Binary Partition Tree as an Efficient Representation for Image Processing, Segmentation, and Information Retrieval

Philippe Salembier, *Member, IEEE*, and Luis Garrido

Abstract—**This paper discusses the interest of** *Binary Partition Trees* **as a region-oriented image representation. Binary Partition Trees concentrate in a compact and structured representation a set of meaningful regions that can be extracted from an image. They offer a multi-scale representation of the image and define a translation invariant 2-connectivity rule among regions. As shown in the paper, this representation can be used for a large number of processing goals such as filtering, segmentation, information retrieval and visual browsing. Furthermore, the processing of the tree representation leads to very efficient algorithms. Finally, for some applications, it may be interesting to compute the Binary Partition Tree once and to store it for subsequent use for various applications. In this context, the last section of the paper will show that the amount of bits necessary to encode a Binary Partition Tree remains moderate.**

Keywords— **Nonlinear filtering, Connected operators, Mathematical morphology, Segmentation, Partition tree, Region Adjacency Graphs, Pruning strategy, Object recognition, Browsing, Information retrieval.**

I. Introduction

An increasing number of image processing applications rely on some type of region-based representations of images. The traditional image representation involving a rectangular array of pixels has major drawbacks. Its elementary unit, the pixel, provides an extremely local information. As a result, image processing at the pixel level has to face major difficulties in terms of scale: the scale of representation is most of the time far too low with respect to the interpretation or decision scale. Another drawback of pixel-based representation is the number of pixels. Most of the time, algorithms working at the pixel level are restricted to be very simple because they have to deal with a very large number of pixels.

Region-based representation of images potentially offers an attractive solution to this problem: the representation can be accurate; it involves a number of regions that is much lower than the number of original pixels and it can be considered as a first level of abstraction with regard to the raw information. Most of the time, region-based representations are created by merging similar pixels and by structuring the resulting regions within a Region Adjacency Graph (RAG). Even if some contributions have been published in the past about image processing on RAGs, it has to be recognized that they are not widely used in practice.

For practical applications, RAGs have two main drawbacks: They just describe one scale of the image and the connectivity between regions is not space invariant. In this paper, we propose a region-based representation called Binary Partition Tree that addresses these two drawback of RAGs. Binary Partition Trees represent images at various scales and the connectivity between regions is translation invariant since the tree encodes the relation between each region and only one of its neighboring regions. It is a 2-connectivity rule.

This study about Binary Partition Trees is related to several fields of image processing where region-based representation of images have proved to be useful: Filtering strategies involving connected operators, segmentation algorithms and content description.

- *Connected operators*: These filtering tools [17], [5], [16] are derived from mathematical morphology. They interact with the signal by means of specific regions called *flat zones* (largest connected components of the space where the image is constant). A connected operator is an operator that only merges flat zones. As it cannot introduce any new contour, it simplifies as well as preserves the contour information. The theory and applications of connected operators are rapidly progressing [19], [21], [18], [1], [4], [8]. However, one of the major drawbacks of classical operators is that they consider that objects composing the scene are either bright or dark image components (as a result, they simplify either bright or dark objects). This very crude assumption limits their usefulness for certain applications. In this paper, we discuss the interest of Binary Partition Trees to create new connected operators that do not suffer from this restriction.
- *Segmentation*: A large number of segmentation techniques such as region growing or watershed rely on iterative merging strategies [2], [10], [9]. These algorithms sequentially merge either pixels or regions. In practice, the class of rules used to control the merging process is restricted. Indeed, rules involving the global optimization of a criterion that has no specific property (such as increasingness) are not straightforward to deal with. In such cases, it is difficult to know, at a given time instant, if a particular merging will eventually lead to the optimization of the

Manuscript received July 17, 1998; revised September 8, 1999. This work was supported in part by France-Telecom/CCETT under contract 96ME22. The associate editor coordinating the review of this manuscript and approving it for publication was prof. Glenn Healey.

The Authors are with the Universitat Politècnica de Catalunya, Barcelona, Spain (e-mail: philippe@gps.tsc.upc.es).

Publisher Item Identifier S 1057-7149(00)02678-6.

global criterion and, for practical reasons, it is generally not realistic to memorize and to keep track of all merging possibilities so that they can be undone in future steps.

Recently, a study on the class of merging techniques has shown that a large set of interesting operators can be viewed either as filtering tools or as segmentation algorithms [7]. The definition of this class of operators relies on three notions:

1. *merging order* defines the notion of region homogeneity;
2. *merging criterion* characterizes the set of regions we are interested in;
3. *region model* defines how regions are represented.

In a large number of classical segmentation algorithms, the notions of merging order and merging criterion are combined into what is called the *segmentation criterion*. As will be seen in this paper, Binary Partition Trees strongly rely on a clear separation between merging order and merging criterion.

- *Content description*: There is currently a strong interest in defining content descriptor for information retrieval (activities within the MPEG-7 forum for example). If we consider a low level descriptor such as color or shape, one of the first issues to be faced is the selection of the scale at which the description has to be done. Since the queries done during the retrieval may deal with very different scales, it is not pertinent to *a priori* fix the description scale. As a result, a color (shape) descriptor should be able to describe the color (shapes) at multiple levels scales. The Binary Partition Tree exhibits this feature of multi-scale representation and could be considered as a basis to structure low-level descriptors dealing with color or shape information.

The organization of this paper is as follows: Section II gives the background material about segmentation algorithms based on merging strategies and connected operators. Binary Partition Trees are presented in section III. The application of Binary Partition Trees to information retrieval, segmentation and filtering is respectively discussed in sections IV, V and VI. For some applications (information retrieval for example), it may be useful to compute the Binary Partition Tree once and to store it. Section VII studies the cost in terms of bits of the representation. Finally, conclusions are given in section VIII.

II. Background on Connected Operators and Segmentation Algorithms Based on Merging Strategies

A. Segmentation algorithm based on merging techniques

A very large number of segmentation tools are based on a merging strategy. Let us consider as an example the general merging algorithm discussed in [7]. The algorithm works on a RAG, that is a set of nodes representing regions (connected components of the space), R_i, and a set of links defining the connectivity between regions. Note that a node of the graph can represent either a region, a flat zone or even a single pixel. A merging algorithm on this graph is simply a technique that removes some of the links and merges the corresponding nodes.

To completely specify a merging algorithm one has to specify three notions: the merging order (the order in which the links are processed), the merging criterion (each time a link is processed, the merging criterion decides if the merging has to be done or not), and the region model (when two regions are merged, the model defines how to represent the union). In the case of a *Region Growing* algorithm, the merging order is defined by a similarity measure between two regions (for example when color or gray level distance), the merging criterion states that the pair of most similar regions have to be merged until a termination criterion is reached (for example a given number of regions has been obtained) and the region model is usually the mean of the pixels gray levels or color values.

Note that the merging order (similarity between neighboring regions) is quite flexible and allows the definition of complex homogeneity models. By contrast, the merging criterion is very simple and crude: it states that the pair of most similar regions have always to be merged until the termination criterion is reached. As will be seen in the sequel, Binary Partition Trees allows us to increase the flexibility of the merging criterion while preserving the strength of the merging order.

B. Connected operators based on Max-tree representation

Connected operators [17], [3], [16] are filtering techniques derived from mathematical morphology that eliminate part of the image content while preserving the contour information of the remaining parts of the image. Their formal definition involves the notion of flat zones and their associated partitions. The flat zones, FZ_i, are the largest connected components of the space where the image is constant. As demonstrated in [19], the set of flat zones of an image, $I(\vec{p})$, creates a partition $P_{FZ}(I(\vec{p})) = \bigcup_i FZ_i$. Furthermore, a partition $P_A = \bigcup_i R_i^A$ is *finer* than a partition $P_B = \bigcup_j R_j^B$ if two pixels belonging to the same region of partition P_A always belong to the same region of partition P_B:

$$\forall i, \vec{p_1} \text{ and } \vec{p_2}, \qquad \vec{p_1}, \vec{p_2} \in R_i^A \Rightarrow \exists j \text{ such that } \vec{p_1}, \vec{p_2} \in R_j^B \tag{1}$$

Definition: A connected operator is an operator that only merges flat zones of the image. Mathematically, an operator, ψ, is connected if the partition of flat zones of its input is always finer than the partition of flat zones of its output.

In practice, connected operators are attractive because they simplify an image without introducing any new contour. An efficient way of creating and implementing a class of connected operators relies on a region-based representation called a *Max-Tree* [16]. The filtering strategy is illustrated by Fig. 1. The image is considered as a 3D relief and the first step is to construct a *Max-Tree* representation of the image. The nodes of the tree represent the binary connected components resulting from the thresholding of the original image at all possible gray level values. The leaves of the tree correspond to the maxima of the image. The links between the nodes describe how the flat zones may be merged. The tree structure is defined by the absolute gray levels of the flat zones. For example, the leaves of the tree correspond to the image maxima. The nodes along the

tree branches are ordered by the gray level values of the corresponding flat zones. Finally, the root node corresponds to the lowest gray level value. Note that there exist fast algorithms to construct the tree, see [16].

Starting from the leaves of the tree, each node is studied and a particular criterion is assessed for each node. If the criterion value is above (below) a given threshold, the node is preserved (removed). One of the most well known criterion is the *size*. It consists in counting the number of pixels of each node. If this number is below a given threshold, the node should be removed. The resulting operator is known as the area opening [20].

If the criterion is increasing, that is if the criterion value of a node is always smaller or equal to the criterion value of its parent node, then the algorithm defines a tree pruning strategy and, at the end of the pruning, the filtered image is reconstructed by stacking the connected components corresponding to the remaining nodes. Note in particular that the size criterion is increasing. If the criterion is not increasing, the definition of the pruning strategy is less straightforward. As discussed in [16], the non-increasingness of a criterion is most of the time a drawback that implies a lack of robustness of the operator (similar images may give different results). In [16], a solution relying on dynamic programming techniques (Viterbi algorithm) was proposed.

Connected operators can be viewed as merging techniques. With respect to the terminology introduced in section I, the merging order is defined by the absolute gray level value of the flat zones. It is therefore *a priori* fixed and does not depend on the actual merging that will be done. The merging criterion can be a very general feature of the region defined in relation with the flat zone (size, geometrical characteristics, texture, motion, etc). Note that it has no direct relation with the merging order. Finally, the model used to represent the union of two regions is their minimum gray level value. As can be seen, the merging order is very simple and does not provide any flexibility to define the notion of object homogeneity: objects are characterized as bright components.

The operator is said to be *anti-extensive* because the gray level value of each pixel of the filtered image is smaller than its value in the original image. In practice, this means that the operator simplifies the image by removing the bright components that do not fulfill a given criterion. To simplify dark components, the dual operator should be used. If $\psi(I(\vec{p}))$ is a connected operator applied on image $I(\vec{p})$, its dual operator is defined by: $\psi^*(I(\vec{p})) = -\psi(-I(\vec{p}))$.

Finally, note that the operator only acts on the regional maxima of the image. Once the regional maxima have been modified to fulfill the merging criterion, the operator does not modify the flat zones below this level. In the case of the area opening, all regional maxima of the filtered image have an area larger than the threshold. However, regional minima or transition areas can be of any size. Fig. 2 illustrates this situation with an area opening where the size threshold has been set to 500 pixels. In the filtered image, a large number of flat zones (minima or transition regions) have a size smaller than 500. We will find a similar issue with the Binary Partition Tree in section VI-B.

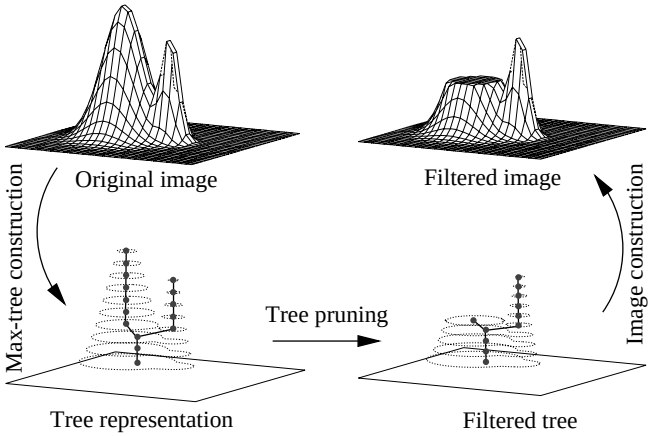


Fig. 1. Connected operator: filtering strategy

Fig. 2. Example of area opening. left: original image, right: area opening with size threshold equals to 500 pixels. The regional maxima of the filtered image have at least 500 pixels, but all minima and transition regions may be of smaller size.

C. Comparison between merging approaches

The idea of creating and processing Binary Partition Trees is an attempt to take benefit from the attractive features of both segmentation algorithms based on merging techniques and connected operators. Table I summarizes the main strong and weak points of both approaches. The strong element of segmentation algorithms based on merging techniques is their flexibility in the definition of the regions (merging order). By contrast, the strong feature of connected operators based on Max-Tree representation is the flexibility they offer to select the interesting regions (merging criterion). As a result, the creation of a Binary Partition Tree will be very similar to a segmentation algorithm and its processing will be similar to the strategy used by connected operators.

III. Binary Partition Trees

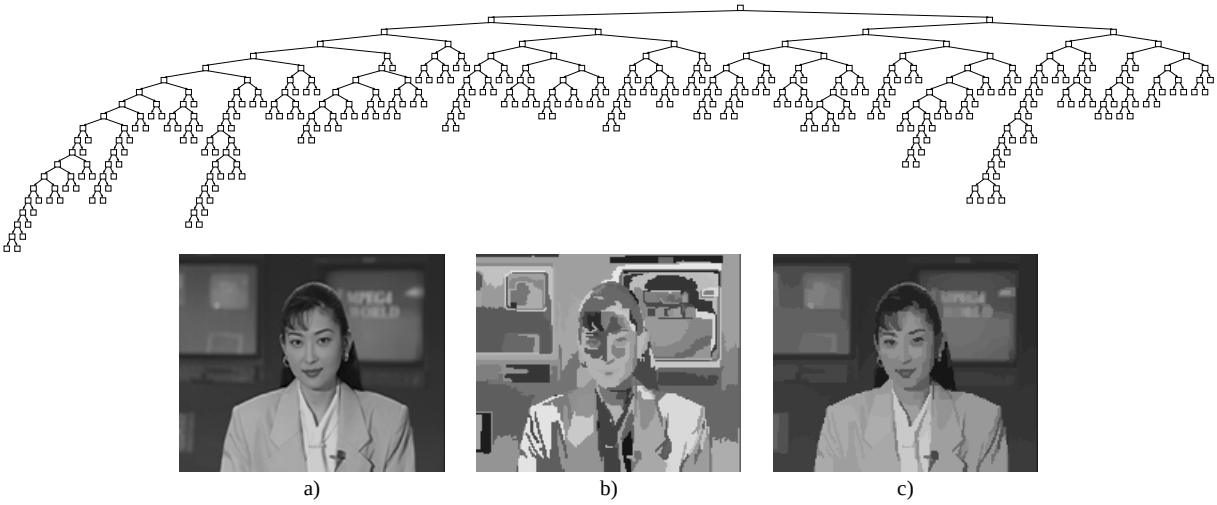
A. Definition

A Binary Partition Tree is a structured representation of the regions that can be obtained from an initial partition. An example is shown in Fig. 3. The leaves of the tree represent regions that belong to the initial partition shown in Fig. 3.b. The remaining nodes of the tree represent regions that are obtained by merging the regions represented by the two children of the node. The root node represents the entire image support. As can

Type of processing	Image representation	Merging Order	Merging criterion
Segmentation	RAG	Flexible (complex distance measure between regions)	Simple (merge until termination criterion is reached)
Connected operators based on Max-trees	Max-Tree or Min-Tree	Simple (absolute gray level value of flat zones)	Flexible (size, geometry, contrast, motion, etc.)

TABLE I

SUMMARY OF STRONG AND WEAK POINTS OF SEGMENTATION ALGORITHMS BASED ON MERGING TECHNIQUES AND CONNECTED OPERATORS.



a) b) c)

Fig. 3. Example of Binary Partition Tree (top). a) original image. b) initial partition with 200 regions. c) regions of the partition represented by their mean value.

be seen, the tree represents a fairly large set of regions at different scales. Large regions appear close to the root whereas small details can be found at lower levels. This representation should be considered as a compromise between representation accuracy and processing efficiency. Indeed, all possible merging of regions belonging to the initial partition (described by the RAG of the initial partition) are not represented in the tree. Only the most "likely" or "useful" merging steps are represented in the Binary Partition Tree. The connectivity encoded in the tree structure is binary in the sense that a region is explicitly connected to its sibling (since their union is a connected component represented by the father), but the remaining connections between regions of the original partition are not represented in the tree. Therefore, the tree encodes only part of the neighborhood relationships between the regions of the initial partition. However, as will be seen in the sequel, the main advantage of the tree representation is that it allows the fast implementation of sophisticated processing techniques.

B. Computation of Binary Partition Trees

The Binary Partition Tree should be created in such a way that the most "interesting" or "useful" regions are represented. This issue can be application dependent. However, a possible solution, suitable for a large number of cases, is to create the tree by keeping track of the merging steps performed by a segmentation algorithm based on region merging (see [10], [7] for example). In the following, this information is called the *merging sequence*. Starting from an initial partition which can be the partition of flat zones or any other pre-computed partition (for example the initial partition of 3.b), the algorithm merges neighboring regions following a homogeneity criterion until a single region is obtained. An example is shown in Fig.4. The original partition involves four regions and the algorithm merges them in three steps. In the first step, the pair of most similar regions, 1 and 2, are merged to create region 5. Then, region 5 is merged with region 3 to create region 6. Finally, region 6 is merged with region 4 and this creates region 7 corresponding to the region of support of the whole image. In this example, the merging sequence is: $(1,2)|(5,3)|(6,4)$. This merging sequence progressively defines the Binary Partition Tree as shown in Fig. 4.

Following the terminology of section II-A, it can be seen that, since the segmentation algorithm keeps on merging regions until a single region is obtained, the tree construction only makes use of the merging order and the region model whereas the merging criterion is trivial. In order to create the Binary Partition Trees used to illustrate the processing examples discussed in this paper, we have used a merging algorithm following the color homogeneity criterion described in [7]. Let us define the merging order $O(R_1, R_2)$ and the region model M_R:

- *Merging order:* At each step the algorithm looks for the pair of most similar regions. The similarity between regions R_1 and R_2 is defined by the following expression:

$$O(R_1, R_2) = N_1||M_{R_1} - M_{R_1 \cup R_2}||_2 + N_2||M_{R_2} - M_{R_1 \cup R_2}||_2 \quad (2)$$

  where N_1 and N_2 are the numbers of pixels of regions R_1 and R_2 and $||.||_2$ denotes the $\mathcal{L}_2$ norm. M_R represents the model for region R. It consists of three constant values describing the YUV components. The interest of this merging order, compared to other classical criteria, is discussed in [7].
- *Region model:* As mentioned previously, each region is modeled by a constant YUV value. M_R is therefore a vector of 3 components. During the merging process, the YUV components of the union of 2 regions, R_1 and R_2, are computed as follows [7]:

$$\begin{array}{lll} \text{if } N_1 < N_2 & \Rightarrow & M_{R_1 \cup R_2} = M_{R_2} \\ \text{if } N_2 < N_1 & \Rightarrow & M_{R_1 \cup R_2} = M_{R_1} \\ \text{if } N_1 = N_2 & \Rightarrow & M_{R_1 \cup R_2} = (M_{R_1} + M_{R_2})/2 \end{array} \quad (3)$$

  As can be seen, if $N_1 \neq N_2$, the model of the union of two regions is equal to the model of the largest region.

It should be noticed that the homogeneity criterion has not to be restricted to color. For example, if the image for which we create the Binary Partition Tree belongs to a sequence of images, motion information should also be used to generate the tree: in a first stage, regions are merged using a color homogeneity criterion, whereas a motion homogeneity criterion is used to merge regions in the second stage [6]. Fig. 5 shows an example of the *Foreman* sequence. In Fig. 5.A, the Binary Partition Tree has been constructed exclusively with the color homogeneity criterion described above. In this case, it is not possible to concentrate the information about the foreground object (head and shoulder regions of Foreman) within a single sub-tree. For example, the face mainly appears in the sub-tree hanging from region A, whereas the helmet regions are located below region D. In practice, the nodes that are close to the root have no clear meaning because they are not homogeneous in color. Fig. 5.B presents an example of Binary Partition Tree created with color and motion criteria. The nodes appearing in the lower part of the tree as white circles correspond to the color criterion, whereas the dark squares correspond to a motion criterion. The motion criterion is formally the same as the color criterion except that the YUV color distance is replaced by the YUV Displaced Frame Difference. As can be seen, the process starts with the color criterion as in Fig. 5.A and then, when a given Peak Signal to Noise Ratio (PSNR) is reached, it changes to the motion criterion. Using motion information, the face and the helmet now appear as a single region E.

Additional information of previous processing or detection algorithms can also be used to generate the tree in a more robust way. For instance, a mask of an object included in the image can be used to impose constraints on the merging algorithm in such a way that the object itself is represented with only one node in the tree. Typical examples of such algorithms are face, skin, character or foreground object detection. An example is illustrated in Fig. 6. Assume for example that the original *Children* image sequence has been analyzed so that the masks of the two foreground objects (children) are available. If the merging algorithm is constrained to merge regions within each mask before dealing with the remaining regions, the region of support of each mask will be represented as a single node in the resulting Binary Partition Tree. In Fig. 6, the nodes corresponding to the background and the two foreground objects are represented by squares. The three sub-trees further decompose each object into elementary regions.

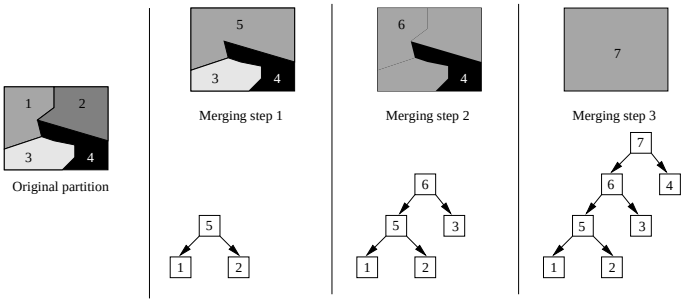
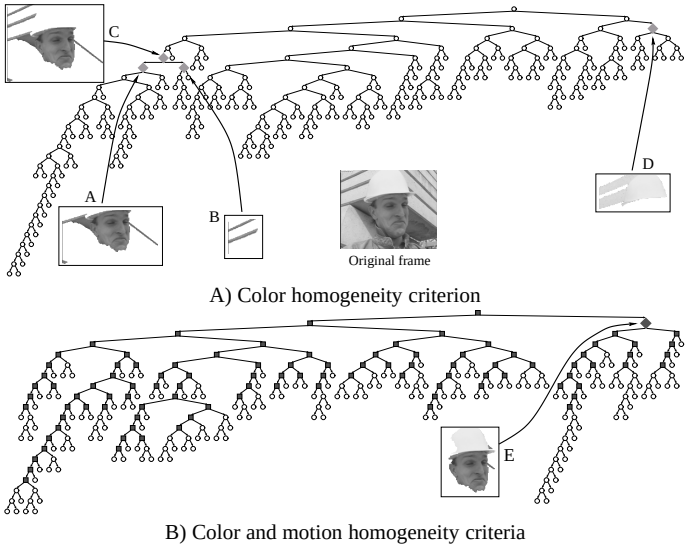


Fig. 4. Example of Binary Partition Tree creation with a region merging algorithm



A) Color homogeneity criterion

B) Color and motion homogeneity criteria

Fig. 5. Examples of creation of Binary Partition Tree

In the sequel, we assume that the Binary Partition Tree has been created and we discuss the interest of this representation for various image processing tasks.

IV. Information Retrieval

A. Detection/Recognition tools

One of the key issues of information retrieval is to be able to efficiently identify objects corresponding to a given query. In the sequel, we describe a simple example of circular objects detection but the approach can be used for any criterion that has to be optimized over a set of regions. The circularity is defined as the squared perimeter divided by the area and is minimum for circular objects for which it is equal to 4π.

In order to detect the presence of circular objects, the Bina-

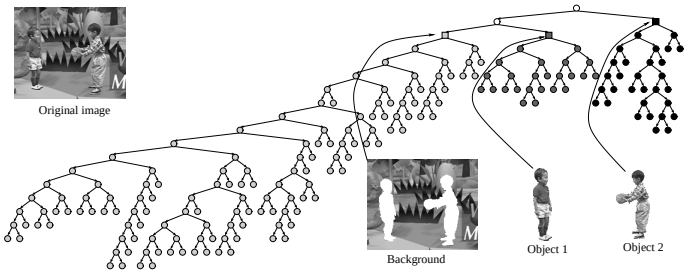


Fig. 6. Example of partition tree creation with restriction imposed by object masks

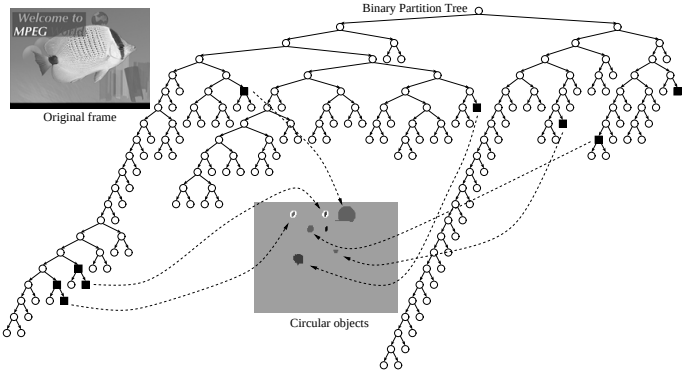


Fig. 7. Example of detection of circular objects (initial partition with 100 regions).

ry Partition Tree is a particularly attractive representation since it proposes a reduced number of regions which are assumed to be the most homogeneous for different scales. Suppose that a size descriptor and a perimeter descriptor have been assigned to each node of the tree. Now consider the example presented in Fig. 7. In this case, the initial partition is made of $N = 100$ regions. As a result, the set of regions represented by the Binary Partition Tree equals $2N - 1 = 199$. So, the algorithm has to measure the circularity of only 199 regions. In the tree of Fig. 7, the circular regions have been represented by dark squares. A spatial representation of these regions is also show. They correspond to the letters "o" appearing in "Welcome to the MPEG World" message and to various circular components of the image. This approach can easily be extended to deal with generic shape recognition tasks. In this case, a general shape descriptor should be assigned to each node.

Note that the tree structure introduces a notion of scalability in the description itself. Indeed, one region is described by its own descriptors but also by the set of descriptors of its children. Assume, for example, that the luminance information of a region is described by a constant value. This is a very crude approximation. However, for a given region R, this approximation can be improved if the set of luminance descriptors of all the regions included in R are considered. The most accurate description is obtained when the descriptors of the leaves of the sub-tree hanging from R are considered. Finally, the tree structure is also an attractive solution to encode the descriptors. Indeed, the tree can be used to take benefit from the correlation and inheritance relationships between descriptors.

B. Visual browsing

The previous example involves the evaluation and the optimization of a local criterion independently on each region. By contrast, the following browsing example discusses an approach where the optimization is global on the entire tree structure.

Browsing is an important functionality for information retrieval. Most of the time, the user would like to have a rough idea on the query results. The goal is not to visualize a high quality image, but simply to be able to discard or not the query result. This issue is not trivial if the transmission channel between the client and the server has a reduced bandwidth. A Binary Partition Tree is also very attractive representation to deal with such a functionality. Indeed, as shown in [15], partition trees in general are appropriate to define optimum pruning strategies in the rate/distortion sense with restriction on the rate to be transmitted or the distortion of the coded image. Let us discuss this approach.

Assume that the visual information is transmitted by selecting some regions described by the Binary Partition Tree and by sending their contours plus a constant color value per region. The definition of the coding strategy consists in finding the best partition created by regions, R_i, contained in the tree such that the global distortion, $\mathcal{D}$, is minimized and the rate or coding cost, $\mathcal{C}$ (in bits), is lower than a given budget. Note that in this section, we assume that the goal is to minimize the distortion under a rate constraint. It is also possible to minimize the rate under a distortion constraint. The only modification would be to exchange the roles of $\mathcal{D}$ and $\mathcal{C}$.

As discussed in [15], the first step consists in analyzing the rate $\mathcal{C}(R_i)$ and distortion $\mathcal{D}(R_i)$ associated to each region R_i in the tree. The computation of the distortion is rather straightforward and the Squared Error between the original and coded frames can be used:

$$\begin{aligned}\mathcal{D}(R_i) &= \sum_{\vec{p}\in R_i}(I^Y(\vec{p}) - M^Y_{R_i})^2 \\ &+ \alpha((I^U(\vec{p}) - M^U_{R_i})^2 + (I^V(\vec{p}) - M^V_{R_i})^2)\end{aligned} \quad (4)$$

where the parameter α is used to balance the luminance and the chrominance distortion. In this equation, $I^Y(\vec{p})$, $I^U(\vec{p})$ and $I^V(\vec{p})$ denote the luminance and chrominance components of the pixel $\vec{p}$ whereas $M^Y_{R_i}$, $M^U_{R_i}$ and $M^V_{R_i}$ represent the components of the region model.

The situation is more complex for the computation of the rate $\mathcal{C}(R_i)$. Indeed the rate associated to a region is composed of 24 bits for the color information plus a certain number of bits for the shape information. In practice, most of the contour coding techniques for partition do not process independently each region because a contour is always shared by two regions. In order to avoid the problem of optimization with complex dependencies between regions, we have used the following approximation of the contour rate: an average number of bits necessary to encode a contour point has been estimated. This number is denoted by $BPCP$ (Bits Per Contour Point). We have assumed that the contour rate assigned to a region is equal to this average figure multiplied by the region perimeter, ∂R_i, divided by two (each contour point is shared by two regions). As a result, the rate per region is given by:

$$\mathcal{C}(R_i) = 24 + (BPCP \;\; \partial R_i)/2 \quad (5)$$

```
Distortion: D; Rate: C; Budget rate: C0; Lagrange parameter: λ;
λl = 0;                          /* Compute D and C for a very low λ */
BottomUpAnalysis(Input: λl, Output: Cl, Dl);
if Cl < C0 then { no solution; exit;}
Cl = Cl; Dl = Dl;
λh = 10^20;                      /* Compute D and C for a very high λ */
BottomUpAnalysis(Input: λh, Output: Ch, Dh);
if Ch > C0 then { no solution; exit;}
Ch = Ch; Dh = Dh;
do {                                        /* Find the optimum λ value */
      λ = (Dl − Dh)/(Ch − Cl);
      BottomUpAnalysis(Input: λ, Output: C, D);
      if C < C0 then { Ch = C; Dh = D; }
                else { Cl = C; Dl = D; }
} until (C ≈ C0)
```

Fig. 8. Algorithm for Rate-distortion optimization

The rate/distortion optimization itself relies on the technique discussed in [11], [12], [13]. The problem can be formulated as the minimization of the distortion $\mathcal{D} = \sum_{R_i} \mathcal{D}(R_i)$ of the image with the restriction that the total rate $\mathcal{C} = \sum_{R_i} \mathcal{C}(R_i)$ is below a given budget $\mathcal{C}_0$. Note that both the rate and the distortion have to be additive over the regions. It is well known that this problem can be reformulated as the minimization of the Lagrangian: $\mathcal{D} + \lambda\mathcal{C}$ where λ is the so-called Lagrange parameter. Both problems have the same solution if we find λ^* such that $\mathcal{C}$ is equal (or very close) to the budget. Therefore, the problem consists in using the Binary Partition Tree to find a set of regions creating a partition such that:

$$\text{Min } \mathcal{D} + \lambda^*\mathcal{C} \text{ , with } \lambda^* \text{ such that } \mathcal{C} \approx \mathcal{C}_0 \tag{6}$$

Assume, in a first step, that the optimum λ^* is known. The definition of the best partition can be done by a bottom-up analysis of the Binary Partition Tree. To initialize the process, all the leaves of the Binary Partition Tree are assumed to belong to the optimum solution. Then, one checks if it is better to code the area represented by two sibling nodes as two independent regions $\{X_1, X_2\}$ or as a single region X (the common parent node of X_1 and X_2). The selection of the best choice is done by comparing the Lagrangian of X with the sum of the Lagrangians of X_1 and X_2:

$$\begin{cases} \text{If } \mathcal{D}(X) + \lambda^*\mathcal{C}(X) \leq \sum_{i=1,2} \mathcal{D}(X_i) + \lambda^*\mathcal{C}(X_i) \\ \text{then,} \quad \text{encode } X \text{ as a single region} \\ \text{else,} \quad \text{encode } X_1 \text{ and } X_2 \text{ as 2 independent regions} \end{cases} \tag{7}$$

The best encoding strategy (encode X as itself or as the union of its children) is stored in X together with the corresponding Lagrangian value. The procedure is iterated up to the root node and defines the best coding strategy.

In practice, of course, the optimum λ^* parameter is not known and the previous bottom-up analysis of the Binary Partition Tree is embedded in a loop that searches for the best λ parameter. The computation of the optimum λ parameter can be done with a gradient search algorithm. The algorithm starts with a very high value λ_h (10^{20}) and a very low value λ_l (0) of λ. For each value of λ, the bottom-up optimization procedure described above is performed. This results in two partitions that should correspond to rates $\mathcal{C}_h$ and $\mathcal{C}_l$ respectively below and above the budget. If none of these rates is close enough to the budget, a new Lagrange parameter is defined as $\lambda = (\mathcal{D}_l - \mathcal{D}_h)/(\mathcal{C}_h - \mathcal{C}_l)$. The procedure is iterated until the rate gets close enough to the budget. The algorithm is described in pseudo-code in Fig. 8. In practice, the optimum λ^* parameter is found with few iterations, typically less than ten iterations. The bottom-up analysis itself is not expensive in terms of computation since the algorithm has simply to perform the comparison of equation 7 for all nodes of the tree.

Fig. 9 shows a Binary Partition Tree corresponding to a initial partition involving 100 regions. If this original image would have to be transmitted for browsing, and assuming that a coding strategy involving the coding of the contours with chain code and of a constant color value for each region is used, the cost in term of bits would be equal to 14000. With respect to the original image in QCIF format, this strategy already provides a reasonable compression factor: the original image involves 176*144*24 = 608256 bits and the corresponding compression factor is equal to 43. However, for visualization purposes, this strategy is not optimum. We show in Fig. 9 three examples of coded images at 3000, 7000 and 11000 bits. As can be seen, the image coded at 11000 bits is visually equal to the initial partition image. In the case where the transmission rate is very low, higher compression factors may be used while allowing the user to have an idea about the image content. Two coding strategies are

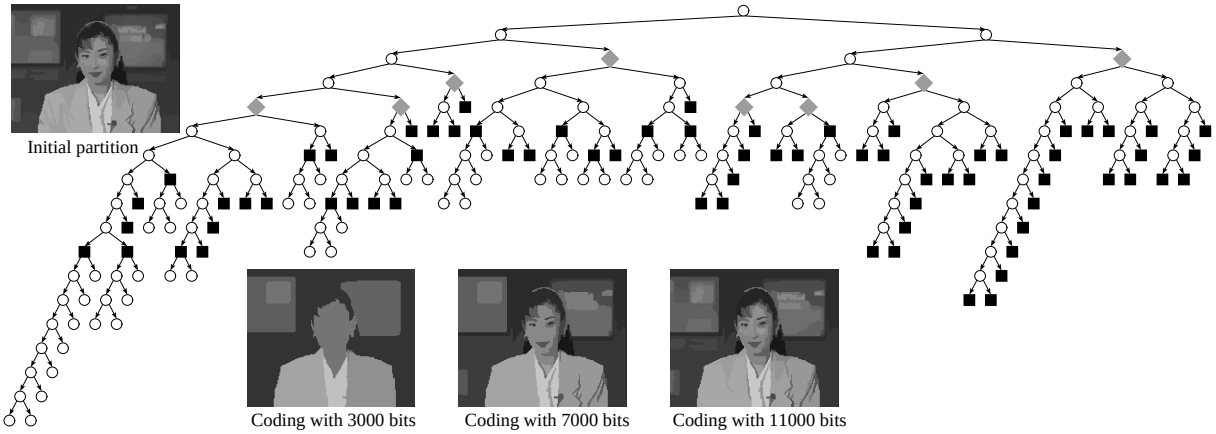


Fig. 9. Examples of pruning for visualization: Black squares (gray rhombus) in the tree define the optimum solution in the rate-distortion sense for 11000 bits (3000 bits).

shown in the tree representation of Fig. 9. The first one corresponds to the optimum solution for 11000 bits and is shown with dark squares. The second one shown in gray rhombus give the optimum solution at 3000 bits. As can be seen, for low bit rates, the algorithm selects regions close to the root of the tree. For higher bit rates, a large number of small regions providing details about the image content can be transmitted. Finally, Fig. 10 gives the complete rate/distortion curve. One can see the evolution of the visual quality as a function of the number of regions introduced in the partition.

V. Segmentation

A. Direct approach

The Binary Partition Tree representation is particularly suitable to generate segmentation results. Two examples of segmentation strategies are discussed in the following. The first one (Fig. 11) is a segmentation following a "direct" approach. It consists in merging the regions that are the most similar until a termination criterion is reached. Examples of termination criteria are the number of regions or the PSNR between the original and the modeled images. Note that, with respect to the segmentation framework described in section II-A, the creation of the Binary Partition Tree has fixed the merging order. The merging criterion (here a termination criterion) is used in a second phase without modifying the order. Therefore, this approach is similar to the one used for connected operators.

The Binary Partition Tree representation is particularly suitable for the "direct" segmentation approach. Indeed, the merging sequence, which has been used to create the tree, defines the similarity between regions. It assumes that the most similar regions are merged first. Therefore, the segmentation can be computed by progressively deactivating the nodes following the merging sequence until the termination criterion is met (required number of regions or PSNR). Using the same initial partition as the one of Fig. 9, we show four examples of segmentation in Fig. 11. In all cases, the termination criterion is defined by the number of regions.

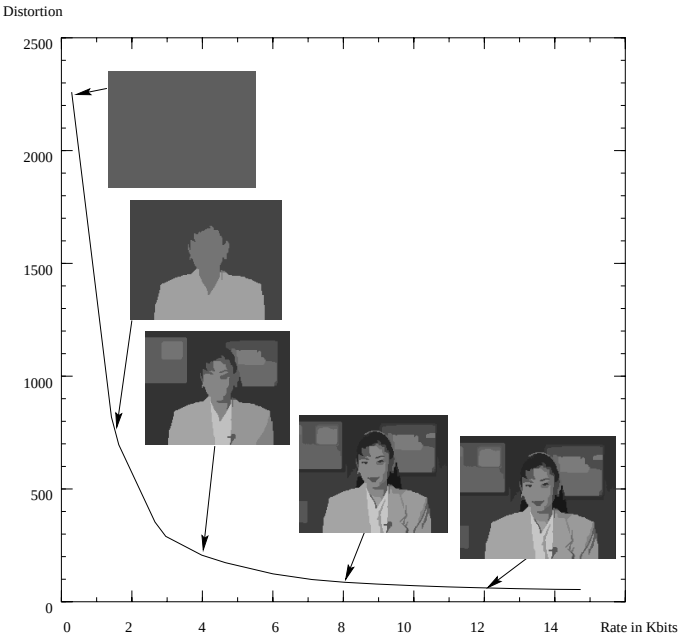


Fig. 10. Rate/distortion curve for the partition tree of Fig. 9

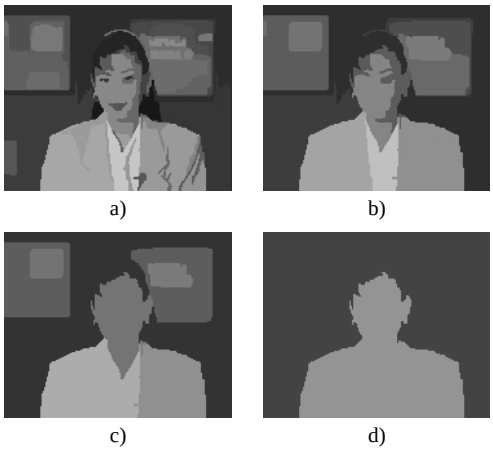


Fig. 11. Four examples of "direct" segmentation: a) 50 regions, b) 15 regions, c) 8 regions, d) 2 regions

B. Marker & propagation approach

An alternative approach to the direct segmentation is the so-called morphological [9] or "Marker & propagation" approach. The strategy consists, first, in "marking" (defining with markers) the interior of the regions to be segmented and, second, in performing a propagation of these markers to eventually define the regions contours. This second step can be viewed as the definition of the zone of influence of each marker. Let us mention, that depending on the application, the markers can be computed automatically [9], [14] or manually.

Propagation processes based on similarity between neighboring regions can be easily implemented in the Binary Partition Tree structure. Let us describe this propagation on a simple example. Fig. 13.a shows the simple image made of four flat zones used in Fig. 4. The Binary Partition Tree indicates that regions 1 and 2 are the most similar. Once merged, their closest region is region 3. Finally, region 4 is the most dissimilar (As can be seen, its gray level value is quite different from the values of other regions). Consider now two markers, A and B, that have to be propagated by merging with neighboring regions. Let us mark the two corresponding nodes on the tree (see 13.b). By construction of the Binary Partition Tree, the most similar neighboring region with respect to a given marker is represented by its sibling and the result of the merging is represented by the marker's parent. Therefore, a marker associated to a node is propagated to its parent. Of course this propagation can only be done if the sibling is not in conflict with the marker, that is if none of the sibling's descendants has been assigned to a different marker. In the example of Fig 13.c, the first marker to be propagated is the marker A corresponding to region 2. It is propagated to its parent, that is region 5. At this level, the propagation has to stop because there is a conflict between the marker of region 5 (marker A) and the marker of region 3 (marker B). Finally a top-down propagation of markers is done so that children nodes have the same label as their parents (see Fig12.d).

Fig. 12 precisely describes an algorithm performing the propagation on the Binary Partition Tree structure. The algorithm works in three main steps: first, assignment of markers to leaf nodes, second, bottom-up propagation of the markers to parents if there are no conflicts between labels and, third, top-down propagation of labels so that children nodes have the same label as their parents. The bottom-up and top-down propagations are controlled by a parameter, `level`, describing the location of the node within the tree hierarchy (the root is at level 0, its children at level 1, etc.). Moreover, the algorithm makes use of the functions `Parent(Node)`, $\texttt{Child}_{\texttt{left}}\texttt{(Node)}$, $\texttt{Child}_{\texttt{rigth}}\texttt{(Node)}$ to access the nodes that are directly related to a given `Node`.

Note that this propagation process does not necessarily assign a marker to all leaves of the tree. In our example, region 4 remains without label. This situation means that the similarity between regions defined by markers A and B is higher than any combination with region 4. As said above, region 4 is indeed the most dissimilar. The propagation process is controlled in the sense that the algorithm does not blindly assign all regions to a marker. In most cases, this control is an attractive feature of the Binary Partition Tree representation. However, for some specific applications, one would like to use a propagation algorithm that actually creates as many regions as markers. For this kind of applications, the problem is to merge unassigned regions to one of the closest neighboring region that has been reached by a marker during the propagation. Here again, this task is easily solved with the help of the Binary Partition Tree. Indeed, consider an unassigned region X that has a sibling in conflict (region 4 in fig. 13). Its closest neighboring region that has been reached by a marker is one of the descendants of its sibling. Indeed, the set of sibling descendants is the set of closest regions with respect to X. Furthermore, at least one of the descendant has been assigned to a marker. Otherwise the propagation process could not have been stopped before reaching the sibling of X. Therefore, starting from the sibling of X, one has simply to scan all the descendants until one region that is, at the same time, neighbor of X and assigned to a marker, is found. Note that in some cases, several regions fulfill this criterion. This situation is illustrated on Fig. 13 where region 4 was unassigned and two regions are at the same time neighbor of region 4 and assigned to a marker: $R_1 = 1 \bigcup 2$ (assigned to marker A) and $R_2 = 3$ (assigned to marker B). In this situation, the most simple solution consists in arbitrarily selecting one of these regions. Of course, if necessary, specific rules based on similarity or geometrical criteria can be designed.

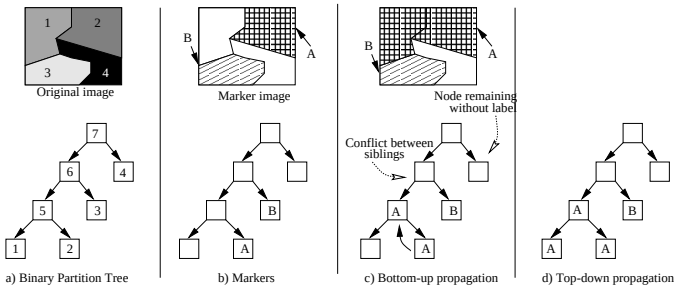


Fig. 13. Propagation process on the Binary Partition Tree

A complete example is shown in Fig. 14. In this example, we assume that a user has defined two markers (dark and gray). The first step is to assign the markers to the leaves of the tree (Fig. 14.Top). Then, the propagation process creates three connected components (Fig. 14.Bottom). The two first ones correspond to the zones of influence of the markers whereas the last one remains without label because it is judged as being "too different". As can be seen, the face and shoulders regions defined by the markers have been properly segmented and the background has been merged with none of these regions.

VI. Filtering Tools

A. Pruning strategy

In this section, our objective is to define new connected operators, in particular self-dual operators. Self-dual operators are operators such that: $\psi(I(\vec{p})) = -\psi(-I(\vec{p}))$. As a result they process similarly bright and dark image components. First, let us recall the classical strategy used for connected operators. As discussed in section II-B, the approach involves: first, the creation of a tree representation of the image (*Max-tree* or *Min-*

```
N_l:  Set of nodes at level l;
Node->label:  label of marker;
Node->conflict:  States if the two sub-trees of node correspond to differ-
ent markers;

                                                        /* Initialization */
for all nodes in the tree:  { Node->label = void; Node->conflict = void;}
Assign a label to leaves that overlap with a marker;

                                          /* Bottom-up propagation of markers */
for (level = level_max; level ≥ 0; level--){
for all nodes Node ∈ N_level that are non-leaf nodes {

                                                   /* Analyze the conflicts */
      if ( (Child_left(Node)->label != void) && (Child_right(Node)->label !=
void) &&
           (Child_left(Node)->label != (Child_right(Node)->label))
           Node->conflict = CONFLICT
      if ( (Child_left(Node)->conflict == CONFLICT) ||
           (Child_right(Node)->conflict == CONFLICT))
           Node->conflict = CONFLICT

            /* If there is no conflict, propagate the label from children */
      if (Node->conflict == void){
           if (Child_left(Node)->label != void)
                Node->label = Child_left(Node)->label;
           else if (Child_right(Node)->label != void)
                Node->label = Child_right(Node)->label;
      }
}}

     /* Propagate label from parent to children (in absence of conflicts) */
for (level = 1; level ≤ level_max; level++){
for all nodes Node ∈ N_level {
if ((Parent(Node)->conflict != CONFLICT) && (Node->conflict != CONFLICT))
      Node->label = Parent(Node)->label;
}}
```

Fig. 12. Algorithm for marker propagation in the Binary Partition Tree structure

tree[1]), second, the assessment of a criterion for each node of the tree and third, the definition of a tree pruning strategy. The pruning defines a new partition and each region is represented by a constant value (minimum in the case of *Max-tree* and maximum in the case of *Min-tree*).

Mathematically, a criterion C assessed on a region R is said to be increasing if the following property holds:

$$\forall R_1 \subseteq R_2 \Rightarrow C(R_1) \leq C(R_2) \tag{8}$$

[1] the *Min-tree* can be defined as the *Max-tree* of the opposite of the original image.

Assume that all nodes corresponding to regions where the criterion value is lower than a given threshold should be removed by merging. If the criterion is increasing, the pruning strategy is straightforward: merge all nodes that should be removed. It is indeed a pruning strategy since the increasingness of the criterion guarantees that if a node has to be removed all its descendants have also to be removed. An example of Binary Partition Tree with increasing decision criterion is shown in Fig. 15. The criterion used to create this example is the size, measured as the number of pixels belonging to the region.

If the criterion is not increasing, the pruning strategy is not

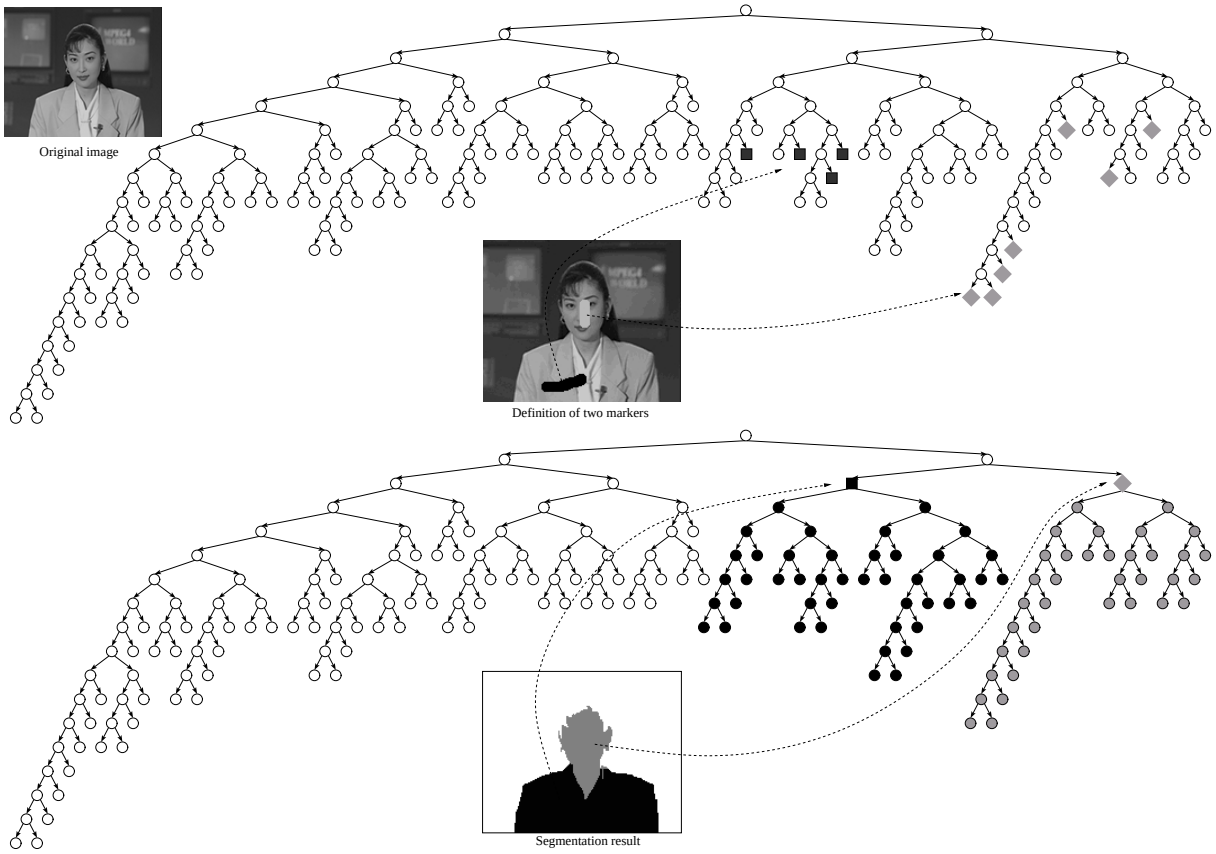


Fig. 14. Example of segmentation with marker & propagation strategy. Top: Binary Partition Tree where the leaves intercepted by markers have been indicated. Bottom: Result of the propagation process.

straightforward since the descendants of a node to be removed have not necessarily to be removed. An example of such criterion is the region perimeter. Fig. 16 illustrates this case. If we follow either Path A or Path B in Fig. 16, we see that there are some oscillations of the remove/preserve decisions. In practice, the non-increasingness of the criterion implies a lack of robustness of the operator. For example, similar images may produce quite different results or small modifications of the criterion threshold involve drastic changes on the output. In [16], a similar issue is discussed in the context of *Max-tree* representation for anti-extensive connected operators. The proposed solution consists in applying a transformation on the set of decisions. The transformation should create a set of increasing decisions while preserving as much as possible the decisions defined by the criterion. This problem may be viewed as dynamic programming issue that can be efficiently solved with a Viterbi algorithm. A similar solution is used here for binary partition trees.

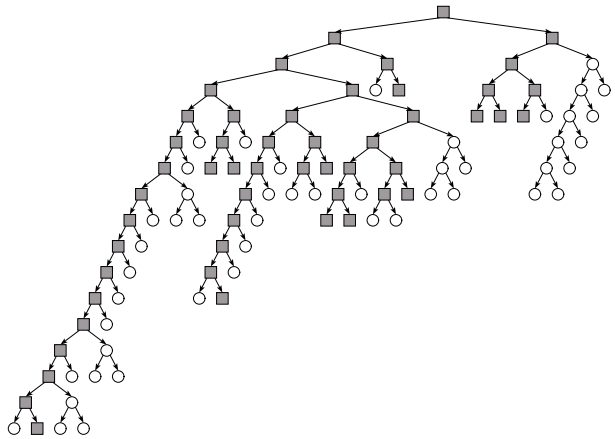


Fig. 15. Example of increasing criterion: size. The pruning strategy is straightforward since the increasingness of the size criterion guarantees that if a node has to be removed all its descendants have also to be removed. Gray squares: nodes to be preserved, white circles: nodes to be removed.

The trellis on which the Viterbi algorithm [22] is applied is illustrated in Fig. 17. It has the same structure as the Binary Partition Tree except that each node N_k of the Binary Partition Tree corresponds to two trellis states: *preserve* N_k^P and *remove* N_k^R. The two states of each child node are connected to the two states of its parent. However, in order to avoid non-increasing decisions, the *preserve* state of a child is not connected to the *remove* state of its parent. As a result, the trellis structure guarantees that if a node has to be removed its children have also to be removed. The cost associated to each state is used to compute the number of modifications the algorithm has to do to create an increasing set of decisions. If the criterion value states that the node of the Binary Partition Tree has to be removed, the cost associated to the *remove* state is equal to zero (no modification) and the cost associated to the *preserve* state is equal to one (one modification). Similarly, if the criterion value states that the n-

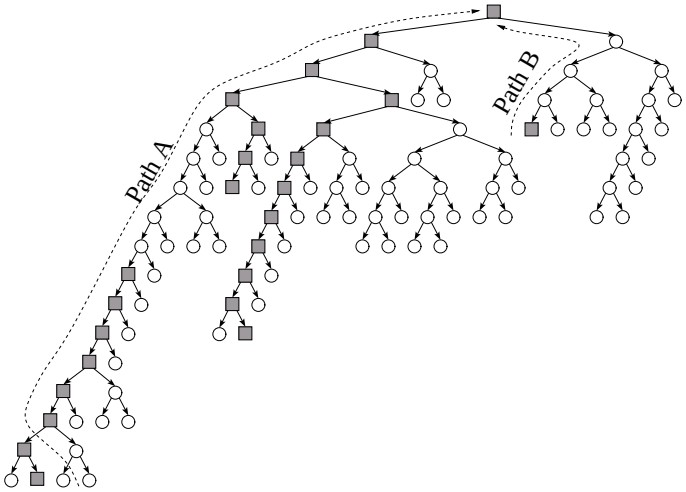


Fig. 16. Example of non-increasing criterion: perimeter. The pruning strategy is not straightforward since the criterion does not guarantee that if a node has to be removed all its descendants have also to be removed (see decisions along path A or path B). Gray squares: nodes to be preserved, white circles: nodes to be removed.

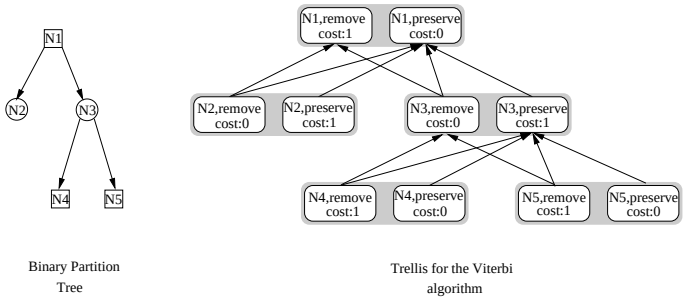


Fig. 17. Creation of the trellis structure for the Viterbi algorithm. A circular (square) node on the Binary Partition Tree indicates that the criterion value states that the node has to be removed (preserved). The trellis on which the Viterbi algorithm is run duplicates the structure of the Binary Partition Tree and defines a *preserve* state and a *remove* state for each node of the tree. Paths from *remove* states to child *preserve* states are forbidden so that the decisions are increasing.

ode has to be preserved, the cost of the *remove* state is equal to one and the cost of the *preserve* state is equal to zero[2]. The cost values appearing in Fig. 17 assume that nodes N_1, N_4 and N_5 should be preserved and that N_2 and N_3 should be removed. The goal of the Viterbi algorithm is to define the set of decisions such that:

$$\text{Min} \sum_k \text{Cost}(N_k) \text{ such that the decisions are increasing} \quad (9)$$

To find the optimum set of decisions, a set of paths going from all leaf nodes to the root node is created. For each node, the path can go through either the *preserve* or the *remove* state of the trellis. The Viterbi algorithm is used to find the paths that minimize the global cost at the root node. Note that the trellis structure itself guarantees that this optimum decision is increasing. The optimization is achieved in a bottom-up iterative fashion. For each node, it is possible to define the optimum paths ending at the *preserve* state and at the *remove* state.

- Let us consider a node N_k and its *preserve* state N_k^P. A path $Path_k$ is a continuous set of transitions between nodes $(N_\alpha \rightarrow N_\beta)$ defined in the trellis:

  $$Path_k = (N_\alpha \rightarrow N_\beta) \cup (N_\beta \rightarrow N_\gamma) \cup \ldots \cup (N_\psi \rightarrow N_\omega).$$

  The path $Path_k^P$ starting from a leaf node and ending at that state is composed of two sub-paths: the first one, $Path_k^{P,Left}$, comes from the left child and the second one, $Path_k^{P,Right}$, from the right child (see Fig. 18). In both cases, the path can emerge either from the *preserve* or from the *remove* state of the child nodes. If N_{k_1} and N_{k_2} are respectively the left and the right child nodes of N_k, we have:

$$\begin{aligned}
Path_k^{P,Left} &= Path_{k_1}^R \bigcup (N_{k_1}^R \rightarrow N_k^P) \\
&\text{or } Path_{k_1}^P \bigcup (N_{k_1}^P \rightarrow N_k^P) \\
Path_k^{P,Right} &= Path_{k_2}^R \bigcup (N_{k_2}^R \rightarrow N_k^P) \\
&\text{or } Path_{k_2}^P \bigcup (N_{k_2}^P \rightarrow N_k^P) \\
Path_k^P &= Path_k^{P,Left} \bigcup Path_k^{P,Right}
\end{aligned} \quad (10)$$

  The cost of a path is equal to the sum of the costs of its individual state transitions. Therefore, the optimum path (path of lower cost) for each child can be easily selected.

$$\begin{aligned}
&\text{If } Cost(Path_{k_1}^R) < Cost(Path_{k_1}^P) \\
&\text{then } \{Path_k^{P,Left} = Path_{k_1}^R \bigcup (N_{k_1}^R \rightarrow N_k^P); \\
&\quad Cost(Path_k^{P,Left}) = Cost(Path_{k_1}^R); \} \\
&\text{else } \{Path_k^{P,Left} = Path_{k_1}^P \bigcup (N_{k_1}^P \rightarrow N_k^P); \\
&\quad Cost(Path_k^{P,Left}) = Cost(Path_{k_1}^P); \} \\
&5mm]\text{If } Cost(Path_{k_2}^R) < Cost(Path_{k_2}^P) \\
&\text{then } \{Path_k^{P,Right} = Path_{k_2}^R \bigcup (N_{k_2}^R \rightarrow N_k^P); \\
&\quad Cost(Path_k^{P,Right}) = Cost(Path_{k_2}^R); \} \\
&\text{else } \{Path_k^{P,Right} = Path_{k_2}^P \bigcup (N_{k_2}^P \rightarrow N_k^P); \\
&\quad Cost(Path_k^{P,Right}) = Cost(Path_{k_2}^P); \} \\
&\\
&Cost(Path_k^P) = Cost(Path_k^{P,Left}) \\
&\qquad + Cost(Path_k^{P,Right}) \\
&\qquad + Cost(N_k^P);
\end{aligned} \quad (11)$$

- In the case of the *remove* state, N_k^R, the two sub-paths can only come from the *remove* states of the children. So, no selection has to be done. The path and its cost are con-

[2] Although some modifications may be much more severe than others, the cost choice has no strong effect on the final result. This issue of cost selection is similar to the hard versus soft decision of the Viterbi algorithm in the context of digital communications [22].

structed as follows:

$$\begin{aligned} Path_k^{R,Left} &= Path_{k_1}^R \bigcup (N_{k_1}^R \rightarrow N_k^R); \\ Path_k^{R,Right} &= Path_{k_2}^R \bigcup (N_{k_2}^R \rightarrow N_k^R); \\ Path_k^R &= Path_k^{R,Left} \bigcup Path_k^{R,Right}; \\ Cost(Path_k^R) &= Cost(Path_{k_1}^R) + Cost(Path_{k_2}^R) \\ &+ Cost(N_k^R); \end{aligned} \tag{12}$$

This procedure is iterated in a bottom-up fashion until the root node is reached. One path of minimum cost ends at the *preserve* state of the root node and another path ends at the *remove* state of the root node. Among these two paths, the one of minimum cost is selected. This path connects the root node to all leaves and the states it goes through define the final decisions. By construction, these decisions are increasing and they are as close as possible to the original decisions.

A complete example of optimization is shown in Fig. 19. The original Binary Partition Tree involves 5 nodes. As before, the *preserve* decisions are shown by a square whereas the *remove* decisions are indicated by a circle. As can be seen, the original tree does not correspond to a set of increasing decisions because N_3 should be removed but N_4 and N_5 should be preserved. The algorithm is initialized by creating the trellis and by populating the states by their respective cost (see Fig. 17). Then, the first step of the algorithm consists in selecting the paths that go from states N_4^R, N_4^P, N_5^R, N_5^P to states N_3^R, N_3^P. The corresponding trellis is shown in the upper part of Fig. 19 together with the corresponding costs of the four surviving paths. The second step iterates the procedure between states N_2^R, N_2^P, N_3^R, N_3^P and states N_1^R, N_1^P. Here again, only four paths survive. They are indicated in the central diagram of Fig. 19. Finally, the last step consists in selecting the path of lowest cost that terminates at the root states. In the example of Fig. 19, the path ending at the *remove* state of the root node (N_1^R) has a cost of 3, whereas the path ending at the *preserve* state (N_1^P) has a cost of 1. This last path is taken since it corresponds to an increasing set of decisions and involves just one modification of the original decisions. In order to find the optimum increasing decisions, one has to track back the selected path from the root to all leaves. In our example, we see that the paths hit the following states: N_1^P, N_2^R, N_3^P, N_4^P and N_5^P. The diagram at the bottom of Fig. 19 shows the final path together with the modified Binary Partition Tree. As can be seen, the only modification has been to change the decision of node N_3 and the resulting set of decisions is increasing.

A complete example of decisions modification is shown in Fig. 20. The original Binary Partition Tree corresponds to the one shown in Fig. 16. The Viterbi algorithm has to modify 5 decisions along path A and one decision along path B (see Fig. 16) to get the optimum set of increasing decisions.

To summarize this section, let us say that the pruning strategy can be applied directly on the tree if the decision criterion is increasing (size is a typical example). In the case of a non-increasing criterion such as the perimeter, the Viterbi algorithm can be used to modify the smallest number of decisions so that increasingness is obtained. These modifications define a pruning strategy.

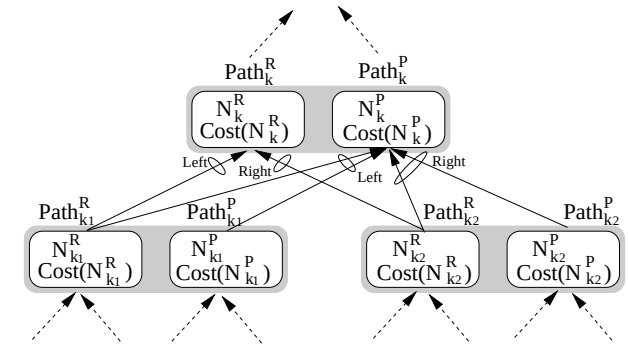


Fig. 18. Definition of $Path$ and cost for the Viterbi algorithm (see equations 10, 11 and 12).

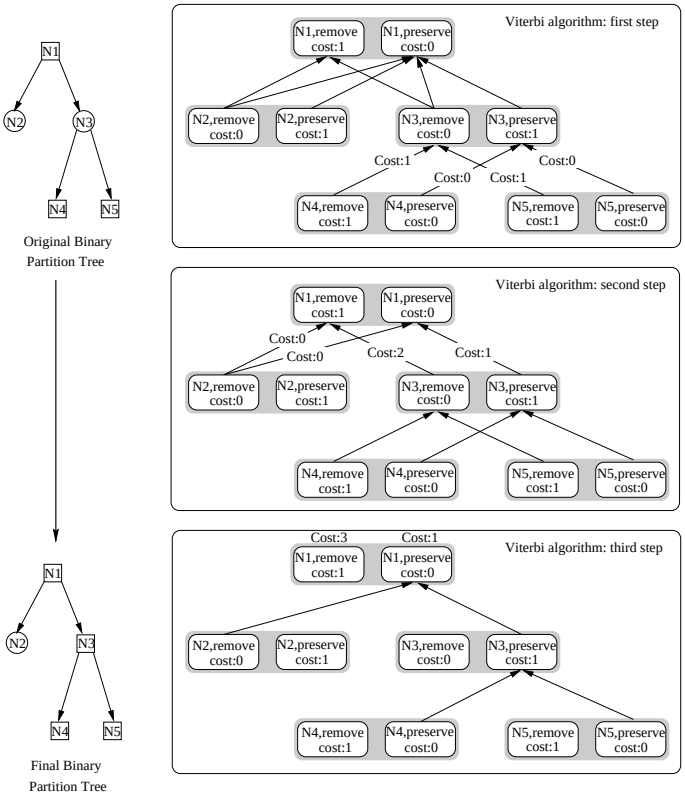


Fig. 19. Definition of the optimum decisions by the Viterbi algorithm.

B. Filtering example

Once the tree has been pruned, the output partition is computed and each region is modeled by a constant value. In the case of anti-extensive connected operators, the minimum gray level value of the pixels in the original image is used. Here, since we are interested in self-dual operators, a self-dual model has to be used. Examples of self-dual models are the mean or the median of the original pixel values. In the following we assume that the median is used.

A first example of size-oriented simplification is shown in Fig. 21.a. The size threshold has been set to 50 pixels. This result may be surprising because a large number of regions smaller than 50 pixels are still visible in the filtered image (the

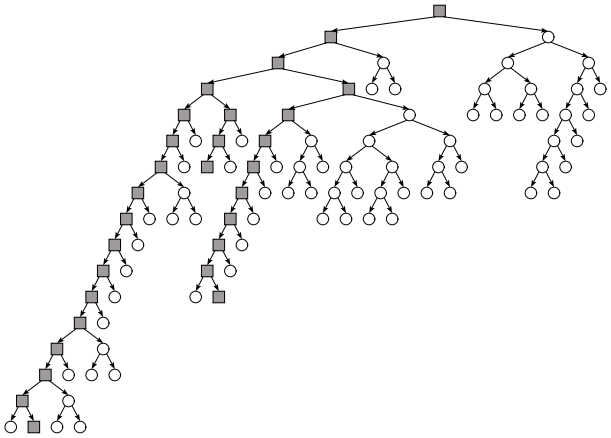


Fig. 20. Set of increasing decisions resulting from from the use of the Viterbi algorithm on the original tree of Fig. 16. Five decisions along path A and one decision along path B have been modified. Gray squares: nodes to be preserved, white circles: nodes to be removed.

texture of the fish for example). To understand this result, let us analyze the example of Binary Partition Tree shown in the left side of Fig. 22 (Note that this tree is presented here as a simple illustration. It is not the tree used to generate the example of Fig. 21). In this tree, one can see a large number of configurations where one node has to be removed whereas its sibling has to be preserved. Note that since the criterion is increasing, the parent of these two nodes has to be preserved. In terms of regions, this configuration means that one of the siblings as well as the parent correspond to large regions whereas the other sibling is of small size. Fig. 23 illustrates this issue on a very simple example: Regions R_1 and R_2 should be preserved because they are of large size, whereas region $R_3 = R_1 \setminus R_2$ is of small size. It should be removed but in the final partition, the space corresponding to $R_1 \setminus R_2$ will appear as a connected component of small area.

In section II-B, we have illustrated the example of an area opening and we have seen that the filtered image involves a large number of small regions (potentially all regions that are not maxima). In fact, once a regional maxima has reached (by merging) the size threshold, it is not merged anymore with its neighboring regions even if these neighboring regions are small. In the example of Fig. 22.left, we have the same issue: once a node corresponds to a region larger than 50 pixels, it is not merged with its sibling even if the sibling is small.

For certain applications, it may be necessary to force the operator to produce an output image where all flat zones are guaranteed to fulfill the simplification criterion. This modification can easily be implemented using the propagation process explained in section V-B. The idea is explained in Fig. 22. The first step consists in defining the markers. These markers are all *Preserve* leaves as well as *Preserve* nodes that have two *Remove* children. In the example of Fig. 22, there are five markers. The second step defines the filtered partition by propagating these markers as in the case of the segmentation described in section V-B. Fig. 22.right shows the result of this propagation on the Binary Partition Tree. A size-oriented simplification of the *Bream* image using this strategy is presented in Fig. 21.b. All regions of size smaller than 50 pixels have been removed.

Fig. 21. Example of size-oriented simplification (Size threshold 50 pixels). a) simple size simplification. b) size simplification with propagation strategy

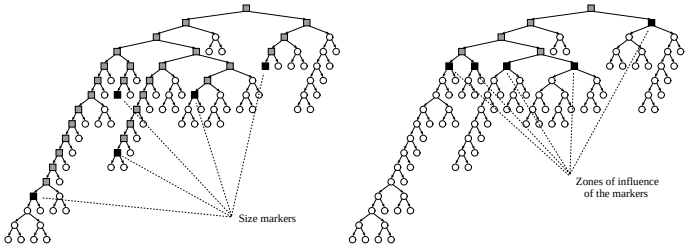


Fig. 22. Size-oriented simplification. Left) Binary Partition Tree with size criterion. The black squares indicate the size markers. Right) Definition of the zones of influence of the size markers.

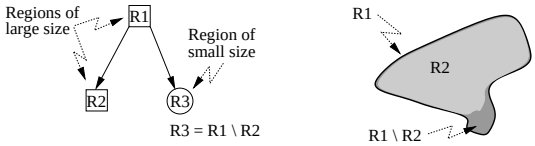


Fig. 23. Illustration of decisions where a node has to be preserved whereas it s sibling has to be removed.

Finally, Fig. 24 illustrates two simplification criteria. Fig. 24.b corresponds to a size criterion whereas Fig. 24.c corresponds to a perimeter criterion. In both cases, the regions that do not fulfill the criterion have been removed by propagation of the markers as explained above. Moreover, the Viterbi algorithm has been used in the case of the perimeter since this criterion is not increasing. The difference between the two simplification criteria can be seen in the simplification of the text appearing in the upper left corner. These filtering tools are self-dual connected operators and generalize the results reported in [16]. They possess the attractive feature of simplifying the image while preserving the contour information.

VII. Coding Binary Partition Trees

As shown in the previous sections, a Binary Partition Tree is an interesting representation to implement a large set of processing tools and functionalities. Moreover, the implementation of these processing tools can be done very efficiently since the number of image components to process is reduced to the number of tree nodes which in practice ranges between a few tenth to a few thousands. Note that the time consuming part of the process is not the tree processing but the definition of the merging sequence. Depending on the application, this comment leads

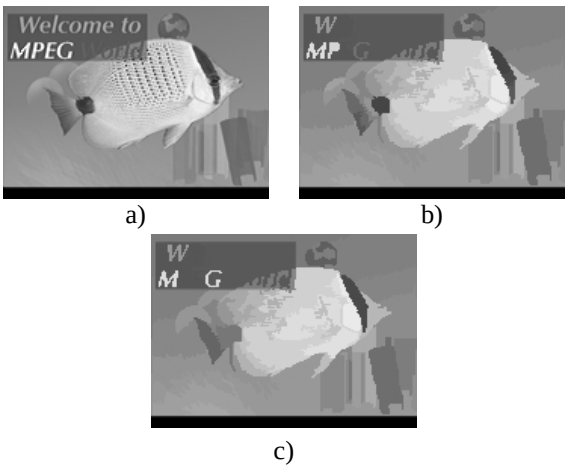


Fig. 24. Example of self-dual connected operators. a) original image. b) size-oriented simplification. c) perimeter-oriented simplification

Image	Number of regions of the initial partition		
	50	100	200
Bream	8008 bits (78,14,8%)	12376 bits (73,18,9%)	19016 bits (66,24,10%)
Akiyo	7880 bits (77,15,8%)	12512 bits (73,18,9%)	19032 bits (65,25,10%)

TABLE II
NUMBER OF BITS TO CODE THE BINARY PARTITION TREE REPRESENTATION. PERCENTAGES ($n_1\%, n_2\%, n_3\%$) INDICATE THE BITSTREAM COMPOSITION: n_1 INITIAL PARTITION, n_2 MERGING SEQUENCE AND n_3 COLOR VALUES. NUMBER OF BITS OF THE ORIGINAL (UNCOMPRESSED) IMAGES IN QCIF FORMAT: 608256 BITS

to the idea of computing the tree once and of storing the representation. In this context, an important question is to know how many bits are necessary to code the full representation, that is the initial partition, the merging sequence and, for example, a color value for each region of the initial partition.

In the following, we have assumed that the partition is represented in QCIF format (for information retrieval applications, the original image may be in a different format) and that the contours of the initial partition have been coded by a simple chain code. Table II defines the number of bits for various levels of granularity of the initial partition. As can be seen, the overall number of bits remains moderate and most of the coding cost is devoted to the initial partition. Note that the coding of the initial partition has been performed with a very simple and lossless technique (chain code). If higher compression is required, more sophisticated lossy techniques could be used.

VIII. CONCLUSIONS

In this paper, we have discussed the interest of Binary Partition Tree representations for several image processing tasks. This representation combines a large number of regions that can be extracted from an image. Although the tree construction was not the main focus of this paper, the use of segmentation algorithms relying on merging techniques has been discussed. Note that this is not the only possibility and top-down or supervised approaches should be investigated. The regions contained in the tree are organized in a hierarchical structure. This organization allows the implementation of fast and sophisticated techniques (for example the marker propagation of section V-B or the Viterbi algorithm described in section VI-A).

The processing of the Binary Partition Tree generally consists in defining which nodes (and corresponding regions) are of interest for a particular image processing task. In this framework, we have discussed examples of minimization of local criteria (circular object detection, section IV-A) as well as the minimization of global criteria (rate/distortion optimization for browsing functionality, section IV-B). The Binary Partition Tree gives access to some (not all) neighborhood as well as similarity relationships between regions. This feature allows the implementation of propagation techniques that are particularly useful for segmentation applications (section V-B). Finally, pruning strategies lead to the definition of new connected operators. In this case, we have seen that the increasingness of the merging criterion is an important issue. In the case of a non-increasing merging criterion, a Viterbi algorithm can be used to define the pruning strategy (section VI-A). Note that the specific criteria (circularity, size, perimeter, etc) used in this paper are just simple examples that were selected to explain the main issues involved in the representation. For a particular application, more useful and possibly more complex criteria may be used. Let us mention for instance complex shape characteristics, texture features, motion information in the case of sequences, etc. We will investigate such criteria in the future.

REFERENCES

[1] E. Breen and R. Jones. An attribute-based approach to mathematical morphology. In P. Maragos, R.W. Schafer, and M.A. Butt, editors, *International Symposium on Mathematical Morphology*, pages 41–48, Atlanta (GA), USA, May 1996. Kluwer Academic Publishers.

[2] C.R. Brice and C.L. Fenema. Scene analysis using regions. *Artificial intelligence*, 1:205–226, 1970.

[3] J. Crespo. *Morphological Connected Filters and Intra-region Smoothing for Image Segmentation*. PhD thesis, Georgia Institute of Technology, 1993.

[4] J. Crespo. Space connectivity and translation-invariance. In P. Maragos, R.W. Schafer, and M.A. Butt, editors, *International Symposium on Mathematical Morphology*, pages 118–126, Atlanta (GA), USA, May 1996. Kluwer Academic Publishers.

[5] J. Crespo, J. Serra, and R.W. Schafer. Theoretical aspects of morphological filters by reconstruction. *Signal Processing*, 47(2):201–225, 1995.

[6] L. Garrido and P. Salembier. Region based analysis of video sequences with a general merging algorithm. In *IX European Signal Processing Conference, EUSIPCO'98*, volume III, pages 1693–1696, Rhodes, Greece, September, 8-11 1998.

[7] L. Garrido, P. Salembier, and D. Garcia. Extensive operators in partition lattices for image sequence analysis. *EURASIP Signal Processing*, 66(2):157–180, April 1998.

[8] H. Heijmans. Connected morphological operators and filters for binary images. In *IEEE Int. Conference on Image Processing, ICIP'97*, volume 2, pages 211–214, Santa Barbara (CA), USA, October 1997.

[9] F. Meyer and S. Beucher. Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, September 1990.

[10] O. Morris, M. Lee, and A. Constantinidies. Graph theory for image analysis: an approach based on the shortest spanning tree. *IEE Proceedings, F*, 133(2):146–152, April 1986.

[11] A. Ortega, K. Ramchandran, and M. Vetterli. Optimal buffer-constrained source quantization and fast approximations. In *Proc. IEEE Int. Symp. Circuits and Systems*, volume 1, May 1992.

[12] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-

distorsion sense. *IEEE Transactions on Image Processing*, 2(2):160–175, April 1993.

[13] E. Reusens. Joint optimization of representation model and frame segmentation for generic video compression. *EURASIP Signal Processing*, 46(11):105–117, September 1995.

[14] P. Salembier. Morphological multiscale segmentation for image coding. *EURASIP Signal Processing*, 38(3):359–386, September 1994.

[15] P. Salembier, F. Marqués, M. Pardàs, R. Morros, I. Corset, S. Jeannin, L. Bouchard, F. Meyer, and B. Marcotegui. Segmentation-based video coding system allowing the manipulation of objects. *IEEE Trans. on Circuits and Systems for Video Technology*, 7(1):60–74, February 1997.

[16] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4):555–570, April 1998.

[17] P. Salembier and J. Serra. Flat zones filtering, connected operators and filters by reconstruction. *IEEE Transactions on Image Processing*, 3(8):1153–1160, August 1995.

[18] P. Salembier, L. Torres, F. Meyer, and C. Gu. Region-based video coding using mathematical morphology. *Proceedings of IEEE (Invited paper)*, 83(6):843–857, June 1995.

[19] J. Serra and P. Salembier. Connected operators and pyramids. In SPIE, editor, *Image Algebra and Mathematical Morphology*, volume 2030, pages 65–76, San Diego (CA), USA, July 1993.

[20] L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In J. Serra and P. Salembier, editors, *First Workshop on Mathematical Morphology and its Applications to Signal Processing*, pages 22–27, Barcelona, Spain, May 1993. UPC.

[21] L. Vincent. Morphological gray scale reconstruction in image analysis: Applications and efficients algorithms. *IEEE, Transactions on Image Processing*, 2(2):176–201, April 1993.

[22] A.J. Viterbi and J.K. Omura. *Principles of Digital Communications and Coding*. Mc Graw-Hill, New York, 1979.