Directed connected operators: asymmetric hierarchies for image filtering and segmentation

Benjamin Perret, Jean Cousty, Olena Tankyevych, Hugues Talbot, Nicolas Passat

Directed connected operators: asymmetric hierarchies for image filtering and segmentation

Benjamin Perret, Jean Cousty, Olena Tankyevych, Hugues Talbot, *Member, IEEE,* Nicolas Passat

Abstract—Connected operators provide well-established solutions for digital image processing, typically in conjunction with hierarchical schemes. In graph-based frameworks, such operators basically rely on symmetric adjacency relations between pixels. In this article, we introduce a notion of *directed connected operators* for hierarchical image processing, by also considering non-symmetric adjacency relations. The induced image representation models are no longer partition hierarchies (i.e., trees), but directed acyclic graphs that generalize standard morphological tree structures such as component trees, binary partition trees or hierarchical watersheds. We describe how to efficiently build and handle these richer data structures, and we illustrate the versatility of the proposed framework in image filtering and image segmentation.

Index Terms—Mathematical morphology, connected operators, hierarchical image representation, antiextensive filtering, segmentation.

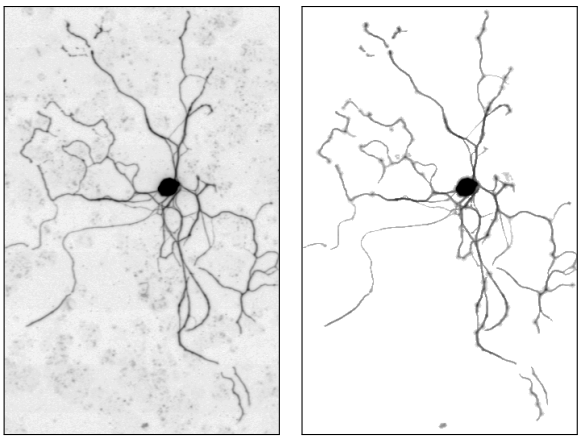
1 Introduction

Graphs are a popular framework for image processing and analysis. They allow for the representation of various adjacency relations (the edges) between pixels (the vertices). Valuation can appear both on the vertices in order to model a luminance information, or on the edges as a measure of dissimilarity. Following the historical symmetric definition of adjacency [1], [2], most methods rely on undirected graphs. Nevertheless, some recent works aimed at extending these beyond the symmetry hypothesis in order to improve the results of popular image segmentation algorithms. These works have led to different seeded segmentation algorithms based on the directed graph framework, and generally show better performances than their symmetric counterpart. Such works include min-cuts [3], the random-walker [4], and shortest path forests [5]. Following these successful attempts, we propose to explore how directed graphs can enrich and improve another family of graph operators: the connected operators. A preliminary version of this work was presented in a recent conference paper [6].

1.1 Connected operators

Connected operators [7], [8], [9] are popular image processing tools defined in the framework of mathematical morphology.

Benjamin Perret, Jean Cousty, and Hugues Talbot are with the ESIEE-Paris and the Université Paris-Est Marne-la-Vallée, LIGM, Paris, France ({b.perret,j.cousty,h.talbot}@esiee.fr).
Olena Tankyevych is with the Université Paris-Est Créteil, LISSI, Paris, France (olena.tankyevych@u-pec.fr).
Nicolas Passat is with the Université de Reims Champagne-Ardenne, CReSTIC, Reims, France (nicolas.passat@univ-reims.fr).



(a) (b)

Fig. 1. (a) Neurite image. (b) Directed connected filtering of (a).

They have been successfully applied in a wide spectrum of applications (see [10] [11, Ch. 7] for a recent surveys). Connected operators focus on the notion of connected components, i.e., maximal sets of vertices in which a path exists between any two vertices. Their very principle is that the only allowed operation is the deletion of connected components, thus ensuring that they can neither create nor shift contours, but only flatten images. The extension of this approach to grayscale images (vertex or edge weighted graphs) leads to the definition of several hierarchical representations: the component tree [12], the binary partition tree [13], or the tree of shapes [14]. Significant effort has been devoted to efficiently construct these hierarchies [12], [15], [16], [17] and to understand the relations that exist between them [18], [19]. A general scheme to define a connected operator consists of a four step procedure: (1) construct the hierarchical representation of the image; (2) compute attributes at each node of the representation; (3) select relevant nodes according to their attribute values; and (4) reconstruct a filtered image or a segmentation map and/or extract features from the selected nodes. From an applicative viewpoint, connected operators have been involved in various image processing and analysis tasks, including filtering [12], segmentation [20], interactive segmentation [21], [22], retrieval [23], classification [24], or registration [25]. Applications range from biomedical imaging [26], [27], via

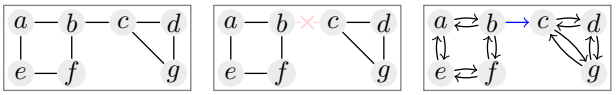


Fig. 2. Differences between undirected and directed graphs (see text).

astronomy [28], [29], remote sensing [30], [31], to document analysis [32], [33].

The design of connected operators faces two major issues: (1) do structures of interest appear in the hierarchical representation? and (2) how can one discriminate structures of interest in the hierarchy? The first issue, often associated to the linkage/leakage problem, has been investigated through the definition of second-generation connections [34], [35], [36], [37], constrained connectivity [38], and hyperconnections [33], [39]. The second issue of selecting relevant nodes of the hierarchy is twofold: (1) the definition of attributes that provide a suitable feature space to separate relevant nodes from the others; and (2) the definition of robust and accurate node selection processes. Although classical shape attributes (area, elongation, various notions of complexity,...) are often considered, a larger effort has been extended in the definition of node selection processes, which have evolved from simple global thresholding [9], [12], [20] to more complex strategies like energy-minimization [40], [41], or connected filtering of the nodes in feature spaces [42].

Although the mentioned solutions are effective, they are not perfect, and we investigate here how the reformulation of connected operators in the context of directed graphs can offer improved practical solutions. Consider the toy example given in Fig. 2(a), the given graph is connected and thus the two possible results of a connected operator are either the empty graph or the graph itself. If we want to achieve a finer result, for instance having identified that the "rectangle" on the left is indeed only *weakly* connected (perhaps due to noise or some topological considerations) to the "triangle" on the right, one possible solution is simply to remove the edge $\{b, c\}$: this corresponds to second generation connections (Fig. 2(b)). However, by doing this, we lose the information about the initial proximity of these two structures. In the directed graph framework, an intermediate solution is to remove an arc in only one direction. Then, if we consider the two strongly connected components, we are able to consider the two parts as separate while still being related (Fig. 2(c)). Moreover, the chosen direction of the remaining arc can convey some useful information for further processing. A practical example of this principle is shown in Fig. 1: here the different parts of the neurite are separated based on a vesselness [43] prior classification and the directed arcs are constructed in order to always go from the least reliable towards the most reliable structures, as identified in the vesselness: from background, to vessels, to blobs. Then a filtering based on two attributes that measure the relations (directional information) among the structures produces the result shown in Fig. 1(b) (Sec. 6).

1.2 Contributions

In this article, we introduce the new notion of *directed connected component* (*directed component* or *D-component*, for brief) which generalizes the notion of connected component to directed graphs (Sec. 2). Furthermore, we establish a bijection theorem (Th. 3) between the D-components and the strongly connected components. In particular, this allows us to rely on well-established tools in graph theory.

We propose the notion of *directed component hierarchy* which extends D-components to weighted graphs (Sec. 2). This structure is a directed acyclic graph, and thus generally is not a tree. But, thanks to the bijection given in Th. 3, we show that this structure indeed generalizes the standard connected component trees [12], [13].

In Sec. 4 we propose an efficient algorithm for building these hierarchies. The algorithm has a $\mathcal{O}(\ell.(n+m))$ time complexity, where m is the number of vertices, n is the number of arcs, and ℓ is the number of weight values.

Then, we present several strategies to select relevant nodes of a D-component hierarchy in order to handle the increased complexity of this structure compared to standard component trees (Sec. 5). More precisely, these strategies are designed to ensure the consistency of the node selection process in terms of D-components.

Finally, we show the versatility of the proposed framework by presenting two applications in image filtering and segmentation (Sec. 6). For both applications, we provide the complete process from the construction of the directed graph to the production of the final result. In particular, we show how prior information can be injected as a directional information in the graphs and we give examples on how the particular structure of the hierarchy can be used to define new kinds of node attributes.

2 Directed connectedness

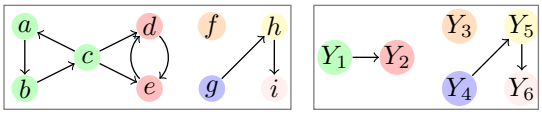
The first goal of this article is to extend connected operators from undirected to directed graphs (Sec. 2.1), via employing the *directed connectedness* (or *D-connectedness*) paradigm, which we introduce in Sec. 2.2. Before investigating the differences between D-connectedness and connectedness (defined in the usual frameworks of undirected graphs or connections [44, Ch. 2] [34]), we first discuss the deep links that exist between D-connectedness and the notion of strong connectedness, usually considered on directed graphs (Secs. 2.3, 2.4).

2.1 Graphs

A *directed graph* (or simply, a *graph*) $\mathfrak{G}$ is a pair $(V, \mathcal{A})$, where V is a nonempty finite set, and $\mathcal{A}$ is composed of pairs of elements of V, i.e., $\mathcal{A}$ is a subset of $V \times V$. Each element of V is called a *vertex*, a *point*, or a *node (of $\mathfrak{G}$)* and each element of $\mathcal{A}$ is called an *arc (of $\mathfrak{G}$)*. A *subgraph of* $\mathfrak{G}$ is a graph $\mathfrak{G}_\star = (V_\star, \mathcal{A}_\star)$ such that $V_\star$ is a subset of V, and $\mathcal{A}_\star$ is a subset of $\mathcal{A}$.

If $\mathfrak{G}$ is a graph, its vertex set is denoted by $V(\mathfrak{G})$ and the arc set by $\mathcal{A}(\mathfrak{G})$.

The *transpose of a graph* $\mathfrak{G}$ is the unique graph with the same vertices as $\mathfrak{G}$, and such that for any of its arcs (x, y),



(a) (b)

Fig. 3. (a) An example of a directed graph (the vertices and arcs are represented by circles and arrows, respectively) whose D-components are $X_1 = \{a, b, c, d, e\}, X_2 = \{d, e\}, X_3 = \{f\}, X_4 = \{g, h, i\}, X_5 = \{h, i\}$, and $X_6 = \{i\}$ and whose S-components are $Y_1 = \{a, b, c\}, Y_2 = \{d, e\}, Y_3 = \{f\}, Y_4 = \{g\}, Y_5 = \{h\}$, and $Y_6 = \{i\}$. (b) The DAG $\mathfrak{D}(\mathfrak{G})$ of the S-components of the graph is depicted in (a).

the pair (y, x) is an arc of $\mathfrak{G}$. We say that $\mathfrak{G}$ is *symmetric* if $\mathfrak{G}$ is the same as its transpose. Thus, $\mathfrak{G}$ is symmetric if for any of its arcs (x, y), the pair (y, x) is also an arc of $\mathfrak{G}$. It is well known that any symmetric graph $\mathfrak{G}$ can be associated to a unique undirected graph, and conversely.

Let $\mathfrak{G}$ be a graph, a *path from a vertex x to a vertex y (in $\mathfrak{G}$)* is a sequence $(x_0, \ldots, x_\ell)$ of vertices of $\mathfrak{G}$ such that $x_0 = x$, $x_\ell = y$, and for any i in $\{1, \ldots, \ell\}$, the pair (x_{i-1}, x_i) is an arc of $\mathfrak{G}$. We say that y is a *successor of x (in $\mathfrak{G}$)* and that x is a *predecessor of y (in $\mathfrak{G}$)* if there exists a path from x to y. The singleton (x) is a (trivial) path and therefore x is a successor and a predecessor of itself.

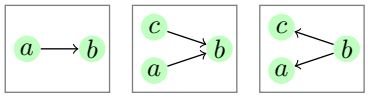
2.2 Directed connected components

In order to take into account "directed subsets" of vertices (i.e., subsets containing some points that play the particular role of "basepoints" or "roots"), we present the notion of a *directed connected component* (or *D-component*).

Definition 1: Let $\mathfrak{G}$ be a graph and let x be a vertex of $\mathfrak{G}$. The *directed connected component of basepoint x* is the set, denoted by $\mathrm{DCC}_{\mathfrak{G}}(x)$, of all the successors of x in $\mathfrak{G}$. This set $\mathrm{DCC}_{\mathfrak{G}}(x)$ is also called a *D-component of* $\mathfrak{G}$, and we denote by $\mathrm{DCC}_{\mathfrak{G}}$ the set of all the D-components of $\mathfrak{G}$.

For instance, in the graph $\mathfrak{G}$ depicted in Fig. 3(a), the vertices g, h and i are the three successors of g. Thus, the D-component $\mathrm{DCC}_{\mathfrak{G}}(g)$ is the set $\{g, h, i\}$. Observe also that a vertex is a basepoint of a D-component if it is a predecessor of all the vertices in this D-component. For instance, the set $\{a, b, c, d, e\}$ is a D-component and b is a predecessor of all the vertices in this D-component. Therefore, the set $\{a, b, c, d, e\}$ is the D-component of basepoint b. Note that this set is also the D-component of basepoints a and c.

In contrast to connected components, the set of all D-components of a graph is not necessarily a partition of its vertex set. Indeed a vertex may belong to several D-components. For instance (see Fig. 4(a)), let us consider two vertices a and b such that b is a successor of a but a is not a successor of b. Then, the point b is in the D-component of basepoint a and in the D-component of basepoint b. These two D-components are distinct since a belongs to the former one but not to the later. However, these components are linked by inclusion: $\mathrm{DCC}_{\mathfrak{G}}(b) \subseteq \mathrm{DCC}_{\mathfrak{G}}(a)$. More generally, some D-components



(a) (b) (c)

Fig. 4. Some elementary graphs.

may intersect without being included in one another. Indeed, let us consider an additional vertex c (see Fig. 4(b)) such that c is a predecessor of b but not a predecessor of a, while c is neither a successor of a nor b. Then, the D-components $\mathrm{DCC}_{\mathfrak{G}}(a)$ and $\mathrm{DCC}_{\mathfrak{G}}(c)$ both contain b but are not included in each other since a is in $\mathrm{DCC}_{\mathfrak{G}}(a)$ but not in $\mathrm{DCC}_{\mathfrak{G}}(c)$ and c is in $\mathrm{DCC}_{\mathfrak{G}}(c)$ but not in $\mathrm{DCC}_{\mathfrak{G}}(a)$. However, similarly to the case of connected components, if a vertex x is in a D-component X, then the whole D-component of basepoint x is included in X. In other words, the underlying binary relation "is a successor of" is in general not an equivalence relation but is always reflexive and transitive. It can also be observed that in general the D-components of a graph and of its transpose are not the same. For instance the graphs depicted in Figs. 4(b) and (c) are the transpose of each other and the D-components of the first are $\{c, b\}$, $\{a, b\}$, and $\{b\}$ whereas the D-components of the second are $\{b, a, c\}$, $\{c\}$, and $\{a\}$.

2.3 Strongly connected components

The notion of a strongly connected component is fundamental in graph theory [45, pp. 552–557].

A subset X of the vertex set of a graph $\mathfrak{G}$ is *strongly connected (for $\mathfrak{G}$)* if any two vertices x and y of X are successors of each other, i.e., $x \in \mathrm{DCC}_{\mathfrak{G}}(y)$ and $y \in \mathrm{DCC}_{\mathfrak{G}}(x)$. A *strongly connected component* (or *S-component*) *of* $\mathfrak{G}$ is a subset X of vertices of $\mathfrak{G}$ that is strongly connected and that is maximal for this property, i.e., any subset of $V(\mathfrak{G})$ which is also a proper superset of X is not strongly connected. We denote by $\mathrm{SCC}_{\mathfrak{G}}$ the set of all S-components of $\mathfrak{G}$.

This set $\mathrm{SCC}_{\mathfrak{G}}$ of all S-components of a graph $\mathfrak{G}$ – contrarily to the set $\mathrm{DCC}_{\mathfrak{G}}$ of all D-components – is a partition of the vertex set of $\mathfrak{G}$, i.e., the union of $\mathrm{SCC}_{\mathfrak{G}}$ is $V(\mathfrak{G})$ and the intersection of any two distinct S-components of $\mathfrak{G}$ is empty. In fact, the relation "is in the same S-component as" is an equivalence relation. Thus, for any vertex x of $\mathfrak{G}$, there is a unique S-component, denoted by $\mathrm{SCC}_{\mathfrak{G}}(x)$, that contains x. For instance, the S-components of the graph depicted in Fig. 3(a) are $\{a, b, c\}$, $\{d, e\}$, $\{f\}$, $\{g\}$, $\{h\}$ and $\{i\}$.

2.4 Links between D- and S-components

Given two vertices x and y of a graph $\mathfrak{G}$ that belong to the same S-component, any successor of x is a successor of y and vice versa. Therefore, the D-components of basepoints x and y are the same. Conversely, if the D-components of basepoints x and y are the same, then x is a successor of y and vice versa, i.e., x and y are in $\mathrm{DCC}_{\mathfrak{G}}(y)$ and in $\mathrm{DCC}_{\mathfrak{G}}(x)$, respectively. In other words x and y are in the same S-component. Hence, S-components and D-components are equivalent according to the following property:

Property 2: Let $\mathfrak{G}$ be a graph. Two vertices x and y of $\mathfrak{G}$ are in the same S-component of $\mathfrak{G}$ if and only if the D-component of basepoint x is equal to the D-component of basepoint y

$$\mathrm{DCC}_{\mathfrak{G}}(x) = \mathrm{DCC}_{\mathfrak{G}}(y) \Longleftrightarrow \mathrm{SCC}_{\mathfrak{G}}(x) = \mathrm{SCC}_{\mathfrak{G}}(y) \quad (1)$$

This implies that all the basepoints enabling the definition of a given D-component form a unique S-component. Each D-component is then associated to a unique S-component. In other words, there is a bijection between the set of D-components of $\mathfrak{G}$ to the set of S-components of $\mathfrak{G}$. This bijection can be expressed based on the *Directed Acyclic Graph* (DAG) of S-components.

We associate to any graph $\mathfrak{G}$ the directed graph $\mathfrak{D}(\mathfrak{G})$, whose vertices are the S-components of $\mathfrak{G}$ and that is such that the pair (X, Y) of S-components of $\mathfrak{G}$ is an arc of $\mathfrak{D}(\mathfrak{G})$ whenever there exists an arc (x, y) of $\mathfrak{G}$ such that x and y are in the S-components X and Y, respectively. This graph has been well studied in graph theory. In particular, it is acyclic, i.e., for any two distinct D-components X and Y, the component X cannot be both a successor and a predecessor of Y in $\mathfrak{D}(\mathfrak{G})$. Therefore, this graph $\mathfrak{D}(\mathfrak{G})$ is called the *DAG of the S-components of* $\mathfrak{G}$. For instance, the DAG of the S-components of the graph depicted in Fig. 3(a) is depicted in Fig. 3(b). For any S-component X of a graph $\mathfrak{G}$, we denote by $B_{\mathfrak{G}}(X)$ the union of the successors of X in the graph $\mathfrak{D}(\mathfrak{G})$

$$B_{\mathfrak{G}}(X) = \bigcup \mathrm{DCC}_{\mathfrak{D}(\mathfrak{G})}(X) \quad (2)$$

Theorem 3: Let $\mathfrak{G}$ be a graph.

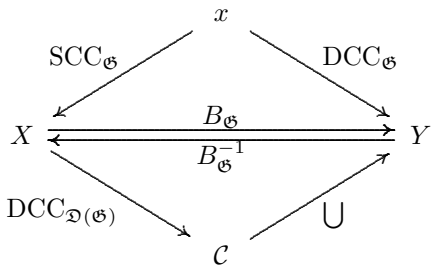
- The map $B_{\mathfrak{G}}$ is a bijection from $\mathrm{SCC}_{\mathfrak{G}}$ to $\mathrm{DCC}_{\mathfrak{G}}$ whose inverse $B_{\mathfrak{G}}^{-1}$ (i.e., $\forall X \in \mathrm{SCC}_{\mathfrak{G}}$, $B_{\mathfrak{G}}^{-1}(B_{\mathfrak{G}}(X)) = X$) is such that for any D-component X of $\mathfrak{G}$ we have

$$B_{\mathfrak{G}}^{-1}(X) = \{x \in V(\mathfrak{G}) \mid \mathrm{DCC}_{\mathfrak{G}}(x) = X\} \quad (3)$$

- The D-component of any basepoint x is the union of the successors of the S-component $\mathrm{SCC}_{\mathfrak{G}}(x)$ in $\mathfrak{D}(\mathfrak{G})$

$$\mathrm{DCC}_{\mathfrak{G}}(x) = B_{\mathfrak{G}}(\mathrm{SCC}_{\mathfrak{G}}(x)) \quad (4)$$

Th. 3 is illustrated in Diag. (5). In particular, for a given graph $\mathfrak{G}$, it can be seen that if one knows the S-component X containing a vertex x of $\mathfrak{G}$, then the D-component Y of basepoint x can be recovered as the direct image of X by the bijection $B_{\mathfrak{G}}$, which can be obtained from the DAG $\mathfrak{D}(\mathfrak{G})$. Conversely, if the directed component Y of basepoint x is known, then one can recover the S-component containing x as the inverse image of X by $B_{\mathfrak{G}}$. This inverse image $B_{\mathfrak{G}}^{-1}(Y)$ of the D-component Y is called the *root of* Y *(for* $\mathfrak{G}$*)*.



(5)

In the next sections, we exploit these links between D-components and S-components to design efficient algorithms for image processing.

3 Directed component hierarchies

Connected operators act on an image represented as a function through the connected components of its level sets (Sec. 3.1). These connected components are organized – via the inclusion relationship – in a tree structure, known as the component tree [12]. In this section, we extend this structure from undirected graphs to (directed) graphs. To this end, we present the notions of *strong component tree* (Sec. 3.2) and of *directed component hierarchy* (Sec. 3.3) that encode the inclusion relations of the S-components and D-components of all level sets. The main result of this section states that the directed component hierarchy can be represented as an enriched version of the strong component tree, that will be further used to define D-connected operators and efficient algorithms to compute them. It is also observed that the directed component hierarchy generalizes the tree structures involved in connected operator definition (Sec. 3.4).

3.1 Stack of graphs

In the framework of undirected graphs, connected operators and component trees have been proposed for the two possible families of weights: those on the vertices and those on the edges. In the first case, a level set is a subset of vertices. In the second case, a level set is made of edges, and one considers the subgraphs induced by these edges to obtain connected components. In both cases, the connected components are defined within a series of nested subgraphs induced by the level sets. In order to handle these two cases in a unified and more general setting, one may consider – instead of weights on either edges or vertices – a series of nested subgraphs. Following this approach, we start this section by presenting the notion of a *stack of graphs*.

Definition 4: A *stack (of graphs)* is a finite sequence $\mathcal{S} = (\mathfrak{G}_0, \ldots, \mathfrak{G}_\ell)$ of graphs such that, for any i in $\{1, \ldots, \ell\}$, the graph $\mathfrak{G}_i$ is a subgraph of $\mathfrak{G}_{i-1}$. For any i in $\{0, \ldots, \ell\}$, we say that $\mathfrak{G}_i$ is a level set of $\mathcal{S}$ (at altitude i). A *S- (resp. D-) component of* $\mathcal{S}$ is a pair (i, X) such that X is a S- (resp. D-) component of the level set of $\mathcal{S}$ at altitude i. The set of all S- (resp. D-) components of the stack $\mathcal{S}$ is denoted by $\mathrm{SCC}_{\mathcal{S}}$ (resp. $\mathrm{DCC}_{\mathcal{S}}$). The stack $\mathcal{S}$ is *connected* whenever $\mathfrak{G}_0$ is strongly connected.

Fig. 5 first row shows a connected stack composed of five graphs $(\mathfrak{G}_0, \ldots, \mathfrak{G}_4)$. In the following, without loss of generality, we assume that the weights of the graphs are positive integers with a maximal value ℓ.

When the domain of an image is considered as the vertex set of a graph $\mathfrak{G}$, i.e., when the vertices correspond to pixels, the image itself directly leads to a stack of graphs: each level set $\mathfrak{G}_i$ (resp. $\mathfrak{G}_{\ell-i}$) is the subgraph induced by the pixels whose value is greater (resp. lower) than i (and whose arc set contains any arc of $\mathfrak{G}$ that links two pixels of value greater (resp. lower) than i). In this case, the obtained stack is said to be *upper-* (resp. *lower-*)*induced by the image*.

For image segmentation tasks, one may also consider similarity measures between pixels that are linked by an arc (for instance, derived from a gradient. Examples of such measures for undirected graphs can be found in [46], [47], [48], [49], [50]). This measure is a function that weights the arcs of the graph $\mathfrak{G}$. Such an arc-weighted graph also leads to a stack of graphs: each level set $\mathfrak{G}_i$ (resp. $\mathfrak{G}_{\ell-i}$) is the subgraph induced by the arcs of weight greater (resp. lower) than i (and whose vertex set contains any pixel of $\mathfrak{G}$ linked by one of these arcs). Such a stack is said to be *upper-* (resp. *lower-*)*induced by the similarity measure*. For segmentation methods based on hierarchies of partitions [13], [38], one may want to ensure that all levels in the graph stack remain a partition of the domain. This can ease further segmentation methods to produce partitions as shown in [18]. A stack obtained by this process is said to be *completed*.

Important notation. In the remaining part of this section, $\mathcal{S} = \{\mathfrak{G}_0, \ldots, \mathfrak{G}_\ell\}$ denotes a connected stack.

3.2 Strong component tree

Let X be a S-component of $\mathfrak{G}_i$, for i in $\{1, \ldots, \ell\}$. Since $\mathfrak{G}_i$ is a subgraph of $\mathfrak{G}_{i-1}$, X is strongly connected in $\mathfrak{G}_{i-1}$. As the S-components of a graph partition its vertex set, the S-component X of $\mathfrak{G}_i$ is included in a unique S-component of $\mathfrak{G}_{i-1}$. This unique S-component of $\mathfrak{G}_{i-1}$ that includes X is denoted by $\mathrm{PAR}_{i-1}(X)$ and is called the $(i-1)$*-parent of* X *(in* $\mathcal{S}$*)*. We also say that the S-component $(i-1, \mathrm{PAR}_{i-1}(X))$ of $\mathcal{S}$ is the *parent* of the S-component (i, X). The set of all S-components of $\mathcal{S}$ equipped with the parent relation is a tree called the *strong component* (or *S-component*) *tree of* $\mathcal{S}$.

Following the usual terminology on trees, given two S-components (i, X) and (j, Y) of the stack $\mathcal{S}$, we say that (j, Y) is an *ancestor of* (i, X) and that (i, X) is a *descendant of* (j, Y) if there exists a sequence $(C_0, \ldots, C_n)$ of S-components of $\mathcal{S}$ such that $C_0 = (i, X)$, $C_n = (j, Y)$, and C_k is the parent of C_{k-1} for any $k \in \{1, \ldots, n\}$. For instance, Fig. 6(a) shows the S-component tree of the stack of Fig. 5.

3.3 Directed component hierarchy

Since distinct D-components of the same graph can be linked by inclusion (see Sec. 2.2), it can be seen that for a given i in $\{1, \ldots, \ell\}$, a D-component of $\mathfrak{G}_i$ can be included in several D-components of $\mathfrak{G}_{i-1}$. Therefore, contrarily to the case of S-components, the inclusion relations between D-components of successive level sets cannot be directly used for organizing the D-components in a tree structure. Actually, as we will see later in this section, the D-components can be arranged as a DAG that is sufficient to recover the inclusion relationship between any two D-component. Furthermore, due to the bijection between S-components and D-components (see Th. 3), this DAG corresponds to an enriched version of the S-component tree. This structure leads to efficient methods, that will be described in Secs. 4, 5, for designing D-connected operators. The next theorem is the key result for establishing the properties of this fundamental DAG.

Theorem 5: Let X and Y be two D-components of $\mathfrak{G}_i$ and $\mathfrak{G}_{i-1}$, respectively, with i in $\{1, \ldots, \ell\}$. The D-component X is a subset of the D-component Y if and only if the $(i-1)$-parent of the root of X is a successor of the root of Y in the DAG of S-components of $\mathfrak{G}_i$

$$X \subseteq Y \Longleftrightarrow \mathrm{PAR}_{i-1}(B^{-1}_{\mathfrak{G}_i}(X)) \in \mathrm{DCC}_{\mathfrak{D}(\mathfrak{G}_{i-1})}(B^{-1}_{\mathfrak{G}_{i-1}}(Y)) \tag{6}$$

More generally, a D-component X of $\mathfrak{G}_i$ is a subset of a D-component Y of $\mathfrak{G}_j$ (with $i \geq j$) if and only if the intersection between the ancestors of the root of X and the successors of the root of Y in $\mathfrak{D}(\mathfrak{G}_j)$ is nonempty. In other words, the set of DAGs of the S-components of all level sets, paired to the parent relation allows us to test the inclusion of any D-components belonging to the stack $\mathcal{S}$.

Definition 6: The *D-component hierarchy of* $\mathcal{S}$ is the graph whose nodes are the S-components of $\mathcal{S}$ and such that there is an arc from a S-component (j, Y) of $\mathcal{S}$ to a S-component (i, X) of $\mathcal{S}$ if

- (j, Y) is the parent of (i, X); or
- $j = i$ and (Y, X) is an arc of the DAG $\mathfrak{D}(\mathfrak{G}_i)$ of S-components of $\mathfrak{G}_i$.

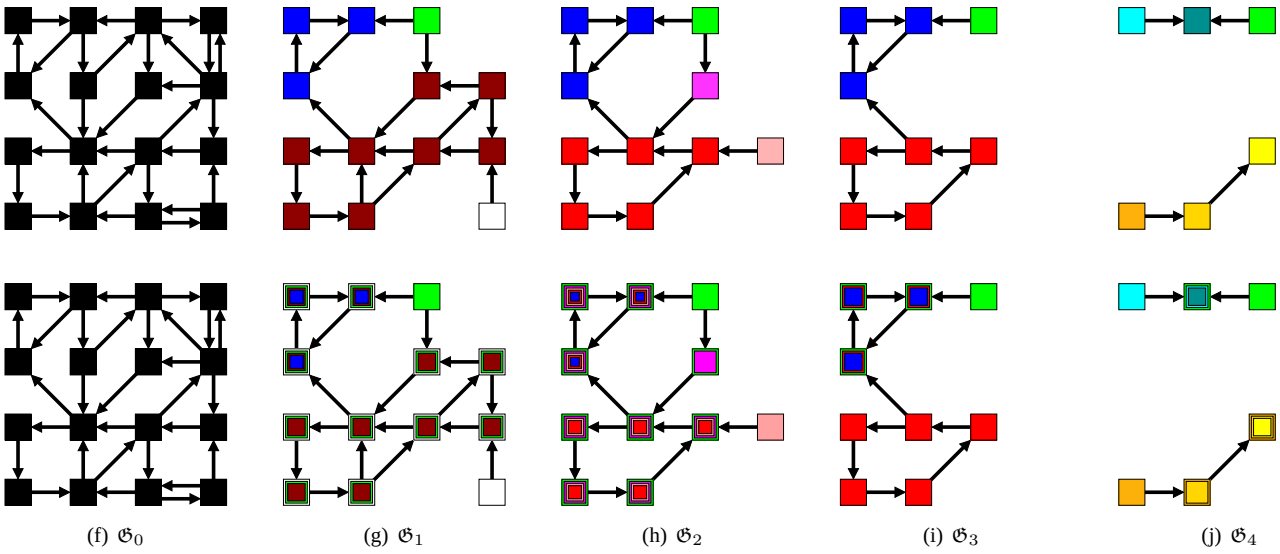
For instance, the D-components of the stack in Fig. 5 are depicted in the second row of Fig. 5. The associated D-component hierarchy is depicted in Fig. 6(c).

As a corollary of Th. 5, there is an isomorphism between the order induced by the D-component hierarchy of the stack $\mathcal{S}$ and the partial order on the D-components of $\mathcal{S}$ such that $(i, X) \sqsubseteq (j, Y)$ if $i \leq j$ and $X \subseteq Y$. In particular, the S-component (j, Y) is the parent of (i, X) if and only if $B_{\mathfrak{G}_j}(Y)$ is the minimal element (for the inclusion relation) among all the D-components of $\mathfrak{G}_j$ that include the D-component $B_{\mathfrak{G}_i}(X)$. A direct consequence of this isomorphism is that the D-component hierarchy of $\mathcal{S}$ is a DAG. In particular, it can be seen that two S-components at the same level set cannot be linked by a cycle since the DAG of S-components of a graph is acyclic. It can also be seen that two S-components of two distinct level sets cannot be linked by a cycle either since a S-component of a given level set cannot be both an ancestor and a descendant of a S-component of another level set.

3.4 Generalization of tree structures

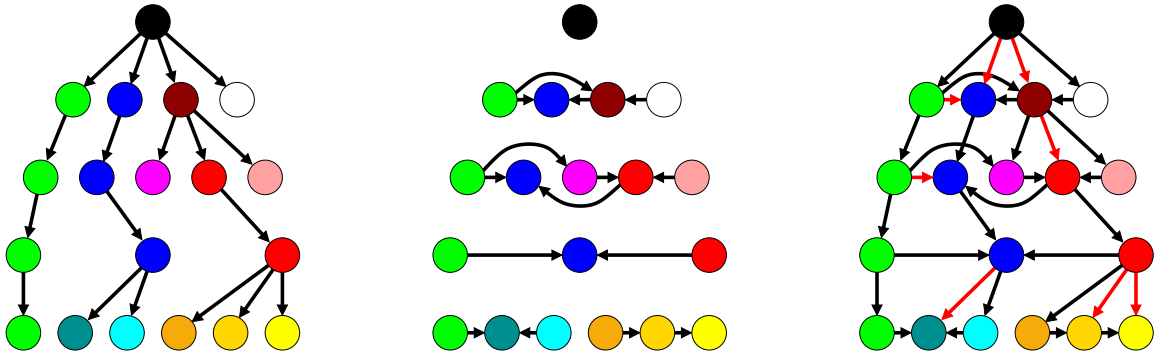
The framework presented in this section for handling the components of a stack of graphs generalizes the handling of connected components via component trees, in both edge- and vertex-weighted undirected graphs.

Indeed, it can be seen that if a graph is symmetric, then a set of vertices is a D-component if and only if it is a connected component in the associated undirected graph. Furthermore, such a set is a D-component if and only if it is a S-component. Hence, in the case of a stack whose level sets are all symmetric graphs, the D-component hierarchy and the S-component tree are indeed the same. Moreover, if a stack is upper (resp. lower) induced by an image, then its D-component hierarchy is also the max- (resp. min-) tree of that image. If a stack is upper (resp. lower) induced by an arc similarity measure, then its D-component hierarchy is the max- (resp. min-) tree of the associated undirected edge-weighted graph. In this last case, if the stack is furthermore completed, then the D-component



(f) $\mathfrak{G}_0$ (g) $\mathfrak{G}_1$ (h) $\mathfrak{G}_2$ (i) $\mathfrak{G}_3$ (j) $\mathfrak{G}_4$

Fig. 5. A stack $\mathcal{S} = (\mathfrak{G}_0, \mathfrak{G}_1, \mathfrak{G}_2, \mathfrak{G}_3, \mathfrak{G}_4)$. First row: Each color represents a S-component. Second row: Each color represents a D-component (vertices with more than one color belong to all the associated D-components).



(a) S-component tree. (b) DAGs of S-components. (c) D-component hierarchy.

Fig. 6. (a) The S-component tree associated to the stack $\mathcal{S}$ of Fig. 5's first row. (b) The DAGs shown from top to bottom row are the DAGs of S-components of the graphs $\mathfrak{G}_0, \ldots, \mathfrak{G}_4$ of Fig. 5's first row. (c) The D-component hierarchy of the stack $\mathcal{S}$ of Fig. 5's second row. This hierarchy is the S-component tree (a) enriched by the relation provided by the DAGs of S-components of all level sets of $\mathcal{S}$ (b). The red arrows correspond to the extra links that can also be deduced by transitivity.

tree is exactly the partition tree [18] (also known as the quasi flat zones hierarchy [38], [51], [52] or α-tree [53]) of the image. As shown in [18], completed stacks also allow us to retrieve the binary partition trees [13] and hierarchical minimum spanning forests or watersheds [54] [11, Ch. 9] [55].

4 Building D-component hierarchies

In this section, we describe how to build the D-component hierarchy of a stack of graphs $\mathcal{S} = \{\mathfrak{G}_0, \ldots, \mathfrak{G}_\ell\}$ (Sec. 4.1), and we discuss the computational cost of this process (Sec. 4.2).

4.1 Algorithm

For the sake of concision, we assume here that the stack $\mathcal{S}$ is constructed from a vertex-weighted graph $\mathfrak{G} = (V, \mathcal{A})$ (see Sec. 3.1). We also assume that graphs are represented by adjacency lists: for each vertex x of V, we store the list of vertices y of V adjacent to x (i.e., such that (x, y) is in $\mathcal{A}$). This representation allows us to access to the list of vertices adjacent to a given vertex in constant time.

The overall construction procedure is described in Alg. 1. Its results consist of: a labeling of each level of the stack $\mathcal{S}$ into S-components ($Label_i$), the adjacency lists of the DAGs of S-components at each level of the stack (Suc_i), and the parent relation between the S-components of successive levels of the stack (PAR_i).

For each level i of the stack, the algorithm consists of three steps: (1) label the vertices of $\mathfrak{G}_i$ into S-components; (2) construct the DAG of S-components of $\mathfrak{G}_i$, i.e., the adjacency lists representing the DAG; and (3) define the parent relation between these S-components and those at altitude $i + 1$.

Step (1) is carried out by one of the Tarjan [56] or Kosaraju-Sharir [57] algorithms, which both produce a labeling in S-components of the vertices of a directed graph in linear

$\mathcal{O}(|V| + |\mathcal{A}|)$ time. We assume that the labels are integers and that the labels at the different levels are all distinct (i.e., $Label_i \cap Label_j = \emptyset$ for $i \neq j$); so they can be used as array indices. For the sake of readability, we consider that the result $Label_i$ is at the same time the set of labels of $\mathfrak{D}(\mathfrak{G}_i)$, denoted by $Label_i$, and the map that associates to each vertex x of $V(\mathfrak{G}_i)$ its label, denoted by $Label_i[x]$.

Algorithm 1: D-component hierarchy construction.

Input: $\mathcal{S} = \{\mathfrak{G}_0, \ldots, \mathfrak{G}_\ell\}$, a stack of graphs.
Output: $Label_i$, S-component labeling for each i in $\{0, \ldots, \ell\}$.
Output: Suc_i, array of adjacency lists for each i in $\{0, \ldots, \ell\}$.
Output: PAR_i, parent relation for each i in $\{0, \ldots, \ell\}$.

```
1  for i from ℓ to 0 do
2      Label_i ← S-component labeling(G_i)
3      Suc_i ← adjacency lists(G_i, Label_i)
4      if i ≠ ℓ then
5          PAR_{i+1} ← parent relation(G_{i+1}, Label_{i+1}, Label_i)
```

Algorithm 2: Adjacency lists construction.

Input: $\mathfrak{G}_i$, a graph.
Input: $Label_i$, S-component labeling of $\mathfrak{G}_i$.
Output: Suc_i, array of adjacency lists of $\mathfrak{G}_i$.

```
1  foreach v ∈ Label_i do
2      Suc_i[v] ← ∅
3      SCC[v] ← ∅
4      Flag[v] ← undefined
5  foreach x ∈ V(G_i) do
6      SCC[Label_i[x]] ← SCC[Label_i[x]] ∪ {x}
7  foreach v ∈ Label_i do
8      foreach x ∈ SCC[v] do
9          foreach (x, y) ∈ A(G_i) do
10             v' ← Label_i[y]
11             if v' ≠ v and Flag[v'] ≠ v then
12                 Suc_i[v] ← Suc_i[v] ∪ {Label_i[y]}
13                 Flag[y] ← v
```

Algorithm 3: Parent relation definition.

Input: $\mathfrak{G}_{i+1}$, a directed graph.
Input: $Label_{i+1}$, a labeling of the S-components of $\mathfrak{G}_{i+1}$.
Input: $Label_i$, a labeling of the S-components of $\mathfrak{G}_i$.
Output: PAR_{i+1}, parent relation on $Label_{i+1}$ and $Label_i$.

```
1  foreach x ∈ V(G_{i+1}) do
2      PAR[Label_{i+1}[x]] ← Label_i[x]
```

Step (2) is performed by Alg. 2. It produces the adjacency list of each vertex of the DAG of the S-components of the level set of $\mathfrak{G}$ at altitude i. To this end, it successively scans each S-component of $\mathfrak{G}_i$. For any scanned S-component $SCC[v]$ of label v, the adjacent vertices of all the vertices in $SCC[v]$ are considered. If one of these adjacent vertices belongs to another S-component $SCC[v']$ of label v' (i.e., if $v' \neq v$, line 11), and if $SCC[v']$ has not yet been reached from $SCC[v]$ (i.e., if $Flag[v'] \neq v$, line 11), then the label v' is added to the adjacency list of v (line 12), and v' is flagged as having been reached from v (line 13). The two outer loops visit each vertex once, and for each vertex, its adjacency list is scanned. The algorithm can thus be run in linear $\mathcal{O}(|V| + |\mathcal{A}|)$ time.

Step (3) is performed by Alg. 3. The algorithm produces an array such that, for every label v of the S-component labeling at level $i+1$, the element of index v in the array is the label of the S-component in the level i that includes the S-component v. The algorithm loops through each vertex x in $V(\mathfrak{G}_{i+1})$ of the previous level $i+1$, and finds the labels $Label_{i+1}[x]$ and $Label_i[x]$ of its associated S-components at levels $i+1$ and i. It then defines the label $Label_i[x]$ as the parent of the label $Label_{i+1}[x]$. In order to achieve a linear $\mathcal{O}(|V|)$ complexity, the labels have to be determined in constant time which is done by storing them for each vertex during the S-component labeling.

4.2 Complexity analysis

The time complexity of Alg. 1 is $\mathcal{O}(\ell.(|V|+|\mathcal{A}|))$. In particular the algorithm is efficient if ℓ is small, for instance in the case of 8-bit images. As there are at most $|V|$ levels (each vertex can have a different weight), this complexity is actually $\mathcal{O}(|V|(|V|+|\mathcal{A}|))$. Moreover, if we do not restrict ourselves to vertex-weighted graphs, there are at most $|V|+|\mathcal{A}|$ levels (each vertex and arc is added one after the other, in the worst case), which leads to a complexity of $\mathcal{O}((|V|+|\mathcal{A}|)^2)$. However, one may notice that we generally have $\ell \ll |V| + |\mathcal{A}|$.

The algorithm can be improved by observing that a S-component at level i is strongly connected at level $i-1$. Based on this fact, we can use a more complex algorithm to build the subgraph at level i. Instead of considering the graph induced by the vertices of weights higher than i, we can add to the DAG of S-components at level $i-1$, the vertices and the arcs that appear at level i. This requires the use of a union-find structure to dynamically manage the S-components. Indeed, it generates an additional cost leading to a $O(\alpha(|V|).(|V| + |\mathcal{A}|)^2)$ worst case complexity, where α is the – extremely slowly growing – inverse of the Ackermann function. Practically, it has been experimentally observed that the improved algorithm is about six times faster than the basic one.

5 Node selection in D-component hierarchies

Similarly to connected operators, D-connected operators consist of processing a hierarchical data structure, namely the D-component hierarchy. This processing requires to select or discard nodes according to criteria that are specifically defined according to the considered application (Sec. 5.1). The selected nodes can then be used, e.g., to obtain a segmentation or to filter an image. Some applications will be described in Sec. 6. Since D-component hierarchies are not tree structures, D-connected operators are more difficult to develop than classically connected ones (Sec. 5.2). In particular, they require specific regularization strategies (Sec. 5.3).

5.1 Node selection criteria

A node selection criterion σ is a mapping that associates a Boolean value to each node/component of a hierarchical data structure. Given a component C, we say that the criterion σ *holds true* (resp. *false*) for C, or that C *satisfies* (resp. *violates*) σ if $\sigma(C)$ equals true (resp. false).

A classical example of a criterion σ_A is one that discards small nodes, often associated to noise, which is defined by

$$\sigma_A(C) = \begin{cases} \text{true} & \text{if Area}(C) > t \\ \text{false} & \text{otherwise} \end{cases} \quad (7)$$

where Area is a measure of the area of the component, for example the number of vertices in C, while t is the area threshold value.

Many other criteria have been proposed in the literature. Most are obtained by replacing Area by some other attribute in Eq. (7). Proposed attributes focus on different aspects: (1) the shape of the component, e.g., the geometrical moments or the compactness; (2) the gray level content of the component, e.g., the volume or the entropy; (3) the topology of the hierarchy, e.g., the number of children of the component; or (4) combination of the previous types of attributes, e.g., the dynamic or the Mumford-Shah energy. It is also possible to replace a constant threshold by a more complex process in the definition of the criterion, e.g., criteria based on energy minimization (Viterbi in [12]) or on shape-space filtering [42].

5.2 The case of D-connected operators

Thinking in terms of D-connected operators, one may desire to mark each D-component as selected or discarded. However – in contrast to the case of connected operators – we may fall into situations such as the one previously depicted in Fig. 4(b), where two D-components overlap. This creates an ambiguous situation whenever only one of them is selected, while the other is discarded. It is not obvious to determine how to proceed with the S-components that correspond to overlapping D-components.

A first partial answer to this question consists of considering the criterion on the S-components instead of the D-components. On the one hand, this choice better suits the data structure constructed in Sec. 4. On the other hand, due to the bijection between D-components and S-components (Th. 3), this strategy is information lossless.

In the following, we consider a stack $\mathcal{S} = \{\mathfrak{G}_0, \ldots, \mathfrak{G}_\ell\}$ of graphs. We also consider, for any i in $\{0, \ldots, \ell\}$, the *DAG at altitude i*, denoted by H_i, defined as the subgraph of the D-component hierarchy of S induced by the S-components of $\mathcal{S}$ at altitude i (i.e., induced by the set of components $\{(i, C) \in \mathrm{SCC}_{\mathcal{S}}\}$). Observe that there is an arc in H_i from the component (i, C_1) to the component (i, C_2) whenever there is an arc from C_1 to C_2 in the dag $\mathfrak{D}(\mathfrak{G}_i)$ of S-components of $\mathfrak{G}_i$. Therefore, in the following and when no confusion may occur, if $C = (i, C')$ is an S-component of $\mathcal{S}$, we use the symbol C, instead of C', for the associated strong component C' of $\mathfrak{G}_i$.

For simplicity – but without loss of generality – we consider the example of the graph H_i, depicted in the first row of Fig. 7, such that $V(H_i)$ is $\{A, B, C\}$ and $\mathcal{A}(H_i)$ is $\{(A, B), (B, C)\}$.

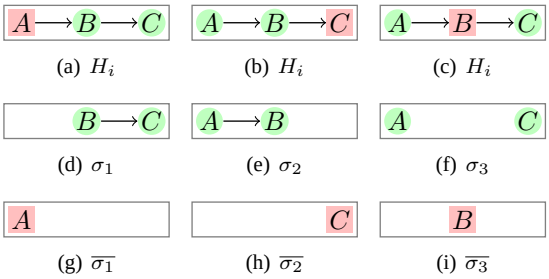


Fig. 7. First row: a graph (DAG) H_i, composed of three nodes that are selected (in green) or discarded (in red) with respect to some criterion σ_j (see Sec. 5.2). Each node corresponds to a S-component (A, B and C, respectively), and models a D-component ($A \cup B \cup C$, $B \cup C$ and C, respectively). Second row: subgraphs induced by the nodes that satisfy the criterion σ_j (represented by green circles). Third row: subgraphs induced by the nodes that violate the criterion σ_j (represented by red squares).

Each column of Fig. 7 corresponds to a different criterion: σ_1 holds true for B, C, and holds false for A; σ_2 holds true for A, B, and holds false for C; and σ_3 holds true for A, C, and holds false for B. The second line shows the graphs associated to each criterion, i.e., the subgraphs induced by the nodes that satisfy the criterion. The third line shows the graphs induced by the negation of the criterion, i.e., the graphs induced by the nodes that violate the criterion. For the first criterion, it can be seen that the D-components of the graph induced by σ_1 are also D-components of H_i, but the D-component of the graph induced by $\overline{\sigma_1}$ is not a D-component of H_i. The converse situation appears for the second criterion: the D-component of the graph induced by $\overline{\sigma_2}$ is also a D-component of H_i, but the D-components of the graph induced by σ_2 are not. Finally, with the third criterion, neither the D-components of the graphs induced by σ_3 nor by $\overline{\sigma_3}$ are D-components of H_i.

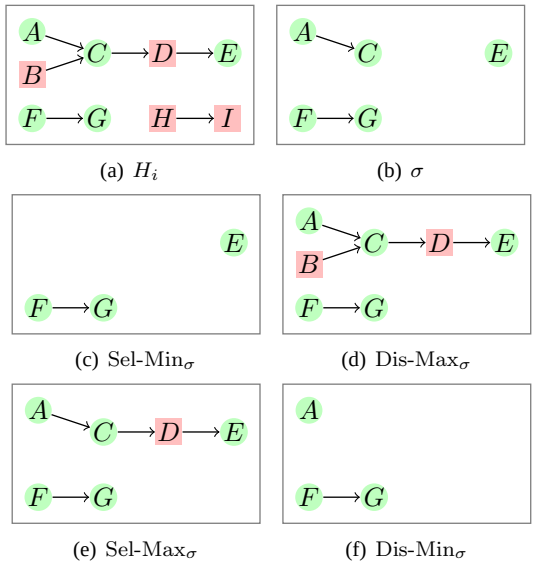
Thus, we identify two desirable, yet generally exclusive, properties. Given a criterion σ we say that:

- σ is *selective* if the D-components of the graph induced by σ on H_i are also D-components of H_i;
- σ is *discarding* if the D-components of the graph induced by $\overline{\sigma}$ on H_i are also D-components of H_i.

Furthermore, we say that a D-component C of H_i is *selected* (resp. *discarded*) by σ if C is also a D-component of the graph induced by σ (resp. $\overline{\sigma}$). Thus, C is selected (resp. discarded) if the criterion σ holds true (resp. false) for every S-component contained in C. Nevertheless, we have seen in Fig. 7 that in general, a criterion σ is neither selective nor discarding. Consequently, we propose several regularization strategies that allow us to propose a selective or a discarding criterion from any criterion σ.

5.3 Regularization strategies

Given a criterion σ, we propose four different regularized criteria of σ. Two of them are selective, namely Sel-Min_σ and Sel-Max_σ. The other two are discarding, namely Dis-Min_σ and Dis-Max_σ. Fig. 8(a) shows a graph H_i, while Fig. 8(b)



(a) H_i (b) σ (c) Sel-Min$_\sigma$ (d) Dis-Max$_\sigma$ (e) Sel-Max$_\sigma$ (f) Dis-Min$_\sigma$

Fig. 8. Node selection (regularization) strategies. (a) The DAG H_i of the S-components $\{A, B, C, D, E, F, G, H, I\}$. We assume that the components represented as green circles (resp. red squares) satisfy (resp. violate) the given criterion σ. (b–f) The graphs induced by the criterion σ, Sel-Min$_\sigma$, Dis-Max$_\sigma$, Sel-Max$_\sigma$, and Dis-Min$_\sigma$.

shows the result of a non-selective and non-discarding criterion σ on H_i. Then, we have the following regularized criteria:

- Sel-Min$_\sigma$ selects the D-components such that every contained S-component satisfies the criterion σ (Fig. 8(c)).
- Dis-Max$_\sigma$ discards the D-components such that every contained S-component violates the criterion σ (Fig. 8(d)).
- Sel-Max$_\sigma$ selects the D-components whose root S-component satisfies the criterion σ (Fig. 8(e)).
- Dis-Min$_\sigma$ discards the D-components whose root S-component violates the criterion σ (Fig. 8(f)).

Thus, for any level i of the hierarchy and for any component C of the DAG H_i at altitude i, we have

$$\text{Sel-Min}_\sigma(C) = \bigwedge_{C' \in \text{DCC}_{H_i}(C)} \sigma(C') \tag{8}$$

$$\text{Dis-Max}_\sigma(C) = \bigvee_{C' \in \text{DCC}_{H_i}(C)} \sigma(C') \tag{9}$$

$$\text{Sel-Max}_\sigma(C) = \bigvee_{C \in \text{DCC}_{H_i}(C')} \sigma(C') \tag{10}$$

$$\text{Dis-Min}_\sigma(C) = \bigwedge_{C \in \text{DCC}_{H_i}(C')} \sigma(C') \tag{11}$$

where $\bigwedge$ and $\bigvee$ are the Boolean "and" and "or" operators.

Property 7: Let σ be a criterion on the D-component hierarchy of the stack $\mathcal{S} = \{\mathfrak{G}_0, \ldots, \mathfrak{G}_\ell\}$. The regularized criterion Sel-Max$_\sigma$ (resp. Dis-Min$_\sigma$) with respect to $\mathcal{S}$ is the same as the regularized criterion Dis-Max$_\sigma$ (resp. Sel-Min$_\sigma$) with respect to the transpose stack $-\mathcal{S} = \{-\mathfrak{G}_0, \ldots, -\mathfrak{G}_\ell\}$, where $-\mathfrak{G}_i$ is the transpose of the graph $\mathfrak{G}_i$. More precisely, for any DAG H_i at altitude i of $\mathcal{S}$ and for any component C of H_i, we have:

$$\bigvee_{C \in \text{DCC}_{H_i}(C')} \sigma(C') = \bigvee_{C' \in \text{DCC}_{-H_i}(C)} \sigma(C') \tag{12}$$

$$\bigwedge_{C \in \text{DCC}_{H_i}(C')} \sigma(C') = \bigwedge_{C' \in \text{DCC}_{-H_i}(C)} \sigma(C') \tag{13}$$

Remark 8: The simplest criteria – that include in particular the criterion σ_A – are those that are *increasing*. We say that a criterion σ is increasing if, for any two S-components C and C' such that the D-component rooted in C is included in the D-component rooted in C', $\sigma(C) =$ true implies that $\sigma(C') =$ true. So, given an increasing criterion σ and a S-component C, if σ holds true for C, we immediately know that all the predecessors of C also satisfy σ. Conversely, if σ holds false for C, we immediately know that all the successors of C violate σ, or, in other words, all the S-components contained in the D-component of root C violate the criterion. Thus, any increasing criterion σ is discarding. We then have σ equals to Dis-Min$_\sigma$ and to Dis-Max$_\sigma$.

The previous discussions focused on node selection for a single level of the hierarchy. Nevertheless, a similar challenge exists in order to ensure result consistency between the different levels of the hierarchy, i.e., in order to avoid "holes" between two or more levels.

The previously defined regularization rules can also be used on the S-component tree, by considering the ancestors (resp. descendants) instead of the the predecessors (resp. successors) These new – but similar – hierarchical criteria are denoted Sel-Max-H, Sel-Min-H, Dis-Max-H, and Dis-Min-H:

- Sel-Max-H$_\sigma$ holds true for all the descendants of a node that satisfies σ.
- Sel-Min-H$_\sigma$ holds false for all the ancestors of a node that violates σ.
- Dis-Min-H$_\sigma$ holds false for the descendants of a node that violates σ.
- Dis-Max-H$_\sigma$ holds true for all the ancestors of a node that satisfies σ.

One may notice that the Dis-Min-H$_\sigma$ (resp. Dis-Max-H$_\sigma$) strategy is indeed the analogue of the usual min (resp. max) filtering rules of the classical component trees [12].

Remark 9: The definition of an increasing criterion is consistent with the parent relation. In other words, given a criterion σ and two S-components C and C' such that C' is a descendant of C' (i.e., $C' \subseteq C$), the D-component rooted in C' is included in the D-component rooted in C, and thus σ is increasing if $\sigma(C') =$ true implies $\sigma(C) =$ true (which is the usual definition of an increasing criterion for the classical component tree). In this context, for an increasing criterion σ, all the proposed regularization strategies of σ yield the same result as σ due to the tree structure of the S-components at the different levels (by opposition to the DAG of the S-components at a single level).

6 Illustrations

The goal of this section is to illustrate the versatility of the proposed framework to solve practical image processing

problems. To this end, two applications are presented. We first show that the proposed framework can be used to design a filtering procedure adapted to the processing of neurite images (Sec. 6.1). Then, we propose a marker-based segmentation procedure that uses a D-component hierarchy to take into account a preliminary classification of the image pixels. Results of this procedure are shown on a cardiac image (Sec. 6.2).

6.1 Image filtering

The D-component hierarchy can be used to perform image filtering, i.e., to obtain a new image based on the node selection procedure (see Sec. 5.1). This requires reconstructing the image from the D-component hierarchy.

Given a D-component hierarchy H of a stack $S = (\mathfrak{G}_0, \ldots, \mathfrak{G}_l)$ and a criterion σ on S, the reconstruction I^H_σ of H for σ is a function that associates to each vertex v of $\mathfrak{G}_0$ the altitude of the smallest node of H that holds true for σ and that contains v. Formally, for all v in $\mathfrak{G}_0$, we have

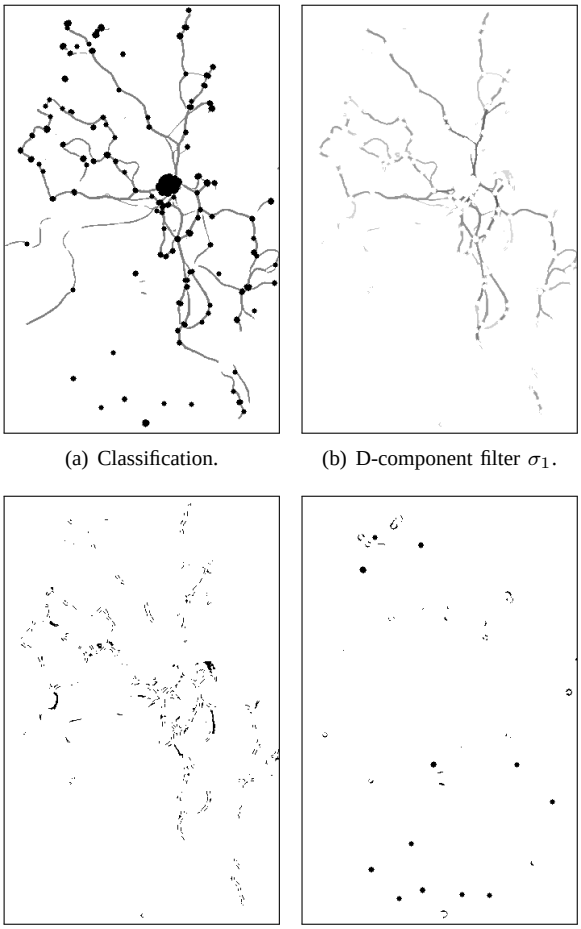
$$I^H_\sigma(v) = \max\{i \in [\![0, \ell]\!] \mid (i, C) \in V(H),\ v \in C,\ \sigma((i, C)) = \text{true}\} \quad (14)$$

The most natural way to construct image filters is thus to obtain a stack from the original image considered as a vertex-weighted graph. In this case, the altitudes in the hierarchy correspond to gray levels. Also note that the reconstruction process (Eq. (14)) does not take account of D-components, and it is thus important to use a regularized criterion (Sec. 5.3) in order to keep or remove D-components during the filtering process. One image filtering strategy using the D-component hierarchy is illustrated in neurite image filtering.

6.1.1 Neurite filtering

In this example, we consider a sample image of a neuron grown in vitro (Fig. 1(a)), with associated neurites (i.e., its axon and dendrites). The objective is to derive measures of neurite tree complexity, which are useful in various toxicology assays, called neurite outgrowth assays [58]. Whereas neurites do form a tree, apparent overlap in vitro complexify the layout. The challenges of such images are low contrast of neurite vs. background elements unrelated to neurite structures as well as noise, making it complex to segment based only on an intensity criterion.

In our application, we rely on a vesselness-like local object caracterization [43], which enables us to classify regions into tubes, blobs and background (Fig. 9(a)). This allows us to construct an asymmetric adjacency relation where tubes can be linked to blobs but not the other way around. However, each of the three classes is linked to its own class, while background is linked to all classes. Then, we take into account both intensity and geometrical classification in order to filter the image. By imposing asymmetric blob-to-vessel connection, we exploit the fact that the tube classification is under-segmented and we seek to complete the missing information by searching the connections from the blob and the background classes with this more robust tube class.



(a) Classification. (b) D-component filter σ_1. (c) Added. (d) Removed.

Fig. 9. D-component filter on a neurite image. (a) A classification of Fig. 1(a) into neurite "tubes" (gray), "blobs" (black), and "background" (white) used to generate the directed adjacency. (b) Results of the filtering criterion σ_1. (c) What has been added in (b) compared to (a). (d) What has been removed from (a) in (b).

6.1.2 Filtering criteria

We propose an original criterion σ_1 which relies exclusively on the specific structure of the D-component hierarchy, compared to the classical component tree. The criterion aims to hold true for neurite tubes using the following two heuristics: (1) a tube must be connected at least to two other structures; and (2) a tube is connected at its extremities: the length of the interval between a tube and the structures it is connected to should be small.

Formally, for any level i of the hierarchy and for any component C of the DAG H_i at altitude i, the criterion σ_1 relies on two attributes: the number of adjacent nodes $\text{NAdj}(C)$ of C and the number of arcs $\text{NOut}(C)$ going out of C into an adjacent node. For example, in Fig. 3(a), the S-component $\{a, b, c\}$ has one adjacent S-component $\{d, e\}$, but it has two arcs going out of it, (c, d) and (c, e).

$$\text{NAdj}(C) = |\{C' \in V(H_i) \mid (C, C') \in \mathcal{A}(H_i)\}| \quad (15)$$

$$\text{NOut}(C) = |\{(c, c') \in \mathcal{A}(\mathfrak{G}_i) \mid c \in C, c' \notin C\}| \quad (16)$$

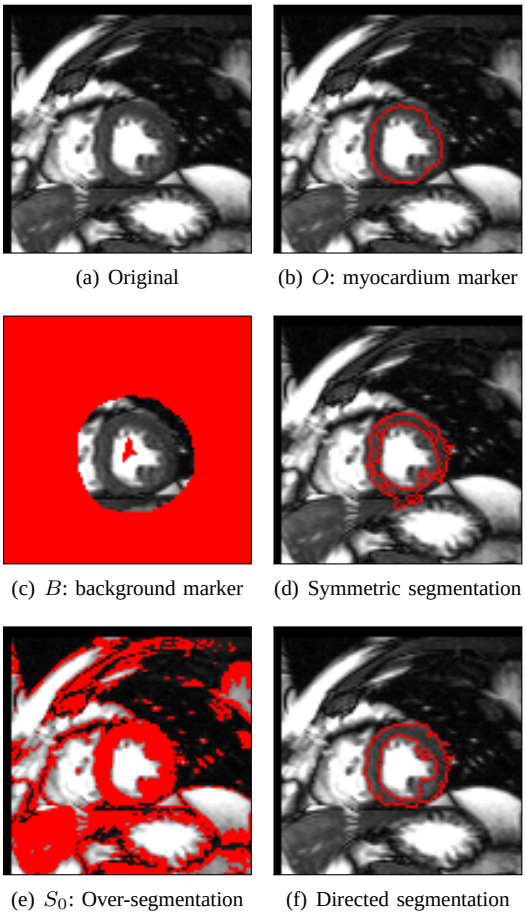
Then, the criterion σ_1 is defined by

$$\sigma_1(C) = \begin{cases} \text{true} & \text{if } \mathrm{NAdj}(C) \geq 2 \text{ and } \mathrm{NOut}(C) \leq 20 \\ \text{false} & \text{otherwise} \end{cases} \quad (17)$$

One can note that the second condition $\mathrm{NOut}(C) \leq 20$ was adjusted empirically. Nevertheless, it roughly corresponds to a frontier of 6 pixels with 8-neighbourhood (3 arcs going out of each pixel on the frontier). More complex measures can be proposed in order to obtain a scale invariance property.

6.1.3 Result

The filtered image $I^H_{\sigma_1}$ of Fig. 1(a) according to σ_1 is represented in Fig. 9(b). It can be seen that the reconstructed structures are mostly thin and elongated as expected. Nevertheless, we still miss a few neurite tubes: this problem can be solved by regularizing the criterion σ_1 (Sec. 5.3). Indeed, the chosen adjacency allows us to say with confidence that whenever the criterion holds true for a node, then its successors (which can only be classified as blobs or tubes by the vesselness) are also part of the neurite. Thus, we want to select the D-components whose root S-component is selected by σ_1 which correspond to the regularized criterion: σ_2 given by $\text{Sel-Max}_{\sigma_1}$. Furthermore, in order to fully reconstruct the brightness of the selected S-components, we can perform a hierarchical regularization, setting the final criterion σ_3 to $\text{Sel-Max-H}_{\sigma_2}$. The filtered image $I^H_{\sigma_3}$ of the image Fig. 1(a) according to σ_3 is represented in Fig. 1(b). Moreover, Fig. 9(c) shows the non null pixels of $I^H_{\sigma_1}$ (Fig. 9(b)) that were classified as background with the vesselness (Fig. 9(a)), i.e., the false negatives. We can observe that the selection process was able to recover several neurite tubes that were classified as background. Conversely, Fig. 9(d) shows the null pixels of $I^H_{\sigma_1}$ (Fig. 9(b)) that were not classified as background with the vesselness (Fig. 9(a)), i.e., the false positives. We can see that isolated structures were correctly identified as non-neurite parts.



(a) Original (b) O: myocardium marker (c) B: background marker (d) Symmetric segmentation (e) S_0: Over-segmentation (f) Directed segmentation

Fig. 10. Illustration of a segmentation procedure based on the D-component hierarchy. In (b,c,e), the considered sets are superimposed in red to the original image whereas in (d,f), the internal border of the segmentation results are superimposed in red to the original image.

6.2 Marker based segmentation

In this section, we illustrate the use of the proposed framework in a marker-based image segmentation procedure. To this end, we consider the magnetic resonance image of the heart shown in Fig. 10(a). From this image, we aim at segmenting the left ventricular myocardium (see the segmented result in Fig. 10(f)) from the two markers of the myocardium and of the background that are shown in red in Figs. 10(b,c). For this illustration, the markers were manually overlaid on the image. However, in a clinical context, these markers may be obtained via an automated procedure (see, e.g., [59]).

As is classically done with graph based segmentation methods, we consider an arc weighted graph $(\mathfrak{G}, w)$ such that the arc weights function w is a dissimilarity measure based on the gradient magnitude of the image. The vertex set of $\mathfrak{G}$ is the domain of the image to be processed (i.e., a rectangular subset of $\mathbb{Z}^2$) and the arc set of $\mathfrak{G}$ is given by the 4-adjacency relation: the pair (x, y) is an arc of $\mathfrak{G}$ if $|x_1 - y_1| + |x_2 - y_2| = 1$, where $x = (x_1, x_2)$ and $y = (y_1, y_2)$. The graph considered in our illustration is symmetric but the weight of an arc (x, y) from a pixel x to a pixel y is not necessarily equal to the weight of the symmetric arc (y, x) from y to x. The setting of the weight map w will be given after the description of the segmentation procedure and a property of its result that guided us for defining the map w.

Given the arc-weighted graph $(\mathfrak{G}, w)$ and two sets of pixels, denoted by O and B, which correspond to the marker of the object (myocardium, see Fig. 10(b)) and of the background (see Fig. 10(c)), respectively, our method consists of two steps:

1) build the D-component hierarchy of the weighted graph $(\mathfrak{G}, w)$ (or more precisely of the completed stack lower induced by the similarity measure w, see Sec. 3.1);
2) select from this hierarchy all D-components rooted in a pixel marked with the label object (i.e., a vertex that belongs to O) and that does not contain any pixel marked as background (i.e., any vertex that belongs to B).

The resulting segmentation, denoted by $S_{O \to B}$, is the union of the selected D-components. More formally, the segmentation is made of the components selected, by the regularization $\text{Sel-Max}_{\sigma_4}$ of a simple criterion σ_4 defined for any level i of the hierarchy and for any component C of the DAG H_i

at altitude i by

$$\sigma_4(C) = \text{true} \Longleftrightarrow C \cap O \neq \emptyset \text{ and } \mathrm{DCC}_{H_i}(C) \cap B = \emptyset \quad (18)$$

Step 1 is performed by the algorithm of Sec. 4, whereas Step 2 is performed by recursive traversals of the hierarchy.

The segmentation result can be characterized owing to a directed version of a measure widely used for segmentation purposes in undirected weighted graphs, namely the connection value [60], [61] (also called degree of connectivity [62] or fuzzy connectedness [46] up to an inversion of w). If $\pi = (x_0, \ldots, x_\ell)$ is a path in $\mathfrak{G}$, the *connection value* $\Upsilon_w(\pi)$ *of* π is the maximum weight of the arcs of π

$$\Upsilon_w(\pi) = \begin{cases} \max\{w(x_i, x_{i-1}) \mid i \in [\![1, \ell]\!]\} & \text{if } \pi \text{ non-trivial} \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

The *(directed) connection value* $\Upsilon_w(x, y)$ *from a vertex* x *of* $\mathfrak{G}$ *to a vertex* y is then defined as the minimum of the connection values of the paths from x to y

$$\Upsilon_w(x, y) = \min\{\Upsilon_w(\pi) \mid \pi \text{ is a path from } x \text{ to } y\} \quad (20)$$

Hence, by the very definition of a D-component hierarchy, it can be shown that the segmentation result $S_{O \to B}$ is the set of all vertices whose directed connection value from a vertex marked as object is less than the one to a vertex marked as background, i.e., for any $x \in V(\mathfrak{G})$, we have

$$x \in S_{O \to B} \Longleftrightarrow \min_{o \in O}\{\Upsilon_w(o, x)\} \leq \min_{b \in B}\{\Upsilon_w(x, b)\} \quad (21)$$

From the previous characterizations of the proposed segmentation method, we can say intuitively that the weights of the arcs should be higher in the border of the object to be segmented than inside and outside this object. In order to produce such a map and, following the work of [5], the weight $w(x, y)$ of the arc (x, y) is the product of a symmetric term $\nabla(x, y)$ and a non necessarily symmetric term $\delta(x, y)$

$$w(x, y) = \nabla(x, y) \times \delta(x, y) \quad (22)$$

The first term $\nabla(x, y)$ plays the role of a gradient magnitude. For our illustration, it is defined as the absolute difference of intensity between the pixels x and y. If I denotes the grayscale image to be segmented we have

$$\nabla(x, y) = |I(x) - I(y)| \quad (23)$$

when no further knowledge on the objects to be segmented exist, symmetric weights can be used. In this case the second term $\delta(x, y)$ is set to a constant value. For instance, the image of Fig. 10(d) is obtained by the proposed method from the markers O and B shown in Figs. 10(b,c) when a constant value is used for the map δ.

However, when domain knowledge is available, the second term $\delta(x, y)$ can be used to ease the connection of a pixel either to the background or to the foreground marker. From Eq. (21), we remark that if the weight of every arc (x, y) ending at a vertex y is set to an infinite value (i.e., a maximal weight), then the vertex y does not belong to the segmented object $S_{O \to B}$. Indeed, in this case, the connection value from any point o of the marker O to the vertex y is maximal and therefore, by Eq. (21), the vertex y does not belong to the segmented object $S_{O \to B}$. On the other hand, when the weight of every arc starting at x is maximal (i.e., in the cases where the connection value from the object marker is not also maximal), the point x generally belongs to the segmented object $S_{O \to B}$. These observations allow us to design a strategy to take into account some prior information in the arcs weights in order to ease the connection of a given point either to the background marker or to object marker. Indeed, if we have good confidence that a pixel x belongs to the object (but not high enough to put it in the object marker), we can increase the weights of the arcs going out from x in order to harden its connection to the background. Conversely, if we are confident that a pixel x belongs to the background, we have to increase the weights of the arcs ending in x.

Following this strategy, we set the term $\delta(x, y)$ in Eq. (22) based on a first classification of the image pixels that produces a lot of false positives but tends to minimize the false negatives. In other words, this classification tends to produce a superset, denoted by S_0, of the expected segmentation result. In our illustration, S_0 is obtained by excluding the extremal intensity values, which corresponds to blood and fat for the brightest pixels and to lungs for the darkest ones(see Fig. 10(e)). Hence, we are confident that the points that do not belong to S_0 do not belong to the object that we seek to segment. Then, the terms $\{\delta(x, y)\}$ are set up in order to ease the connection of the pixels that do not belong to S_0, to the background marker B. This is done by multiplying by a constant value the weight of any arc (x, y) ending at a pixel classified as background

$$\delta(x, y) = \begin{cases} 1 & \text{if } y \notin S_0 \\ K & \text{otherwise} \end{cases} \quad (24)$$

where $K > 1$.

The result of our method (with $K = 1.5$) for the myocardium segmentation is presented in Fig. 10(f).

7 Conclusion

In this article, we have introduced and investigated a notion of directed connectedness. This has led us to the proposal of new (directed) connected operators, no longer based on partition hierarchies organized as trees, but on partition covers organized as DAGs.

From a theoretical viewpoint, we have provided a relevant way to generalize various connected operators, based on tree structures, previously proposed in the literature. This may lead to a better understanding of the common properties between these operators, but also to clarify some subtle differences between those that lie in the framework of directed connected operators, and those that do not, such as hyperconnections. In this context, it is relevant to develop an axiomatization of directed connectedness such as was done in [63], in order to compare it to the axiomatizations already proposed for connections [44, Ch. 2] and hyperconnections [64].

From both the theoretical and algorithmic viewpoints, it will also be useful to compare the links that exist between the DAGs induced by directed connectedness, with other non-tree

structures that have been recently introduced to extend the framework of connected operators, for instance in the case of hypertrees [37] or component graphs [65], [66], that constitute an extension of component trees to multivalued images.

From a methodological point of view, we have shown that the cover hierarchies obtained when considering directed connectedness can be efficiently handled by taking advantage of the intrinsic links that exist between directed connected and strongly connected components, the latter ones being organized as trees. Based on these properties, the complexity of the initial algorithm proposed in this article for building cover hierarchies, can be improved by using the recent incremental algorithm proposed in [67] for building the DAG of strongly connected components in $\mathcal{O}(N^{3/2})$ time complexity. Moreover, beyond the standard attribute-based antiextensive filtering developed in this article, other approaches initially devoted to tree structures can be adapted to the case of directed connected operators, and in particular the optimal tree-cut segmentation paradigms initially proposed in [40], and further formalized in the framework of connected operators [68].

From an applicative viewpoint, we note that the directed connectedness framework is quite versatile and useful for filtering and segmentation tasks. Applications will be more extensively proposed in further works, with complete validation protocols and comparison with similar approaches proposed in the literature, for instance in [5]. Moreover, the image segmentation method proposed in Sec. 6.2 is built upon the classical symmetric 4-adjacency relation on pixels and upon a preliminary classification of the image pixels. Promising perspectives in this direction include considering a directed pixel adjacency graph where each pixel is adjacent to its nearest neighbors in a feature space such as done notably in [69] and considering a fuzzy classification instead of a crisp one.

Source code corresponding to this article is available at the following url: http://www.esiee.fr/~perretb/dc-hierarchy.html.

ACKNOWLEDGMENT

This work was partially funded with French *Agence Nationale de la Recherche* grant agreements ANR-10-BLAN-0205 and ANR-12-MONU-0010.

REFERENCES

[1] A. Rosenfeld, "Connectivity in digital pictures," *J Assoc Comput Mach*, vol. 17, pp. 146–160, 1970.
[2] ——, "Adjacency in digital pictures," *Inform Control*, vol. 26, pp. 24–33, 1974.
[3] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *Int J Comput Vision*, vol. 70, pp. 109–131, 2006.
[4] D. Singaraju, L. Grady, and R. Vidal, "Interactive image segmentation via minimization of quadratic energies on directed graphs," in *CVPR*, 2008.
[5] P. A. V. Miranda and L. A. C. Mansilla, "Oriented image foresting transform segmentation by seed competition," *IEEE T Image Process*, in Press.
[6] O. Tankyevych, H. Talbot, and N. Passat, "Semi-connections and hierarchies," in *ISMM*, ser. Lect Notes Comput Sc, vol. 7883, 2013, pp. 157–168.
[7] H. J. A. M. Heijmans, "Connected morphological operators for binary images," *Comput Vis Image Und*, vol. 73, pp. 99–120, 1999.
[8] P. Salembier and J. Serra, "Flat zones filtering, connected operators, and filters by reconstruction," *IEEE T Image Process*, vol. 4, pp. 1153–1160, 1995.
[9] E. J. Breen and R. Jones, "Attribute openings, thinnings, and granulometries," *Comput Vis Image Und*, vol. 64, pp. 377–389, 1996.
[10] P. Salembier and M. H. F. Wilkinson, "Connected operators: A review of region-based morphological image processing techniques," *IEEE Signal Proc Mag*, vol. 26, pp. 136–157, 2009.
[11] L. Najman and H. Talbot, Eds., *Mathematical Morphology: From Theory to Applications*. ISTE/J. Wiley & Sons, 2010.
[12] P. Salembier, A. Oliveras, and L. Garrido, "Anti-extensive connected operators for image and sequence processing," *IEEE T Image Process*, vol. 7, pp. 555–570, 1998.
[13] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation and information retrieval," *IEEE T Image Process*, vol. 9, pp. 561–576, 2000.
[14] P. Monasse and F. Guichard, "Scale-space from a level lines tree," *J Vis Commun Image R*, vol. 11, pp. 224–236, 2000.
[15] L. Najman and M. Couprie, "Building the component tree in quasi-linear time," *IEEE T Image Process*, vol. 15, pp. 3531–3539, 2006.
[16] T. Géraud, E. Carlinet, S. Crozet, and L. Najman, "A quasi-linear algorithm to compute the tree of shapes of nD images," in *ISMM*, ser. Lect Notes Comput Sc, vol. 7883, 2013, pp. 97–108.
[17] E. Carlinet and T. Géraud, "A comparison of many max-tree computation algorithms," in *ISMM*, ser. Lect Notes Comput Sc, vol. 7883, 2013, pp. 73–84.
[18] J. Cousty, L. Najman, and B. Perret, "Constructive links between some morphological hierarchies on edge-weighted graphs," in *ISMM*, ser. Lect Notes Comput Sc, vol. 7883, 2013, pp. 86–97.
[19] L. Najman, J. Cousty, and B. Perret, "Playing with Kruskal: Algorithms for morphological trees in edge-weighted graphs," in *ISMM*, ser. Lect Notes Comput Sc, vol. 7883, 2013, pp. 135–146.
[20] R. Jones, "Connected filtering and segmentation using component trees," *Comput Vis Image Und*, vol. 75, pp. 215–228, 1999.
[21] M. A. Westenberg, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Volumetric attribute filtering and interactive visualization using the max-tree representation," *IEEE T Image Process*, vol. 16, pp. 2943–2952, 2007.
[22] N. Passat, B. Naegel, F. Rousseau, M. Koob, and J.-L. Dietemann, "Interactive segmentation based on component-trees," *Pattern Recogn*, vol. 44, pp. 2539–2554, 2011.
[23] L. Chen, M. W. Berry, and W. W. Hargrove, "Using dendronal signatures for feature extraction and retrieval," *Int J Imag Syst Tech*, vol. 11, pp. 243–253, 2000.
[24] E. R. Urbach, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images," *IEEE T Pattern Anal*, vol. 29, pp. 272–285, 2007.
[25] J. Mattes, M. Richard, and J. Demongeot, "Tree representation for image matching and object recognition," in *DGCI*, ser. Lect Notes Comput Sc, vol. 1568, 1999, pp. 392–405.
[26] M. H. F. Wilkinson and M. A. Westenberg, "Shape preserving filament enhancement filtering," in *MICCAI*, ser. Lect Notes Comput Sc, vol. 2208, 2001, pp. 770–777.
[27] A. Dufour, O. Tankyevych, B. Naegel, H. Talbot, C. Ronse, J. Baruthio, P. Dokládal, and N. Passat, "Filtering and segmentation of 3D angiographic data: Advances based on mathematical morphology," *Med Image Anal*, vol. 17, pp. 147–164, 2013.
[28] C. Berger, T. Géraud, R. Levillain, N. Widynski, A. Baillard, and E. Bertin, "Effective component tree computation with application to pattern recognition in astronomical imaging," in *ICIP*, 2007, pp. 41–44.
[29] B. Perret, S. Lefèvre, C. Collet, and E. Slezak, "Connected component trees for multivariate image processing and applications in astronomy," in *ICPR*, 2010, pp. 4089–4092.
[30] C. Kurtz, N. Passat, P. Gançarski, and A. Puissant, "Extraction of complex patterns from multiresolution remote sensing images: A hierarchical top-down methodology," *Pattern Recogn*, vol. 45, pp. 685–706, 2012.
[31] A. Alonso-González, S. Valero, J. Chanussot, C. López-Martínez, and P. Salembier, "Processing multidimensional SAR and hyperspectral images with binary partition tree," *P IEEE*, vol. 101, pp. 723–747, 2013.
[32] B. Naegel and L. Wendling, "A document binarization method based on connected operators," *Pattern Recogn Lett*, vol. 31, pp. 1251–1259, 2010.
[33] B. Perret, S. Lefèvre, C. Collet, and E. Slezak, "Hyperconnections and hierarchical representations for grayscale and multiband image processing," *IEEE T Image Process*, vol. 21, pp. 14–27, 2012.
[34] C. Ronse, "Set-theoretical algebraic approaches to connectivity in continuous or digital spaces," *J Math Imaging Vis*, vol. 8, pp. 41–58, 1998.

[35] J. Serra, "Connectivity on complete lattices," *J Math Imaging Vis*, vol. 9, pp. 231–251, 1998.
[36] G. K. Ouzounis and M. H. F. Wilkinson, "Mask-based second-generation connectivity and attribute filters," *IEEE T Pattern Anal*, vol. 29, pp. 990–1004, 2007.
[37] N. Passat and B. Naegel, "Component-hypertrees for image segmentation," in *ISMM*, ser. Lect Notes Comput Sc, vol. 6671, 2011, pp. 284–295.
[38] P. Soille, "Constrained connectivity for hierarchical image partitioning and simplification," *IEEE T Pattern Anal*, vol. 30, pp. 1132–1145, 2008.
[39] G. K. Ouzounis and M. H. F. Wilkinson, "Hyperconnected attribute filters based on k-flat zones," *IEEE T Pattern Anal*, vol. 33, pp. 224–239, 2011.
[40] L. Guigues, J.-P. Cocquerez, and H. Le Men, "Scale-sets image analysis," *Int J Comput Vision*, vol. 68, pp. 289–317, 2006.
[41] J. Serra, "Tutorial on connective morphology," *IEEE J Sel Top Signal*, vol. 6, pp. 739–752, 2012.
[42] Y. Xu, T. Géraud, and L. Najman, "Morphological filtering in shape spaces: Applications using tree-based image representations," in *ICPR*, 2012, pp. 485–488.
[43] A. F. Frangi, W. J. Niessen, R. M. Hoogeveen, T. van Walsum, and M. A. Viergever, "Model-based quantitation of 3-D magnetic resonance angiographic images," *IEEE T on Med Imaging*, vol. 18, no. 10, pp. 946–956, 1999.
[44] J. Serra, Ed., *Image Analysis and Mathematical Morphology, II: Theoretical Advances*. London: Academic Press, 1988.
[45] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, 2001.
[46] J. K. Udupa and S. Samarsekara, "Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," *CVGIP-Graph Model Im*, vol. 58, pp. 246–261, 1996.
[47] I. Bloch, H. Maître, and M. Anvar, "Fuzzy adjacency between image objects," *Int J Uncertain Fuzz*, vol. 5, pp. 615–654, 1997.
[48] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE T Pattern Anal*, vol. 23, pp. 1222–1239, 2001.
[49] A. X. Falcão, J. Stolfi, and R. A. Lotufo, "The image foresting transform: Theory, algorithm and applications," *IEEE T Pattern Anal*, vol. 26, pp. 19–29, 2004.
[50] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Minimum spanning forests and the drop of water principle," *IEEE T Pattern Anal*, vol. 31, pp. 1362–1374, 2009.
[51] M. Nagao, T. Matsuyama, and Y. Ikeda, "Region extraction and shape analysis in aerial photographs," *Comput Vision Graph*, vol. 10, pp. 195–223, 1979.
[52] F. Meyer and P. Maragos, "Morphological scale-space representation with levelings," in *Scale-Space*, ser. Lect Notes Comput Sc, vol. 1682, 1999, pp. 187–198.
[53] G. Ouzounis and P. Soille, "Pattern spectra from partition pyramids and hierarchies," in *ISMM*, ser. Lect Notes Comput Sc, vol. 6671, 2011, pp. 108–119.
[54] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE T Pattern Anal*, vol. 18, pp. 1163–1173, 1996.
[55] J. Cousty and L. Najman, "Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts," in *ISMM*, ser. LNCS, vol. 6671, 2011, pp. 272–283.
[56] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J Comput*, vol. 1, pp. 146–160, 1972.
[57] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Comput Math Appl*, vol. 7, pp. 67–72, 1981.
[58] J. Bilsland, M. Rigby, L. Young, and S. Harper, "A rapid method for semi-quantitative analysis of neurite outgrowth from chick DRG explants using image analysis," *J Neurosci Meth*, vol. 92, pp. 75–85, 1999.
[59] J. Cousty, L. Najman, M. Couprie, S. Clément-Guinaudeau, T. Goissen, and J. Garot, "Segmentation of 4D cardiac MRI: Automated method based on spatio-temporal watershed cuts," *Image Vision Comput*, vol. 28, pp. 1229–1243, 2010.
[60] G. Bertrand, "On topological watersheds," *J Math Imaging Vis*, vol. 22, pp. 217–230, 2005.
[61] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Thinnings, shortest-path forests and topological watersheds," *IEEE T Pattern Anal*, vol. 32, pp. 925–939, 2010.
[62] A. Rosenfeld, "On connectivity properties of grayscale pictures," *Pattern Recogn*, vol. 16, pp. 47–50, 1983.
[63] C. Ronse, "Axiomatics for oriented connectivity," *Pattern Recogn Lett*, submitted.
[64] B. Perret, S. Lefèvre, and C. Collet, "Toward a new axiomatic for hyperconnections," in *ISMM*, ser. LNCS, vol. 6671, 2011, pp. 85–95.
[65] N. Passat and B. Naegel, "Component-trees and multivalued images: Structural properties," *J Math Imaging Vis*, in Press, doi:10.1007/s10851-013-0438-3.
[66] B. Naegel and N. Passat, "Toward connected filtering based on component-graphs," in *ISMM*, ser. Lect Notes Comput Sc, vol. 7883, 2013, pp. 350–361.
[67] B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan, "Incremental cycle detection, topological ordering, and strong component maintenance," *ACM T Algo*, vol. 8, p. 3, 2012.
[68] B. R. Kiran and J. Serra, "Global-local optimizations by hierarchical cuts and climbing energies," *Pattern Recogn*, in Press, doi:10.1016/j.patcog.2013.05.012.
[69] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *Int J Comput Vision*, vol. 59, pp. 167–181, 2004.

Benjamin Perret received his M.Sc. in Computer Science in 2007, and his Ph.D. in Image Processing in 2010 from the Université de Strasbourg (France). He currently holds an assistant professor position at ESIEE Paris, affiliated with the Laboratoire d'informatique Gaspard-Monge, Université Paris-Est. His current research interests include image filtering and segmentation with applications in medical imaging, astronomical imaging and document imaging.

Jean Cousty received his Ingénieur's degree from the Ecole Supérieure d'Ingénieurs en Électrotechnique et Électronique (ESIEE Paris, France) in 2004 and the Ph.D. degree from the Université de Marne-la-Vallée (France) in 2007. After a one-year post-doctoral period in the ASCLEPIOS research team at INRIA (SophiaAntipolis, France), he is now teaching and doing research with the Informatics and Telecom Department, ESIEE Paris, and with the Laboratoire d'informatique Gaspard Monge, Université Paris-Est. His current research interests include image analysis and discrete mathematics.

Olena Tankyevych received her Master's degree from Uppsala University (Sweden) in 2006 and the Ph.D. degree from the Université de Marne-la-Vallée (France) in 2010. After a short post-doctoral period in Telecom ParisTech at the Signal and Image Processing department (Paris, France), she is now teaching and doing research with the Laboratory of Images, Signals and Intelligent Systems at Université Paris-Est Créteil. Her current research interests include medical image analysis and mathematical morphology.

Hugues Talbot graduated from École Centrale de Paris in 1989, obtained the MSc from Université Paris 6 in 1990 and the PhD from École des Mines de Paris in 1993. He was a principal research scientist at CSIRO, Sydney, Australia, between 1994 and 2004. He is now an associate professor at Université Paris-Est / ESIEE in Paris, France. He is the co-author or co-editor of 6 books and has published over 120 articles in the area of image processing, image analysis and computer vision. His main interests include mathematical morphology, discrete geometry, combinatorial and continuous optimization.

Nicolas Passat obtained the MSc and PhD from Université Strasbourg 1 in 2002 and 2005, and Habilitation from Université de Strasbourg in 2011. He was an assistant professor at Université de Strasbourg, France, between 2006 and 2012. He is now a full professor at Université de Reims Champagne-Ardenne, France. His scientific interests include mathematical morphology, discrete topology, medical imaging and remote sensing.