st(t_2, \mathcal{R}(t_2)))| = 1)$.

If we assume Theorem 3.1 is verified for $Q_{p_1}(st(t_1, \mathcal{R}(t_1)))$, $Q_{p_2}(st(t_2, \mathcal{R}(t_2)))$, p_1, p_2, we get (Eq. (11)):

$$\forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(n), \quad \mathcal{P}(s) \subseteq \mathcal{P}(n)$$

because:

- $\forall t \in st(t_1, \mathcal{R}(t_1))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$;
- $\forall t \in st(t_2, \mathcal{R}(t_2))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$;
- $\forall n \in \mathcal{W}(t_1, \mathcal{R}(t_1)) \cup \mathcal{W}(t_2, \mathcal{R}(t_2))\ \mathcal{P}(n) \subseteq \mathcal{P}(\mathcal{R}(t_3))$ since $\mathcal{P}(\mathcal{R}(t_3)) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2))$;
- by hypothesis $\mathcal{P}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2))) = \mathcal{P}(\mathcal{R}((Q_{p_1}(st(t_1, \mathcal{R}(t_1))) \cup \mathcal{P}(Q_{p_2}(st(t_2, \mathcal{R}(t_2)))$.

Similarly (Eq. (10)):

$$\forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(n), \quad \mathcal{A}(s) > \mathcal{A}(n)$$

because:

- $\forall t \in st(t_1, \mathcal{R}(t_1))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{A}(s) > \mathcal{A}(n)$;
- $\forall t \in st(t_2, \mathcal{R}(t_2))), \forall n \in \mathcal{N}(t), \forall s \in \mathcal{W}(t, n)\ \mathcal{A}(s) > \mathcal{A}(n)$;
- $\forall n \in \mathcal{W}(t_1, \mathcal{R}(t_1)) \cup \mathcal{W}(t_2, \mathcal{R}(t_2))\ \mathcal{A}(n) > \mathcal{A}(\mathcal{R}(t_3))$ since $\mathcal{A}(\mathcal{R}(t_3)) = \min(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{A}(\mathcal{R}(t_2)))$;
- by hypothesis $\mathcal{A}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2))) = \min(\mathcal{A}(\mathcal{R}((Q_{p_1}(st(t_1, \mathcal{R}(t_1))), \mathcal{A}(Q_{p_2}(st(t_2, \mathcal{R}(t_2))))$.

Finally, we have (Eq. (12)):

$$\forall n \in \mathcal{N}(t_3), \quad \bigcap_{s \in \mathcal{W}(t_3, n)} \mathcal{P}(s) = \emptyset$$

because:

- $\forall t \in st(t_1, \mathcal{R}(t_1))), \forall n \in \mathcal{N}(t) \bigcap_{s \in \mathcal{W}(t,n)} \mathcal{P}(s) = \emptyset$;
- $\forall t \in st(t_2, \mathcal{R}(t_2))), \forall n \in \mathcal{N}(t) \bigcap_{s \in \mathcal{W}(t,n)} \mathcal{P}(s) = \emptyset$;
- $\bigcap_{s \in \cup\{\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2))\}}^{\mathcal{W}(t_1, \mathcal{R}(t_1)) \cup \mathcal{W}(t_2, \mathcal{R}(t_2))} \mathcal{P}(s) = \emptyset$ because
  - $\forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2)\ \ \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset$;
  - and $\mathcal{P}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), Q_{p_2}(st(t_2, \mathcal{R}(t_2))), p_1, p_2)))) = \mathcal{P}(\mathcal{R}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))))) \cup \mathcal{P}(\mathcal{R}(Q_{p_2}(st(t_2, \mathcal{R}(t_2))))))$;
  - and $\bigcap_{s \in \mathcal{W}(t_1, \mathcal{R}(t_1))} \mathcal{P}(s) = \emptyset$;
  - and $\bigcap_{s \in \mathcal{W}(t_2, \mathcal{R}(t_2))} \mathcal{P}(s) = \emptyset$.

Under the former assumptions, we derive that $t_3 \in \mathcal{T}^e$, and noting that $\mathcal{A}(\mathcal{R}(\overline{M}(t_1, t_2, p_1, p_2)) = \min(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{A}(\mathcal{R}(t_2)))$ and $\mathcal{P}(\mathcal{R}(\overline{M}(t_1, t_2, p_1, p_2))) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2)))$ leads us to the conclusion that under the same assumptions Theorem 3.1 is verified.

Case 3: $(\mathcal{A}(\mathcal{R}(t_1)) < \mathcal{A}(\mathcal{R}(t_2))) \wedge (|Q_{p_1}(st(t_1, \mathcal{R}(t_1)))| \neq 0))$.

Making the assumption that Theorem 3.1 is verified for $\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2)$ allows us to derive:

- $\forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(t_3, n)\ \mathcal{P}(s) \subseteq \mathcal{P}(n)$ (Eq. (11)) because:
  - $\mathcal{P}(\mathcal{R}(t_3)) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2)))$;
  - and $t_1 \in \mathcal{T}^e$;
  - and by hypothesis we have $\mathcal{P}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2))) = \mathcal{P}(\mathcal{R}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))))) \cup \mathcal{P}(\mathcal{R}(t_2))$;
- $\forall n \in \mathcal{N}(t_3), \forall s \in \mathcal{W}(t_3, n)\ \mathcal{A}(s) > \mathcal{A}(n)$ (Eq. (10)) because:
  - $\mathcal{A}(\mathcal{R}(t_3)) = \min(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{A}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2))))$;
  - and $t_1 \in \mathcal{T}^e$;

- and by hypothesis we have $\mathcal{A}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2))) = \min(\mathcal{A}(\mathcal{R}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))))), \mathcal{A}(\mathcal{R}(t_2)))$;

- $\forall n \in \mathcal{N}(t_3) \bigcap_{s \in \mathcal{W}(t_3,n)} \mathcal{P}(s) = \emptyset$ (Eq. (12)) because:
  - by hypothesis $\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2) \in \mathcal{T}^e$;
  - $\forall t \in st(t_3, \mathcal{R}(t_3)))\ \ t \in \mathcal{T}^e$;
  - $\bigcap_{s \in \mathcal{W}(t_3, \mathcal{R}(t_3))} \mathcal{P}(s) = \emptyset$ since $\forall n_1 \in \mathcal{N}(t_1), \forall n_2 \in \mathcal{N}(t_2)\ \ \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset$ and $\mathcal{P}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2))) = \mathcal{P}(\mathcal{R}(t_2)) \cup \mathcal{P}(\mathcal{R}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))))))$.

Thus, under the previous assumption, $t_3 \in \mathcal{T}^e$. Furthermore, if we remark that $\mathcal{A}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2))) = \min(\mathcal{A}(\mathcal{R}(t_1)), \mathcal{A}(\mathcal{R}(t_2)))$ and that $\mathcal{P}(\mathcal{R}(\overline{M}(Q_{p_1}(st(t_1, \mathcal{R}(t_1))), t_2, p_1, p_2))) = \mathcal{P}(\mathcal{R}(t_1)) \cup \mathcal{P}(\mathcal{R}(t_2)))$, we conclude that Theorem 3.1 is verified.

Case 5: In the other configurations, we can come back to one of the previous cases ($\overline{M}(t_1, t_2, p_1, p_2) = \overline{M}(t_2, t_1, p_2, p_1)$).

Conclusion: Theorem 3.1 is therefore true for all (t_1, t_2) in $\mathcal{T}^{e^2}$ and for all (p_1, p_2) in Ω^2 since it holds for all terminal and non-terminal cases. □

Appendix B. Proof of Theorem 3.4

Proof. Let $\tilde{p} \in \Omega$, $P = \{p_1..p_k\} \in 2^{\Omega}$, $\tilde{t} \in \mathcal{T}^e$. Let the hypothesis $H_i, i \in [[0..k]]$ be defined as: $M(\tilde{p}, \{p_1..p_i\}, \tilde{t}) \in \mathcal{T}^e$

Initialization: By definition, $M(\tilde{p}, \{\}, \tilde{t}) = \tilde{t} \in \mathcal{T}^e$ so H_0 is true.

Induction step: Let us assume that H_i is true

- $H_i \Rightarrow M(\tilde{p}, \{p_1..p_i\}, \tilde{t}) \in \mathcal{T}^e$;
- $M(\tilde{p}, \{p_1..p_{i+1}\}, \tilde{t}) = subst(M(\tilde{p}, \{p_1..p_i\}, t), t', t'')$ with, using the definition of M, $\mathcal{R}(t') = \mathcal{R}(t'')$;
- Let us show that $t'' \in \mathcal{T}^e$
  - $Q_{\tilde{p}}(T)$ and $Q_{p_i}(T)$ are sub-trees of a same nested tree because H_i is true.
  - Furthermore, by definition of M, $\mathcal{R}(Q_{\tilde{p}}(T)) \in \mathcal{W}(t', \mathcal{R}(t'))$ and $\mathcal{R}(Q_{p_i}(T)) \in \mathcal{W}(t', \mathcal{R}(t'))$ thus $\mathcal{P}(\mathcal{R}(Q_{\tilde{p}})) \cap \mathcal{P}(\mathcal{R}(Q_{p_i})) = \emptyset$.
  - Applying Eq. (11) on $Q_{\tilde{p}}(T)$ and $Q_{p_i}(T)$, we get $\forall n_1 \in \mathcal{N}(Q_{\tilde{p}}(T)), \forall n_2 \in \mathcal{N}(Q_{p_i}(T))\ \mathcal{P}(n_1) \cap \mathcal{P}(n_2) = \emptyset$.
  - Thus conditions of Theorem 3.1 are verified, so we have:
    - $\overline{M}(Q_{\tilde{p}}(T), Q_{p_i}(T), \tilde{p}, p_i) \in \mathcal{T}^e$;
    - $\mathcal{A}(\mathcal{R}(\overline{M}(Q_{\tilde{p}}(T), Q_{p_i}(T), \tilde{p}, p_i))) = \min(\mathcal{A}(\mathcal{R}(Q_{\tilde{p}}(T))), \mathcal{A}(\mathcal{R}(Q_{p_i}(T))))$;
    - $\mathcal{P}(\mathcal{R}(\overline{M}(Q_{\tilde{p}}(T), Q_{p_i}(T), \tilde{p}, p_i))) = \mathcal{P}(\mathcal{R}(Q_{\tilde{p}}(T))) \cup \mathcal{P}(\mathcal{R}(Q_{p_i}(T)))$.
  - By definition of t'', the only nodes and edges of t' that can be modified are the ones of $Q_{\tilde{p}}(T)$ and $Q_{p_i}(T)$. But, since these two last trees are replaced by $\overline{M}(Q_{\tilde{p}}(T), Q_{p_i}(T), \tilde{p}, p_i)$, which is a nested tree whose the root membership degree is the minimum of the ones of $Q_{\tilde{p}}(T)$ and $Q_{p_i}(T)$, and since the set of points of this new root is the union of $\mathcal{P}(\mathcal{R}(Q_{\tilde{p}}(T)))$ and $\mathcal{P}(\mathcal{R}(Q_{p_i}(T)))$, we have $t'' \in \mathcal{T}^e$.
  - Since $M(\tilde{p}, \{p_1..p_i\}, \tilde{t}) \in \mathcal{T}^e$ and $t' \in \mathcal{T}^e$ and $t'' \in \mathcal{T}^e$, we can use Theorem 3.3 and conclude that H_{i+1} is true.

Conclusion: Since H_0 is true and since $\forall i \in [[0..k-1]]\ H_i \Rightarrow H_{i+1}$, we can conclude that H_i is true for any i. Moreover, the proof of H_i is valid for any given P, $\tilde{p}$ and $\tilde{t}$ without any assumption on them, thus we can conclude that Theorem 3.4 is verified. □

Appendix C. Proof of Theorem 3.5

Proof. Let us verify Eqs. (13)–(15).

Eq. (13). $M(\tilde{p}, ngbh(\tilde{p}), \tilde{t}) \in \mathcal{T}^e$

The tree $\tilde{t}$ is nested because $\tilde{p}$ belongs to the parent of $S_{\tilde{p}}^{f'(\tilde{p})}$ whose membership degree is higher than the one of its parent (by definition $f'(\tilde{p} > f(\tilde{p}))$), and because no sibling of $S_{\tilde{p}}^{f'(\tilde{p})}$ contains $\tilde{p}$. Using Theorem 3.4, we get $M(\tilde{p}, ngbh(\tilde{p}), \tilde{t}) \in \mathcal{T}^e$.

Eq. (14). $\forall p \in \Omega, \forall \alpha \in [0,1]\ p \in f_\alpha \Rightarrow \exists n \in \mathcal{N}(M(\tilde{p}, ngbh(\tilde{p}), \tilde{t}))/\mathcal{P}(n) = \Gamma^p_{f_\alpha}$.

Let $p \in \Omega$ and $\alpha \in]f(\tilde{p}), 1]$. Let $\{p_1..p_k\} = ngbh(\tilde{p})$ the set of k neighbors of $\tilde{p}$. Let $H_i, i \leq k$, be the following induction hypothesis:

$$p \in \Gamma^{\tilde{p}}_{f'_\alpha} \Rightarrow \exists n \in \mathcal{N}(M(\tilde{p}, \{p_1..p_i\}, \tilde{t}))/\mathcal{P}(n) = \bigcup_{p' \in \{p_1..p_i\}/p' \in f_\alpha} \Gamma^{p'}_{f_\alpha} \cup \{\tilde{p}\}$$

Initialization: $M(\tilde{p}, \{\}, \tilde{t}) = \tilde{t}$, but $S_{\tilde{p}}^{f'(\tilde{p})} \in \mathcal{N}(\tilde{t})$ thus H_0 is true.

Induction step: Let us assume H_i true

- $M(\tilde{p}, \{p_1..p_{i+1}\}, \tilde{t}) = subst(M(\tilde{p}, \{p_1..p_i\}, \tilde{t}), t', t'')$;
- by hypothesis, there exists $n_1 \in \mathcal{N}(M(\tilde{p}, \{p_1..p_i\}, \tilde{t}))$ such that $\mathcal{P}(n_1) = \bigcup_{p' \in \{p_1..p_i\}/p' \in f_\alpha} \Gamma^{p'}_{f_\alpha} \cup \{\tilde{p}\}$;
- if $p_{i+1} \in f_\alpha$ (or even $p_{i+1} \in f'_\alpha$) then $\exists n_2 \in \mathcal{N}(t')/\mathcal{P}(n_2) = \Gamma^{p_{i+1}}_{f_\alpha}$;
- since p_{i+1} and $\tilde{p}$ are neighbors, we are ensured to be able to consider a couple (n_1, n_2) of compatible nodes that are consequently merged by the operator $\overline{M}$ (see Theorem 3.2);
- thus, there exists a node $\tilde{n}$ in t'' and consequently in $M(\tilde{p}, \{p_1..p_{i+1}\}, \tilde{t})$ that verifies $\mathcal{P}(\tilde{n}) = \bigcup_{p' \in \{p_1..p_{i+1}\}/p' \in f_\alpha} \Gamma^{p'}_{f_\alpha} \cup \{\tilde{p}\}$, which means that H_{i+1} is true.

Conclusion: By induction, H_i is true for any $i \in [[0..k]]$.

If we use this result, with $\{p_1..p_k\}$ the set $ngbh(\tilde{p})$ of neighbors of $\tilde{p}$, and if we remark that for configurations $\alpha \in [0,1]$, $p \in \Omega$ such that $p \notin \Gamma^{\tilde{p}}_{f'_\alpha}$ we have $\Gamma^p_{f_\alpha} = \Gamma^p_{f'_\alpha}$ and that concerned nodes are not modified by M, we derive that Eq. (14) is verified.

Eq. (15). $\forall n \in \mathcal{N}(M(\tilde{p}, ngbh(\tilde{p}), \tilde{t})), \forall p \in \Omega\ p \in \mathcal{P}(n) \Rightarrow \Gamma^p_{f_{\mathcal{A}(n)}} = \mathcal{P}(n)$.

Now if we remark that nodes $n \in \mathcal{N}(t)$ that contain a neighbor of $\tilde{p}$ verifying $f(\tilde{p}) < \mathcal{A}(n) \leq f'(\tilde{p})$ (i.e. the nodes that do not represent a connected component in f') are removed by applications of $\overline{M}$, and that, as previously shown, new nodes correspond to connected components of f' that were not already in f, we can conclude that Eq. (15) is verified.

Conclusion: Since $M(\tilde{p}, ngbh(\tilde{p}), \tilde{t})$ verifies Eqs. (13)–(15), it is a representation of f'. □

Remark C.1. In the induction step of the demonstration of Eq. (14), there may be several pairs of nodes (n_1, n_2) verifying $\mathcal{P}(n_1) = \bigcup_{p' \in \{p_1..p_i\}/p' \in f_\alpha} \Gamma^{p'}_{f_\alpha} \cup \{\tilde{p}\}$ and $\mathcal{P}(n_2) = \Gamma^{p_{i+1}}_{f_\alpha}$. Nonetheless, only one of them is composed of compatible nodes. For instance, in Fig. 10(d), for $\alpha_4 < \alpha \leq \alpha_5$, there is only one candidate for n_1 ($\mathcal{P}(n_1) = E \cup \{\tilde{p}\}$ and $\mathcal{A}(n_1) = \alpha_5$) and two candidates for n_2 ($\mathcal{P}(n_2) = D$ and $\mathcal{A}(n_2) = \alpha_5$ or $\mathcal{A}(n_2) = \alpha_6$) but only the node n_2 that verifies $\mathcal{A}(n_2) = \alpha_5$ is compatible with n_1. In the same example, cf. Fig. 10(e), these nodes produce a node $\tilde{n}$ such that $\mathcal{P}(\tilde{n}) = E \cup D \cup \{\tilde{p}\}$ and $\mathcal{A}(\tilde{n}) = \alpha_5$.

Appendix D. Proof of Theorem 4.2

Proof. Let $t \in \mathcal{T}^e$, $\tilde{p} \in \Omega$ and $\alpha \in [0,1]$.

Let us show that $\mathcal{Y}(t, \tilde{p}, \alpha)$ is a nested tree ($\mathcal{Y}(t, \tilde{p}, \alpha) \in \mathcal{T}^e$):

Let us show that Eqs. (10) *and* (11) *are verified* $\forall n \in \mathcal{N}(\mathcal{Y}(t, \tilde{p}, \alpha)), \forall s \in \mathcal{W}(\mathcal{Y}(t, \tilde{p}, \alpha), n)$, two cases are possible:

- either $s \in \mathcal{W}(t, n)$, which implies that Eqs. (10) and (11) are verified because t is nested,
- or $s \notin \mathcal{W}(t, n)$, in that case, using Eq. (23) we have that $s \in \mathcal{D}(t, n)$, which allows us to use Theorem 4.1 in order to conclude that Eqs. (10) and (11) are verified.

Let us show that Eq. (12) *is verified* $\forall n \in \mathcal{N}(\mathcal{Y}(t, \tilde{p}, \alpha)), \forall s \in \mathcal{W}(\mathcal{Y}(t, \tilde{p}, \alpha), n)$, we exclusively have:

- either $s \in \mathcal{W}(t, n)$,
- or $\exists! s' \in \mathcal{W}(t, n)/(s \in \mathcal{D}(t, s')) \wedge (\forall s'' \neq s \in \mathcal{W}(\mathcal{Y}(t, \tilde{p}, \alpha), n)\ s'' \notin \mathcal{D}(t, s'))$.

This means that each child of a given node is either replaced by itself or by one of its descendants (alternatively it can disappear).

Using Theorem 4.1, Eq. (12) is verified. Thus, $\mathcal{Y}(t, \tilde{p}, \alpha)$ is nested.

Let us show now that $\mathcal{Y}(t, \tilde{p}, \alpha)$ is a representation of $f' \subseteq f^t$ defined as follows:

$$\forall\, p \in \Omega$$

$$f'(p) = \max\{\beta \in [0,1] / (p \in f^t_\beta) \wedge ((\beta \leq f'(\tilde{p})) \vee (\beta > f^t(\tilde{p})) \vee (p \notin \Gamma^{\tilde{p}}_{f^t_\beta}))\} \tag{24}$$

Let us show that Eq. (14) is verified. Let $\beta \in [0,1]$ and $p \in f'_\beta$.

- if $\beta > f^t(\tilde{p})$ $\Gamma^p_{f'_\beta} = \Gamma^p_{f^t_\beta}$ and the corresponding nodes are kept in $\mathcal{Y}(t, \tilde{p}, \alpha)$,
- if $\beta \leq \max \mathcal{A}(n)_{n \in \mathcal{N}(t)/(\tilde{p} \in \mathcal{P}(n)) \wedge \mathcal{A}(n) \leq \alpha}$, we are in the same case,
- otherwise $\max \mathcal{A}(n)_{n \in \mathcal{N}(t)/(\tilde{\in} \mathcal{P}(n)) \wedge \mathcal{A}(n) \leq \alpha} < \beta \leq f^t(\tilde{p})$ and for such a membership degree, only nodes of t that do not contain $\tilde{p}$ are kept in $\mathcal{Y}(t, \tilde{p}, \alpha)$ (cf. Eq. (22)). Furthermore, if $p \notin \Gamma^{\tilde{p}}_{f_\beta}$, we have $\Gamma^p_{f'_\beta} = \Gamma^p_{f_\beta}$. Finally if $p \in \Gamma^{\tilde{p}}_{f_\beta}$ there exists a $\beta' \leq \max \mathcal{A}(n)_{n \in \mathcal{N}(t)/(\tilde{\in} \mathcal{P}(n)) \wedge \mathcal{A}(n) \leq \alpha}$ or $\beta' > f^t(p)$ that verifies $\Gamma^p_{f'_\beta} = \Gamma^p_{f'_{\beta'}}$ (cf. Eq. (24)), and we can refer to one of the two first cases.

Consequently, the tree $\mathcal{Y}(t, \tilde{p}, \alpha)$ verifies Eq. (14).

Let us show that Eq. (15) is verified. By construction of $\mathcal{Y}(t, \tilde{p}, \alpha)$ (Eq. (22)) and by definition of f' (Eq. (24)), Eq. (15) is verified.

To conclude, Theorem 4.2 is verified. □

Remark D.1. The expression of f' given at Eq. (24) comes from Eqs. (22) and (23) where the operator $\mathcal{Y}$ only deletes nodes (see Eq. (22)) which correspond to nodes in $\mathcal{N}(t)$ that have a degree ranging from α to $f^t(\tilde{p})$ and that contain $\tilde{p}$ (see definition of $Z^{\tilde{p}}_{f^t(\tilde{p}),\alpha}$). Figs. 11(b) and (d) illustrate Eq. (24).

References

[1] L. Vincent, Grayscale area openings and closings, their efficient implementation and applications, in: First Workshop on Mathematical Morphology and its Applications to Signal Processing, Barcelona, Spain, 1993, pp. 22–27.

[2] J. Serra, P. Salembier, Connected operators and pyramids, in: Image Algebra and Morphological Image Processing IV, SPIE Proceedings, Vol. 2030, 1993, pp. 65–76.

[3] P. Salembier, J. Serra, Flat zones filtering, connected operators, and filters by reconstruction, IEEE Transactions on Image Processing 4 (8) (1995) 1153–1160.

[4] H. Heijmans, Connected Morphological Operators and Filters for Binary Images, in: IEEE Internat. Conf. on Image Processing, Vol. 2, 1997, pp. 211–214.

[5] C. Vachier, Morphological scale-space analysis and feature extraction, in: IEEE Internat. Conf. on Image Processing, 2001, pp. III:676–III:679.

[6] F. Meyer, From connected operators to levelings, in: Proc. Fourth Internat. Symp. on Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer Academic Publishers, Norwell, MA, USA, 1998, pp. 191–198.

[7] F. Meyer, The levelings, in: Proc. Fourth Internat. Symp. on Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer Academic Publishers, Norwell, MA, USA, 1998, pp. 199–206.

[8] J. Crespo, R.W. Schafer, J. Serra, C. Gratin, F. Meyer, The flat zone approach: a general low-level region merging segmentation method, Signal Processing 62 (1) (1997) 37–60.

[9] P. Salembier, A. Oliveras, L. Garrido, Antiextensive connected operators for image and sequence processing, IEEE Transactions on Image Processing 7 (4) (1998) 555–570.

[10] A. Meijster, M.H.F. Wilkinson, A comparison of algorithms for connected set openings and closings, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 484–494.

[11] C. Berger, T. Géraud, R. Levillain, N. Widynski, A. Baillard, E. Bertin, Effective component tree computation with application to pattern recognition in astronomical imaging, in: IEEE Internat. Conf. on Image Processing, IEEE, New York, 2007, pp. 41–44.

[12] M. Nachtegael, T. Mélange, E.E. Kerre, The possibilities of fuzzy logic in image processing, in: Second Internat. Conf. Pattern Recognition and Machine Intelligence, Lecture Notes in Computer Science, Vol. 4815, Springer, Berlin, 2007, pp. 198–208.

[13] P. Couto, M. Pagola, H. Bustince, E. Barrenechea, P. Melo-Pinto, Image segmentation using A-IFSs, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Malaga, Spain, 2008, pp. 1620–1627.

[14] H.R. Tizhoosh, Type II fuzzy image segmentation, in: Fuzzy Sets and their Extensions: Representation, Aggregation and Models, Springer, Berlin, pp. 607–619.

[15] S. Tehami, A. Bigand, O. Colot, Color image segmentation based on type-2 fuzzy sets and region merging, Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Science, Vol. 4678, Springer, Berlin, 2007, pp. 943–954.
[16] S. Schulte, B. Huysmans, A. Pizurica, E. Kerre, W. Philips, A new fuzzy-based wavelet shrinkage image denoising technique, in: Advanced Concepts for Intelligent Vision Systems, 2006, pp. 12–23.
[17] S. Schulte, M. Nachtegael, V. de Witte, D. van der Weken, E. Kerre, A fuzzy impulse noise detection and reduction method, IEEE Transactions on Image Processing 15 (5) (2006) 1153–1162.
[18] G. Palma, I. Bloch, S. Muller, Fuzzy connected filters for fuzzy gray scale images, in: Information Processing and Management of Uncertainty in Knowledge-based Systems, Malaga, Spain, 2008, pp. 667–674.
[19] G. Palma, O. Nempont, I. Bloch, S. Muller, Extraction de “zones plates floues” dans des images de quantités floues, in: Rencontre Francophones sur la Logique Floue et ses Applications, Lens, France, 2008.
[20] A. Rosenfeld, Fuzzy digital topology, Information and Control 40 (1979) 76–87.
[21] U. Braga-Neto, J. Goutsias, A theoretical tour of connectivity in image processing and analysis, Journal of Mathematical Imaging and Vision 19 (1) (2003) 5–31.
[22] O. Nempont, J. Atif, E. Angelini, I. Bloch, A new fuzzy connectivity class. Application to structural recognition in images, in: Discrete Geometry for Computer Imagery, Lyon, 2008, pp. 446–557.
[23] E.J. Breen, R. Jones, Attribute openings, thinnings, and granulometries, Computer Vision and Image Understanding 64 (3) (1996) 377–389.
[24] P. Soille, Constrained connectivity for hierarchical image partitioning and simplification, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (7) (2008) 1132–1145.
[25] F. Meyer, P. Maragos, Morphological scale-space representation with levelings, in: Proc. Second Internat. Conf. on Scale-Space Theories in Computer Vision, Springer, London, UK, 1999, pp. 187–198.
[26] F. Meyer, P. Maragos, Nonlinear scale-space representation with morphological levelings, Journal of Visual Communication and Image Representation 11 (2000) 200.
[27] L. Najman, M. Couprie, Building the component tree in quasi-linear time, IEEE Transactions on Image Processing 15 (11) (2006) 3531–3539.
[28] M.H. Wilkinson, H. Gao, W.H. Hesselink, J.-E. Jonker, A. Meijster, Concurrent computation of attribute filters on shared memory parallel machines, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (10) (2008) 1800–1813.
[29] G. Palma, I. Bloch, S. Muller, R. Iordache, Fuzzifying images using fuzzy wavelet denoising, in: IEEE Internat. Conf. on Fuzzy Systems FUZZ-IEEE’09, Jeju, Korea, 2009.