Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

1D Component tree in linear time and space and its application to gray-level image multithresholding

David Menotti[1,2], Laurent Najman[1] and Arnaldo de A. Araújo[2]

[1] *Université Paris-Est (UPE), LABINFO-IGM, UMR CNRS 8049, A2SI-ESIEE, France*
{d.menotti,l.najman}@esiee.fr

[2] *Universidade Federal de Minas Gerais (UFMG), Belo Horizonte, Brazil*
{menotti,arnaldo}@dcc.ufmg.br

Abstract The upper-weighted sets of a signal are the sets of points with weight above a given threshold. The components of the upper-weighted sets, thanks to the inclusion relation, can be organized in a tree structure, which is called the component tree. In this work, we present a linear time and space algorithm to compute the component tree of one-dimensional signals. From this algorithm we derive an efficient gray-level image multithresholding method, which is based on the hypothesis that objects which appear on an image can be represented by salient classes present in the histogram of this image. These classes are modelled as the most significative components of the histogram's component tree. We show results of the proposed method and compare it with classical methods.

Keywords: component-tree, weighted ordered sets, multithresholding.

1. Introduction

The upper-weighted sets of a signal are the sets of points with weight above a given threshold. The components of the upper-weighted sets, thanks to the inclusion relation, can be organized into a tree structure, that is called the component tree. The component tree captures some essential features of a signal. It has been used (under several variations) in numerous applications including image filtering and segmentation [5], video segmentation [13], image registration [9], image compression [13]. In the literature, there are several algorithms to compute the component tree [2, 8, 10, 13]. The algorithm with the best time complexity to compute the component tree for N dimensional signals (e.g., a mapping from $\mathbb{Z}^N$ to $\mathbb{Z}$, where $N \in \mathbb{N}$) was recently proposed in [10] and it is quasi linear.

In this work, we propose a time and space linear algorithm to compute the component tree of a weighted ordered set (WOS), i.e., a model of 1D signals. As a possible application, we propose a gray-level image multithresholding method for image segmentation, which is based on the

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

hypothesis that objects which appear on an image can be represented by salient classes present in the histogram of the image. These salient classes are modelled as the most significative components of the component tree, where the importance corresponds to the volume attribute.

The remaining of this paper is organized as follows. In Section 2, we introduce definitions for WOS and define the component tree in this framework. An algorithm to compute the component tree in linear time and space for WOS is presented in Section 3. In Section 4, we introduce a new method for gray-level image multithresholding. An experimental comparison with classical methods is performed. In Section 5, conclusions are pointed out.

2. Weighted ordered sets and the component tree

In this section we introduce the notion of WOS, which allows us to model 1D signals, and the component tree of such WOS.

2.1 Basic notions for ordered set

Let P be a finite set of points and let $\prec$ be a binary relation on P (i.e., a subset of the Cartesian product $P \times P$) which is transitive ($(x,y) \in \prec, (y,z) \in \prec \Rightarrow (x,z) \in \prec$), and trichotomous (i.e., exactly one of $(x,y) \in \prec, (y,x) \in \prec$ and $x = y$ is true). The relation $\prec$ defines an (total) order on P, and the pair $(P, \prec)$ is a *(totally) ordered set*. Let $(P, \prec)$ be an ordered set and let $x, y, z \in P$. If $(x,y) \in \prec$ and there is no z such that $(x,z) \in \prec$ and $(z,y) \in \prec$, then we say that y is the *successor* of x and x is the *predecessor* of y. Let $(P, \prec)$ be an ordered set. Let $X = \{x_0, x_1, ..., x_n\}$ be a subset of points of P where $x_0, x_1, ..., x_n$ are arranged in increasing order. If for any $i \in [1, n]$, x_i is the successor of x_{i-1}, then we say that X is a *connected set*. We also say that x_0 and x_n are the starting and the ending points of X, respectively.

2.2 Basic notions for weighted ordered set

We denote by $\mathcal{F}(P, D)$, or simply by $\mathcal{F}$, the set composed of all mappings from P to D, where D is any set equipped with a total order e.g., the set of rational numbers or the set of integers). For a mapping $F \in \mathcal{F}$, the triplet $(P, \prec, F)$ is called a *weighted ordered set* (WOS). For a point $p \in P$, $F(p)$ is called the *weight* (or *level*) of p. Let $F \in \mathcal{F}$ and $h \in D$, we define the *h upper-weighted set of* F, denoted by F_h, as $\{p \in P | F(p) \geq h\}$. A connected set X of an upper-weighted set F_h, which is maximal (i.e., $X = Y$ whenever $X \subseteq Y \subseteq P$ and Y is connected), is called a (*h-weighted*) *connected component* (of F). A h-weighted connected component of F that does not contain a $(h+1)$-weighted connected component of F is called a (*regional*) *maximum* of F. We define $h_{min} = min\{F(p) | p \in P\}$ and

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

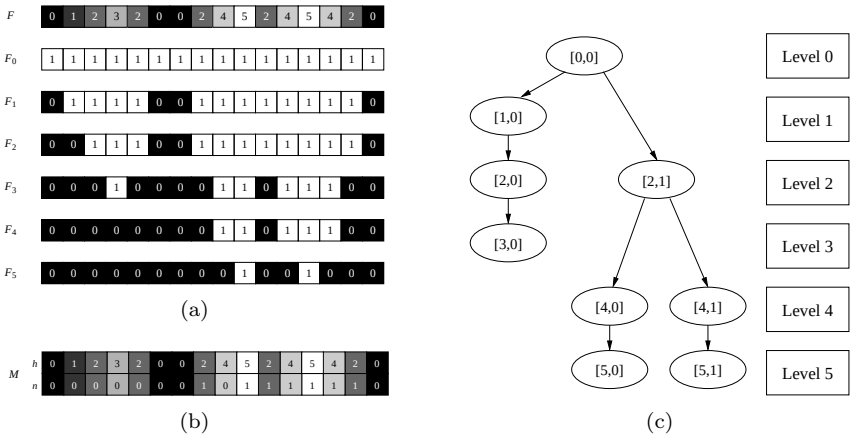

Figure 1. A component tree example. (a) A weighted ordered set $(P, \prec, F)$ and its upper-weighted sets at weights 0, 1, 2, 3, 4 and 5. (b) The associated component mapping M. (c) The component tree $\mathcal{C}(F)$ of F.

$h_{max} = max\{F(p)|p \in P\}$ as the minimum and the maximum weights in the mapping F, respectively.

Figure 1(a) shows a WOS $(P, \prec, F)$ with 16 points and the 6-upper-weighted sets of F, from $h_{min} = 0$ to $h_{max} = 5$. The set F_5 is made of two connected components which are regional maxima of F. The set F_3, in turn, is made of three connected components - one of them being a regional maximum of F.

2.3 Component tree

From the example shown in Figure 1(a) we observe that the weighted connected components of the different upper-weighted sets may be organized to form a tree structure, thanks to the inclusion relation.

Let $F \in \mathcal{F}$ and let $s \subseteq P$ be a connected component of F. We set $f(s) = max\{h|s$ is a (h-weighted) connected component of $F\}$. Note that $f(s) = min\{F(p)|p \in s\}$. Let $h = f(s)$, we say that s is a (h-*weighted*) *(proper) component of* F. We define $\mathcal{C}(F)$ as the set of all components of F. Let $F \in \mathcal{F}$ and let x and y be distinct elements of $\mathcal{C}(F)$. We say that x is the *parent* of y if $y \subset x$ and there is no other $z \in \mathcal{C}(F)$ such that $y \subset z \subset y$. In this case, we also say that y is the *child* of x. In a parent-children relationship, $\mathcal{C}(F)$ forms a directed tree named *component tree* of F, which will also be denoted by $\mathcal{C}(F)$ by abuse of terminology.

Any element of $\mathcal{C}(F)$ is called a *node*. The node that has no parent, in turn, is called the *root* of the component tree. In the following, for the sake of algorithm description, we denote by $c_{h,n}$ the $(n+1)$ -th h-weighted component of $\mathcal{C}(F)$, where the order of the h-weighted components is derived

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

from the order of their starting point in $(P, \prec)$. In applications, we need to recover the component to which a given point belongs to. For such aim, let us consider the *component mapping* M defined for any point $p \in P$ by $M(p) = [h, n]$, where $h = f(p)$ and $c_{h,n}$ contains p.

Figure 1(c) and Figure 1(b) show the component tree of the WOS depicted in Figure 1(a) and the associated component mapping, respectively. The component $c_{0,0}$ at weight 0 is associated with the node $[0, 0]$, the component $c_{1,0}$ at weight 1 is associated with the node $[1, 0]$, and so on.

3. Linear component tree algorithm for WOS

3.1 Description

In this work, we build the component tree $\mathcal{C}(F)$ of a WOS $(P, \prec, F)$ (i.e., 1D signal) by detecting the components and the parent-children relationship among them. Simultaneously, we build its respective component mapping M. The components of $\mathcal{C}(F)$ are detected in the WOS by analyzing the connected components of F. In an 1D space, the connected components of the upper-weighted sets can be determined by their limits, i.e., the starting and the ending points of the connected components. The connected component limits of upper-weighted sets provide the component limits and, therefore, information for the detection of the components.

In fact, in order to build the component tree we do not need to know exactly the position of the components in the WOS. What we need is to know the components hierarchy, since they respect an inclusion relation. In order to build the component tree in linear time, we propose to analyze the WOS from the starting point up to the ending point. By processing the WOS point by point, we determine every connected components and, consequently, the components present in the WOS with a single scan. In this same scan, we can also establish the hierarchy of the components necessary to create the parent-children relationship. Adopting this approach we can compute the component tree for 1D signals in linear time.

The proposed algorithm roughly works as follows. For each point in the WOS, it checks if the point is a starting point, an ending point, or an inner point (a point which is neither a starting nor an ending point) of a component of F. During this analysis of the WOS points, if a component indicated by a point is found to have descendants it is stored into a stack. The stack plays a fundamental role to maintain the hierarchy of the component tree, as the parent-children relationships are created as edges between parent and child components. In the following paragraphs we explain in details how the component tree and component mapping are build.

The first point, or the starting point p in the WOS $(P, \prec, F)$ receives a special treatment. It belongs to the first component at weight $p_h = F(p)$, and receives the label 0 at the weight p_h. Hence, the node $[p_h, 0]$ is associated with the point p on the component mapping. Once the starting point has

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

been analyzed, we consider the next points. For the sake of simplicity, consider the point p as the current point being analyzed and suppose we want to make decisions about the component, indicated by its predecessor r. We first analyze the weights $p_h = F(p)$ and $r_h = F(r)$, we can find three possibilities: $p_h > r_h$, $p_h = r_h$, and $p_h < r_h$, as shown in Figures 2(a), 2(b), and 2(c).

In the first case, where $p_h > r_h$ (Figure 2(a)), we create a new component at weight p_h, that is, p is the starting point of a component and receives a new label p_n at weight p_h. The node $[p_h, p_n]$ is associated with the point p on the component mapping. Since the node $[r_h, r_n]$ has at least one descendant, i.e., $[p_h, p_n]$, it will be inserted into the stack. Note that no other component with weight smaller or equal to p_h will be inserted into the stack while the component $[p_h, p_n]$ is there.

In the second case, where $p_h = r_h$ (Figure 2(b)), we know that the point p belongs to the same component indicated by the point r. Therefore, the node $[r_h, r_n]$ is associated with the point p, which is the same node as the one which contains r, in the component mapping.

In the last case, where $p_h < r_h$ (Figure 2(c) — an ending or inner point), we know for certain that the component to which r belongs to is already analyzed, i.e., the point r is the ending point of the component. In this situation, we have to decide which component is the parent of the node $[r_h, r_n]$. This decision is based onto the relationship of the nodes in the stack (nodes with descendants to be analyzed) and the node $[p_h, p_n]$. Four scenarios might appear here, as shown in Figures 2(c), 2(d), 2(e), and 2(f).

The first scenario involves the stack being empty. If there are no elements left on the stack (Figure 2(c)), we conclude that the node $[p_h, p_n]$ is the parent of $[r_h, r_n]$. In this situation, we know that the point p belongs to a new component, which is assigned a new label, i.e., p_n, at weight p_h. Hence, the node $[p_h, p_n]$ is associated with the point p on the component mapping. Note that in this case, the point p is not the starting point of this component.

In the other three scenarios assume that there are elements left on the stack, and consider that the stack head element corresponds to the node $[q_h, q_n]$. If $[q_h, q_n]$ has its weight q_h smaller than p_h, i.e., $p_h > q_h$ (Fig-

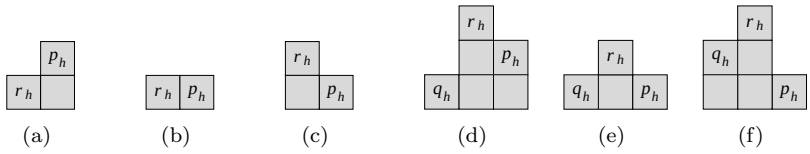

Figure 2. Possible predecessors (r's) of p and the disposition of p in relation to the stack (q's): (a) $p_h > r_h$; (b) $p_h = r_h$; (c) $p_h < r_h$; (d) $p_h > q_h$; (e) $p_h = q_h$; (f) $p_h < q_h$.

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

ure 2(d)), we have the same situation as when there are no elements on the stack.

In contrast, if $[q_h, q_n]$ has its weight q_h equal to p_h, i.e., $p_h = q_h$ (Figure 2(e)), we observe that the point p belongs to the same component as q, i.e., $p_h = q_h$ and $p_n = q_n$ (the node $[p_h, p_n]$ is associated with the point p on the component mapping). Hence, the node $[p_h, p_n]$ (or $[q_h, q_n]$) is the parent of $[r_h, r_n]$. In this case, the node $[q_h, q_n]$ is removed from the stack, since the possible descendant components between the points q and p have already been computed. Nevertheless, the node $[q_h, q_n]$ can be inserted again into the stack later.

The last possible scenario shows that the node $[q_h, q_n]$ has its weight q_h greater than p_h, i.e., $p_h < q_h$ (Figure 2(f)). Here we have that $[q_h, q_n]$ is the parent of $[r_h, r_n]$. The node $[q_h, q_n]$ is removed from the stack, since $[p_h, p_n]$ has a weight smaller than it. This is necessary to keep the consistency of the stack.

After the node is removed from the stack, we need to find its parent. The node $[p_h, p_n]$ or the new node on the stack head are candidate parents of the removed node $[q_h, q_n]$. The decision about which component is the parent is done by naming the removed component $[q_h, q_n]$ as $[r_h, r_n]$, and starting the decision process again according to the situations presented in Figures 2(c), 2(f), 2(d), and 2(e), as described previously.

Once all points in the WOS were processed, we can still have some components left on the stack. In this case, the component on the stack head is the parent of the component pointed by the ending point of the WOS. The next component on the stack, if there is any, is the parent of the stack head, and so on. These components are removed from stack one by one, and edges are inserted between the parent and child components such that the parent-children relationship is finished. When the stack is empty the component tree is complete.

3.2 Implementation

Algorithm 1 shows an implementation (with low level details) of the algorithm described in Section 3.1 to compute in linear time and space the component tree $\mathcal{C}(F)$, the component mapping M of a WOS $(P, \prec, F)$. The component tree structure (CT) obtained from the algorithm is composed of vectors which store pairs of the child and parent components (the parent-children relationship). Another vector, *nnodes*, composes the CT structure. It is used to indicate the number of nodes at each weight. Thus, the vector *nnodes* is used to generate unique labels for the new nodes at each weight during the processing of the WOS. The stack used, CP (to store pairs $[q_h, q_n]$), implement four basic operations: `StackPush`, `StackPop`, `StackEmpty`, and `StackView`. In order to build the parent-children relationship of the component tree, the function `InsEdge` is used to insert edges between nodes in the CT structure.

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

Algorithm 1: BuildComponentTree.

Data: $(P, \prec, F)$ - weighted ordered set with n points
Result: CT - component tree structure
Result: M - a map from P to $[h_{min}...h_{max}, 0...n-1]$

```
1  CT.nnodes[F(0)] + + ; M(0) ← [F(0), 0];
2  for i ← 1 ; i < n ; i + + do
3      p_h ← F(i) ; [r_h, r_n] ← M(i − 1);
4      if (p_h > r_h) then
5          p_n ← CT.nnodes[p_h] + + ; M(i) ← [p_h, p_n];
6          StackPush(CP, [r_h, r_n]);
7      else if (p_h = r_h) then
8          p_n ← r_n ; M(i) ← [p_h, p_n];
9      else if (p_h < r_h) then
10         while (!StackEmpty(CP)) do
11             [q_h, q_n] ← StackView(CP);
12             if (p_h ≥ q_h) then break ;
13             InsEdge(CT, [q_h, q_n], [r_h, r_n]) ;
14             StackPop(CP);
15             [r_h, r_n] ← [q_h, q_n];
16         if (StackEmpty(CP) and (p_h < r_h)) or (p_h > q_h) then
17             p_n ← CT.nnodes[p_h] + + ; M(i) ← [p_h, p_n];
18             InsEdge(CT, [p_h, p_n], [r_h, r_n]);
19         else if (p_h = q_h) then
20             p_n ← q_n; M(i) ← [p_h, p_n];
21             InsEdge(CT, [p_h, p_n], [r_h, r_n]);
22             StackPop(CP);
23 while (!StackEmpty(CP)) do
24     [q_h, q_n] ← StackPop(CP);
25     InsEdge(CT, [q_h, q_n], [p_h, p_n]);
26     [p_h, p_n] ← [q_h, q_n];
27 DefineRoot(CT, [p_h, p_n]);
```

3.3 Complexity analysis

Initially, we stated upper bounds for the data structures used in Algorithm 1 in order to perform the space complexity (SC) analysis. Let n denote the numbers of points in the WOS $(P, \prec, F)$, i.e., $n = |P|$, and let m denote the ordered set amplitude, i.e., $m = h_{max} - h_{min} + 1$. The size of vector *nnodes* corresponds to the domain's size of the mapping F, i.e., the SC of *nnodes* is $O(m)$. A WOS with n points can have a maximum of n components (when each component is composed of a single point). Therefore, the maximum size of the stacks is the number of points in the ordered set, i.e., the SC of

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

CP is $O(n)$. Now we consider the rooted tree. It is a tree with a root node, where every node has a single parent, but the root node does not have a parent. From that we have that all rooted trees with e nodes have $e-1$ edges. As a component tree is a rooted tree, the component tree of a WOS can have at maximum $n-1$ edges. The vectors inside the CT structure have a maximum of n elements ($n-1$ elements and the field for the root node), and therefore the SC of these vectors is $O(n)$. After this analysis of data structures used in Algorithm 1 we can state that the SC of the algorithm proposed is $O(max(n,m))$, i.e., it is linear.

Now considering a time complexity (TC) analysis we have that all functions implemented in the Algorithm 1 are atomic, i.e., they can be executed in $O(1)$. Then, to obtain the TC of Algorithm 1 we have to analyze the two main loops. The first loop (`for` in Line 2) is executed $n-1$ times while analyzing $n-1$ points. Although this loop uses a stack, the only insertion point into the stack CP is on the line 6. This fact confirms the SC $O(n)$ of the stack CP. Continuing on the loop presented in Line 2, we have an inner loop (`while`) guided by the stack CP in Line 10. This loop has an amortized TC at maximum $n-1$, since each time it is analyzed i.e., the nodes $[p_h, p_n]$, $[q_h, q_n]$ are compared) one edge will be inserted into the tree. The insertion will be done by either the loop itself (Line 13) or the conditions of the others two conditionals (when $p_h > q_h$ in Line 18 and when $p_h = q_h$ in Line 21). Then, the first loop has a linear TC,*i.e.*, i.e., $O(n)$. The second main loop (`while`) in Line 23 is executed whereas there are elements on the stack. Every time it is executed one element is removed from the stack. Thus this loop can be executed at maximum $n-1$ turns, i.e., the maximum stack size. Then, the second loop also has TC of $O(n)$. Therefore, the algorithm has a linear TC, i.e., $O(n)$.

3.4 Attributes

The component tree based approaches use measures extracted from the nodes of the tree structure. These measures are called attributes. Let $[h,n] \in \mathcal{C}(F)$. We define the height, the surface, and the volume attributes of the component $c_{h,n}$ as being:

$$\begin{aligned} ht(c_{h,n}) &= \max_{x \in c_{h,n}} \{F(x) - h_p\}, \\ s(c_{h,n}) &= \text{cardinality}(c_{h,n}), \\ v(c_{h,n}) &= \sum_{x \in c_{h,n}} (F(x) - h_p), \end{aligned}$$

respectively, where h_p is the parent weight of $c_{h,n}$. It is possible to compute, simultaneously, the component tree and these attributes of components without changing the time complexity of the algorithm.

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

4. Multithresholding

We now turn towards an application for the 1D component tree. Segmentation by multiple-threshold selection, or simply multithresholding, relies to the assumption that homogeneous regions present in the image can be detected in the histogram of the image. This segmentation method consists of selecting threshold levels by analyzing the histogram of the image. These thresholds determine histogram classes, and therefore any image pixel is classified according to the histogram class it belongs to.

In this work, we propose a method for multithresholding gray-level images in K levels. This method is based on the hypothesis that objects which appear on an image can be represented by salient classes present in the histogram of the image. These classes can be represented as the K most significative components extracted from component tree of the histogram of the image (an histogram is modelled as a WOS) - see Appendix for details.

The proposed method can be described in five main steps: 1) Histogram computation from gray-level image; 2) Computation of the Component tree of the histogram of the image; 3) Identification of the salient markers present in the histogram by means of the extraction of K most significative components of the histogram's component tree; 4) Histogram segmentation by watercourse transform (i.e., the dual of the watershed transform [1,3]) using the salient markers extracted in step 3; 5) Image segmentation by applying the segmented histogram to the original image.

Figure 3 shows the application of the proposed method to six classical images, namely: lena, goldhill, fruits, barbara, cameraman, and house. The first four images are of size 512×512 pixels and the last two of size 256×256 pixels. They are shown in the first column of Figure 3. We multithreshold all the histograms of image in five classes, and therefore the image in five levels. We chose the same number of classes for all images because all of them have at least five concise regions. On the other columns of Figure 3, we have, starting from the second column, the histograms of the input images, the five most important leaf components of the histograms of images and their not overlapped ascendant components coverage (as lighter as important - remark that the histograms are not smoothed), the segmented histograms in five classes (the classes are separated by vertical lines), and the output images with five levels (where the level for each histogram region was chosen as the nearest integer of the mean level in the respective histogram region).

We perform a quantitative comparison of our method, Kapur et al. [6], Khotanzad and Bouarfa [7], and Otsu [11] using a well-known objective measure, i.e., the Peak Signal to Noise Ratio ($PSNR$) [12]. The results are shown in Table 1. We observe that our method obtains $PSNR$ values close to the $PSNR$ value achieved by the other methods on four images: lena, barbara, cameraman and house. Indeed, we can see that our method segments the images in concise/homogeneous regions in 4 out of 6 images. However, in the goldhill and fruits images (second and third rows) the salient

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

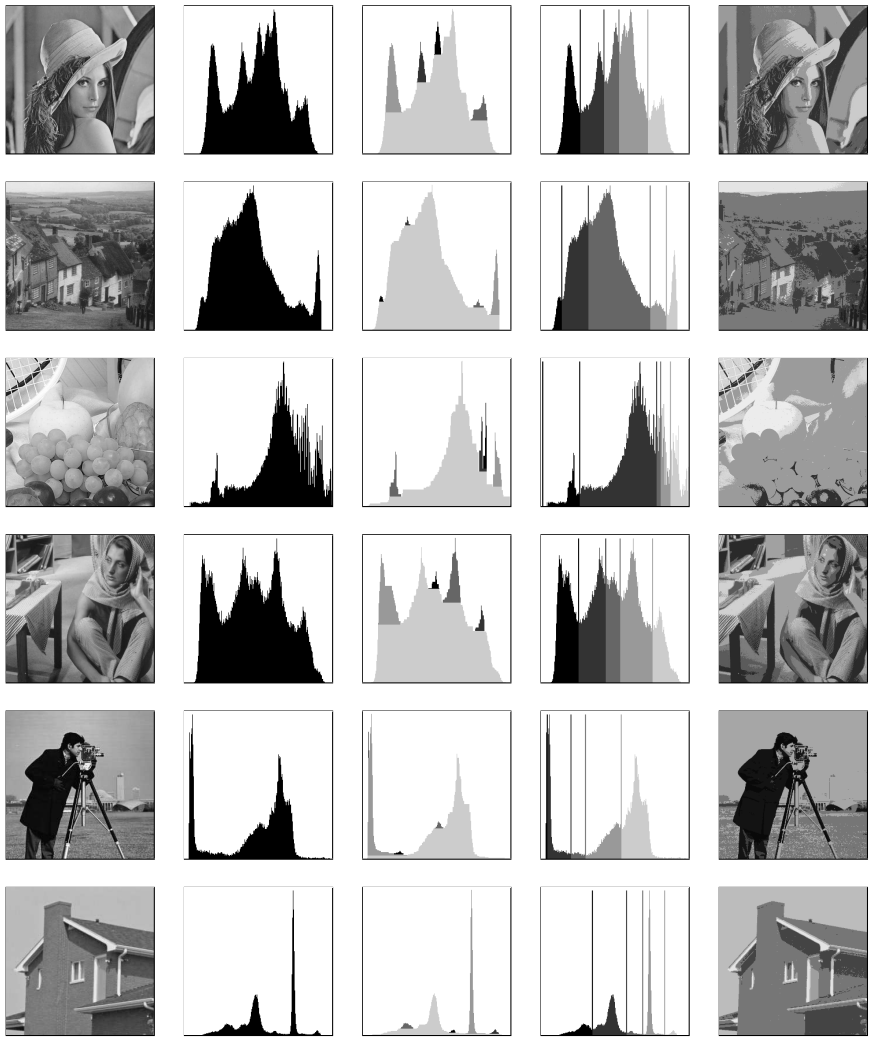

Figure 3. Real examples of classical images illustrating our multithresholding method. Columns from left to right: input original image, input image histogram, five (5) most important leaf components (maxima), segmented histogram in five (5) regions, and output segmented image in five (5) levels.

classes of the histograms of images are overlapped, and so our method is not suitable. In the cases where the histogram hypothesis holds, we argue that our method segments the images in regions more homogeneous than the other methods.

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

Table 1. PSNR for test images.

Images	Kapur	Khotanzad	Otsu	Our method
lena	25.3574	27.0722	28.2001	27.5316
goldhill	21.8978	22.4819	27.0583	21.6181
fruits	20.7996	22.5991	26.3987	19.6554
barbara	25.4540	26.1957	27.1348	26.4002
cameraman	19.3428	25.5831	27.8837	25.2907
house	20.1270	28.2576	29.3351	28.1030

5. Conclusion

In this paper we introduced, described, and illustrated a time and space linear complexity algorithm to compute the component tree for weighted ordered sets, i.e., 1D signals.

We proposed a new method for gray-level image multithresholding, based on the hypothesis that objects which appear on an image can be represented by salient classes present in a histogram of the image. These salient classes were modelled as the most significative components, where the importance corresponds to the volume attribute. Experiments showed that our method is competitive compared to classical ones when the hypothesis hold.

For future works, we plan to establish some methodology to select automatically the number of the most significative components present in the component tree, yielding an automatic multithresholding algorithm with respect to the number of classes in the output image. We also plan to extend our method to segment color images [4].

Acknowledgments

We would like to acknowledge support for this research from UFMG, CAPES/MEC, CNPq/MCT and FAPEMIG.

References

[1] S. Beucher and F. Meyer, *The morphological approach to segmentation: The watershed transform*, Mathematical Morphology in Image Processing, 1992, pp. 433–481.

[2] E. J. Breen and R. Jones, *Attribute openings, thinnings and granulometries*, Computer Vision and Image Understading **64** (1996), no. 3, 377–389.

[3] L. H. Croft and J. A. Robinson, *Subband Image Coding Using Watershed and Watercourse Lines of the Wavelet Transform*, IEEE Transactions on Image Processing **3** (1994), no. 6, 759–772.

[4] T. Geraud, P.-Y. Strub, and J. Darbon, *Color Image Segmentation based on Automatic Morphological Clustering*, IEEE ICIP (2001), pp. 70–73.

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10 –13, 2007, MCT/INPE, v. 1, p. 437–448.
http://urlib.net/dpi.inpe.br/ismm@80/2007/03.10.10.19

[5] R. Jones, *Component trees for image filtering and segmentation*, IEEE Workshop on Nonlinear Signal and Image Processing (1997).

[6] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, *A new method for Gray-Level Picture Thresholding Using the Entropy of the Histogram*, Computer Vision, Graphics, and Image Processing **29** (1985), 273–285.

[7] A. Khotanzad and A. Bouarfa, *Image Segmentation by a Parallel, Non-Parametric Histogram Based Clustering Algorithm*, Pattern Recognition **23** (1990), no. 9, 961–973.

[8] J. Mattes and J. Demongeot, *Efficient algorithms to implement the confinement tree*, DGCI (2000), LNCS, vol. 1953, pp. 392–405.

[9] J. Mattes, M. Richard, and J. Demongeot, *Tree representation for image matching and object recognition*, DGCI (1999), LNCS, vol. 1568, pp. 298–312.

[10] L. Najman and M. Couprie, *Building the component tree in quasi-linear time*, IEEE Transaction on Image Processing **15** (2006), no. 11, 3531–3539.

[11] N. Otsu, *A threshold selection method from grey-level histograms*, IEEE Transactions on Systems, Man and Cybernetics **9** (1979), no. 1, 41–47.

[12] M. Rabbani and P. W. Jones, *Digital Image Compression Techniques*, 1st, Society of Photo-Optical Instrumentation Engineers (SPIE), Bellingham, WA, USA, 1991.

[13] P. Salembier, A. Oliveras, and L. Garrido, *Anti-extensive connected operators for image and sequence processing*, IEEE Transaction Image Processing **4** (1998), no. 7, 555–570.

Appendix: Extracting the most significative components

In introduction, we have mentioned as simple use of the component tree the image filtering (removing nodes of the tree whose attribute value is below a given threshold). Here, we show a more advanced use for the component tree; determination of the K most significative components of the component tree. We hypothesized the volume attribute can model the importance of a salient region present in the histogram of the image. By using the tree, this task reduces to the search for the K nodes that have the largest attribute values and are not bound with each other (even transitively) by the inclusion relation. An algorithm to achieve this task is proposed in [10, Algorithm 3]. Its complexity is $O(sort(n) + n)$, where n is the number of points in the WOS and $sort(n)$ is the complexity of the sorting algorithm (it can be linear). Once the K most significative components are selected, we go back to the initial component tree and take as markers for the salient classes the leaf components corresponding to those K components. These markers are used in histogram segmentation by the watercourse transform.

Note that similar results could be obtained by performing attribute based operations using several volume threshold values.