Component-hypertrees for image segmentation

Nicolas Passat, Benoît Naegel

Component-hypertrees for image segmentation

Nicolas Passat[(1)] and Benoît Naegel[(2)]

(1) Université de Strasbourg, LSIIT, UMR CNRS 7005, France
(2) Université Nancy 1, LORIA, UMR CNRS 7503, France
passat@unistra.fr, benoit.naegel@loria.fr

Abstract. An interactive segmentation method, based on component-trees, has been proposed recently. In this article, we describe an extension of this method relying on mask-based connectivity. This improved version uses a *component-hypertree*, which models the component-trees of an image at various connectivity levels, and the relations of the nodes/connected components between these levels. The use of this data structure is less costly than the use of several component-trees. In particular, its (partial) processing enables to compute segmentation results at all the considered connectivity levels, and to choose the most satisfactory one. Application examples illustrate the relevance of this approach.

Key words: Component-tree, segmentation, mask-based connectivity, hypertree.

1 Introduction

The component-tree [1] is a graph-based structure which models some characteristics of a grey-level image by considering its binary level-sets obtained from successive thresholdings. It is particularly well-suited for the design of fast segmentation methods [2, 3], based on hypotheses related to the topology (connectedness) and the specific intensity (locally extremal) of structures of interest.

In this context, a new segmentation method has been proposed recently [4]. For a given (manual) presegmentation, it computes the best segmentation induced by the nodes of the image component-tree. By opposition to the standard component-tree approaches, based on attributes [5], this "best segmentation" is here the one which minimises a cost function modelling false positives/negatives between manual approximate segmentation and the proposed solution.

This method provides results for a given connectivity (in general, a connectivity induced by the standard adjacencies on $\mathbb{Z}^n$). In order to take into account simultaneously several (richer) connectivities, and in particular mask-based connectivities [6], an extension of this method is proposed. Based on a *component-hypertree*, which models the component-trees of a same image at various connectivity levels (induced by increasing masks), and the decomposition of each node/connected component in these successive component-trees, it computes all the segmentation results induced by these connectivity levels.

The article is organised as follows. In Sec. 2, a (short) state of the art related to component-tree segmentation is proposed. In Sec. 3, the segmentation method

defined in [4] is recalled. Sec. 4 presents the component-hypertree, and how to use it to segment an image at various connectivity levels, in the same way as in Sec. 3. In Sec. 5, experiments illustrate the behaviour of this technique.

2 Previous works

Component-trees have been involved in the development of several applications related to image filtering and/or segmentation [6–11]. All these proposed methods have been designed to detect the structures of interest by using information related to the value of attributes stored at each node of the tree. In such strategies, an attribute, or more generally a set of attributes [5], are chosen according to the hypotheses related to the applicative context. These attributes are assumed to model some characteristic properties of the structures of interest, and can be used in different ways:

- the desired values of the attributes can be chosen by the user in order to select the relevant nodes inducing the correct segmentation [6–8, 10];
- these values can be determined by analysing the signature of the attributes (*i.e.*, their evolution w.r.t. the grey-level of the nodes) [2];
- they can be learnt from examples, *e.g.*, by providing a ground-truth characterising the shape of the objects to detect [9], or by feeding a classification process when the set of attributes becomes too large [11–13].

In such works, component-trees have been used for their ability to discriminate nodes w.r.t. attributes, leading to automated/parametric methods.

It is however possible to directly use the component-tree structure by taking advantage of the decomposition of the image into nodes/connected components that it provides, in order to perform interactive segmentation. Methods based on such an alternative strategy should compute a segmentation result, no longer thanks to node attributes, but to a user-defined approximate result, which should then be matched at best by a relevant set of nodes. To our knowledge, the methodology proposed in the next sections is the first one based on this strategy.

3 Component-tree segmentation method

3.1 Component-trees

Let $E \subset \mathbb{Z}^n$ ($n \geq 1$) be a finite connected set (for a given adjacency relation). Let $V = [\![\bot, \top]\!] \cup \{-\infty\}$ with $[\![\bot, \top]\!] \subset \mathbb{Z}$. A grey-level image is a function $I : \mathbb{Z}^n \to V$ such that $I^{-1}(\{-\infty\}) = \mathbb{Z}^n \setminus E$. By abuse of notation, we will implicitly restrict I to E, and note $I : E \to V$, or $I \in V^E$. We assume, without loss of generality, that $\top = \max\{I(x) \mid x \in E\}$.

The thresholding function $X_v : V^E \to 2^E$ ($v \in V$) is defined by $X_v(I) = \{x \in E \mid v \leq I(x)\}$. The cylinder function $C_{X,v} : E \to V$ ($v \in V$, $X \subseteq E$) is defined by $C_{X,v}(x) = v$ if $x \in X$ and $-\infty$ otherwise. An image $I \in V^E$ can be written as $I = \bigvee_{v \in V} \bigvee_{X \in \mathcal{C}[X_v(I)]} C_{X,v}$ where $\bigvee$ is the pointwise supremum for the sets of functions, and $\mathcal{C}[\,.\,]$ is the set of the connected components of a set.

Let $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(I)]$ be the set of the connected components generated by the thresholdings of I at all values $v \in V$. The component-tree of I is obtained from the Hasse diagram of the partially ordered set $(\mathcal{K}, \subseteq)$.

Definition 1 (Component-tree) *Let $I \in V^E$ be a grey-level image. The* component-tree *of I is the rooted tree $T = (\mathcal{K}, L, R)$ such that:*
(i) $\mathcal{K} = \bigcup_{v \in V} \mathcal{C}[X_v(I)]$;
(ii) $L = \{(X, Y) \in \mathcal{K}^2 \mid Y \subset X \wedge \forall Z \in \mathcal{K}, Y \subseteq Z \subset X \Rightarrow Y = Z\}$;
(iii) $R = \sup(\mathcal{K}, \subseteq) = X_\perp(I) = E$.
The elements of $\mathcal{K}$ (resp. of L) are the nodes *(resp. the* edges*) of T. The node R is the* root *of T. For any $N \in \mathcal{K}$, we set $ch(N) = \{N' \in \mathcal{K} \mid (N, N') \in L\}$; $ch(N)$ is the set of the* children *of N. If $ch(N) = \emptyset$, we say that N is a* leaf.

3.2 Problem to solve

Component-trees can be used to develop segmentation procedures which consist of determining a subset $\widehat{\mathcal{K}} \subseteq \mathcal{K}$ among the nodes of the component-tree $T = (\mathcal{K}, L, R)$ of an image $I : E \to V$. The binary result $I_s \subseteq E$ is then defined as $I_s = \bigcup_{X \in \widehat{\mathcal{K}}} X$. A way to consider this segmentation problem is to search the set of nodes $\widehat{\mathcal{K}} \subseteq \mathcal{K}$ which enables to generate a binary object being as similar as possible to a target (*e.g.*, a manual presegmentation). This issue can be formalised as the resolution of the following optimisation problem

$$\widehat{\mathcal{K}} = \arg \min_{\mathcal{K}' \subseteq \mathcal{K}} \big\{ d\big(\bigcup_{N \in \mathcal{K}'} N, M \big) \big\} \tag{1}$$

where $M \subseteq E$ is the (binary) target, and d is a (pseudo-)distance on 2^E. An intuitive solution for determining such a useful pseudo-distance is to consider the amount of false positives/negatives induced by $X = \bigcup_{N \in \mathcal{K}'} N$ w.r.t. M

$$d^\alpha(X, M) = \alpha.|X \setminus M| + (1 - \alpha).|M \setminus X| \text{ with } \alpha \in [0, 1].$$

3.3 Segmentation method

As the set $\mathcal{K}$ is finite, there exists a solution to Eq. (1). The function $\mathcal{F}^\alpha$ proposed in Def. 2 enables to build a binary image whose connected components form a set $\widehat{\mathcal{K}}$ which is a solution of Eq. (1) (see Prop. 3).

Definition 2 ([4]) *Let $\alpha \in [0, 1]$. Let $I \in V^E$. Let $T = (\mathcal{K}, L, R)$ be the component-tree of I. Let $M \subseteq E$. Let $\prec \in \{<, \leq\}$. Let $\mathcal{F}^\alpha : \mathcal{K} \to 2^{\mathcal{K}}$ and $c^\alpha : \mathcal{K} \to \mathbb{R}^+$ be the functions recursively cross-defined, for all $N \in \mathcal{K}$, by*

$$\begin{cases} \mathcal{F}^\alpha(N) = \{N\} \\ c^\alpha(N) = \alpha.n(N, M) \end{cases}$$

if

$$\alpha.n(N, M) \prec (1 - \alpha).p^*(N, M) + \sum_{N' \in ch(N)} c^\alpha(N') \tag{2}$$

and

$$\begin{cases} \mathcal{F}^\alpha(N) = \bigcup_{N' \in ch(N)} \mathcal{F}^\alpha(N') \\ c^\alpha(N) = (1-\alpha).p^*(N,M) + \sum_{N' \in ch(N)} c^\alpha(N') \end{cases}$$

otherwise, where $p^*(N,M) = |(N \setminus \bigcup_{N' \in ch(N)} N') \cap M|$, *and* $n(N,M) = |N \setminus M|$.

Proposition 3 ([4]) *We set* $M^\alpha = \bigcup_{N \in \mathcal{F}^\alpha(E)} N$. *Then, we have*

$$d^\alpha(M^\alpha, M) = c^\alpha(E) = \min_{\mathcal{K}' \subseteq \mathcal{K}} \left\{ d^\alpha\Big(\bigcup_{N \in \mathcal{K}'} N, M \Big) \right\} .$$

Proposition 4 ([4]) $\mathcal{F}^\alpha(E) = \mathcal{C}[M^\alpha]$ *(and thus* M^α*) can be computed with the linear algorithmic complexity* $\mathcal{O}(\max\{|\mathcal{K}|, |E|\})$.

4 Component-hypertree segmentation method

4.1 Mask-based connectivity

In $\mathbb{Z}^n$, connectivity is generally handled thanks to standard notions such as $2n$- and (3^n-1)-adjacencies [14]. A (morphological) alternative definition has been proposed with the notion of second-generation connectivity [15]. In this context, mask-based connectivity [6] proposes to use some masks in order to characterise connected sets. In the binary case, by only considering masks which are either subsets or supersets of an image, we derive from [6] the following definition.

Definition 5 (Mask-based connectivity) *Let* $X \subseteq E$. *Let* $\omega(X) \subseteq X$ *(or* $\supseteq X$*) be a mask of* X. *The* ω*-connected components of* X, *noted* $\mathcal{C}_\omega[X]$, *are*
- *the connected components of* $\omega(X)$*; and*
- *the singleton sets* $\{x\}$ *for any* $x \in X \setminus \omega(X)$.

if $\omega(X) \subseteq X$ *and*
- *the sets* $X \cap Y$, *for any connected component* Y *of* $\omega(X)$*;*

if $\omega(X) \supseteq X$.

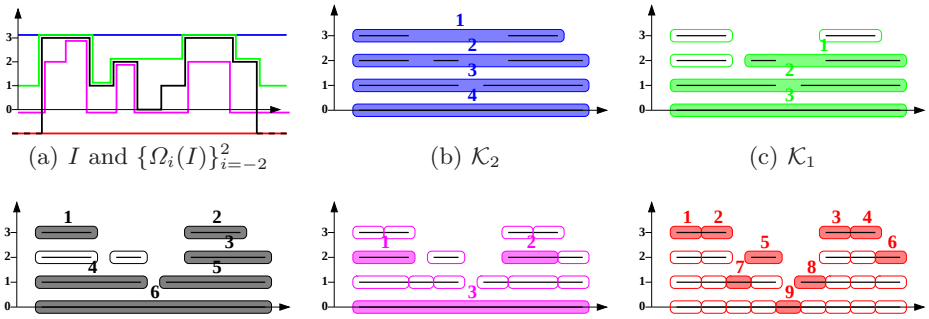
In the sequel, for a given image $I : E \to V$, we consider extensive (resp. antiextensive) masks $\Omega_\star(I) : E \to V$. We call $\Omega_\star$-connected components of I, and we note $\mathcal{K}_\star$ the set of all the $\omega_\star$-connected components of $X_v(I)$ induced by the masks $\omega_\star(X_v(I)) = X_v(\Omega_\star(I))$, at all values $v \in V$.

We consider, in particular, the families of masks $\{\Omega_i(I)\}_{i=t}^u$ $(t \leq 0 \leq u)$ such that (i) $\Omega_t(I) = C_{E,-\infty}$, (iii) $\Omega_0(I) = I$, (ii) $\Omega_u(I) = C_{E,\top}$, and (iv) $\Omega_i(I) < \Omega_j(I)$ for any $t \leq i < j \leq u$. An example of such a family $\{\Omega_i(I)\}_{i=-2}^2$ is depicted for a 1-D image, in Fig. 1.

Typical examples of families of masks verifying these properties are those induced by erosions/dilations (resp. openings/closings) (with a structuring element containing $0_{\mathbb{Z}^n}$), *e.g.*:

$$\ldots < \delta^k \circ \epsilon^k(I) < \ldots < \delta \circ \epsilon(I) < I < \epsilon \circ \delta(I) < \ldots < \epsilon^k \circ \delta^k(I) < \ldots$$

We can, of course, build the ($\Omega_\star$-)component-tree of I induced by the $\Omega_\star$-connected components of the successive level-sets of I. In particular, it can be observed that any node $N \in \mathcal{K}_i$ of the Ω_i-component-tree of I is partitioned into (one or several) node(s) of the Ω_{i-1}-component-tree of I.



(a) I and $\{\Omega_i(I)\}_{i=-2}^{2}$ (b) $\mathcal{K}_2$ (c) $\mathcal{K}_1$

(d) $\mathcal{K}_0 = \mathcal{K}$ (e) $\mathcal{K}_{-1}$ (f) $\mathcal{K}_{-2}$

Fig. 1. A 1-D image $I : \mathbb{Z} \to [\![0, 3]\!] \cup \{-\infty\}$, and a family $\{\Omega_i\}_{i=-2}^{2}$ of mask images of I. (a) I (in black); $\Omega_2(I) = C_{E,2}$ (in blue); $\Omega_1(I)$ (in green); $\Omega_0(I) = I$ (in black); $\Omega_{-1}(I)$ (in magenta); $\Omega_{-2}(I) = C_{E,-\infty}$ (in red). (b–f) Black lines: connected components of I; black lines inside (filled and unfilled) colour boxes: $\Omega_\star$-connected components of I (from b to f: Ω_2- to Ω_{-2}-connected components).

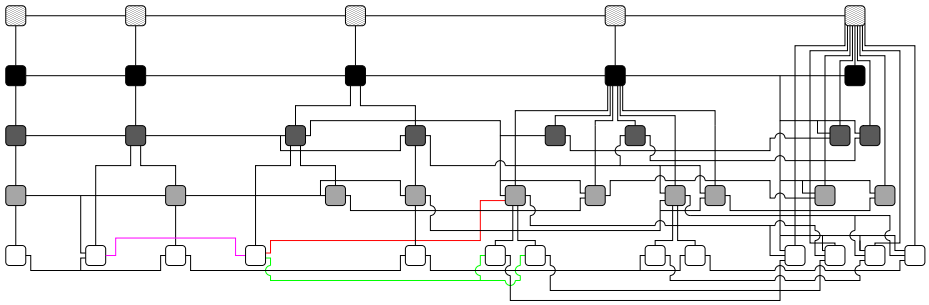


Fig. 2. Component-hypertree related to Fig. 1. The set of nodes $\mathcal{H}$ is depicted by the square boxes (their colour represent the threshold value at which they appear: from black (0) to white (3), and dashed for $-\infty$). The thick lines represent the edges of $L_\downarrow$, while the thin lines represent the edges of $L_\rightarrow$. From left to right, we can observe the five Ω_i-component-trees for i from 2 to -2.

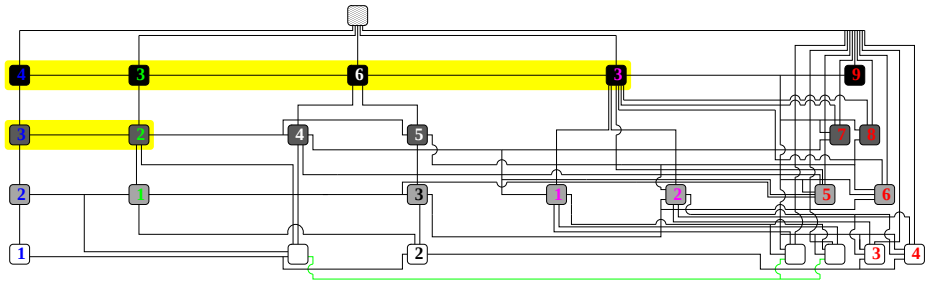


Fig. 3. Simplified component-hypertree (obtained from Fig. 2) related to Fig. 1. The numbers in nodes refer to the associated $\Omega_\star$-connected components in Fig. 1.

4.2 Component-hypertrees

We consider the definitions of Sec. 3.1, with the following modifications. An image I is now considered as defined from $\mathbb{Z}^n$ to V (but still with $E = I^{-1}(V \setminus \{-\infty\})$ connected and finite). We also associate to I a mask image $\Omega_\star(I) : \mathbb{Z}^n \to V$ satisfying the hypotheses of Sec. 4.1, *i.e.*, verifying either $\Omega_\star(I) \leqslant I$ or $\Omega_\star(I) \geqslant I$. Under such conditions, the $\Omega_\star$-connected components of I enable to generate an $\Omega_\star$-component-tree of I similar to the one described in Def. 1, with the following differences: (i) $\mathbb{Z}^n \in \mathcal{K}_\star$, (ii) $R = \sup(\mathcal{K}_\star, \subseteq) = X_{-\infty}(I) = \mathbb{Z}^n$. All the properties previously stated for component-trees remain, however, valid.

Definition 6 (Component-hypertree) *Let $I \in \mathbb{Z}^{n^V}$. Let $\{\Omega_i(I)\}_{i=t}^{u}$ ($t \leq 0 \leq u$) be a set of mask images of I. The* component-hypertree *of I is the triplet $H = (\mathcal{H}, L_\downarrow, L_\rightarrow)$ such that:*
(i) *$\mathcal{H}$ is the multiset*[1] *defined by: $\mathcal{H} = \bigcup_{i=t}^{u} \mathcal{K}_i$*
(ii) *$\forall i \in [\![t, u]\!]$, the subgraph of $(\mathcal{H}, L_\downarrow)$ induced by the subset of nodes $\mathcal{K}_i \subseteq \mathcal{H}$ is the Ω_i-component-tree of I;*
(iii) *$\forall v \in V$, the subgraph of $(\mathcal{H}, L_\rightarrow)$ induced by the subset of nodes $\mathcal{S}_v = \bigcup_{i \in [\![t,u]\!]} \mathcal{C}_{\omega_i}[X_v(I)] \subseteq \mathcal{H}$ is the Hasse diagram of the partially ordered (multi)set*[2] *$(\mathcal{S}_v, \subseteq)$.*

The component-hypertree related to Fig. 1 is illustrated in Fig. 2.

Remark 7 *Since $\Omega_t(I) = C_{E,-\infty}$, we have $\mathcal{K}_t = \{\mathbb{Z}^n\} \cup \{\{x\} \mid x \in E\}$,* i.e.*, the Ω_t-component-tree is composed of a root $\mathbb{Z}^n$ and $|E|$ leaves corresponding each one to a point in E (see right part of Fig. 2). Since $\Omega_u(I) = C_{E,\top}$, we have $\mathcal{K}_u = \{X_v(I)\}_{v \in V}$,* i.e.*, the Ω_u-component-tree is composed of the $|V|$ binary images corresponding to the successive thresholdings of I (see left part of Fig. 2).*

Remark 8 *A node $N \in \mathcal{K}_i$ can have several (distinct) decompositions in $\mathcal{K}_{i-1}$ (this may happen since $\mathcal{K}_i$, is* not *a multiset). In particular, a node N which is a Ω_i-connected component of $X_v(I)$ for all $v \in [\![v^-, v^+]\!] \subseteq V$ can have up to $v^+ - v^- + 1$ decompositions in $\mathcal{K}_{i-1}$, at the same grey-levels. The nodes of these decompositions form a set of (sub)trees of the Ω_{i-1}-component-tree of I (see Fig. 2 for a node with two decompositions (red and green edges), and a node with the same decomposition at values* 2 *and* 3 *(magenta edge)).*

4.3 Simplification

Performing segmentation in the way of Sec. 3 consists of solving Eq. (1), *i.e.*, of computing $\mathcal{F}^\alpha(\mathbb{Z}^n)$, for all the $\Omega_\star$-component-trees (in order to allow the user to define, by "connectivity tuning", the most satisfactory result). Instead of processing these component-trees independently, we can compute $\mathcal{F}^\alpha$ in a global fashion in the component-hypertree (in particular, to avoid redundant work).

[1] Several nodes of $\mathcal{H}$ which are the same subset of $\mathbb{Z}^n$ can possibly refer to $\Omega_\star$-connected components of distinct $\Omega_\star$-component-trees.
[2] We set $X \subset Y$ whenever $X \in \mathcal{K}_i$, $Y \in \mathcal{K}_j$ are the same subset of $\mathbb{Z}^n$, with $i < j$.

Based on this purpose, let us first state some remarks enabling to simplify (*i.e.*, factorise) a part of the component-hypertree data structure.

Remark 9 *The root $\mathbb{Z}^n$ (which role is mainly to guarantee that each $\Omega_\star$-component-tree is actually a tree) only needs to be represented once*[3]*.*

Remark 10 *Let $N_i \in \mathcal{K}_i$, $N_{i-1} \in \mathcal{K}_{i-1}$ be the same subset of $\mathbb{Z}^n$, and the subtrees of the Ω_i- and Ω_{i-1}-component-trees of root N_i and N_{i-1}, be identical. From Def. 2, we have $\mathcal{F}^\alpha(N_i) = \mathcal{F}^\alpha(N_{i-1})$. Then, several successive $\Omega_\star$-component-trees can share there similar "bottom" parts, thus reducing both space and time complexity. In particular, this is true for the $|E|$ singleton sets $\{x\}$ ($x \in E$) which need only to be represented once (for instance in the Ω_t-component-tree).*

By opposition, a node which appears in two successive $\Omega_\star$-component-trees must be represented twice if its subtrees differ (which may modify their value of c^α, and thus $\mathcal{F}^\alpha$, see yellow parts of Fig. 3 for examples of such nodes).

Remark 11 *As stated in Rem. 8, a node may have several decompositions. However, from an algorithmic point of view, only one decomposition is necessary. We should, in particular, preserve the decomposition which corresponds to the finer subdivision of the node,* i.e.*, the one composed by the nodes at the threshold value v^+ defined in Rem. 8 (see the green edge in Fig. 3, for an example).*

From these remarks, a component-hypertree can be simplified to contain only nodes which are computationally useful (see Figs. 2 and 3).

Note that a simplified component-hypertree preserves its principal two specificities: (*i*) the subgraph of $(\mathcal{H}, \mathcal{L}_\downarrow)$ induced by the subset of nodes $\mathcal{K}_\star$ is the $\Omega_\star$-component-tree, and (*ii*) the subgraph of $(\mathcal{H}, \mathcal{L}_\rightarrow)$ induced by the subset of nodes $\mathcal{S}_v$ corresponding to the $\Omega_\star$-connected components obtained at the threshold value v provides a (deterministic) hierarchic decomposition of $X_v(I)$ into Ω_i-connected components (for decreasing values of i).

4.4 Segmentation

The main purpose is now to compute $\mathcal{F}^\alpha(\mathbb{Z}^n)$ for all the $\Omega_\star$-component-trees of the component-hypertree. (Note that we restrict, in the sequel, our study to $\alpha \in\ ]0,1[$, since the cases $\alpha = 0$ or 1 can be treated in a more simple way.)

For any $N \in \mathcal{H}$, we define the following notions. If $\mathcal{F}^\alpha(N) = \{N\}$ (resp. $\mathcal{F}^\alpha(N) \neq \{N\}$), we say that N is *on* (resp. *off*). Let $d(N)$ be the set of nodes D such that $(N, D) \in \mathcal{L}_\rightarrow$, *i.e.*, the nodes forming the coarsest partition of N in H. Let $d_\bullet(N) = \{D \in d(N) \mid \mathcal{F}^\alpha(D) = \{D\}\}$ and $d_\circ(N) = d(N) \setminus d_\bullet(N)$ be the subsets of $d(N)$ composed of the nodes which are on and off, respectively. We set

$$\begin{cases} \delta^+(N) = \alpha.n(N,M) - (1-\alpha).p^*(N,M) - \sum_{N' \in ch(N)} c^\alpha(N') \\ \delta^-(N) = c^\alpha(N) - \sum_{D \in d(N)} c^\alpha(D) \end{cases}$$

[3] From an algorithmic point of view, it needs not to be represented since we never have $\mathcal{F}^\alpha(\mathbb{Z}^n) = \{\mathbb{Z}^n\}$, except in the useless case $\alpha = 0$ and $\prec$ is $\leq$ (when $\alpha = 0$, the case $\prec$ is $<$ should be considered, to get access to the smallest subset including M).

Property 12 *If $d(N) = \emptyset$ (*i.e.*, $N = \{x\}$ $(x \in E)$ is a singleton node), then we have $\mathcal{F}^\alpha(N) = \{N\}$ if $x \in M$ and $\emptyset$ if $x \notin M$ (see Def. 2). Moreover, we straightforwardly have $\delta^-(N) = 0$ while $\delta^+(N) = \alpha$ if $x \notin M$ and $\alpha - 1$ if $x \in M$.*

Property 13 *In the case where $d(N) \neq \emptyset$, it derives from the above definitions that Eq. (2) is equivalent to the following inequality*

$$\sum_{D \in d(N)} \delta^+(D) \prec \sum_{N' \in ch(N)} \delta^-(N') \tag{3}$$

Property 14 *Based on this property, in the case where $d(N) \neq \emptyset$, we have $\mathcal{F}^\alpha(N) = \{N\}$ (*i.e.*, N is on) if Eq. (3) is true and $\mathcal{F}^\alpha(N) = \bigcup_{N' \in ch(N)} \mathcal{F}^\alpha(N')$ (*i.e.*, N is off) if it is false. It can also be proved that we have $\delta^-(N) = \sum_{D \in d_\circ(N)} \delta^+(N)$ if N is on, and $\sum_{N' \in ch(N)} \delta^-(N) - \sum_{D \in d_\bullet(N)} \delta^+(N)$ if N is off. Finally, we also have $\delta^+(N) = \sum_{D \in d(N)} \delta^+(D) - \sum_{N' \in ch(N)} \delta^-(N')$.*

From Props. 12–14, we obtain an algorithmic process to recursively compute (in a "bottom-up/right-to-left" fashion) the sets $\mathcal{F}^\alpha(\mathbb{Z}^n)$ in the component-hypertree, by storing the values δ^-, δ^+ and performing (at most) one comparison and a few additions at each node.

4.5 Optimisation

As stated in Sec. 4.3, the simplification of the component-hypertree data structure provides a first way to decrease the cost of the computation of $\mathcal{F}^\alpha$. Some supplementary optimisations derive from the following properties. (These optimisations require a longer discussion which will be proposed in a further issue.)

Property 15 *When $\sum_{D \in d(N)} \delta^+(D) \prec 0$ (which happens* a fortiori*, but not necessarily, when all the nodes of d(N) are on), we have $\mathcal{F}^\alpha(N) = \{N\}$. Then, we can avoid to compute $\mathcal{F}^\alpha$, δ^- and δ^+ for the nodes $N' \in ch(N)$ and their respective subtrees. However it may (sometimes) be necessary to compute latter δ^- and δ^+ for some of these (temporarily) unprocessed nodes, due to the (possible) non-increasingness of M^α w.r.t. the increasingness of the mask images.*

When the segmentations have been computed for a given α, the computation of new segmentations, for another α, may require to process, once again (potentially) all the nodes of the component-hypertree. However, we have the following property (the proof of which will be displayed in a next issue).

Property 16 *Let $0 \leq \alpha_1 < \alpha_2 \leq 1$. Then we have $M^{\alpha_2} \subseteq M^{\alpha_1}$.*

Its main consequence is the existence, for each node N, of a critical value α_c such that N is on iff $\alpha \prec \alpha_c$. Consequently, when a node N is switched on (resp. off) during the processing of the component-hypertree, at a given value α, we know that α_c belongs to $[\alpha, 1]$ (resp. $[0, \alpha]$). At each new process, the bounding $[\alpha_c^-, \alpha_c^+] \subseteq [0, 1]$ of α_c at each node can then be refined. In particular, this may enable to avoid computation at several nodes, by only checking (in the favourable cases) whether $\alpha \prec \alpha_c^-$ or $\alpha \not\prec \alpha_c^+$.

5 Experiments

LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE - LES COMPONENT-HYPERTREES, ÇA FAIT MAL AU CRANE

6 Conclusion

A new data structure, the component-hypertree, has been presented. It models several component-trees of a same image, induced by second-order mask-based connectivity. An improved version of a segmentation method relying on component-trees has been proposed for this structure. It enables in particular to efficiently compute, with a low computational cost, the segmentation of an image for several connectivities. This approach, whose relevance has been experimentally assessed, will be described in a more detailed fashion in further works. Il encore reste de la place pour écrire des trucs formidables sur toute une ligne.

References

1. Salembier, P.: Connected operators based on tree pruning strategies. In Najman, L., Talbot, H., eds.: Mathematical morphology: from theory to applications. ISTE/J. Wiley & Sons (2010) 179–198
2. Jones, R.: Connected filtering and segmentation using component trees. Computer Vision and Image Understanding **75**(3) (1999) 215–228
3. Najman, L., Couprie, M.: Building the component tree in quasi-linear time. IEEE Transactions on Image Processing **15**(11) (2006) 3531–3539
4. Passat, N., Naegel, B.: Selection of relevant nodes from component-trees in linear time. In: DGCI. Volume xxxx of LNCS., Springer (2011) xxx–xxx
5. Urbach, E.R., Boersma, N.J., Wilkinson, M.H.F.: Vector attribute filters. In: ISMM. Volume 30 of Computational Imaging and Vision., Springer (2005) 95–104
6. Ouzounis, G.K., Wilkinson, M.H.F.: Mask-based second-generation connectivity and attribute filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(6) (2007) 990–1004
7. Wilkinson, M.H.F., Westenberg, M.A.: Shape preserving filament enhancement filtering. In: MICCAI. Volume 2208 of LNCS., Springer (2001) 770–777
8. Urbach, E.R., Wilkinson, M.H.F.: Shape-only granulometries and gray-scale shape filters. In: ISMM, CSIRO Publishing (2002) 305–314
9. Naegel, B., Passat, N., Boch, N., Kocher, M.: Segmentation using vector-attribute filters: methodology and application to dermatological imaging. In: ISMM. Volume 1., INPE (2007) 239–250
10. Berger, C., Géraud, T., Levillain, R., Widynski, N., Baillard, A., Bertin, E.: Effective component tree computation with application to pattern recognition in astronomical imaging. In: ICIP. (2007) 41–44
11. Naegel, B., Wendling, L.: Combining shape descriptors and component-tree for recognition of ancient graphical drop caps. In: VISAPP. Volume 2. (2009) 297–302
12. Urbach, E.R., Roerdink, J.B.T.M., Wilkinson, M.H.F.: Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(2) (2007) 272–285
13. Caldairou, B., Naegel, B., Passat, N.: Segmentation of complex images based on component-trees: Methodological tools. In: ISMM. Volume 5720 of LNCS., Springer (2009) 171–180
14. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. Computer Vision, Graphics, and Image Processing **48**(3) (1989) 357–393
15. Serra, J.: Connectivity on complete lattices. Journal of Mathematical Imaging and Vision **9**(3) (1998) 231–251