Extensive operators in partition lattices for image sequence analysis

Luis Garrido, Philippe Salembier, David Garcia

ETSETB - Universitat Politècnica de Catalunya, Campus Nord-TSC, Módulo D5, c/ Gran Capità, s/n, 08034 Barcelona, SPAIN

Abstract

This paper deals with the class of extensive operators in the lattice of partitions. These operators are merging techniques. They can be used as filtering tools or as segmentation algorithms. In the first case, they are known as *connected operators* and, in the second case, they are *region growing* techniques. This paper discusses the basic elements that have to be defined to create a merging algorithm: merging order, merging criterion and region model. This analysis highlights the similarity and differences between a filtering tool, such as a connected operator, and a segmentation algorithm. Taking benefit from the filtering and segmentation viewpoints, we propose a general merging algorithm that can be used to create new connected operators (in particular self-dual operators) and efficient segmentation algorithms (robust criteria and efficient implementation).

Keywords

partition lattices, region growing, connected operators, image segmentation and filtering, sequence analysis.

Corresponding author: Luis Garrido Tel: (34) 93 401 1627, Fax: (34) 93 401 6447, E-Mail: oster@gps.tsc.upc.es
France-Telecom/CCETT under the contract 96ME22

1. Introduction

Image sequence analysis is becoming a very active field of research because of its large set of potential applications. For example, the MPEG4 video coding standard specifies how to code and compose video objects. However, the standard does not say how these objects can be extracted from an original sequence. In some cases, the extraction can easily be done at the production stage, but most of the time, unsupervised segmentation tools are necessary to set up an application. Services involving large video databases are also rapidly appearing. The efficient use of these services implies the definition of automatic analysis tools that can, at least partially, give access to what is happening in the sequence.

A very large set of tools and algorithms can be considered as "sequence analysis" tools. They range from the study of low level characteristics of the signal such as the color of the pixels to high level feature extraction such as the spatio-temporal definition of objects with a semantic meaning. In this paper, we focus on an intermediate level that combines what is traditionally seen either as "filtering" or as "segmentation". Tools studied in this paper aim at the definition of regions based on some homogeneity criteria (segmentation aspect) possibly with some constraints (filtering notion). Note that in order to define a complete segmentation algorithm, both filtering and segmentation tools are generally used.

Filters are generally used as pre-processing to remove part of the image content. They generally remove noise or some details that are not pertinent for the segmentation. In the context of a segmentation application, these filters should eliminate part of the information but preserve as much as possible the shape of the remaining objects. Examples of filters with this property are known as *connected operators* [12], [2], [11]. They interact with the image by merging the connected components of the space where the signal is constant (called *flat zones*). As a result, they do not introduce any new contour in the image. They perfectly preserve some contours and remove others. These filters are extensively used in segmentation applications [18], [17], [7], [13].

A large number of popular segmentation approaches also relies on merging strategies. The approach consists in, first, defining a set of initial regions and, second, in progressively merging regions to create a partition of the image. For instance in the *Split&Merge* [4] algorithm, the set of initial regions is defined by the Split process and the merging is performed between the initial regions depending on a homogeneity criterion. The *Region growing* algorithm [1] is another example: it relies on the merging of the set of initial regions with individual neighboring pixels that belong to an uncertainty area. Finally, the classical morphological tool for segmentation is the *watershed* [8], [19]. It also relies on a merging strategy: the initial regions are the regional minima of the gradient of the image to segment, and these minima are progressively expanded by merging of neighboring pixels in an order defined by the gradient value. Finally, the coding oriented segmentation algorithm presented in [6] also follows this basic merging strategy taking into account the coding cost of the resulting segmentation.

Connected operators, Split&Merge, Region growing and the watershed algorithms have been created in different contexts and for different applications. Connected operators have been designed as a filtering tool whereas the others are devoted to segmentation. However, they all rely on the same fundamental merging process. The objective of this paper is to investigate the basic features of the merging process and to see how the "filtering viewpoint" can benefit from the "segmentation viewpoint" and vice-versa. This study is done within the framework of sequence analysis and the *lattice of partitions* will be used as theoretical framework to study merging algorithms. We will see that the algorithms previously mentioned are extensive operators in this lattice. As major output of this study, we will create self-dual connected operators, that is connected operators having the same effect on dark and bright parts of the image and we will see how ideas coming from the "filtering viewpoint" can be used to select robust segmentation criteria and to efficiently implement them. Note that the implementation issue is a key point since the widespread use of a (sequence analysis) tool strongly depends on its computational efficiency.

The organization of this paper is as follows. Sec-

tion 2 reviews the basic notions of *connected operators, region growing* and proposes a general merging strategy. Then, the general framework of partition lattice is presented. An efficient implementation of the merging process is proposed in section 3. Based on the general merging process, section 4 defines new connected operators useful for sequence preprocessing and segmentation. Section 5 discusses sequence segmentation algorithms relying either on gray level or on motion homogeneity. Finally, section 6 is devoted to the conclusions.

2. Extensive operators in partition lattices

2.1. Region growing algorithm and connected operators

In the classical region growing algorithm, the set of initial regions is the set of image pixels. Then, based on a similarity measure or a distance, neighboring pixels are progressively merged to create regions. A classical example of similarity measure is the difference of mean gray level values. Note that the merging order is not known at the beginning of the process. Indeed, after each merging, the algorithm has to look for the neighboring pixel of a region that minimizes its distance to the region. However, when one pixel is merged with a region, the distance between this region and the remaining neighboring pixels may change. As a result, the merging order can only be defined step by step during the merging process itself. This merging process is iterated until a termination criterion is reached. To use a region growing algorithm, the main difficulty consists in finding a good similarity criterion and an efficient implementation.

Connected operators are morphological filtering tools that cannot introduce new contours in the image and can only merge flat zones. The most well known connected operators are the *opening* (or *closing*) *by reconstruction* and the *area opening* (or *closing*). Most of them are either extensive or anti-extensive operators in the lattice of gray level functions[1]. In practice, this means that they either deal with bright or dark image components. Let us describe the filtering process in the case of an anti-extensive operator such as an opening. First, the connected components of the space corresponding to each regional maximum are computed. Then, the operator assesses a given criterion value for each connected component (in the case of an area opening, the criterion value is the number of pixels). If the criterion value (the number of pixels) is higher than a given threshold, the connected component is preserved, otherwise it is merged with the flat zone(s) of closest lower gray level value. Finally, the procedure is repeated on the connected component resulting from the merging until the criterion value becomes higher than the threshold. Note that the merging order is defined by the absolute gray level value. The first pair of regions that is studied is the pair involving the brightest regions. Then, progressively, pairs of regions of lower gray level values are processed. The operator is anti-extensive because the gray level values of the pixels of the filtered image are lower than the gray level values of the original pixels.

Let us note that the algorithm used for the connected operator has strong similarities with a region growing process. It also involves an initialization step defining a set of initial regions (connected components corresponding to regional maxima) and it involves a merging step. There are, however, some important differences:

1. The merging order is known a priori. Indeed, the algorithm starts with regions of high gray level value and progressively goes down in the gray scale. The merging steps that are performed do not modify the processing order. This key difference explains why several efficient implementations of connected operators have been proposed in the past [18]. Generally, the idea is to use First-In-First-Out (FIFO) queues or even hierarchical FIFO queues to have a very fast access to the pair of regions to merge in the next step.
2. The second difference between connected operators and region growing algorithm is that connected operators make use of a criterion to decide whether a given merging has to be done or not. In our previous example, the criterion is the size of the connected components. In fact, the merging

[1] An operator ψ is extensive (anti-extensive) in the lattice of gray level functions if, for each pixel, the gray level value of the filtered image is always larger (smaller) than the gray level value of the input image.

order defines the order in which pairs of regions are processed by the algorithm, and the merging criterion decides whether a pair of regions has to be merged or not. In the case of a region growing algorithm, the merging criterion is simply a termination criterion. That is all pairs of regions defined by the merging order are actually merged until a termination criterion is reached (for example, the number of regions). This difference is rather natural since a segmentation algorithm, such as the region growing, intends to define the objects present in the image. However, connected operators are sieving tools. Therefore, beside the homogeneity notion represented by the merging order, they make use of a criterion (such as the size) to select the objects of interest.

3. Finally, when a connected operator merges two flat zones, the lowest gray level value of the pixels contained in the region is assigned to the resulting connected component. This gray level assignation strategy means that there is an implicit modeling of the gray level values: each region is represented by the lowest gray level value of its pixels. Connected operators are efficient simplification tools especially in the context of segmentation, however, in practice they have a major drawback: they just deal with regional maxima. By using the dual operator[2], one can create extensive connected operators that deal with region minima. However, they do not deal with regions of intermediate gray level values and are not self-dual (an operator ψ is said to be self-dual if it is dual of itself. In our case, it means: $\psi(f(x)) = -\psi(-f(x))$).

2.2. General merging algorithm

Taking into account the interesting features of the region growing algorithm and connected operators let us define now a general merging strategy. The proposed algorithm works on the Region Adjacency Graph (RAG). The RAG is a set of nodes representing connected components of the space (either regions or flat zones) and a set of links connecting two neighboring nodes. Each RAG represents a partition of the image. Note that, in this approach, each node of the graph can represent a region, a flat zone (which is a specific kind of region) or even a single pixel. A merging algorithm on this graph is simply a technique that removes some of the links and merges the corresponding nodes. In the sequel, we assume that the merging is done in an iterative way. To completely specify a merging algorithm one has to define three notions:

1. The **merging order**: it defines the scanning order of the links, that is the order in which the links are studied to know whether or not they should disappear. This order $O(R_1, R_2)$ is a real value and is a function of each pair of neighboring regions R_1, R_2.
2. The **merging criterion**: each time a link is processed, the merging criterion decides if the merging has actually to be done or not. It is also a function $C(R_1, R_2)$ of two neighboring regions R_1 and R_2, but it can only take two values ("merge" and "do not merge").
3. The **region model**: when two regions are merged, the model defines how to represent the union and what are the main characteristics that should be kept in order to continue the process (that is to compute future merging criterion values and possibly the future merging order). Let us denote this model by $M(R)$.

Note that this merging strategy allows the implementation of both region growing algorithms and connected operators. In the case of a region growing algorithm, the three notions have the following features: first, the merging order is defined by the similarity measure between two regions. Second, the merging criterion always states that two regions have to be merged until a termination criterion is reached (for example, a minimum number of regions or a quality criterion). Third, the model defines a gray level function (for example the mean) that can be used in the future to compute the similarity between the regions and their neighbors. In the case of a connected operator such as an opening by reconstruction or an area opening, the three notions are also present: first, the merging order is simply defined by the absolute gray level value. Second, the merging criterion relies on the assessment of a criterion (for example, the number of pixels of each region in the case of

[2]If ψ is a connected operator and $f(x)$ the original image, the dual operator ψ^* is defined by: $\psi^*(f(x)) = -\psi(-f(x))$.

an area opening) and, third, each region is modeled, after merging, by the lowest gray level value.

The definition of a complete algorithm requires the specification of the merging order, the merging criterion and the model. To illustrate this approach, let us intuitively analyze the meaning of the merging order and of the merging criterion. The merging order defines the homogeneity notion and is closely related to the notion of objects. In the case of a region growing algorithm, if the similarity measure is equal to the gray level difference, objects are assumed to be connected components and homogeneous in gray level value. In the example of connected operators previously described, objects are assumed to be bright parts of the image. As already pointed out, the difference between a filter and a segmentation algorithm mainly relies on the merging criterion. In the case of a segmentation algorithm, we would like to define all objects, this means that the merging criterion should always state that the merging has to done until a termination criterion is reached. In the case of a filter, the objective is to select some objects among the set of all possible objects. This means that the merging criterion should act as a sieve among the set objects defined by the merging order.

2.3. Partition lattice

In order to theoretically study the characteristics of RAG oriented processing of images, it is convenient to define a mathematical structure called the *partition lattice* [16]. A lattice is a set of elements with a partial order relation, $\leq$, in which any set of elements possesses a supremum, $\vee$, and an infimum, $\wedge$. In the case of a partition lattice, elements are all the possible partitions P_i of the image E. Denote by $\{R_i^n\}_n$ the set of regions of partition P_i. Let us use the following partial order relation:

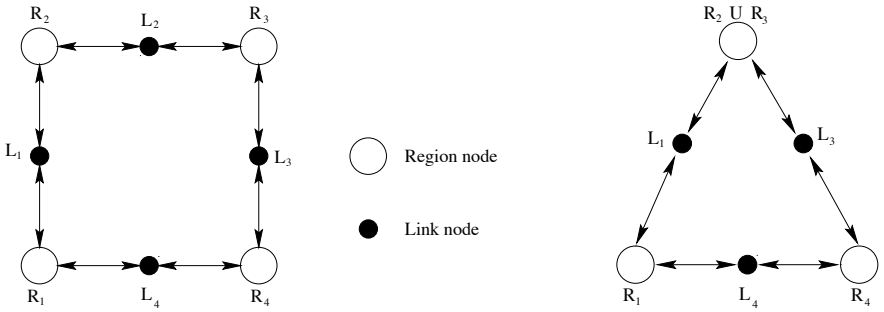
Definition 1: Partition P_i is smaller than partition P_j if all pixels belonging to a given region of P_i also belong to a single region of partition P_j.

$$P_i \leq P_j \Longleftrightarrow \forall x, y \in R_i^n, \exists m \text{ such that } x, y \in R_j^m \tag{1}$$

Partition P_i is said to be smaller or finer than partition P_j. By duality, the $\geq$ relation can be defined: $P_i \geq P_j$ if for each pixel x, y belonging to two different regions of P_i, x, y also belong to two different regions of partition P_j. Note that $\leq$ is only a partial order and that two arbitrary partitions may not be comparable. However, when partitions can be compared, the order relation characterizes a hierarchical structure between them. The largest partition is the partition made of one single region: the image itself. The smallest partition is created by considering that each individual pixel forms one region. In a lattice, two dual operators are generally defined: the infimum and the supremum. The infimum of a set of partitions, $\wedge\{P_i\}$, is a partition made of the intersections of all regions in the set of partitions. By contrast, two pixels x, y belong to the same region of the supremum of a set of partitions, $\vee\{P_i\}$, if there exists at least one region in one of the partitions P_i that contains these two pixels.

The definition of a lattice is mainly useful to study the properties of the operators that act on its elements. The most important properties of a lattice operator ψ are the following:

- ψ is **anti-extensive** if its output is always smaller than its input. Here, for any partition P_i, $\psi(P_i) \leq P_i$. In terms of segmentation, the operator is a splitting algorithm that maintains the contour of the input partition and defines some new contours by splitting the existing regions. Examples of such operators can be found in particular in [9], [13].
- ψ is **extensive** if its output is always larger than its input: $P_i \leq \psi(P_i)$. The operator is a merging algorithm. This property is the common point between connected operators and region growing algorithms. The objective of this paper is to focus on this class of *extensive partition operators*. Note that, in the case of the partition lattice, "extensive" means "merging". It has not to be mistaken with "extensive" in the context of the lattice of gray level functions. In this latter case, a closing is extensive because the output gray level values are higher than the input gray level values.
- An operator ψ is **idempotent** if $\forall P_i, \psi(\psi(P_i)) = \psi(P_i)$. This property is quite important because it guarantees that the operator has totally processed the input and there is no need to use it



A) Original RAG B) RAG after merging of R_2 and R_3

Fig. 1. Region Adjacency Graph used for the merging algorithm

twice. In the context of an extensive partition operator, this property means that the merging process has a clear and stable termination criterion.

- An operator ψ is **increasing** if $\forall P_i, P_j,\ P_i \leq P_j \Rightarrow \psi(P_i) \leq \psi(P_j)$. In other words, the operator preserves the relation between the partitions. In particular, if there is a hierarchical relation between two input partitions, this hierarchical relation is preserved after the processing by ψ. As will be seen in the sequel, this property is not easy to obtain with simple connected operators or region growing algorithms.

In the following, we focus on extensive partition operators. As mentioned previously, the definition of such an operator relies on three major points: the merging order, the merging criterion and the region model. Before analyzing some examples, let us describe how they can be efficiently implemented.

3. Efficient implementation of the merging process

3.1. Merging algorithm on a RAG structure

This section is devoted to the efficient implementation of the merging algorithm described in section 2. As will be seen, the algorithm uses a hierarchical queue able to deal with floating point priorities and a RAG structure. The RAG structure is depicted in Fig. 1.A. The classical RAG is a graph where each node represents a region of the image and is connected to its neighbors. The graph structure used here is similar but contains some more information. The white nodes in Fig. 1.A, called "region-nodes", represent the regions of the image, whereas the black nodes, called "link-nodes", represent the links between regions. Observe that each link-node points to the two region-nodes it is linking while each region-node points to the neighboring link-nodes. By using this representation, the access to the set of links-nodes that should be updated after the merging of two region-nodes can be done efficiently. Suppose, for example, that R_2 and R_3 are merged (see Fig. 1.B). The link to remove is L_2, and the link-nodes whose ordering has to be updated are L_1 and L_3.

In Fig. 2, the general scheme of the merging process is represented. In each block, the merging notion involved (merging order, merging criterion and region model) is described. The merging algorithm can be divided into two stages. The first one ("initialization" block) is devoted to the initialization of the structures needed for the merging, i.e. the RAG structure and the hierarchical queue (the latter will be discussed in the sequel). Each region is initialized by computing its model. The set of initial regions can be either the set of pixels (each pixel is one region), the set of flat zones or any pre-segmentation. The next step consists in calculating the merging order for each link-node using the model of the associated region-nodes. Finally, each link-node is inserted in the hierarchical queue at the position defined by the merging order. This ordering (represented by a floating point number) defines the insertion point of each link-node into the hierarchical queue.

Once the first stage is completed, the merging process begins. Observe that the merging algorithm is

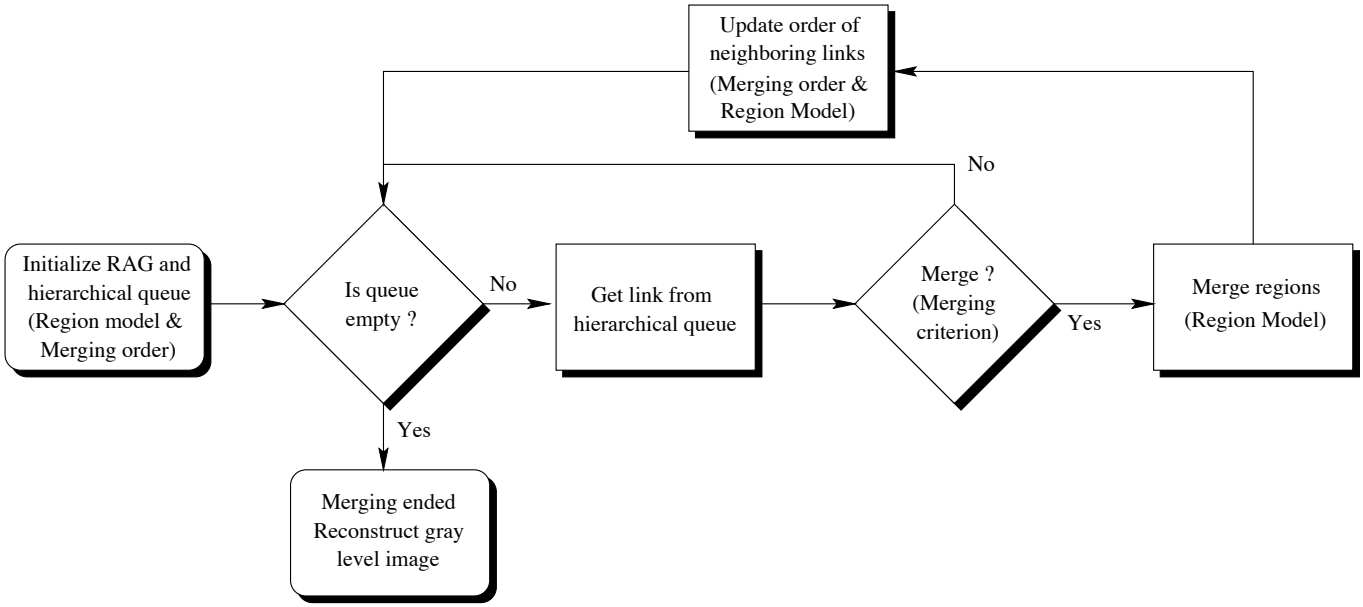


Fig. 2. Block diagram of the general region merging algorithm

an iterative process that ends once the hierarchical queue is empty. The first step in this iterative process is to extract the highest priority link-node of the queue. The next step consists in deciding whether the link-node has to be merged or not. This merging, as discussed in previous sections, is defined by the "merging criterion". If the merging criterion decides not to merge the regions, the algorithm returns to the first block of the merging algorithm. Note that this decision is final in the sense that the link-node is removed from the queue and the corresponding pair of regions will never be merged. If the merging criterion decides to merge, the next step consists in merging the information of both regions associated to the link. As a result, the region model has to be updated. In order to obtain an efficient implementation, a recursive algorithm is needed to avoid redundant computations. "Recursive" here means that the model of the union of the two regions should be computed from the models of the two initial regions. The drawback of this strategy is the need of memory allocation for each region model and, therefore, the memory cost may be high. Once the regions have been merged, the values of the merging order of the neighboring links are updated. This implies the extraction of the corresponding links from the queue, the computation of the new priority (using the models of neighboring regions) and the insertion of the links into the queue with their new priority. At this point the merged RAG structure has been computed and updated: the iterative process starts again by checking if the queue is empty.

This implementation of the merging algorithm has strong similarities with efficient implementations of connected operators involving the so-called reconstruction process and of the watershed algorithm [19], [18]. In all cases, the key element of the algorithm is the hierarchical queue. In the context of our general merging algorithm, the main features of the queue should be: first, fast access, insertion and deletion of an element and, second, no constraints on the dynamic range of the priority (floating point ordering).

3.2. Hierarchical queues and binary trees

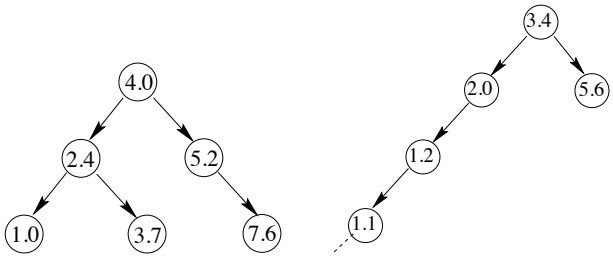
A hierarchical queue is a queue where each element has a given priority. The elements can be introduced in any order in the queue, and the extraction is done in descending order of priority (the elements with higher priority are extracted first). Elements having the same priority follow a First-In-First-Out (FIFO) rule. Hierarchical queue structures have been extensively used for fast implementation of connected operators

or for the watershed algorithm [19], [18]. The main difference with the implementation proposed here is that the merging order is a priori defined in the case of classical connected operators or of the watershed, whereas the merging order has to be constantly updated in our case. Therefore, the hierarchical queue has to be updated and re-organized on line. It can be seen as a dynamic hierarchical queue by contrast to the more classical "static" hierarchical queue. Another drawback of the classical hierarchical queue is that it is generally limited in the range of priority it can deal with and does not allow a floating point priority.

The solution proposed here is based on a binary tree [5]. Binary trees are extensively used to implement fast searching, insertion and deletion algorithms and have no constraints on the dynamic range of the data allocated in it. The basic idea behind the implementation of hierarchical queues with binary trees is depicted in Fig. 3.A. Each node of the tree represents a given priority. A left child node is characterized by having a priority lower than its father, while a right child node has a priority greater than its father. At each node, the link-nodes having the same priority are stored in a FIFO structure.

In order to extract the node (and therefore the list of link-nodes) with highest priority, one begins with the root node and walks down the tree using the right branches until a node with no right branch is found.

Search, insertion and deletion of nodes in the queue can be done in $O(log_2 N)$ steps where N is the number of nodes in the tree [5]. One of the drawbacks of using binary trees to implement hierarchical queues is that the tree may degenerate. This problem happens when the input data (in the merging algorithm the input data is the merging order) is not random. Suppose, for example, that the ordering depends on the area of the region: for example, the higher the area, the lower the priority. As a result, the priority of the links will decrease as the regions grow in area. This results in a tree that "leans" too much to the left (see Fig. 3.B). In this case, the $O(log_2 N)$ efficiency does not hold anymore, because the tree is becoming a simple FIFO queue. Fortunately, there is a solution to this problem called "balanced trees". The method [5], [20] is based on keeping track of the balance factor of the tree each time an element is inserted or deleted. A rough idea is presented in Fig. 4: at each node, the height of the left tree minus the height of the right one is computed (the height at a given node is defined as the length of the longest path from the node to a leaf node). A binary tree is called balanced if the balance factor of every node of the tree never differs by more than ± 1. If unbalancing occurs because of insertion or deletion, the balancing is reestablished properly by "rotating" some nodes of the tree.



A) Balanced binary tree B) Unbalanced binary tree

Fig. 3. Examples of binary trees, in each node the priority is indicated

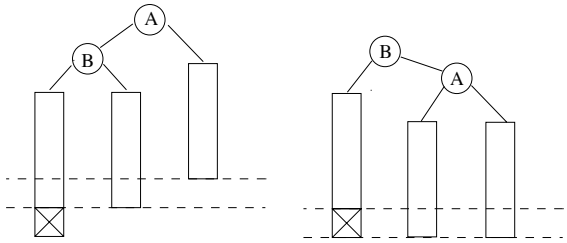
The algorithm involved in the balancing of the tree is very efficient if one has to manage thousands of nodes. Efficient implementations of insertion and deletion using balancing techniques can be found in [20].

3.3. Efficiency of the algorithm

The efficiency of the algorithm turns out to be very high. In practice, for simple models (constant or first order), the computation time is about the same as for a connected operator or a watershed algorithm. For instance, if one starts at the pixel level (initial regions made of one pixel) and merge progressively regions until the partition is made of one single region, the CPU time is about 1,5 second (about ten seconds) for a QCIF (CIF) frame on a 200MHz Pentium.

4. Simplification tools

The objective of this section is to present some new connected operators that can be used to simplify an image or an image sequence either to analyze it or as pre-processing before a segmentation algorithm. As discussed in the introduction, connected operators used in practice are interesting as region-based



A) Unbalancing due to insertion B) Restoring the balance

Fig. 4. Example of binary tree balancing

simplification tools. However they deal with either bright or with dark image components. In particular, they do not deal with transition areas between dark and bright components. With the structure defined in sections 2 and 3, it is rather easy to create new connected operators that are self-dual in the gray level sense. To this end, one has to define the merging order $O(R_1, R_2)$, the merging criterion $C(R_1, R_2)$, and the region model $M_R(x)$ in such a way that the resulting operator be self-dual.

4.1. Region model: M_R

The first choice that has to be made is how each region is going to be modeled. For instance, in the case of an opening (a closing), each region is modeled by the lowest (highest) gray level value of its pixels. Obviously, this model does not allow the creation of self-dual operators. In this section we assume that the model is constant within the region ($M_R(x) = C$). Two simple self-dual models of a gray level distribution are the mean and the median. In order to allow a fast implementation of the merging process, the model of region $R = R_1 \cup R_2$ should be computed recursively from the models of the two merged regions R_1 and R_2. In the case of the mean model, one has to compute the weighted average of the mean of each region:

$$M_R = (N_1 M_{R_1} + N_2 M_{R_2})/(N_1 + N_2) \qquad (2)$$

where N_1 and N_2 denote the numbers of pixels of each region. In the case of the median, one has simply to select the model of the largest region [3]:

$$\text{if } N_1 < N_2 \quad \Rightarrow \quad M_R = M_{R_2} \qquad (3)$$
$$\text{if } N_1 > N_2 \quad \Rightarrow \quad M_R = M_{R_1} \qquad (4)$$
$$\text{if } N_1 = N_2 \quad \Rightarrow \quad M_R = (M_{R_1} + M_{R_2})/2 \qquad (5)$$

The median is known to be a robust estimator and, from our practical experience, it is indeed more robust than the mean. In fact, it rapidly defines areas where the model is stable. These areas can be seen as the core of the final regions. Then, small regions are progressively merged to the core without modifying the model. This property does not hold in the case of the mean, since the merging of a small region with a large one also modifies the model of the large region. In the sequel, we assume that the median is used in all cases (except if stated differently). Note that the model can be used for gray level images as well as for color or multichannel images such as dense motion fields. In this case, M_R is a multidimensional model and each component is modeled independently. Finally, let us mention that the median and the mean models are constant within the region. They can be considered as simple zero order models. However, in section 5, the model will be extended to a first order model in order to deal with motion homogeneity. Of course, in this last case, the definition of connected operator should be extended since flat zones are not constant any more.

4.2. Merging order: $O(R_1, R_2)$

The merging order defines the notion of object. It can be seen as a measure of the likelihood that two neighboring regions belong to the same object. Let us assume in a first step that we deal with gray level images. In this case, a natural measure is the squared error between the original image and the model in the region of support defined by $R = R_1 \cup R_2$: $O(R_1, R_2) = \sum_{x \in R_1 \cup R_2} (f(x) - M_{R_1 \cup R_2}(x))^2$, if $f(x)$ denotes the gray level value of the original image at position x. This criterion actually allows the extraction of meaningful objects that are homogeneous in gray level, however it does not define precisely the

[3] Note that the resulting model is not exactly the median of the original pixels since the median is computed in an iterative way.

contours. This drawback is related to the size dependence of the criterion. Indeed, small regions have a tendency to merge together since the squared error contribution of the union of two small regions is small compared to the contribution resulting from the merging with a large region. An alternative solution is to use the difference between the models of the two neighboring regions: $O(R_1, R_2) = (M_{R_1} - M_{R_2})^2$ or the Mean Square Error (MSE): $O(R_1, R_2) = \sum_{x \in R_1 \cup R_2} (f(x) - M_{R_1 \cup R_2}(x))^2 / (N_1 + N_2)$, where N_1 and N_2 denote the numbers of pixels of regions R_1 and R_2. These criteria are independent of the size and provide a very good definition of the contours, however, they do not define in a robust way the regions themselves. In practice, they produce a few large regions surrounded by a very large number of tiny regions. To obtain a compromise between the two previous orders, we propose the following merging order:

$$\begin{aligned} O(R_1, R_2) &= N_1 (M_{R_1} - M_{R_1 \cup R_2})^2 \\ &+ N_2 (M_{R_2} - M_{R_1 \cup R_2})^2 \end{aligned} \quad (6)$$

Note that, in the case of the median model and if R_1 is smaller than R_2, the model after merging is M_{R_2} and the order reduces to $O(R_1, R_2) = N_1 (M_{R_1} - M_{R_2})^2$. It is the squared difference between the models weighted by the size of the smallest region. Finally, note that the merging order can be easily extended to deal with color or multichannel images. In this case, the order is defined as a linear combination of the order values computed on each component.

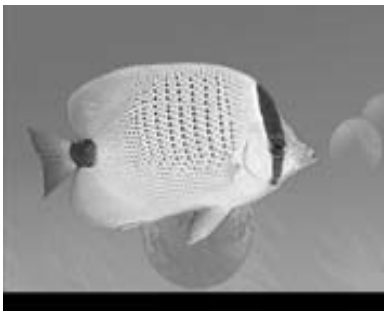
4.3. Merging criterion: $C(R_1, R_2)$

The function $O(R_1, R_2)$ defines the order in which the regions have to be processed. Following a given homogeneity notion, it extracts the pair of regions that most likely belongs to the same object. Now, the objective of the merging criterion is to select among the set of possible objects a few that fulfill a given criterion. This function allows the definition of filtering, that is sieving, tools.

For analysis or preprocessing, two criteria are particularly useful: area and contrast.

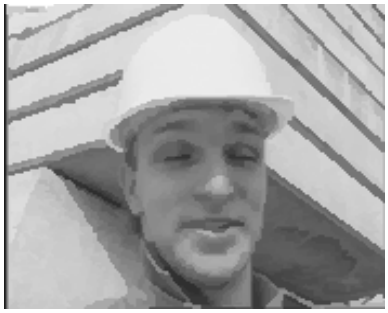
- *Area* merging criterion: The objective of this criteria is to remove from the original image all objects that are smaller than a given threshold. To this end, the criterion simply states that two regions have to be merged if at least one of them is smaller that a given threshold (the size of the region is defined as its number of pixels).
To see the effect of the operator, two original frames presented in Fig. 5 have been processed. The resulting frames are shown in Fig. 6. Fig. 6.A and B show the effect of a simplification when the area threshold is set to 10 pixels. These images contain most of the original information except the very small regions. For example, the texture of the fish has been removed. The process is a connected operator and the contour information of the objects that have not been removed is well preserved. Note that both images are made of flat zones of at least 10 pixels. These images can be used as starting point for a segmentation algorithm. The simplification effect can also be seen in the number of flat zones: the filtered *Foreman* (*Bream*) frame is made of 974 (574) flat zones compared to the 18585 (10719) flat zones of the original frame. Fig. 6.C and D show the same results when the area threshold is increased to 50 pixels. These images are clearly simplified with respect to the original frames (all flat zones have at least 50 pixels!). Area simplification of rather large size is useful for example if one wants to define regions of reasonable size to allow the estimation of the motion of each flat zone. Note that the simplification effect of this area operator is different from that of an area opening or closing. The operator proposed here not only deals with bright or dark areas but also with transition areas.
- *Contrast* merging criterion: the contrast between two neighboring regions can be defined as the difference between the models of each region. Assuming that the model is constant within the region, we have: $Contrast(R_1, R_2) = (M_{R_1} - M_{R_2})^2$. The contrast criterion merges two region if $Contrast(R_1, R_2)$ is lower than a given threshold. This operator is illustrated in Fig. 7. The upper row shows the frame resulting from a contrast of 5 whereas the lower row deals with a contrast of 20. As previously, the simplification is "flat zone oriented" and the resulting numbers

A) Foreman sequence

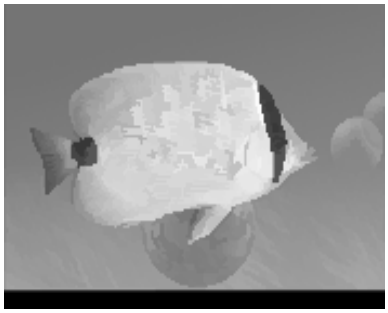


B) Bream sequence

Fig. 5. Original frames. Foreman 18585 flat zones, Bream 10719 flat zones



A) Area 10 pixels, 974 flat zones



B) Area 10 pixels, 574 flat zones

C) Area 50 pixels, 211 flat zones

D) Area 100 pixels, 80 flat zones

Fig. 6. Area connected operator

of flat zones are given with the images.

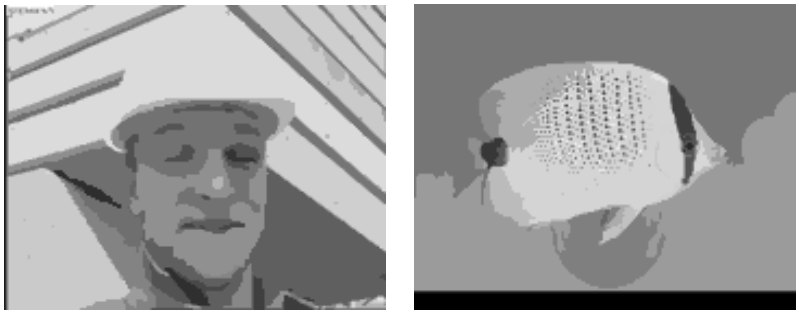
From the partition lattice viewpoint, it can be shown that both the area and the contrast operators are extensive, but only the area operator is idempotent. However, the iteration of the contrast operator is idempotent. If we consider them as operators in the lattice of gray level functions, they are self-dual operators (process dark and bright areas in a similar way).

5. Image and sequence segmentation

5.1. Region model, merging order and merging criterion

The main difference between a segmentation algorithm and a filter is the merging criterion which has not to select some of the objects but rather to accept all of them until a termination criterion is reached. The termination criterion depends to a large extend on the particular type of segmentation one wants to

A) Contrast 5, 4979 flat zones B) Contrast 5, 3190 flat zones



C) Contrast 20, 923 flat zones D) Contrast 20, 965 flat zones

Fig. 7. Contrast connected operator

achieve. It will be more precisely described in the following sections.

The model used within each region is a compromise between its capacity to represent the homogeneity notion of interest and its complexity. In section 4, constant models have been used and the model corresponding to the merging of two regions was computed either as the weighted average or as the median of the models of each region. This model has the advantage of being very simple and is adequate for processing gray level or color images if the number of regions remains rather high. Of course, if the expected number of regions is very low (regions will likely represent more complex gray level distributions) or if the input information has specific characteristics, the constant model can be extended to a general polynomial model. In this section, we will use first order polynomials of the type: $M_R(i,j) = \alpha i + \beta j + \gamma$, where i, j denote the spatial coordinates of the pixels. When two regions are merged, this model can be updated either by a mean square rule or by a median rule. In the first case, the resulting polynomial is computed by linear regression, whereas, in the case of the median, the resulting model is equal to the model of the largest region. In the framework of sequence analysis, first order polynomial models are particularly useful to represent dense motion fields. Indeed, a very large number of natural motion such as translation, rotation, zoom can be modeled on a region basis with this kind of polynomials.

As in the case of filters, the merging order defines the homogeneity notion of the objects. In the following, the homogeneity notion defined by equation 6 is used since it gives a good compromise between region detection and contour localization.

5.2. Intra-frame segmentation

For intra-frame or still image segmentation, the decision criterion simply defines the end of the merging process. Two useful termination criteria are:

- the Peak Signal to Noise Ratio (PSNR) between the original image and the modeled one,
- the number of regions.

At each merging, the PSNR and the number of regions decrease. Both criteria provide a clear termination point. Let us mention that if we consider the

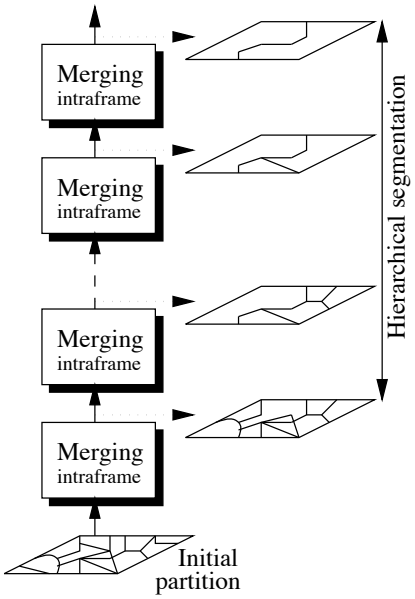


Fig. 8. Hierarchical segmentation by merging

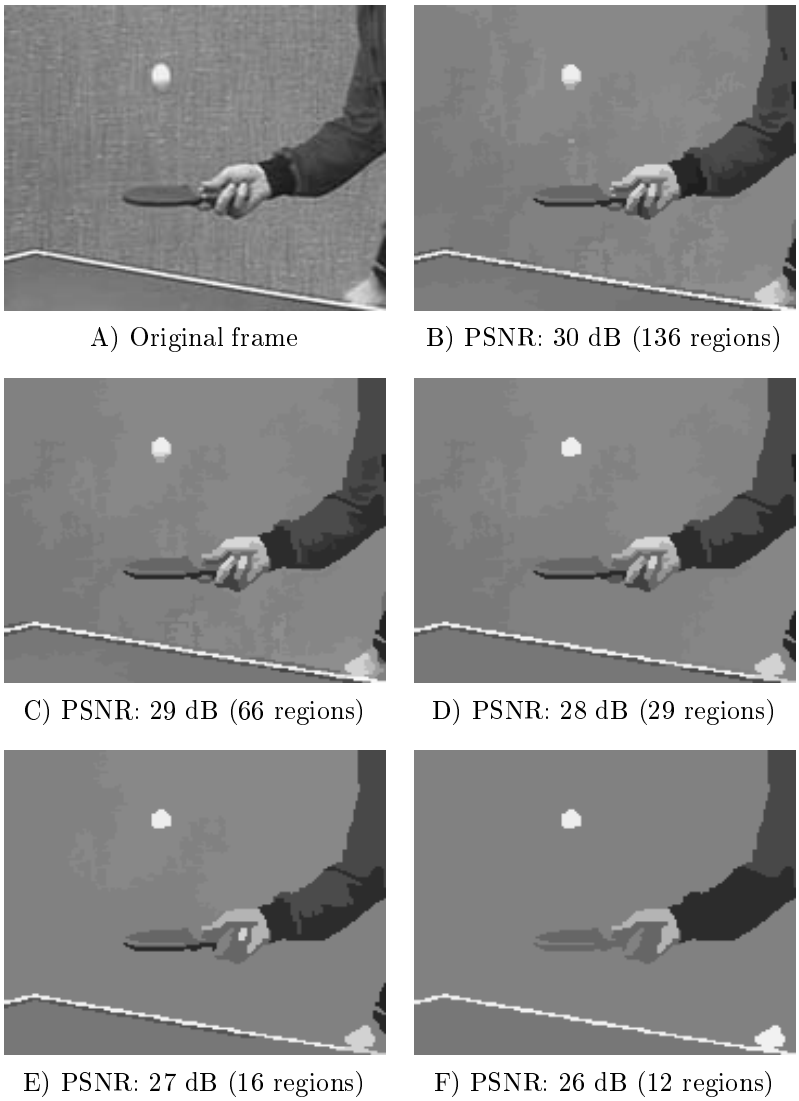
segmentation as an extensive operator in the partition lattice, the termination criterion involving the number of region creates an idempotent operator but not the PSNR criterion.

The merging strategy defined in section 2 is particularly useful to compute a set of hierarchical partitions. The approach is illustrated in Fig. 8. Starting from an initial partition, which can be either the smallest partition of the space (partition where each individual pixel is a region) or the partition of flat zones, several merging steps are performed. Note that with the implementation of section 3, it is possible to compute the full hierarchy with only one run of the merging algorithm. Indeed, one has only to output some of the partitions created at intermediate stages of the merging process. This approach is therefore an efficient way to compute an entire hierarchy of partitions (for example, the average computation time is about 1,5 second for QCIF images on a 200MHz Pentium).

An example of hierarchical segmentation is shown in Fig. 9. The input image of the merging process is the original frame. As a result, the segmentation creates regions that are homogeneous in gray levels. In Fig. 9, we show images where each region has been represented by its model (median in this case). Five partitions corresponding to PSNR values ranging from 30 to 26 dB are presented. As can be seen, partitions corresponding to 28-27dB provide simple approximations of the original frame in terms of number of regions. However, the main objects of the scene are precisely defined. Note that these partitions are structured in a hierarchical way.

The same scheme can be used to deal with motion-oriented segmentation. Assume that we start from an initial segmentation resulting from a spatial segmentation (for instance, one of the levels of the previous example). The motion can be estimated on each region of the initial estimation. To this end, we have used the technique described in [3], [14]. This technique assigns to each region a polynomial model describing the apparent motion in the horizontal and the vertical directions. The model is estimated using differential methods. From this motion model, two dense motion fields can be created by assigning to each pixel the values of its horizontal and vertical displacements. These two dense motion field images are now used as input to the merging algorithm. The original partition and the dense motion fields are shown on the upper row of Fig. 10. To encode the motion fields, a gray value of 128 has been assigned to a displacement of zero and bright (resp. dark) pixels denote a positive (resp. negative) displacement. The merging process has to deal with a two component image. As a result, the algorithm defines regions that are homogeneous in the sense of its input images, that is motion. The results of the segmentation are shown in Fig. 10. As can be seen, the algorithm merges region on the basis of their motion homogeneity. The final segmentation shown in the lower row of Fig. 10 involves 5 regions: 3 for the face and 2 for the background. Note that the motion field is not re-estimated during the merging process. This choice was done to limit the computational complexity of the algorithm, however, in future work we will investigate fast techniques to re-estimate the motion and we will study the improvement that is obtained.

Before closing this section on intra-frame segmentation, let us discuss a potential use of the merging algorithm. An image analysis strategy often used in mathematical morphology is the so-called *granulometry* [15]. The approach consists in using a hie-



A) Original frame B) PSNR: 30 dB (136 regions)

C) PSNR: 29 dB (66 regions) D) PSNR: 28 dB (29 regions)

E) PSNR: 27 dB (16 regions) F) PSNR: 26 dB (12 regions)

Fig. 9. Example of gray level intra-frame segmentation (termination criterion: PSNR).

rarchical filtering structure as the one presented in Fig. 8 (classically, the filters are morphological opening with various size of the structuring element) and in measuring an image characteristic (classically the integral of the gray level image) at each filter output. This strategy allows the characterization of what has been removed by each filter. Considering the segmentation algorithm as a filtering tool, the same approach can be used to characterize the content of each frame. Assume for example, that the segmentation/filter follows a gray level homogeneity criterion and that the termination criterion is the PSNR. PSNR values ranging from 40dB to 25dB are assigned to 16 levels of the hierarchical structure of Fig. 8 and let us suppose that we measure the number of regions of each resulting partition. The results are shown in Fig. 11. Three typical frames (frames #50, #200 and #270) of the *Foreman* sequence have been processed and the three corresponding curves are shown. These curves define how many regions are necessary to achieve a given PSNR. Intuitively, this number is function of the image "complexity": for simple frames with a few objects (homogeneous in gray level), the number of regions necessary to achieve a given PSNR is rather

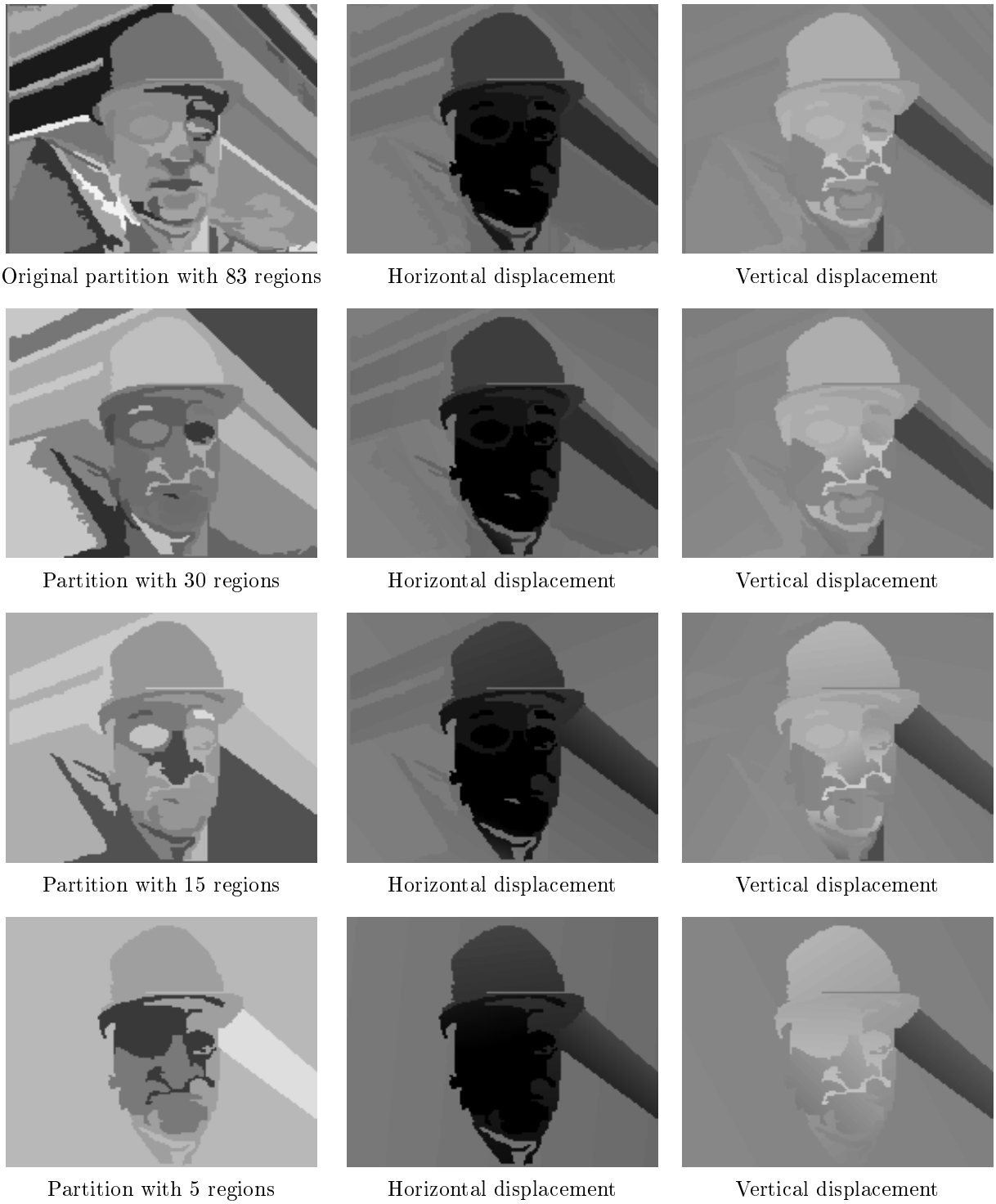


Fig. 10. Example of motion segmentation; first row: initial (spatial) segmentation, last row: final segmentation, intermediate rows: intermediate results of the merging process.

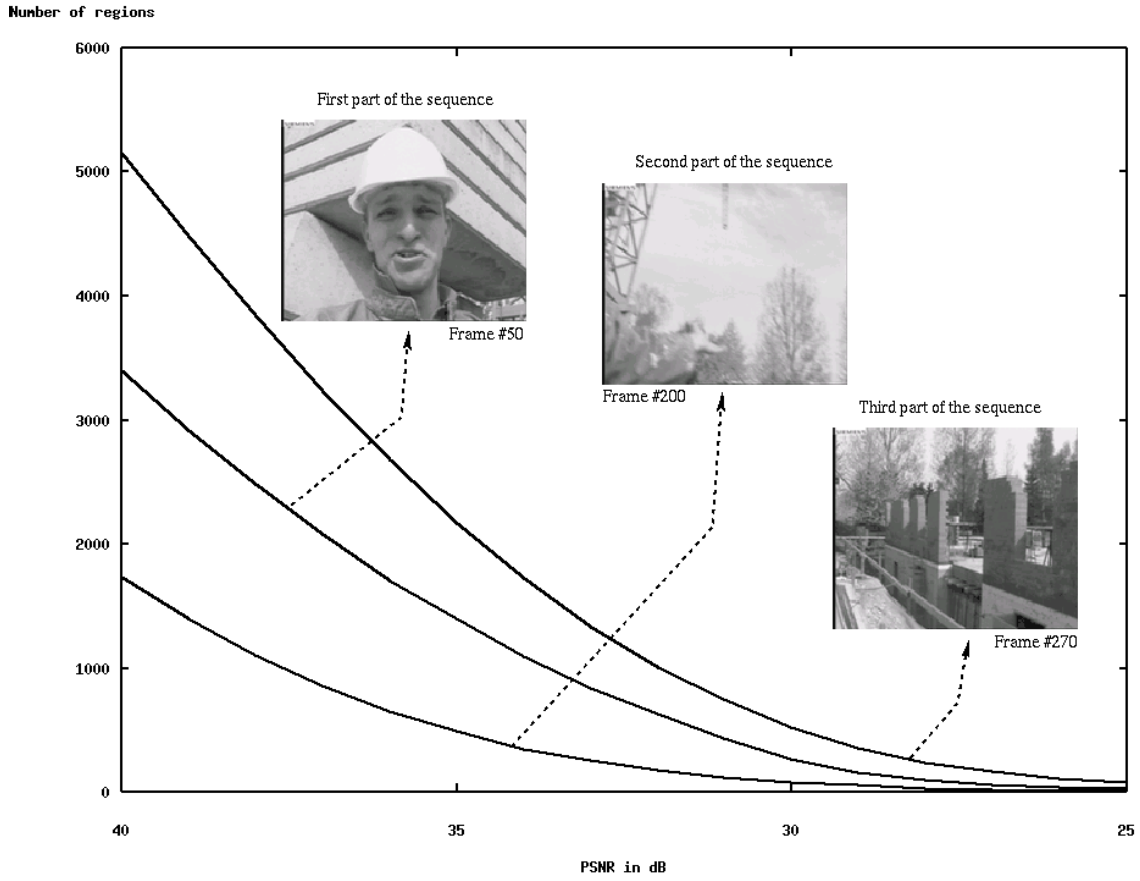


Fig. 11. Number of regions as a function of the PSNR for three typical frames of the Foreman sequence.

low. By contrast, if the image involves a large number of contrasted objects, a high number of regions is necessary to reach the same PSNR. This intuitive discussion is confirmed by Fig. 11: among the three frames, the simplest one is the frame #200. It involves a few objects and the sky covers most of the frame. The corresponding curve is the lowest one. By contrast, the most complex frame is frame #270. It involves a fairly high number of contrasted objects and the corresponding curve is the highest one. Finally, frame #50 is of intermediate complexity and produces a curve in-between the two previous ones. Each curve (or a reduced set of curve points) can be used to characterize the image content.

This strategy can be extended to the analysis of the sequence. Fig. 12 shows the number of regions necessary to obtain a given PSNR as a function of the frame number (one over ten frames has been processed). It is plotted as a 3D surface. Clearly, three different time sections can be identified in this surface. The first section involves frames #0 to #180 which are of moderate complexity. The scene shows a worker talking in front of building. The second section (frames #190 to #210) involves simple frames resulting from a panning of the camera. Finally, the third section (frames #220 to #290) is composed of complex frames with a large number objects. As can be seen, the approach can be used to extract scenes that have a similar "complexity". Finally, note that the interest of this approach relies not only on the feature it extracts but also on its computational simplicity.

5.3. Sequence segmentation

In this section, we focus on a more complex segmentation scheme combining several homogeneity crite-

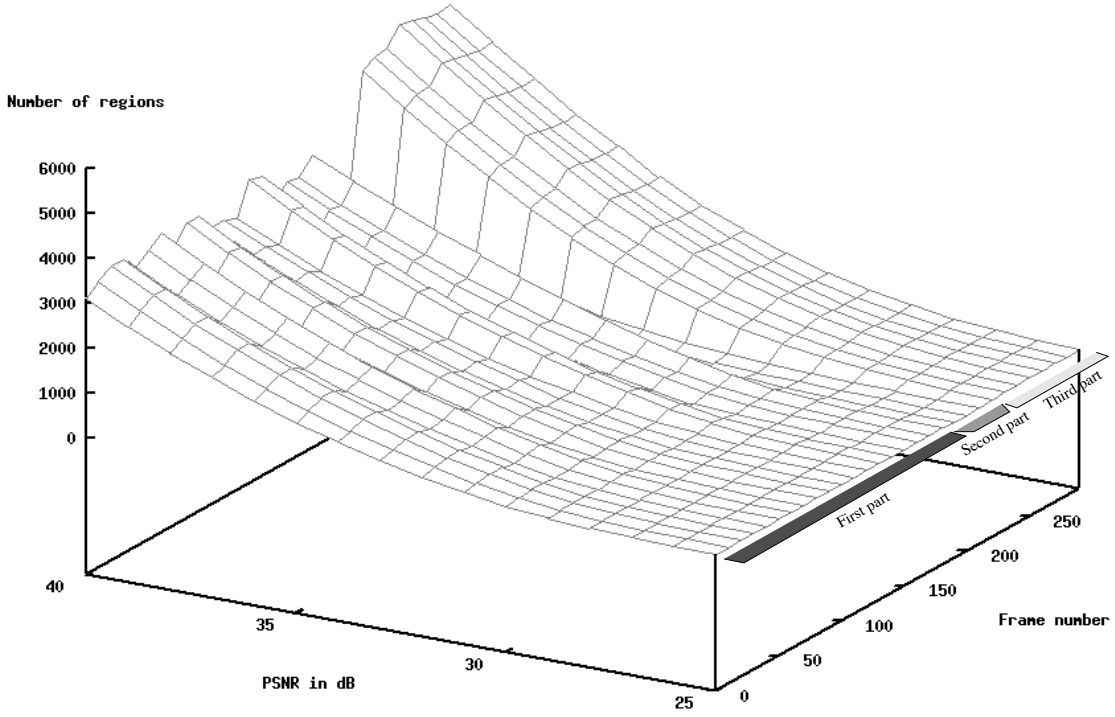


Fig. 12. Sequence analysis: Number of regions to obtain a given PSNR as a function of the frame number (Foreman sequence).

ria. The goal of the algorithm is to segment a video sequence in a recursive and causal way. To this end, we propose to define at each time instant t a gray partition $P_g(t)$ (partition where regions are homogeneous in gray level[4]) and motion partition $P_m(t)$ (partition where regions are homogeneous in motion). The gray partition is created by merging regions belonging to an initial partition which is the partition of flat zones $P_{fz}(t)$. Similarly, the motion partition is created by merging regions of the gray partition. As can be seen, the three partitions are structured in a hierarchical way: the lower level is made of flat zones, the second level is the spatial segmentation and the upper level is the motion segmentation. This structure has been selected for the two following reasons: first, a contour between two "motion regions" (regions homogeneous in motion) should coincide with a contour between two "spatial regions". Indeed, a precise spatial definition of the contour can only be achieved in the original image since no estimation has to be performed. Second, in order to estimate the motion and to perform a motion segmentation, elementary regions should have been previously defined. The motion estimation can therefore rely on a partition that is related to the image. This approach avoids drawbacks of block-based motion estimation.

Moreover, for the two upper levels of the hierarchy, we would like to track regions in time, that is to relate regions of partitions $P_g(t)$ and $P_m(t)$ with regions of partitions $P_g(t-1)$ and $P_m(t-1)$. Region tracking is useful for the segmentation itself. For example, when estimating the motion of a region of the gray partition at time t, one can discard pixels not belonging to this region at time $t-1$. Moreover, region tracking is mandatory if one wants to analyze the time evolution of the regions.

The global scheme is depicted in Fig. 13. Assume that the partition hierarchy $P_{fz}(t-1)$, $P_g(t-1)$ and $P_m(t-1)$ at time $t-1$ is known. The purpose of the

[4]Note that we have actually implemented and tested a gray level segmentation scheme, however as described in section 2, the approach can be easily extended to color segmentation.

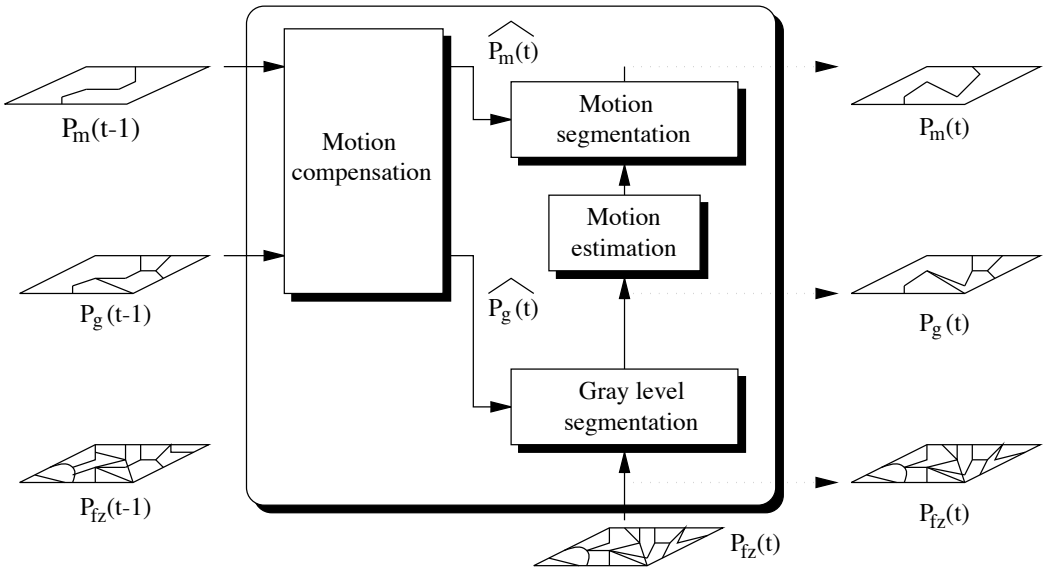


Fig. 13. Sequence segmentation algorithm. Hierarchy of flat zones, gray level and motion partitions.

algorithm is to create a similar hierarchy at time t. Note that $P_{fz}(t)$ can be directly extracted from the original frame. The first step of the algorithm is to motion compensate partitions $P_g(t-1)$ and $P_m(t-1)$. This creates two predicted partitions $\widehat{P_g(t)}$ and $\widehat{P_m(t)}$. Then, the first segmentation to be performed is the gray level segmentation. It creates $P_g(t)$ based on the knowledge of the current partition of flat zones $P_{fz}(t)$ and the predicted gray partition $\widehat{P_g(t)}$. Once, the gray partition $P_g(t)$ has been created, the motion of each region is estimated and a second segmentation step is performed to create the motion segmentation $P_m(t)$. In the following, these steps are more precisely described and illustrated.

5.3.1. Motion compensation

The compensation of partitions $P_g(t-1)$ and $P_m(t-1)$ is achieved in two steps. First, the motion of each region of $P_m(t-1)$ is estimated. This motion describes the time evolution between frames at time $t-1$ and t. As in section 5-B, a polynomial motion model for each region is estimated by differential optimization techniques [3], [14]. Then, both partitions $P_g(t-1)$ and $P_m(t-1)$ are compensated using the technique described in [10]. The compensation is done on a pixel basis and can be viewed as a forward projection (from $t-1$ towards t). It defines three sets of areas: areas where the prediction is done without conflict, overlapping areas and uncovered areas. Overlapping areas correspond to pixels where more than one region is compensated whereas uncovered areas are areas where no region is compensated. These conflicting areas do not correspond to any region and are considered as uncertainty areas. Note that the compensation result cannot be considered as a real partition since a large number of non-connected uncertainty areas are present. Moreover, the compensation process does not guarantee to have only one connected component for each region. The reader is referred to [10] for more details about this compensation issue.

Examples of compensation can be seen in Fig. 14 and Fig. 15. These figures are made of three columns: the first one shows the partitions and modeled images at time $t-1$, the second one shows the compensated information and the last column shows the partitions at time t. Assume that partitions $P_g(t-1)$ (Fig. 14.A&.D) and $P_m(t-1)$ (Fig. 15.A&.A') are known. The compensation results are presented in Fig. 14.B&.E and in Fig. 15.B&.B'. Partitions are presented by assigning an arbitrary gray level value (label) to its pixels. In both cases, conflicting areas are displayed in black. As can be seen, the compensation defines quite precisely the main part of the

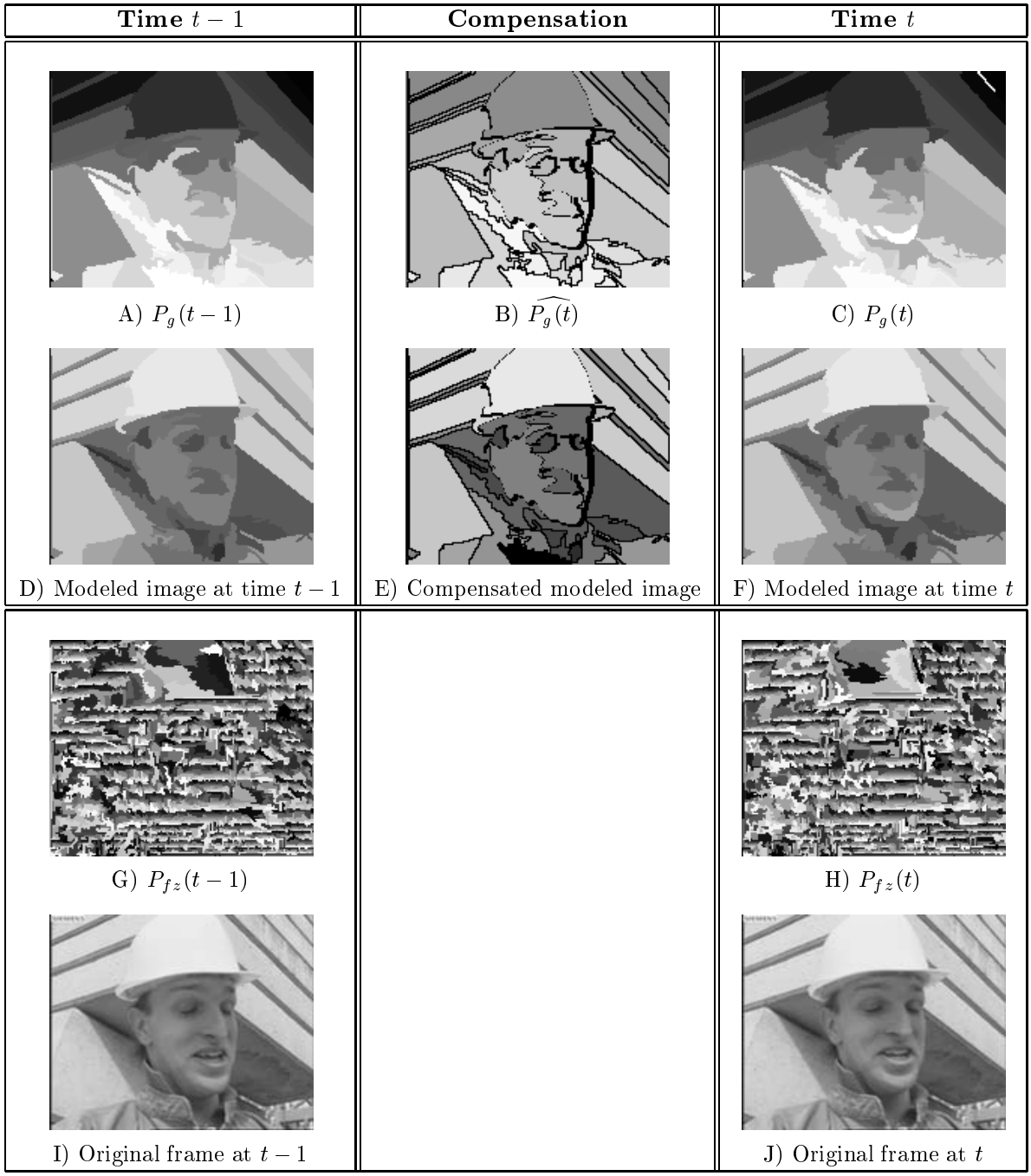


Fig. 14. Example of gray level sequence segmentation: first column: partitions and frames at time $t-1$, second column: compensated information, third column: partition and frames at time t. Partitions are represented with their label to see the region tracking ability of the algorithm. Flat zones partitions involve more than 10000 regions and partitions $P_g(t-1)$ or $P_g(t)$ involve about 40 regions.

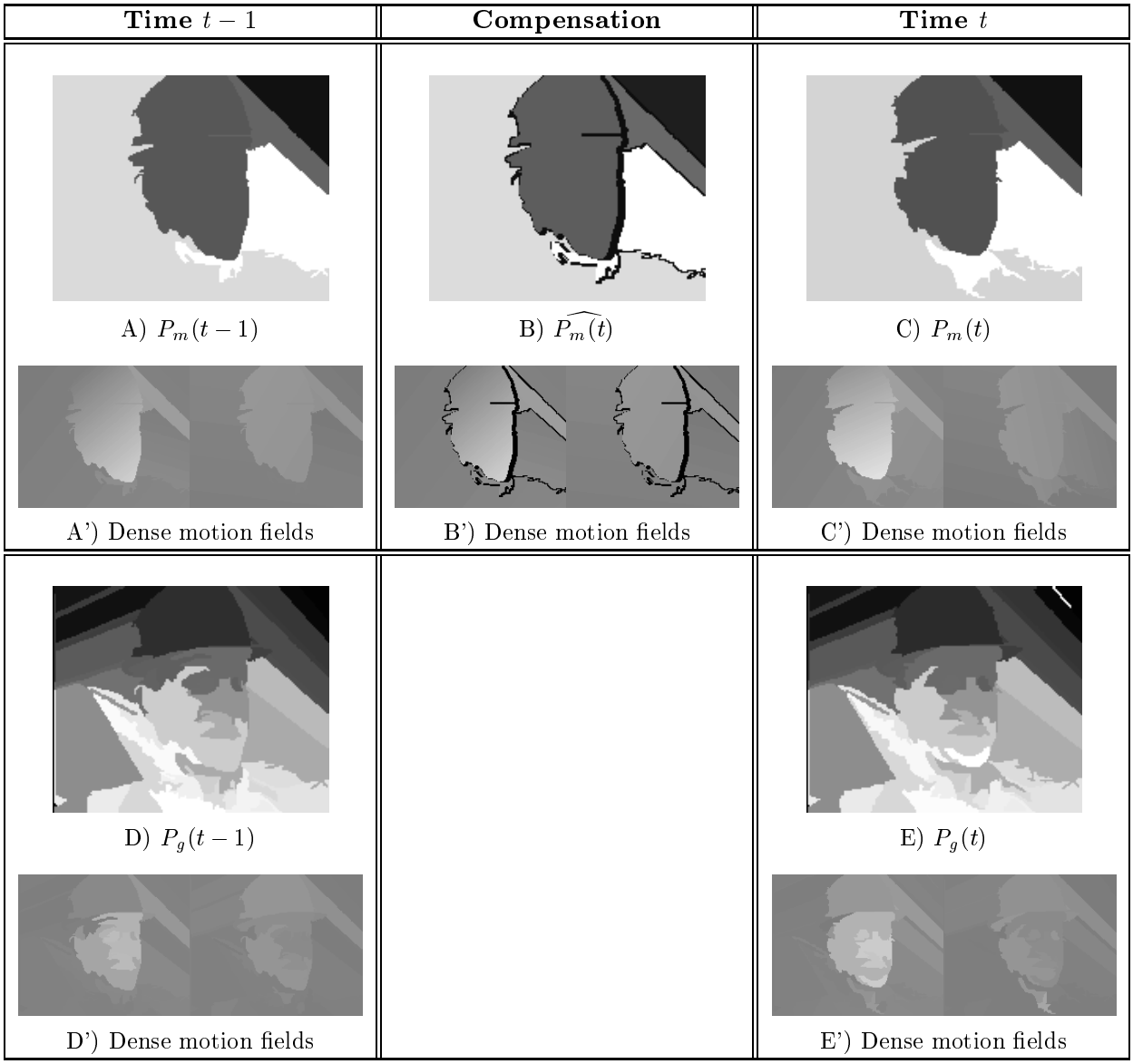


Fig. 15. Example of motion sequence segmentation: first column: partitions and frames at time $t-1$, second column: compensated information, third column: partition and frames at time t. Partitions are represented with their label to see the region tracking ability of the algorithm. Gray level partitions $Pg(t-1)$ or $P_g(t)$ involve about 40 regions and motion partitions $P_m(t-1)$ or $P_m(t)$ involve about 5 regions.

regions and conflicting areas appear at the transitions between regions. Finally, let us mention, that together with the partitions, the modeled gray level frames are also compensated (Fig. 14.E and 15.B'). These compensated frames will be necessary in the segmentation step.

5.3.2. Gray level segmentation

The goal of this segmentation step is to merge regions of the flat zone partition $P_{fz}(t)$ to create the gray partition $P_g(t)$ and to temporally link this partition with $P_g(t-1)$. To this end, two steps are used: first, a temporal link is created and, second, a partition is created. Both steps rely on the general merging

algorithm described in section 2. The main differences are the merging criteria and the way the labels (region numbers) are handled.

- Temporal link: This first step has some similarities with the work reported in [6]. Its goal is to define which regions of $P_g(t-1)$ are still present at time t and what are their approximate positions. To this end, the algorithm allows the merging of regions of $P_{fz}(t)$ that fall within the same compensated region. This step can be implemented as the merging of regions of $P_{fz}(t)$ with the following region model, merging order and criterion:
  - Region model: as previously, the model is the median (constant within the region). However, the processed image is a two-component image made of the current frame and of the compensated frame. When, two regions of $P_{fz}(t)$ are merged together, two models are updated: one for the current frame and one for the compensated frame.
  - Merging order: The order is the one defined by equation 6 modified to deal with the two components of the model: one order value is computed by taking into account the gray level values of the original frame at time t, and a second order value is computed with the values of the compensated frame. The final order is defined as the average of the orders defined by each model. This approach allows the introduction of some memory in the merging order computation.
  - Merging criterion: the criterion defines a termination point based on the PSNR as in the intra-frame segmentation described in section 4. The PSNR limit should be rather high in order to create reliable temporal links. However, the criterion is not only a termination criterion but also a decision criterion since only regions that fall within the same compensated region are allowed to merge. Therefore, for each possible merging, the compensated label corresponding to the two regions are analyzed and if the regions are not totally included in the same compensated region, the merging is not allowed. Moreover, regions that even partially fall within uncertainty areas of the compensation are not merged.

  At the end of the process, some regions of $P_{fz}(t)$ have merged. They can be considered as the core of the regions that were present at time $t-1$ and that are still present at time t. They will be identified by the label of their corresponding past region. Time tracking is therefore possible. The last processing step is to clean the resulting label image so that only one connected component is preserved for each label. In practice, when one label has several connected components, the largest one is preserved.
- Partition creation: Starting from the output of the temporal link, a partition can be created by continuing the merging process and relaxing the constraints imposed by the merging criterion. In fact, the processing is the same as the one used for intra-frame segmentation. The region model is the median (only the model of the current frame is necessary), the merging order is given by equation 6 and the merging criterion is simply a termination criterion defined by the PSNR. Note that in this last merging process, some regions are merged with regions defined during the temporal link. These regions allow the precise definition of the shape of regions that were present at time $t-1$. By contrast, some regions are merged together without involving any of the regions defined by the temporal link. These regions are new regions appearing at time t. They receive a new label.

As can be seen, the gray level segmentation process can be implemented as two separate merging steps or as a single one. In this last case, the merging order and criterion are modified during the merging process first, to create the temporal link and second, to define the final partition. The change is done when a given PSNR is obtained.

This gray level segmentation process is illustrated by Fig. 14. The original frame at time t and its corresponding flat zone partition are shown in Fig. 15.H&.J. This partition involves a very high number of regions (more than 10000, most of them with only one or two pixels). The gray level segmentation algorithm mer-

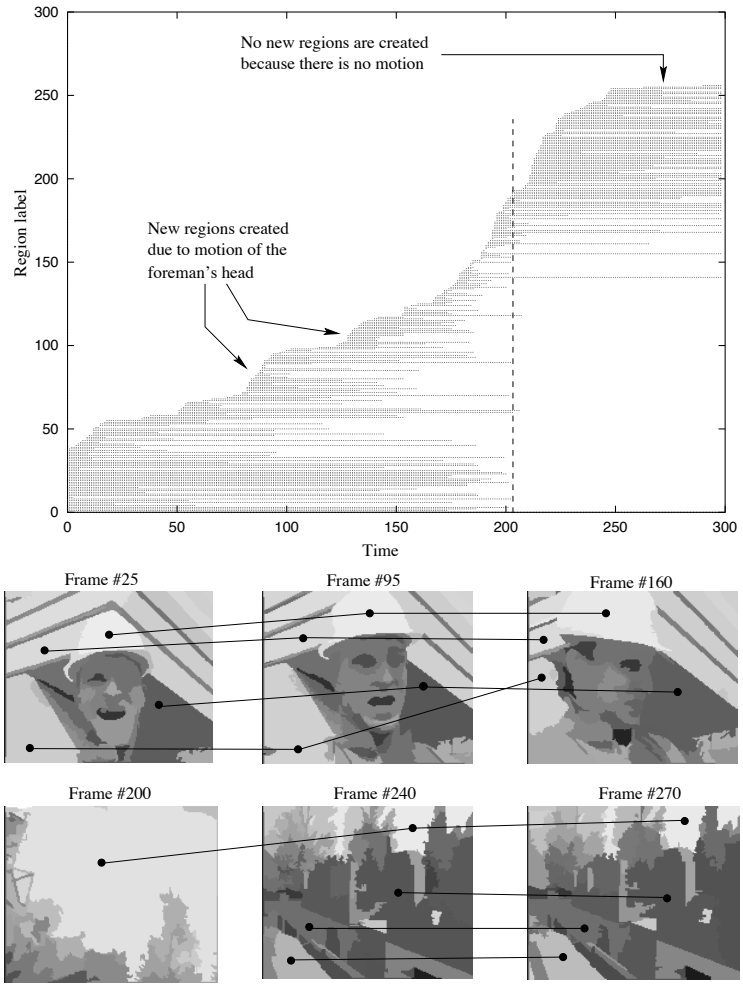


Fig. 16. Time evolution of regions in the Foreman sequence, upper part: Life span of regions, lower part: example of region correspondence.

ges the regions of the flat zone partition based on the information of the image at time t ($P_{fz}(t)$ and the original frame) and the compensated information ($\widehat{P_g(t)}$ and the compensated frame). The final result (after temporal link and partition creation) is shown in Fig. 14.C&.F. This result illustrates the time tracking ability of the algorithm. Partitions $P_g(t-1)$ and $P_g(t)$ involves about 40 regions. Most of the regions present at time $t-1$ are still present at time t. They can be easily identified because they have the same label. However, the algorithm has introduced some new regions displayed with a bright label (region around the chin, for example).

5.3.3. Motion estimation and segmentation

Once the gray level partition $P_g(t)$ has been created, a motion estimation is performed to assign a dense motion field to each region. The same motion estimation as the one previously discussed is used (see section 5-B for example) and the segmentation itself works as the gray level segmentation. The only difference is that the region model deals with the horizontal and vertical displacements. It is a first order polynomial to model a wide range of motions. Note that as in section 5-B, no motion re-estimation is performed.

This segmentation step is illustrated in Fig. 15. This figure has a similar layout than Fig. 14 but the

original images are two dense motion fields instead of gray level values (as before, the gray level value 128 represents a zero motion). The algorithm merges regions of partition $P_g(t)$ (Fig. 15.E) based on the information at time t and of the compensated information $\widehat{P_m(t)}$ (Fig. 15.B) and produces the final segmentation $P_m(t)$ (Fig. 15.C). This partition involves 8 regions (two for the face and six for the background). These regions are homogeneous in motion and are related to the corresponding regions at time $t-1$ (partition $P_m(t-1)$, Fig. 15.A). As in the gray level case, the algorithm is able to track, to introduce or to eliminate regions. Finally, as can be noticed, at each time instant, the set of partitions $P_{fz}(t)$, $P_g(t)$ and $P_m(t)$ is structured in a hierarchical way.

5.3.4. Analysis of the Time evolution of regions

This segmentation tool is particularly adequate to study the time evolution of the regions. Indeed, the time correspondence between regions of the gray level and motion partitions has been established. Therefore, it is possible to study when regions are created and when they disappear. An example of such a study done on the gray level partition is shown in Fig. 16 (obviously, the same study can be done on the motion partition). The upper part of the figure shows the time evolution of all regions created in the Foreman sequence. Each region is represented by a horizontal line. The region label defines the vertical position of the line and the life span of the region is defined by the beginning and end of the line. As can be seen, the algorithm is able to track regions over a large number of frames. Moreover, the rate of creation and elimination of regions is clearly related to specific events in the scene. For example, around frame #200, almost all past regions disappear and a large number of new regions are created. This behavior is typical of a change in the scene content. In the case of the Foreman sequence, this corresponds to the fast panning of the camera. Around frames 140 and 170, there is also an increase in the number of regions because of the presence of new content (uncovered part of the face and of the background). Finally, the lower part of the figure shows the time correspondence between some of the regions. This part of the figure illustrates the efficiency of the tracking.

6. Conclusions

This paper has focussed on the class of operators that are extensive in the lattice of partition. These operators are merging techniques that can be viewed either as filtering tools or as segmentation algorithms. The operator definition relies on three notions: first, the *merging order* that defines the notion of objects homogeneity, second, the *merging criterion* that characterizes the set of objects we are interested in and, third, the *region model* that define how objects are represented.

The efficient implementation of a merging operator relies on a good management of the distance value, that is the merging order, between each pair of neighboring regions. At each time instant, one should have, first, a very fast access to the next pair of regions to merge, and, second, an easy way to modify the various merging order values as the merging process goes on. The key element for a fast implementation is a dynamic hierarchical queue. A very efficient solution to this queue problem is to use a balanced binary tree. The resulting operators offer a processing (filtering or segmentation) time of about 1,5 second for QCIF images on 200MHz Pentium.

Using the same philosophy and implementation, new filtering and segmentation tools can be created. Self-dual connected operators based on area and on contrast have been proposed and illustrated. These operators are particularly useful as pre-processing. For the segmentation aspect, several algorithms have been described. They can be used to segment images or sequences with either a spatial or a motion criterion. They can also be used to recursively segment a sequence and to track its regions in time. Finally, let us mention that the approach is particularly attractive to create sets of hierarchical partitions. The hierarchy can be homogeneous dealing either with a gray level criterion or with a motion criterion. However, heterogeneous hierarchies can also be created, for example involving spatial partitions on lower levels and motion partitions on upper levels.

References

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

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

[3] J. L. Dugelay and H. Sanson. Differential methods for the identification of 2D and 3D motion models in image sequences. *Signal Processing, Image Communication*, 7:105–127, 1995.

[4] 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.

[5] D. Knuth. *Sorting and searching, in "The Art of Computer Programming"*. Add.-Wesley, 73.

[6] B. Marcotegui. Segmentation algorithm by multicriteria region merging. In P. Maragos, R.W. Schafer, and M.A. Butt, editors, *Third workshop on Mathematical morphology and its applications to image processing*, pages 313–320, Atlanta, USA, May 1996. Kluwer Academic Publishers.

[7] B. Marcotegui, J. Crespo, and F. Meyer. Morphological segmentation using texture and coding cost. In I. Pitas, editor, *1995 IEEE workshop on Nonlinear Signal and Image Processing*, pages 246–249, Halkidiki, Greece, June 20-22 1995.

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

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

[10] P. Salembier, F. Marqués, and A. Gasull. Coding of partition sequences. In L. Torres and M. Kunt, editors, *Video Coding: The Second Generation Approach*. Kluwer, 1996. ISBN: 0 7923 9680 4.

[11] P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, To be published, 1998.

[12] 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.

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

[14] H. Sanson. Toward a robust parametric identification of motion on regions of arbitrary shape by non-linear optimization. In *Proceedings of IEEE International Conference on Image Processing, ICIP'95*, volume I, pages 203–206, October 1995.

[15] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.

[16] J. Serra. *Image Analysis and Mathematical Morphology, Vol II: Theoretical advances*. Academic Press, 1988.

[17] 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.

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

[19] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE, Transactions on Pattern Analyis and Machine Intelligence*, 39(12):1845–1855, December 1991.

[20] N. Wirth. *Algorithms & Data Structures*. Prentice-Hall, 1986.