Antiextensive Connected Operators for Image and Sequence Processing

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

***Abstract*— This paper deals with a class of morphological operators called *connected operators*. These operators filter the signal by merging its *flat zones*. As a result, they do not create any new contours and are very attractive for filtering tasks where the contour information has to be preserved. This paper shows that connected operators work implicitly on a structured representation of the image made of flat zones. The *max-tree* is proposed as a suitable and efficient structure to deal with the processing steps involved in antiextensive connected operators. A formal definition of the various processing steps involved in the operator is proposed and, as a result, several lines of generalization are developed. First, the notion of connectivity and its definition are analyzed. Several modifications of the traditional approach are presented. They lead to connected operators that are able to deal with texture. They also allow the definition of connected operators with less leakage than the classical ones. Second, a set of simplification criteria are proposed and discussed. They lead to simplicity-, entropy-, and motion-oriented operators. The problem of using a nonincreasing criterion is analyzed. Its solution is formulated as an optimization problem that can be very efficiently solved by a Viterbi algorithm. Finally, several implementation issues are discussed showing that these operators can be very efficiently implemented.**

***Index Terms*— Connected operators, connectivity, mathematical morphology, motion criterion, optimization, sequence processing, simplicity criterion, Viterbi algorithm, watershed.**

I. Introduction

THE FIRST *connected operators* reported in the literature are known as *binary opening by reconstruction* [1]. These operators independently act on each connected component of the binary image: they eliminate the connected components that would be totally removed by an erosion with a given structuring element and they leave the other components unchanged. This filtering approach offers the advantage of simplifying the image, because some components are removed, as well as preserving the contour information, because the components that are not removed are perfectly preserved.

This approach has been generalized for gray-level functions using the so-called *reconstruction process* [2]. Beside opening by reconstruction, λ-max operators, area opening [3], dynamics filters [4], and more recently, volumic [5], complexity [6], motion [7], and moment-oriented [8] operators have been proposed. These operators offer various simplification criteria (size, contrast, shape, etc.) while preserving contours. This property makes them very attractive for a large number of applications such as noise cancellation, segmentation, pattern recognition, etc.

The extensive use of connected operators has motivated some theoretical studies. For instance, the notions of connected operators and of flat zones are discussed in a formal way in [9]–[11]. Connectivity issues related to connected operators are analyzed in [12]–[14]. Finally, relations with structured representations of images such as region adjacency graphs and trees are discussed in [7] and [15].

The purpose of this paper is to focus on the class of antiextensive connected operators (and by duality, extensive connected operators). Based on a formal operator definition involving a tree representation of the image called a *max-tree*, several contributions are proposed. First, the notion of connectivity is analyzed. Several modifications of the traditional approach are presented. They lead to connected operators that are able to deal with texture or to connected operators that have much less leakage than classical operators. Second, a set of new simplification criteria are proposed and discussed. In particular, simplicity-, entropy-, and motion-oriented operators are defined. The problem of using a nonincreasing criterion is analyzed and its solution is formulated as an optimization problem that can be very efficiently solved by a Viterbi algorithm. Finally, several applications as well as implementation issues are discussed. Note that part of the work reported here can be found in conference proceedings [6], [7], [13]. One of the objectives of the paper is to review these contributions. However, some new contributions are also presented here. These original contributions mainly concern the max-tree creation and processing, the use of the Viterbi algorithm to deal with nonincreasing criteria, and the entropy connected operator.

The organization of this paper is as follows. Section II is devoted to the notion of binary and gray-level connected operators. This presentation will highlight three major processing steps: tree creation, tree filtering, and image restitution. These three steps are, respectively, discussed in Sections III–V. Finally, Section VI is devoted to the conclusions.

II. Connected Operators

A. Theoretical Definition

In [9], [10], the concept of binary connected operators is formally defined as follows:

Manuscript received August 23, 1996; revised June 4, 1997. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Henri Maitre.

The authors are with E.T.S.E.T.B-Universitat Politècnica de Catalunya, 08034 Barcelona, Spain (e-mail: philippe@gps.tsc.upc.es).

Publisher Item Identifier S 1057-7149(98)02462-2.

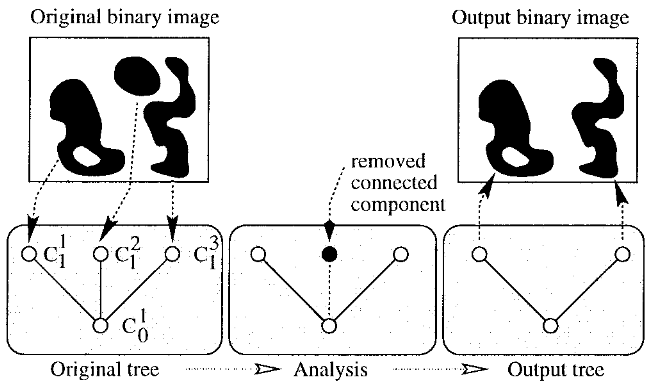


Fig. 1. Binary connected operator.

Definition 1—Binary Connected Operators: A binary operator ψ is connected when for any binary image X, the set difference $X \setminus \psi(X)$ is exclusively composed of connected components of X or of its complement X^c.

The extension of connected operators for gray-level functions relies on the concept of partition [9], [10]. Let us recall that a partition of the space E is a set of connected components $\{A_i\}$ which are disjoint and the union of which is the entire space. Each A_i is called a partition class. Moreover, a partition $\{A_i\}$ is said to be *finer* than another partition $\{B_i\}$ if any pair of points belonging to the same class A_i also belongs to a unique partition class B_j. Consider now a binary image and define its *associated partition* as the partition made of the connected components of the binary sets and of their complement. The definition of connected operators can be expressed using associated partitions as follows.

Theorem 2—Binary Connected Operators via Partition: A binary operator ψ is connected if and only if, for any binary image X, the associated partition of X is finer than the associated partition of $\psi(X)$.

The concept of gray-level connected operators can be introduced if we define a partition associated to a function. To this end, the use of *flat zones* was proposed in [9] and [10]. The set of flat zones of a gray-level function f is the set of the largest connected components of the space where f is constant (a flat zone can be reduced to a single point). The set of flat zones of a function is a partition, called the *partition of flat zones* and leads to the following definition.

Definition 3—Gray-Level Connected Operators: An operator Ψ acting on gray-level functions is connected if, for any function f, the partition of flat zones of f is finer than the partition of flat zones of $\Psi(f)$.

Let us see how this definition implicitly means that the operator works on a structured representation of the image.

B. Binary Antiextensive Connected Operators

In the sequel, we restrict ourselves to the case of antiextensive operators ($\forall X, \psi(X) \subseteq X$). Therefore, a binary antiextensive connected operator is an operator that can only remove connected components of X. The filtering process can easily be explained if a tree representation of the image is used as shown in Fig. 1.

The original image X is composed of three connected components. It can be represented by a tree structure with four nodes: the root node C_0^1 represents the set of pixels belonging to the background X^c, and $\{C_1^k\}_{1 \leq k \leq 3}$ represent the three connected components of the image. In this representation, the filtering process consists in analyzing each node C_1^k by assessing the value of a particular criterion. Assume for example that the criterion consists in counting the number of pixels belonging to a node (area opening [3]). Then, for each node, the criterion value is compared to a given threshold λ and the node is removed if the criterion is lower than λ. In the example of Fig. 1, node C_1^2 is removed because its area is small. As a result, its pixels are moved to the background node C_0^1 (the connected component is removed). As can be seen, the tree links represent the pixels' migration (toward the father) when a node is removed. All antiextensive binary connected operators can be described by this process, the only modification being the criterion that is assessed.

C. Gray-Level Antiextensive Connected Operators

As seen in Definition 3, the extension of connected operators to gray-level images uses the partition of flat zones. This extension can also be seen as a simple generalization of the tree representation to the gray-level case.

The idea consists in creating the tree recursively by a study of the thresholded versions of the image at all possible gray levels. An example is presented in Fig. 2. The original image is composed of seven flat zones identified by a letter $\{A, B, C, D, E, F, G\}$. The number following each letter defines the gray-level value of the flat zone. In our example, the gray-level values range from zero to two. In the first step, the threshold h is fixed to the gray-level value zero. The image is binarized: all pixels at level $h = 0$ (pixels of region A) are assigned to the root node of the tree $C_0^1 = \{A\}$. Furthermore, the pixels of gray-level value strictly higher than $h = 0$ form two connected components that are temporarily assigned to two nodes: $TC_1^1 = \{G\}$ and $TC_1^2 = \{B, C, D, E, F\}$. This creates the first tree (for gray levels [0, 1]). This procedure is the same as the one used for the binary image. In a second step, the threshold is increased by one: $h = 1$. Each node $TC_{h=1}^k$ is processed as the original image. Consider, for instance, the node $TC_1^2 = \{B, C, D, E, F\}$; all its pixels at level $h = 1$ remain unchanged and create the final node C_1^2. However, pixels of gray-level values strictly higher than h (here $\{E, C\}$) create two different connected components and are moved to two temporary child nodes $TC_2^2 = \{C\}$ and $TC_2^3 = \{E\}$. The complete tree construction is done by iterating this process for all nodes k at level h and for all possible thresholds h (from zero to the highest gray-level value). The algorithm can be summarized saying that, at each temporary node TC_h^k, a "local" background is defined by keeping all pixels of gray-level value equal to h (the "local" background itself may not be connected) and that the various connected components formed by the pixels of gray-level value higher than h create the child nodes of the tree.

In this procedure, some nodes may become empty. Therefore, at the end of the tree construction, the empty nodes

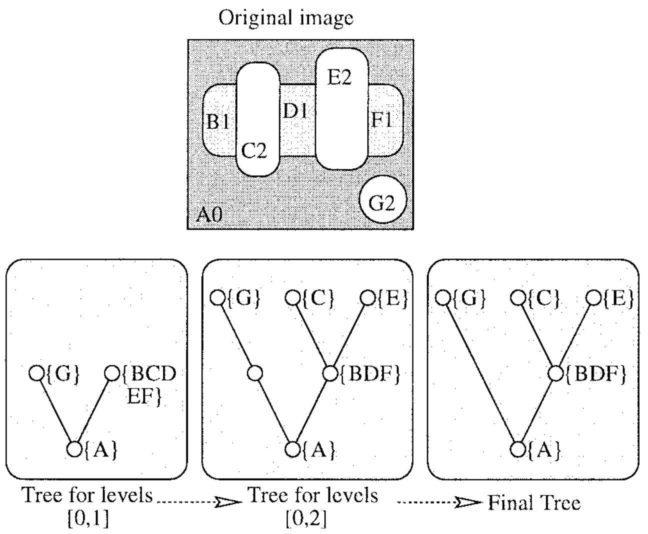


Fig. 2. Max-tree representations.

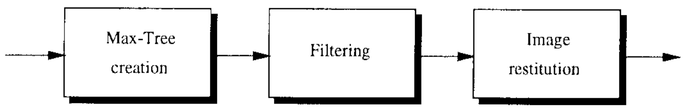


Fig. 3. Connected operators with max-tree representations.

are removed. The final tree is called a *max-tree* in the sense that it is a structured representation of the image which is oriented toward the maxima of the image (maxima are simply the leaves of the tree) and toward the implementation of antiextensive operators. Note that this description does not necessarily correspond to the actual implementation of the tree construction. For this purpose, an efficient algorithm is proposed in the Appendix.

The filtering itself is similar to the one used for the binary case. A criterion $\mathcal{M}(.)$ is assessed for each node C_h^k. Based on the $\mathcal{M}(C_h^k)$ value, the node is either preserved or removed. In this last case, the node's pixels are moved toward its father's node. At the end of the process, the output max-tree is transformed into a gray-level image by assigning the gray value h to the pixels of $C_h^k, \forall k, h$.

D. Connected Operators and Max-Tree Representation

Based on the previous description, a general filtering scheme is illustrated in Fig. 3. It involves a first step of max-tree creation, the goal of which is to structure the pixels in a suitable way for the filtering process. The max-tree representation has also the advantage of leading to very efficient implementations of connected operators. The second step is the filtering itself, which analyzes each node and takes a decision on which node has to be preserved and which node has to be removed. Finally, the last step restores the filtered image by transforming the output max-tree into a gray-level image. The discussion of this paper will focus on antiextensive operators and max-trees. By the duality $f \longleftrightarrow -f$, the same notions can be applied to extensive operators and *min-trees*. In the following, we describe the three steps of Fig. 3 with more details.

III. Max-Tree Creation

The objective of this step is to create the max-tree, that is the set of nodes C_h^k and the links between the father and child nodes. As described in Section II-C, for each temporary node TC_h^k, the set of pixels belonging to the local background is defined and assigned to the max-tree node C_h^k. This is the *binarization* step. Then, the set of pixels belonging to the complement of the local background, that is $TC_h^k \setminus C_h^k$, are analyzed and its connected components create the temporary child nodes TC_{h+1}^k (which will be further analyzed). This is the *connected components definition* step. In the sequel, the *binarization* and the *connected components definition* steps are analyzed and some generalizations are proposed.

A. Binarization

The most natural way of defining the "local" background for each node at level h consists in taking all pixels of gray-level value h. Formally, the node C_h^k is composed of the pixels of level h of the temporary node TC_h^k, that is

$$C_h^k = \{(i, j) \in TC_h^k \text{ such that } f(i, j) = h\} \qquad (1)$$

where $f(i, j)$ represents the gray-level value of the pixel (i, j).

This binarization process extracts the flat zones of the image. This step is closely related to the definition of the basic entities on which the filter is going to act. Equation (1) means that the basic entities are characterized by a strictly flat gray-level value. However, in practice, "visual" entities may not be strictly flat because of noise or texture. To deal with such cases, less strict binarization techniques can be used. To this end, a useful criterion relies on the definition of a bound Δ on the gray-level fluctuations. The corresponding "soft" binarization rule is the following:

$$\begin{aligned} C_h^k = \{&(i, j) \in TC_h^k \text{ such that} \\ &\{\text{either } f(i, j) = h \\ &\text{or } \exists (i', j') \in C_h^k \text{ and neighbor of} \\ &(i, j), \|f(i, j) - f(i', j')\| \le \Delta\}. \end{aligned} \qquad (2)$$

A flat zone is composed of pixels with low gray-level fluctuations. The particular case $\Delta = 0$ corresponds to the classical situation where the flat zones are strictly flat. Figs. 4 and 5 illustrate the evolution of the flat zones as a function of Δ. To judge intuitively this evolution, we show, on the left side of Fig. 4, images where each flat zone has been filled by its mean and, on the right side, the corresponding contours of the flat zones. When $\Delta = 0$, the image on the left side is the original image and most flat zones involve one or two pixels (right image). Fig. 5 shows the reduction of the number of flat zones. Of course, this curve has a direct relation with the evolution of the max-tree complexity.

The interest of this approach can be foreseen by looking at the flat zones corresponding to the water areas. If a "strict" binarization is used ($\Delta = 0$), the water is represented by a very high number of small flat zones and will not be processed as a single entity. By contrast, if a "soft" binarization is used, these small flat zones are grouped together to form larger entities that

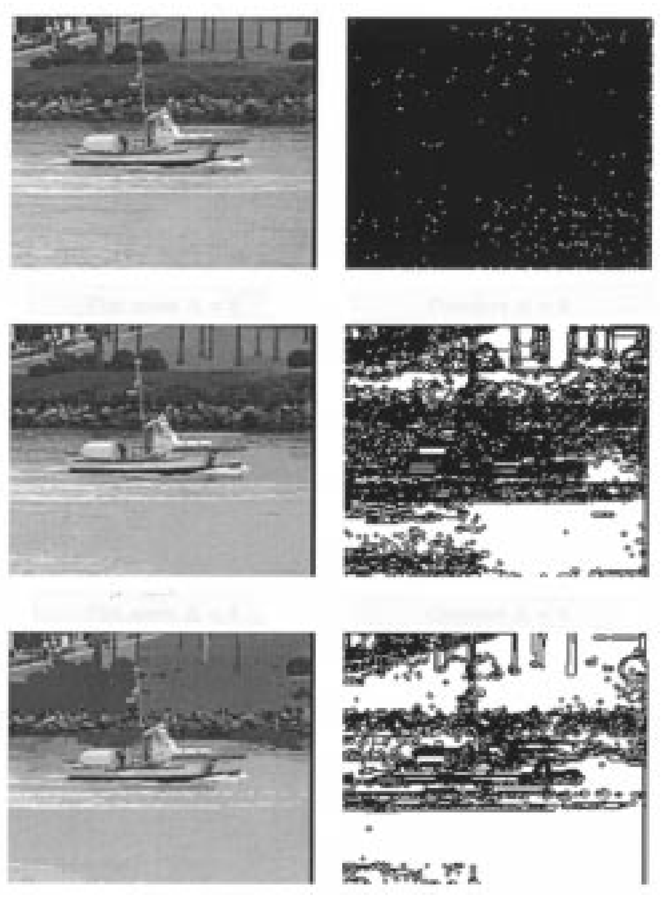


Fig. 4. Flat zones resulting from the "nonflat" binarization (2). Left: gray-level image where each flat zone has been filled with its mean. Right: contour of the flat zones. Upper left: Flat zones $\Delta = 0$. Middle left: Flat zones $\Delta = 4$. Bottom left: Flat zones $\Delta = 8$. Upper right: Contours $\Delta = 0$. Middle right: Contours $\Delta = 4$. Bottom right: Contours $\Delta = 8$.

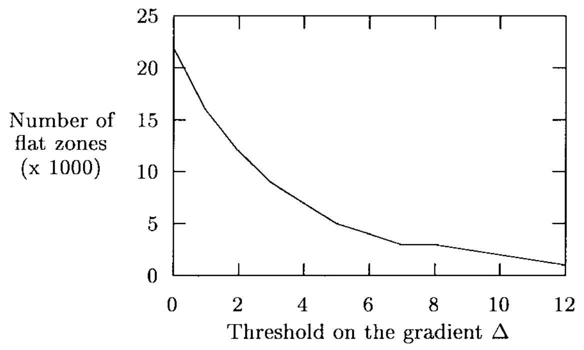


Fig. 5. Evolution of the number of flat zones of the image of Fig. 4.

will be processed in a coherent way by the connected operator. Note, however, that the pixels assigned to each node of the max-tree do not have the same gray-level value. Therefore, the restitution step after the filtering cannot simply assign the gray-level h to pixels belonging to the node C_h^k. Let us postpone this discussion until Section V dealing with restitution issues. In the sequel (and until Section V), we assume that a "strict" binarization rule ($\Delta = 0$) is used. Let us concentrate on the problem of connected components definition: once the "local" background (that is C_h^k) has been defined, its complement has to be analyzed to create the new temporary nodes at level $h+1$ (that is $\{TC_{h+1}^k\}_k$).

B. Connected Components Definition

In the case of discrete images, the simplest approach consists in selecting a connectivity (4-, 6-, or 8-connectivity) and in labeling the set of pixels of the temporary node that does not belong to the "local background" following this connectivity rule. Note that, as the binarization, this step is very important because it implicitly defines the notion of objects that will be processed by the operator. The objective of this section is to discuss modifications of the connectivity and their influence on the resulting connected operator. Let us recall the connectivity definition.

Definition 4—Connectivity Class: A *connectivity class* $\mathcal{C}$ is defined on the subsets of a set E when

1) $\emptyset \in \mathcal{C}$ and $\forall x \in E$, $\{x\} \in \mathcal{C}$;
2) for each family $\{\mathcal{C}_i\}$ of $\mathcal{C}$, $\bigcap \mathcal{C}_i \neq \emptyset \Rightarrow \bigcup \mathcal{C}_i \in \mathcal{C}$.

It was shown in [16] that this definition is equivalent to the definition of a family of *connected pointwise openings* $\{\gamma_x, x \in E\}$, associated to each point of E. Let us recall the following result.

Theorem 5—Connectivity Characterized by Openings: The definition of a connectivity class $\mathcal{C}$ is equivalent to the definition of a family of openings $\{\gamma_x, x \in E\}$ such that:

1) $\forall x \in E$, $\gamma_x(\{x\}) = \{x\}$;
2) $\forall x, y \in E$ and $X \subseteq E$, $\gamma_x(X)$ and $\gamma_y(X)$ are either equal or disjoint;
3) $\forall x \in E$ and $X \subseteq E$, $x \notin X \Rightarrow \gamma_x(X) = \emptyset$.

Intuitively, the opening $\gamma_x(X)$ extracts the connected component of X that contains x. Based on this definition of the connectivity, a generalization was proposed in [16]. It relies on the definition of a new connected pointwise opening:

$$\nu_x(X) = \gamma_x[\delta(X)] \cap X, \qquad \text{if } x \in X$$

and

$$\nu_x(X) = \emptyset, \qquad \text{if } x \notin X \tag{3}$$

where δ is an extensive dilation. This new operator ν_x is a connected pointwise opening and therefore defines a new connectivity. This connectivity is less "strict" than the usual ones in the sense that it considers that two objects that are close to each other (that is they touch each other if they are dilated by δ) belong to the same connected component. This generalization can lead to interesting connected filters. However, in this paper we concentrate on a different issue: in practice, connected operators are known to present a drawback called "leakage" that results from the connection of different objects. These connections are created because there exist thin connected paths between large objects. A solution to this problem consists in breaking the thin connections of the binary connected components and in segmenting the components into a set of elementary shapes to be processed separately. As a result, the connected operator can take individual decision on each elementary shape. Ideally, the shapes should correspond to our perception of the main parts of the object. This approach can be seen as the definition of a "strict" connectivity. To our knowledge, two attempts have been reported in the literature to define "strict" connectivities.

- *Segmentation by Openings* [17]: Given a family of *connected pointwise openings*, γ_x, and an opening γ with a connected structuring element, a new family of *connected pointwise opening*, σ_x, can be created as follows:

$$\sigma_x(X) = \gamma_x\gamma(X), \qquad \text{if } x \in \gamma(X)$$

and

$$\sigma_x(X) = \{x\}, \qquad \text{if } x \in X \setminus \gamma(X) \tag{4}$$

and as usual $\sigma_x(X) = \emptyset$, if $x \notin X$. It can be shown that σ_x is actually a *connected pointwise opening* and therefore defines a connectivity. Intuitively, this connectivity considers that the connected components of a binary set are made of the connected components of its opening by γ. The points that are removed by the opening are considered as isolated points, that are connected components of size one.

Even if this solution is theoretically sound, in practice it turns out that this way of segmenting the connected components leads to a loss of one of the main features of connected operators. In practice, connected operators are used because they can simplify while preserving the shape information of the remaining image components. Suppose now that we use an area opening of size larger than one with the connectivity defined by the *connected pointwise opening* of (4). The filter will eliminate all the isolated points (area equal to one) and all the small connected components resulting from the opening. The shape information of the remaining components will not be preserved because most of the time, this shape information relies on the set of isolated points. To solve this problem, we propose the following approach.

- *Segmentation by Watershed [6], [13]*: The idea of this approach is to rely on classical morphological segmentation tools. Morphological segmentation generally involves two steps: marker extraction and watershed segmentation [18]. The marker extraction defines the interior of the regions that should be segmented and the watershed precisely defines the contours of these regions. The segmentation procedure is illustrated by Fig. 6. The original gray-level image can be seen in Fig. 6(a), and the set of binary connected components resulting from a thresholding at level 70 in Fig. 6(b) (note that different gray-level values have been assigned to each connected component). In this last figure, there is a very large connected component involving the two speakers, the screen in the background of the scene and the letters of the word "MPEG." These objects are processed as single entity by the operator because there exist thin connections between them.

  — *Marker Extraction:* In the context of connected operators, this step defines the number of connected components created by the segmentation. A simple idea consists in using as markers the connected components of the ultimate erosion of X, denoted by $\epsilon^{\mathcal{U}}(X)$. However, in practice, a segmentation driven by the ultimate erosion creates a very large number of connected components. As an illustration, Fig. 6(g) shows the ultimate erosion of the binary components of Fig. 6(b). The number of connected components can be reduced by computing the union of $\epsilon^{\mathcal{U}}(X)$ with the erosion of X with a structuring element B of size l [denoted by $\epsilon_{B_l}(X)$]. The set of markers is defined by $M_l(X) = \epsilon^{\mathcal{U}}(X) \cup \epsilon_{B_l}(X)$. The erosion merges some connected components of the ultimate erosion. In particular, if $l = 0$, the markers are the connected components of X itself, $M_0(X) = X$ (the connected components are not broken), whereas if $l = \infty$, the set of markers is the ultimate erosion, $M_\infty(X) = \epsilon^{\mathcal{U}}(X)$. Fig. 6(d) presents the set of markers $M_3(X)$ resulting from an erosion with a structuring element of size 3 × 3.

  — *Watershed Segmentation:* Once the markers are defined, the watershed algorithm propagates them to precisely define the shape of the connected components. In [6] and [13], it is proposed to use one of the classical tools for binary segmentation that is to rely on the opposite of the distance function $-Dist_X$ to perform the propagation. This segmentation is geometric, since it only takes into account the shape of X. More formally, let us define $\mathcal{CB}_x\{-Dist_X, M_l(X)\}$ the transformation that assigns to x the *catchment basin* of the function $-Dist_X$ that contains x taking into account the markers $M_l(X)$. Consider now the operator

$$\mathcal{CC}_x^l(X) = \mathcal{CB}_x\{-Dist_X, M_l(X)\} \cap X, \qquad \text{if } x \in X$$

and

$$\mathcal{CC}_x^l(X) = \emptyset, \qquad \text{if } x \notin X. \tag{5}$$

This transformation reduces to the classical *connected pointwise opening* γ_x when $l = 0$. For $l>0$, it only creates a *pseudoconnectivity.* Indeed, in that case, all conditions of Theorem 5 are met except one: $\mathcal{CC}_x^l$ is not increasing and therefore not an opening. This is a drawback but, using the watershed as segmentation tool, our main concern is to segment the components of X in a small number of regions and to keep as much as possible the contour information of X. Moreover, in practice for small values of l, this theoretical problem does not prevent the creation of useful operators. This segmentation is illustrated in Fig. 6(e) and (h) for the two sets of markers $M_3(X)$ and $M_\infty(X)$. As can be seen, for a small value of l [Fig. 6(e)] the segmentation corresponds well to various objects of the scene. This is, however, not the case for high values of l. In particular,

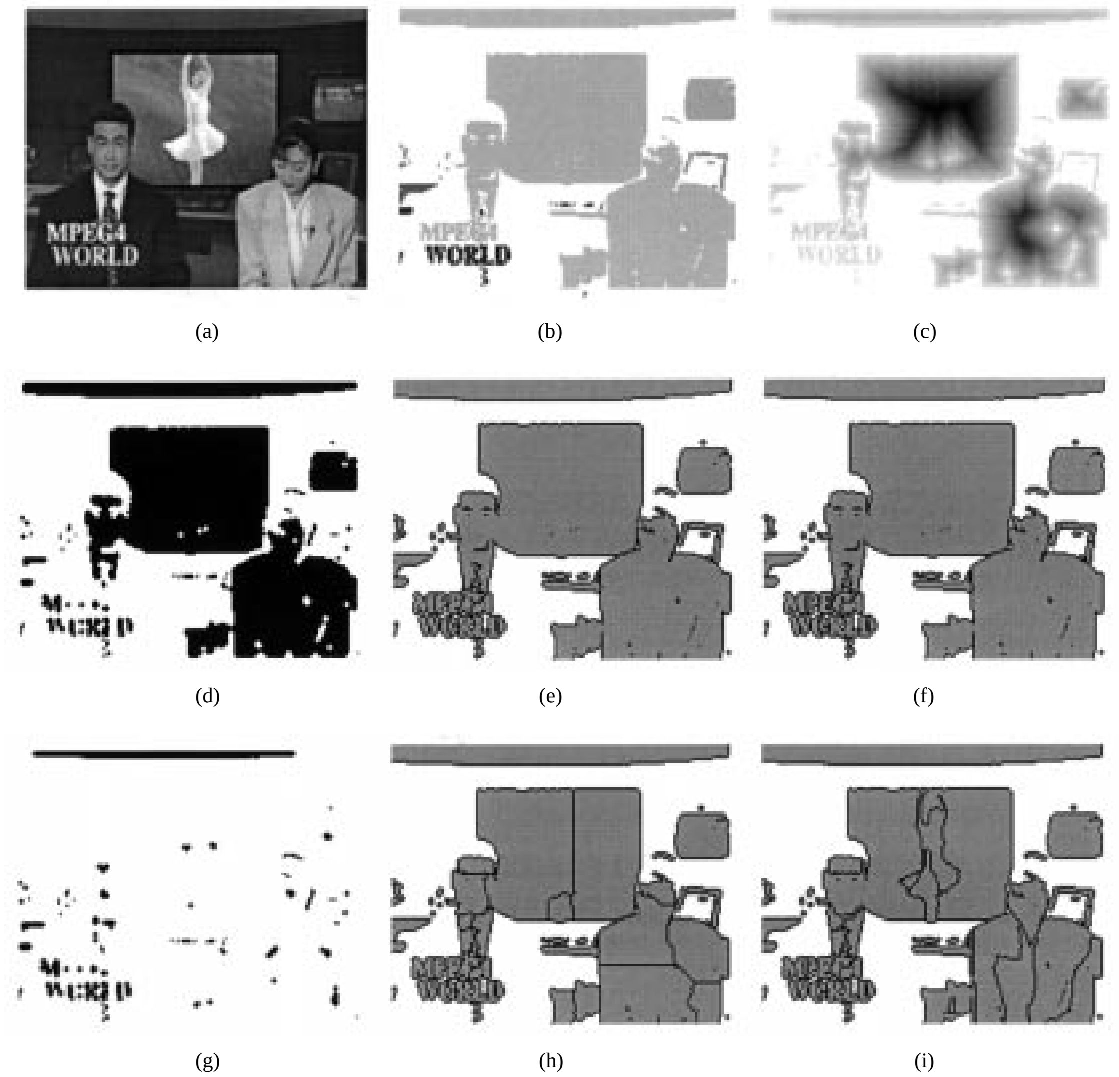


Fig. 6. Segmentation of binary components. (a) Original. (b) Binary components (X). (c) Distance function ($-Dist_x$). (d) Marker (erosion of size 3), ($M_3(X)$). (e) Geometric segmentation, Marker $M_3(X)$. (f) Gray-level segmentation, Marker $M_3(X)$. (g) Marker (ultimate erosion), ($M_\infty(X)$). (h) Geometric segmentation, Marker $M_\infty(X)$. (i) Gray-level segmentation, Marker $M_\infty(X)$.

Fig. 6(h) shows the presence of contours that are not related to the scene content (contours inside the screen or the speaker on the right side).

In [13], a modification of this approach was proposed to increase the robustness in the case of large values of l. The idea was to compute the distance function not on X but on the result of a closing by reconstruction of X. In the following, we propose an alternative approach that consists in using the gray-level information on the support of X and not its shape. To this end, one can simply replace the distance function used in $\mathcal{CB}_x$ by the gradient of the image defined on the support of X: $\mathcal{CB}'_x\{\nabla f(i, j)_{(i,j)\in X}, M_l(X)\}$, where ∇f denotes the gradient of the original function f. The set of markers is the same as previously, but the segmentation is gray-level oriented (in the sequel, this approach is called *gray-level segmentation*). The result obtained with this approach are illustrated on Fig. 6(f) and (i) As can be seen, differences are minor for low values of l but quite significant for larger values.

As mentioned in Section II, once the max-tree has been created by recursive use of binarization and connected components definition steps, a criterion is assessed for each node. If the criterion value is below a given threshold the node is removed and its pixels are moved toward the father node. Fig. 7 illustrates examples of area filtering [3] with three notions of connectivity. The area criterion consists in measuring the number of pixels of each connected component. The filter is an area open-close, that is, an area opening followed by an area closing.[1] The classical area open-close (4-connectivity) can be seen in Fig. 7(b). This example illustrates a typical leakage problem of connected operators. Small objects like the letters of the "MPEG4" word should have been removed. This is however not the case for example for the "G" and the "4" because there are thin connections between these letters

[1] If $\gamma^{\text{area}}(f)$ denotes the area opening, its dual is a closing defined by $\varphi^{\text{area}}(f) = -\gamma^{\text{area}}(-f)$.

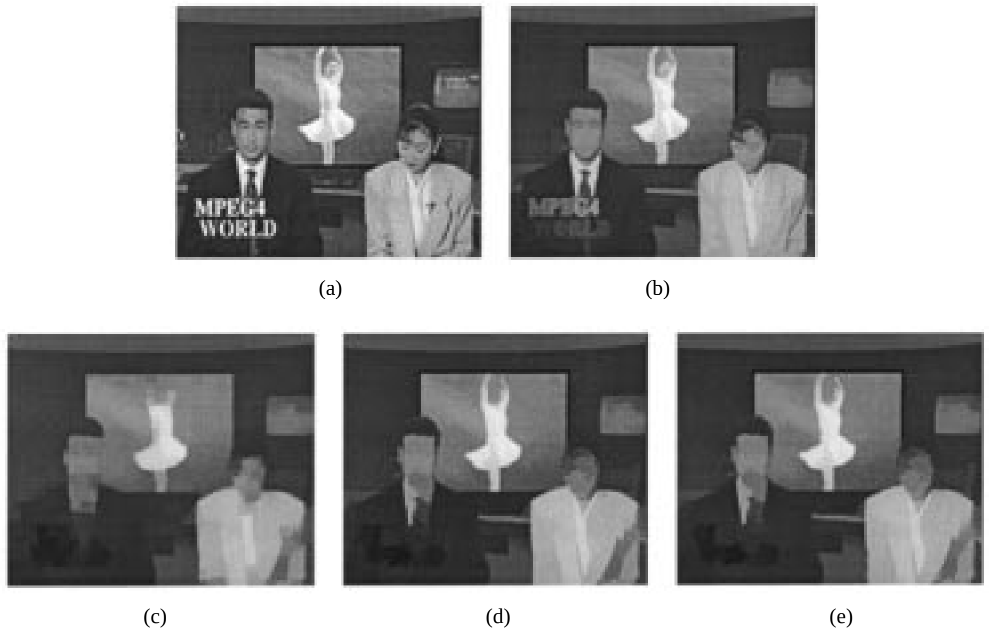


Fig. 7. Area open-close (size $\lambda = 100$) for various strict connectivities. (a) Original. (b) Area open-close with 4-connectivity. (c) Connectivity by opening (3×3) [17]. (d) Pseudoconnectivity by geometric seg. $(t = 3)$. (e) Pseudoconnectivity by gray level seg. $(t = 3)$.

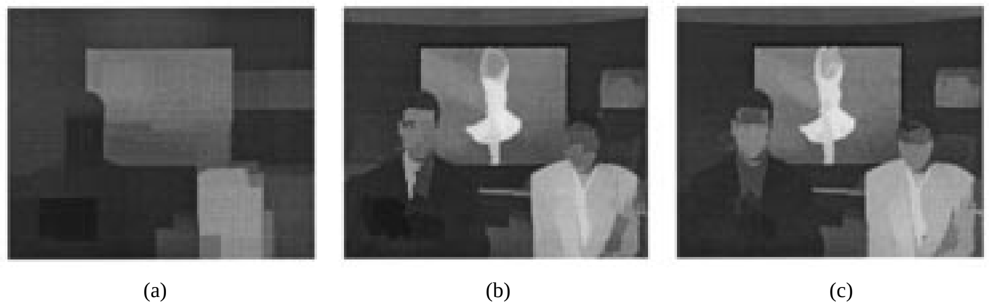


Fig. 8. Area open-close (size $\lambda = 100$) for connectivities defined by segmentation. (a) Connectivity by opening (20×20). (b) Pseudoconnectivity by geometric seg. $(l = 20)$. (c) Pseudoconnectivity by gray level seg. $(l = 20)$.

and the shirt of the man. Using the classical connectivity, the operator processes the shirt and the "G" and "4" as a single object and the connected operator reconstructs "too much." Fig. 7(c) gives the result obtained by the "segmentation by opening" with a structuring element of size 3×3. As mentioned previously, this approach does not preserve the contour information. Finally, Fig. 7(d) and (e) present similar results but with the "geometric" and the "gray-level segmentation by watershed" ($l = 3$). For low values of l, both approaches lead to very similar results. In both cases, the leakage problem has disappeared. Thin connections between components are broken and the final result corresponds more to a "natural" size-oriented simplification.

The examples of Fig. 8 illustrate the difference between the various segmentation approaches for high values of the l parameters. The segmentation by opening (20×20) shown in Fig. 8(a) is useless. Comparison between the geometric [Fig. 8(b)] and the gray level [Fig. 8(c)] segmentation reveals that the gray-level approach is more robust and introduces fewer "false" contours.

In this section, we have shown how to create (pseudo)connectivities that are more strict than classical ones. This approach allows the reduction of the leakage problem while preserving the main feature of the connected operators. However, in practice, this approach has to be used with care because it is, in some sense, in contradiction with the basic principle of connected operators, since it introduces new contours in the image. From our experience, this is not a drawback in practice if small values of l are used, that is if only thin connections are split. In this case, new contours are indeed introduced in the image, but at very specific locations corresponding generally to transitions between two different objects. Note finally that in this case, gray-level and geometric segmentation results are almost equivalent.

IV. Filtering

Once the max-tree has been created, the filtering step analyzes each node by measuring a specific criterion and takes a decision on the elimination or preservation of the node.

A. Classical Criteria

As examples, let us briefly recall some classical criteria used for the *opening by reconstruction*, the *area opening*, and the λ-max operator. In this section, we assume that the node C_h^k is analyzed and we denote by $\widehat{C_h^k}$ the set of pixels belonging to C_h^k and all its descendant nodes (in Section II, this notion has been used to explain the max-tree creation and the corresponding set of pixels was stored in a temporary node called TC_h^k).

- *Opening by Reconstruction [1]*: This filter preserves a node C_h^k if the binary erosion of the set of pixels included in $\widehat{C_h^k}$ by a structuring element of size λ is not the empty set. This operator has a size-oriented simplification effect: it removes the bright components smaller than the structuring element. By duality, a closing by reconstruction can be defined. Its simplification effect is similar to that of the opening but on dark components.
- *Gray-Level Area Opening [3]*: This filter has been used in Section III. It is similar to the previous one except that it preserves the C_h^k if the number of pixels of $\widehat{C_h^k}$ is larger than a limit λ. It has also a size-oriented simplification effect, but the notion of size is different from the one used in the *opening by reconstruction*. By duality an *area closing* can be defined.
- λ-$\max$ *Operator*: The idea is to preserve C_h^k if this node has at least one nonempty descendant node at level $h+\lambda$ ($\exists k'$ such that $C_{h+\lambda}^{k'} \cap \widehat{C_h^k} \neq \emptyset$). The simplification effect of this operator is contrast-oriented in the sense that it eliminates image components with a contrast lower than λ. The λ-$\max$ is an operator and not a morphological filter because it is not idempotent. By duality, the λ-$\min$ operator can be defined.

All these operators are extensively used in practice. A large set of examples can be found in particular in [2], [3], [10], [18]–[20], and in the references they contain.

B. Nonincreasing Criteria

The classical criteria and the resulting operators are increasing. In the lattice framework, an operator ψ is said to be increasing if it does not modify the order between any pair of elements of the lattice $\forall x \leq y$, $\psi(x) \leq \psi(y)$. In the context of connected operators, this property is directly related to the criterion assessed for each node. Assume that the node $C_{h_1}^{k_1}$ is a child node of $C_{h_2}^{k_2}$ (that is $C_{h_1}^{k_1} \cap \widehat{C_{h_2}^{k_2}} \neq \emptyset$). Because of the tree structure, we have the following relations: $h_1 \geq h_2$ and $\widehat{C_{h_1}^{k_1}} \subseteq \widehat{C_{h_2}^{k_2}}$. If the criterion $\mathcal{M}(.)$ is increasing, we have also $\mathcal{M}(\widehat{C_{h_1}^{h_1}}) \leq \mathcal{M}(\widehat{C_{h_2}^{k_2}})$.

Let us analyze the effect of having an increasing or a nonincreasing criterion on the max-tree representation and on the decision process. Consider a maximum of the image, that is a leaf node of the max-tree, and the sequence of all its ancestor nodes going down to the root node. In the example of Fig. 9, if we start by the maximum corresponding to C_2^3, the sequence is $C_2^3 \rightarrow C_1^1 \rightarrow C_0^1$. Consider now the sequence of the criterion values $\mathcal{M}(C_h^k)$ obtained by scanning successively all the ancestors of a maximum. In the example of Fig. 9, the *criterion sequence* starting from C_2^3 is $M(h) = [\mathcal{M}(C_2^3), \mathcal{M}(C_1^1), \mathcal{M}(C_0^1)]$ and is represented as a curve (function of h) on the right side. Note that the parameter

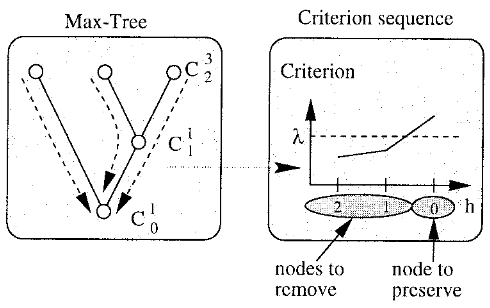


Fig. 9. Max-tree and *criterion sequence* for each local maximum.

h itself is decreasing because the nodes are scanned starting for the maximum and going down to the root. If the criterion is increasing, the *criterion sequence* is itself increasing and there is no problem defining the level h where the criterion is higher than a given limit λ. In this case, all nodes such that $\mathcal{M}(C_h^k) < \lambda$ are removed and the corresponding pixels are moved to the first ancestor node such that $\mathcal{M}(C_h^k) \geq \lambda$.

If the criterion is nonincreasing, the *criterion sequence* $M(h)$ may fluctuate around the λ value and the definition of the set of nodes to remove is less straightforward. Three rules have been reported in the literature [9], [10], [21] to deal with the nonincreasing case, as follows:

1) *"Direct" Decision*: This is the most straightforward approach that consists in considering that a node C_h^k is preserved if and only if $\mathcal{M}(C_h^k) \geq \lambda$. The content of the nodes that are removed are merged with their nearest preserved ancestors. This solution is illustrated in Fig. 10(b). In this figure, we show on the upper level, a simple gray-level function (input) and the corresponding output function. Components with a criterion value higher than λ are shown in dotted line on the output function. Components with a criterion value lower than λ are not shown. On the lower level of the figure, we show the evolution of the *criterion sequence* and the corresponding nodes (and therefore components) to be preserved.

   The "direct" approach has the advantage of being simple; however, in practice it is not robust because the decisions are local and do not depend on the decision of neighboring nodes (in the case of an increasing criterion, the decision is also local but, because of the increasing property, the relation between the decisions on various levels is known).
2) *"Min" Decision:* A node C_h^k is preserved if $\mathcal{M}(C_h^k) \geq \lambda$ and if all its ancestors are also preserved. This rule preserves less nodes than the first one. It is illustrated in Fig. 10(a).
3) *"Max" Decision*: The last rule is the dual of the previous one: a node is removed if $\mathcal{M}(C_h^k) < \lambda$ and if all its descendant nodes are also removed [see Fig. 10(c)].

From our practical experience, the min and max rules are generally more robust and lead to more coherent decisions. The direct decision is the less robust approach and gives noisy results (presence of isolated small connected components that should "intuitively" be removed but are not). However, this

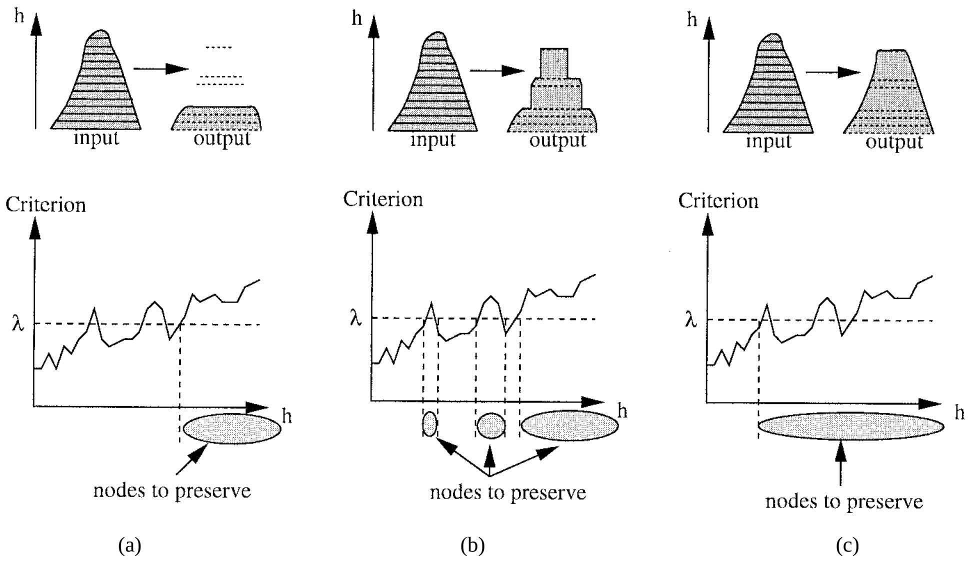


Fig. 10. Illustration of various decision rules in the case of nonincreasing criterion. (a) "Min" decision. (b) "Direct" decision. (c) "Max" decision.

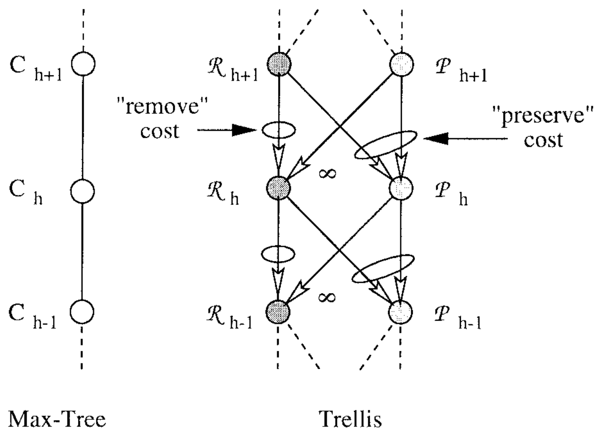


Fig. 11. Trellis construction for the decision in the case of a single branch tree.

conclusion is highly influenced by the type of criterion and its "degree of nonincreasingness." In [7], an improvement of the decision robustness is proposed. It consists in filtering the decision sequence, for example, with a median filter. This solution actually provides more robustness, however, in the following, a different solution is proposed. It turns out to be much more robust than any of the previous ones. It relies on a formulation of the decision process as an optimization problem.

For each node C_h^k, a binary decision, preserve or remove, has to be taken. Assume that a decision cost is assigned to each possible decision and to each node. In this case, the decision problem can naturally be seen as finding the set of lowest cost paths that go from the leaves of the tree down to the root node. Let us describe this approach in detail with a simple tree involving a single branch as shown in Fig. 11.

First, assign to each node C_h of the max-tree two states, $\mathcal{P}_h$ and $\mathcal{R}_h$, describing the two possible decisions, "preserve" or "remove." Second, a trellis is constructed, as shown in Fig. 11, by creating transitions linking the possible decisions of one node and of its father. Between C_h^k and C_{h-1}^k, there are four possible transitions: $\mathcal{R}_h \rightarrow \mathcal{R}_{h-1}$, $\mathcal{R}_h \rightarrow \mathcal{P}_{h-1}$, $\mathcal{P}_h \rightarrow \mathcal{R}_{h-1}$, and $\mathcal{P}_h \rightarrow \mathcal{P}_{h-1}$. Furthermore, a cost is assigned to each transition. The same cost is assigned to the two transitions going to a preserve state. This cost should reflect the reliability of the preserve decision for that node. This reliability can be measured for example by the difference between the decision threshold and the criterion value $\lambda - \mathcal{M}(C_h)$ (if the reliability is very high, the cost is very low). In the case of a transition emanating from a remove state and going to a remove state, the situation is similar and the value $\mathcal{M}(C_h) - \lambda$ can be assigned as transition cost. This is, however, not the case for the transition emanating from a preserve state and coming to a remove state. Indeed, these transitions should be avoided because we want to define a level h above which all nodes are removed and below which all nodes are preserved. We choose this strategy to get back to the situation where the criterion is indeed increasing. These transitions can be eliminated if an infinite cost is assigned to them. Now, the decision consists in finding in this trellis, the path of lowest cost that starts from the maximum and ends in the preserve state of the root node (at least, the root node should be preserved). The cost of a path is defined as the sum of the costs of its transitions.

This formulation is a classical dynamic programming problem that can be very efficiently solved by the Viterbi algorithm [22]. Let us briefly describe this algorithm: assume that the two optimum (lowest cost) paths starting from the maximum and ending at $\mathcal{P}_{h+1}$ and $\mathcal{R}_{h+1}$ are known. Let us call $\text{Path}_{h+1}^{\mathcal{P}}$ and $\text{Path}_{h+1}^{\mathcal{R}}$ these two optimum paths. The definition of the two optimum paths ending at $\mathcal{P}_h$ and $\mathcal{R}_h$ can be easily defined by a local decision. For example, the optimum path ending in $\mathcal{R}_h$, that is $\text{Path}_h^{\mathcal{R}}$, is defined by the following rule:

$$\begin{aligned}
&\text{If} \quad \text{Cost}\,(\text{Path}_{h+1}^{\mathcal{P}}) + \text{Cost}(\mathcal{P}_{h+1} \rightarrow \mathcal{R}_h) \\
&\qquad \leq \text{Cost}(\text{Path}_{h+1}^{\mathcal{R}}) + \text{Cost}(\mathcal{R}_{h+1} \rightarrow \mathcal{R}_h), \\
&\text{then} \quad (\text{Path}_h^{\mathcal{R}}) = (\text{Path}_{h+1}^{\mathcal{P}}) \cup \{\mathcal{P}_{h+1} \rightarrow \mathcal{R}_h\}, \\
&\text{else} \quad (\text{Path}_h^{\mathcal{R}}) = (\text{Path}_{h+1}^{\mathcal{R}}) \cup \{\mathcal{R}_{h+1} \rightarrow \mathcal{R}_h\}. \qquad (6)
\end{aligned}$$

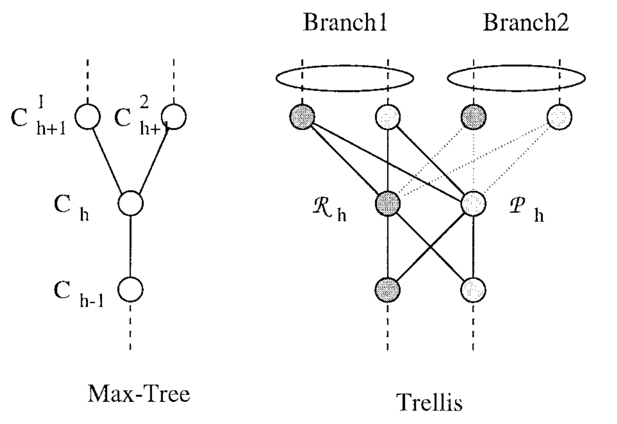


Fig. 12. Trellis construction for the decision in the case of a multiple branches tree.

This rule simply states that the optimum path ending at state $\mathcal{R}_h$ has to go through either state $\mathcal{R}_{h+1}$ or state $\mathcal{P}_{h+1}$ and that the best path is the one leading to the lowest additive cost. A similar decision rule can be defined for the best path ending at state $\mathcal{P}_h$. This process is iterated until the root node $h = 0$ is reached and the optimum path is progressively constructed on the basis of local decisions. Finally, once the optimum path is found, the states it goes through define the decisions for each node.

This rather simple procedure has to be extended to deal with trees with various branches. The extension is described in Fig. 12 in the case of the junction of two branches, but the procedure is general and can deal with an arbitrary number of branches. Let us analyze the case of the state $\mathcal{R}_h$: There is not one but two optimum paths ending at this state. One path comes from branch 1 whereas another one comes from branch 2. They are independent from each other. As a result, we have to define independently these two paths. In Fig. 12, two sets of transitions, identified by solid and doted lines, can be seen. The decision defined by (6) is used on both sets of transitions. Once these two optimum paths have been defined, their union is considered as "the optimum path" ending at state $\mathcal{R}_h$ and its cost is equal to the sum of the costs of two paths.

In the following section, the interest in this decision algorithm will be illustrated. At this stage, let us mention that this algorithm is very robust. In practice, the robustness means that similar input images lead to similar output results. From our practical experience, the decision using the Viterbi algorithm is drastically superior to the other approaches. This advantage is obtained because the decision is global and not local as for the direct, min, or max decisions. The robustness of the Viterbi approach is also reflected by the fact that the decisions do not strongly depend on the cost assigned to each transition. For example, in practice, similar results are obtained if the costs proposed previously are replaced by their sign: either 1 or -1. Finally, when the criterion is increasing, all restitution techniques are equivalent.

The decision algorithm proposed in this section allows us to deal in a robust way with a very large number of criteria. For a given application, one has simply to characterize by a measure the flat zones of interests. In the following sections, several examples are described and illustrated. We have selected one purely geometrical criterion: *the simplicity*, one criterion purely dealing with the gray-level distribution: *the entropy*, and finally, one devoted to image sequences: *motion*. None of these criteria is increasing and the decision technique using the Viterbi algorithm is assumed to be used in all cases (except if stated differently).

C. Example of Geometrical Criterion: Simplicity

In [6], a connected operator dealing with the complexity of objects in a context of segmentation-based coding is proposed. The idea is to define an operator that removes complex connected components. To this end, simplification criteria relying on the ratio between the area $\mathcal{A}$ and the perimeter $\mathcal{P}$ can be used:

$$\text{Complexity}(C_h^k) = \mathcal{P}(\widehat{C_h^k})/\mathcal{A}(\widehat{C_h^k})$$

and

$$\text{Simplicity}(C_h^k) = \mathcal{A}(\widehat{C_h^k})/\mathcal{P}(\widehat{C_h^k}). \tag{7}$$

In the following, we will use the simplicity criterion because it is easier to combine with the optimization procedure described in the previous section. This criterion is not the compactness criterion [23] defined as the ratio between the area and the square of the perimeter. The compactness criterion is size independent, whereas the simplicity criterion is size dependent. Of course, the selection of a particular criterion mainly depends on the application. Interest in the simplicity criterion can be seen in segmentation-based coding applications (for which it was designed). Indeed, in segmentation-based coding, one has often to decide if a specific area of the image has to be segmented or not. In the first case, the contours of the region are sent to the receiver, and part of the coding cost is proportional to the length of the contour to code, that is the perimeter. In the second case, the area is considered as texture information, and its coding cost is generally proportional to its area. As can be seen, the simplicity operator allows the classification of the objects following a contour/texture cost criterion and may simplify the coding decision problem. The names *simplicity* and *complexity* were assigned to the criteria of (7) because intuitively, it can be seen that if a connected component has a small area but a very long perimeter, it corresponds to a complex object.

The simplicity operator removes complex and bright objects from the original image. As usual, a dual operator dealing with dark objects can be defined. An example of processing can be seen in Fig. 13. The original image is composed of various objects with different complexity. In particular the text and the texture of the fish can be considered as being complex by comparison with the shape of the fish and the books on the lower right corner. Fig. 13(b) shows the output of the simplicity operator. On this result, the dual operator is applied [Fig. 13(c)]. The global processing can be considered as an alternated operator. As illustrated on this example, the simplicity operators efficiently remove complex image components (text and texture of fish) while preserving the contours of the objects that have not been eliminated. In both

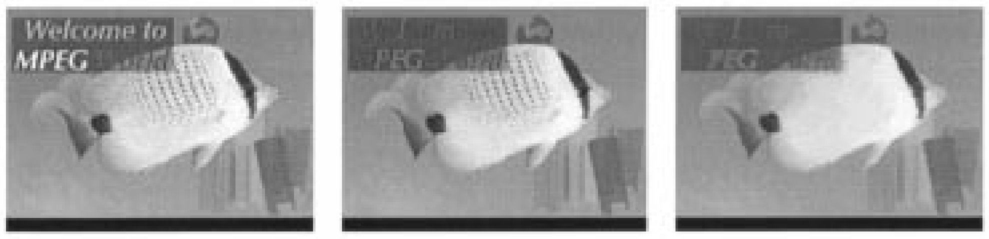

(a) (b) (c)

Fig. 13. Example of processing with the simplicity connected operator. (a) Original image. (b) Simplicity operator. (c) Dual operator.

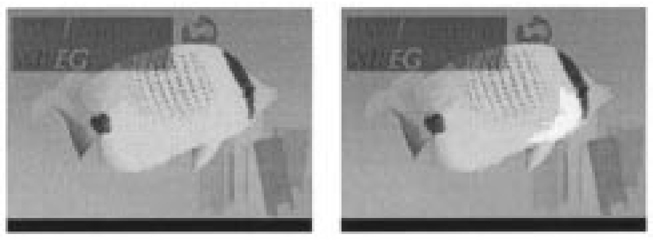

(a) (b)

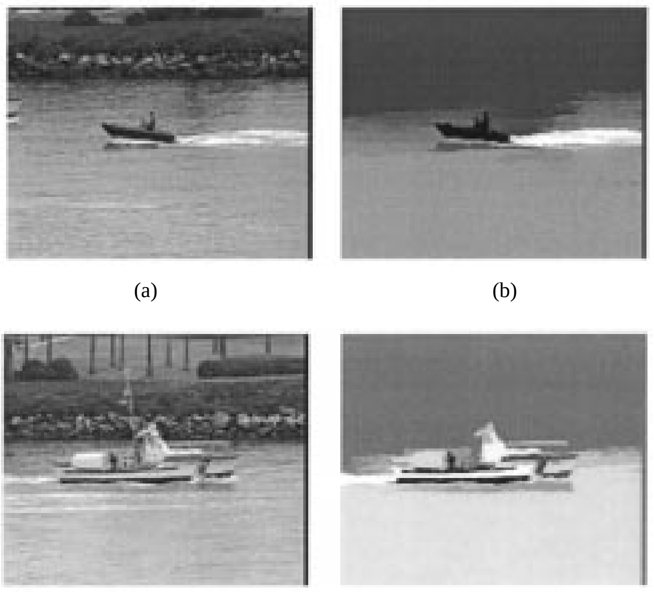
Fig. 14. Comparison between the Viterbi and direct decisions with the simplicity operator. (a) Decision with Viterbi algorithm. (b) "Direct" decision.

cases, the simplicity threshold has been set to one. Note that the simplification effect is not size-oriented, because the filters have removed large objects (the "MPEG" word) as well as small objects (the texture of the fish). The simplification is not contrast-oriented as can be seen by the difference in contrast between "Welcome to" and "MPEG" which have been jointly removed.

The simplicity criterion is not increasing. Fig. 14 makes a comparison between the decision using the Viterbi algorithm and the "direct" decision. The lack of robustness of the direct decision can be noticed in the lower right part of the fish where one bright component that has no specific visual importance appears. This component has not been removed because its simplicity is just above the threshold. However, all the components that are just above and below it have a simplicity below the threshold and are removed. In the sequel, the approach using the Viterbi algorithm is assumed to be used when the criterion is not increasing.

D. Example of Gray-Level Criterion: Entropy

The complexity and simplicity criteria of the previous section involve a purely geometrical characterization of the connected components. An opposite approach consists in selecting the components only on the basis of the gray-level distribution of the pixels inside their support. For each connected component, the histogram of the gray-level values can be computed and a specific characteristic can be assessed on the histogram in order to decide if this connected component has to be preserved or removed. In the following, we illustrate an example that consists in measuring the entropy of the gray-level distribution. The entropy measures from a statistical viewpoint the amount of information given by each connected component. Let us recall its definition. Once the histogram of the pixels belonging to a connected component has been


(a) (b)

(c) (d)

Fig. 15. Example of processing with the entropy connected operator. The entropy operator has removed all components with entropy lower than 5 b. (a) Original. (b) Entropy operator + dual. (c) Original. (c) Entropy operator + dual.

computed, the probability of each gray-level value can be estimated. Let $p_{C_h^k}(l)$ denote the probability of occurrence of the gray level l as estimated on the histogram of pixels belonging to the component $\widehat{C_h^k}$. The entropy measured in bits is defined as

$$\text{Entropy}(C_h^k) = -\sum_l p_{C_h^k}(l)\,\log_2[p_{C_h^k}(l)]. \tag{8}$$

The entropy of an area of constant value is equal to zero, whereas the entropy is maximum for a random texture of uniform probability density function. As the simplicity criterion, the entropy criterion is not increasing. Two examples are shown on Fig. 15. For each example, the original image and the result of applying the entropy connected operator followed by its dual are presented. The entropy operator has removed all components with entropy lower than 5 b. The operator output has been normalized for a better visualization. As can be seen, the operator has removed most of the texture information and most of the smooth gray-level variation. In each case, it has preserved the boat while producing a simple image made of flat zones. These images can be used as input to a simple segmentation algorithm in order to get a representation of the scene with a reduced number of objects. The entropy operator can also be used for coding applications, since it provides a classification of the image content as a function of its entropy.

E. Example of Sequence Criterion: Motion

The last example of criterion deals with the motion information in image sequences. Denote by $f_t(i, j)$ an image sequence where i and j represent the coordinates of the pixels and t the time instant. Our objective now is to define a connected operator able to eliminate the image components that do not undergo a given motion. The first step is therefore to define the motion model giving, for example, the displacement field at each position $\{\Delta_i(i, j), \Delta_j(i, j)\}$. The field can be constant $\{\Delta_i, \Delta_j\}$ if one wants to extract all objects following a translation, but in general the displacement can depend on the spatial position (i, j) to deal with more complex motion models such as affine or quadratic models.

The sequence processing is performed as follows: each frame is transformed into its corresponding max-tree representation and each node C_h^k is analyzed. To check whether or not the pixels contained in a given node C_h^k are moving in accordance to the motion field $\{\Delta_i(i, j), \Delta_j(i, j)\}$, a simple solution consists in considering the region created by the pixels of $\widehat{C_h^k}$ and to compute the opposite of the mean displaced frame difference (DFD) ($\mathcal{D}$) of this region with the previous frame. Note that, the opposite of the mean DFD is used so that the criterion value for a region that has to be preserved is higher than the corresponding value when the region has to be removed. The criterion can be expressed [7] as

$$\mathcal{D}_{f_t}^{f_{t-1}}(C_h^k) = \frac{-\sum_{i,j\in\widehat{C_h^k}} |f_t(i, j) - f_{t-1}(i-\Delta_i, j-\Delta_j)|}{\sum_{i,j\in\widehat{C_h^k}} 1}. \tag{9}$$

In practice, however, it is not very reliable to state on the motion of part of the image on the basis of only two frames. The criterion should have a reasonable memory of the past decisions. This idea can be easily introduced in the criterion by adding a recursive term. Two mean DFD's are measured: one between the current frame f_t and the previous frame f_{t-1} and a second one between the current frame and the previous *filtered* frame $\Psi(f_{t-1})$ (Ψ denotes the connected operator). The motion criterion is finally defined as:

$$\text{Motion}(C_h^k) = \alpha\mathcal{D}_{f_t}^{f_{t-1}}(C_h^k) + (1-\alpha)\mathcal{D}_{f_t}^{\Psi(f_{t-1})}(C_h^k) \tag{10}$$

where $0 \leq \alpha \leq 1$. If α is equal to one, the criterion is memoryless, whereas low values of α allow the introduction of an important recursive component in the decision process. In a way similar to all recursive filtering schemes, the selection of a proper value for α depends on the application: if one wants to detect very rapidly any changes in motion, the criterion should be mainly memoryless ($\alpha \approx 1$), whereas if a more reliable decision involving the observation of a larger number of frames is necessary, then the system should rely heavily on the recursive part ($0 \leq \alpha \ll 1$).

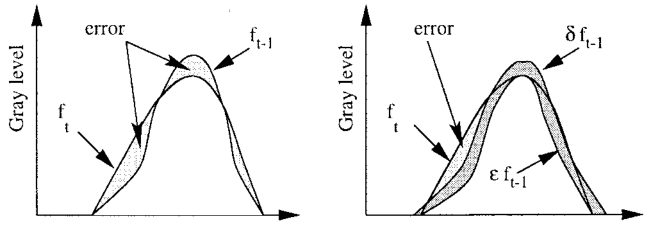


Fig. 16. Motion criterion. Left: criterion for one motion parameter. Right: criterion for a range of motion.

The motion criterion described by (9) and (10) deals with one set of motion parameters. Objects that do not follow the given motion are removed. For some applications, it may be useful to preserve objects that are within a given range of motion (notion of "motion bandwidth"). To this end, the criterion of (9) can be modified by introducing an erosion ϵ and a dilation δ of the previous frame. The difference $|f_t - f_{t-1}|$ in the DFD ($\mathcal{D}$) is replaced at each point (i, j) either by $f_t - \delta(f_{t-1})$ if $f_t > \delta(f_{t-1})$, or by $\epsilon(f_{t-1}) - f_t$ if $f_t < \epsilon(f_{t-1})$, or by zero if $\epsilon(f_{t-1}) \leq f_t \leq \delta(f_{t-1})$. This approach is illustrated in Fig. 16. As can be seen, the erosion and the dilation of f_{t-1} create a "tube" in which the function f_t can remain without contributing to the DFD. The size of the structuring element used in the dilation and the erosion defines the motion "bandwidth."

A first motion filtering example is shown in Fig. 17. The objective of the operator is to remove all moving objects. The motion model is defined by $(\Delta_i, \Delta_j) = (0, 0)$. 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 application of the connected operator $\Psi(f)$ described previously removes all bright moving objects [Fig. 17(b)]. The application of the dual operator $\Psi^*(f)$ removes all dark moving objects [Fig. 17(c)]. The residue (that is the difference with the original image) presented in Fig. 17(d) shows what has been removed by the operator. As can be seen, the operator has very precisely extracted the ballerina and the (moving) details of the speaker's face.

The example illustrated in Fig. 18 shows a decomposition of the original image into three sequences. The original sequence shows two boats on a river. The camera is following the black boat in the center. Therefore, the river and the background have an apparent motion (called the dominant motion), whereas the black boat is still. First the dominant translation is estimated giving the following motion model $(\Delta_i, \Delta_j) = (2, 0)$. Objects following this translation are obtained by application of the motion operator followed by its dual. As can be sen in Fig. 18(b), the background and the river regions are obtained. Then, the difference between the original frame and the filtered frame is computed. This difference involves only the two boats. On this difference also called residue, still objects $(\Delta_i, \Delta_j) = (0, 0)$ are extracted. As shown on Fig. 18(c), the black boat has been extracted. Finally, the remaining components are shown in Fig. 18(d). This is a decomposition in the sense that the sum of the

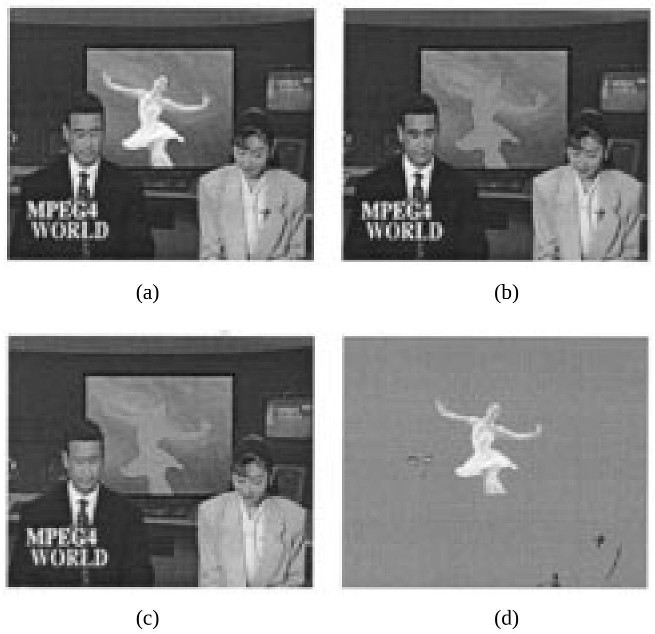


Fig. 17. Example of motion connected operator preserving fixed objects. (a) Original frame f. (b) Motion connected operator $\Psi(f)$. (c) Dual operator $\Psi^*(\Psi(f))$. (d) Residue $f - \Psi^*(\Psi(f))$.

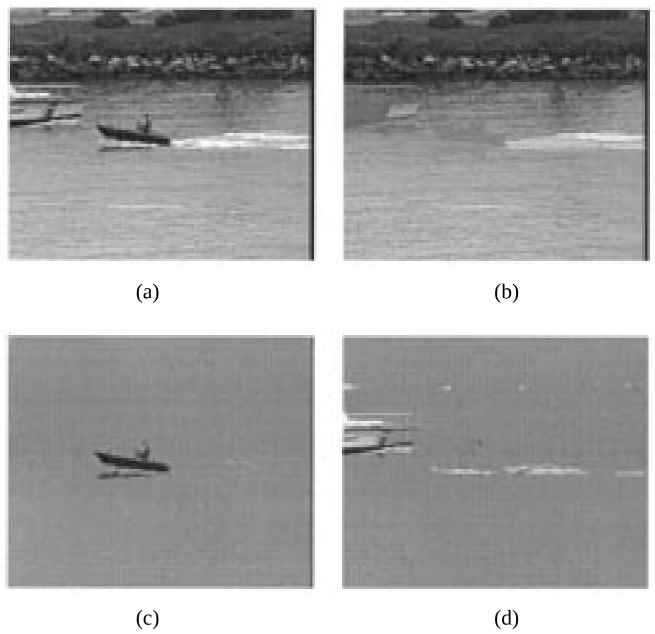


Fig. 18. Example of motion-oriented decomposition (A = B + C + D). (a) Original frame. (b) Objects with translation (2,0). (c) Objects with translation (0,0). (d) Remaining objects.

three sequences restores the original sequence. As can be seen, the filtering has separated the background and the two boats moving in two different directions.

The motion connected operator can potentially be used for a large set of applications. It opens the door in particular to different ways of handling the motion information. Indeed, generally, motion information is measured without knowing anything about the image structure. Connected operators take a different viewpoint by making decisions on the basis of the analysis of all possible flat zones, that is, of all possible structures of the image. 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. The application of theses operators to motion-oriented segmentation of sequences as well as to motion estimation seems to be a very interesting field of research.

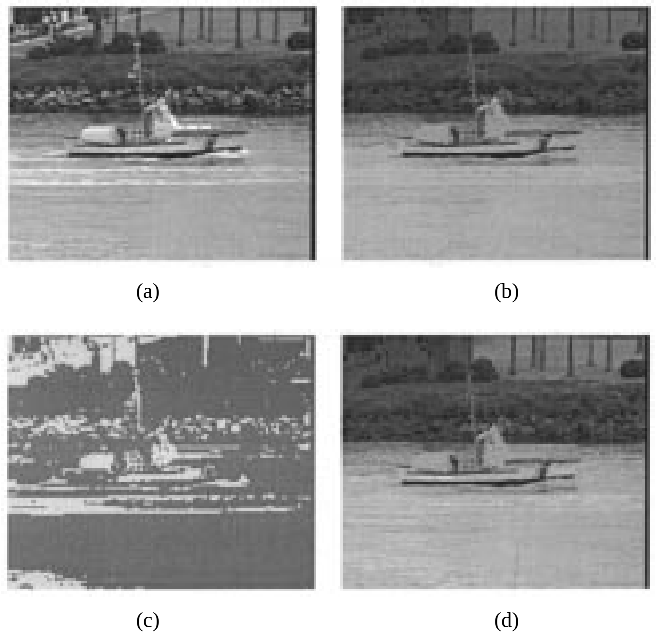


Fig. 19. Comparison between "flat" and "nonflat" area opening. (a) Original. (b) "Flat" area opening. (c) Decision map. (d) Nonflat area opening.

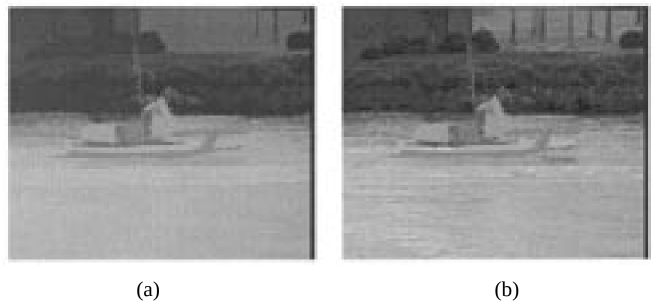


Fig. 20. Examples of "flat" and "nonflat" area filtering. (a) Flat area open-close. (b) Nonflat area open-close.

V. Image Restitution

After the max-tree creation, the criterion assessment and the decision, the last step of the filtering process consists in transforming the output max-tree into an output image. In all the previous examples, it has been assumed that the following procedure was used: assign to pixel $p(i, j)$ the gray-level value h of the node C_h^k it belongs to. This is one of the simplest rules, but it can be modified for specific applications.

Indeed, in the previous section, it has been seen that the decision classifies the nodes, and their corresponding pixels, into two classes: nodes to be removed and nodes to be preserved. A different restitution technique can therefore be assigned to each class. This approach can be seen as a "toggle

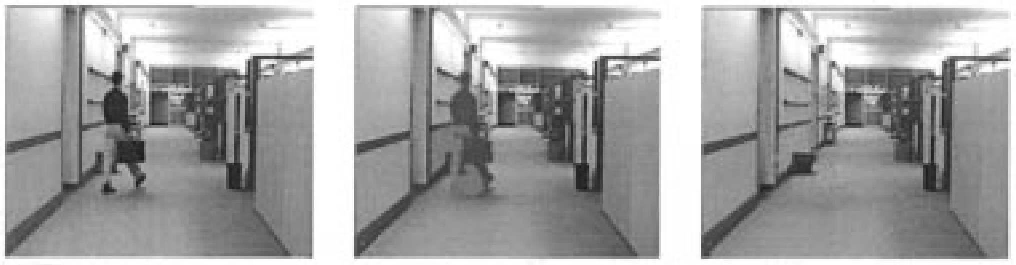


Fig. 21. (a) Original image. Example of restitution (b) without compensation and (c) with motion compensation in the case of motion connected operators.

mapping" problem [24]. It is quite natural to assume that, if a node has to be preserved, its content should not be modified by the connected operator. As a result, the gray-level values of the original image are assigned to the pixels of preserved nodes. By contrast, nodes to be removed correspond to areas that should disappear from the image. One approach consists in estimating the gray-level values that would be seen if this area was actually not present in the image. In the sequel, two examples are described. The first one involves area filters where the flat zones are defined by (2), that is, where flat zones are not strictly flat. The second example involves a motion operator.

The first example is illustrated by Figs. 19 and 20. Fig. 19(a) and (b), respectively, show an original image and the result of a classical area opening [binarization defined by (1)] with area parameter $\lambda = 1000$. Now, a max-tree is created using the nonflat binarization approach of (2) (with $\Delta = 8$). An area opening is applied on the max-tree and a decision is taken for each node. The resulting decision map is shown in Fig. 19(c). Dark (bright) areas represent nodes to be preserved (removed). Finally, Fig. 19(d) shows the final result where pixels to be preserved are equal to their original values and pixels to be removed are set to the values they have using the "flat" approach. In this example, the decision map simply makes a selection between images of Fig. 19(a) and (b). The result of the flat approach is used as an estimate of the image gray-level values "behind" the areas to be removed. A simple technique would have been to compute the mean of the pixels of the areas to be removed. However, in practice, this approach gives results where the transitions between removed and preserved nodes are very visible. Finally, flat and nonflat area open-close results are compared in Fig. 20. The interest in the nonflat can be seen in this example: in both cases, the area open-close has eliminated objects of size smaller than 1000 pixels. However, in the nonflat case, the texture of large areas (water, background) has been preserved.

The second example is shown in Fig. 21. It is an example of sequence filtering with a motion-oriented connected operator. The objective is to preserve all image components that do not move. This sequence involves a fixed scene of a corridor with a person walking. The image of Fig. 21(b) presents the result obtained by using in cascade the motion operator followed by its dual. The classical restitution where each removed node is merged with its first nonremoved ancestor has been used. As can be seen, the person has been removed and replaced by flat zones of the background. However, since we are processing a time sequence, information of what is behind this person can be extracted from previous frames. This idea can be seen as a toggle mapping defined by the following rule: following the notations of Section IV-E, if a pixel (i, j) belongs to a node to be preserved, the output gray-level value $g_t(i, j)$ at time t is simply equal to the input gray-level value $g_t(i, j) = f_t(i, j)$. If the pixel belongs to a node to be removed, the output gray-level value can be defined by motion compensation of the previous filtered frame $g_t(i, j) = g_{t-1}(i - \Delta_i, j - \Delta_j)$. As can be seen in the example of Fig. 21(c), the compensation has successfully estimated the image content behind the person.

VI. Conclusion

This paper has focused on morphological *connected operators*. These operators interact with the signal by merging *flat zones*. As a result, they do not create any new contours and are very attractive for filtering tasks where the contour information has to be preserved.

This paper has shown that connected operators work implicitly on a structured representation of the image made of flat zones. The max-tree was proposed as a suitable and efficient structure to deal with the processing steps involved in antiextensive connected operators. The processing steps have been analyzed in details:

The first step is the max-tree creation. It relies on an iterative procedure involving a binarization step and a definition of the connected components. The binarization step can be modified in order to deal with flat zones that are not strictly flat. This approach leads to connected operators that can deal differently with textured areas. It was also shown how the connected components definition step can be viewed as a segmentation problem allowing a severe reduction of the leakage problem of classical connected operators.

The second step consists of measuring a criterion and of taking a binary decision (remove or preserve) for each connected component. The issue of nonincreasing criteria has been extensively discussed. After presenting the classical solutions, an optimization formulation of the problem has been proposed for the decision step. The optimization problem can be very efficiently solved by the Viterbi algorithm. As examples, simplicity-, entropy-, and motion-oriented connected operators have been defined and illustrated.

The last step, called restitution, creates the output image from the filtered max-tree. Depending on the application,

```
flood(h)                                      /* Flood gray level $h$ */
                                              /* First step:  propagation */
    while not hqueue-empty(h)
        p ← hqueue-first(h)
        STATUS(p) ← number-nodes(h)           /* Process p */
        for every neighbor q of p             /* 4 or 8 connectivity */
            if STATUS(q) == ``Not-analyzed''
                hqueue-add(ORI(q), q)
                STATUS(q) ← ``In-the-queue''
                node-at-level(ORI(p)) ← true
                if (ORI(q) > ORI(p))          /* We have found a child at level q */
                    m = ORI(q)
                    repeat                    /* Flood the child */
                        m ← flood(m)
                    until m = h
    number-nodes(h) ← number-nodes(h) + 1
                                              /* Second step:  define the father */
    m ← h - 1
    while m ≥ 0 and node-at-level(m) = false  /* Look for the father */
        m ← m - 1
    if m ≥ 0                                  /* Assign the father */
        i ← number-nodes(h) - 1
        j ← number-nodes(m)
        father of $\mathcal{C}_h^i$ ← node $\mathcal{C}_m^j$
    else
        $\mathcal{C}_h^i$ has no father ($\mathcal{C}_h^i$ is root node)
    node-at-level(h) ← false
    return m
```

Fig. 22. Flooding procedure for the max-tree creation.

this step can be seen as a toggle mapping allowing several restitution strategies. Finally, in the Appendix, several implementation issues are discussed. It is shown in particular that, using the max-tree representation, connected operators can be efficiently implemented and, as a result, can be very intensively used in practice.

Appendix
Fast Implementation of the Max-Tree Creation

The objective of this section is to describe an example of fast implementation of the max-tree creation. This implementation is valid for the classical case where the connected components are defined by 4- or 8-connectivity (it is not valid for the segmentation approach proposed in Section III-B). The implementation relies on the use of a hierarchical first-in-first-out (FIFO) queue, queues where each individual queue is assigned to a particular gray-level value h. These queues are used to define an appropriate scanning and processing order of the pixels. In order to create the max-tree, the following three queue functions are necessary.

- **hqueue-add(h,p)**: Add the pixel p (of gray level h) in the queue of priority h.
- **hqueue-first(h)**: Extract the first available pixel of queue of priority h.
- **hqueue-empty(h)**: Return "TRUE" if queue of priority h is empty.

We will also make use of the following notations: **number-nodes(h)** defines the number of nodes $\mathcal{C}_h^k$ at level h. Its values are initialized to zero at the beginning of the tree construction. $ORI(p)$ denotes the original gray-level value of pixel p and STATUS(p) stores the information of the pixel status: the pixel can be "not-analyzed," "in-the-queue," or assigned to the node k of level h. In this last case, STATUS$(p) = k$. As can be seen, pixel p belongs to the node $\mathcal{C}_{ORI(p)}^{\text{STATUS(p)}}$ (at least in the case of strictly flat zones).

The max-tree creation relies on a simple recursive flooding procedure. The STATUS is initialized to "not-analyzed" and one of the pixel of lowest gray-level value $h_{\min}$ is put in the queue. The tree is created by calling flood$(h_{\min})$. The flooding procedure flood(h) is precisely described in Fig. 22. It has two basic steps: the first one actually performs the propagation and the updating of the STATUS, whereas the second step defines the father/child relationships. The execution time is typically less than 1 s on a Sun-Sparc 10 for 256 $\times$ 256 images of 256 gray levels.

The computation time devoted to the analysis of the max-tree (criterion assessment and decision) is a function of the criterion complexity. In general, this amount of time remains small if the criterion can be computed recursively, that is, if it is possible to take into account the evaluation done for the child nodes while computing the criterion of the current node. This is, in particular, the case for the simplicity, entropy, and motion-connected operators discussed in this paper. In all these cases, the amount of time devoted to the criterion computation and the decision using the Viterbi algorithm is of the order of one second for max-trees resulting from 256 $\times$ 256 images. Finally, the time devoted to the restitution can be neglected. In conclusions, if a fast implementation of the max-tree creation (as the one proposed in Fig. 22) is used and if the criterion can be computed recursively, the computation time devoted to the whole filtering process is of the order of a few seconds on a Sun-Sparc 10 (256×256 images).

REFERENCES

[1] J. C. Klein, "Conception et réalization d'une unité logique pour l'analyze quantitative d'images," Ph.D. dissertation, Nancy Univ., France, 1976.

[2] L. Vincent, "Morphological gray scale reconstruction in image analysis: Applications and efficient algorithms," *IEEE Trans. Image Processing,* vol. 2, pp. 176–201, Apr. 1993.

[3] ——, "Grayscale area openings and closings, their efficient implementation and applications," in *Proc. 1st Workshop on Mathematical Morphology and Its Applications to Signal Processing*, J. Serra and P. Salembier, Eds., Barcelona, Spain, May 1993, pp. 22–27.

[4] M. Grimaud, "A new measure of contrast: The dynamics," in *Proc. SPIE Visual Communications and Image Processing'92,* San Diego, CA, July 1992, vol. SPIE 1769, pp. 292–305.

[5] C. Vachier, *Extraction de caractéristiques, segmentation d'image et morhpologie mathématique,* Ph.D. dissertation, Paris School of Mines, France, 1995.

[6] A. Oliveras and P. Salembier, "Generalized connected operators," in *Proc. Visual Communication and Image Processing,* Orlando, FL, Mar. 1996, vol. 2727, pp. 761–773.

[7] P. Salembier, A. Oliveras, and L. Garrido, "Motion connected operators for image sequences," in *Proc. EUSIPCO'96,* Trieste, Italy, vol. II, pp. 1083–1086.

[8] E. Breen and R. Jones, "An attribute-based approach to mathematical morphology," in *Proc. Int. Symp. Mathematical Morphology,* P. Maragos, R. W. Schafer, and M. A. Butt, Eds. Atlanta, GA, May 1996, pp. 41–48.

[9] J. Serra and P. Salembier, "Connected operators and pyramids," in *Proc. Conf. Image Algebra and Mathematical Morphology,* San Diego, CA, July 1993, vol. 2030, pp. 65–76.

[10] P. Salembier and J. Serra, "Flat zones filtering, connected operators and filters by reconstruction," *IEEE Trans. Image Processing,* vol. 3, pp. 1153–1160, Aug. 1995.

[11] J. Crespo, J. Serra, and R.W. Schafer, "Theoretical aspects of morphological filters by reconstruction," *Signal Process.,* vol. 47, pp. 201–225, 1995.

[12] J. Serra, "Connectivity on complete lattices," in *Proc. Int. Symp. on Mathematical Morphology,* P. Maragos, R. W. Schafer, and M. A. Butt, Eds., Atlanta, GA, May 1996, pp. 81–96.

[13] P. Salembier and A. Oliveras, "Practical extensions of connected operators," in *Proc. Int. Symp. Mathematical Morphology,,* P. Maragos, R. W. Schafer, and M. A. Butt, Eds., Atlanta, GA, May 1996, pp. 97–110.

[14] J. Crespo, "Space connectivity and translation-invariance," in *Proc. Int. Symp.on Mathematical Morphology*, P. Maragos, R. W. Schafer, and M. A. Butt, Eds., Atlanta, GA, May 1996, pp. 118–126.

[15] F. K. Potjer, "Region adjacency graphs and connected morphological operators," in *Proc. Int. Symp. on Mathematical Morphology,* P. Maragos, R. W. Schafer, and M. A. Butt, Eds., Atlanta, GA, May 1996, pp. 111–118.

[16] J. Serra, *Image Analysis and Mathematical Morphology, vol. II: Theoretical Advances.* New York: Academic, 1988.

[17] C. Ronse, "Set-theoretical algebraic approaches to connectivity in continuous or digital spaces," *J. Math. Imaging Vis.,* vol. 8, pp. 41–58, 1997. Also available at http://dpt-info.u-strasbg.fr/~ronse/.

[18] F. Meyer and S. Beucher, "Morphological segmentation," *J. Vis. Commun. Image Represent.,* vol. 1, pp. 21–46, Sept. 1990.

[19] P. Salembier and M. Kunt, "Size-sensitive multiresolution decomposition of images with rank order based filters," in *EURASIP Signal Process.,* vol. 27, pp. 205–241, May 1992.

[20] P. Salembier, L. Torres, F. Meyer, and C. Gu, "Region-based video coding using mathematical morphology," *Proc. IEEE*, vol. 83, pp. 843–857, June 1995.

[21] H. Heijmans, "Theoretical aspects of gray level morphology," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 568–592, 1991.

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

[23] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* Reading, MA: Addison-Wesley, 1992.

[24] J. Serra, "Toggle mappings," in *From Pixels to Features,* J. C. Simon, Ed. Amsterdam, The Netherlands: North Holland, 1989, pp. 61–72.

Philippe Salembier (M'96) received the engineering degree from the Ecole Polytechnique, Paris, France, in 1983, the engineering degree in telecommunications from the Ecole Nationale Superieure des Telecommunications, Paris, in 1985, and the Ph.D. degree from the Swiss Federal Institute of Technology (EPFL), Lausanne, in 1991.

From 1985 to 1989, he was with Laboratoires d'Electronique Philips, Limeil-Brevannes, France, working in digital communications and signal processing for HDTV. In 1989, he joined EPFL to work on image processing. He was a Post-doctoral Fellow at the Harvard Robotics Laboratory, Cambridge, MA, in 1991. At the end of 1991, after a stay at the Harvard Robotics Laboratory, he joined the Polytechnic University of Catalonia, Barcelona, Spain, where he teaches digital signal and image processing. His current research interests include image and sequence coding, image modeling, segmentation problems, video sequence analysis, mathematical morphology, and nonlinear filtering.

Dr. Salembier is currently serving as an Area Editor of the *Journal of Visual Communication and Image Representation* and as an AdCom Officer of the European Association for Signal Processing (EURASIP) in charge of the newsletter.

Albert Oliveras (M'82) received the Tech. Eng. degree in 1983 and the M.S. degree in computer science in 1989, both from the Universitat Politécnica de Catalunya (UPC) in Barcelona.

From October 1989 to September 1992, he was Assistant Professor of computer architecture and technology, Computer Science Faculty, UPC. In October 1992, he joined the Signal Theory and Communications Department, UPC. His current research interests include nonlinear image and signal processing and mathematical morphology.

Luis Garrido received the engineering degree in telecommunications in 1996 from Universitat Politécnica de Catalunya (UPC), Barcelona, Spain. He is currently pursuing the Ph.D. degree at the Image Processing Group, UPC, dealing with video sequence segmentation. His current areas of interest are graph-based segmentation, morphological filtering, and computer vision.