

# Connected operators based on region-tree pruning strategies

Philippe Salembier, Luis Garrido

Universitat Politècnica de Catalunya, Barcelona, SPAIN, {philippe,oster}@gps.tsc.upc.es

## Abstract

This paper discusses region-based representations useful to create connected operators. The filtering approach involves three steps: first, a region tree representation of the input image is constructed. Second, the simplification is obtained by pruning the tree and third, and output image is constructed from the pruned tree. The paper focuses in particular on the pruning strategies that can be used depending of the increasingness of the simplification criteria.

## 1. Introduction

Filtering techniques commonly used in image processing are defined by an input/output relationship that relies on a specific signal called impulse response, window or structuring element. The impulse response of a linear filter defines the filter properties but introduces some blurring in the output image. The major drawback of median filtering is that every region tends to be round after filtering with most commonly used windows (circles, squares, etc.). This effect is due to the shape of the window combined with the median processing. Morphological opening and closing also introduce severe distortions due to the shape of the structuring element. Most people would say that the heart of the filter design is to appropriately select the impulse response, the window or the structuring element. However, in the framework of image processing, this selection implies some drawbacks because severe distortions are introduced in the output images.

Many connected operators used in practice choose a completely different approach: the filtering is done without using any specific signal such as an impulse response, a window or a structuring element. In fact, the structure of the input signal itself is used to filter. As a result, no distortion related to a priori selected signals is introduced in the output. Gray level connected operators [6] are operators that act by merging of elementary regions called flat zones. They cannot create new contours and, as a result, they cannot introduce in the output a structure that is not present in the input. Furthermore, they cannot modify the position of existing boundaries between regions and, therefore, have very good contour preservation properties. Several approaches can be used to create connected operators. One of the most popular approach consists in using the classical pixel-based representation of the image and a reconstruction process [7, 2]. An alternative approach

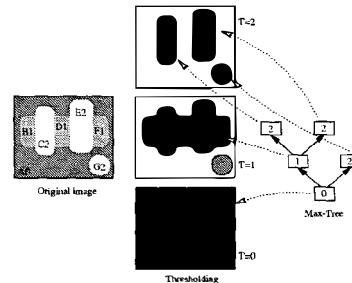


Figure 1. Max-tree representation of images.

relies on the definition of a region-based representation of the image and the definition of a region merging process [5, 4]. The goal of this paper is to discuss this second approach assuming that the region-based representation is a tree. The paper organization is as follows. Section 2 defines two region tree representations: the Max-tree (or Min-tree) and the Binary Partition Tree. The filtering strategies are discussed in section 3. Finally, conclusions are reported in section 4.

## 2. Region Tree Representations

### 2.1. Max-tree and Min-tree

The first tree representation is called a Max-tree [5]. This representation enhances the maxima of the signal. Each node  $\mathcal{N}_k$  in the tree represents a connected component of the space that is extracted by the following thresholding process: for a given threshold  $T$ , consider the set of pixels  $X$  that have a gray level value larger than  $T$  and the set of pixels  $Y$  that have a gray level value equal to  $T$ :

$$\begin{aligned} X &= \{x, \text{ such that } f(x) \geq T\} \\ Y &= \{x, \text{ such that } f(x) = T\} \end{aligned} \quad (1)$$

The tree nodes  $\mathcal{N}_k$  represent the connected components of  $X$  such that  $X \cap Y \neq \emptyset$ . An example of Max-tree is shown in Fig. 1. The original image is made of 7 flat zones:  $\{A, \dots, G\}$ . The number following each letter defines the gray level value of the flat zones. The binary images,  $X$ , resulting from the thresholding with  $0 \leq T \leq 2$  are shown in the center of the figure. Finally, the Max-tree is given in the right side. It is composed of 5 nodes that represent the connected components shown in black. The number inside each square represents the threshold value where the component was extracted.

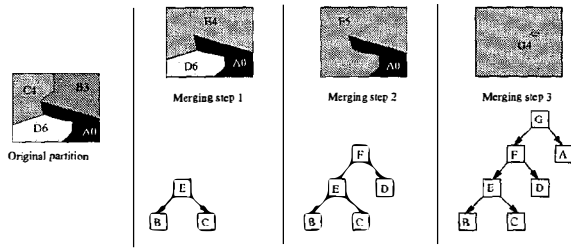


Figure 2. Example of Binary Partition Tree creation with a region merging algorithm.

Finally, the links in the tree represent the inclusion relationships among the connected components following the threshold values. Note that when the threshold is set to  $T = 1$ , the circular component does not create a connected component that is represented in the tree because none of its pixels has a gray level value equal to 1. However, the circle itself is obtained when  $T = 2$ . The maxima are represented by three leaves and the tree root represents the entire image support.

## 2.2. Binary Partition Tree

The second example of region-based representation is the Binary Partition Tree [4]. It represents a set of regions that can be obtained from the partition of flat zones. The leaves of the tree represent the flat zones of the original signal. The remaining nodes represent regions that are obtained by merging the regions represented by the children. As in the cases of the Max-tree and Min-tree, the root node represents the entire image support. This representation should be considered as a compromise between representation accuracy and processing efficiency. Indeed, all possible merging of flat zones are not represented in the tree. Only the most “useful” merging steps are represented. 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.

The Binary Partition Tree should be created in such a way that the most “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 [3, 1]). In the following, this information is called the *merging sequence*. Starting from the partition of flat zones, the algorithm merges neighboring regions following a homogeneity criterion until a single region is obtained. An example is shown in Fig. 2. The original partition involves four regions. The regions are indicated by a letter and the number indicates the grey level value of the flat zone. The algorithm merges the four regions in three steps. In the first step, the pair of most similar regions,  $B$  and  $C$ , are merged to create region  $E$ . Then, region  $E$  is merged with region  $D$  to create region  $F$ . Finally, region  $F$  is

merged with region  $A$  and this creates region  $G$  corresponding to the region of support of the whole image. In this example, the merging sequence is:  $(B, C)|(E, D)|(F, A)$ . This merging sequence progressively defines the Binary Partition Tree as shown in Fig. 2.

To create the Binary Partition Trees used in this paper, the merging algorithm following the color homogeneity criterion described in [1] has been used. It should be noticed however 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 in the second stage. Furthermore, additional information of previous processing or detection algorithms can also be used to generate the tree in a more robust way. For instance, an object mask can be used to impose constraints on the merging algorithm in such a way that the object itself is represented as a single node in the tree. Typical examples of such algorithms are face, skin, character or foreground object detection. If the functions used to create the tree are self-dual, the tree itself is a self-dual representation appropriate to derive self-dual connected operators. By contrast, the Max-tree (Min-tree) is adequate for anti-extensive (extensive) connected operators. Note that in all cases, the trees are hierarchical region-based representations. They encode a large set of regions and partitions that can be derived from the flat zones partition of the original image without adding new contours.

## 3. Filtering Strategy

Once the tree representation has been created, the filtering strategy consists in *pruning* the tree and in reconstructing an image from the pruned tree. The simplification is performed by pruning because the idea is to eliminate the image components that are represented by the leaves and branches of the tree. The nature of these components depends on the tree. In the case of Max-trees (Min-trees), the components that may be eliminated are regional maxima (minima) whereas the elements that may be simplified in the case of Binary Partition Trees are unions of the most similar flat zones. The simplification itself is governed by a criterion which may involve simple notions such as size, contrast or more complex ones such as texture, motion or even semantic criteria.

### 3.1. Increasing criteria

One of the interests of the tree representations is that the set of possible merging steps is fixed (defined by the tree branches). As a result, a large number of simplification (pruning) strategies may be designed. A typical example of such strategy deals with non-increasing simplification criteria. A criterion  $C$  assessed on a region  $R$  is said to be increasing if the following property holds:  $\forall R_1 \subseteq R_2 \Rightarrow C(R_1) \leq$



Figure 3. Area filtering: original (left), area opening (center), area open-close (right).

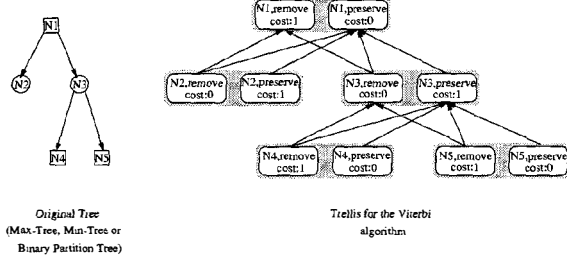


Figure 4. Creation of the trellis for the Viterbi algorithm.

$\mathcal{C}(\mathcal{R}_2)$ . Assume that all nodes 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 since the increasingness of the criterion guarantees that if a node has to be removed all its descendants have also to be removed. A typical example is the area opening. One of the possible implementation of the area opening consists in creating a Max-tree and in measuring the area (the number of pixels)  $\mathcal{A}_k$  contained in each node  $\mathcal{N}_k$ . If the area  $\mathcal{A}_k$  is smaller than a threshold,  $\mathcal{T}_A$ , the node is removed. The simplification effect of the area opening is illustrated in Fig. 3. As expected, the operator removes small bright components of the image. If this simplified image is processed by the dual operator, the area closing, small dark components are also removed. Using the same strategy, a large number of connected operators can be obtained.

### 3.2. Non-increasing criteria

If the criterion is not increasing, the pruning strategy is not trivial since the descendants of a node to be removed have not necessarily to be removed. In practice, the criterion non-increasingness implies a lack of robustness of the operator [5]. For example, similar images may produce quite different results or small modifications of the criterion threshold involve drastic changes. A possible 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 can be viewed as dynamic programming problem that can be efficiently solved with the Viterbi algorithm.

The dynamic programming algorithm will be explained and illustrated in the sequel on a binary tree (see Fig. 4). The extension to N-ary trees is straightforward. The trellis on which the Viterbi algorithm is applied has the same structure as the region tree except that two trellis states, *preserve*  $\mathcal{N}_k^P$  and *remove*  $\mathcal{N}_k^R$ , correspond to each node  $\mathcal{N}_k$  of the tree. The two states of each child node are connected to the two states of its parent. However, 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 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 node 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. The cost values appearing in Fig. 4 assume that nodes  $\mathcal{N}_1$ ,  $\mathcal{N}_4$  and  $\mathcal{N}_5$  should be preserved and that  $\mathcal{N}_2$  and  $\mathcal{N}_3$  should be removed. The goal of the Viterbi algorithm is to define the set of increasing decisions such that  $\sum_k \text{Cost}(\mathcal{N}_k)$  is minimized.

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. 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  $\mathcal{N}_k$  and its *preserve* state  $\mathcal{N}_k^P$ . A path  $Path_k$  is a continuous set of transitions between nodes ( $\mathcal{N}_\alpha \rightarrow \mathcal{N}_\beta$ ) defined in the trellis:  $Path_k = (\mathcal{N}_\alpha \rightarrow \mathcal{N}_\beta) \cup (\mathcal{N}_\beta \rightarrow \mathcal{N}_\gamma) \cup \dots \cup (\mathcal{N}_\psi \rightarrow \mathcal{N}_k)$ . The path  $Path_k^P$  starting from a leaf node and ending at that state is composed of two sub-paths<sup>1</sup>: 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. 5). In both cases, the path can emerge either from the *preserve* or from the *remove* state of the child nodes. If  $\mathcal{N}_{k_1}$  and  $\mathcal{N}_{k_2}$  are respectively the left and the right child nodes of  $\mathcal{N}_k$ , we have:

$$\begin{aligned} Path_k^{P,Left} &= Path_{k_1}^R \cup (\mathcal{N}_{k_1}^R \rightarrow \mathcal{N}_k^P) \quad \text{or} \\ & Path_{k_1}^P \cup (\mathcal{N}_{k_1}^P \rightarrow \mathcal{N}_k^P) \\ Path_k^{P,Right} &= Path_{k_2}^R \cup (\mathcal{N}_{k_2}^R \rightarrow \mathcal{N}_k^P) \quad \text{or} \\ & Path_{k_2}^P \cup (\mathcal{N}_{k_2}^P \rightarrow \mathcal{N}_k^P) \end{aligned} \quad (2)$$

$$Path_k^P = Path_k^{P,Left} \cup Path_k^{P,Right}$$

<sup>1</sup>In the general case of an N-ary tree, the number of incoming paths may be arbitrary.

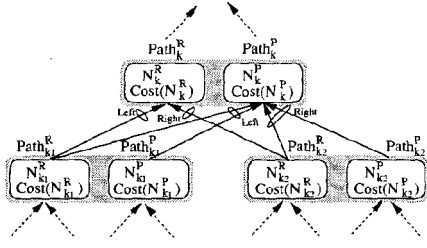


Figure 5. Definition of *Path* and cost for the Viterbi algorithm (see Eqs. 2, 3 and 4).

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

$$\begin{aligned}
 &\text{If } \text{Cost}(\text{Path}_{k_1}^R) < \text{Cost}(\text{Path}_{k_1}^P) \\
 &\text{then } \{\text{Path}_{k_1}^{P,Left} = \text{Path}_{k_1}^R \cup (\mathcal{N}_{k_1}^R \rightarrow \mathcal{N}_k^P); \\
 &\quad \text{Cost}(\text{Path}_{k_1}^{P,Left}) = \text{Cost}(\text{Path}_{k_1}^R); \} \\
 &\text{else } \{\text{Path}_{k_1}^{P,Left} = \text{Path}_{k_1}^P \cup (\mathcal{N}_{k_1}^P \rightarrow \mathcal{N}_k^P); \\
 &\quad \text{Cost}(\text{Path}_{k_1}^{P,Left}) = \text{Cost}(\text{Path}_{k_1}^P); \} \\
 &\text{If } \text{Cost}(\text{Path}_{k_2}^R) < \text{Cost}(\text{Path}_{k_2}^P) \\
 &\text{then } \{\text{Path}_{k_2}^{P,Right} = \text{Path}_{k_2}^R \cup (\mathcal{N}_{k_2}^R \rightarrow \mathcal{N}_k^P); \\
 &\quad \text{Cost}(\text{Path}_{k_2}^{P,Right}) = \text{Cost}(\text{Path}_{k_2}^R); \} \\
 &\text{else } \{\text{Path}_{k_2}^{P,Right} = \text{Path}_{k_2}^P \cup (\mathcal{N}_{k_2}^P \rightarrow \mathcal{N}_k^P); \\
 &\quad \text{Cost}(\text{Path}_{k_2}^{P,Right}) = \text{Cost}(\text{Path}_{k_2}^P); \} \\
 &\text{Cost}(\text{Path}_k^R) = \text{Cost}(\text{Path}_{k_1}^{P,Left}) + \\
 &\text{Cost}(\text{Path}_{k_2}^{P,Right}) + \text{Cost}(\mathcal{N}_k^P);
 \end{aligned} \quad (3)$$

In the case of the *remove* state,  $\mathcal{N}_k^R$ , the 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 given by:

$$\begin{aligned}
 \text{Path}_{k_1}^{R,Left} &= \text{Path}_{k_1}^R \cup (\mathcal{N}_{k_1}^R \rightarrow \mathcal{N}_k^R); \\
 \text{Path}_{k_2}^{R,Right} &= \text{Path}_{k_2}^R \cup (\mathcal{N}_{k_2}^R \rightarrow \mathcal{N}_k^R); \\
 \text{Path}_k^R &= \text{Path}_{k_1}^{R,Left} \cup \text{Path}_{k_2}^{R,Right}; \\
 \text{Cost}(\text{Path}_k^R) &= \text{Cost}(\text{Path}_{k_1}^R) + \text{Cost}(\text{Path}_{k_2}^R) \\
 &\quad + \text{Cost}(\mathcal{N}_k^R);
 \end{aligned} \quad (4)$$

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. 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 as close as possible to the original decisions.

An example of motion filtering is shown in Fig. 6. The objective of the operator is to remove all moving objects. The criterion is the mean displaced frame difference estimated on each node (non-increasing criterion). In this sequence, all objects are still except the ballerina behind the two speakers and



Figure 6. Example of motion connected operator preserving still objects: original frame (left), operator (center), residue (right).

the speaker on the left side who is speaking. The construction of a region tree and the application of the connected operator with the Viterbi algorithm removes all moving components.

## 4. Conclusions

This paper has discussed two region-based representations useful to create connected operators: Max-tree (Min-tree) and Binary Partition Tree. The filtering approach involves three steps: first, a region-based representation of the input image is constructed. Second, the simplification is obtained by pruning the tree and third, an output image is constructed from the pruned tree. The tree creation defines the set of regions that the pruning strategy can use to create the final partition. It represents a compromise between flexibility and efficiency: on the one hand side, not all possible merging of flat zones are represented in the tree, but on the other hand side, once the tree has been defined complex pruning strategies can be defined. In particular, it is possible to deal with non-increasing criteria using dynamic programming approach such as the Viterbi algorithm.

## References

- [1] 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.
- [2] F. Meyer. The levelings. In *Fourth Int. Symposium on Mathematical Morphology, ISMM'98*, pages 199–206, Amsterdam, The Netherlands, June 1998. Kluwer.
- [3] 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.
- [4] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation and information retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576, April, 2000.
- [5] 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.
- [6] 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.
- [7] L. Vincent. Morphological gray scale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, April 1993.