CONNECTED OPERATORS BASED ON REGION-TREES

Philippe Salembier

Universitat Politècnica de Catalunya, Barcelona, SPAIN

ABSTRACT

Connected operators [1] are operators that act by merging of elementary regions called *flat zones*. They cannot create new contours nor modify their position. Therefore, they have very good contour preservation properties. This paper discusses connected operators based on region trees. 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, the output image is constructed from the pruned tree. The paper focuses on the trees used in practice and discusses various pruning strategies used in practice.

Index Terms— Connected operator, Max-tree, Min-Tree, Inclusion Tree, Binary Partition Tree, Pruning, Morphological filters

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. However, for some image processing applications, this distortion introduced by classical filtering strategies represents a severe drawback.

Connected operators [1] address this issue by using the structure of the input signal itself to filter it. As a result, no distortion related to a priori selected signals is introduced in the output. Connected operators act by merging of elementary regions called *flat zones*. They cannot create new contours nor modify the position of existing ones and, therefore, have very good contour preservation properties.

Gray level connected operators rely on the notion of partition of flat zones. A partition is a set of non-overlapping connected components or regions that fills the entire space. We assume that the connectivity is defined on the digital grid by a translation invariant, reflexive and symmetric relation[1]. Typical examples are the 4- and 8-connectivity. Let $\mathcal{P}$ denotes a partition and $\mathcal{P}(n)$ the region that contains pixel n. A partial order relationship among partitions can be created: $\mathcal{P}_1$ "is finer than" $\mathcal{P}_2$ (written as $\mathcal{P}_1 \sqsubseteq \mathcal{P}_2$), if $\forall n, \mathcal{P}_1(n) \subseteq \mathcal{P}_2(n)$. The set of flat zones of an image f is a partition of the space, $\mathcal{P}_f$. Based on these notions, a gray level operator ψ is connected if the partition of flat zones of its input f is always finer than the partition of flat zones of its output, that is:

$$\mathcal{P}_f \sqsubseteq \mathcal{P}_{\psi(f)}, \forall f \tag{1}$$

This work has been partially supported by project TEC2007-66858

[1] Studies on non classical connectivities can be found in [2, 3, 4, 5].

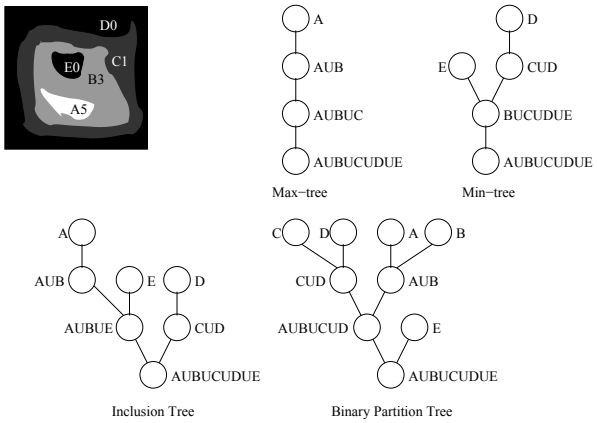


Fig. 1. Tree representations of images.

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 [6, 7]. An overview of this approach can be found in [8]. An alternative approach relies on the definition of a region-based representation of the image and the definition of a region merging process [9, 10, 11]. 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 region tree representations that have been used in practice to create connected operators: Max-tree, Min-tree, Inclusion Tree and Binary Partition Tree. The filtering strategies are discussed and illustrated in section 3. Finally, conclusions are reported in section 4.

2. REGION TREE REPRESENTATIONS

2.1. Max-tree, Min-tree and Inclusion Tree

One of the simplest tree representations is the Max-tree [9]. 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 with gray level value larger than T and the set of pixels Y with gray level value equal to T:

$$\begin{array}{rcl} X & = & \{x \text{ , such that } f(x) \geq T\} \\ Y & = & \{x \text{ , such that } f(x) = T\} \end{array} \tag{2}$$

The tree nodes $\mathcal{N}_k$ represent the connected components of X such that $X \bigcap Y \neq \emptyset$. An example of Max-tree is shown in Fig. 1. The original image is made of 5 flat zones: {A,...,E}. The number following each letter defines the gray level value of the flat zones. The

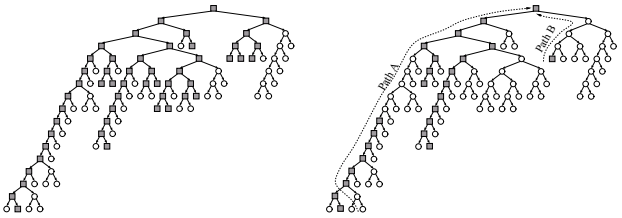
corresponding Max-tree has only one branch because the image has a unique maximum. The dual structure is called a Min-tree. It can be computed directly by duality, that is by computing the Max-tree of $-f$. As the image of Fig. 1 has two minima, the Min-tree has two leaves. Several efficient algorithms have been proposed to compute these trees [9, 12, 5]. The pruning of a Max-tree (Min-tree) acts on the image maxima (minima). If one would like to act simultaneously on maxima and minima, a self-dual representation, where maxima and minima play a symmetrical role, has to be used. An example of such a structure is the Inclusion Tree [11]. It structures the connected components of the set X following the inclusion relationship of the *saturation* of X. The saturation operation fills the holes of the connected components of X. An example of inclusion tree is illustrated in Fig. 1. As the original image has three extrema, the tree has three leaves. The processing of these trees by pruning will be discussed in section 3.

2.2. Binary Partition Tree

The last example of region-based representation is the Binary Partition Tree [10]. It represents a set of regions that can be obtained from the flat zones partition. The leaves represent the flat zones of the original image. The remaining nodes represent regions obtained by merging the regions represented by the children. As in section 2.1, the root node represents the entire image support. All possible merging of flat zones cannot be represented in the tree. In practice, only the most "useful" merging are represented. However, the main advantage of the tree representation is that it allows the fast implementation of sophisticated pruning 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. 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. The image of Fig. 2.1 involves five flat zones. The algorithm merges them in four steps. In the first step, the pair of most similar regions, C and D are merged. Then, A is merged with B; then $C \cup D$ with $A \cup B$. Finally, E is merged with $A \cup B \cup C \cup D$. The resulting merging sequence is: $(C, D)|(A, B)|(C \cup D, A \cup B)|(E, A \cup B \cup C \cup D)$ corresponding to the Binary Partition Tree of Fig. 2.1.

To create Binary Partition Trees, a simple strategy is to merge regions following a color homogeneity criterion as the one described in [9]. Note however that the homogeneity criterion has not to be restricted to color. For example, if the image belongs to a sequence, motion information can 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, if available, an object mask can be used to impose constraints on the merging process in such a way that the object is represented as a single node in the tree. Typical examples of such algorithms are face, skin or foreground object detection. If the functions creating the tree are self-dual, the tree is appropriate to derive self-dual operators as the inclusion tree. By contrast, the Max-tree (Min-tree) is adequate for anti-extensive (extensive) connected operators (the Max-tree removes maxima and the min-tree minima). In all cases, the trees are hierarchical region-based representations of images. They encode a large set of regions that can be derived from the input flat zones partition without adding new contours.



(a) Increasing criterion (b) Non-increasing criterion

Fig. 2. Pruning strategies: Squares: nodes to be preserved, Circle: nodes to be pruned.

(a) original (b) area opening (c) area filtering with inclusion tree

Fig. 3. Area filtering

3. FILTERING STRATEGIES

Once the tree has been created, the filtering strategy consists in *pruning* the tree and in reconstructing an image from the pruned tree. The simplification is done 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-, Min- or Inclusion Trees, the components are respectively regional maxima, minima or extrema. 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 that may involve simple notions such as size, contrast or more complex ones such as texture, motion or even criteria representing semantic notions or objects. In this context, an important issue is the increasingness of the criterion. Let us analyze this point.

3.1. Increasing criteria

A criterion $\mathcal{C}$ assessed on a region R is increasing if the following property holds: $\forall R_1 \subseteq R_2 \Rightarrow \mathcal{C}(R_1) \leq \mathcal{C}(R_2)$. Assume that nodes where the criterion value is lower than a given threshold should be pruned. 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. This situation is illustrated on Fig. 2.a. A typical example is the area opening which is obtained by measuring the area (the number of pixels) of each node of a Max-tree. If the area is smaller than a threshold, the node is removed. The simplification effect of the area opening is illustrated in Fig. 3.b. The operator removes small bright components. If this simplified image is processed by the dual operator, the area closing, small dark components are also removed. The same strategy implemented on an inclusion tree simplifies small extremas (Fig. 3.c). Using this strategy, a large number of connected operators can be obtained by changing the pruning criterion.

3.2. Non-increasing criteria

If the criterion is not increasing, the pruning strategy is not straightforward since the descendants of a node to be removed have not

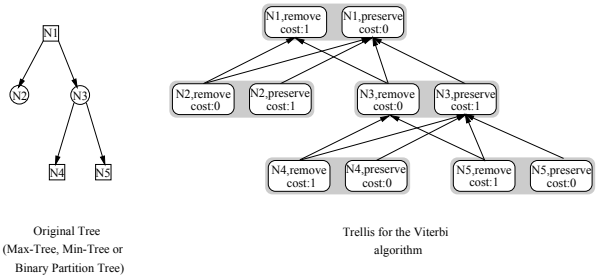


Fig. 4. Creation of the trellis for the Viterbi algorithm.

(a) input (b) output (c) residue

Fig. 5. Example of motion connected operator preserving still objects and its corresponding residue (input - output).

necessarily to be removed. An example of this situation is illustrated in Fig. 2.b, see in particular the decisions along paths A and B. Several simple strategies can be used in this case such as the Min, Max, direct rules [9] or the subtractive rule [13]. However, in some applications, the criterion non-increasingness 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. A possible solution to this problem 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 a 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 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 leaves 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 of pathes that connect the leaves to the root (see [9] for a complete description of the algorithm).

An example of motion filtering is shown in Fig. 5. The objective of the operator is to remove all moving objects. The criterion is the mean displaced frame difference estimated on each node (this is indeed a non-increasing criterion). In this sequence, all objects are still except the ballerina behind the two speakers and 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 as illustrated in Fig. 5.b. In this example, a Max-tree was used first to remove moving maxima and then a Min-tree is used to remove moving minima. This motion connected operator can potentially be used for a large set of applications and can represent an alternative approach to the classical ways of handling the motion information. Indeed, generally, motion information is measured without knowing anything about the image structure. By using motion connected operators, we can "inverse" the classical approach to motion and, for example, analyze simplified sequences where objects are following a known motion. Various connected operators involving nonincreasing criteria such as entropy, simplicity, perimeter can be found in [9, 10].

3.3. Global Optimization Under Constraint

In this section, we illustrate a pruning strategy involving global optimization under constraint. Let us denote by $\mathcal{C}$ a criterion that has to be optimized (we assume, for example, that the criterion has to be minimized) and by $\mathcal{K}$ the constraint. The problem is to minimize the criterion $\mathcal{C}$ with the restriction that the constraint $\mathcal{K}$ is below a given threshold $\mathcal{T}_\mathcal{K}$. Moreover, we assume that both the criterion and the constraint are additive over the regions represented by the nodes $\mathcal{N}_k$: $\mathcal{C} = \sum_{\mathcal{N}_k} \mathcal{C}(\mathcal{N}_k)$ and $\mathcal{K} = \sum_{\mathcal{N}_k} \mathcal{K}(\mathcal{N}_k)$. The problem is therefore to define a pruning strategy such that:

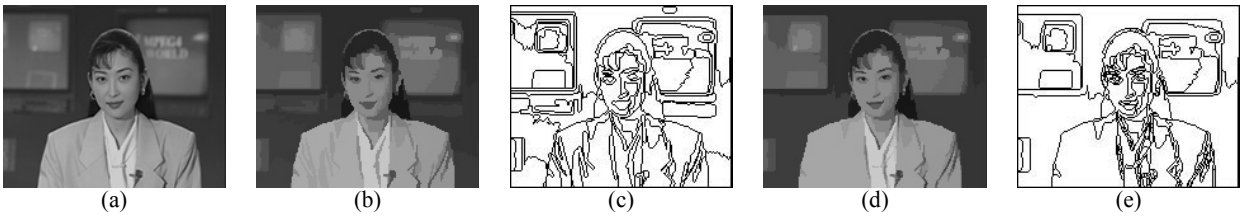
$$\text{Min} \sum_{\mathcal{N}_i} \mathcal{C}(\mathcal{N}_i) \text{ , with } \sum_{\mathcal{N}_i} \mathcal{K}(\mathcal{N}_i) \leq \mathcal{T}_\mathcal{K} \tag{3}$$

This problem can be reformulated as the minimization of the Lagrangian: $\mathcal{L} = \mathcal{C} + \lambda\mathcal{K}$ where λ is the so-called Lagrange parameter. Both problems have the same solution if we find λ^* such that $\mathcal{K}$ is equal (or very close) to the constraint threshold $\mathcal{T}_\mathcal{K}$. Therefore, the problem consists in using the tree to find by pruning a set of nodes creating a partition such that:

$$\text{Min} \left(\sum_{\mathcal{N}_i} \mathcal{C}(\mathcal{N}_i) + \lambda^* \sum_{\mathcal{N}_i} \mathcal{K}(\mathcal{N}_i)\right) \tag{4}$$

Assume, in a first step, that the optimum λ^* is known. In this case, the pruning is done by a bottom-up analysis of the tree. If the Lagrangian value corresponding to a given node $\mathcal{N}_0$ is smaller than the sum of the Lagrangians of the children nodes $\mathcal{N}_i$, then the children are pruned. This procedure is iterated up to the root node.

In practice, the optimum λ^* is not known and the previous bottom-up analysis is embedded in a loop searching for the best λ parameter. This can be done with a gradient search algorithm. The bottom-up analysis itself is not expensive in terms of computation since the algorithm has simply to perform a comparison of Lagrangians for all nodes of the tree. The part of the algorithm



(a) (b) (c) (d) (e)

Fig. 6. Optimization under constraint (squared error = 31 dB). (a) original, (b) Minimization of the flat zone number, (c) flat zone contours of (b) (number of flat zones: 87, perimeter length: 4491), (d) Minimization of the perimeter length, (e) contours of the flat zones of (d) (number of flat zones: 219, perimeter length: 3684).

that might be expensive is the computation of the criterion and the constraint values associated to the regions. Note, however, that this computation has to be done once.

This type of pruning strategy is illustrated by two examples relying on a Binary Partition Tree. In the first example, the goal is to simplify the input image by minimizing the number of flat zones of the output image: $\mathcal{C}_1 = \sum_{\mathcal{N}_k} 1$. In the second example, the criterion is to minimize the total length of the flat zones contours: $\mathcal{C}_2 = \sum_{\mathcal{N}_k} \mathit{Perimeter}(\mathcal{N}_k)$. In both cases, the criterion has no meaning if there is no constraint because the algorithm would prune all nodes. The constraint we use is to force the output image to be a faithful approximation of the input image: the squared error between the input and output images $\mathcal{K} = \sum_{\mathcal{N}_k} \sum_{n \in \mathcal{N}_k} (\psi(f)(n) - f(n))^2$ is constrained to be below a given threshold. In the examples shown in Fig. 6, the squared error constrain is equal to 31 dB. Fig. 6(b) shows the output image when the criterion is the number of flat zones. The image is visually a good approximation of the original image but it involves a much lower number of flat zones: the original image is composed of 14335 flat zones whereas only 87 flat zones are present in the filtered image. The second criterion is illustrated in Fig. 6(d). The approximation provided by this image is of the same quality as the previous one. However, the characteristics of its flat zones are quite different. The total length of the perimeter of its flat zones is equal to 3684 pixels whereas the example of Fig. 6(b) involves a total perimeter length of 4491 pixels. The reduction of perimeter length is obtained at the expense of a drastic increase of the number of flat zones: 219 instead of 87. Fig.s 6(e) and 6(f) show the flat zone contours. The flat zone contours are more complex in the first example but the number of flat zones is higher in the second one.

This kind of strategy can be applied for a large number of criteria and constraints. Note that without defining a tree structure, it would be extremely difficult to implement this kind of connected operators.

4. CONCLUSIONS

This paper has discussed several region-based representations useful to create connected operators: Max-tree, Min-tree, Inclusion 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 or global optimization under constraint.

5. REFERENCES

[1] P. Salembier and J. Serra, "Flat zones filtering, connected operators and filters by reconstruction," *IEEE Transactions on Image Processing*, vol. 3, no. 8, pp. 1153–1160, August 1995.

[2] J. Crespo, "Space connectivity and translation-invariance," in *Int. Symp. on Mathematical Morphology*, P. Maragos et al., Ed., Atlanta (GA), USA, May 1996, pp. 118–126, Kluwer.

[3] C. Ronse, "Set-theoretical algebraic approaches to connectivity in continuous or digital spaces," *Journal of Mathematical Imaging and Vision*, vol. 8, pp. 41–58, 1998.

[4] J. Serra, "Connectivity on complete lattices," *Journal of Mathematical Imaging and Vision*, vol. 9, pp. 231–251, 1998.

[5] G.K. Ouzounis and M.H.F. Wilkinson, "Mask-based second generation connectivity and attribute filters," *IEEE Trans. on PAMI*, vol. 29, no. 2, pp. 990–1004, 2007.

[6] L. Vincent, "Morphological gray scale reconstruction in image analysis: Applications and efficients algorithms," *IEEE Trans. on Image Processing*, vol. 2, no. 2, pp. 176–201, April 1993.

[7] F. Meyer, "The levelings," in *Fourth Int. Symposium on Mathematical Morphology, ISMM'98*, Amsterdam, The Netherlands, June 1998, pp. 199–206, Kluwer.

[8] M.H.F. Wilkinson, "Connected filtering by reconstruction: Basis and new advances," in *IEEE Int. Conference on Image Processing, ICIP'08*, San Diego (CA), USA, October 2008.

[9] P. Salembier et al., "Anti-extensive connected operators for image and sequence processing," *IEEE Trans. on Image Processing*, vol. 7, no. 4, pp. 555–570, April 1998.

[10] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation and information retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 561–576, April, 2000.

[11] P. Monasse and F. Guichard, "Fast computation of a contrast invariant image representation," *IEEE Transactions on Image Processings*, vol. 9, no. 5, pp. 860–872, May 2000.

[12] L. Najman and M. Couprie, "Building the component tree in quasi-linear time," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3531–3539, 2006.

[13] E.R. Urbach et al., "Connected shape-size pattern spectra for rotation and scale-invariante classification of gray scale images," *IEEE Tr. on PAMI*, vol. 29, no. 2, pp. 272–285, 2007.