IMAGE SEQUENCE ANALYSIS AND MERGING ALGORITHMS

Philippe Salembier, Luis Garrido, David Garcia

Universitat Politècnica de Catalunya
Campus Nord, Modulo D5
C. Gran Capità, sn
08950 Barcelona, SPAIN
E-mail: philippe@gps.tsc.upc.es

ABSTRACT

This paper deals with merging techniques that can be used as filters 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 issues of a merging algorithm: merging order, merging criterion and region model. This analysis highlights the similarity and differences between a filtering tool 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 auto-dual operators) and efficient segmentation algorithm (robust criteria and efficient implementation).

1. INTRODUCTION

Image sequence analysis is becoming a very active field of research because of its large set of potential applications. Indeed, the efficient use of an MPEG-4 coder or services involving large video databases 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 can be considered as "sequence analysis" tools. In this paper, we focus on tools that combine what is traditionally seen either as "filtering" or as "segmentation". They aim at the definition of regions based on some homogeneity criteria (segmentation aspect) possibly with some constraints (filtering notion).

Filters are generally used as pre-processing to remove part of the image content such as noise or some details. For a segmentation application, these filters should simplify the image but preserve the shape of the remaining objects. Examples of filters possessing this property are known as *connected operators* [10, 2, 9]. They do not introduce any new contour because they can only merge the zones of the space where the signal is constant (called *flat zones*). These filters are extensively used in segmentation applications [15, 14, 11].

A large number of popular segmentation approaches also relies on merging strategies. They, first, define a set of initial regions and, second, progressively merge regions to create a partition of the image. In the *Split&Merge* [4] algorithm, the initial regions are defined by the Split process and the merging is performed between the initial regions themselves. The *Region growing* algorithm [1] merges initial regions (sometimes called seeds) with individual neighboring pixels that belong to an uncertainty area. Finally, the classical morphological tool for segmentation, *watershed* [7], also relies on a merging strategy: the initial regions are the regional gradient minima, and these minima are progressively expanded by merging of neighboring pixels in an order defined by the gradient value.

This work has been supported by France-Telecom/CCETT under the contract 96ME22

Connected operators, Split&Merge, Region growing and the watershed algorithms have been created in different context and for different applications. 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. As major output of this study, we will create 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. Section 2 defines a general merging strategy. An efficient implementation of the merging process is proposed in section 3. Based on the general merging process, section 4 presents 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. GENERAL MERGING ALGORITHM

In this section we propose a general merging strategy. The 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. A merging algorithm on this graph is simply a technique that removes some 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 $\mathcal{O}(R_1, R_2)$ 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 $\mathcal{C}(R_1, R_2)$ 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. Let us denote by $\mathcal{M}(R)$ this model.

Note that this merging strategy allows the implementation of both region growing algorithm and connected operators. In the case of a region growing algorithm, the merging order is defined by the similarity measure between two regions, the merging

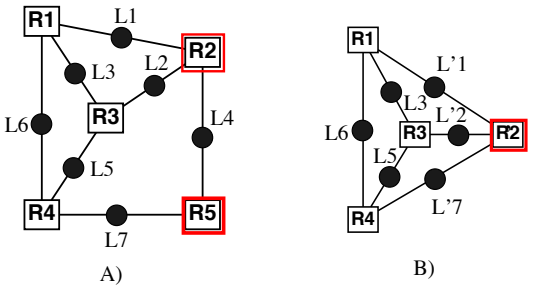


Figure 1: Region Adjacency Graph: A) Original RAG. B) RAG after merging of R_2 and R_5

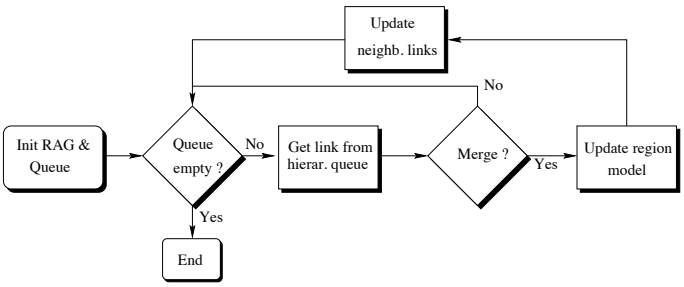


Figure 2: Block diagram of the general region merging algorithm

criterion always states that two regions have to be merged until a termination criterion is reached (for example, a minimum number of region or a quality criterion) and the model defines a gray level function, for example the mean. In the case of a connected operator such as an opening by reconstruction or an area opening (see [15, 14] for example), the merging order is simply defined by the absolute gray level value, the merging criterion relies on the assessment of a criterion (for example, the number of pixels of each region in the case of an area opening) and each region is modeled, after merging, by the lowest gray level value.

The merging order 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 case of connected operators, objects are assumed to be either bright or dark parts of the image. The main difference between a filter and a segmentation algorithm relies on the merging criterion. In the case of a segmentation algorithm, we would like to extract 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.

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 key points of the implementation is a hierarchical queue able to deal with floating point priorities and the "link node" of a Region Adjacency Graph structure (RAG). The RAG structure is depicted in Fig. 1A. The classical RAG is a graph where each node represents a region of the image and where each node is connected to its neighbors. The graph structure used here is similar but contains some more information. The white nodes in Fig. 1A, 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_5 are merged (see Fig. 1B). The link to remove is L_4, and the link-nodes whose ordering has to be updated are L_1, L_2 and L_7.

In Fig. 2, the general scheme of the merging process is represented. 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. The hierarchical queue is used to order the various links and have a fast access to the link of highest priority. The ordering (represented by a floating point number) defines the insertion point of link-node into the hierarchical queue.

Once the first stage is completed, the merging process begins. Observe that the merging algorithm is 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 the previous section, 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 (in Fig 1, the link L_4 has the highest priority and regions R_2 and R_5 are merged). As a result, the region model has to be updated (creating region $R'_2 = R_2 \cup R_5$). 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 (in Fig 1, neighboring links of L_4 are L_1, L_2 and L_7). This implies the extraction of the corresponding links from the queue, the computation of the new priority (using the models of neighboring models) and the insertion of the links into the queue with their new priority (links L'_1, L'_2 and L'_7 in Fig 1.B). 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 or of the watershed algorithm [15]. 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, sec-

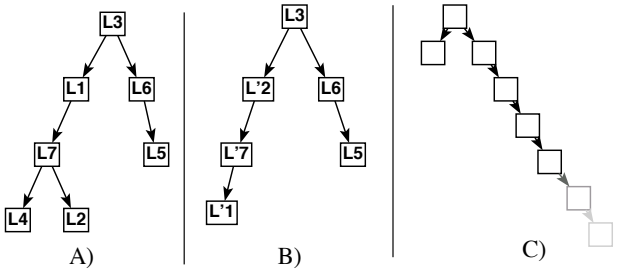


Figure 3: Hierarchical queues and binary trees: A) Example of tree corresponding to Fig 1.A; B) Example of tree corresponding to Fig 1.B; C) Example of unbalanced tree

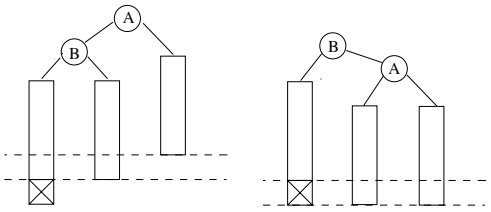


Figure 4: Example of binary tree balancing. Left: Unbalancing due to insertion. Right: Restoring the balance

ond, 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. For our merging algorithm, the hierarchical queue should be updated and re-organized on line. It can be seen as a dynamic hierarchical queue. The solution proposed here is based on a binary tree [5]. 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 of a link node of the RAG. A right child node is characterized by having a priority lower than its father, while a left child node has a priority greater than its father. At each node, the list of link-nodes having the same priority are stored in a FIFO structure. In the example of Fig. 3.A, the order of the links is: $L_4 < L_7 < L_2 < L_1 < L_3 < L_6 < L_5$.

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 left branches until a node with no letf branch is found. Search, insertion and deletion of nodes in the queue can be done efficiently ($O(log_2 N)$ steps where N is the number of tree nodes). One of the drawbacks of using binary trees to implement hierarchical queues is that the tree may degenerate. This problem happens when 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 right (see Fig. 3.C). In this case, the $O(log_2 N)$ efficiency does not hold anymore, because the tree is becoming a simple FIFO queue. The solution to this problem is to "balanced" the tree. The method [5, 16] 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. Efficient implementations of insertion and deletion using balancing techniques can be found in [16].

The efficiency of the merging 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. SEQUENCE SIMPLIFICATION TOOLS

In this section, we propose new connected operators that can be used either to analyze a sequence or as pre-processing before a segmentation algorithm. As discussed in the introduction, connected operators are region-based simplification tools but they generally 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 auto-dual in the gray level sense. To this end, one has to define the merging order $\mathcal{O}(R_1, R_2)$, the merging criterion $\mathcal{C}(R_1, R_2)$, and the region model $\mathcal{M}_R(x)$ in such a way that the resulting operator be auto-dual.

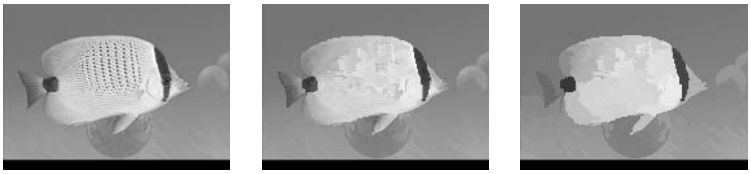
1. **Region model, $\mathcal{M}_R$:** The first choice that has to be made is how each region is going to be modeled. Two simple auto-dual models are the mean and the median. From our practical experience, the median is more robust than the mean. To allow a fast implementation, the median of region $R = R_1 \cup R_2$ should be computed recursively from the median of the two merged regions R_1 and R_2. Here, one has simply to select the model of the largest region [1]:

$$\begin{aligned} &\text{if } N_1 < N_2 \quad \Rightarrow \quad \mathcal{M}_R = \mathcal{M}_{R_2} \\ &\text{if } N_1 > N_2 \quad \Rightarrow \quad \mathcal{M}_R = \mathcal{M}_{R_2} \\ &\text{if } N_1 = N_2 \quad \Rightarrow \quad \mathcal{M}_R = (\mathcal{M}_{R_1} + \mathcal{M}_{R_2})/2 \end{aligned} \tag{1}$$

   where N_1 and N_2 denote the numbers of pixel of each region. 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, $\mathcal{M}_R$ is a multidimensional model and each component is modeled independently. Finally, let us mention, that the median model is constant within the region. It can be considered as simple zero order model. However, in section 5, the model will be extended to a first order model in order to deal motion homogeneity.

2. **Merging order, $\mathcal{O}(R_1, R_2)$:** The merging order defines the notion of objects. It can be seen as a measure of the likelihood that two neighboring regions belong to the same object. Let us assume that we deal with gray level images. A fairly high number of order have been test and

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



A) Original frame
10719 flat zones

B) Area: 10 pix.
574 flat zones

C) Area: 100 pix.
80 flat zones

Figure 5: Area operator

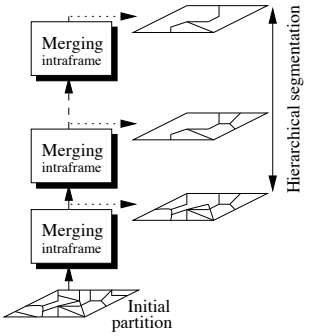


Figure 6: Hierarchical segmentation by merging

a good compromise between region extraction and contour definition can be obtained with:

$$\mathcal{O}(R_1, R_2) = N_1(\mathcal{M}_{R_1} - \mathcal{M}_R)^2 + N_2(\mathcal{M}_{R_2} - \mathcal{M}_R)^2 \quad (2)$$

Note that, in the case of the median model and if R_1 is smaller than R_2, the model after merging is $\mathcal{M}_{R_2}$ and the order reduces to $\mathcal{O}(R_1, R_2) = N_1(\mathcal{M}_{R_1} - \mathcal{M}_{R_2})^2$. 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.

3. Merging criterion: $\mathcal{C}(R_1, R_2)$: The function $\mathcal{O}(R_1, R_2)$ 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 regions that fulfill a given criterion. This function allows the definition of filtering, that is sieving, tools.

   As an example, let us discus the *area* criterion. Its goal 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). The effect of the operator can be seen in Fig. 5. Fig. 5.B and C show the simplification effect when the area is set to either 10 pixels or 100 pixels. These images contain most of the original information (contour are precisely and sharply defined) except small regions. For example, the texture of the fish has been removed with a simplification of size 10 and a large number of small regions have disappeared with a simplification of 100. In this last case, the image is made of only 80 flat zones. These images can be used as starting point for a segmentation algorithm or as pre-segmentation for motion estimation on regions of reasonable size. 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.

5. IMAGE AND SEQUENCE SEGMENTATION

5.1. Region model, merging order and 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 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 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 distribution) 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: $\mathcal{M}_R(i,j) = \alpha i + \beta j + \gamma$, if i, j denote the spatial coordinates of the pixels. When two regions are merged, this model can be updated by a median rule. The resulting model is equal to the model of the largest region. For of sequence analysis, first order polynomial models are particularly useful to represent dense motion fields.

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 2 is used.

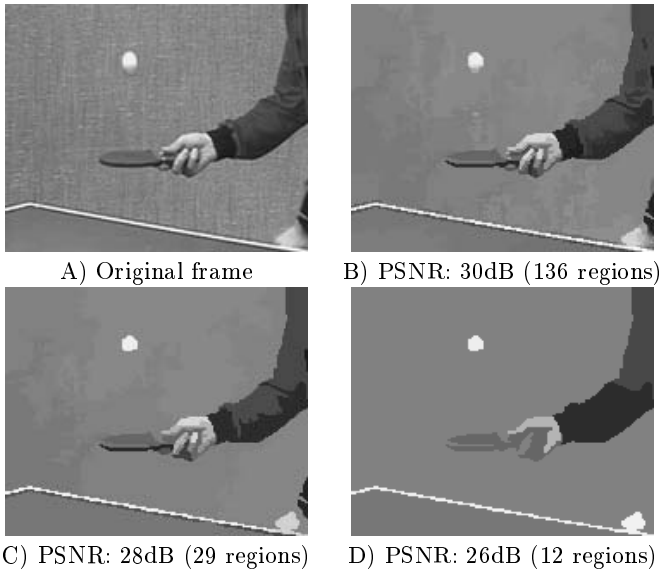
5.2. Intra-frame segmentation

For intra-frame 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 and modeled images,
- the number of regions.

The merging strategy defined in section 2 is particularly useful to compute a set of hierarchical partitions. The approach is illustrated in Fig. 6. Starting from an initial partition, which can be either the finest partition of the space (partition where each individual pixel is a region) or the partition of flat zones, several merging steps are performed. With the proposed implementation, 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.

An example of hierarchical segmentation is shown in Fig. 7. The segmentation creates regions that are homogeneous in gray levels. In Fig. 7, we show images where each region has been represented by its model (median in this case). Three partitions corresponding to PSNR values ranging from 30 to 26 dB are



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

C) PSNR: 28dB (29 regions) D) PSNR: 26dB (12 regions)

Figure 7: Example of gray level intra-frame segmentation (termination criterion: PSNR)

Original gray level partition
83 Regions

Motion partition
5 Regions

Figure 8: Example of motion segmentation. Left: Initial gray level partition, Right: Motion partition

presented. As can be seen, these partitions provide simple approximations of the original frame in terms of number of regions but a good definition of the main objects and their contour. 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, 12]. 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 which has to deal with a two component image. As a result, the algorithm defines regions that are homogeneous in the sense of motion. The result of the segmentation is shown in Fig. 8. The final segmentation 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

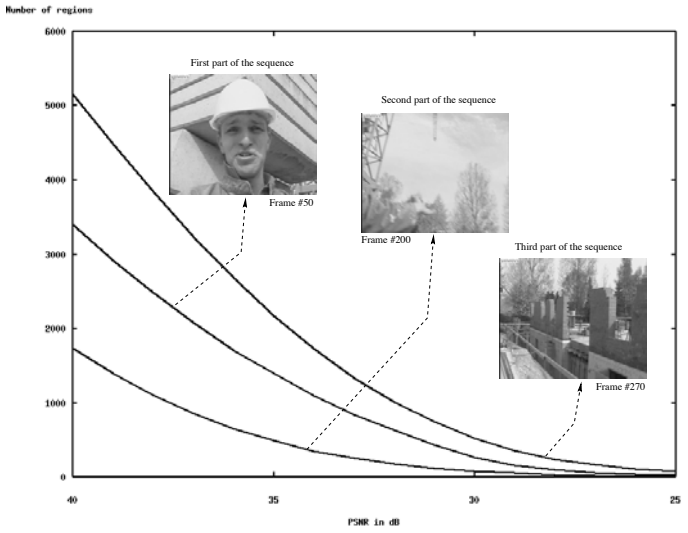


Figure 9: Number of regions as a function of the PSNR for three typical frames the Foreman sequence

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* [13]. The approach consists in using a hierarchical filtering structure as the one presented in Fig. 6 and in measuring an image characteristic 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 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. 6 and let us suppose that we measure the number of regions of each resulting partition. The results are shown in Fig. 9. 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 low. By contrast, if the image involves a large number of contrasted objects, a high number of regions are necessary to reach the same PSNR. This intuitive discussion is confirmed by Fig. 9: 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.

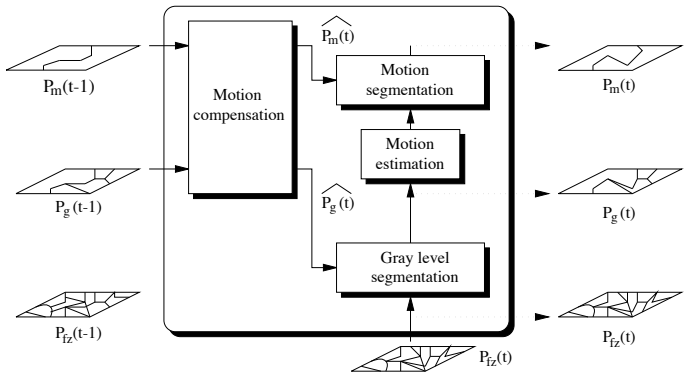


Figure 10: Sequence segmentation algorithm. Hierarchy of partition: flat zones P_{fz}, gray level P_g and motion P_m

5.3. Sequence segmentation

In this section, we focus on a more complex segmentation scheme combining several homogeneity criteria. 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) 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.

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. 10. 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 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 forward motion of each region of $P_m(t-1)$ is estimated. This motion describes the time evolution between frames at time t and $t-1$. As in section 5.2, a polynomial motion model for each region is estimated by differential optimization techniques [3, 12]. Then, both partitions $P_g(t-1)$ and $P_m(t-1)$ are compensated using the technique described in [8]. 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.

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 position. 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 components 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 2 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 are 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.

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. 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.

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 5.2 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 5.2, no motion re-estimation is performed.

The segmentation step is illustrated in Fig. 11. This figure shows the set of partitions at time $t-1$ (first column), the compensated partitions (middle column) and the partitions at time t (right column). The flat zone partitions involve more than 10000 regions, the gray level partitions involve about 50 regions and the motion partitions has 8 regions. Note that most regions present at time $t-1$ are also present at time t. They can be easily identified because they have the same label. 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 hierarchical way.

6. CONCLUSIONS

This paper has focused on a class of merging techniques which 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.

Using the same philosophy and implementation, new filtering and segmentation tools can be proposed. New connected operators such as the area operator can be created. 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.

7. 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"*. Addison-Wesley, 1973.

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

[8] 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.

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

[10] 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.

[11] 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.

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

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

[14] 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.

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

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

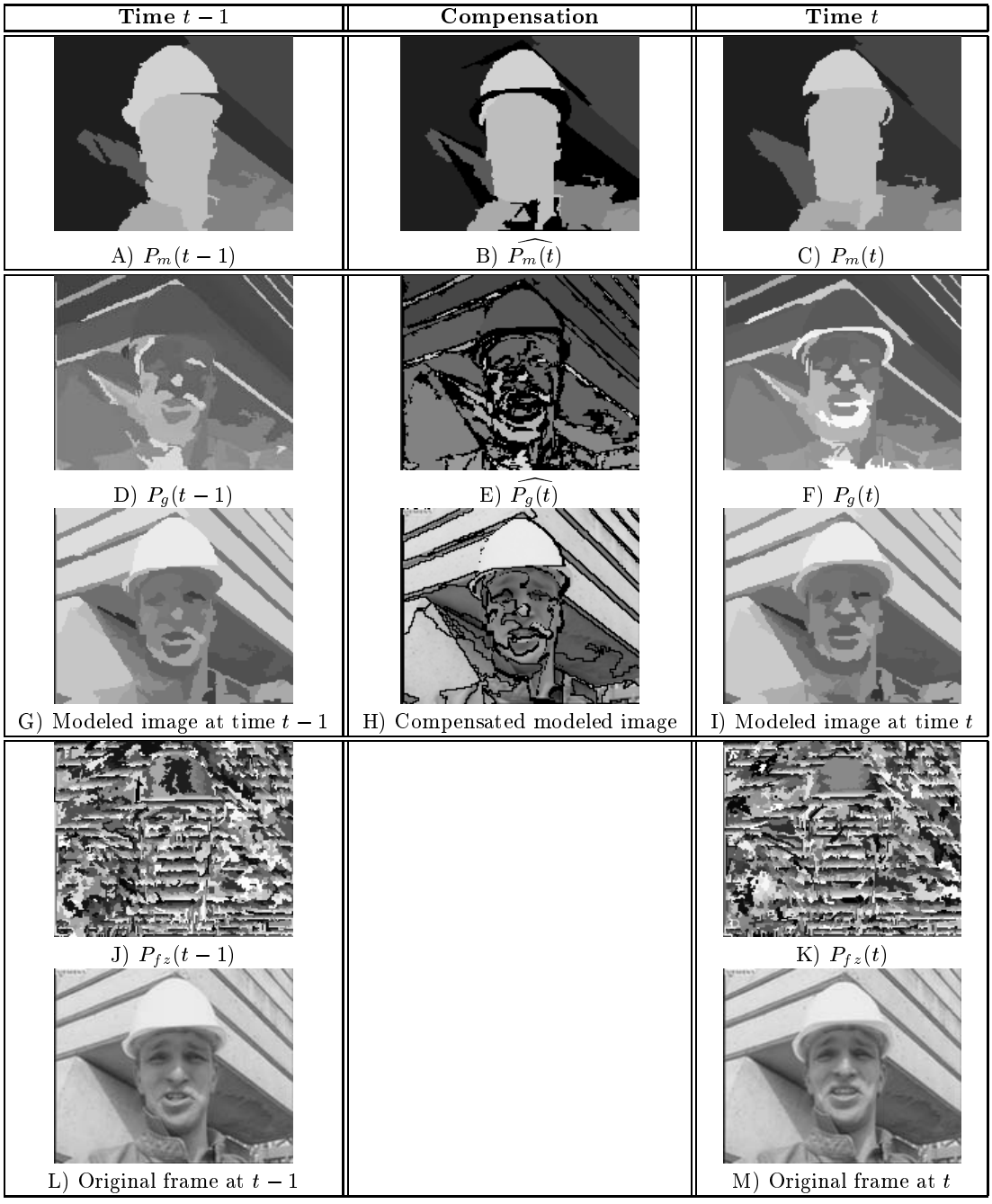


Figure 11: Example of gray level sequence segmentation: first column: partitions and frames at time $t-1$, second column: compensated information, third column: partition and frames at time t. Partitions are represented with their label to see the region tracking ability of the algorithm.