New algorithms for multivalued component-trees

Nicolas Passat, Romain Perrin, Jimmy Francky Randrianasoa, Camille Kurtz, Benoît Naegel

New Algorithms For Multivalued Component Trees*

Nicolas Passat[1], Romain Perrin[2], Jimmy Francky Randrianasoa[2,3], Camille Kurtz[4], and Benoît Naegel[2]

[1] Université de Reims Champagne Ardenne, CRESTIC, Reims, France
[2] Université de Strasbourg, CNRS, ICube, Strasbourg, France
[3] EPITA Research Laboratory (LRE), Le Kremlin-Bicêtre, France
[4] Université Paris Cité, LIPADE, Paris, France

Abstract. Tree-based structures can model images—and more generally valued graphs—for processing and analysis purpose. In this framework, the component tree was natively designed for grey-level images—and more generally totally ordered valued graphs. Ten years ago, the notion of a multivalued component tree was introduced to relax this grey-level / total order constraint. In this algorithmic paper, we provide new tools to handle multivalued component trees. Our contributions are twofold: (1) we propose a new algorithm for the construction of the multivalued component tree; (2) we propose two strategies for building hierarchical orders on value sets, required to further build the multivalued component trees of images / graphs relying on such value sets. Codes available at: https://github.com/bnaegel/multivalued_component_tree.

Keywords: algorithmics · images / valued graphs · multivalued component trees · hierarchical ordering · connected operators · mathematical morphology

1 Introduction

Building trees for modeling images is a historical research topic which was mainly investigated in field of mathematical morphology. The trees developed in this framework model in a compact way the space of the possible partitions of an image induced by its mixed spatial-spectral composition. These so-called morphological trees can be subdivided into two families, which build upon either total or partial partitions. The archetype of the first family is the binary partition tree [23] while the archetype of the second is the component tree [24].

Based on these trees, various image processing and analysis methods were developed, gathered under the name of connected operators [25,26]. The success of morphological trees and connected operators relies on their low cost in terms of construction and handling. Regarding their construction, they can be built in quasi-linear time [23,3]. Regarding their involvement in image processing / analysis tasks, the two main paradigms of attribute-based node selection [2,8] and optimal cut computation [6,9] can be carried out in linear time.

* This work was supported by the French *Agence Nationale de la Recherche* (grants ANR-20-THIA-0006, ANR-20-CE45-0011, ANR-22-CE45-0034 and ANR-23-CE45-0015).

Over the last years, efforts were geared towards enriching the framework of morphological hierarchies with new structures that generalize classical ones. In this context, the notion of multivalued component tree [10] was proposed ten years ago as a subfamily of the component graphs [17,16] which generalize the classical component tree [24]. This new paradigm of multivalued component tree had been designed in order to build a component tree on images where values are organised with respect to a (partial) hierarchical order relation, whereas the standard component tree requires a total order.

This is an algorithmic article. It provides new tools mandatory for handling the multivalued component tree. First, we propose a new approach for building the multivalued component tree (Sect. 4). By contrast with an initial algorithm proposed in [10] (which required some pre- and post-processings to rewrite multivalued component tree construction as component tree construction), we now provide a standalone algorithm that directly builds a multivalued component tree from its multivalued image. Second, we provide two strategies for endowing a set of values with hierarchical order relations, adapted to the further construction of multivalued component trees (Sect. 5). Both strategies rely on the construction of morphological trees—component trees or binary partition trees—on/from a set of values considered itself as an image—or a valued graph.

The other parts of this article are organized as follows. In Sect. 2, we recall related works on the morphological trees. In Sect. 3, we provide the required definitions and notions related to the (multivalued) component tree. Sect. 6 concludes the article.

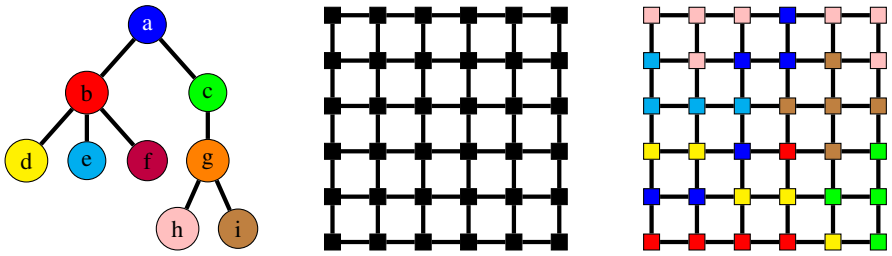
2 Related works

A morphological tree models an image defined as a function $\mathcal{F} : \Omega \to \mathbb{V}$ that associates to each point $\mathbf{x}$ of its support Ω a value $\mathcal{F}(\mathbf{x})$ within the set of values $\mathbb{V}$. In general, Ω is endowed with an adjacency relation $\frown$. In other words, $(\Omega, \frown)$ is a non-directed graph. By side effect, a morphological tree can model valued graphs, and not only images.

Morphological trees are partition trees. Indeed, they are created by stacking a finite sequence of partitions of Ω. Each partition is composed of subsets $X \subseteq \Omega$ that constitute the nodes (root, internal nodes, leaves) of the tree, and a tree models the inclusion relation between them. Such trees can be classified in two main families: those originated either from (1) total partitions or (2) partial partitions of Ω.

The archetype of the total partition trees is the binary partition tree [23] (also declined under variants: α-tree [27], watershed tree [12], etc.). Except the leaves, each node is a connected subset $X \subseteq \Omega$ which has two children nodes X_1 and X_2 that form a partition of X, leading to a top-down binary decomposition of the root Ω of the tree into subsets of decreasing size. The construction of such trees is guided by one or many [20] criteria which determine the merging order of the smallest subsets provided by an initial partition of Ω, that defines the leaves of the tree.

The archetype of the partial partition trees is the component tree [24] (also declined under variants: hyperconnection tree [18], tree of shapes [11], topological tree of shapes [14], complete tree of shapes [15], etc.). The component tree is built from successive threshold sets, at each value of $\mathbb{V}$ of the image $\mathcal{F}$. The component tree models the inclusion relation between the connected components of these threshold sets. Each node



(a) $(\mathbb{V}, \leqslant)$ (b) $\mathfrak{G} = (\Omega, \frown)$ (c) $\mathcal{F} : \Omega \to \mathbb{V}$

Fig. 1. (a) A set of values $\mathbb{V}$ = {a,b,c,d,e,f,g,h,i}, endowed with a hierarchical order $\leqslant$ such that the minimum is a and the maximal elements are d, e, f, h, i. This ordered set is depicted here as its Hasse diagram, which is—by definition—a tree. (b) A set $\Omega = [\![0, 5]\!]^2$ (squares) endowed with an adjacency $\frown$ (segments), leading to the graph $\mathfrak{G} = (\Omega, \frown)$. (c) A multivalued image $\mathcal{F} : \Omega \to \mathbb{V}$ built on the support Ω (b) and taking its values in $\mathbb{V}$ (a). The colour of each square is associated to the value of the corresponding point.

is a subset $X \subseteq \Omega$ corresponding to a connected component at a given value $v \in \mathbb{V}$. If X is not a flat zone of the image, it has $k \geq 1$ children nodes X_i $(1 \leq i \leq k)$ that form a partition of a strict subset $Y \subset X$, which corresponds to the part of X where the values of $\mathcal{F}$ are strictly greater than v.

Many efforts were dedicated to the efficient construction of morphological trees, and especially the component tree. An overview of the classical algorithms, based e.g. on flooding or union-find paradigms, can be found in [3]. A recent trend is also to develop parallel algorithms, based on distributed paradigms [5,4] or GPU-based approaches [1].

3 Multivalued component tree

Let Ω be a finite set and $\frown$ an adjacency (irreflexive, symmetric) relation on Ω that induces the (equivalence) connectedness relation by reflexive-transitive closure of $\frown$. The couple $\mathfrak{G} = (\Omega, \frown)$ is a non-directed graph. For any subset $X \subseteq \Omega$, we note $C[X]$ the set of the connected components (i.e. the maximal connected sets) of the subgraph $(X, \frown)$ of $\mathfrak{G}$ induced by X. We assume that $\mathfrak{G}$ is connected, i.e. $C[\Omega] = \{\Omega\}$. See Fig. 1(b).

Let $\mathbb{V}$ be a finite set and $\leqslant$ a hierarchical order on $\mathbb{V}$, i.e. an order (1) which admits a minimum (resp. a maximum) and (2) such that for any $v \in \mathbb{V}$, the subset of the elements lower (resp. greater) than v is totally ordered by $\leqslant$. See Fig. 1(a).

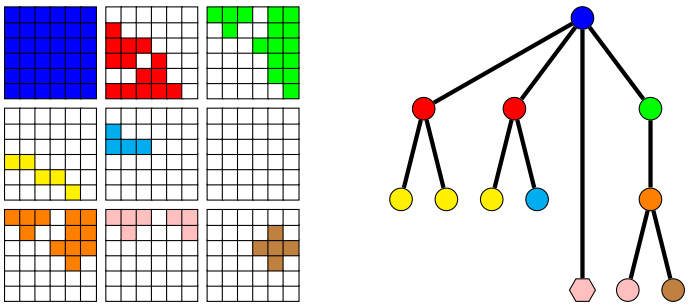
A total order is a hierarchical order. Thus, all the definitions given below for the multivalued component tree [10] generalize those of the classical component tree [24].

Let us consider an image $\mathcal{F}$ defined as a function $\mathcal{F} : \Omega \to \mathbb{V}$. See Fig. 1(c). The threshold set of $\mathcal{F}$ at value $v \in \mathbb{V}$ is defined by

$$\Lambda_v(\mathcal{F}) = \{\mathbf{x} \in \Omega \mid v \leqslant \mathcal{F}(\mathbf{x})\} \tag{1}$$

See Fig. 2(a). We set

$$\Theta = \bigcup_{v \in \mathbb{V}} C[\Lambda_v(\mathcal{F})] \tag{2}$$



(a) $\Lambda_v(\mathcal{F})$ for $v \in \mathbb{V}$ (b) $\mathfrak{T} = (\Theta, \lhd)$

Fig. 2. (a) The nine threshold sets $\Lambda_v(\mathcal{F})$ for the image $\mathcal{F}$ of Fig. 1(c). The squares depicted in color (resp. white) belong (resp. do not belong) to $\Lambda_v(\mathcal{F})$. Note that $\Lambda_{\mathrm{f}}(\mathcal{F}) = \emptyset$ since the image $\mathcal{F}$ has no point of value f. Also note that $\Lambda_{\mathrm{g}}(\mathcal{F}) \neq \emptyset$ whereas $\mathcal{F}$ has no point of value g. This is justified by the fact that $\Lambda_{\mathrm{g}}(\mathcal{F})$ is partitioned by $\Lambda_{\mathrm{h}}(\mathcal{F})$ and $\Lambda_{\mathrm{i}}(\mathcal{F})$. A connected component ("T-shaped", in the upper-left part of the image) is common to the threshold sets $\Lambda_{\mathrm{c}}(\mathcal{F})$, $\Lambda_{\mathrm{g}}(\mathcal{F})$ and $\Lambda_{\mathrm{h}}(\mathcal{F})$. (b) The multivalued component tree $\mathfrak{T} = (\Theta, \lhd)$ of the image $\mathcal{F}$ of Fig. 1(c). Each disk / hexagon corresponds to a node $X \subseteq \Omega$ of Θ (the unique hexagonal node X corresponds to the three occurrences of the "T-shaped" connected component). The color of the disk / hexagon corresponds to the value $\omega(X)$ of the node.

which gathers the connected components at each threshold set $\Lambda_v(\mathcal{F})$ ($v \in \mathbb{V}$). The elements of Θ are called the nodes of the multivalued component tree.

A node may be generated at many threshold values (see the "T-shaped" connected component in Fig. 2(a)). In particular, for any $X \in \Theta$, we set

$$\mathbb{I}(X) = \{v \in \mathbb{V} \mid X \in C[\Lambda_v(\mathcal{F})]\} \tag{3}$$

and we define the remanence $\tau(X)$ of X as the number of threshold sets to which X belongs, i.e. as

$$\tau(X) = |\mathbb{I}(X)| \tag{4}$$

We also set $\omega(X)$ as the maximal value of threshold sets to which X belongs, i.e. as

$$\omega(X) = \bigvee^{\leqslant} \mathbb{I}(X) \tag{5}$$

For instance, for the "T-shaped" connected component X in Fig. 2, which belongs to $\Lambda_{\mathrm{c}}(\mathcal{F})$, $\Lambda_{\mathrm{g}}(\mathcal{F})$ and $\Lambda_{\mathrm{h}}(\mathcal{F})$, we have $\mathbb{I}(X) = \{\mathrm{c},\mathrm{g},\mathrm{h}\}$, the remanence of X is $\tau(X) = 3$ and we have have $\omega(X) = \mathrm{h}$, since $\mathrm{c} \leqslant \mathrm{g} \leqslant \mathrm{h}$.

The inclusion relation $\subseteq$ is a hierarchical order on Θ. We note $\lhd$ the reflexive-transitive reduction of $\subseteq$ with respect to Θ. The couple $\mathfrak{T} = (\Theta, \lhd)$, i.e. the Hasse diagram of $(\Theta, \subseteq)$, is a tree called the multivalued component tree. See Fig. 2(b).

For any node $X \in \Theta$, we define the proper part $\rho(X) \subseteq \Omega$ of X as

$$\rho(X) = X \setminus \bigcup_{Y \lhd X} Y = \{\mathbf{x} \in \Omega \mid \mathcal{F}(\mathbf{x}) = \omega(X)\} \tag{6}$$

The multivalued component tree $\mathfrak{T}$ is an image model of the image $\mathcal{F}$. Indeed, we can reconstruct $\mathcal{F}$ from $\mathfrak{T}$ as follows

$$\forall \mathbf{x} \in \Omega, \mathcal{F}(\mathbf{x}) = \bigvee_{X \in \Theta}^{\leqslant} \mathbf{1}_{(X,\omega(X))}(\mathbf{x}) \tag{7}$$

where $\mathbf{1}_{(A,u)} : \Omega \to \mathbb{V}$ is the cylinder function defined by $\mathbf{1}_{(A,u)}(\mathbf{x}) = u$ if $\mathbf{x} \in A \subseteq \Omega$ and $\bigwedge^{\leqslant} \mathbb{V}$ (the minimum of $(\mathbb{V}, \leqslant)$) otherwise.

These definitions given for the multivalued component tree are similar to those of the standard component tree. The only differences are the following:

- $\leqslant$ is a hierarchical order whereas it is a total order for the component tree;
- it may happen that $\rho(X) = \emptyset$ whereas we have $\rho(X) \neq \emptyset$ for the component tree.

4 Building the multivalued component tree

4.1 Some reminders of the previous algorithm

In [10], a first strategy had been proposed for building the multivalued component tree. The main idea was to rewrite the image $\mathcal{F} : \Omega \to \mathbb{V}$ as an image $\widehat{\mathcal{F}} : \widehat{\Omega} \to \widehat{\mathbb{V}}$ where $\widehat{\Omega} \supseteq \Omega$ and $\widehat{\mathbb{V}} = [\![0, p]\!] \subset \mathbb{N}$ with $p \leq |\mathbb{V}|$. The set $\widehat{\mathbb{V}}$ was endowed with the total order $\leq$ on $\mathbb{N}$ such that there is a homomorphism from $(\mathbb{V}, \leqslant)$ to $(\widehat{\mathbb{V}}, \leq)$ induced by the equivalence relation on $\mathbb{V}$ that gathers the values of equal distance with respect to the minimum $\bigwedge^{\leqslant} \mathbb{V}$ in the Hasse diagram of $(\mathbb{V}, \leqslant)$. The set $\widehat{\Omega}$ was endowed with an adjacency $\frown_{\widehat{\Omega}}$ such that there is an increasing homeomorphism from the graph $(\Omega, \frown)$ to the graph $(\widehat{\Omega}, \frown_{\widehat{\Omega}})$. The latter can be defined by adding a new vertex $\varepsilon_{\{x,y\}}$ in $\widehat{\Omega}$, and replacing the adjacency link $x \frown y$ by the two links $x \frown \varepsilon_{\{x,y\}}$ and $\varepsilon_{\{x,y\}} \frown y$, whenever the two vertices $x, y \in \Omega$ are such that $x \frown y$ while $\mathcal{F}(x)$ and $\mathcal{F}(y)$ are non-comparable with respect to $\leqslant$. It was proved that the component tree $\widehat{\mathfrak{T}}$ of $\widehat{\mathcal{F}}$ is isomorphic with the multivalued component tree $\mathfrak{T}$ of $\mathcal{F}$. This was then possible to build a multivalued component tree by using any algorithm dedicated to the construction of the component tree, at the cost of (1) the preprocessing that builds $\widehat{\mathcal{F}}$ from $\mathcal{F}$, and (2) a post-processing that retrieves $\mathfrak{T} = (\Theta, \lhd)$ from $\widehat{\mathfrak{T}} = (\widehat{\Theta}, \widehat{\lhd})$ by removing from the proper part $\rho(X)$ of each node $X \in \widehat{\Theta}$ the elements of $X \setminus \Omega$ and by substituting the values of $\mathbb{V}$ to those of $\widehat{\mathbb{V}}$ in the definition of $\omega(X)$.

4.2 A new algorithm

We now present a new alternative algorithm that no longer requires such pre-conditioning of the image $\mathcal{F}$. The construction scheme is detailed in Alg. 1 and Func. Flood. The proposed strategy is derived from the component tree construction presented by Salembier et al. in [24]. It also finds inspiration in the mask-based algorithm developed by Ouzounis et al. in [13].

The proposed algorithm relies on the following data structures:

Algorithm 1: Building the multivalued component tree

Input: $(\Omega, \frown), (\mathbb{V}, \leqslant), \mathcal{F} : \Omega \to \mathbb{V}$
Output: $\mathfrak{T} = (\Theta, \lhd)$

1 Build `nodes`
2 Build `points`
3 Build `status`
4 Build `nb_nodes`
5 Build `index`
6 Build `progress`
7 $v_{\min} := \bigwedge^{\leqslant} \mathbb{V}$
8 Choose $\mathbf{x}_{\min} \in \Omega$ such that $\mathcal{F}(\mathbf{x}_{\min}) = v_{\min}$
9 `points`$[v_{\min}]$`.add(`$\mathbf{x}_{\min}$`)`
10 `progress`$[v_{\min}]$:= *true*
11 Flood$(v_{\min})$

- `nodes`: a 2D array which stores the nodes of the multivalued component tree. The first dimension is indexed by the values of $\mathbb{V}$. The second dimension is indexed by the identifiers of the nodes. In other words, `nodes` encodes Θ; `nodes`$[v]$ encodes the nodes of Θ at value v; and `nodes`$[v][i]$ encodes the ith node of Θ at value v;
- `points`: a 2D array which stores the processed points of the image. The first dimension is indexed by the values of $\mathbb{V}$. In other words, `points`$[v]$ encodes all the points $\mathbf{x} \in \Omega$ currently processed "at value v";
- `status`: a 1D array which stores the status of each point of the image. For any point $\mathbf{x} \in \Omega$, we have `status`$[\mathbf{x}] = -1$ if $\mathbf{x}$ is unprocessed; `status`$[\mathbf{x}] = 0$ if $\mathbf{x}$ belongs to `points`; and `status`$[\mathbf{x}] = i > 0$ if $\mathbf{x}$ belongs to the proper part $\rho(X)$ of the node X stored in `nodes`$[\mathcal{F}(\mathbf{x})][i]$;
- `nb_nodes` and `index`: two 1D arrays which store the number of nodes already fully built and the index of the node currently built at each value of $\mathbb{V}$, respectively;
- `progress`: a 1D array which indicates if there exists a node at value v, currently under construction or to be built, which is an ancestor of the node at value u currently being defined.

By comparison with the component tree construction detailed in [24], the one proposed here for multivalued component tree construction differs with regard to Flood as follows:

- In [24], we have $\mathbf{x} \in$ `nodes`$[\mathcal{F}(\mathbf{x})]$. Here (Lines 7–11), we may have $\mathbf{x} \in$ `nodes`$[u]$ with $u \neq \mathcal{F}(\mathbf{x})$. This happens when $\mathbf{x}$ is stored in `nodes` as the neighbour of another point with a non-comparable value. In such case, the chosen value u is the infimum of these two non-comparable values, and we have in particular $u < \mathcal{F}(\mathbf{x})$;
- For two adjacent points $\mathbf{x} \frown \mathbf{y}$, it may occur that $\mathcal{F}(\mathbf{x})$ and $\mathcal{F}(\mathbf{y})$ be non-comparable. In particular, the "else" case at Line 19 means that either $u > w$ or u and w are non-comparable. In the first case, $\widehat{w}$ is set to w. In the second case, $\widehat{w}$ is the infimum of u and w, distinct from them. In this last case, the point $\mathbf{y}$ of value w is added to `nodes`$[\widehat{w}]$ and not to `nodes`$[w]$. This will further result in the scenario discussed above (Lines 7–11).

Function Flood

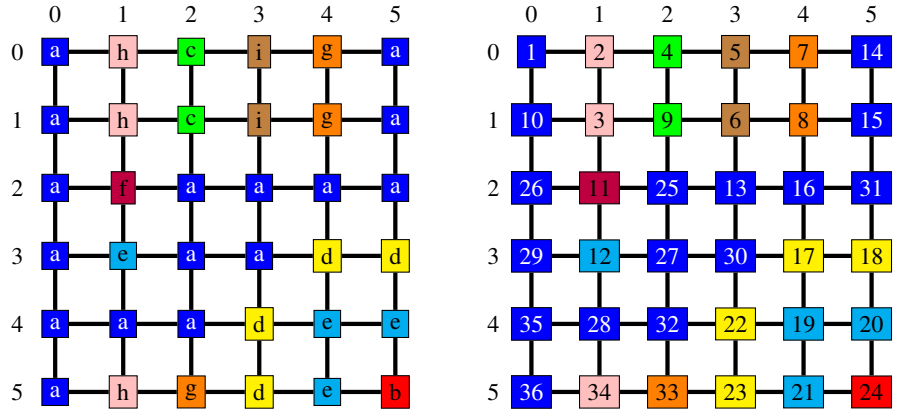
Input: $u \in \mathbb{V}$: current level

Output: $w \in \mathbb{V}$: value of the parent node of the root of the built (partial) multivalued component tree at value u (or ε if the node has no parent)

```
1  while !(points[u].empty()) do
2      x := points[u].remove()
3      if index[u] > nb_nodes[u] then
4          nb_nodes[u] := index[u]                    // in practice, nb_nodes[u]++
5          X := create_node()                         // new node in Θ
6          nodes[u].insert(X)
7      if F(x) ≠ u then
8          w := F(x)
9          points[w].add(x)
10         progress[w] := true
11         while u < w do w := Flood(w)
12     else
13         status[x] := index[u]
14         nodes[u][index[u]].add_to_proper_part(x)
15         foreach y ⌢ x do
16             w := F(y)
17             if status[y] = −1 then
18                 if u ⩽ w then ŵ := w
19                 else ŵ := ⋀^≤{u, w}
20                 points[ŵ].add(y)
21                 status[y] := 0
22                 progress[ŵ] := true
23                 while u < ŵ do ŵ := Flood(ŵ)
24 if u = v_min then
25     w := ε
26 else
27     w := ⋁^≤{w' ∈ 𝕍 | w' < u}
28     while progress[w] = false do w := ⋁^≤{w' ∈ 𝕍 | w' < w}
29     create_edge(nodes[u][index[u]],nodes[w][index[w]])   // new edge in ◁
30 progress[u] = false
31 index[u]++
32 return w
```

Note that `nodes`, `points`, `status`, `nb_nodes`, `index`, `progress`, are handled as global variables. In practice, Flood is then called for an input value v, *with a given configuration of these variables* and modifies them.

An example of the behaviour of the algorithm is provided in Figs. 3–4. For the image $\mathcal{F} : \Omega \to \mathbb{V}$ of Fig. 3(a), the processing order of the points of Ω is given in Fig. 3(b), from the first (1) to the last one (36). Fig. 4 shows the progress of the construction of the multivalued component tree of $\mathcal{F}$ with respect to the processed points.



(a) $\mathcal{F} : \Omega \to \mathbb{V}$ (b) Order of processing of Ω

Fig. 3. (a) An image $\mathcal{F} : \Omega \to \mathbb{V}$, following the same conventions as in Fig. 1. (b) The order of processing of the points of Ω by Alg. 1, from the first processed point ("1") to the last processed point ("36"). At Line 15 of Alg. 1, the points $\mathbf{y}$ adjacent to $\mathbf{x}$ are considered in the clockwise order, starting from the point on the right of $\mathbf{x}$.

4.3 Complexity analysis

In this analysis, we assume that $|\frown| = O(|\Omega|)$, which is the case in digital images. We note $\kappa(\mathbb{V})$ the time cost required to compare two elements of $\mathbb{V}$ or to compute their infimum. Depending on the way $(\mathbb{V}, \leqslant)$ is modeled, $\kappa(\mathbb{V})$ may vary from $O(1)$ (with a space cost of $O(|\mathbb{V}|^2)$) to $O(\log |\mathbb{V}|)$ or $O(|\mathbb{V}|)$ (depending on the equilibrium of the Hasse diagram, with a space cost of $O(|\mathbb{V}|)$). We note $h(\mathbb{V}) \in \mathbb{N}$ the height of the Hasse diagram of $(\mathbb{V}, \leqslant)$.

Regarding the data structures:

- The size of `nodes` is $O(|\Omega|)$. It is initialized with a time cost $O(1)$. When accessing `nodes[`v`]` for reading or writing, the induced time cost is $O(\log(|\Omega|))$.
- The size of `points` is $O(|\Omega|)$. It is initialized with a time cost $O(|\Omega| \cdot \kappa(|\mathbb{V}|))$. When accessing a set `points[`v`]` for reading or writing, the induced time cost is $O(1)$.
- The size of `status` is $O(|\Omega|)$. It is initialized with a time cost $O(|\Omega|)$. Accessing it for reading or writing has a time cost $O(1)$.
- The size of `nb_nodes` and `index` is $O(|\Omega|)$. They are initialized with a time cost $O(1)$. Accessing them for reading or writing has a time cost $O(\log(|\Omega|))$.
- The size of `progress` is $O(h(\mathbb{V}))$. It is initialized with a time cost $O(1)$. Accessing it for reading or writing has a time cost $O(\log(h(\mathbb{V})))$.

Based on these considerations, the time cost for Alg. 1 (except Line 11) is $O(|\Omega| \cdot \kappa(|\mathbb{V}|))$.

The time cost of Flood depends on:

- the size of Θ. In particular, for each node of Θ, Flood is called once, with an induced time cost $O(\log(|\Omega|) + \log(h(\mathbb{V})))$ related to Lines 1 and 30–31;

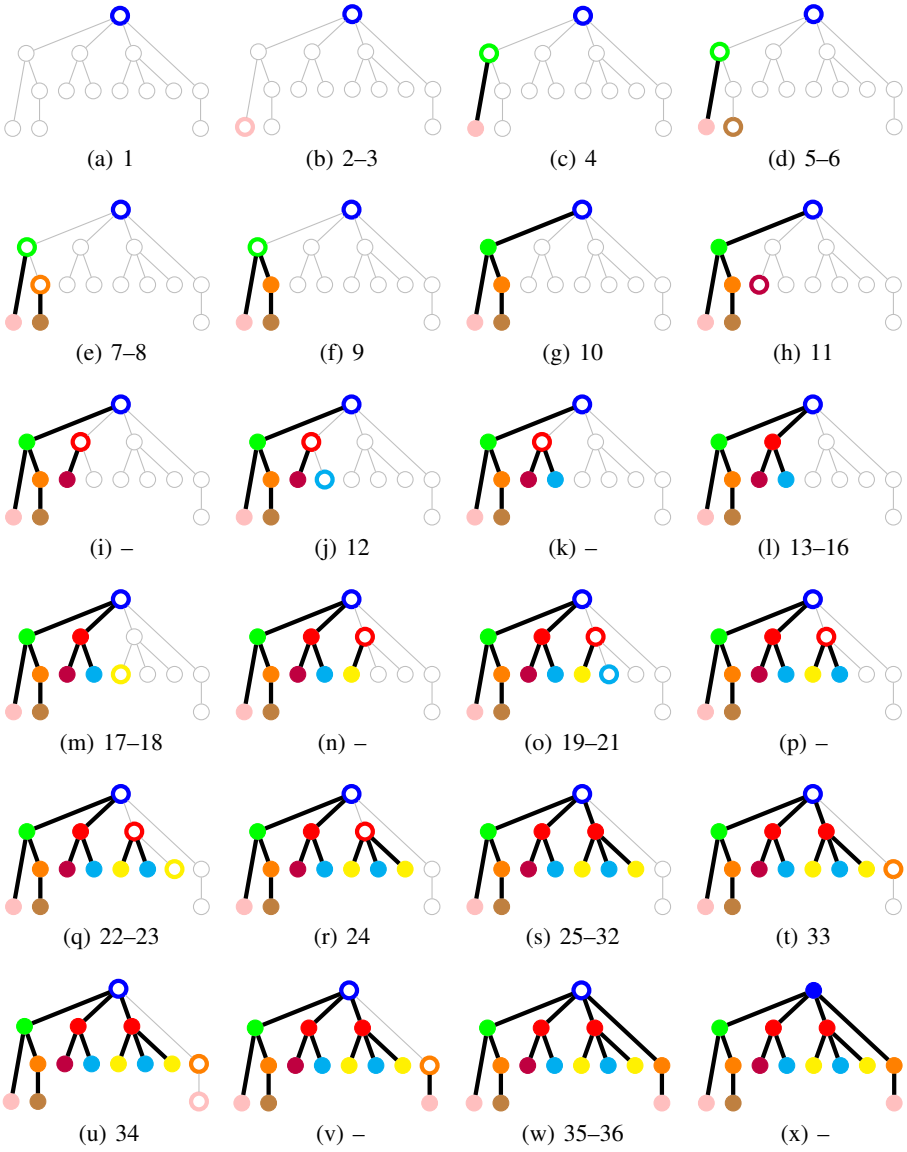


Fig. 4. Construction of the multivalued component tree of the image $\mathcal{F} : \Omega \to \mathbb{V}$ of Fig. 3(a). The number(s) in the subfigure captions (a–x) correspond to the points $\mathbf{x} \in \Omega$ processed by Alg. 1 at the current stage, as depicted in Fig. 3(b). At a current stage: a plain coloured node is fully built; a contour-colored node is under construction; a non-colored node has not been considered yet; a black edge is built; a light gray edge is not built.

– the size of $\lhd$. In particular, for each edge of $\lhd$, Flood is called once, with an induced time cost $O(\tau(X) \cdot (\log(h(\mathbb{V}))) + \kappa(|\mathbb{V}|))$ related to Lines 24–29 (where X is the node associated to the processed edge (X, Y));

– the number of points of Ω. In particular, for each point $\mathbf{x} \in \Omega$, the while loop of Flood (Lines 2–23) is run once or twice, with an induced time cost $O(\log(|\Omega|) + \log(h(\mathbb{V})) + \kappa(\mathbb{V}))$.

It follows that the overall time cost of the construction process (Alg. 1 and Func. Flood) is

$$\mathcal{T} = O\Big(|\Omega| \cdot \big(\log(|\Omega|) + h(\mathbb{V}) \cdot \log(h(\mathbb{V})) + h(\mathbb{V}) \cdot \kappa(|\mathbb{V}|)\big)\Big) \tag{8}$$

If the Hasse diagram of $(\mathbb{V}, \leqslant)$ is well balanced, the time cost becomes

$$\mathcal{T} = O\Big(|\Omega| \cdot \Big(\log(|\Omega|) + (\log(|\mathbb{V}|))^2\Big)\Big) \tag{9}$$

The algorithms dedicated to build the standard component tree present a quasi-linear computational cost $O(|\Omega| \cdot \log(|\Omega|))$. The initial algorithm dedicated to build the multivalued component tree [10] (see Sect. 4.1) relies on such quasi-linear time cost algorithms. In addition, it requires a pre- and a post-processing step. During the pre-processing, the support of the image is extended from Ω to $\widehat{\Omega}$, and the time cost of the subsequent component tree construction is then $O(|\widehat{\Omega}| \cdot \log(|\widehat{\Omega}|))$. We have $|\widehat{\Omega}| \geq |\Omega|$, and the size of $\widehat{\Omega}$ depends on the size of the subset of edges of $\frown$ that link vertices of Ω with non-comparable values. More precisely, we have $|\widehat{\Omega}| = |\Omega|$ in the best scenario, i.e. when all the couples of adjacent vertices $\mathbf{x} \frown \mathbf{y}$ are such that $\mathcal{F}(\mathbf{x})$ and $\mathcal{F}(\mathbf{y})$ are comparable. By contrast, we have $|\widehat{\Omega}| = |\Omega| + O(|\frown|)$ when all the couples of adjacent vertices $\mathbf{x} \frown \mathbf{y}$ are such that $\mathcal{F}(\mathbf{x})$ and $\mathcal{F}(\mathbf{y})$ are non-comparable. This last case may generally occur whenever the Hasse diagram of $(\mathbb{V}, \leqslant)$ is well-balanced. In the case of a d-dimensional digital image, the size of $\frown$ is $d \cdot |\Omega|$. In this context, the overall time cost of the computation of the multivalued component-tree is $O(d \cdot |\Omega| \log |\Omega|)$. We observe in particular that the initial algorithm [10] and the new one proposed here are not sensitive to the same parameters. The first has a time cost that progressively degrades while the dimension of the image increases, while the second has a time cost that progressively degrades while the size of the value space increases. It follows that both algorithms are complementary, and may be considered depending on the application hypotheses.

5 Hierarchical order construction

Building the multivalued component tree of an image $\mathcal{F} : \Omega \to \mathbb{V}$ requires a hierarchical order on the set of values $\mathbb{V}$. In this section, we discuss the ways to endow $\mathbb{V}$ with such hierarchical orders (or, more generally, preorders) $\leqslant$. Two strategies are proposed:

– building a preorder $\leqslant$ on the only values of $\mathbb{V}$ (Sect. 5.1);
– enriching $\mathbb{V}$ with additional values leading to a larger set $\mathbb{W}$ and defining an order $\leqslant$ on $\mathbb{W}$ so that the values of $\mathbb{V}$ are the maximal elements of $(\mathbb{W}, \leqslant)$ (Sect. 5.2).

5.1 (Pre)ordering the value set

We first aim to build a hierarchical preorder $\leqslant_{\mathbb{V}}$ on $\mathbb{V}$. Equivalently, we must set:

– an equivalence relation $\sim$ on $\mathbb{V}$ that gathers values which are mutually and symmetrically comparable, leading to a quotient set $\mathbb{V}/\!\sim$, noted $\mathbb{K}$;

– a hierarchical order $\leqslant_{\mathbb{K}}$ on $\mathbb{K}$.

This preorder $\leqslant_{\mathbb{V}}$ is then defined, for all $u, v \in \mathbb{V}$, by

$$(u \leqslant_{\mathbb{V}} v) \Longleftrightarrow ([u]_{\sim} \leqslant_{\mathbb{K}} [v]_{\sim}) \tag{10}$$

Let us come back to the notion of a component tree (see Sect. 3 by assuming that $\leqslant$ is a total order). We consider a graph $\mathfrak{G}_{\Delta} = (\Delta, \frown_{\Delta})$ where Δ is a finite set and $\frown_{\Delta}$ is an adjacency on Δ, and a function $\delta : \Delta \to \mathbb{N}$ (with $\mathbb{N}$ endowed with the usual $\leq$ relation). Following Sect. 3, one can build the component tree $\mathfrak{T}_{\Delta} = (\Theta_{\Delta}, \lhd_{\Delta})$ of $(\mathfrak{G}_{\Delta}, \delta)$.

The set Θ_{Δ} is a cover of Δ. More precisely, we have $\bigcup \Theta_{\Delta} = \Delta$ and $\forall A \in \Theta_{\Delta}, A \neq \emptyset$. However, two distinct elements $A, B \in \Delta$ may have a non-empty intersection. Indeed, for all $A, B \in \Delta$ we have $A \cap B \neq \emptyset \Rightarrow A \subseteq B \vee B \subseteq A$. This last point may prohibit Θ_{Δ} to be a partition of Δ. Nonetheless, we can define the set $\Delta^{\star}$ from Δ composed of the (non-empty) proper parts of the nodes of Θ_{Δ}. Given a node $A \in \Theta_{\Delta}$, the subset $A^{\star} = \rho(A) \subseteq A$ is defined as in Eq. (6). The set $\Theta_{\Delta}^{\star}$ is then defined as

$$\Theta_{\Delta}^{\star} = \{A^{\star} \mid A \in \Theta\} \tag{11}$$

In particular, the application that maps Θ_{Δ} onto $\Theta_{\Delta}^{\star}$ is a bijection, and $\Theta_{\Delta}^{\star}$ is a partition of Δ. It follows that Θ_{Δ} defines an equivalence relation $\sim_{\Delta}$ on Δ.

The component tree $(\Theta_{\Delta}, \lhd_{\Delta})$ is the Hasse diagram of the ordered set $(\Theta_{\Delta}, \subseteq)$. Since Θ_{Δ} and $\Theta_{\Delta}^{\star}$ are in bijection, we can derive the order $\subseteq^{\star}$ on $\Theta_{\Delta}^{\star}$ by

$$\left(A^{\star} \subseteq^{\star} B^{\star}\right) \Longleftrightarrow (A \subseteq B) \tag{12}$$

The Hasse diagram $(\Theta_{\Delta}^{\star}, \lhd_{\Delta}^{\star})$ of $(\Theta_{\Delta}^{\star}, \subseteq^{\star})$ is then isomorphic to the Hasse diagram $(\Theta_{\Delta}, \lhd_{\Delta})$, i.e. the component tree of $(\mathfrak{G}_{\Delta}, \delta)$.

Following the notations given at the beginning of this section, and setting $\Delta = \mathbb{V}$, $\sim_{\Delta} = \sim$, $\mathbb{K} = \Theta_{\Delta}^{\star}$ and $\subseteq^{\star} = \leqslant_{\mathbb{K}}$, we can define a hierarchical preorder $\leqslant_{\mathbb{V}}$ on $\mathbb{V}$ from a component tree. In particular, it is only required that $\mathbb{V}$ be endowed with the two elements necessary for building this component tree, namely:

– an adjacency relation $\frown_{\mathbb{V}}$, allowing to map a graph structure on $\mathbb{V}$;
– a function $\delta_{\mathbb{V}} : \mathbb{V} \to \mathbb{N}$, allowing to associate to each element of $\mathbb{V}$ a value within the totally ordered set $(\mathbb{N}, \leq)$.

Example Let us consider a colour image $\mathcal{F} : \Omega \to \mathbb{V}$ where the colour values are encoded in the 8-bit per band RGB space $\mathbb{V} = [\![0, 255]\!]^3$. We model $\mathbb{V}$ as the RGB cube, where each colour $v = (r, g, b) \in [\![0, 255]\!]^3$ corresponds to a point in the Cartesian space. We endow $\mathbb{V}$ with the standard 6-adjacency $\frown_{\mathbb{V}}$ defined in digital topology [22], that models the 1-distance between two colours with respect to the ℓ_1 norm. We set $\mathfrak{G}_{\Delta} = (\mathbb{V}, \frown_{\mathbb{V}})$. Let us define $\delta_{\mathbb{V}}$ as the histogram of the image $\mathcal{F}$. In other words, for any colour $v \in \mathbb{V}$, we set $\delta_{\mathbb{V}}(v) = |\{\mathbf{x} \in \Omega \mid \mathcal{F}(\mathbf{x}) = v\}|$. Based on the above discussion, the component tree $\mathfrak{T}_{\Delta} = (\Theta_{\Delta}, \lhd_{\Delta})$ built from $(\mathfrak{G}_{\Delta}, \delta_{\mathbb{V}})$ defines the Hasse diagram of a hierarchical preorder $\leqslant_{\mathbb{V}}$ on $\mathbb{V}$. This example is illustrated in a simplified version in Fig. 5. From $\mathcal{F}$ and $\leqslant_{\mathbb{V}}$, it is then possible to build the multivalued component tree of $\mathcal{F}$ induced by its histogram.

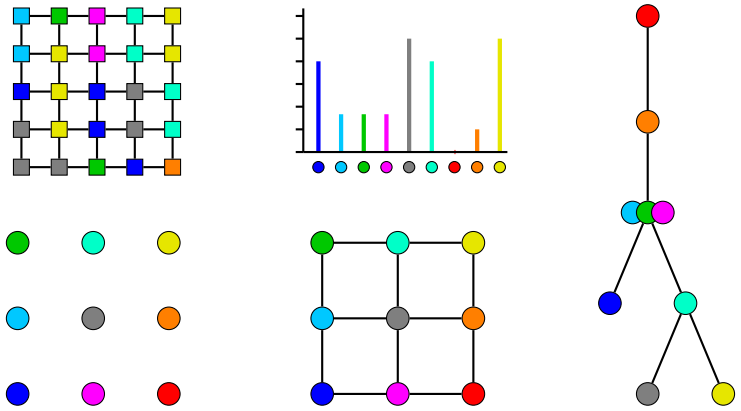


Fig. 5. Illustration of the construction of a hierarchical preorder on a value set (see Sect. 5.1). (a) Top: an image $\mathcal{F} : \Omega \to \mathbb{V}$. The set Ω is equal to $[\![0, 4]\!]^2$ and is endowed with an adjacency relation $\frown$ corresponding to the 4-adjacency. Bottom: the set $\mathbb{V}$ composed of 9 values. (b) Top: the histogram of the image $\mathcal{F}$, used as function $\delta_{\mathbb{V}} : \mathbb{V} \to \mathbb{N}$. Bottom: the set $\mathbb{V}$ is endowed with an adjacency $\frown_{\mathbb{V}}$. (c) The component tree of $(\mathbb{V}, \delta_{\mathbb{V}})$ seen as an image from the set of values $\mathbb{V}$ to $\mathbb{N}$ where $\mathbb{V}$ is endowed with $\frown_{\mathbb{V}}$. This component tree defines a hierarchical preorder $\leqslant$, where the red value is the minimum, the dark blue, yellow and grey values are the maximal elements, and where the three values light blue, green and fushia are mutually greater and lower. Once $\mathbb{V}$ is endowed with $\leqslant$, it becomes possible to compute the multivalued component tree of the image $\mathcal{F} : \Omega \to \mathbb{V}$ of (a) with respect to $\leqslant$.

5.2 Ordering the enriched value set

We now aim to build a hierarchical (pre)order $\leqslant_{\mathbb{W}}$ on a superset $\mathbb{W}$ of $\mathbb{V}$ so that $\mathbb{V} = \nabla^{\leqslant_{\mathbb{W}}}\mathbb{W}$, i.e. the elements of $\mathbb{V}$ are the maximal elements with respect to $\leqslant_{\mathbb{W}}$.

The smallest set $\mathbb{W}$ that can be proposed is $\mathbb{W} = \mathbb{V} \cup \{\bot\}$ where $\bot \notin \mathbb{V}$ is a unique element added to $\mathbb{V}$ that acts as minimum for $\leqslant_{\mathbb{W}}$ (such paradigm was investigated in [21]). The induced hierarchical preorder (which is indeed an order) is the relation $\leqslant_{\mathbb{W}}$ defined as $\{(\bot, v) \mid v \in \mathbb{V}\}$ with $\bot = \bigwedge^{\leqslant_{\mathbb{W}}} \mathbb{W}$ and $\mathbb{V} = \nabla^{\leqslant_{\mathbb{W}}}\mathbb{W}$. It is possible to build larger supersets $\mathbb{W}$ and to endow them with hierarchical preorders $\leqslant_{\mathbb{W}}$ that also fulfill the above assumption. Such sets $\mathbb{W}$ can be of arbitrary size.

We first observe that defining a preorder instead of an order is not relevant (by contrast with Sect. 5.1). Let $\mathbb{W}$ be a set and $\leqslant_{\mathbb{W}}$ a hierarchical preorder on $\mathbb{W}$. We assume that $\nabla^{\leqslant_{\mathbb{W}}}\mathbb{W} = \mathbb{V}$ and $\bigwedge^{\leqslant_{\mathbb{W}}} \mathbb{W} = \bot$, with $\bot \in \mathbb{W} \setminus \mathbb{V}$. For any $w \in \mathbb{W}$, we set $\mathbb{V}_w = \{v \in \mathbb{V} \mid w \leqslant_{\mathbb{W}} v\}$ (note that $\mathbb{V}_w \neq \emptyset$). Let $w_1, w_2 \in \mathbb{W}$. For any $w_1, w_2 \in \mathbb{W}$, we have

$$(w_1 \leqslant_{\mathbb{W}} w_2 \wedge w_2 \leqslant_{\mathbb{W}} w_1) \Longrightarrow (\mathbb{V}_{w_2} = \mathbb{V}_{w_1}) \tag{13}$$

The image $\mathcal{F}$ can also be seen as a function $\mathcal{F} : \Omega \to \mathbb{W}$. For any $w \in \mathbb{W}$, we have (see Eq. (1))

$$\Lambda_w(\mathcal{F}) = \{\mathbf{x} \in \Omega \mid w \leqslant_{\mathbb{W}} \mathcal{F}(\mathbf{x})\} = \{\mathbf{x} \in \Omega \mid \mathcal{F}(\mathbf{x}) \in \mathbb{V}_w\} \tag{14}$$

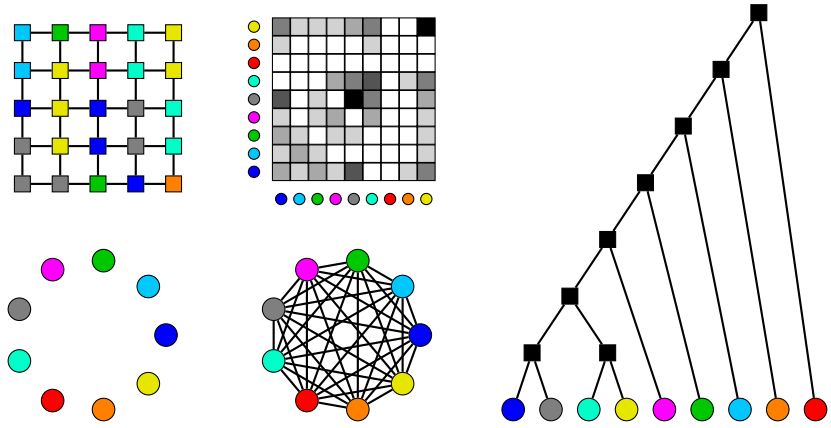


Fig. 6. Illustration of the construction of an order on a value set (see Sect. 5.2). (a) Top: an image $\mathcal{F} : \Omega \to \mathbb{V}$. The set Ω is equal to $[\![0,4]\!]^2$ and is endowed with an adjacency relation $\frown$ corresponding to the 4-adjacency. Bottom: the set $\mathbb{V}$ composed of 9 values, without initial ordering. (b) Top: the co-occurence matrix of the image $\mathcal{F}$, used as a priority function $\delta_{\frown_\mathbb{V}} : \frown_\mathbb{V} \to \mathbb{N}$. Bottom: the set $\mathbb{V}$ is endowed with an adjacency $\frown_\mathbb{V}$. (c) The binary partition tree of $(\mathbb{V}, \frown_\mathbb{V})$ induced by $\delta_{\frown_\mathbb{V}}$. This binary partition tree defines a hierarchical order $\leqslant$ on an enriched set of values $\mathbb{W}$ where the values of $\mathbb{V}$ are the maximal elements.

Now, let us consider two distinct values $w_1, w_2 \in \mathbb{W}$ such that $w_1 \leqslant_\mathbb{W} w_2$ and $w_2 \leqslant_\mathbb{W} w_1$. From Eqs. (13–14) it follows that $C[\Lambda_{w_1}(\mathcal{F})] = C[\Lambda_{w_2}(\mathcal{F})]$. In other words, relaxing the antisymmetry property to define a preorder instead of an order is useless, since it leads to define many times the same nodes which are modeled once in Θ (see Eq. (2)). We can then assume without loss of generality that $\leqslant_\mathbb{W}$ is a hierarchical order.

Although $\mathbb{W}$ may be of arbitrary size, we now observe that it is sufficient to define some sets $\mathbb{W}$ that may not be larger than twice the size of $\mathbb{V}$. Let us set $\mathbb{V}_\mathbb{W} = \{\mathbb{V}_w \mid w \in \mathbb{W}\} \subseteq 2^\mathbb{V}$. We define the equivalence relation $\equiv$ on $\mathbb{W}$ by

$$(w_1 \equiv w_2) \iff (\mathbb{V}_{w_1} = \mathbb{V}_{w_2}) \tag{15}$$

Let $w \in \mathbb{W}$. The equivalence class $[w]_\equiv$ is totally ordered by $\leqslant_\mathbb{W}$. We set $\widehat{w} = \bigvee^{\leqslant_\mathbb{W}} [w]_\equiv$. We note $\widehat{\mathbb{W}} = \{\widehat{w} \mid w \in \mathbb{W}\}$. Let $w_1, w_2 \in \mathbb{W}$. We have

$$(w_1 \leqslant_\mathbb{W} w_2) \Longrightarrow (\widehat{w}_1 \leqslant_\mathbb{W} \widehat{w}_2) \tag{16}$$

In other words, there is a homomorphism from $(\mathbb{W}, \leqslant_\mathbb{W})$ to $(\widehat{\mathbb{W}}, \leqslant_{\widehat{\mathbb{W}}})$ where $\leqslant_{\widehat{\mathbb{W}}}$ is the restriction of $\leqslant_\mathbb{W}$ to $\widehat{\mathbb{W}}$.

By construction, the set $\widehat{\mathbb{W}}$ is in bijection with $\mathbb{V}_\mathbb{W}$. Since $\mathbb{V} = \nabla^{\leqslant_\mathbb{W}}\mathbb{W}$, we also have $\mathbb{V} \subseteq \widehat{\mathbb{W}}$. Following Eq. (2), we set $\Theta_\mathbb{W} = \bigcup_{w\in\mathbb{W}} C[\Lambda_w(\mathcal{F})]$ and $\Theta_{\widehat{\mathbb{W}}} = \bigcup_{w\in\widehat{\mathbb{W}}} C[\Lambda_w(\mathcal{F})]$. We have $(\Theta_\mathbb{W}, \subseteq) = (\Theta_{\widehat{\mathbb{W}}}, \subseteq)$. It follows that the two associated multivalued component trees are equal. As a conclusion, instead of using a set $\mathbb{W}$ arbitrarily large and potentially infinite, a same multivalued component tree is obtained by considering the set $\widehat{\mathbb{W}}$, which

is finite. Indeed, from the definition of $\equiv$, $\widehat{\mathbb{W}}$ is in bijection with $\mathbb{V}_{\mathbb{W}} \subseteq 2^{\mathbb{V}}$. We even have a stronger result, since the bijection between $\widehat{\mathbb{W}}$ and $\mathbb{V}_{\mathbb{W}}$ induces an isomorphism between $(\widehat{\mathbb{W}}, \leqslant_{\widehat{\mathbb{W}}})$ and $(\mathbb{V}_{\mathbb{W}}, \subseteq)$.

Let us now focus on the nature of the Hasse diagram of $(\mathbb{V}_{\mathbb{W}}, \subseteq)$. The inclusion $\subseteq$ on $\mathbb{V}_{\mathbb{W}}$ is a hierarchical order. The Hasse diagram $(\mathbb{V}_{\mathbb{W}}, \lhd)$ is, in particular, a partition tree. The fact that $\mathbb{V} = \nabla^{\leqslant_{\mathbb{W}}}\mathbb{W}$ implies that $\{\{v\} \mid v \in \mathbb{V}\} = \Delta^{\subseteq}\mathbb{V}_{\mathbb{W}}$. It follows that $(\mathbb{V}_{\mathbb{W}}, \lhd)$ is a total partition tree. A corollary of this property is that $|\mathbb{V}_{\mathbb{W}}| < 2 \cdot |\mathbb{V}|$.

To conclude on this analysis, it appears that for building a hierarchical order $\leqslant_{\mathbb{W}}$ on a superset $\mathbb{W}$ of $\mathbb{V}$ so that the elements of $\mathbb{V}$ be the maximal elements of $\leqslant_{\mathbb{W}}$, i.e. $\mathbb{V} = \nabla^{\leqslant_{\mathbb{W}}}\mathbb{W}$, the most simple, yet general solution is to build a total partition tree from the initial, finest partition of $\mathbb{V}$, namely $\{\{v\} \mid v \in \mathbb{V}\}$. This can be done by building a (binary) partition tree, following the standard construction algorithms proposed in [23]. To this end, it is only required that $\mathbb{V}$ be endowed with:

- an adjacency relation $\frown_{\mathbb{V}}$, allowing to map a graph structure on $\mathbb{V}$;
- a priority function $\delta_{\frown_{\mathbb{V}}} : \frown_{\mathbb{V}} \to \mathbb{N}$, allowing to determine the couples of nodes to be merged in priority.

Example Let us consider an image $\mathcal{F} : \Omega \to \mathbb{V}$. The support Ω is endowed with an adjacency relation $\frown$. The value space $\mathbb{V}$ is endowed with the adjacency relation $\frown_{\mathbb{V}}$ so that $\mathfrak{G}_{\Delta} = (\mathbb{V}, \frown_{\mathbb{V}})$ is an irreflexive complete graph (i.e. $\forall u, v \in \mathbb{V}, u \neq v \Leftrightarrow u \frown_{\mathbb{V}} v$). We define the co-occurrence matrix [7] of the image $\mathcal{F}$. This matrix $\mathcal{M} = (m_{u,v})_{u,v \in \mathbb{V}}$ is of dimension $|\mathbb{V}| \times |\mathbb{V}|$. For each couple $(u, v) \in \mathbb{V} \times \mathbb{V}$, it is defined by $m_{u,v} = |\{(\mathbf{x}, \mathbf{y}) \in \Omega \times \Omega \mid \mathbf{x} \frown \mathbf{y} \wedge \mathcal{F}(\mathbf{x}) = u \wedge \mathcal{F}(\mathbf{y}) = v\}|$. We define the priority function $\delta_{\frown_{\mathbb{V}}} : \frown_{\mathbb{V}} \to \mathbb{N}$ so that for any $(u, v) \in \frown_{\mathbb{V}}$, i.e. for any $u \frown_{\mathbb{V}} v$, we have $\delta_{\frown_{\mathbb{V}}}((u, v)) = m_{u,v}$. Based on the above discussion, the partition-tree $\mathfrak{T}_{\Delta} = (\Theta_{\Delta}, \lhd_{\Delta})$ built [23] from $(\mathfrak{G}_{\Delta}, \delta_{\frown_{\mathbb{V}}})$ defines the Hasse diagram of a hierarchical order $\leqslant_{\mathbb{V}}$ on $\mathbb{V}$. This example is illustrated in a simplified version in Fig. 6. From $\mathcal{F}$ and $\leqslant_{\mathbb{V}}$, it is then possible to build the multivalued component tree of $\mathcal{F}$ induced by its co-occurrence matrix.

6 Conclusion

In this article, we have provided new algorithmic tools for building the multivalued component tree, but also for designing hierarchical orders on sets of values, which is a required condition of images to be modeled via multivalued component trees. In previous works [10], it had already been observed that the multivalued component tree could be efficiently involved for processing label images, especially on the context of hierarchical classification. Recent advances in the study of component trees have shed light on their links with persistent homology [15], and more generally their ability to model high-level topological information. In this context, component trees are being increasingly considered as relevant topological data descriptors to be embedded in deep-learning frameworks, e.g. for the design of loss functions [19] or to model the image structure information in self-supervised learning [28]. The contributions proposed in this article allow to efficiently handle component trees not only on grey-level images, but more generally on any multivalued images endowed with a hierarchical order. This generalization paves the way to the involvement of the (multivalued) component trees

in various computer vision tasks (in particular based on deep-learning) especially in the context of multivalued data, which is for instance the case in semantic segmentation.

References

1. Blin, N., Carlinet, E., Lemaitre, F., Lacassagne, L., Géraud, T.: Max-tree computation on GPUs. IEEE Transactions on Parallel and Distributed Systems **33**, 3520–3531 (2022)
2. Breen, E.J., Jones, R.: Attribute openings, thinnings, and granulometries. Computer Vision and Image Understanding **64**, 377–389 (1996)
3. Carlinet, E., Géraud, T.: A comparative review of component tree computation algorithms. IEEE Transactions on Image Processing **23**, 3885–3895 (2014)
4. Gazagnes, S., Wilkinson, M.H.F.: Distributed connected component filtering and analysis in 2D and 3D tera-scale data sets. IEEE Transactions on Image Processing **30**, 3664–3675 (2021)
5. Götz, M., Cavallaro, G., Géraud, T., Book, M., Riedel, M.: Parallel computation of component trees on distributed memory machines. IEEE Transactions on Parallel and Distributed Systems **29**, 2582–2598 (2018)
6. Guigues, L., Cocquerez, J.P., Le Men, H.: Scale-sets image analysis. International Journal of Computer Vision **68**, 289–317 (2006)
7. Haralick, R.M., Shanmugam, K.S., Dinstein, I.: Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics **3**, 610–621 (1973)
8. Jones, R.: Connected filtering and segmentation using component trees. Computer Vision and Image Understanding **75**, 215–228 (1999)
9. Kiran, B.R., Serra, J.: Global-local optimizations by hierarchical cuts and climbing energies. Pattern Recognition **47**, 12–24 (2014)
10. Kurtz, C., Naegel, B., Passat, N.: Connected filtering based on multivalued component-trees. IEEE Transactions on Image Processing **23**, 5152–5164 (2014)
11. Monasse, P., Guichard, F.: Scale-space from a level lines tree. Journal of Visual Communication and Image Representation **11**, 224–236 (2000)
12. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **18**, 1163–1173 (1996)
13. Ouzounis, G.K., Wilkinson, M.H.F.: Mask-based second-generation connectivity and attribute filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**, 990–1004 (2007)
14. Passat, N., Kenmochi, Y.: A topological tree of shapes. In: DGMM, Procs. pp. 221–235 (2022)
15. Passat, N., Mendes Forte, J., Kenmochi, Y.: Morphological hierarchies: A unifying framework with new trees. Journal of Mathematical Imaging and Vision **65**, 718–753 (2023)
16. Passat, N., Naegel, B., Kurtz, C.: Component-graph construction. Journal of Mathematical Imaging and Vision **61**, 798–823 (2019)
17. Passat, N., Naegel, N.: Component-trees and multivalued images: Structural properties. Journal of Mathematical Imaging and Vision **49**, 37–50 (2014)
18. Perret, B., Lefèvre, S., Collet, C., Slezak, É.: Hyperconnections and hierarchical representations for grayscale and multiband image processing. IEEE Transactions on Image Processing **21**, 14–27 (2012)
19. Perret, B., Cousty, J.: Component tree loss function: Definition and optimization. In: DGMM, Procs. pp. 248–260 (2022)

20. Randrianasoa, J.F., Kurtz, C., Passat, N.: Binary partition tree construction from multiple features for image segmentation. Pattern Recognition **84**, 237–250 (2018)
21. Ronse, C., Agnus, V.: Morphology on label images: Flat-type operators and connections. Journal of Mathematical Imaging and Vision **22**, 283–307 (2005)
22. Rosenfeld, A.: Digital topology. The American Mathematical Monthly **86**, 621–630 (1979)
23. Salembier, P., Garrido, L.: Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. IEEE Transactions on Image Processing **9**, 561–576 (2000)
24. Salembier, P., Oliveras, A., Garrido, L.: Anti-extensive connected operators for image and sequence processing. IEEE Transactions on Image Processing **7**, 555–570 (1998)
25. Salembier, P., Serra, J.: Flat zones filtering, connected operators, and filters by reconstruction. IEEE Transactions on Image Processing **4**, 1153–1160 (1995)
26. Salembier, P., Wilkinson, M.H.F.: Connected operators. IEEE Signal Processing Magazine **26**, 136–157 (2009)
27. Soille, P.: Constrained connectivity for hierarchical image decomposition and simplification. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**, 1132–1145 (2008)
28. Tang, Q., Du, B., Xu, Y.: Self-supervised learning based on max-tree representation for medical image segmentation. In: IJCNN, Procs. pp. 1–6 (2022)