Algorithms for hierarchical segmentation based on the Felzenszwalb-Huttenlocher dissimilarity

Edward Jorge Yuri Cayllahua Cahuina, Jean Cousty, Yukiko Kenmochi, Arnaldo de Albuquerque Araujo, Guillermo Cámara-Chávez

Algorithms for hierarchical segmentation based on the Felzenszwalb-Huttenlocher dissimilarity

Edward Cayllahua Cahuina*‡, Jean Cousty*†, Yukiko Kenmochi*, Arnaldo de Albuquerque Araujo ‡, Guillermo Cámara-Chávez §

* *Université Paris-Est, LIGM, ESIEE Paris - CNRS, France*

†*Université Paris Descartes, Laboratoire MAP5, UMR CNRS 8145, France*

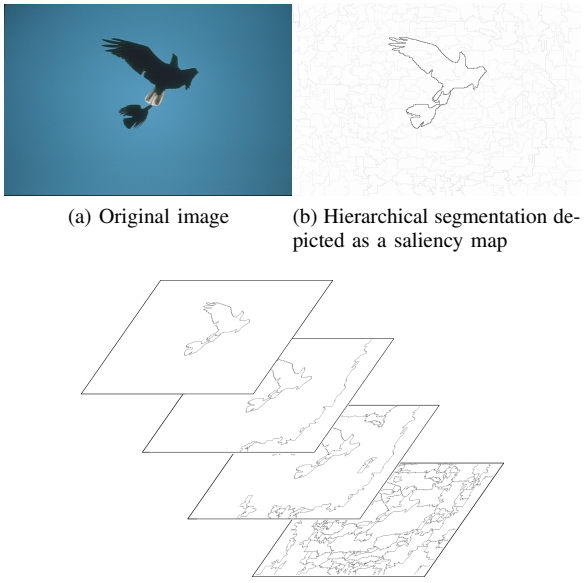
‡*Universidade Federal de Minas Gerais, Computer Science Dept., Brazil*

§*Universidade Federal de Ouro Preto, Computer Science Dept., Brazil*

***Abstract*—Hierarchical image segmentation provides a region-oriented scale-space, *i.e.*, a set of image segmentations at different detail levels in which the segmentations at finer levels are nested with respect to those at coarser levels. Most image segmentation algorithms, such as region merging algorithms, rely on a criterion for merging that does not lead to a hierarchy. Guimarães *et al.* proposed in 2012 a hierarchical graph-based image segmentation method relying on a criterion popularized by Felzenszwalb and Huttenlocher in 2004, hence hierarchizing the popular Felzenszwalb-Huttenlocher method. However, Guimarães *et al.* did not provide an algorithm to compute the proposed hierarchy. We propose a series of algorithms to compute the result of this hierarchical graph-based image segmentation method. For an image of size 321 $\times$ 481 pixels, the most efficient algorithm produces the result in half a second whereas the most naive one requires more than four hours.**

I. Introduction

A hierarchical image segmentation is a series of image segmentations at different detail levels where the segmentations at higher detail levels are produced by merging regions from segmentations at finer detail levels. Consequently, the regions at finer detail levels are nested in regions at coarser levels. A hierarchical image segmentation is illustrated in Fig. 1.



(a) Original image (b) Hierarchical segmentation depicted as a saliency map

(c) Some levels of this hierarchical segmentation

Fig. 1: Illustration of a hierarchical image segmentation.

Hierarchical image segmentation provides a multi-scale approach to image analysis. Hierarchical image analysis was pioneered by [1] and has received a lot of attention since then, as attested by the popularity of [2]. In [3], the global information is used to create the initial regions and then the region merging process is treated as a series of optimization problems. Mathematical morphology is also used in hierarchical image analysis with, *e.g.*, hierarchical watersheds [4, 5], binary partition trees [6], scale-set theory [7], or quasi-flat zones hierarchies [8].

In [9] (see [10] for its preliminary version), the quasi-flat zone hierarchy is used to perform a hierarchical image segmentation. This work relies on the graph-based (GB) image segmentation algorithm proposed in [11]. GB algorithm uses a merging predicate to decide if, at a certain scale parameter, two adjacent regions of an image should be merged into a single one, thus, producing a segmented image. In its original form, GB algorithm does not directly lead to a hierarchical image segmentation. In [9], the merging predicate of GB algorithm is used along with the quasi-flat zone hierarchy to produce a hierarchical version of GB method called HGB.

The HGB method produces satisfactory segmentation results (see [9]). Nonetheless, a precise algorithm to compute efficiently the result of HGB method is not provided in [9]. The core of HGB method is based on solving a minimization problem whose solution is the minimum observation scale at which adjacent regions in the image have to be merged. To solve this minimization, the method considers all positive real values to find such minimum observation scale.

In this article, we study HGB method and we focus on two problems that make difficult its implementation. A first difficulty is related to solving the minimization problem of HGB method for which a precise algorithmic solution is not given in [9]. We analyze this minimization process and propose three algorithms that solve it. The first one solves the minimization by searching the result in a large space of possible values, we then reduce this search space to avoid redundant computations, leading to two efficient algorithms. The second problem is related to the quasi-flat zone computation. One approach can

be to use an efficient algorithm, such as [12], to compute it at every step of HGB method. However, efficiency can be improved by only updating at each iteration the existing quasi-flat zone hierarchy instead of recomputing it from scratch. This is done with a procedure similar to the one proposed in [13, 14]. Overall, the most efficient proposed algorithm computes the result of HGB method for an image of size 321 $\times$ 481 pixels in about half a second whereas it takes over four hours with the most naive algorithm.

II. Hierarchical graph-based image segmentation

This section aims at explaining the method of hierarchical graph-based image segmentation (HGB) [10]. The hierarchy is constructed from an image via a graph representation, based on the notion of a quasi-flat zone hierarchy [8]. We first give a series of necessary notions, and then explain HGB method.

A. Basic notions

1) Hierarchies: Given a finite set V, a *partition* of V is a set $\mathbf{P}$ of nonempty disjoint subsets of V whose union is V. Any element of $\mathbf{P}$ is called a *region* of $\mathbf{P}$. Given two partitions $\mathbf{P}$ and $\mathbf{P}'$ of V, $\mathbf{P}'$ is said to be a refinement of $\mathbf{P}$, denoted by $\mathbf{P}' \preceq \mathbf{P}$, if any region of $\mathbf{P}'$ is included in a region of $\mathbf{P}$. A hierarchy on V is a sequence $\mathcal{H} = (\mathbf{P}_0, \ldots, \mathbf{P}_\ell)$ of partitions of V, such that $\mathbf{P}_{i-1} \preceq \mathbf{P}_i$, for any $i \in \{1, \ldots, \ell\}$.

2) Graph and connected-components partition: A *graph* is a pair $G = (V, E)$ where V is a finite set and E is a subset of $\{\{x, y\} \subseteq V | x \neq y\}$. Each element of V is called a *vertex* of G, and each element of E is called an *edge* of G. A subgraph of G is a graph (V', E') such that $V' \subseteq V$ and $E' \subseteq E$. If X is a graph, its vertex and edge sets are denoted by $V(X)$ and $E(X)$, respectively.

Let x and y be two vertices of a graph G. A *path from x to y* in G is a sequence $(x_0, \ldots, x_m)$ of vertices of G such that $x_0 = x$, $x_m = y$ and $\{x_{i-1}, x_i\}$ is an edge of G for any i in $\{1, \ldots, m\}$. The graph G is *connected* if, for any vertices x and y of G, there exists a path from x to y. Let A be a subset of $V(G)$. The graph induced by A in G is the graph whose vertex set is A and whose edge set contains any edge of G made of two elements in A. If the graph induced by A is connected, then we say that A is *connected*. The subset A of $V(G)$ is a *connected component* of G if it is connected for G and maximal for this property. We denote by $\mathbf{C}(G)$ the set of all connected components of G. Note that $\mathbf{C}(G)$ is a partition of $V(G)$, which is called the *connected-components partition* induced by G.

3) Quasi-flat zones hierarchies: Let us now present the quasi-flat zones hierarchies which provide a bijection between an edge-weighted graph and a hierarchy (see more details in [15]).

Given a graph $G = (V, E)$, let w be a map from E into the set $\mathbb{R}$ of real numbers. For any edge u of G, the value $w(u)$ is called the weight of u (for w), and the pair (G, w) is called an *edge-weighted graph*.

Given an edge-weighted graph (G, w), let X be a subgraph of G and let λ be a value of $\mathbb{R}$. The *λ-level edge set* of X for w is defined by $w_\lambda(X) = \{u \in E(X) \mid w(u) < \lambda\}$, and the *$\lambda$-level graph* of X for w is defined as the subgraph $w^V_\lambda(X)$ of X, such that $w^V_\lambda(X) = (V(X), w_\lambda(X))$. Then, the connected-components partition $\mathbf{C}(w^V_\lambda(X))$ induced by $w^V_\lambda(X)$ is called the *λ-level partition* of X for w.

As we consider only finite graphs and hierarchies, the set of considered level values is reduced to a finite subset of $\mathbb{R}$ that is denoted by $\mathbb{E}$ in the remaining parts of this article. In order to browse the values of this set and to round real values to values of $\mathbb{E}$ we define, for any $\lambda \in \mathbb{R}$:

$$\begin{aligned} \mathrm{p}_{\mathbb{E}}(\lambda) &= \max\{\mu \in \mathbb{E} \cup \{-\infty\} \mid \mu < \lambda\}; \\ \mathrm{n}_{\mathbb{E}}(\lambda) &= \min\{\mu \in \mathbb{E} \cup \{\infty\} \mid \mu > \lambda\}; \text{ and} \\ \hat{\mathrm{n}}_{\mathbb{E}}(\lambda) &= \min\{\mu \in \mathbb{E} \cup \{\infty\} \mid \mu \geq \lambda\}. \end{aligned}$$

Let (G, w) be an edge-weighted graph and let X be a subgraph of G. The sequence of all λ-level partitions of X for w ordered by increasing value of λ, namely $(\mathbf{C}(w^V_\lambda(X)) \mid \lambda \in \mathbb{E} \cup \{\infty\})$, is a hierarchy, called the *quasi-flat zone hierarchy of X for w*, denoted by $\mathcal{QFZ}(X, w)$. Let $\mathcal{H}$ be the quasi-flat zone hierarchy of G for w. Given a vertex x of G and a value λ in $\mathbb{E}$, the region that contains x in the λ-level partition of the graph G is denoted by $\mathcal{H}^\lambda_x$.

In the remaining parts of this article, the symbol G denotes a connected graph, the symbol w denotes a map from E into $\mathbb{R}$, and the symbol T denotes a minimum spanning tree of (G, w). It has been shown in [15] that the quasi-flat zone hierarchy $\mathcal{QFZ}(T, w)$ of T for w is the same as the quasi-flat zone hierarchy $\mathcal{QFZ}(G, w)$ of G for w. This indicates that the quasi-flat zone hierarchy for G can be handled by its minimum spanning tree.

B. Hierarchical graph-based segmentation method

In this article, we consider that the input is the edge-weighted graph (G, w) representing an image, where the pixels correspond to the vertices of G and the edges link adjacent pixels. The weight of each edge is given by a dissimilarity measure between the linked pixels such as the absolute difference of intensity between them.

Before explaining HGB method, we first describe the following observation scale dissimilarity [9], which is required by the method and whose idea originates from the region merging criterion proposed in [11].

1) Observation scale dissimilarity: In the case of the Felzenszwalb-Huttenlocher image segmentation algorithm [11], two regions of an image are merged based on a region merging predicate. This predicate was later reformulated as an observation scale dissimilarity measure to produce the hierarchical segmentation of an image [9] as follows.

Let R_1 and R_2 be two adjacent regions, the dissimilarity measure compares the so-called inter-component and within-component differences [11]. The *inter-component difference* between R_1 and R_2 is defined by $\Delta_{inter}(R_1, R_2) = \min\{w(\{x, y\}) | x \in R_1, y \in R_2, \{x, y\} \in E(T)\}$, while the *within-component difference* of a region R is defined by $\Delta_{intra}(R) = \max\{w(\{x, y\}) | x, y \in R, \{x, y\} \in E(T)\}$. It

Method 1: HGB method

Input : A minimum spanning tree T of an edge-weighted graph (G, w)
Output: A hierarchy $\mathcal{H} = \mathcal{QFZ}(T, f)$

1 **for** *each* $u \in E(T)$ **do** $f(u) := \max\{\lambda \in \mathbb{E}\}$;
2 **for** *each* $u \in E(T)$ *in non-decreasing order for* w **do**
3 $\quad \mathcal{H} := \mathcal{QFZ}(T, f)$;
4 $\quad f(u) := \mathrm{p}_{\mathbb{E}}(\lambda^{\star}_{\mathcal{H}}(u))$;
5 **end**
6 $\mathcal{H} := \mathcal{QFZ}(T, f)$;

leads to the observation scale of R_1 relative to R_2, defined by $S_{R_2}(R_1) = (\Delta_{inter}(R_1, R_2) - \Delta_{intra}(R_1))\,|R_1|$, where $|R_1|$ is the cardinality of R_1. Then, a symmetric metric between R_1 and R_2, called the observation scale dissimilarity, is defined by

$$D(R_1, R_2) = \max\{S_{R_2}(R_1), S_{R_1}(R_2)\}. \tag{1}$$

This dissimilarity is used to determine if two regions should be merged or not at a certain observation scale in the following.

2) Method: The HGB method is presented in Method 1. The input is an image represented by a graph G with its associated weight function w, where the minimum spanning tree T of G is taken indeed. From (T, w), HGB method computes a new weight function f which leads to a new hierarchy $\mathcal{H} = \mathcal{QFZ}(T, f)$. The resulting hierarchy $\mathcal{H}$ is considered as the hierarchical image segmentations of the initial image. Thus, the core of the method is the generation of the weight function f for T.

To compute the new map f, the HGB method first initializes all values of f to infinity (see Line 1). Then, an observation scale value $f(u)$ is computed for each edge $u \in E(T)$ in non-decreasing order with respect to the original weight w (see Line 2). Note that each iteration in the loop requires computing the hierarchy $\mathcal{H} = \mathcal{QFZ}(T, f)$ (see Line 3). Once $\mathcal{H}$ is obtained, the value $\lambda^{\star}_{\mathcal{H}}(u)$ of a finite subset $\mathbb{E}$ of $\mathbb{R}$ is obtained by the minimization:

$$\lambda^{\star}_{\mathcal{H}}(\{x, y\}) = \min\left\{\lambda \in \mathbb{E} \mid D\left(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y\right) \leq \lambda\right\}. \tag{2}$$

We first consider the regions $\mathcal{H}^{\lambda}_x$ and $\mathcal{H}^{\lambda}_y$ at a level λ. Using the dissimilarity measure D we check if $D\left(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y\right) \leq \lambda$. Equation (2) states that $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$ is the minimum value λ that fulfills this minimization. Observe that the minimization involved in Equation (2) has a solution only if the maximum of $\mathbb{E}$ is greater than the maximum possible dissimilarity value. In the following, we assume that this assumption always holds true. Fig. 2 illustrates an example of application of Method 1.

As mentioned above, Guimarães *et al.* did not provide a practically efficient algorithm to compute Method 1. In order to fill this gap, the problem is twofold. Indeed, it is necessary to propose efficient (i.e., exact and fast) algorithms for (i) solving the minimization involved in Equation (2); and (ii) computing the quasi-flat zone hierarchy $\mathcal{QFZ}(T, f)$ at each iteration of Method 1 (Lines 3 and 6).

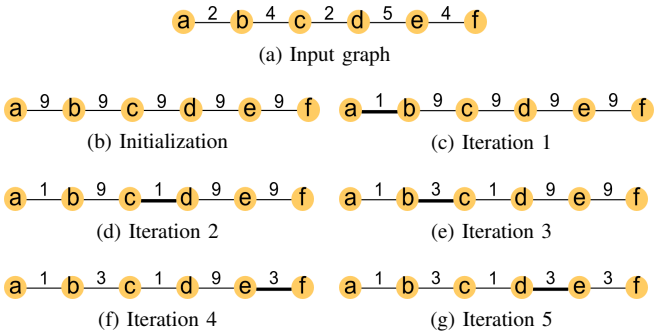


Fig. 2: Illustration of Method 1 with $\mathbb{E} = \{0, 1, \ldots, 9\}$: (a) the input graph (T, w), (b-g) the graph (T, f) at each iteration of Method 1 and (g) the resulting quasi-flat zone hierarchy corresponding to graph (T, f).

III. Algorithms for HGB method

In this section, we investigate algorithms to compute the results of HGB method. In Sections III-A, III-B, and III-C, three algorithms to perform the minimization involved at Line 4 of Method 1 are presented. In Section III-C, we present non-incremental and incremental algorithms to obtain the quasi-flat zone hierarchy of a weight map as requested at Lines 3 and 6 of Method 1.

A. Naive minimization algorithm

We first present a naive algorithm, namely Algorithm 1, to compute the value $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$ given a hierarchy $\mathcal{H}$ and an edge $\{x, y\}$. According to Equation (2), it simply consists of considering the values of $\mathbb{E}$ in increasing order until finding a value $\lambda \in \mathbb{E}$ such that $D\left(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y\right) \leq \lambda$. We remark that, when $\mathbb{E}$ is a set of consecutive integers, for any λ in $\mathbb{E}$, the result of $\mathrm{n}_{\mathbb{E}}(\lambda)$ and $\mathrm{p}_{\mathbb{E}}(\lambda)$ can be obtained with the simple integer instruction $\lambda + 1$ and $\lambda - 1$, respectively.

Algorithm 1: HGB Naive minimization of Equation 2

Input : A hierarchy $\mathcal{H}$, an edge $u = \{x, y\}$
Output: The value $\lambda^{\star}$ such that $\lambda^{\star} = \lambda^{\star}_{\mathcal{H}}(\{x, y\})$

1 $\lambda^{\star} := \min\{\lambda \in \mathbb{E}\}$;
2 **while** $D\left(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y\right) > \lambda^{\star}$ **do**
3 $\quad \lambda^{\star} := \mathrm{n}_{\mathbb{E}}(\lambda^{\star})$;
4 **end**

B. Minimization by range

In Algorithm 1, $D\left(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y\right)$ is computed for every value λ in $\mathbb{E}$. However, it may well arise that at two successive values of $\mathbb{E}$, the regions $\mathcal{H}^{\lambda}_x$ and $\mathcal{H}^{\lambda}_y$ remain the same and in this case so does $D\left(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y\right)$. In this section, we present a second algorithm to compute $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$ which allows us to reduce the amount of redundant calculation compared to Algorithm 1.

In order to obtain this reduction of redundant calculation, we search for a partition of $\mathbb{E}$ (which, according to Equation (2), is the range of the possible values of $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$)

into a finite number of discrete intervals such that, in each interval $I =]\!]I_{\min}, I_{\max}]\!] \subseteq \mathbb{E}$, for any two values λ_1 and λ_2 in I the regions containing x and y at level λ_1 and at level λ_2 remain unchanged. Thus, in this case, we would have $D\left(\mathcal{H}_x^{\lambda_1}, \mathcal{H}_y^{\lambda_1}\right) = D\left(\mathcal{H}_x^{\lambda_2}, \mathcal{H}_y^{\lambda_2}\right) = D_I$ which means in short that, in such interval I, the dissimilarity should be computed only once. Then, it can be observed that, for any value λ in $I \cap]\!]D_I, I_{\max}]\!]$, the dissimilarity between the regions containing x and y at level λ of the hierarchy $\mathcal{H}$ is below the value λ and that, for any value λ in $I \cap]\!]I_{\min}, D_I[\![$, the dissimilarity between the regions containing x and y at level λ of the hierarchy $\mathcal{H}$ is above λ. Hence, the solution $\lambda^\star_{\mathcal{H}}(\{x, y\})$ to our optimization problem can be obtained by browsing all the intervals of the considered partition of $\mathbb{E}$.

In order to obtain such partition of $\mathbb{E}$, we remark that if there is no edge in $E(T)$ whose weight is between λ_1 and λ_2, then the λ_1-level edge set of T for f is equal to the λ_2-level edge set of T for f, which implies in turn that the λ_1 level of $\mathcal{H} = \mathcal{QFZ}(T, f)$ is equal to the λ_2 level of $\mathcal{H}$. Hence, to obtain a partition of $\mathbb{E}$, such as the one described in the previous paragraph, it is sufficient to ensure that for every considered interval $]\!]I_{\min}, I_{\max}]\!]$, there is no edge whose weight is strictly between $I_{\min}$ and $I_{\max}$. In order to find the desired partition $\mathbb{E}$, it is relevant to consider the successive values of the range $\mathbb{R}_f = \{f(u) \mid u \in E(T)\}$ of the weight function f. More precisely, from the above discussion, we deduce the following property.

Property 1: Let $\mathcal{H}$ be a hierarchy, let f be a map from $E(T)$ to $\mathbb{E}$ such that $\mathcal{H} = \mathcal{QFZ}(T, f)$ and let $\{x, y\}$ be any edge of T. Then, we have

$$\lambda^\star_{\mathcal{H}}(\{x, y\}) = \min\{\max\left(\hat{\mathrm{n}}_{\mathbb{E}}\left(D\left(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda\right)\right), \mathrm{n}_{\mathbb{E}}\left(\mathrm{p}_{\mathbb{R}_f}(\lambda)\right)\right) \mid \lambda \in \mathbb{R}_f, D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}. \quad (3)$$

Thanks to Property 1, we compute $\lambda^\star_{\mathcal{H}}(\{x, y\})$ by browsing the values of $\mathbb{R}_f$ in increasing order until a value λ such that $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda$ is found and by taking the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$ which is simply the maximum between $\hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda)\right)$ and $\mathrm{n}_{\mathbb{E}}\left(\mathrm{p}_{\mathbb{R}_f}(\lambda)\right)$. In order to make such process computable, it is necessary to browse the range of f in increasing order. To this end, we propose to store the values of f in a sorted linked list. Algorithm 2 provides a precise description of this process. It can be observed that when the value $\mathrm{p}_{\mathbb{E}}(\lambda^\star_{\mathcal{H}}(\{x, y\}))$ is not yet present in the range of f, the linked list representing this range is updated so that it is ready for the next iteration of the main loop in Method 1. It has to be also noted that in Method 1, the weight of every edge is initialized to the maximal value of $\mathbb{E}$. In other words, the linked list must be initialized in Method 1 with the singleton $\{\max\{\lambda \in \mathbb{E}\}\}$.

Algorithm 2: HGB Minimization by range

Input : A hierarchy $\mathcal{H}$, a weight map f such that $\mathcal{H} = \mathcal{QFZ}(T, f)$, an edge $\{x, y\}$ of T, a linked list L of the values of $\mathbb{R}_f$ in increasing order

Output: The value $\lambda^\star$ such that $\lambda^\star = \lambda^\star_{\mathcal{H}}(\{x, y\})$, the updated linked list L of the values of $\mathbb{R}_f \cup \{\mathrm{p}_{\mathbb{E}}(\lambda^\star)\}$ in increasing order

```
l := L.head; λ := l.value; λ_prev := −∞ ;
while D(H_x^λ, H_y^λ) > λ do
    λ_prev := λ; l := l.next; λ := l.value;
end
λ* := max(n_E(λ_prev), n̂_E(D(H_x^λ, H_y^λ)));
if p_E(λ*) ≠ λ_prev then L.insert(p_E(λ*));
```

C. Minimization by branch

In the previous section, we reduce the size of the search space of the minimization defined in Equation (2) by considering the range $\mathbb{R}_f$ of the function f (*i.e.*, a characteristic function of the considered hierarchy $\mathcal{H}$) instead of the set $\mathbb{E}$ of all possible scales of the hierarchy $\mathcal{H}$ (see Property 1). In this section, we show that this search space can be further reduced, leading to a third algorithm for computing the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$, given any hierarchy $\mathcal{H}$ and any edge $\{x, y\}$.

In order to obtain this second reduction, we observe in Equation (2) that the only regions of the hierarchy involved in the minimization are those containing x and y. Therefore, while searching for the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$, it is unnecessary to consider a scale of $\mathcal{H}$ (*i.e.*, a value in $\mathbb{R}_f$) at which the regions containing x and y are the same as those at the preceding scale. In other words, rather than considering the scales in $\mathbb{R}_f$ for which there is a global change in the hierarchy, one can focus on the scales for which the change of the hierarchy is local to x and y, *i.e.*, when the change involves a region containing either x or y.

Let x be any vertex of V and let us denote by $\mathcal{B}_{\mathcal{H}}(x)$ the set which contains every region R of the hierarchy $\mathcal{H}$ such that x belongs to R. The set $\mathcal{B}_{\mathcal{H}}(x)$ is called the *branch of x in $\mathcal{H}$*. The *level* of a region R, denoted by $level_{\mathcal{H}}(R)$, in $\mathcal{H}$ is the lowest index of a partition that contains R in $\mathcal{H}$. The *(branch) range of $\mathcal{H}$ for x*, denoted by $\mathbb{R}_f^x$, is defined as the set that contains the level of every region of the branch of x in $\mathcal{H}$: $\mathbb{R}_f^x = \{level_{\mathcal{H}}(R) \mid R \in \mathcal{B}_{\mathcal{H}}(x)\}$. Using this notion of a branch range the following property can be deduced. The difference with Property 1 is that the range of f (f being such that $\mathcal{H} = \mathcal{QFZ}(T, f)$) is replaced by the union of the branch ranges of $\mathcal{H}$ for x and for y.

Property 2: Let $\mathcal{H}$ be a hierarchy and let $\{x, y\}$ be any edge of T. Then, we have:

$$\lambda^\star_{\mathcal{H}}(\{x, y\}) = \min\{\max\left(\hat{\mathrm{n}}_{\mathbb{E}}\left(D\left(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda\right)\right), \mathrm{n}_{\mathbb{E}}\left(\mathrm{p}_{\mathbb{B}}(\lambda)\right)\right) \mid \lambda \in \mathbb{B}, D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda\}, \quad (4)$$

where $\mathbb{B} = \mathbb{R}_f^x \cup \mathbb{R}_f^y$.

Due to Property 2, to compute $\lambda^\star_{\mathcal{H}}(\{x, y\})$, it is sufficient to browse in increasing order the levels of the regions in the branches of x and of y until a value λ, such that $D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda) \leq \lambda$, is found. Finally, the value $\lambda^\star_{\mathcal{H}}(\{x, y\})$ is determined as the maximum between $\hat{\mathrm{n}}_{\mathbb{E}}\left(D(\mathcal{H}_x^\lambda, \mathcal{H}_y^\lambda)\right)$ and $\mathrm{n}_{\mathbb{E}}(\mathrm{p}_{\mathbb{B}}(\lambda))$, where $\mathbb{B} = \mathbb{R}_f^x \cup \mathbb{R}_f^y$. In order to propose such an algorithm,

we need to browse in increasing order the levels of the regions in the branches of x and of y. This can be done with a tree data structure, called a component tree, which represents the hierarchy. The component tree is used for various image processing tasks and is well studied in the field of mathematical morphology (see, *e.g.*, [16] for its definition on vertex weighted graphs, [17] for the case of edge-weighted graphs and quasi-flat zone, and [18] for their generalization to directed graphs). In classification, this tree is often called the dendrogram of the hierarchy.

As any tree, the component tree of $\mathcal{H}$ can be defined as a pair made of a set of nodes and of a binary (parent) relation on the set of nodes. More precisely, the *component tree of* $\mathcal{H}$ is the pair $\mathcal{T}_{\mathcal{H}} = (\mathcal{N}, parent)$ such that $\mathcal{N}$ is the set of all regions of $\mathcal{H}$ and such that a region R_1 in $\mathcal{N}$ is a *parent* of a region R_2 in $\mathcal{N}$ whenever R_1 is a minimal (for inclusion relation) proper superset of R_2. Note that every region in $\mathcal{N}$ has exactly one parent except the region V which has no parent and is called the *root* of the component tree of $\mathcal{H}$. Any region which is not the parent of another one is called a *leaf* of the tree. It can be observed that any singleton of V is a leaf of $\mathcal{T}_{\mathcal{H}}$ and that conversely any leaf of $\mathcal{T}_{\mathcal{H}}$ is a singleton of V.

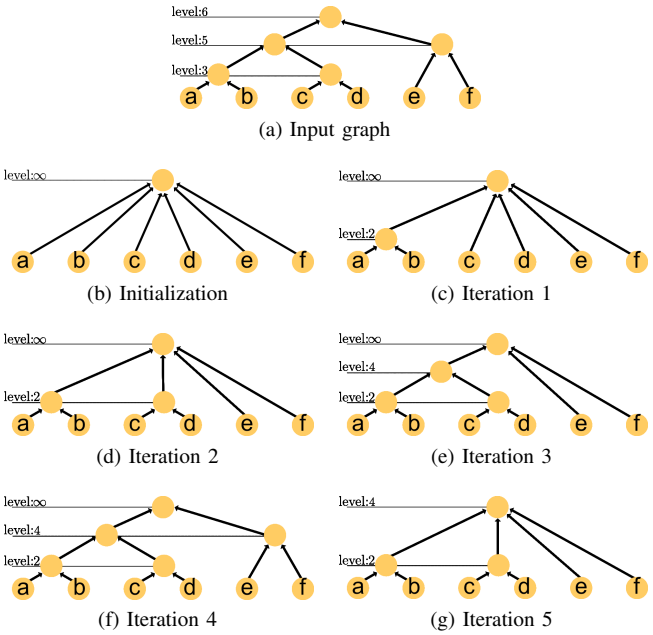
In order to browse the branch of x in $\mathcal{H}$ from its component tree, it is enough to follow the next steps: (1) start with the node C that is the leaf $\{x\}$, (2) consider the parent of C, and (3) repeat step (2) until the root is found. Furthermore, it can be observed that the $level_{\mathcal{H}}$ attribute is increasing in the branch of x: for any non-root node C in $\mathcal{N}$, the level of the parent of C is never less than the level of C. Hence, the branch browsing process also allows browsing the branch range of $\mathcal{H}$ for x in increasing order. According to Property 2, in order to find the value $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$, for any edge $\{x, y\}$ of T and any hierarchy $\mathcal{H}$, we have to consider the union of the ranges of $\mathcal{H}$ for x and for y, sorted in increasing order. This can be done by simultaneously browsing in the component tree $\mathcal{T}_{\mathcal{H}}$ the branches of x and of y. Algorithm 3 provides a precise description of a complete algorithm to find $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$ using such a simultaneous branch browsing.

Algorithm 3: HGB Minimization by branch

Input : The component tree $\mathcal{T} = (\mathcal{N}, parent)$ of a hierarchy $\mathcal{H}$, an edge $u = \{x, y\}$ of T, an array $level$ that stores the level of every region of $\mathcal{H}$

Output: The value $\lambda^{\star}$ such that $\lambda^{\star} = \lambda^{\star}_{\mathcal{H}}(\{x, y\})$

1 $C_x := \{x\}$; $C_y := \{y\}$; $\lambda := -\infty$; $\lambda_{prev} := -\infty$;
2 **while** $D(C_x, C_y) > \lambda$ **do**
3 $\quad \lambda_{prev} := \lambda$;
4 $\quad \lambda := \min(level[parent[C_x]], level[parent[C_y]])$;
5 $\quad$ **if** $level[parent[C_x]] = \lambda$ **then** $C_x := parent[C_x]$;
6 $\quad$ **if** $level[parent[C_y]] = \lambda$ **then** $C_y := parent[C_y]$;
7 **end**
8 $\lambda^{\star} := \max(\mathrm{n}_{\mathbb{E}}(\lambda_{prev}), \hat{\mathrm{n}}_{\mathbb{E}}(D(\mathcal{H}^{\lambda}_x, \mathcal{H}^{\lambda}_y)))$;



(a) Input graph (b) Initialization (c) Iteration 1 (d) Iteration 2 (e) Iteration 3 (f) Iteration 4 (g) Iteration 5

Fig. 3: Tree representations of the quasi-flat zones hierarchies of the graphs of Fig. 2; (g) shows the output hierarchy computed by HGB Method.

D. QFZ computation

In this section, we focus on Lines 3 and 6 of Method 1 that is, on computing the quasi-flat zone hierarchy of a weight map f. This computation is repeated at every iteration of the method (*i.e.*, for every edge of the tree T). Hence, finding an efficient way to perform this task in the context of Method 1 presents a high speedup potential.

A first implementation for this task consists simply of computing, at every iteration, the quasi-flat zone hierarchy of f using an efficient algorithm such as the one presented in [12]. However, from one iteration of the main loop of Method 1 to the next one, only one weight of the graph is updated and therefore most parts of the component tree remain unchanged (see, for instance, Fig. 3). Hence, rather than recomputing from scratch the whole component tree at each iteration, one can guess that an important speedup can be reached by updating only the part of the component tree which is affected by the single weight update considered at the present iteration. Such computation is referred to as an incremental quasi-flat zone update. In [13] and [14], the authors propose an algorithm to merge the component trees of two disjoint (adjacent) image blocks in order to obtain the component tree of the image consisting of these two blocks. Since the weight of an edge is updated only once during the whole execution of Method 1, from the initial value $\max\{\lambda \in \mathbb{E}\}$ to its final value, the algorithms described in [13] and [14] can be adapted to the problem of this article. The update algorithm modifies the tree structure in the following manner: first, given an edge $u = \{x, y\}$ of updated weight $\lambda = f(u)$, the components $\mathcal{H}^{\lambda}_x$ and $\mathcal{H}^{\lambda}_y$ are identified in the tree and then a

TABLE I: Execution times from the image of Fig. 4(a) (321×481 pixels). The resulting hierarchy contains 5218 levels.

QFZ Algorithm	Minimization Algorithm	Execution times (seconds)		
		Total	QFZ	Minimization
Non-Incremental	Algorithm 1	14666.08	13186.31	1479.56
	Algorithm 2	13392.51	13375.29	17.02
	Algorithm 3	13166.25	13165.54	0.49
Incremental	Algorithm 1	1487.96	0.13	1487.75
	Algorithm 2	15.42	0.13	15.21
	Algorithm 3	0.49	0.10	0.32

new node is created in the tree structure at level $\lambda + 1$, which represents the union of these two components. Finally, the algorithm identifies the ancestors of these components in the tree, and updates the parenthood relationship of these nodes. This is done until the root is found. Consequently, only the components containing x and y are involved in the update algorithm and we do not need to recompute a whole hierarchy at every iteration.

IV. Assessments

The experiment aims at measuring and comparing the execution times of all the variations of our algorithms, which are previously presented for HGB method; as we have three variations for the minimization step (Line 4 in Method 1), Algorithms 1, 2 and 3, and two variations for quasi-flat zone computation (Lines 3 and 6 in Method 1), the non-incremental one [12] and the incremental one [13, 14], the total number of all the combinations is six. The algorithms were implemented in C and executed on a computer with a 3.2 GHz CPU, 8GB RAM. The six algorithms were executed on the image of Fig. 4 (a). Fig. 4 (b) shows the resulting hierarchical segmentation, from which we see that a large number of regions and hierarchical levels were produced.

Table I shows the results for all the variations. We observe that using the incremental quasi-flat zone computation provides a great gain in efficiency compared to the non-incremental approach. For the minimization step, Algorithm 1 is the least efficient of all. It is important to notice that Algorithm 3 is much faster than Algorithm 2, which validates that minimization by branch is the most efficient of the three algorithms to solve the minimization problem. For further assessment, we tested our fastest algorithm on the 500 images of the Berkeley dataset leading to an average execution time of 0.19 ± 0.02 seconds.

V. Conclusions

We first investigated the HGB method [10] with the aim of proposing practical algorithms for its implementation on images. We focused on two steps for improving efficiency: (i) the minimization involved in Equation (2), and (ii) the computation of the quasi-flat zones hierarchies. Concerning (i), we presented two properties which allow us to improve an algorithm to compute the minimum value $\lambda^{\star}_{\mathcal{H}}(\{x, y\})$ step by step. The most efficient one is Algorithm 3, as confirmed by the assessments (see Table I). In order to compute efficiently the quasi-flat zone hierarchy (ii), the incremental update strategy [13] was used. The improvement over the non-incremental strategy is also confirmed by the experiment.

(a) Input image (b) Hierarchical segmentation

Fig. 4: Image used for the algorithm assessment and the resulting HGB hierarchy represented as a saliency map.

References

[1] T. Pavlidis, *Structural Pattern Recognition*. Springer, 1977.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *TPAMI*, pp. 898–916, 2011.

[3] J.-H. Syu, S.-J. Wang, and L. Wang, "Hierarchical image segmentation based on iterative contraction and merging," *TIP*, pp. 2246 – 2260, 2017.

[4] S. Beucher, "Watershed, hierarchical segmentation and waterfall algorithm," in *ISMM*, 1994, pp. 69–76.

[5] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *TPAMI*, pp. 1163–1173, 1996.

[6] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *TIP*, pp. 561–576, 2000.

[7] L. Guigues, J. P. Cocquerez, and H. Le Men, "Scale-sets image analysis," *IJCV*, pp. 289–317, 2006.

[8] F. Meyer and P. Maragos, "Morphological scale-space representation with levelings," in *Scale-Space Theories in Computer Vision*, 1999, pp. 187–198.

[9] S. Guimarães, Y. Kenmochi, J. Cousty, Z. Patrocinio Jr., and L. Najman, "Hierarchizing graph-based image segmentation algorithms relying on region dissimilarity - the case of the Felzenszwalb-Huttenlocher method," *Math. Morphol. Theory Appl.*, pp. 1–22, 2017.

[10] S. Guimarães, J. Cousty, Y. Kenmochi, and L. Najman, "A hierarchical image segmentation algorithm based on an observation scale," in *SSPR*, 2012, pp. 116–125.

[11] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, pp. 167–181, 2004.

[12] L. Najman, J. Cousty, and B. Perret, "Playing with kruskal: algorithms for morphological trees in edge-weighted graphs," in *ISMM*, 2013, pp. 135–146.

[13] J. Havel, F. Merciol, and S. Lefèvre, "Efficient tree construction for multiscale image representation and processing," *JRTIP*, pp. 1–18, 2016.

[14] M. H. Wilkinson, H. Gao, W. H. Hesselink, J.-E. Jonker, and A. Meijster, "Concurrent computation of attribute filters on shared memory parallel machines," *TPAMI*, pp. 1800–1813, 2008.

[15] J. Cousty, L. Najman, Y. Kenmochi, and S. Guimarães, "Hierarchical segmentations with graphs: Quasi-flat zones, minimum spanning trees, and saliency maps," *JMIV*, pp. 1–22, 2017.

[16] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *TIP*, pp. 555 – 570, 1998.

[17] J. Cousty, L. Najman, and B. Perret, "Constructive links between some morphological hierarchies on edge-weighted graphs," in *ISMM*, 2013, pp. 135–146.

[18] B. Perret, J. Cousty, O. Tankyevych, H. Talbot, and N. Passat, "Directed connected operators: asymmetric hierarchies for image filtering and segmentation," *TPAMI*, pp. 1162–1176, 2015.