Implicit component-graph: A discussion

Nicolas Passat, Benoît Naegel, Camille Kurtz

Implicit Component-Graph: A Discussion

Nicolas Passat[1], Benoît Naegel[2], and Camille Kurtz[3]

[1] Université de Reims Champagne-Ardenne, CReSTIC, France
[2] Université de Strasbourg, CNRS, ICube, France
[3] Université Paris-Descartes, LIPADE, France

Abstract. Component-graphs are defined as the generalization of component-trees to images taking their values in partially ordered sets. Similarly to component-trees, component-graphs are a lossless image model, and can allow for the development of various image processing approaches (e.g., antiextensive filtering, segmentation by node selection). However, component-graphs are not trees, but directed acyclic graphs. This induces a structural complexity associated to a higher combinatorial cost. These properties make the handling of component-graphs a non-trivial task. We propose a preliminary discussion on a new way of building and manipulating component-graphs, with the purpose of reaching reasonable space and time costs. Tackling these complexity issues is indeed required for actually involving component-graphs in efficient image processing approaches.

Keywords: component-graph, algorithmics, data-structure.

1 Introduction

In mathematical morphology, connected operators [1] are often defined from hierarchical image models, namely trees, that represent an image by considering simultaneously its spatial and spectral properties. Among the most popular tree structures modelling images, one can cite the component-tree (CT) [2], the tree of shapes (ToS) [3], and the binary partition tree (BPT) [4]. The CT and ToS —by contrast with the BPT, that derives from both an image and extrinsic criteria— are intrinsic models, that depend only on the image pixel values, and are built in a deterministic way.

Initially, the CT and the ToS (which can be seen as an autodual version of the CT) were defined for grey-level images, i.e. images whose values are totally ordered. Different ways to extend the notions of CT and ToS to multivalued images, i.e. images whose values are partially ordered, were investigated during the last years. The purpose was, in particular, to allow for the use of such image models in a wider range of image processing applications. The first attempt of extending the notion of CT to partially-ordered values was proposed in [5], leading to the pioneering notion of component-graph (CG). The main structural properties of CGs were further established [6]. From an algorithmic point of view, first strategies for building CGs were investigated. Except in a specific case —where the partially ordered set of values is structured itself as a tree [7]— these first attempts emphasised the high computational cost of the construction process, and the high spatial cost of the directed acyclic graph (DAG) explicitly modelling a CG [8,

9]. By relaxing certain constraints, leading to an improved complexity, the notion of CG was successfully involved in the development of an efficient extension of ToS to multivalued images [10]. From an applicative point of view, the notion of CG, coupled with the recent notion of shaping [11], also led to preliminary, yet promising, results for multimodal image processing [12].

In this paper, we propose a preliminary discussion on a new way of building and manipulating CGs. A CG is complex due to its size (i.e. its number of nodes and edges) and its structural complexity (as a DAG), which both induce high space and time computational costs. Our strategy is to take advantage of the structural properties of a CG in order to build dedicated data-structures gathering information useful for handling it, without explicitly computing its whole DAG. We then hope to obtain a reasonable trade-off between the cost of modelling the CG, and the cost of navigating within it for further image processing purposes.

The following discussion is mainly theoretical, providing preliminary ideas; in particular, we will present neither experimental studies nor application cases. The remainder of the paper is organized as follows. Sections 2 and 3 provide notations and recall basic notions on CGs. In Section 4, we enumerate the different functionalities that should be offered by implicit CG data-structures. Sections 5–11 constitute the core of the paper, where we develop our discussion. Section 12 provides concluding remarks.

2 Notations

The used notations are the same as in [6–9]. We only recall here the non-standard ones.

For any symbol further used to denote an order relation ($\subseteq$, $\leqslant$, $\trianglelefteq$, etc.), the inverse symbol ($\supseteq$, $\geqslant$, $\trianglerighteq$, etc.) denotes the associated dual order, while the symbol without lower bar ($\subset$, $<$, $\triangleleft$, etc.) denotes the associated strict order.

If $(X, \leqslant)$ is an ordered set and $x \in X$, we note $x^{\uparrow} = \{y \in X \mid y \geqslant x\}$ and $x^{\downarrow} = \{y \in X \mid y \leqslant x\}$, namely the sets of the elements greater and lower than x, respectively. If $Y \subseteq X$, the sets of all the maximal and minimal elements of Y are noted $\bigtriangledown^{\leqslant} Y$ and $\bigtriangleup^{\leqslant} Y$, respectively.

3 Basic Notions on Component-Graphs

Let Ω be a nonempty finite set. Let V be a nonempty finite set equipped with an order (i.e. reflexive, transitive, antisymmetric) relation $\leqslant$. We assume that $(V, \leqslant)$ admits a minimum, noted $\bot$.

We call an *image* any function

$$\left| \begin{array}{l} I : \Omega \longrightarrow V \\ \quad x \longmapsto v \end{array} \right. \tag{1}$$

Without loss of generality, we assume that $I^{-1}(\{\bot\}) = \{x \in \Omega \mid I(x) = \bot\} \neq \emptyset$.

For any $v \in V$, the thresholding function at value v is defined by

$$\left| \begin{array}{l} \lambda_v : V^{\Omega} \longrightarrow 2^{\Omega} \\ \quad I \longmapsto \{x \in \Omega \mid v \leqslant I(x)\} \end{array} \right. \tag{2}$$

(a) $(V, \prec)$ (b) $I : \Omega \to V$ (c) $\mathfrak{G} = (\Theta, \blacktriangleleft)$

(d) $\lambda_b(I)$ (e) $\lambda_c(I)$ (f) $\lambda_d(I)$ (g) $\lambda_e(I)$ (h) $\lambda_f(I)$ (i) $\lambda_g(I)$ (j) $\lambda_h(I)$ (k) $\lambda_i(I)$ (l) $\lambda_j(I)$

Fig. 1. (a) The Hasse diagram $(V, \prec)$ of an ordered set $(V, \leqslant)$, with $V = \{a, b, c, d, e, f, g, h, i, j\}$ (for the sake of readability, each value of V is associated to an arbitrary colour). (b) An image $I : \Omega \to V$. (c) The component-graph $\mathfrak{G}$ of I. (d–l) Thresholded images $\lambda_v(I)$ for $v \in V$ (the threshold image $\lambda_a(I)$ is not depicted: it is composed of an unique connected component A equal to Ω). The letters (A–S) in nodes (c) correspond to the associated connected components in (d–l).

Let $\frown$ be an adjacency (i.e. irreflexive, symmetric) relation on Ω (the restriction of $\frown$ to any subset of Ω is also noted $\frown$). For any $X \subseteq \Omega$, the set of the connected components of the graph $(X, \frown)$, i.e. the equivalence classes of X with respect to the connectedness relation (i.e. the reflexive–transitive closure) induced by $\frown$, is noted $\mathcal{C}[X]$. Without loss of generality, we assume that $(\Omega, \frown)$ is connected, i.e. $\mathcal{C}[\Omega] = \{\Omega\}$.

Definition 1 (Valued connected component) *Let $v \in V$ and $X \in \mathcal{C}[\lambda_v(I)]$. The couple $K = (X, v)$ is called a* valued connected component*; X is the* support *of K while v is its* value*. We define the set Θ of all the valued connected components of I as*

$$\Theta = \bigcup_{v \in V} \mathcal{C}[\lambda_v(I)] \times \{v\} \tag{3}$$

From the order relation $\leqslant$ on V, and the inclusion relation $\subseteq$ on 2^Ω (the power-set of Ω), we define the order $\trianglelefteq$ on Θ as

$$(X_1, v_1) \trianglelefteq (X_2, v_2) \iff (X_1 \subset X_2) \vee (X_1 = X_2 \wedge v_2 \leqslant v_1) \tag{4}$$

In other words, we enrich the standard inclusion relation, by considering the order on V whenever two valued connected components have the same support. We note $\blacktriangleleft$ the cover relation associated to $\trianglelefteq$, i.e. for all $K_1 \neq K_2 \in \Theta$, we have $K_1 \blacktriangleleft K_2$ iff $K_1 \trianglelefteq K_2$ and there is no $K_3 \in \Theta$ distinct from K_1, K_2 such that $K_1 \trianglelefteq K_3 \trianglelefteq K_2$.

We then have the following definition for the component-graphs.

Definition 2 (Component-graph [5, 6]) *The* Θ-component-graph *(or simply, the* component-graph*) of* I *is the Hasse diagram* $\mathfrak{G} = (\Theta, \blacktriangleleft)$ *of the ordered set* $(\Theta, \trianglelefteq)$*. The elements of* Θ *are called* nodes*; the elements of* $\blacktriangleleft$ *are called* edges*;* $(\Omega, \perp)$ *is called the* root*; the elements of* $\bigwedge^{\trianglelefteq} \Theta$ *are called the* leaves *of the component-graph.*

The component-graph $\mathfrak{G}$ of I is then the Hasse diagram of the ordered set $(\Theta, \trianglelefteq)$ (see Figure 1, for an example). Two other (simpler) variants[1] of CGs were also proposed in [6]. They will not be considered in this study for the sake of concision.

4 Problem Statement

Let $I : \Omega \to V$ be an image defined on a set Ω endowed with an adjacency $\frown$, and taking its values in a set V equipped with an order $\leqslant$. Our purpose is to build dedicated data-structures that model the component-graph $\mathfrak{G}$ of I without explicitly building its whole DAG, and to be able to use them in an efficient way for further image processing purposes. In particular, the main questions that should be easily answered thanks to such data-structures are the following:

(i) Who are the nodes of $\mathfrak{G}$ (i.e. how to identify them)?
(ii) What is a node of $\mathfrak{G}$ (i.e. what are its support and its value)?
(iii) Is a node of $\mathfrak{G}$ lower, greater, or non-comparable to another, with respect to $\trianglelefteq$?

The following sections aim at discussing about the construction of ad hoc data-structures that could allow for answering these questions.

5 Flat Zones and Leaves

5.1 Flat Zone Image

Let $x, y \in \Omega$ be two points of the image I. If x and y are adjacent, i.e. $x \frown y$, and share the same value, i.e. $I(x) = I(y)$, then it is plain that they belong the same valued connected components of $\mathfrak{G}$, i.e. for any $K = (X, v) \in \Theta$, we have $x \in X$ iff $y \in X$. As an immediate corollary, the component-graph obtained from I is isomorphic to the component-graph of the flat zone image[2] associated to I.

A linear time cost $O(|\frown|)$ flat zone computation can then allow us to simplify the image I into its flat zone analogue, thus reducing the space complexity of the image. From now on, and without loss of generality, we will work on such flat zone images, still noted $I : \Omega \to V$ for the sake of readability, and since they are indeed equivalent for CG building. In particular, under this hypothesis, we have the following property.

Property 3 *Let* $x, y \in \Omega$*. If* x *and* y *are adjacent, then their values are distinct, i.e.* $x \frown y \Rightarrow I(x) \neq I(y)$*.*

[1] These two variants, called $\dot{\Theta}$- and $\ddot{\Theta}$-component-graphs, rely on sets of nodes defined as subsets of Θ. A $\ddot{\Theta}$-component graph only contains nodes that are necessary to model I in a lossless way, while the $\dot{\Theta}$-component-graph contains nodes with maximal values for a given support.

[2] The image where each flat zone (i.e. maximal connected region of constant value) is replaced by a single point, and where the adjacency relation between these flat zones is inherited from $\frown$ (i.e. two flat zones are adjacent iff one point of the first is adjacent to one point of the second).

5.2 Detection of the Leaves

Since Ω is finite and $\leqslant$ is antisymmetric, there exist $n \geq 1$ points $x \in \Omega$ such that for all $y \succ x$, we have $I(x) \not< I(y)$, i.e. $I(x) \in V$ is a locally maximal value of the image I. Then, it is plain that $K_x = (\{x\}, I(x))$ is a node of $\mathfrak{G}$, i.e. $K_x \in \Theta$. More precisely, K_x is a minimal element of $\mathfrak{G}$, i.e. $K_x \in \bigwedge^{\trianglelefteq} \Theta$, and is then a leaf of $\mathfrak{G}$ (see Definition 2). As an example, the leaves of the component-graph $\mathfrak{G}$ depicted on Figure 1(c) correspond to the nodes I, S, R, and P.

The characterization of the leaves relying on a local criterion, we can then detect all of them by an exhaustive scanning of Ω, with a linear time cost[3] $O(|\succ|)$. In the sequel, we will denote by $\Lambda \subseteq \Omega$ the set of all the points of Ω that correspond to supports of leaves. In other words, we have $\bigwedge^{\trianglelefteq} \Theta = \{K_x = (\{x\}, I(x)) \mid x \in \Lambda\}$.

6 Node Encoding

Let $K = (X, v) \in \Theta$ be a node of $\mathfrak{G}$. If K is not a leaf, i.e. $K \notin \bigwedge^{\trianglelefteq} \Theta$, then there exists a leaf $K_x = (\{x\}, I(x)) \in \bigwedge^{\trianglelefteq} \Theta$ such that $K_x \trianglelefteq K$. In particular, Equation (4) implies that $x \in X$ and $I(x) > v$. In [6, Property 1], we observed that for a given leaf $K_x = (\{x\}, I(x)) \in \bigwedge^{\trianglelefteq} \Theta$, and for any value $v \leqslant I(x)$, there exists exactly one node $K = (X, v) \in \Theta$ such that $x \in X$.

This allows us to define the function[4]

$$\left|\begin{array}{lll} \kappa : \Lambda \times V \longrightarrow \Theta \cup \{K_\top\} & & \\ \quad (x, v) \longmapsto K = (X, v) \in \Theta \text{ s.t. } x \in X & \text{if } v \leqslant I(x) & \\ \quad (x, v) \longmapsto K_\top & \text{otherwise} & \end{array}\right. \tag{5}$$

This function encodes each node $(X, v) \in \Theta$ of $\mathfrak{G}$ based on two kinds of information: (1) the value v of the node, and (2) a point $x \in \Lambda \cap X$ of locally maximal value, lying within the support of the node.

The function κ is obviously surjective[5]: each node $K = (X, v) \in \Theta$ corresponds to a couple $(x, v) \in \Lambda \times V$ such that $\kappa((x, v)) = K$. This justifies the following property.

Property 4 *The set Θ of the nodes of $\mathfrak{G}$ is defined as $\Theta = \kappa(\Lambda \times V) \setminus \{K_\top\}$.*

However, κ is generally not injective. It is then necessary to handle the synonymy of nodes with respect to κ.

---

[3] In digital imaging, we have $|\succ| = O(|\Omega|)$ (e.g. with 4-, 8-, 6- and 26-adjacencies on $\mathbb{Z}^2$ and $\mathbb{Z}^3$, we have $|\succ| \simeq k.|\Omega|$, with $k = 2$, 4, 3 and 13, respectively), and this generally still holds for the induced flat zone images. Under such hypotheses, the detection of leaves can be performed in linear time $O(|\Omega|)$ with respect to the size of the image. For the sake of concision, we will assume, from now on, that we have indeed $|\succ| = O(|\Omega|)$.

[4] By convention, we define a supplementary node $K_\top$ for handling the cases when the considered value v is greater than (or non-comparable with) the value $I(x)$ associated to x, in order to define κ on the whole Cartesian set $\Lambda \times V$. This does not induce any algorithmic nor structural issue.

[5] Actually, it is surjective iff $\kappa^{-1}(\{K_\top\}) \neq \emptyset$. If $\kappa^{-1}(\{K_\top\}) = \emptyset$, we can simply define $\kappa : \Lambda \times V \to \Theta$, and then the surjectivity property still holds.

7 Node Synonymy Handling

Let $K_1 = (X_1, v_1), K_2 = (X_2, v_2) \in \Theta$ be nodes of $\mathfrak{G}$. From Property 4, there exist $x_1, x_2 \in \Lambda$ such that $\kappa((x_1, v_1)) = K_1$ and $\kappa((x_2, v_2)) = K_2$. If $v_1 \neq v_2$, it is plain that $K_1 \neq K_2$. However, when $v_1 = v_2$, we may have $x_1 \neq x_2$ while $K_1 = K_2$, i.e. $X_1 = X_2$.

In other words, a node $K = (X, v) \in \Theta$ —and more precisely its support X— can be represented by *any* point $x \in \Lambda \cap X$, that also corresponds to the support of a leaf $K_x = (\{x\}, I(x)) \in \bigwedge^{\trianglelefteq} \Theta$. In order to provide an actual modelling of the nodes of $\mathfrak{G}$ from κ, we then have to gather the elements of $\Lambda \times V$ that encode a same node.

To this end, we define the equivalence relation $\sim$ on $\Lambda \times V$ as $(x_1, v_1) \sim (x_2, v_2) \Leftrightarrow \kappa((x_1, v_1)) = \kappa((x_2, v_2))$. From this relation and the function κ, we derive a new function

$$\left| \begin{array}{l} \kappa_\sim : (\Lambda \times V)/\sim \longrightarrow \Theta \cup \{K_\top\} \\ \qquad [(x, v)]_\sim \quad \longmapsto \kappa((x, v)) \end{array} \right. \tag{6}$$

that inherits the surjectivity of κ while guaranteeing, by construction, injectivity. In order to obtain a modelling of the set of nodes Θ of $\mathfrak{G}$ based on a $\Lambda \times V$ encoding, it is then sufficient to compute the equivalence classes induced by the relation $\sim$.

It is plain that for any $(x, v) \in \Lambda \times V$, the equivalence class $[(x, v)]_\sim$ is composed of elements (x', v) with different $x' \in \Lambda$, but a same value v. Then, it is possible to partition the set of equivalence classes of $\sim$ with respect to the different values of V. Indeed, for any $v \in V$, we can define the equivalence relation $\sim_v$ on Λ as $x_1 \sim_v x_2 \Leftrightarrow \kappa((x_1, v)) = \kappa((x_2, v))$. In particular, we have $[(x, v)]_\sim = [x]_{\sim_v} \times \{v\}$.

Another important property is the increasingness of these equivalence classes with respect to the decreasingness of $\leqslant$, that is the fact that for any $x \in \Lambda$, we have $v_1 \leqslant v_2 \Rightarrow [x]_{\sim_{v_2}} \subseteq [x]_{\sim_{v_1}}$. This property can also be written as

$$v_1 \leqslant v_2 \Longrightarrow ((x \sim_{v_2} y) \Rightarrow (x \sim_{v_1} y)) \tag{7}$$

then justifying the following property.

Property 5 *The characterization of $\sim$ only requires to define, for each $v \in V$, the relations $x \sim_v y$ such that for $\forall v' > v, x \nsim_{v'} y$.*

Practically, this property states that the issue of synonymy between nodes could take advantage of the monotony of $\sim_v$ ($v \in V$) with respect to $\leqslant$, to avoid the storage of information already carried by the structure of $(V, \leqslant)$ (see Section 12).

The next step is now to define a way to actually build these equivalence classes.

8 Reachable Zones

In order to build the equivalence relation $\sim$, let us come back to the image I and its adjacency graph $(\Omega, \frown)$. Each point $x \in \Omega$ has a given value $I(x) \in V$. It is also adjacent to other points $y \in \Omega$ (i.e. $x \frown y$) with values $I(y)$. From Property 3, we have either $I(x) < I(y)$ or $I(x) > I(y)$ or $I(x), I(y)$ non-comparable.

From a point $x \in \Lambda$, we can reach certain points $y \in \Omega$ by a descent paradigm. More precisely, for such points y, there exists a path $x = x_0 \frown \ldots \frown x_i \frown \ldots \frown x_t = y$ ($t \geq 0$) in Ω such that for any $i \in [\![0, t-1]\!]$, we have $I(x_i) > I(x_{i+1})$. In such case, we note $x \searrow y$. This leads to the following notion of a reachable zone.

Definition 6 (Reachable zone) *Let $x \in \Lambda$. The* reachable zone *of x (in I) is the set $\rho(x) = \{y \in \Omega \mid x \searrow y\}$.*

When a point $y \in \Omega$ belongs to the reachable zone $\rho(x)$ of a point $x \in \Lambda$, the adjacency path from x to y and the $>$ relation between its successive points imply that the node $(Y, I(y)) \in \Theta$ with $y \in Y$ satisfies $x \in Y$. This justifies the following property.

Property 7 *Let $x \in \Lambda$. Let $v \in V$ such that $v \leqslant I(x)$. Let $K = \kappa((x, v)) = (X, v) \in \Theta$. We have*

$$\{y \in \rho(x) \mid v \leqslant I(y)\} \subseteq X \tag{8}$$

This property states that the supports of the nodes $K = (X, v) \in \Theta$ of $\mathfrak{G}$ can be *partially computed* from the reachable zones of the points of Λ. These computed supports may be yet incomplete, since X can lie within the reachable zones of several points of Λ within a same equivalence class $[x]_{\sim_v}$.

However, the following property, that derives from Definition 6, guarantees that no point of Ω will be omitted when computing the nodes of Θ from unions of reachable zones.

Property 8 *The set $\{\rho(x) \mid x \in \Lambda\}$ is a cover of Ω.*

The important fact of this property is that $\bigcup_{x\in\Lambda} \rho(x) = \Omega$. However, the set of reachable zones is *not* a partition of Ω, in general, due to possible overlaps. For instance, let $x_1, x_2 \in \Lambda$, $y \in \Omega$, and let us suppose that $x_1 \frown y \frown x_2$ and $x_1 > y < x_2$; then we have $y \in \rho(x_1) \cap \rho(x_2) \neq \emptyset$.

Remark 9 *The computation of the reachable zones can be carried out by a seeded region-growing, by considering Λ as set of seeds. The time cost is output-dependent, since it is linear with respect to the ratio of overlap between the different reachable zones. More precisely, it is $O(\sum_{x\in\Lambda} |\rho(x)|) = O((1 + \gamma).|\Omega|)$, with $\gamma \in [0, |\Omega|/4] \subset \mathbb{R}$ the overlap ratio[6], varying between 0 (no overlap between reachable zones, i.e. $\{\rho(x) \mid x \in \Lambda\}$ is a partition of Ω) and $|\Omega|/4$ (all reachable zones are maximally overlapped).*

The fact that two reachable zones may be adjacent and / or overlap will allow us to complete the construction of the nodes $K \in \Theta$ of $\mathfrak{G}$.

9 Reachable Zone Graph

9.1 Reachable Zone Adjacency

When two reachable zones are adjacent —a fortiori when they overlap— they contribute to the definition of common supports for nodes of the component-graph.

Let $x_1, x_2 \in \Lambda$ ($x_1 \neq x_2$), and let us consider their reachable zones $\rho(x_1)$ and $\rho(x_2)$. Let $y_1 \in \rho(x_1)$ and $y_2 \in \rho(x_2)$ be two points such that $y_1 \frown y_2$. Let us consider a value $v \in V$ such that $v \leqslant I(y_1)$ and $v \leqslant I(y_2)$. Since we have $y_1 \frown y_2$, and from the very definition of reachable zones (Definition 6), there exists a path $x_1 = z_0 \frown \ldots \frown z_i \frown \ldots \frown z_t = x_2$ ($t \geq 1$) within $\rho(x_1) \cup \rho(x_2)$ such that $v \leqslant I(z_i)$ for any $i \in [\![0, t]\!]$. This justifies the following property.

[6] The exact upper bound of γ is actually $(|\Omega| - 1)^2/4|\Omega|$, and is reached when $|\Lambda| = (|\Omega| + 1)/2$.

Property 10 *Let $x_1, x_2 \in \Lambda$ with $x_1 \neq x_2$. Let us suppose that there exists $y_1 \frown y_2$ with $y_1 \in \rho(x_1)$ and $y_2 \in \rho(x_2)$. Then, for any $v \in V$ such that $v \leqslant I(y_1)$ and $v \leqslant I(y_2)$, we have $\kappa((x_1, v)) = \kappa((x_2, v))$, i.e. $(x_1, v) \sim (x_2, v)$, i.e. $x_1 \sim_v x_2$.*

This property provides a way to build the equivalence classes of $\sim_v$ ($v \in V$) and then $\sim$. In particular, we can derive a notion of adjacency between the points of Λ or, equivalently, between their reachable zones, leading to a notion of reachable zone graph.

Definition 11 (Reachable zone graph) *Let $\frown_\Lambda$ be the adjacency relation defined on $\Lambda \subseteq \Omega$ as*

$$x_1 \frown_\Lambda x_2 \iff \exists (y_1, y_2) \in \rho(x_1) \times \rho(x_2), y_1 \frown y_2 \tag{9}$$

The graph $\mathfrak{R} = (\Lambda, \frown_\Lambda)$ is called reachable zone graph *(of I).*

9.2 Reachable Zone Graph Valuation

The structural description provided by the reachable zone graph $\mathfrak{R}$ of I is necessary, but not sufficient for building the equivalence classes of $\sim$, and then actually building the nodes of the component-graph $\mathfrak{G}$ via the function κ. Indeed, as stated in Property 10, we also need information about the values of V associated to the adjacency links. This leads us to define a notion of valuation on $\frown_L$ or, more generally[7], on $\Lambda \times \Lambda$.

Definition 12 (Valued reachable zone graph) *Let $\mathfrak{R} = (\Lambda, \frown_\Lambda)$ be the reachable zone graph of an image $I : \Omega \to V$. We define the valuation function σ on the edges of $\mathfrak{R}$ as*

$$\left| \begin{array}{lll} \sigma : \Lambda \times \Lambda & \longrightarrow 2^V & \\ \quad (x_1, x_2) & \longmapsto \{v \mid \exists (y_1, y_2) \in \rho(x_1) \times \rho(x_2), y_1 \frown y_2 \wedge I(y_1) \geqslant v \leqslant I(y_2)\} & \text{if } x_1 \frown_\Lambda x_2 \\ \quad (x_1, x_2) & \longmapsto \emptyset & \text{if } x_1 \not\frown_\Lambda x_2 \end{array} \right. \tag{10}$$

The couple $(\mathfrak{R}, \sigma)$ is called valued reachable zone graph *(of I).*

Remark 13 *For any $x_1, x_2 \in \Lambda$, we have $\sigma((x_1, x_1)) = \emptyset$ and $\sigma((x_1, x_2)) = \sigma((x_2, x_1))$.*

Actually, the definition of σ only requires to consider the couples of points *located at the borders* of reachable zones, as stated by the following reasoning. Let $x_1, x_2 \in \Lambda$ ($x_1 \neq x_2$). Let $v \in \sigma((x_1, x_2))$. Then, there exist $y_1 \in \rho(x_1)$ and $y_2 \in \rho(x_2)$ with $y_1 \frown y_2$, such that $I(y_1) \geqslant v \leqslant I(y_2)$. Now, let us assume that we also have $y_1 \in \rho(x_2)$ and $y_2 \in \rho(x_1)$, i.e. $y_1, y_2 \in \rho(x_1) \cap \rho(x_2)$. There exists a path $x_1 = z_0 \frown \ldots \frown z_i \frown \ldots \frown z_t = y_1$ ($t \geq 1$) within $\rho(x_1)$, with $I(z_i) > I(z_{i+1})$ for any $i \in [\![0, t-1]\!]$. Let $j = \max\{i \in [\![0, t]\!] \mid z_i \notin \rho(x_2)\}$ (we necessarily have $j < t$). Let $v' = I(z_{j+1})$. By construction, we have $(z_j, z_{j+1}) \in \rho(x_1) \times \rho(x_2)$, $z_j \frown z_{j+1}$ and $I(z_j) \geqslant v' \geqslant v \leqslant v' \leqslant I(z_{j+1})$. Then v could be characterized by considering a couple of points (z_j, z_{j+1}) with $z_j \in \rho(x_1) \setminus \rho(x_2)$ and $z_{j+1} \in \rho(x_2)$, i.e. with z_{j+1} at the border of $\rho(x_2)$. This justifies the following property.

[7] The adjacency relation $\frown_L$ is indeed a subset of the Cartesian product $\Lambda \times \Lambda$; so it would be sufficient to define the function $\sigma : \frown_L \to 2^V$. However, such notation is unusual and probably confusing for many readers; then, we prefer to define, without loss of correctness, $\sigma : \Lambda \times \Lambda \to 2^V$, with useless empty values for the couples $(x_1, x_2) \in \Lambda \times \Lambda$ such that $x_1 \not\frown_\Lambda x_2$.

Property 14 *Let $x_1, x_2 \in \Lambda$. Let $v \in V$. We have $v \in \sigma((x_1, x_2))$ iff there exists $y_1 \in \rho(x_1) \setminus \rho(x_2)$ and $y_2 \in \rho(x_2)$ such that $y_1 \frown y_2$ and $I(y_1) \geqslant v \leqslant I(y_2)$.*

Based on this property, the construction of the sets of values $\sigma((x_1, x_2)) \in 2^V$ for each couple $(x_1, x_2) \in \Lambda \times \Lambda$ can be performed by considering only a reduced subset of edges of $\frown$, namely the couples $y_1 \frown y_2$ such that $y_1 \in \rho(x_1) \setminus \rho(x_2)$ and $y_2 \in \rho(x_2)$, i.e. the border of the reachable zone of x_2 adjacent to / within the reachable zone of x_1. In particular, these specific edges can be identified during the computation of the reachable zones, without increasing the time complexity of this step.

Now, let us focus on the determination of the values $v \in V$ to be stored in $\sigma((x_1, x_2))$, when identifying a couple $y_1 \frown y_2$ such that $y_1 \in \rho(x_1) \setminus \rho(x_2)$ and $y_2 \in \rho(x_2)$. Practically, two cases, can occur. **Case 1**: $I(y_1) > I(y_2)$, i.e. $y_2 \in \rho(x_1) \cap \rho(x_2)$. Then all the values $v \leqslant I(y_2)$ are such that $I(y_1) \geqslant v \leqslant I(y_2)$; in other words, we have $I(y_2)^{\downarrow} \subseteq \sigma((x_1, x_2))$. **Case 2**: $I(y_2)$ and $I(y_1)$ are non-comparable, i.e. $y_2 \in \rho(x_2) \setminus \rho(x_1)$. Then, the values that belong to $\sigma((x_1, x_2))$ with respect to this specific edge are those simultaneously below $I(y_1)$ and $I(y_2)$; in other words, we have $(I(y_1)^{\downarrow} \cap I(y_2)^{\downarrow}) \subseteq \sigma((x_1, x_2))$.

Remark 15 *Instead of exhaustively computing the whole set of values $\sigma((x_1, x_2)) \in 2^V$ associated to the edge $x_1 \frown_\Lambda x_2$, it is sufficient to determine the* maximal values *of such a subset of V. We can then consider the function $\sigma_\triangledown : \Lambda \times \Lambda \to 2^V$, associated to σ, and defined by $\sigma_\triangledown((x_1, x_2)) = \bigvee^{\leqslant} \sigma((x_1, x_2))$. In particular for any $x_1 \frown_\Lambda x_2$, we have $v \in \sigma((x_1, x_2))$ iff there exists $v' \in \sigma_\triangledown((x_1, x_2))$ such that $v \leqslant v'$.*

10 Algorithmic Sketch

Based on the above discussion, we now recall the main steps of the algorithmic process for the construction of data-structures allowing us to handle a component-graph.

10.1 Input / Output

The input of the algorithm is an image I, i.e. a function $I : \Omega \to V$, defined on a graph $(\Omega, \frown)$ and taking its values in an ordered set $(V, \leqslant)$. Note that I is first preprocessed in order to obtain a flat zone image (see Section 5.1); this step presents no algorithmic difficulty and allows for reducing the space cost of the image without altering the structure of the component-graph to be further built.

The output of the algorithm, i.e. the set of data-structures required to manipulate the component-graph implicitly modelled, is basically composed of:

- the ordered set $(V, \leqslant)$ (and / or its Hasse diagram $(V, \prec)$);
- the initial graph $(\Omega, \frown)$, equipped with the function $I : \Omega \to V$;
- the set of leaves $\Lambda \subseteq \Omega$;
- the function $\rho : \Lambda \to 2^\Omega$ that maps each leaf to its reachable zone (and / or its "inverse" function $\rho^{-1} : \Omega \to 2^\Lambda$ that indicates, for each point of Ω, the reachable zone(s) where it lies);
- the reachable zone graph $\Re = (\Lambda, \frown_\Lambda)$;
- the valuation function $\sigma : \Lambda \times \Lambda \to 2^V$ (practically, $\sigma : \frown_\Lambda \to 2^V$) or its "compact" version $\sigma_\triangledown$.

10.2 Algorithmics and Complexity

The initial graph $(\Omega, \frown)$ and the ordered set $(V, \leqslant)$ are provided as input. The space cost of $(\Omega, \frown)$ is $O(|\Omega|)$ (we assume that $|\frown| = O(|\Omega|)$). The space cost of $(V, \leqslant)$ is $O(|V|)$ (by modelling $\leqslant$ via its Hasse diagram, i.e. its transitive reduction, we can assume that $|\leqslant| = O(|V|)$). The space cost of I is $O(|\Omega|)$.

As stated in Section 5.2, the set of leaves $\Lambda \subseteq \Omega$ is computed by a simple scanning of Ω, with respect to the image I. This process has a time cost $O(|\Omega|)$. The space cost of Λ is $O(|\Omega|)$ (with, in general, $|\Lambda| \ll |\Omega|$).

As stated in Section 8, the function ρ (and equivalently, ρ^{-1}) is computed by a seeded region-growing. This process has a time cost which is output-dependent, and in particular equal to the space cost of the output; this cost is $O(|\Omega|^\alpha)$ with $\alpha \in [1, 2]$. One may expect that, for standard images, we have $\alpha \simeq 1$.

The adjacency $\frown_\Lambda$ can be computed on the fly, during the construction of the reachable zones. By assuming that, at a given time, the reachable zones of Λ^+ are already constructed while those of Λ^- are not yet ($\Lambda^+ \cup \Lambda^- = \Lambda$), when building the reachable zone of $x \in \Lambda^-$, if a point $y \in \Omega$ is added to $\rho(x)$, we add the couple (x, x') to $\frown_\Lambda$ if there exists $z \in \rho(x')$ in the neighbourhood of y, i.e. $z \frown y$. This process induces no extra time cost to that of the above seeded region-growing. The space cost of $\frown_\Lambda$ is $O(|\Lambda|^\beta)$ with $\beta \in [1, 2]$. One may expect that, for standard images, we have $\beta \simeq 1$.

The valuation function $\sigma : \Lambda \times \Lambda \to 2^V$ is built by scanning the adjacency links of $\frown$ located on the borders of the reachable zones[8]. This subset of adjacency links has a space cost $O(|\Omega|^\delta)$, with $\delta \in (0, 1]$. One may expect that $\delta \ll 1$, due to the fact that we consider borders of regions[9]. For each adjacency link of this subset of $\frown$, one or several values (their number can be expected to be, in average, a low constant value k) are stored in σ. This whole storage then has a time cost $O(k.|\Omega|^\delta)$. Then, some of these values are removed, since they do not belong to $\sigma_\triangledown$. If we assume that we are able to compare the values of $(V, \leqslant)$ in constant time[10] $O(1)$, this removal process, that finally leads to $\sigma_\triangledown$, has in average, a time cost of $O((k.|\Omega|^\delta)^2/|\Lambda|^\beta)$. The space cost of $\sigma_\triangledown$ is $O(k.|\Omega|^\delta)$.

11 Data-Structure Manipulation

Once all these data-structures are built, it is possible to use them in order to manipulate an implicit model of component-graph. In particular, let us come back to the three questions stated in Section 4, that should be easily answered for an efficient use of component-graphs.

Who are the nodes of $\mathfrak{G}$ (i.e. how to identify them)? A node $K = (X, v) \in \Theta$ can be identified by a leave $K_x \in \bigwedge^\trianglelefteq \Theta$ (i.e. a flat zone $x \in \Lambda$) and its value $v \in V$, by considering the κ encoding, that is "$\kappa((x, v))$, the node of value v whose support contains the

[8] Actually, half of these borders, due to the symmetry of the configurations, see Property 14.

[9] For instance, for a digital images in $\mathbb{Z}^3$ (resp. $\mathbb{Z}^2$) of size $|\Omega| = N^3$ (resp. N^2), we may expect a space cost of $O(N^2)$ (resp. $O(N^1)$), i.e. $\delta = 2/3$ (resp. $\delta = 1/2$).

[10] A sufficient condition is to model $(V, \leqslant)$ as an intermediate data-structure of space cost $O(|V|^2)$.

flat zone x". More generally, since the ρ^{-1} function provides the set of reachable zones where each point of Ω lies, it is also possible to identify the node $K = (X, v) \in \Theta$ by any point of X and its value v, that is "the node of value v whose support contains x".

What is a node of $\mathfrak{G}$ (i.e. what are its support and its value)? A node $K = (X, v) \in \Theta$ is identified by at least one of the flat zones $x \in \Lambda$ within its support X, and its value v. The access to v is then immediate. By contrast, the set X is constructed on the fly, by first computing the set of all the leaves forming the connected component $\Lambda_v^x \ni x$ of the thresholded graph $(\Lambda, \lambda_v(\frown_\Lambda))$, where $\lambda_v(\frown_\Lambda) = \{x_1 \frown_\Lambda x_2 \mid v \in \sigma((x_1, x_2))\}$. This process indeed corresponds to a seeded region-growing on a multivalued map. Once the subset of leaves $\Lambda_v^x \subseteq \Lambda$ is obtained, the support X is computed as the union of the threshold sets of the corresponding reachable zones, i.e. $X = \bigcup_{y \in \Lambda_v^x} \lambda_v(I_{\lfloor \rho(y) \rfloor})$.

Is a node of $\mathfrak{G}$ lower, greater, or non-comparable to another, with respect to $\trianglelefteq$? Let us consider two nodes defined as $K_1 = \kappa(x_1, v_1)$ and $K_2 = \kappa(x_2, v_2)$. Let us first suppose that $v_1 \leqslant v_2$. Then, two cases can occur: (1) $\kappa(x_1, v_1) = \kappa(x_2, v_1)$ or (2) $\kappa(x_1, v_1) \neq \kappa(x_2, v_1)$. In case 1, we have $K_2 \trianglelefteq K_1$; in case 2, they are non-comparable. To decide between these two cases, it is indeed sufficient to compute —as above— the set of all the leaves forming the connected component $\Lambda_{v_1}^{x_1}$ of the thresholded graph $(\Lambda, \lambda_{v_1}(\frown_\Lambda))$, and to check if $x_2 \in \Lambda_{v_1}^{x_1}$ to conclude. Let us now suppose that v_1 and v_2 are non-comparable. A necessary condition for having $K_2 \trianglelefteq K_1$ is that $\Lambda_{v_2}^{x_2} \subseteq \Lambda_{v_1}^{x_1}$. We first compute these two sets; if the inclusion is not satisfied, then K_1 and K_2 are non-comparable, or $K_1 \trianglelefteq K_2$. If the inclusion is satisfied, we have to check, for each leaf $x \in \Lambda_{v_2}^{x_2}$, whether $\lambda_{v_2}(I) \cap \rho(x_2) \subseteq \lambda_{v_1}(I) \cap \rho(x_1)$; this iterative process can be interrupted as soon as a negative answer is obtained. If all these inclusions are finally satisfied, then we have $K_2 \trianglelefteq K_1$.

Remark 16 *By contrast with a standard component-graph —that explicitly models all the nodes Θ and edges $\blacktriangleleft$ of the Hasse diagram— we provide here an implicit representation of $\mathfrak{G}$ that requires to compute on the fly certain information. This computation is however reduced as much as possible, by subdividing the support of the nodes as reachable regions, and manipulating them in a symbolic way, i.e. via their associated leaves and the induced reachable region graph, whenever a —more costly— handling of the "real" regions of Ω is not required. Indeed, the actual use of these regions of Ω is considered only when spatial information is mandatory, or when the comparison between nodes no longer depends on information related to the value part of $\trianglelefteq$, i.e. $\leqslant$, but to its spatial part, i.e. $\subseteq$ (see Equation* (4)*).*

12 Concluding Remarks

This paper has presented a preliminary discussion about a way to handle component-graphs without explicitly building them. In previous works, we had observed that the computation of a whole component-graph led not only to a high time cost, but also to a data-structure whose space cost forbade an efficient use for image processing. Based on these considerations, our purpose was here to build, with a reasonable time cost, some

data-structures of reasonable space cost, allowing us to manipulate an implicit model of component-graph, in particular by computing on the fly, the required information.

This is a work in progress, and we do not yet pretend that the proposed solutions are relevant. We think that the above properties are correct, and that the proposed algorithmic processes lead to the expected results (implementation in progress). At this stage, our main uncertainty is related to the real space and time cost of these data-structures, their construction and their handling. It is plain that these costs are input/output-dependent, and in particular correlated with the nature of the order $\leqslant$, and the size of the value set V. Further experiments will aim at clarifying these points.

Our perspective works are related to (*i*) the opportunities offered by our paradigm to take advantage of distributed algorithmics (in particular via the notion of reachable zones, that subdivide Ω); (*ii*) the development of a cache data-structure (e.g., based on the Hasse diagram $(V, \prec)$), in order to reuse some information computed on the fly, taking advantage in particular of Property 5; and (*iii*) investigation of the links between this approach and the concepts developed on directed graphs [13], of interest with respect to the notion of descending path used to build reachable zones.

References

1. Salembier, P., Wilkinson, M.H.F.: Connected operators: A review of region-based morphological image processing techniques. IEEE Signal Processing Magazine **26** (2009) 136–157
2. Salembier, P., Oliveras, A., Garrido, L.: Anti-extensive connected operators for image and sequence processing. IEEE Transactions on Image Processing **7** (1998) 555–570
3. Monasse, P., Guichard, F.: Scale-space from a level lines tree. Journal of Visual Communication and Image Representation **11** (2000) 224–236
4. Salembier, P., Garrido, L.: Binary partition tree as an efficient representation for image processing, segmentation and information retrieval. IEEE Transactions on Image Processing **9** (2000) 561–576
5. Passat, N., Naegel, B.: An extension of component-trees to partial orders. In: ICIP. (2009) 3981–3984
6. Passat, N., Naegel, B.: Component-trees and multivalued images: Structural properties. Journal of Mathematical Imaging and Vision **49** (2014) 37–50
7. Kurtz, C., Naegel, B., Passat, N.: Connected filtering based on multivalued component-trees. IEEE Transactions on Image Processing **23** (2014) 5152–5164
8. Naegel, B., Passat, N.: Toward connected filtering based on component-graphs. In: ISMM. Volume 7883 of Lecture Notes in Computer Science., Springer (2013) 350–361
9. Naegel, B., Passat, N.: Colour image filtering with component-graphs. In: ICPR. (2014) 1621–1626
10. Carlinet, E., Géraud, T.: MToS: A tree of shapes for multivariate images. IEEE Transactions on Image Processing **24** (2015) 5330–5342
11. Xu, Y., Géraud, T., Najman, L.: Connected filtering on tree-based shape-spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence **38** (2016) 1126–1140
12. Grossiord, É., Naegel, B., Talbot, H., Passat, N., Najman, L.: Shape-based analysis on component-graphs for multivalued image processing. In: ISMM. Volume 9082 of Lecture Notes in Computer Science., Springer (2015) 446–457
13. Perret, B., Cousty, J., Tankyevych, O., Talbot, H., Passat, N.: Directed connected operators: Asymmetric hierarchies for image filtering and segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **37** (2015) 1162–1176