Connected Operators for Signal and Image Processing

Philippe Salembier

Universitat Politecnica de Catalunya, Barcelona, Spain
philippe@gps.tsc.upc.edu
http://gps-tsc.upc.es/imatge/

1 Introduction

Data and signal modeling for images and video sequences is experiencing important developments. Part of this evolution is due to the need to support a large number of new multimedia services. Traditionally, digital images were represented as rectangular arrays of pixels and digital video was seen as a continuous flow of digital images. New multimedia applications and services imply a representation that is closer to the real world or, at least, that takes into account part of the process that has created the digital information. Content-based compression and indexing are two typical examples of applications where new modeling strategies and processing tools are necessary:

- For content-based image or video compression, the representation based on an array of pixels is not appropriate if one wants to be able to act on objects, to encode differently the areas of interest, or to assign different behaviors to the entities represented in the image. In these applications, the notion of object is essential. As a consequence, the data modeling has to include, for example, regions of arbitrary shapes to represent objects.
- Content-based indexing applications are also facing the same kind of challenges. For instance, the video representation based on a flow of frames is inadequate for many video indexing applications. Among the large set of functionalities involved in a retrieval application, let us consider browsing. The browsing functionality should go far beyond the "fast forward" and "fast reverse" allowed by VCRs. One would like to have access to a table of contents of the video and to be able to jump from one item to another. This kind of functionality implies at least a structuring of the video in terms of individual shots and scenes. Of course, indexing and retrieval involve also a structuring of the data in terms of objects, regions, semantic notions, etc.

In both examples, the data modeling has to take into account part of the creation process: an image is created by projection of a visual scene composed of 3D objects onto a 2D plane. Modeling the image in terms of regions is an attempt to know the projection of the 3D object boundaries in the 2D plane. Video shots detection also aims at finding what has been done during the video editing process and where boundaries between elementary components have been

M. Faundez-Zanuy et al. (Eds.): NOLISP 2005, LNAI 3817, pp. 37–65, 2005.

introduced. In both cases, the notion of region turns out to be central in the modeling process. Note that regions may be spatial connected components but also temporal or spatio-temporal connected components in the case of video.

Besides the modeling issue, it has to be recognized that most image processing tools are not suited to region-based representations. For example, the vast majority of low level processing tools such as filters are very closely related to the classical pixel-based representation of signals. Typical examples include linear convolution with an impulse response, median filter, morphological operators based on erosion and dilation with a structuring element, etc. In all cases, the processing strategy consists in modifying the values of individual pixels by a function of the pixels values in a local window.

Early examples of region-based processing can be found in the literature in the field of segmentation. For example, the classical *Split & Merge* algorithm [1] defines first a set of elementary regions (the split process) and then interacts directly on these regions allowing them to merge under certain conditions.

Recently, a set of morphological filtering tools called *Connected Operators* has received much attention. Connected operators are region-based filtering tools because they do not modify individual pixel values but directly act on the connected components of the space where the image is constant, the so-called *flat zones*. Intuitively, connected operators can remove boundaries between flat zones but cannot add new boundaries nor shift existing ones. The related literature rapidly grows and involves theoretical studies [2, 3, 4, 5, 6, 7, 8, 9, 10], algorithm developments [11, 12, 13, 14, 15, 16] and applications [17, 18, 19, 20]. The goal of this paper is 1) to provide an introduction to connected operators for gray level images and video sequences and 2) to discuss the techniques and algorithms that have been up to now the most successful within the framework of practical applications.

The organization of this paper is as follows: The following section introduces the notation and highlights the main drawbacks of classical filtering strategies. Then, the next section presents the basic notions related to connected operators and discuss some early examples of connected operators. In practice, the two most successful strategies to define connected operators are based either on reconstruction processes or on tree representations. Both approaches are discussed in separate sections. Finally, conclusions are given in the last section.

2 Classical Filtering Approaches

In this section, we define the notation to be used in the sequel and review some of the basic properties of interest in this paper [21, 22]. We deal exclusively with discrete images $f[n]$ or video sequences $f_t[n]$ where n denotes the pixel or space coordinate (a vector in the case of 2D images) and t the time instant in the case of a video sequence. In the lattice of grey level functions, an image f is said to be smaller than an image g if and only if:

$$f \leq g \Longleftrightarrow \forall n, f[n] \leq g[n] \tag{1}$$

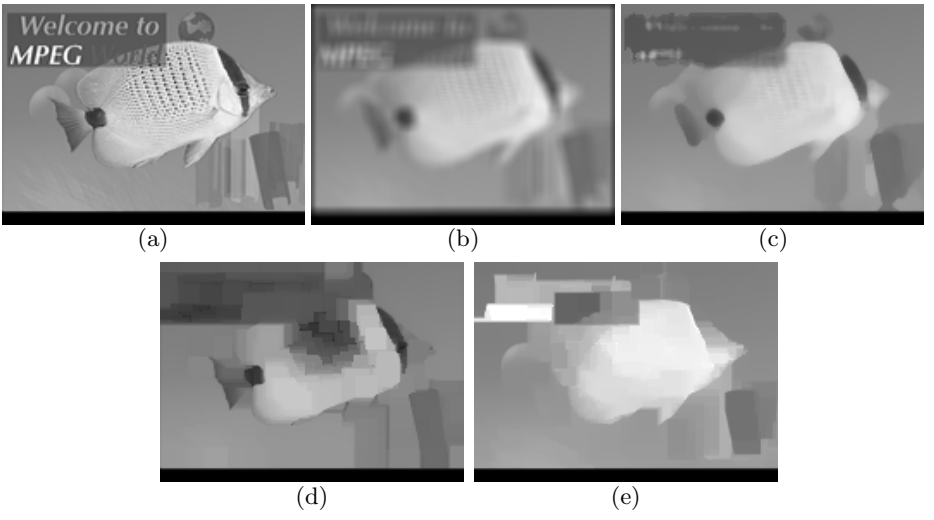


Fig. 1. Example of classical filters: (a) original image, (b) low-pass filter (7x7 average), (c) median (5x5), (d) opening (5x5), (e) closing (5x5)

An operator ψ acting on an input f is said to be:

- *increasing*: $\forall f, g, \quad f \leq g \Longrightarrow \psi(f) \leq \psi(g)$
  (The order is preserved by the filtering)
- *idempotent*: $\forall f, \quad \psi(\psi(f)) = \psi(f)$
  (Iteration of the filtering is not needed)
- *extensive*: $\forall f, \quad f \leq \psi(f)$
  (The output is always greater than the input)
- *anti-extensive*: $\forall f, \quad \psi(f) \leq f$
  (The output is always smaller than the input)
- *a morphological filter*: if it is increasing and idempotent
- *an opening*: if it is an anti-extensive morphological filter
- *a closing*: if it is an extensive morphological filter
- *self-dual*: $\forall f, \quad \psi(f) = -\psi(-f)$
  (Same processing is for bright & dark components)

Almost all filtering techniques commonly used in image processing are defined by a computation rule and a specific signal $h[n]$ that may be called impulse response, window or structuring element. Let us review these classical cases:

– Linear convolution and impulse response: the output of a linear translation-invariant system is given by: $\psi_h(f)[n] = \sum_{k=-\infty}^{\infty} h[k]f[n-k]$. The impulse response, $h[n]$, defines the properties of the filter. An example of linear filtering result is shown in Fig. 1(b). The original image shown in Fig. 1(a). As can be seen, most of the details of the original image are attenuated by the filter (average of size 7x7). However, details are not really removed but

simply blurred. The characteristics of the blurring is directly related to the extension and shape of the impulse response.

- Median filter and window: The output of a median filter with window W is defined by: $\psi_W(f)[n] = Median_{k \in W}\{f[n-k]\}$. Here also the basic properties of the filter are defined by its window. An example is shown in Fig. 1(c). Here, small details are actually removed (for example the texture of the fish). The major drawback of the filtering strategy is that every region tends to be round after filtering. This effect is due to the shape of the window combined with the median processing.
- Morphological erosion/dilation and structuring elements: morphological dilation by a structuring element $h[n]$ is defined in a way similar to the convolution: $\delta_h(f)[n] = \bigvee_{k=-\infty}^{\infty}(h[k] + f[n-k])$, where $\bigvee$ denotes the supremum (or maximum in the discrete case). The erosion is given by: $\epsilon_h(f)[n] = \bigwedge_{k=-\infty}^{\infty}(h[k] - f[n+k])$, where $\bigwedge$ denotes the infimum (or minimum in the discrete case). In practice, erosion and dilation are seldom used on their own because they do not preserve the position of contours. For example, the dilation enlarges the size of bright components and decreases the size of dark components by displacing their contours. However, they provide a simplification effect: a dilation (erosion) removes dark (bright) components that do not fit within the structuring element. Based on these two primitives, morphological opening and closing can be constructed.

  The opening is given by: $\gamma_h(f) = \delta_h(\epsilon_h(f))$ and the closing by: $\varphi_h(f) = \epsilon_h(\delta_h(f))$. These operators are morphological filters (that is, at the same time, increasing and idempotent). Moreover, the opening is anti-extensive (it removes bright components) whereas the closing is extensive (it removes dark components). The Processing results are shown in Fig.s 1(d) and 1(e). In the case of opening (closing) with a square structuring element of size 5x5, small bright (dark) components have been removed. As can be seen, the contours remain sharp and centered on their original position. However, the shape of the components that have not been removed are not perfectly preserved. In both examples, square shapes are clearly visible in the output image. This is due to the square shape of the structuring element.

Once a processing strategy has been selected (linear convolution, median, morphological operator, etc.), the filter design consists in carefully choosing a specific signal $h[n]$ which may be the impulse response, the window or the structuring element. While most people would say that this is the heart of the filter design, our point here is to highlight that, for image processing, the use of $h[n]$ has some drawbacks. In all examples of Fig. 1, $h[n]$ is not related to the input signal and its shape clearly introduces distortions in the output. The distortion effect depends on the specific filter, but for a large range of applications requiring high precision on contours, none of these filtering strategies is acceptable.

To reduce the distortion, one possible solution is to adapt $h[n]$ to the local structures of the input signal. This solution may improve the results but still remains unacceptable in many circumstances. An attractive solution to this problem is provided by connected operators. Most connected operators used in

practice rely on a completely different filtering strategy: the filtering is done without using any specific signal such as an impulse response, a window or a structuring element. In fact, the structures of the input signal are used to act on the signal itself. As a result, no distortion related to a priori selected signals is introduced in the output.

3 Connected Operators

3.1 Definitions and Basic Properties

Gray level connected operators act by merging flat zones. They cannot create new contours and, as a result, they cannot introduce in the output image a structure that is not present in the input image. Furthermore, they cannot modify the position of existing boundaries between regions and, therefore, have very good contour preservation properties.

Gray level connected operators originally defined in [2] 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 us denote by $\mathcal{P}$ a partition and by $\mathcal{P}(n)$ 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 $\forall n, \mathcal{P}_1(n) \subseteq \mathcal{P}_2(n)$.

It can be shown that the set of flat zones of an image f is a partition of the space, $\mathcal{P}_f$. Based on these notions, connected operators are defined as:

Definition 1. *(Connected operators) 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:*

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

This definition clearly highlights the region-based processing of the operator: indeed, regions of the output partition are created by union of regions of the input partition. An alternative (and equivalent) definition of connected operators was introduced in [6]. This definition enhances the role of the boundaries between regions and turns out to be very useful to derive leveling.

Definition 2. *(Connected operators) A gray level operator ψ is connected if $\forall f$ input image and $\forall n, n'$ neighboring pixels,*

$$\psi(f)[n] \neq \psi(f)[n'] \Longrightarrow f[n] \neq f[n'].$$

This definition simply states that if two neighboring pixels of the output image have two different gray level values, they have also two different gray level values in the input image, in other words, the operator cannot create new boundaries.

[1] In the context of connected operators, several studies have been carried out on the definition of less usual connectivities. The reader is referred to [22, 23, 24, 9, 8] for more details on this issue.

New connected operators can be derived from the combination of primitive connected operators. The following properties give a few construction rules:

Proposition 1. *(Properties of connected operators)*

- *If ψ is a connected operator, its dual ψ^* defined by: $\psi^*(f) = -\psi(-f)$, is also connected.*
- *If ψ_1, ψ_2 are connected operators, $\psi_2\psi_1$ is also connected.*
- *If $\{\psi_i\}$ are connected operators, their supremum $\bigvee_i \psi_i$ and infimum $\bigwedge_i \psi_i$ are connected.*

3.2 Early Examples of Connected Operators

The first known connected operator is the *binary opening by reconstruction* [25]. This operator eliminates the connected components that would be totally removed by an erosion with a given structuring element and leaves the other components unchanged. This filtering approach offers the advantage of simplifying the image (some components are removed) as well as preserving the contour information (the components that are not removed are perfectly preserved). It can be shown that the process is increasing, idempotent and anti-extensive, that is an opening. Moreover, it was called "by reconstruction" because of the algorithm used for its implementation. From the algorithmic viewpoint, if X is the original binary image, the first step is to compute an erosion with a structuring element B_k of size k, $\epsilon_{B_k}(X)$. This erosion is used to "mark" the connected components that should be preserved. The final result is obtained by progressively dilating the erosion inside the mask defined by the original image:

1. $Y_0 = \epsilon_{B_k}(X)$
2. $Y_k = \delta_C(Y_{k-1}) \bigcap X$, where C is a binary structuring element defining the connectivity, e.g. square of 3x3 (cross) for the 8-connectivity (4-connectivity).
3. Iterate step 2 until idempotence.

The first gray level connected operator was obtained by a transposition of the previous approach to the lattice of gray level functions [22, 11]. It is known as an *opening by reconstruction of erosions*:

1. $g_0 = \epsilon_{h_k}(f)$, where f is the input and h_k a structuring element of size k.
2. $g_k = \delta_C(g_{k-1}) \bigwedge f$, where C is a flat structuring element defining the connectivity, e.g. square or cross.
3. Iterate step 2 until idempotence.

It was shown in [2] that this operator is connected. Intuitively, the erosion acts as a simplification step by removing small bright components. The reconstruction process restores the contours of the components that have not been completely removed by the erosion.

There are several ways to construct connected operators and many new operators have been recently introduced. From the practical viewpoint, the most successful strategies rely on a reconstruction process or on region-tree pruning. Operators resulting from these two strategies are discussed in the sequel.

4 Connected Operators Based on Reconstruction Processes

4.1 Anti-extensive Reconstruction and Connected Operators

The Anti-extensive Reconstruction Process. The most classical way to construct connected operators is to use an anti-extensive reconstruction process. It is defined as follows:

Definition 3. *(Anti-extensive reconstruction) If f and g are two images (respectively called the "reference" and the "marker" image), the anti-extensive reconstruction $\rho^{\downarrow}(g|f)$ of g under f is given by:*

$$\begin{array}{ll} g_k & = \delta_C(g_{k-1}) \bigwedge f \text{ and} \\ \rho^{\downarrow}(g|f) & = \lim_{k\to\infty} g_k \end{array} \tag{2}$$

where $g_0 = g$ and δ_C is the dilation with the flat structuring element defining the connectivity (3x3 square or cross).

It can be shown that the series, g_k, always converges and the limit always exists. Of course by duality, an extensive reconstruction may be defined:

Definition 4. *(Extensive reconstruction) If f and g are two images (respectively called the "reference" and the "marker" image), the extensive reconstruction $\rho^{\uparrow}(g|f)$ of g above f is given by:*

$$\begin{array}{ll} g_k & = \epsilon_C(g_{k-1}) \bigvee f \text{ and} \\ \rho^{\uparrow}(g|f) & = \lim_{k\to\infty} g_k \end{array} \tag{3}$$

where $g_0 = g$ and ϵ_C is the erosion with the flat structuring element defining the connectivity (3x3 square or cross).

Note that Eqs. (2) and (3) define the reconstruction processes but do not provide efficient implementations. Indeed, the number of iterations is generally fairly high. The most efficient reconstruction algorithms rely on the definition of a clever scanning of the image and are implemented by First-in-First-out (FIFO) queues. A review of the most popular reconstruction algorithms can be found in [11]. Here, we describe a simple but efficient one: the basic idea of the algorithm is to start from the regional maxima of the marker image g and to propagate them *under* the original image f. The algorithm works in two steps:

1. The *initialization* consists in putting in the queue the location of pixels that are on the boundary of the regional maxima of the marker image. Regional maxima are the set of connected components where the image has a constant gray level value and such that every pixel in the neighborhood of the regional maxima has strictly a lower value. Algorithms to compute regional maxima can be found in [26].
2. The *propagation* extracts the first pixel, n, from the queue (note that n is a pixel of the marker image g). Then, it assigns to each of its neighbors, n',

that have a strictly lower gray level value than $g[n]$ (that is, if $g[n'] < g[n]$), the minimum between the gray level value of n and the gray level value of the pixel of the original image at the same location than n', that is $g[n'] = g[n] \bigwedge f[n']$. Finally, the pixel n' is introduced in the queue. This propagation process has to be carried on until the queue is empty. The algorithm is very efficient because the image pixels are processed only once.

In practice, useful connected operators are obtained by considering that the marker image g is a transformation $\phi(f)$ of the input image f. As a result, most connected operators ψ obtained by reconstruction can be written as:

$$\begin{array}{l} \psi(f) = \rho^{\downarrow}(\phi(f)|f) \ \text{ (anti-extensive operator), or} \\ \psi(f) = \rho^{\uparrow}(\phi(f)|f) \ \text{ (extensive operator).} \end{array} \tag{4}$$

In the following, a few examples are discussed.

Size Filtering. The simplest size-oriented connected operator is obtained by using as marker image, $\phi(f)$, the result of an erosion with a structuring element h_k of size k. It is the opening by reconstruction of erosion[2]:

$$\psi(f) = \rho^{\downarrow}(\epsilon_{h_k}(f)|f) \tag{5}$$

It can be demonstrated that this operator is an opening. By duality, the closing by reconstruction is given by:

$$\psi^*(f) = \rho^{\uparrow}(\delta_{h_k}(f)|f) \tag{6}$$

An example of opening by reconstruction of erosion is shown in Fig. 2(a). In this example, the original signal f has 11 maxima. The marker signal g is created by an erosion with a flat structuring element which eliminates the narrowest maxima. Only 5 maxima are preserved after erosion. Finally, the marker is reconstructed. In the reconstruction, only the 5 maxima that were present after erosion are visible and narrow maxima have been eliminated. Moreover, the transitions of the reconstructed signal correspond precisely to the transitions of the original signal.

As can be seen, the simplification effect, that is the elimination of narrow maxima is almost perfectly done. However, the preservation effect may be criticized: although the maxima contours are well preserved, their shape and height are distorted. To reduce this distortion, a new connected operator can be built on top of the first one. Let us construct a new marker image, $m[n]$, indicating the pixels where the reconstruction has been inactive, that is where the final result is equal to the erosion.

$$m[n] = \begin{cases} f[n] \text{ , if } \rho^{\downarrow}(\epsilon_{h_k}(f)|f)[n] = \epsilon_{h_k}(f)[n] \\ 0 \qquad \text{otherwise.} \end{cases} \tag{7}$$

[2] Note that it can be demonstrated that the same operator is obtained by changing the erosion, ϵ_{h_k}, by an opening, γ_{h_k}, with the same structuring element, h_k.

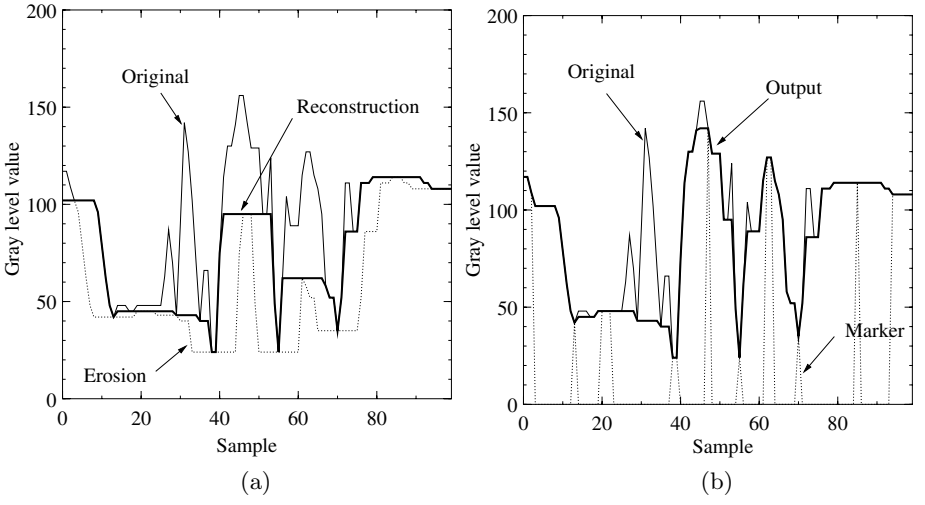


Fig. 2. Size-oriented connected operators: (a) Opening by reconstruction, (b) New marker indicating where the reconstruction has been inactive and second reconstruction

This marker image is illustrated in Fig. 2(b). As can be seen, it is equal to 0 except for the five maxima that are present after erosion and also for the local minima. At that locations, the gray level values of the original image, $f[n]$, are assigned to the marker image. Finally, the second connected operator is created by the reconstruction of the marker, m under f:

$$\psi(f) = \rho^{\downarrow}(m|f) \tag{8}$$

This operator is also an opening by reconstruction. The final result is shown in Fig. 2(b). The five maxima are better preserved than with the first opening by reconstruction whereas the remaining maxima are perfectly removed. The difference between both reconstructions is also clearly visible in the examples of Fig. 3. The first opening by reconstruction removes small bright details of the image: the text in the upper left corner. The fish is a large element and is not

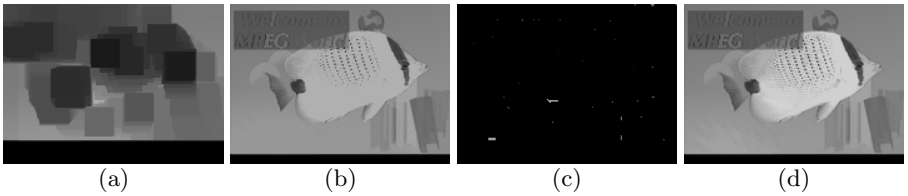


Fig. 3. Size filtering with opening by reconstruction: (a) erosion of the original image of Fig. 1(a) by a flat structuring element of size 10x10, (b) reconstruction of the erosion, (c) marker indicating where the first reconstruction has not been active (Eq. 7) and (d) second reconstruction

removed. It is indeed visible after the first opening by reconstruction (Fig. 3(b)) but its gray level values are not well preserved. This drawback is avoided by using the second reconstruction. Finally, let us mention that by duality closings by reconstruction can be defined. They have the same effect than the openings but on dark components.

Contrast Filtering. The previous section considered size simplification. A contrast simplification can be obtained by substituting the erosion in Eq. 5 by a subtraction of a constant, c, from the original image f:

$$\phi(f) = \rho^{\downarrow}(f - c|f) \tag{9}$$

This operator, known as λ-max operator, is connected, increasing and anti-extensive but not idempotent. Its effect is illustrated in Fig. 4(a). As can be seen, the maxima of small contrast are removed and the contours of the maxima of high contrast are well preserved. However, the height of the remaining maxima are not well preserved. As in the previous section, this drawback can be

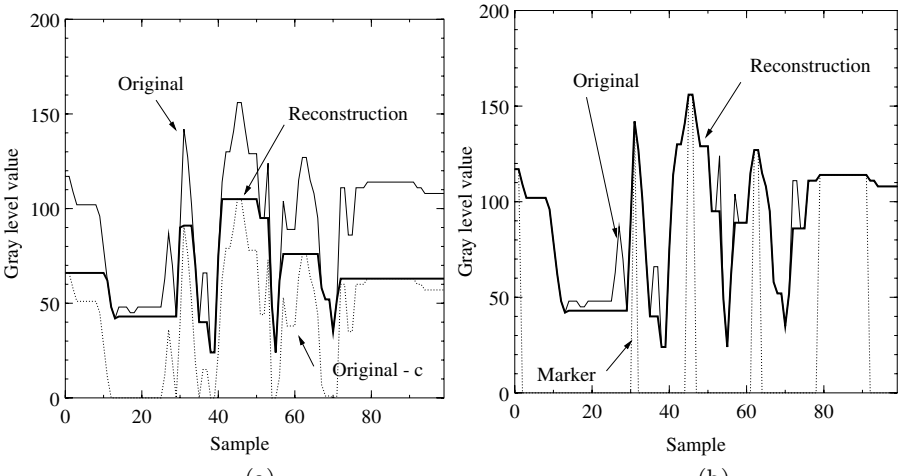


Fig. 4. Contrast-oriented connected operators: (a) Reconstruction of $f - c$, (b) Second reconstruction

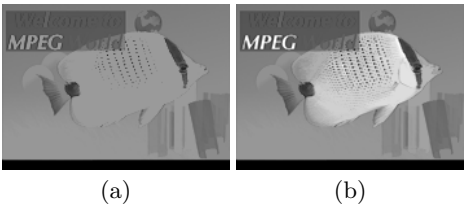


Fig. 5. Contrast filtering: (a) λ-max operator, (b) dynamic opening

removed if a second reconstruction process is used. This second reconstruction process is exactly the same as the previous one defined by Eq. 7 ($m[n] = f[n]$ if $\rho^{\downarrow}(f - c|f)[n] = f[n] - c$). This second connected operator is an opening. It is called a dynamic opening [27].

The operators effect is illustrated in Fig. 5. Both operators remove maxima of contrast c lower than 100 gray level values. However, the λ-max operator produces an output image of low contrast, even for the preserved maxima. By contrast, the dynamic opening successfully restores the retained maxima.

4.2 Self-dual Reconstruction and Levelings

The connected operators discussed in the previous section were either anti-extensive or extensive. They allow the simplification of either bright or dark image components. For some applications, this behavior is a drawback and one would like to simplify in a symmetrical way all components. From the theoretical viewpoint, this means that the filter has to be self-dual, that is $\psi(f) = -\psi(-f)$.

With the aim of constructing self-dual connected operators, the concept of levelings was proposed in [6] by adding some restrictions in Definition 2:

Definition 5. *(Leveling) The operator ψ is a leveling if $\forall n, n'$ neighboring pixels, $\psi(f)[n] > \psi(f)[n'] \Longrightarrow f[n] \geq \psi(f)[n]$ and $\psi(f)[n'] \geq f[n']$.*

This definition not only states that if a transition exists in the output image, it was already present in the original image (Definition 2) but also that 1) the sense of gray level variation between n and n' has to be preserved and 2) the variation $\|\psi(f)[n] - \psi(f)[n']\|$ is bounded by the original variation $\|f[n] - f[n']\|$.

The theoretical properties of levelings are studied in [6, 7], in particular:

- Any opening or closing by reconstruction is a leveling.
- If ψ_1, ψ_2 are levelings, $\psi_2\psi_1$ is also a leveling.
- If $\{\psi_i\}$ are levelings, their supremum $\bigvee_i \psi_i$, and infimum $\bigwedge_i \psi_i$, are levelings.

The most popular technique to create levelings relies on the following self-dual reconstruction process:

Definition 6. *(Self-dual reconstruction) If f and g are two images (respectively called the "reference" and the "marker"image), the self-dual reconstruction $\rho^{\updownarrow}(g|f)$ of g with respect to f is given by:*

$$\begin{array}{ll} g_k & = \epsilon_C(g_{k-1}) \bigvee [\delta_C(g_{k-1}) \bigwedge f] \\ & = \delta_C(g_{k-1}) \bigwedge [\epsilon_C(g_{k-1}) \bigvee f] \text{ (equivalent expression) and} \\ \rho^{\updownarrow}(g|f) & = \lim_{k \to \infty} g_k \end{array} \tag{10}$$

where $g_0 = g$ and δ_C and ϵ_C are respectively the dilation and the erosion with the flat structuring element defining the connectivity (3x3 square or cross).

An example of self-dual reconstruction is shown in Fig. 6. In this example, the marker image is constant everywhere except for two points that mark a maximum and a minimum of the reference image. After reconstruction, the output has only

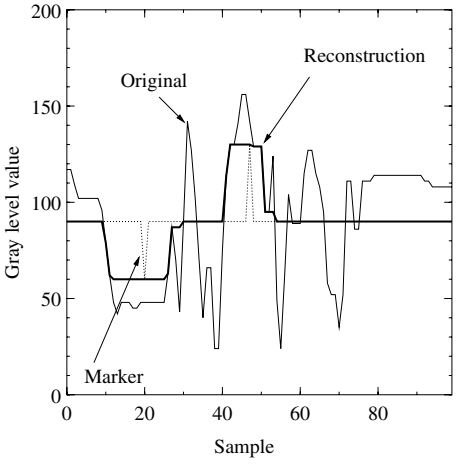


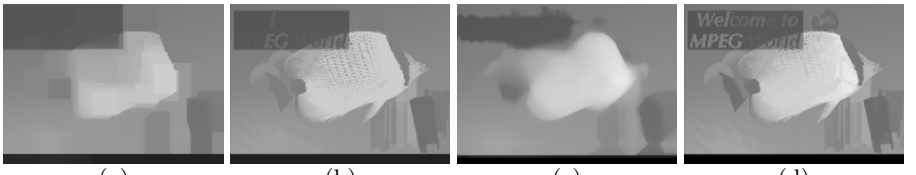
Fig. 6. Example of leveling with self-dual reconstruction

one maximum and one minimum. As can be seen, the self-dual reconstruction is the anti-extensive reconstruction of Eq. 3 for the pixels where $g[n] < f[n]$ and the extensive reconstruction of Eq. 4 for the pixels where $f[n] < g[n]$.

As in the case of anti-extensive reconstruction, Eq. 10 does not define an efficient implementation of the reconstruction process. In fact, an efficient implementation of the self-dual reconstruction can be obtained by combination of the strategies used for anti-extensive and extensive reconstruction processes: the *initialization* step consists in putting in the FIFO queue: 1) the boundary pixels of marker maxima when the marker is smaller than the reference and 2) the boundary pixels of marker minima when the marker is greater than the reference.

The *propagation* step is done in a similar fashion than the one described for anti-extensive reconstruction: the anti-extensive propagation is used when the marker is below the reference and the extensive propagation is used when the marker is above the reference.

In practice, the self-dual reconstruction is used to restore the contour information after a simplification performed by an operator that is neither extensive nor anti-extensive. A typical example is an alternating sequential filter: $g = \varphi_{h_k}\gamma_{h_k}\varphi_{h_{k-1}}\gamma_{h_{k-1}}\cdots\varphi_{h_1}\gamma_{h_1}(f)$, where φ_{h_k} and γ_{h_k} are respectively a closing and an opening with a structuring element h_k. This example is illustrated in Fig.s 7(a) and 7(b). Note the simplification effect which deals with both maxima and minima, and how the contour distortion introduced by the alternating sequential filter is removed by the reconstruction. However, from a theoretical viewpoint, the operator: $\rho^{\updownarrow}(\varphi_{h_k}\gamma_{h_k}\ldots\varphi_{h_1}\gamma_{h_1}(f)|f)$ is not self-dual because the alternating sequential filter itself is not self-dual. In order to create a self-dual operator, the creation of the marker has also to be self-dual. Fig.s 7(c) and 7(d) show an example where the marker is created by a median filter (that is self-dual). This kind of results can be extended to any linear filter and the self-dual reconstruction can be considered as a general tool that restores the contour in-



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

Fig. 7. Size filtering with leveling: (a) Alternating sequential filter of the original image of Fig. 1(a), (b) Self-dual reconstruction of the alternating sequential filter, (c) Median filter and (d) Self-dual reconstruction of the median filter

formation after a filtering process. In other words, the reconstruction allows to create a connected version $\rho^{\updownarrow}(\psi(f)|f)$ of any filter: $\psi(f)$.

5 Connected Operators Based on Region-Tree Pruning

5.1 Tree Representations and Connected Operators

The reconstruction strategies discussed in the previous section can be viewed as tools that work on a pixel-based representation of the image and that provide a way to create connected operators. In this section, we present a different approach: the first step of the filtering process is to construct a region-based representation of the image, then the simplification effect is obtained by direct manipulation of the tree. The approach may be considered as being conceptually more complex than the reconstruction however, it provides more flexibility in the choice of the simplification criterion.

Two region-based representations are discussed in the sequel: the Max-tree / Min-tree [14] and the Binary Partition Tree [15]. The first one leads to anti-extensive connected operators whereas the second one is a basis for self-dual connected operators. Let us discuss first these two region-based representations.

Max-tree and Min-tree. The first representation is called a Max-tree [14]. It enhances the maxima of the signal. Each tree node $\mathcal{N}_k$ represents a connected component of the space that is extracted by the following thresholding process: for a given threshold T, consider the set of pixels X that have a gray level value larger than T and the set of pixels Y that have a gray level value equal to T:

$$\begin{aligned} X &= \{n \text{ , such that } f[n] \geq T\} \\ Y &= \{n \text{ , such that } f[n] = T\} \end{aligned} \tag{11}$$

The tree nodes $\mathcal{N}_k$ represent the connected components of X such that $X \bigcap Y \neq \emptyset$. A simple example of Max-tree is shown in Fig. 8. The original image is made of 7 flat zones identified by a letter {A,...,G}. The number following each letter defines the gray level value of the flat zones. The binary images, X, resulting from the thresholding with $0 \leq T \leq 2$ are shown in the center of the figure. Finally, the Max-tree is given in the right side. It is composed of 5 nodes that represent the connected components shown in black. The number inside each square

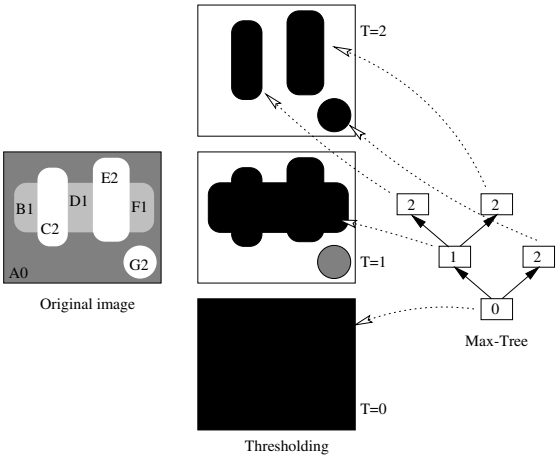


Fig. 8. Max-tree representation of images

represents the threshold value where the component was extracted. Finally, the links in the tree represent the inclusion relationships among the connected components following the threshold values. Note that when the threshold is set to $T = 1$, the circular component does not create a connected component that is represented in the tree because none of its pixels has a gray level value equal to 1. However, the circle itself is obtained when $T = 2$. The three regional maxima are represented by three leaves and the tree root represents the entire support of the image. The computation of Max-tree can be done in an efficient way (see [14] for more details).

Binary Partition Tree (BPT). The second example of region-based representation of images is the BPT [15]. It represents a set of regions obtained from an initial partition that we assume to be the partition of flat zones. The leaves of the tree represent the flat zones of the original signal. The remaining tree nodes represent regions that are obtained by merging the regions represented by the children. The root node represents the entire image support. The tree represents a fairly large set of regions at different scales. Large regions appear close to the root whereas small details can be found at lower levels. This representation should be considered as a compromise between representation accuracy and processing efficiency. Indeed, all possible merging of regions belonging to the initial partition are not represented in the tree. Only the most "likely" or "useful" merging steps are represented in the BPT. The connectivity encoded in the tree structure is binary in the sense that a region is explicitly connected to its sibling (since their union is a connected component represented by the father), but the remaining connections between regions of the original partition are not represented in the tree. Therefore, the tree encodes only part of the neighborhood relationships between the regions of the initial partition. However, as will be seen in the sequel, the main advantage of the tree representation is that it allows the fast implementation of sophisticated processing techniques.

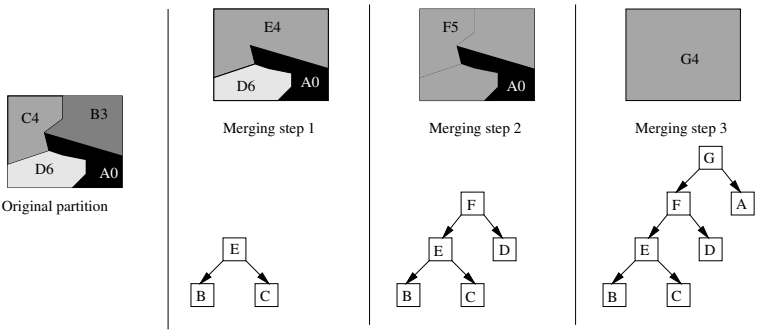


Fig. 9. Example of BPT creation with a region merging algorithm

The BPT should be created in such a way that the most "interesting" or "useful" regions are represented. This issue can be application dependent. However, a possible solution, suitable for a large number of cases, is to create the tree by keeping track of the merging steps performed by a segmentation algorithm based on region merging (see [28, 29] for example). In the following, this information is called the *merging sequence*. Starting from the partition of flat zones, the algorithm merges neighboring regions following a homogeneity criterion until a single region is obtained. An example is shown in Fig. 9. The original partition involves four regions. The regions are indicated by a letter and the number indicates the grey level value of the flat zone. The algorithm merges the four regions in three steps. In the first step, the pair of most similar regions, B and C, are merged to create region E. Then, region E is merged with region D to create region F. Finally, region F is merged with region A and this creates region G corresponding to the region of support of the whole image. In this example, the merging sequence is: $(B, C)|(E, D)|(F, A)$. This merging sequence defines the BPT as shown in Fig. 9.

To completely define the merging algorithm, one has to specify the region merging order and the region model, that is the model used to represent the union of two regions. In order to create the BPTs used to illustrate the processing examples discussed in this paper, we have used a merging algorithm following the color homogeneity criterion described in [29]. Let us define the merging order $O(R_1, R_2)$ and the region model M_R:

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

$$O(R_1, R_2) = N_1||M_{R_1} - M_{R_1 \cup R_2}||_2 + N_2||M_{R_2} - M_{R_1 \cup R_2}||_2 \qquad (12)$$

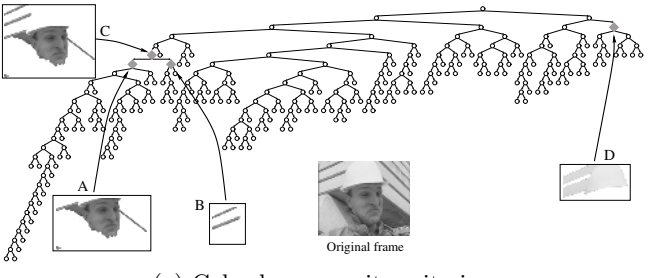
where N_1 and N_2 are the numbers of pixels of regions R_1 and R_2 and $||.||_2$ denotes the $\mathcal{L}_2$ norm. M_R represents the model for region R. It consists of three constant values describing the YUV components. The interest of this merging order, compared to other classical criteria, is discussed in [29].

– **Region model:** as mentioned previously, each region is modeled by a constant vector YUV value: M_R. During the merging process, the YUV components of the union of 2 regions, R_1 and R_2, are computed as follows [29]:

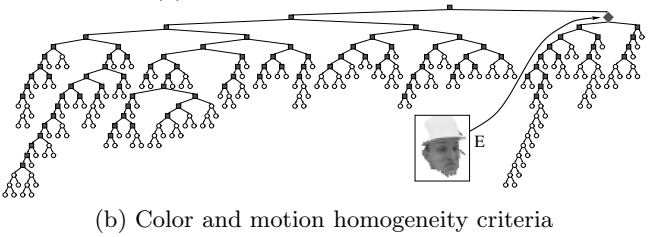
$$\begin{aligned} &\text{if } N_1 < N_2 \Rightarrow M_{R_1 \cup R_2} = M_{R_2} \\ &\text{if } N_2 < N_1 \Rightarrow M_{R_1 \cup R_2} = M_{R_1} \\ &\text{if } N_1 = N_2 \Rightarrow M_{R_1 \cup R_2} = (M_{R_1} + M_{R_2})/2 \end{aligned} \tag{13}$$

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

It should be noticed that the homogeneity criterion has not to be restricted to color. For example, if the image for which we create the BPT belongs to a sequence of images, motion information can also be used: in a first stage, regions are merged using a color homogeneity criterion, whereas a motion homogeneity criterion is used in the second stage. Fig. 10 shows an example of the *Foreman* sequence. In Fig. 10(a), the BPT has been constructed exclusively with the color criterion described above. In this case, it is not possible to concentrate the information about the foreground object (head and shoulder regions of Foreman) within a single sub-tree. For example, the face mainly appears in the sub-tree hanging from region A, whereas the helmet regions are located below region D. In practice, the nodes close to the root have no clear meaning because they are not homogeneous in color. Fig. 10(b) presents an example of BPT created with color and motion criteria. The nodes appearing as white circles correspond to the color criterion, whereas the dark squares correspond to a motion criterion. The motion criterion is formally the same as the color criterion except that the



(a) Color homogeneity criterion



(b) Color and motion homogeneity criteria

Fig. 10. Examples of creation of BPT

YUV color distance is replaced by the YUV Displaced Frame Difference. The process starts with the color criterion as in Fig. 10(a) and then, when a given peak signal to noise ratio (PSNR) is reached, it changes to the motion criterion. Using motion information, the face and helmet now appear as a single region E.

As can be seen, the construction of a BPT is fairly more complex than the creation of a Max-tree or a Min-tree. However, BPTs offer more flexibility because one can chose the homogeneity criterion through the proper selection of the region model and the merging order. Furthermore, if the functions defining the region model and the merging order are self-dual, the tree itself is self-dual. The same BPT can be used to represent f and $-f$. The BPT representation is appropriate to derive self-dual connected operators whereas the Max-tree (Min-tree) is adequate for anti-extensive (extensive) connected operators. Note that in all cases, trees are hierarchical region-based representations. They encode a large set of regions and partitions that can be derived for the flat zones partition of the original image without adding new contours.

Filtering Strategy. Once the representation has been created, the filtering strategy consists in *pruning* the tree and in reconstructing an image from the pruned tree. The global processing strategy is illustrated in Fig. 11. The simplification effect of the filter 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-trees (Min-trees), the components that may be eliminated are regional maxima (minima) whereas the elements that may be simplified in the case of BPTs are unions of the most similar flat zones. The simplification itself is governed by a criterion which may involve simple notions such as size, contrast or more complex ones such as texture, motion or even semantic criteria.

One of the interests of the tree representations is that the set of possible merging steps is fixed (represented by the tree branches). As a result, sophisticated pruning strategies may be designed. An example of such strategy deals with non-increasing simplification criteria. Mathematically, a criterion $\mathcal{C}$ assessed on a region R is said to be increasing if the following property holds:

$$\forall R_1 \subseteq R_2 \Rightarrow \mathcal{C}(R_1) \leq \mathcal{C}(R_2) \tag{14}$$

Assume that all nodes corresponding to regions where the criterion value is lower than a given threshold should be pruned. If the criterion is increasing, the

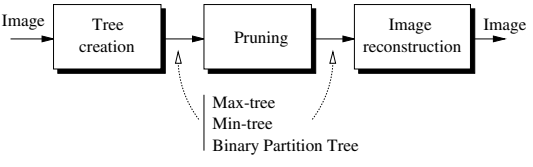


Fig. 11. Connected operators based on Tree representations

pruning strategy is straightforward: merge all nodes that should be removed. It is indeed a pruning strategy since the increasingness of the criterion guarantees that, if a node has to be removed, all its descendants have also to be removed. An example of BPT with increasing decision criterion is shown in Fig. 12. The criterion used to create this example is the size, measured as the number of pixels belonging to the region, which is indeed increasing. Note that this example involves a BPT but the same issue also applies to Max/Min-tree representations.

If the criterion is not increasing, the pruning strategy is not straightforward since the descendants of a node to be removed have not necessarily to be removed. An example of such criterion is the region perimeter. Fig. 13 illustrates this case. If we follow either Path A or Path B in Fig. 13, we see that there are some oscillations of the remove/preserve decisions. In practice, the non-increasingness of the criterion implies a lack of robustness of the operator. For example, similar images may produce quite different results or small modifications of the criterion threshold involve drastic changes on the output.

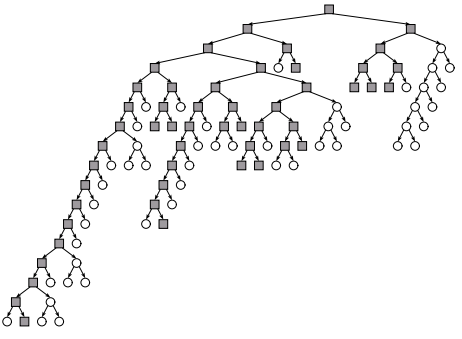


Fig. 12. Example of increasing criterion (size). If a node has to be removed, all its descendants have also to be removed. Gray squares: nodes to be preserved, white circles: nodes to be removed.

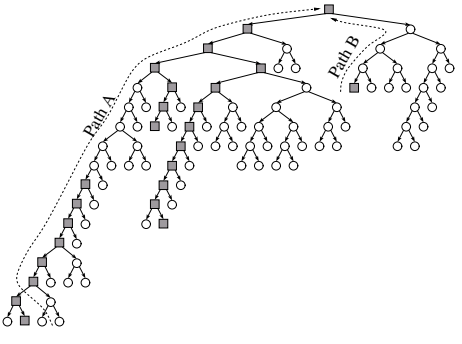


Fig. 13. Example of non-increasing criterion (perimeter). No relation exists between the decisions among descendants (see decisions along path A or path B). Gray squares: nodes to be preserved, white circles: nodes to be removed.

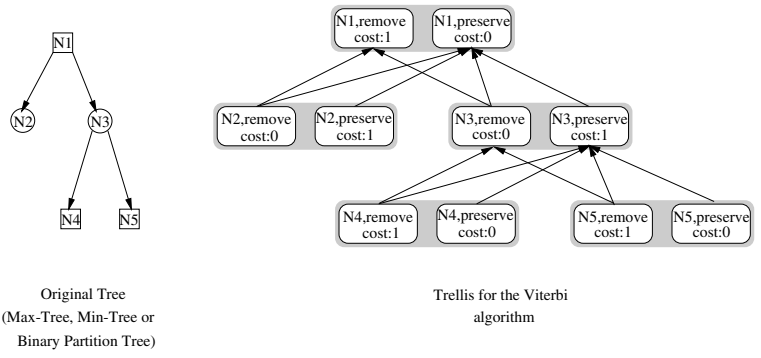


Fig. 14. Trellis structure for the Viterbi algorithm. A circular (square) node on the Tree indicates that the criterion value states that the node has to be removed (preserved). The trellis on which the Viterbi algorithm is run duplicates the structure of the Tree and defines a *preserve* state and a *remove* state for each tree node. Paths from *remove* states to child *preserve* states are forbidden so that the decisions are increasing.

A possible solution to the non-increasingness of the criterion consists in applying a transformation on the set of decisions. The transformation should create a set of increasing decisions while preserving as much as possible the decisions defined by the criterion. This problem may be viewed as dynamic programming issue that can be efficiently solved with the Viterbi algorithm.

The dynamic programming algorithm is explained and illustrated in the sequel assuming that the tree is binary. The extension to N-ary trees is straightforward and the example of binary tree is used here only to simplify the notation. An example of trellis on which the Viterbi algorithm [30] is applied is illustrated in Fig. 14. The trellis has the same structure as the tree except that two trellis states, *preserve* $\mathcal{N}_k^P$ and *remove* $\mathcal{N}_k^R$, correspond to each node $\mathcal{N}_k$ of the tree. The two states of each child node are connected to the two states of its parent. However, to avoid non-increasing decisions, the *preserve* state of a child is not connected to the *remove* state of its parent. As a result, the trellis structure guarantees that, if a node has to be removed, its children have also to be removed. The cost associated to each state is used to compute the number of modifications the algorithm has to do to create an increasing set of decisions. If the criterion value states that the node of the tree has to be removed, the cost associated to the *remove* state is equal to zero (no modification) and the cost associated to the *preserve* state is equal to one (one modification). Similarly, if the criterion value states that the node has to be preserved, the cost of the *remove* state is equal to one and the cost of the *preserve* state is equal to zero[3]. The cost values appearing in Fig. 14 assume

[3] Although some modifications may be much more severe than others, the cost choice has no strong effect on the final result. This issue of cost selection is similar to the hard versus soft decision of the Viterbi algorithm in the context of digital communications [30].

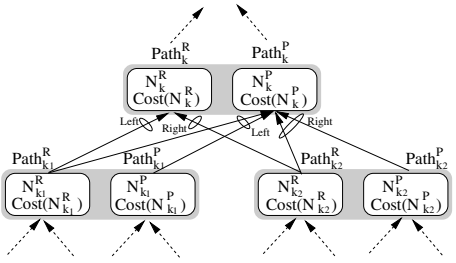


Fig. 15. Definition of *Path* and cost for the Viterbi algorithm (see Eqs. 17, 18 and 19)

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 decisions such that:

$$\text{Min} \sum_k \text{Cost}(\mathcal{N}_k) \text{ such that the decisions are increasing.} \tag{15}$$

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

– Let us consider a node $\mathcal{N}_k$ and its *preserve* state $\mathcal{N}_k^P$. A path $Path_k$ is a continuous set of transitions between nodes $(\mathcal{N}_\alpha \rightarrow \mathcal{N}_\beta)$ defined in the trellis:

$$Path_k = (\mathcal{N}_\alpha \rightarrow \mathcal{N}_\beta) \cup (\mathcal{N}_\beta \rightarrow \mathcal{N}_\gamma) \cup ... \cup (\mathcal{N}_\psi \rightarrow \mathcal{N}_k) \tag{16}$$

The path $Path_k^P$ starting from a leaf node and ending at that state is composed of two sub-paths[4]: the first one, $Path_k^{P,Left}$, comes from the left child and the second one, $Path_k^{P,Right}$, from the right child (see Fig. 15). In both cases, the path can emerge either from the *preserve* or from the *remove* state of the child nodes. If $\mathcal{N}_{k_1}$ and $\mathcal{N}_{k_2}$ are respectively the left and the right child nodes of $\mathcal{N}_k$, we have:

$$\begin{aligned} Path_k^{P,Left} &= Path_{k_1}^R \bigcup (\mathcal{N}_{k_1}^R \rightarrow \mathcal{N}_k^P) \quad \text{or } Path_{k_1}^P \bigcup (\mathcal{N}_{k_1}^P \rightarrow \mathcal{N}_k^P) \\ Path_k^{P,Right} &= Path_{k_2}^R \bigcup (\mathcal{N}_{k_2}^R \rightarrow \mathcal{N}_k^P) \quad \text{or } Path_{k_2}^P \bigcup (\mathcal{N}_{k_2}^P \rightarrow \mathcal{N}_k^P) \\ Path_k^P &= Path_k^{P,Left} \bigcup Path_k^{P,Right} \end{aligned} \tag{17}$$

[4] In the general case of an N-ary tree, the number of incoming paths may be arbitrary.

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

$$\begin{aligned}
&\text{If } Cost(Path_{k_1}^R) < Cost(Path_{k_1}^P) \\
&\quad \text{then } \{ \quad Path_k^{P,Left} = Path_{k_1}^R \bigcup (\mathcal{N}_{k_1}^R \rightarrow \mathcal{N}_k^P); \\
&\qquad\qquad Cost(Path_k^{P,Left}) = Cost(Path_{k_1}^R); \} \\
&\quad \text{else } \{ \quad Path_k^{P,Left} = Path_{k_1}^P \bigcup (\mathcal{N}_{k_1}^P \rightarrow \mathcal{N}_k^P); \\
&\qquad\qquad Cost(Path_k^{P,Left}) = Cost(Path_{k_1}^P); \} \\
&\text{If } Cost(Path_{k_2}^R) < Cost(Path_{k_2}^P) \\
&\quad \text{then } \{ \quad Path_k^{P,Right} = Path_{k_2}^R \bigcup (\mathcal{N}_{k_2}^R \rightarrow \mathcal{N}_k^P); \\
&\qquad\qquad Cost(Path_k^{P,Right}) = Cost(Path_{k_2}^R); \} \\
&\quad \text{else } \{ \quad Path_k^{P,Right} = Path_{k_2}^P \bigcup (\mathcal{N}_{k_2}^P \rightarrow \mathcal{N}_k^P); \\
&\qquad\qquad Cost(Path_k^{P,Right}) = Cost(Path_{k_2}^P); \} \\
&Cost(Path_k^P) = Cost(Path_k^{P,Left}) + Cost(Path_k^{P,Right}) + Cost(\mathcal{N}_k^P);
\end{aligned} \tag{18}$$

– In the case of the *remove* state, $\mathcal{N}_k^R$, the two sub-paths can only come from the *remove* states of the children. So, no selection has to be done. The path and its cost are constructed as follows:

$$\begin{aligned}
Path_k^{R,Left} &= Path_{k_1}^R \bigcup (\mathcal{N}_{k_1}^R \rightarrow \mathcal{N}_k^R); \\
Path_k^{R,Right} &= Path_{k_2}^R \bigcup (\mathcal{N}_{k_2}^R \rightarrow \mathcal{N}_k^R); \\
Path_k^R &= Path_k^{R,Left} \bigcup Path_k^{R,Right}; \\
Cost(Path_k^R) &= Cost(Path_{k_1}^R) + Cost(Path_{k_2}^R) + Cost(\mathcal{N}_k^R);
\end{aligned} \tag{19}$$

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

A complete optimization example is shown in Fig. 16. The original tree involves 5 nodes. The *preserve* decisions are shown by a square whereas the *remove* decisions are indicated by a circle. The original tree does not correspond to a set of increasing decisions because $\mathcal{N}_3$ should be removed but $\mathcal{N}_4$ and $\mathcal{N}_5$ should be preserved. The algorithm is initialized by creating the trellis and populating its states by their respective cost (see Fig. 14). Then, the first step of the algorithm consists in selecting the paths that go from states $\mathcal{N}_4^R$, $\mathcal{N}_4^P$, $\mathcal{N}_5^R$, $\mathcal{N}_5^P$ to states $\mathcal{N}_3^R$, $\mathcal{N}_3^P$. The corresponding trellis is shown in the upper part of Fig. 16 together with the corresponding costs of the four surviving paths. The second step iterates the procedure between states $\mathcal{N}_2^R$, $\mathcal{N}_2^P$, $\mathcal{N}_3^R$, $\mathcal{N}_3^P$ and states $\mathcal{N}_1^R$, $\mathcal{N}_1^P$. Here again, only four paths survive. They are indicated in the central diagram of Fig. 16. Finally, the last step consists in selecting the path of lowest cost that terminates at the root states. In the example of Fig. 16, the path ending at the

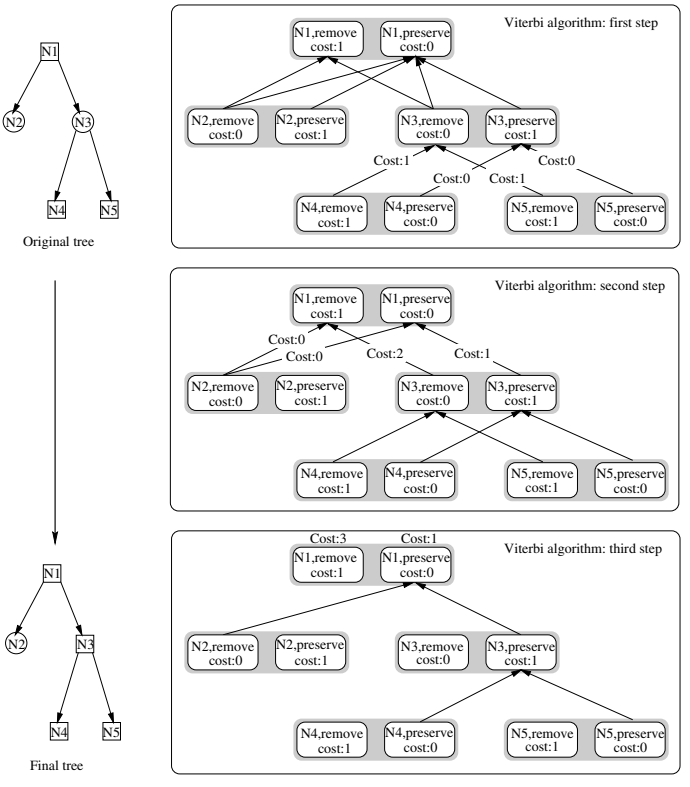


Fig. 16. Definition of the optimum decisions by the Viterbi algorithm

remove state of the root node ($\mathcal{N}_1^R$) has a cost of 3, whereas the path ending at the *preserve* state ($\mathcal{N}_1^P$) has a cost of 1. This last path is taken since it corresponds to an increasing set of decisions and involves just one modification of the original decisions. To find the optimum increasing decisions, one has to track back the selected path from the root to all leaves. In our example, we see that the paths hit the following states: $\mathcal{N}_1^P$, $\mathcal{N}_2^R$, $\mathcal{N}_3^P$, $\mathcal{N}_4^P$ and $\mathcal{N}_5^P$. The diagram at the bottom of Fig. 16 shows the final path together with the modified tree. As can be seen, the only modification has been to change the decision of node $\mathcal{N}_3$ and the resulting set of decisions is increasing. A complete example of is shown in Fig. 17. The original tree corresponds to the one shown in Fig. 13. The Viterbi algorithm has to modify 5 decisions along path A and one decision along path B (see Fig. 13) to get the optimum set of increasing decisions.

To summarize, let us say that any pruning strategy can be applied directly on the tree if the decision criterion is increasing. In the case of a non-increasing criterion, the Viterbi algorithm can be used to modify the smallest number of decisions so that increasingness is obtained. These modifications define a pruning strategy. Once the pruning has been performed, it defines an output partition and each region is filled with a constant value. In the case of a Max-tree (Min-

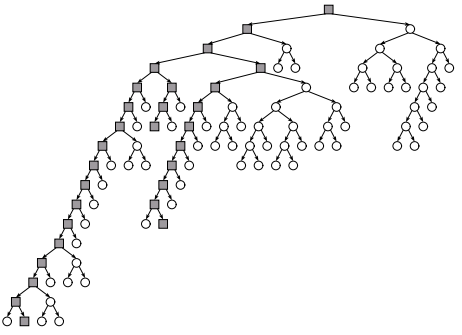
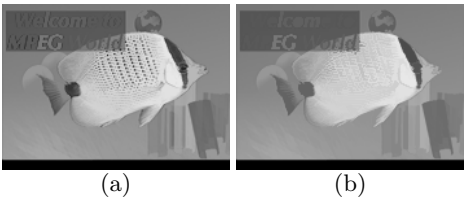


Fig. 17. Set of increasing decisions resulting from the Viterbi algorithm used on the original tree of Fig. 13. Five decisions along path A and one decision along path B have been modified. Gray squares: preserve, white circles: remove.

tree), the constant value is equal to the minimum (maximum) gray level value of the original pixels belonging to the region. As a result, the operator is anti-extensive (extensive). In the case of a BPT, the goal is to define a self-dual operator. Therefore, each region of the output partition has to be filled by a self-dual model, such as the mean or the median of the original pixels belonging to the region.

5.2 Example of Connected Operators Based on Tree Representations

Increasing Criterion ⇒ Direct Pruning. The first example deals with situations where the criterion is increasing. In this case, the comparison of the criterion value with a threshold directly defines a pruning strategy. A typical example is the area opening [12]. One possible implementation of the area opening consists in creating a Max-tree and in measuring the area (the number of pixels) $\mathcal{A}_k$ contained in each node $\mathcal{N}_k$. If the area $\mathcal{A}_k$ is smaller than a threshold, $\mathcal{T}_\mathcal{A}$, the node is removed. The area criterion is increasing and the Viterbi algorithm does not have to be used. It can be shown that the area opening is equal to the supremum of all possible openings by a connected structuring element involving



(a) (b)

Fig. 18. Area filtering: (a) area opening, γ^{area}, (b) area opening followed by area closing, $\varphi^{area}\gamma^{area}$

$\mathcal{T}_{\mathcal{A}}$ pixels. The simplification effect of the area opening is illustrated in Fig. 18(a). As expected, the operator removes small bright components of the image. If this simplified image is processed by the dual operator, the area closing, small dark components are also removed (see Fig. 18(b)).

Using the same strategy, a large number of connected operators can be obtained. For example, if the criterion is the volume: $\sum_{n\in R} f[n]$ (also increasing), the resulting operator is the volumic opening [31]. The reader is referred to [15] to see examples of this situation involving a BPT.

Non-increasing Criterion ⇒ Modification of the Decision (Viterbi algorithm) and Pruning. This situation is illustrated here by a motion-oriented connected operator [14]. Denote by $f_t[n]$ an image sequence where n represents the pixel coordinates and t the time instant. The goal of the connected operator is 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 $\Delta[n]$. The field can be constant Δ if one wants to extract all objects following a translation, but in general the displacement depends on the spatial position n to deal with more complex motion models.

The sequence processing is performed as follows: each frame is transformed into its corresponding Max-tree representation and each node $\mathcal{N}_k$ is analyzed. To check whether or not the pixels contained in a given node $\mathcal{N}_k$ are moving in accordance to the motion field $\Delta[n]$, a simple solution consists in computing the Mean Displaced Frame Difference (DFD) of this region with the previous frame:

$$\mathrm{DFD}_{f_t}^{f_{t-1}}(\mathcal{N}_k) = \sum_{n\in\mathcal{N}_k} |f_t[n] - f_{t-1}[n-\Delta[n]]| / \sum_{n\in\mathcal{N}_k} 1 \tag{20}$$

In practice, however, it is not very reliable to assess the motion on the basis of only two frames. The criterion should include a reasonable memory of the past decisions. This idea can be easily introduced in the criterion by adding a recursive term. Two mean DFD 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:

$$Motion(\mathcal{N}_k) = \alpha \mathrm{DFD}_{f_t}^{f_{t-1}}(\mathcal{N}_k) + (1-\alpha)\mathrm{DFD}_{f_t}^{\psi(f_{t-1})}(\mathcal{N}_k) \tag{21}$$

with $0 \leq \alpha \leq 1$. If α is equal to 1, the criterion is memoryless. Low values of α allow the introduction of an important recursive component in the decision process. In a way similar to 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 ($0 \leq \alpha \ll 1$).

The motion criterion described by Eqs. 20 and 21 deals with one set of motion parameters. Objects that do not follow the given motion produce a high DFD and should be removed. The criterion is not increasing and the Viterbi algorithm

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

Fig. 19. Example of motion connected operator preserving fixed objects: (a) Original frame, (b) Motion connected operator ψ, (c) Dual operator: $\psi^*\psi(f)$ and (d) residue: f - $\psi^*(\psi(f)$

has to be used. This motion-oriented pruning strategy can be used on Max-tree, Min-tree or BPT representations.

A motion filtering example relying on a Max-tree is shown in Fig. 19. The operator goal is to remove all moving objects. The motion model is defined by: $\Delta[n] = (0,0), \forall n$. 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 connected operator $\psi(f)$ removes all bright moving objects (Fig. 19(b)). The dual operator: $\psi^*(f) = -\psi(-f)$ removes all dark moving objects (Fig. 19(c)). The residue (the difference with the original image) presented in Fig. 19(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 motion connected operator can potentially be used for a large set of applications. It permits 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 flat zones. 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 [14, 15].

5.3 Pruning Strategies Involving Global Optimization Under Constraint

In this section, we illustrate a more complex pruning strategy involving global optimization under constraint. To fix the notations, let us denote by $\mathcal{C}$ the criterion that has to be optimized (we assume, without loss of generality, 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 the resulting partition is composed of nodes $\mathcal{N}_i$ such that:

$$\text{Min} \sum_{\mathcal{N}_i} \mathcal{C}(\mathcal{N}_i) \text{ , with } \sum_{\mathcal{N}_i} \mathcal{K}(\mathcal{N}_i) \leq \mathcal{T}_\mathcal{K} \tag{22}$$

It has been shown [32] that 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:

$$\text{Min} \left(\sum_{\mathcal{N}_i} \mathcal{C}(\mathcal{N}_i) + \lambda^* \sum_{\mathcal{N}_i} \mathcal{K}(\mathcal{N}_i) \right) \tag{23}$$

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:

$$\begin{aligned} \text{If } \mathcal{C}(\mathcal{N}_0) + \lambda^*\mathcal{K}(\mathcal{N}_0) < \sum_{\mathcal{N}_i} \mathcal{C}(\mathcal{N}_i) \\ + \lambda^* \sum_{\mathcal{N}_i} \mathcal{K}(\mathcal{N}_i), \text{ prune the children nodes } \mathcal{N}_i. \end{aligned} \tag{24}$$

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 that searches for the best λ parameter. The computation of the optimum λ parameter 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 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. Finally, the theoretical properties depend mainly on the criterion and on the constraint. In any case, the operator is connected and self-dual.

This type of pruning strategy is illustrated by two examples relying on a BPT. 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

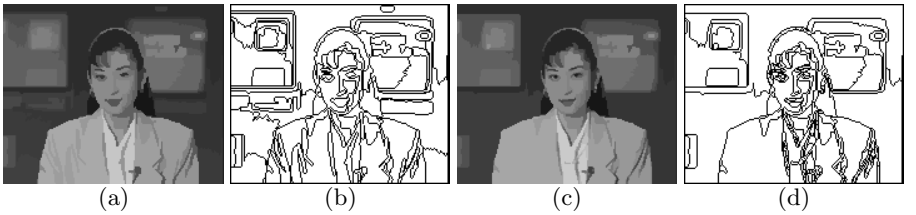


Fig. 20. Example of optimization strategies under a squared error constraint of 31 dB. (a) Minimization of the number of the flat zones, (b) contours of the flat zones of Fig. 20(a) (number of flat zones: 87, perimeter length: 4491), (c) Minimization of the total perimeter length, (d) contours of the flat zones of Fig. 20(b) (number of flat zones: 219, perimeter length: 3684).

second example, the criterion is to minimize the total length of the flat zones contours: $\mathcal{C}_2 = \sum_{\mathcal{N}_k} Perimeter(\mathcal{N}_k)$. 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. 20, the squared error is constrained to be of at least 31 dB. Fig. 20(a) 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. 20(c). The approximation provided by this image is of the same quality as the previous one (squared error of 31 dB). 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. 20(a) 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 20(b) and 20(d) show the flat zone contours. As can be seen, 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 such as a Max-tree or a BPT, it would be extremely difficult to implement this kind of connected operators.

6 Conclusions

This paper has presented and discussed a region-based processing technique involving connected operators. There is currently an interest in defining processing tools that do not act on the pixel level but on a region level. Connected operators are examples of such tools that come from mathematical morphology.

Connected operators are operators that process the image by merging flat zones. As a result, they cannot introduce any contours or move existing ones. The two most popular approaches to create connected operators have been reviewed. The first one work on a pixel-based representation of the image and involves a reconstruction process. The operator involves first a simplification step based on a "classical" operator (such as morphological open, close, low-pass filter, median filter, etc) and then a reconstruction process. Three kind of reconstruction processes have been analyzed: anti-extensive, extensive and self-dual. The goal of the reconstruction process is to restore the contour information after the simplification. In fact, the reconstruction can be seen as a way to create a connected version of an arbitrary operator. Note that the simplification effect is defined and limited by the first step. The examples we have shown include simplification in terms of size or contrast.

The second second strategy to create connected operators involves three steps: in the first step, a region-based representation of the input image is constructed.

Three examples have been discussed: Max-tree, Min-tree and Binary Partition Tree. In the second step, the simplification is obtained by pruning the tree and, in the third step, the 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 in a robust way with nonincreasing criteria. Criteria involving the notions of area, motion and optimization under a quality constraint have been demonstrated.

References

1. S. L. Horowitz and T. Pavlidis. Picture segmentation by a directed split-and-merge procedure. In *Second Int. joint Conference on Pattern Recognition*, pages 424–433, 1974.
2. J. Serra and P. Salembier. Connected operators and pyramids. In SPIE, editor, *Image Algebra and Mathematical Morphology*, volume 2030, pages 65–76, San Diego (CA), USA, July 1993.
3. J. Crespo, J. Serra, and R.W. Schafer. Theoretical aspects of morphological filters by reconstruction. *Signal Processing*, 47(2):201–225, 1995.
4. H. Heijmans. Connected morphological operators and filters for binary images. In *IEEE Int. Conference on Image Processing, ICIP'97*, volume 2, pages 211–214, Santa Barbara (CA), USA, October 1997.
5. G. Matheron. Les nivellements. Technical report, Paris school of Mines, Center for Mathematical Morphology, 1997.
6. F. Meyer. From connected operators to levelings. In *Fourth Int. Symposium on Mathematical Morphology, ISMM'98*, pages 191–198, Amsterdam, The Netherlands, June 1998. Kluwer.
7. F. Meyer. The levelings. In *Fourth Int. Symposium on Mathematical Morphology, ISMM'98*, pages 199–206, Amsterdam, The Netherlands, June 1998. Kluwer.
8. J. Serra. Connectivity on complete lattices. *Journal of Mathematical Imaging and Vision*, 9:231–251, 1998.
9. C. Ronse. Set-theoretical algebraic approaches to connectivity in continuous or digital spaces. *Journal of Mathematical Imaging and Vision*, 8:41–58, 1998.
10. P. Maragos. Differential morphology. In S. Mitra and G. Sicuranza, editors, *Nonlinear Image Processing*, chapter 13. Academic Press, 2000.
11. L. Vincent. Morphological gray scale reconstruction in image analysis: Applications and efficients algorithms. *IEEE, Transactions on Image Processing*, 2(2):176–201, April 1993.
12. L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In J. Serra and P. Salembier, editors, *First Workshop on Mathematical Morphology and its Applications to Signal Processing*, pages 22–27, Barcelona, Spain, May 1993. UPC.
13. E. Breen and R. Jones. An attribute-based approach to mathematical morphology. In P. Maragos, R.W. Schafer, and M.A. Butt, editors, *International Symposium on Mathematical Morphology*, pages 41–48, Atlanta (GA), USA, May 1996. Kluwer Academic Publishers.

14. P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7(4):555–570, April 1998.
15. P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation and information retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576, April, 2000.
16. C. Gomila and F. Meyer. Levelings in vector space. In *IEEE International Conference on Image Processing, ICIP'99*, Kobe, Japan, October 1999.
17. P. Salembier and M. Kunt. Size-sensitive multiresolution decomposition of images with rank order based filters. *EURASIP Signal Processing*, 27(2):205–241, May 1992.
18. P. Salembier and J. Serra. Flat zones filtering, connected operators and filters by reconstruction. *IEEE Transactions on Image Processing*, 3(8):1153–1160, August 1995.
19. J. Crespo, R.W. Shafer, J. Serra, C. Gratin, and F. Meyer. A flat zone approach: a general low-level region merging segmentation method. *Signal Processing*, 62(1):37–60, October 1997.
20. V. Vilaplana and F. Marqués. Face segmentation using connected operators. In *Fourth Int. Symposium on Mathematical Morphology, ISMM'98*, pages 207–214, Amsterdam, The Netherlands, June 1998. Kluwer.
21. J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.
22. J. Serra. *Image Analysis and Mathematical Morphology, Vol II: Theoretical advances*. Academic Press, 1988.
23. J. Crespo. Space connectivity and translation-invariance. In P. Maragos, R.W. Schafer, and M.A. Butt, editors, *International Symposium on Mathematical Morphology*, pages 118–126, Atlanta (GA), USA, May 1996. Kluwer Academic Publishers.
24. H. Heijmans. Connected morphological operators for binary images. Technical Report PNA-R9708, CWI, Amsterdam, The Netherlands, 1997.
25. J.C. Klein. *Conception et réalisation d'une unité logique pour l'analyse quantitative d'images*. PhD thesis, Nancy University, France, 1976.
26. L. Vincent. Graphs and mathematical morphology. *EURASIP, Signal Processing*, 16(4):365–388, April 1989.
27. M. Grimaud. A new measure of contrast: the dynamics. In Serra Gader, Dougherty, editor, *SPIE Visual Communications and Image Processing'92*, volume SPIE 1769, pages 292–305, San Diego (CA), USA, July 1992.
28. O. Morris, M. Lee, and A. Constantinidies. Graph theory for image analysis: an approach based on the shortest spanning tree. *IEE Proceedings, F*, 133(2):146–152, April 1986.
29. L. Garrido, P. Salembier, and D. Garcia. Extensive operators in partition lattices for image sequence analysis. *EURASIP Signal Processing*, 66(2):157–180, April 1998.
30. A.J. Viterbi and J.K. Omura. *Principles of Digital Communications and Coding*. Mc Graw-Hill, New York, 1979.
31. F. Meyer, A. Oliveras, P. Salembier, and C. Vachier. Morphological tools for segmentation: connected filters and watersheds. *Annales des Télécommunications*, 52(7-8):366–379, July-Aug. 1997.
32. Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36:1445–1453, September 1988.