Region-Based Video Coding Using Mathematical Morphology

PHILIPPE SALEMBIER, LUIS TORRES, FERNAND MEYER, AND CHUANG GU

Invited Paper

This paper presents a region-based coding algorithm for video sequences. The coding approach involves a time-recursive segmentation relying on the pixels homogeneity, a region-based motion estimation, and motion compensated contour and texture coding. This algorithm is mainly devoted to very low bit rate video coding applications. One of the important features of the approach is that no assumption is made about the sequence content. Moreover, the algorithm structure leads to a scalable coding process giving various levels of quality and bit rates. The coding as well as the segmentation are controlled to regulate the bit stream. Finally, the interest of morphological tools in the context of region-based coding is extensively reviewed.

I. INTRODUCTION

In the framework of very low bit rate video coding, there is an increasing interest in second generation image compression techniques [4]. These techniques eliminate the redundant information within and between frames, taking advantage of the properties of the human visual system. In particular, region-based compression methods describe the images in terms of a set of regions, that is a partition, and of some information for each region to be used by the receiver to reconstruct the image. These techniques have been successfully applied to the coding of still images [4]. For sequences, region-based schemes have been developed in particular in [10], [12], [23].

The main difference between the techniques proposed in [10], [12], [23] can be described by the relative importance they assign to the spatial or the motion information. Reference [10] proposes a coding scheme where motion plays the central role and the image is restored on the receiver side by motion compensation of past restored frames. A partition of each frame is used mainly to define the regions that should be compensated. This approach leads to good results if the sequence can actually be compensated. That is, if no new objects are allowed to be introduced in the scene and if scene changes are prohibited. As a result this technique is mainly dedicated to very specific applications such as "head and shoulders" sequences. The approaches described in [12], [23] are more general in the sense that they mainly deal with the spatial information of the scene. The sequence is considered as a 3D signal (two spatial plus one temporal dimensions) and 3D blocks are segmented on the basis of the gray-level information. Finally, the 3D partition is coded as well as the texture of each region. Note that the system can *a priori* code any sequence and no assumptions have to be made about the introduction of new objects or about scene changes. However, in this case, motion information is not used and the coding efficiency remains moderate because the temporal redundancy is not really exploited.

In this paper, we propose a coding algorithm which combines a spatial analysis of the sequence with a motion compensation of the transmitted information. On the one hand, the spatial analysis is used to get a general scheme able to deal with any kind of sequences and scene changes. On the other hand, motion information is used to increase the coding efficiency by compensation of the spatial information that has to be transmitted (partition and texture).

From the information theory viewpoint, coding generally relies on the statistical properties of signals. However, in a region-based approach, the geometrical information is of prime importance and should be used as much as possible. Classical linear signal processing tools are not well suited for a geometrical approach and other tools coming from nonlinear signal processing or from computer vision may be attractive for this purpose. Mathematical morphology [19], [20] has been developed as a geometrical approach to signal processing. Our objective in this paper is twofold: First, to describe a region-based video coding algorithm

Manuscript received July 1, 1994; revised January 17, 1995. This work was supported by the MORPHECO project under the European RACE program.

P. Salembier and L. Torres are with the Department of Signal Theory and Communications of the Polytechnic University of Catalonia (UPC), Barcelona, Spain.

F. Meyer is with the Center for Mathematical Morphology, Paris School of Mines (CMM), Fontainebleau, France.

C. Gu is with the Signal Processing Laboratory of the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.

IEEE Log Number 9410572.

and, second, to discuss the usefulness of morphological tools in this context.

The organization of this paper is as follows: Section II describes the general codec structure and its strategy. Four steps are highlighted: Segmentation, motion estimation, contour coding, and texture coding. These steps are respectively discussed in Sections III–VI. Finally, Section VII is devoted to the presentation of some results of very low bit rate video coding. Through out the paper, the interest of using nonlinear signal processing tools such as mathematical morphology is discussed. The appendix briefly defines and comments on the major morphological tools that are mentioned in the paper.

II. Structure of the Algorithm

In order to define the algorithm structure, let us start with a review of the basic system requirements.

- The codec is mainly devoted to very low video bit rates, that is below 64 kb/s.
- It should be generic with respect to the input sequences. In particular, no assumption about the scene content should be made.
- It should be flexible with respect to the coded sequence quality. The coding strategy should provide an easy way to define various levels of quality.
- The system is principally devoted to fixed rate transmission. As a consequence, the bit stream should be regulated.
- It should be suitable for interactive applications. Therefore, large processing delays should be avoided.

To design such a system, we will follow one of the *second-generation* coding ideas which states that, as the compression ratio increases (as for very low bit rates), the most vital part of the image information is represented by contours [4]. This point of view leads to a two-component image representation: contour/texture. As a result, the encoder involves a segmentation step, which defines a partition of the image, followed by a coding step, which codes the contour of the partition and the gray-level or color information of each region (called *texture* in the following).

As mentioned in the introduction, motion compensation is one of the best available techniques to take benefit of the temporal redundancy. Therefore, motion information will be used for contour as well as for texture coding. The segmentation criterion will not deal with the motion field but with the pixels' homogeneity in the spatial domain. Motion will not be used as segmentation criterion, because we are interested in a spatial analysis of the signal to be able to deal with any kind of sequence.

This analysis leads to the general coding structure described in Fig. 1. It involves four steps: Segmentation, motion estimation, coding of contours, and coding of texture. The segmentation defines the partition of the sequence. The motion estimation is performed after the segmentation and can therefore be region-based. Finally, contour and texture are coded by motion compensation techniques. Note

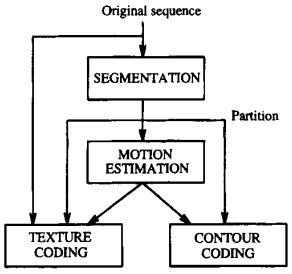


Fig. 1. General structure of the encoder.

that the texture coding makes use of the partition. It is a region-based texture coding approach.

III. Segmentation

With the assumptions made in the previous section, a good segmentation should extract the visually important regions of the scene, be coherent in time, avoid as much as possible random fluctuations of the contours and solve the region correspondence problem. This last requirement means that one should be able to follow the time evolution of a given region (a given object). With these requirements, it is very difficult to obtain a good segmentation if the processing is only 2D or intraframe. Indeed, if frames are segmented independently, both the time coherence and the region correspondence problems are difficult to solve. Somehow, the temporal relationship between frames must be exploited.

In this direction, a first solution consists in dealing with the sequence as a 3D (2D plus time) signal and in performing a 3D segmentation. This approach implies to split the sequence into 3D blocks of a given number of frames and to segment these 3D blocks. Examples of this approach can be found in [17], [12], [23]. However, these techniques have some drawbacks. Indeed, if the temporal size of the 3D blocks is large, the approach implies huge memory requirements, a high computational load and the introduction of a large processing delay disallowing any interactive application. Besides, if the temporal size of the 3D blocks is small, the temporal correlation between frames will be ignored at each block transition, and that is very often.

However, most of the information contained in a frame is already present in the previous frame. The changes are mainly due to object motion and the appearance of new objects. This point of view leads to a segmentation process that can be called time-recursive [11]. The segmentation approach involves two modes of operation: intraframe and interframe segmentation. During the intraframe mode, a frame is segmented. It is a purely 2D process. Then, during the interframe mode, each frame is recursively segmented using the segmentation of the previous frame. Note that in the following, the segmentation is assumed to be performed on the luminance signal because most of the visually important transitions are defined by the luminance signal.

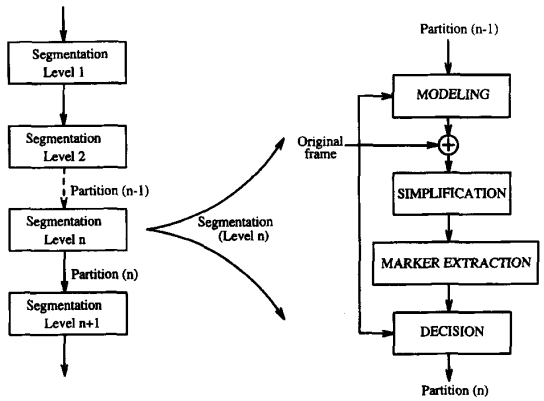


Fig. 2. Intraframe mode of segmentation.

A. Intraframe Mode

The intraframe segmentation relies on the hierarchical method proposed for still images in [14]. The algorithm first produces a simplified segmentation in the sense that it involves a reduced number of regions. Then, the segmentation is progressively improved by introducing more regions. Typically four segmentation levels are used. Each segmentation level involves four basic steps shown in Fig. 2: Modeling, simplification, marker extraction, and decision.

- *Modeling*: Assume that, at level $n-1$, a current estimation of the partition is known (at the very first level, the entire frame is considered as a single region). Then, the next hierarchical segmentation (level n) will have to improve this estimation dealing with the regions that cannot be well represented by the coding process. To have information about these regions, each region is actually coded. Ideally, the real coding algorithm should be used to code the contours and the texture. However, most of the time, a simplified version of the coding can be used. In the following, only a texture coding technique will be used (that is, the contour coding process is assumed to be lossless). In practice, each region is filled with a gray-level function which approximates the coded version of the region the receiver will have. Then, the difference between the coded image and the original one, called the *modeling residue*, is computed. Since this residue concentrates all the information about the poorly coded regions, it is used as the signal to be segmented in the following steps.
- *Simplification*: In this step, images are simplified to make them easier to segment. The simplification controls the *nature* and *amount* of information that is kept for segmentation at this level of the hierarchy. Different simplification tools can be used depending on the segmentation criterion. For coding applications, several visually important criteria can be used: size, contrast [14], [13], or dynamics [8]. A "size" segmentation means that all regions larger than a given limit are segmented. Using a "contrast" or a "dynamic" criterion, regions of high contrast are extracted. In practice, a combination of size and contrast criteria are used because, most of the image structure is represented by large regions but an important part of the meaningful information is defined by small but contrasted details.
  As shown in [14], morphological *filters by reconstruction* are very efficient for size simplification (see the appendix for the definition of this morphological filter). Morphological h-maxima or h-minima operators are particularly suitable for contrast simplification [13]. These morphological operators belong to the class of so-called *connected operators* [21], [18]. They are very attractive for segmentation purposes because they simplify the signal by removing small (*filters by reconstruction*) or poorly contrasted (h-max/min) regions *without corrupting the contour information*. Moreover, they interact with the signal by producing and merging flat zones (zones of constant gray level value). See the appendix for more information about connected operators. The concept of flat zones is very useful for the marker extraction step.
- *Marker extraction*: The goal of this step is to detect the presence of homogeneous regions. It produces markers identifying the interior of the regions that will be segmented. In practice, a marker is a connected component of the image with a specific gray level value indicating the number of the region. The marker defines the set of pixels which surely belong to the region and, in general, it defines the major part of the region interior.
  The marker extraction technique depends on the segmentation criterion. As explained in [13], for size-oriented segmentation, the marker extraction consists in labeling the interior of large flat zones after the simplification. This can be easily done thanks to the simplification process which has produced flat zones. For contrast-oriented segmentation, the marker extraction relies on the labeling of the extremal flat zones produced by the h-maxima/minima operators [13].
- *Decision*: After marker extraction, the number and the interior of the regions to be segmented are known. However, a large number of pixels are not assigned to any region. These pixels correspond to uncertainty areas mainly concentrated around the contours of the regions. Assigning these pixels to a given region can be viewed as a decision process that precisely defines the partition. The classical morphological decision tool is the *watershed* [9]. It is generally used on the morphological gradient of the image to segment. However, as discussed in [14] the use of the morphological gradient results in a loss of information on the contour position of $\pm$ 1 pixel which is generally too high for coding applications. In the case of still image coding and depending on the application, one may consider that it is not a serious problem. However for image sequences, the loss of information about the contour

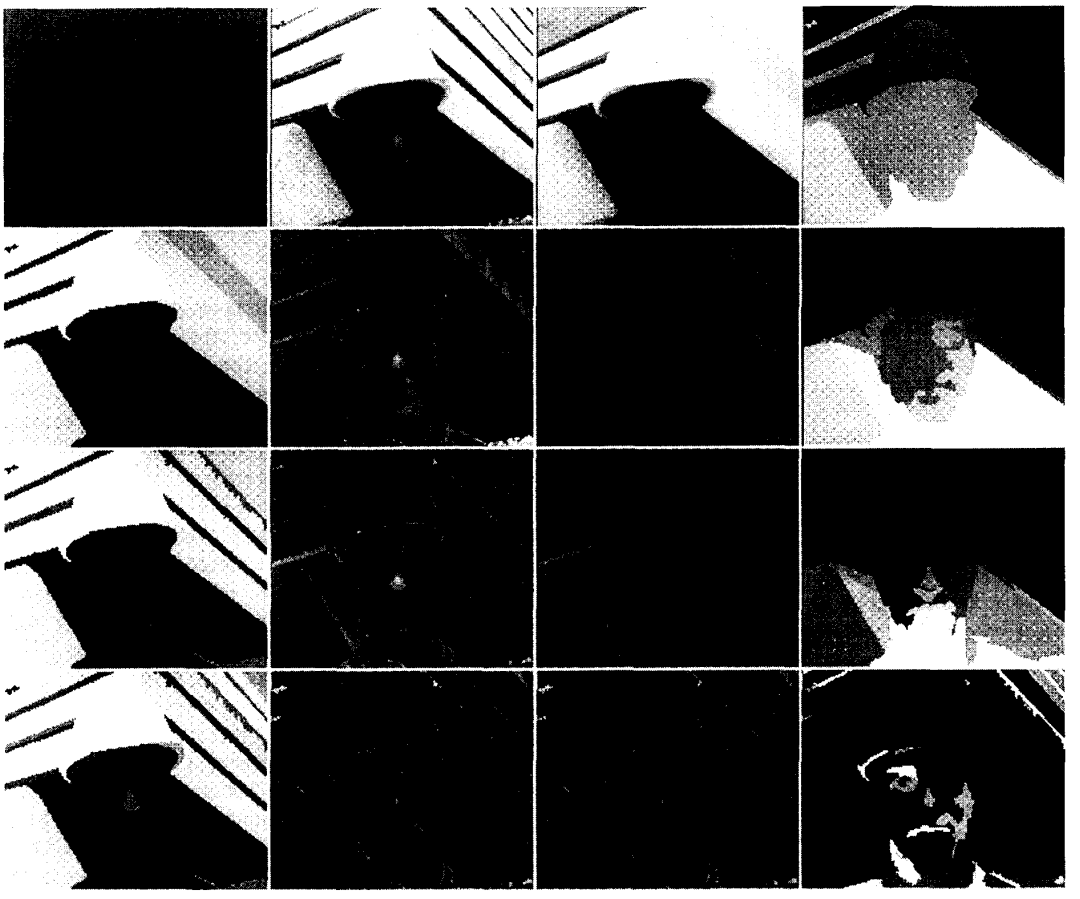


Fig. 3. Example of intraframe segmentation. Each row represents a given segmentation level with a size criterion for the three first rows (size: 671, 219, 94 pixels) and a contrast criterion for the last one (contrast: 25). The four columns give respectively the modeled image, the residue, the simplified error and the partition.

position depends on the region motion and can become very large. Therefore, the use of the gradient should be avoided [17].

To solve this problem, it has been proposed to work on interpolated signals in [14] or to use a different version of the *watershed* algorithm working on the original signal in [17], [13]. The idea of using the *watershed* algorithm directly on the signal to segment was first proposed in [7] to deal with color images. The resulting algorithm is a region growing process: the set of markers is extended until they occupy all the available space. During the extension, pixels of the uncertainty areas are assigned to a given marker. A point is assigned to a specific region because it is in the neighborhood of at least one marker and it is more similar (in the sense defined by a specific criterion) to this marker than to any other marker of its neighborhood. A detailed description of this modified version of the *watershed* can be found in [17].

A possible similarity criterion is the gray tone difference between the pixel under consideration and the mean of the pixels that have already been assigned to the region. This basic similarity measure has to be modified to take into account the complexity of the contours. Indeed, for coding, if the gray level transition between two regions is not very strong, it is useful to have simple contours, even if some precision on the position is lost. For this aim, the similarity criterion is defined as the weighted sum of the gray-level difference between the pixel and the mean of the region plus a penalty term corresponding to the contour complexity

$$\text{Similarity} = \alpha \text{ Difference in gray-tone} + (1-\alpha) \text{ Contour complexity.} \quad (1)$$

The measure of contour complexity is made by counting the number of contour points that are added if the pixel is assigned to that region. The weighting factor α allows more importance to be given to the gray level measure or to the contour complexity. The decision produces the new partition n which can be used as input to a new segmentation level. In practice, the

partition is represented by a gray-level image where the gray-level values define the region number. This kind of image is also called a *label* image.

Typically four hierarchical levels of segmentation are used for the intraframe processing. The first three deal with a size criterion and the last with a contrast criterion. At each level, the same procedure is repeated and the only difference is the simplification parameter which is decreased to allow the progressive introduction of small or low-contrasted regions. This hierarchical procedure has the following main advantages: it produces a good segmentation of the image taking into account the possibilities of the texture coding (it segments the coding residue). It allows the progressive estimation of the segmentation parameters, size and contrast, to get a segmentation result compatible with the coding objective (appropriate number of regions or of contour points, etc.).

Four segmentation levels are illustrated in Fig. 3. For each level the modeled image, the residue, the simplified residue and the segmentation are shown. This example illustrates how the information is progressively extracted from the original frame and introduced in the segmentation. Note in particular the evolution of the modeling error which becomes simpler at each level. Ideally, this error should tend toward zero. Comparison between the modeling error and its simplification illustrates the usefulness of using *connected operators* such as filters by reconstruction. For example, in the first level, the simplified image is really a coarse approximation of the original. Only large objects are present. However, the contours of the remaining objects are well defined. This simplified image is an "easy" image to segment. The simplified image involves a large number of flat zones which are very useful for marker extraction. The marker extraction step assigns a label to these flat zones. Finally, the segmentation is represented by the corresponding label image. The number of segmented regions is, respectively, 12, 28, 53, and 71 for each level.

B. Interframe Mode

Two different steps can be distinguished in the interframe mode: First, to define the time evolution of the regions that are present in the segmentation of the previous frame and, second, to detect the possible appearance of new regions [11].

The first problem is basically a projection problem. Denote F_{t-1} and F_t the original frames at time $t-1$ and t. Assume that the segmentation at time $t-1$, S_{t-1}, is known. Our goal is to find the segmentation, S_t, at time t without introducing new regions. For this purpose, two 3D signals are constructed. As illustrated by Fig. 4, frames F_{t-1} and F_t are grouped together to form a temporal block $\mathcal{F}$ of size 2 in the time direction. Similarly, the frame S_{t-1} is grouped with an empty frame S_o representing an entire frame of uncertainty. The resulting 3D signal denoted $\mathcal{S}$ is considered as the set of markers that should be used to segment the signal $\mathcal{F}$. The marker extension itself

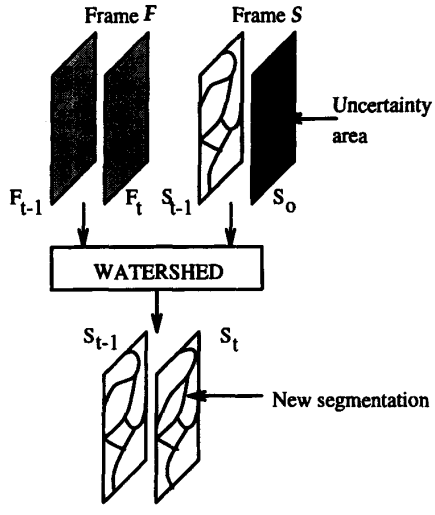


Fig. 4. Projection of markers of frame $n-1$ into frame n.

is achieved by the same decision algorithm used in the intraframe segmentation, that is *the watershed*. The only difference is the nature of the signals that are now 3D signals. The *watershed* extends the markers defined by S_{t-1} into the empty frame S_o. Each pixel of the uncertainty area (that is of frame S_o) is assigned to a region of frame S_{t-1} based on a similarity criterion.

As in the interframe mode, the similarity criterion combines a gray tone distance with a contour complexity measure. The contour complexity is assessed by counting the number of contour points that are added if the pixel is assigned to a particular region. A six connected neighborhood is assumed for this purpose, four pixels in the spatial dimension and two for the temporal. The contour complexity similarity in the time dimension plays an important role with respect to the temporal stability of the contours.

Once the labels of the previously segmented regions have been propagated into the current frame, new regions have to be extracted. From now on, the process is purely intraframe. First of all, the residue image is computed. As for the intraframe segmentation discussed previously, the residue image is obtained by texture-coding of the segmentation obtained from the extension of the past regions. A simplified version of the texture coding is used. In the following, the texture coding used in intraframe mode (see Section VI) for luminance is assumed to be used for the segmentation. Then, the difference with the original image gives the residue, where new regions can be detected. The segmentation of new regions is performed with the method used for the intraframe segmentation. That is, it consists of the basic segmentation steps: Simplification, marker extraction, and decision. The segmentation can be performed according to a size or a contrast criterion. Our experience has lead us to use a contrast segmentation only since, most of the time, large and noncontrasted

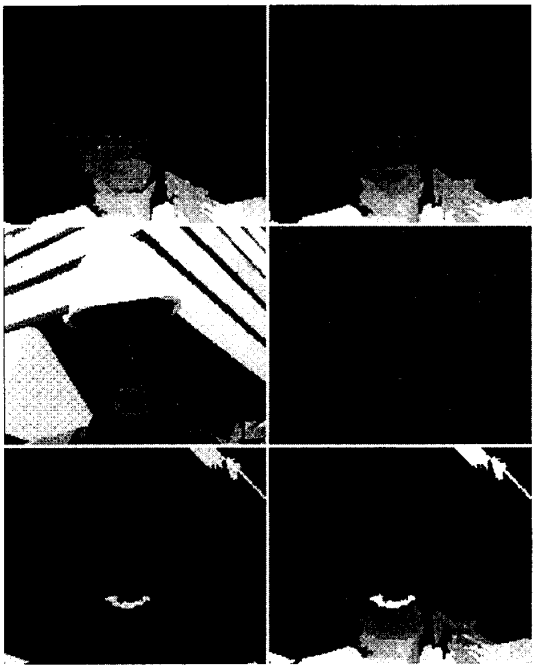


Fig. 5. Example of interframe segmentation. First row: segmentation of frame $n-1$ and its projection in frame n. Second row: modeled projection and the residue. Third row: new regions and final segmentation of frame n.

regions of the residue are not visually important. Note that the interframe mode of segmentation is very similar to the intraframe mode. The main difference is in the projection step which does not exist in the intraframe mode. In intraframe mode, it can be considered that the algorithm segments a sequence of still images and that the segmentation parameters are modified to increase the number of regions. In interframe mode, the hierarchy is not spatial but temporal. The sequence is moving and the segmentation parameters are mainly updated to maintain the partition characteristics.

The interframe mode of segmentation is illustrated in Fig. 5. The first row presents the segmentation of frame $n-1$ and its projection at time n. The second row gives the modeled projection and the corresponding residue. Finally, the last row shows the new regions and the final segmentation of frame n. As can be seen, three new regions have been extracted.

C. Regulation of the Segmentation

The regulation of the segmentation is done both in the intra- and interframe modes to control the final bit stream. In the intraframe mode, the number of segmentation levels and the segmentation parameters, *size* or *contrast*, can be easily controlled to produce a segmentation with given characteristics such as number of regions or number of contour points [15]. Controlling the number of contour points of the segmentation does not precisely regulate the bit stream. However, in practice, it gives a reasonable stability of the bit stream produced by the contour coding. It can be considered as an approximate regulation. Texture coding is used afterwards to absorb the deviation in number of bits.

Moreover, thanks to the segmentation criteria used in the scheme, the segmentation parameter estimation can be simply done. Assuming that we only want to regulate the bit rate for the contour information and that this bit rate is directly proportional to the number of contour points $\mathcal{CP}$ produced by the segmentation [15], then the parameters estimation can be done as follows:

- *Estimation of the Size Parameter*: For a very large number of shapes, it can be assumed that the perimeter P is proportional to the square root of the area S (size): $P = \alpha_1\sqrt{S}$. Moreover, the number N of shapes of size S that fits within a fixed area (that is the frame) is generally inversely proportional to the size S: $N = \alpha_2/S$. Finally, the number of contour points (and therefore the number of bits for contour) can be approximated by

$$\mathcal{CP} = \frac{\alpha}{\sqrt{S}}. \tag{2}$$

  In the hierarchical segmentation procedure described previously, the α parameter can easily be estimated by averaging the size parameters S and the actual number of contour points obtained for the previous segmentation level.
- *Estimation of the Contrast Parameter*: The contrast parameter can be estimated by using two thresholds at c and $-c$ on the *residue* image. The number of new contour points, $\Delta\mathcal{CP}$, created by the contrast-oriented segmentation with c as parameter is assumed to be proportional to the number of points where the residue image crosses level c and $-c$. As in the case of size segmentation, the proportionality factor is estimated by taking into account the previous segmentation level.

Once the intraframe segmentation has been performed, the interframe mode should mainly preserve the characteristics of the segmentation, that is the number of regions and of contour points. Assume that we want to work with a constant number of contour points $\mathcal{CP}$. If the number of contour points of the current segmentation is below this threshold $\mathcal{CP}$, then the contrast parameter (only contrast oriented is performed in the interframe mode) should be decreased to allow the introduction of a few new regions. On the contrary, if the number of contour points of the current segmentation is above $\mathcal{CP}$, then the contrast parameter can be progressively increased in order to prevent the introduction of new regions and wait for possible disappearance of regions. Note however, than depending on the sequence, this procedure does not guarantee the actual decrease of the number of contour points. Therefore, if the number of contour points is really too high, and reaches for example 1.2 $\mathcal{CP}$, regions are merged together on the basis of their mean gray level. In practice, this merging procedure is very seldom used.

IV. Region-Based Motion Estimation

Within the framework of region-based coding techniques, contour coding represents a bottleneck in terms of compression rate. In the 2D case, the modified chain code method proposed in [6] needs in practice 1.3 b/contour point. In the 3D case, the spatial-temporal contours make the situation even worse. For sequences, motion estimation and compensation is generally used to exploit the temporal redundancy of gray-level pixels. Motion estimation analyzes the pixels' motion in the sequence, and the interframe displacement information allows the prediction of the current pixel value. In order to achieve a high compression ratio, the temporal redundancy of the contour image sequence, that is of the segmentation, should be exploited.

The segmentation algorithm described previously solves the region correspondence problem: We know exactly how the past regions are transformed into current regions and where the new regions are. Motion estimation is introduced to code the current partition by prediction from the previous coded partition. Furthermore, the resulting motion information can also be used for texture compensation. However, it has to be noticed that, for a given region, the motion of its contours may not coincide with the motion of its texture. This is in particular the case for background regions. Indeed, the contour modifications of a background region are due to the motion of the surrounding foreground regions and not to its own motion. In the following, the motion information, that is estimated and used for compensation, is the motion of the contours because contours represent very sensitive information and contour errors result in a rapid increase of the bit stream. On the contrary, texture information is less sensitive and graceful degradation of the texture quality can more easily be achieved.

Therefore, the goal of the motion estimation is to assign to each region a set of motion parameters allowing the prediction of its shape. A translational motion model is used here for simplicity and to reduce the cost associated to the coding of the motion parameters.

Assume that A, B, v, respectively, denote a region in the previous frame $n-1$, the corresponding region in the current frame n and a translation vector. The goal of the motion estimation is to minimize the following prediction error

$$\min\{(A_v \cup B) - (A_v \cap B)\} \tag{3}$$

where A_v denotes the compensated version of A. Note that the motion estimation is performed from frame $n-1$ toward frame n. This does not correspond to the classical approach of motion estimation for compensation. However, in a region-based approach, the receiver has to compensate regions (by translation). The only regions that are known at time n before contour decoding are the previously reconstructed regions, that is, the regions corresponding to frame $n-1$.

For each region, the prediction error is calculated as the total number of pixels of the frame n which are not assigned to their correct region by motion compensation. Since an accurate estimation of the region motion considerably reduces the prediction error, a full-search motion estimation algorithm is used. A fast full-search region matching algorithm has been developed to minimize the prediction error within reasonable time. This fast algorithm avoids the estimation of the motion of each region separately, but deals with them all together.

For a given motion vector v, the whole partition at time $n-1$ is translated. Then, the compensation error for each region is calculated by scanning the image once. The error computation is done by comparing the current partition with the translation of the previous partition. Let us denote by $p_n(x,y)$ the region number of a pixel at position (x,y) in the current partition and by $p_{n-1,v}(x,y)$ the region number at the same position after translation of the previous partition. For each $p_n(x,y)$, one of the three following situations may occur:

1) $p_n(x,y) = p_{n-1,v}(x,y)$: The compensation gives the correct result and the pixel does not contribute to the error.
2) $p_n(x,y)$ has no corresponding pixel because the translation of the previous partition would have taken pixels outside the frame limits. The error corresponding to the region number $p_n(x,y)$ is incremented by one.
3) $p_n(x,y) \neq p_{n-1,v}(x,y)$: The two errors corresponding to the region numbers $p_n(x,y)$ and $p_{n-1,v}(x,y)$ are incremented by one.

By using this method, the compensation error of all regions for a specific translation v can be obtained by a single scanning of the image. Comparing with a full-search motion estimation algorithm for each region, there is a significant decrease of complexity which can be estimated by a factor k given by

$$k = \frac{O(\text{old})}{O(\text{new})} = \frac{2 \cdot S_x \cdot S_y \cdot L_x \cdot L_y \cdot R}{S_x \cdot S_y \cdot (L_x \cdot L_y + R)} \approx 2 \cdot R \tag{4}$$

where R is the total number of regions, S_x, S_y are the lengths of the searching window, L_x, L_y are the sizes of the current image. Finally, once the error produced by all possible translations for all regions has been computed, the translation creating the minimum error is assigned to each region of the previous partition. This motion information is used in the following for both contour and texture compensation.

V. Motion Compensated Contour Coding

The transmission of the contour information follows the classical motion predictive technique illustrated in Fig. 6. Based on the previous partition image stored in the *contour memory* and on the motion information, a compensated partition image is created by the *contour compensation* block and its difference with the current partition defined by the segmentation is computed. Since we are dealing with partition, the difference operator has to be considered as a set difference. The difference, called *contour error*, is simplified, coded, and transmitted to the receiver [3].

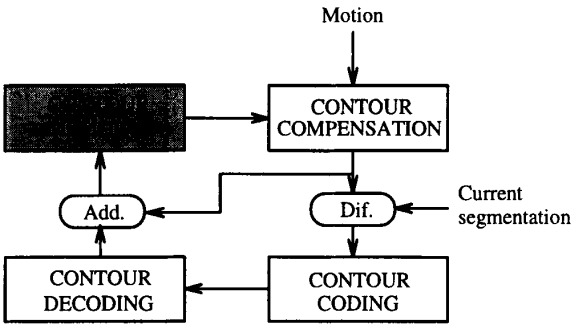


Fig. 6. Structure of the contour encoding.

A. Contour Compensation

1) Separation of Foreground and Background Regions: Since a forward motion estimation is used, adjacent regions in the previous frame may overlap after compensation in the current frame. Moreover, two adjacent regions share a common contour segment. If a contour segment has already been considered in one of the two adjacent regions, it should not be taken into account again in the compensation of the other region. Furthermore, if all the contour segments defining a region have already been considered, we only need to send a region number to code that region. Such regions are defined as background regions, while the others are defined as foreground regions.

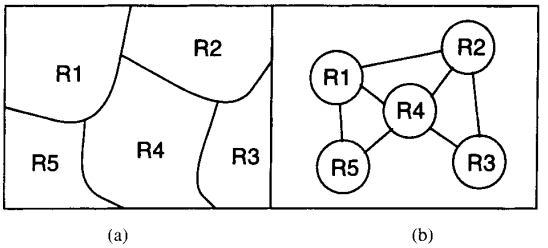
In order to distinguish between background and foreground regions, a region adjacency graph (RAG) representing the adjacency relation between the regions of the current frame is constructed. Depending on the prediction error, an order list is built to indicate the translation order of each region. A region associated with a small prediction error should be translated before a region producing a large prediction error. An example is shown in Fig. 7. Suppose that the order list is $R_1R_4R_3R_2R_5$ where R_1 has the minimal prediction error and R_5 has the maximal prediction error. Based on the RAG and this order list, the regions in the current frame can be easily classified into foreground and background regions. Following the list order, each region is examined. If all its surrounding regions have already been defined as foreground regions, this region is defined as a background region. If not, it is defined as a foreground region. According to this procedure, in the case of Fig. 7, $R_1R_4R_3$ are defined as foreground regions and R_2R_5 as background regions.

Only the prediction errors for the foreground regions need to be transmitted. This is the reason why the regions creating a low error should be processed first. The boundaries of background regions are naturally defined by the reconstructed foreground regions. Eliminating the background regions greatly reduces the prediction error components and leads to an efficient representation.

2) The Morphological Filtering of Prediction Error: After motion compensation, the resulting prediction error may be noisy. If the prediction error is coded losslessly, the resulting number of bits is very high. To achieve a high compression ratio, a lossy method should be used. Since small prediction errors of one pixel width represent a large part of the bit stream and have little visual importance, they may be removed. Morphological filters are shape-oriented and are particularly suitable for removing objects that are smaller than a specific size. They are introduced to purge the prediction error of each foreground region.

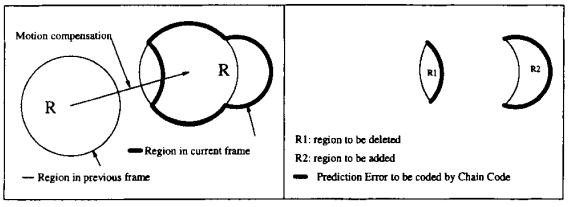
A *morphological opening* can remove objects of size smaller than the size of the structuring element. But it will also cause contour degradation of the remaining large objects [16]. *Openings by reconstruction* (see appendix) do not have this drawback. A structuring element of size $2 * 2$ is used. It results in a maximal error of one pixel for the contour location. This simplification ensures the preservation of the boundaries for large prediction error while erasing the very small error components.

To further reduce the prediction errors of the foreground regions, a mask is incrementally constructed in the current frame to indicate the areas where regions have already been settled. Any prediction fragment falling into this mask is not considered.



(a) (b)

Fig. 7. (a) Labeled image in current frame and (b) region adjacency graph (RAG).



(a) (b)

Fig. 8. (a) Region-based motion compensation and (b) prediction error coding.

B. Contour Error Coding

1) Prediction Error Coding: After motion compensation of the foreground regions, the prediction error is obtained for each foreground region. An efficient coding method has to be found to code this prediction error. From Fig. 8(a), it can be seen that the boundaries of the prediction error are composed of motion compensated boundaries and of the new boundaries in the current frame. Only the new boundaries need to be coded. These new boundaries are divided into two groups: Boundaries defining areas to be added and boundaries defining areas to be deleted.

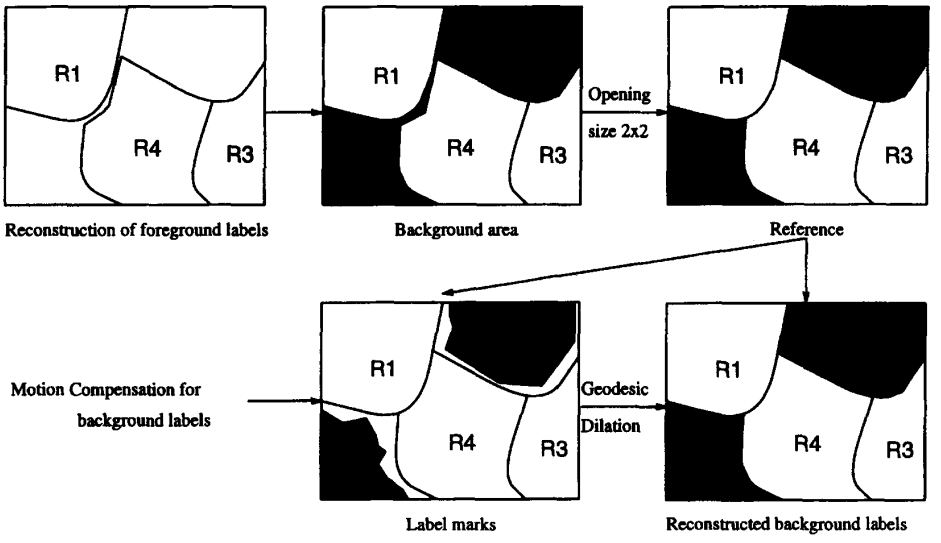


Fig. 9. Reconstruction of background regions.

From Fig. 8(b), it can be seen that actually only the thick curves need to be coded. For the coding of a single continuous digital curve, a derivative chain code is used [1], [6]. The curve coding needs three symbols to indicate the movement of the continuous boundary and the starting points of the prediction error segments are coded differentially.

Finally, it should be mentioned that the new appeared regions in the current frame are directly coded by chain code since there are no corresponding regions in the previous frame that can be used to predict them. Note that, in the intraframe mode all regions are processed as new regions and are coded by chain code. At the end, the information to transmit is composed of the motion vectors, the prediction error and the order of the foreground regions. All this information is entropy coded by a first order adaptive arithmetic coder.

2) Reconstruction of Foreground and Background Regions: At the decoder side, the foreground regions are reconstructed first. The reconstruction is based on the previous received partition, the motion vectors, the prediction error and the order of translation of foreground regions. This simple procedure consists in translating the regions according to the list order and their motion. As in the encoder side, a mask is incrementally constructed to indicate the area where regions have already been settled. Any motion compensated part falling into this area is neglected. Then, the prediction error which has not been removed by the morphological filter will be either added to or deleted from the motion compensated regions.

After the reconstruction of the foreground regions, the background regions should be restored with the correct region number. First, a reference mask in the current frame is built to indicate all the background areas where the region number has to be defined, see Fig. 9. Because of the *opening by reconstruction* of size $2 * 2$ used to simplify the prediction error for foreground regions, small cracks may appear between foreground regions. The number of the region in these crack areas are uncertain. Moreover, these cracks may join together background regions that were originally disconnected. To keep the correct topological relation in the current image, a *morphological opening* of size $2 * 2$ is applied to the background area to obtain the correct reference mask.

Once the shapes of the background regions have been defined on the receiver side, a region number should be assigned to each of them. Each region of the previous frame which does not correspond to a foreground region is motion compensated. The biggest connected component defines the region number. In fact, the resulting image might be considered as a partially reconstructed background partition. Afterwards, a reconstruction by *geodesic dilation* leads to a complete reconstruction of the whole background regions, see Fig. 9.

Finally, the reconstructed foreground and background regions image are put together to form the current partition. A post-processing procedure is carried out to remove small isolated connected components of one pixel width.

VI. Motion Compensated Texture Coding

Once the partition has been transmitted to the receiver, the texture or color information has to be coded. The strategy follows the classical motion predictive technique illustrated in Fig. 10: Based on the previous texture image stored in the *texture memory* and on the motion information, a compensated texture image is created by the *texture compensation* block and its difference with the original texture frame is computed. The difference, called *texture error*, is coded and transmitted to the receiver. The texture image is created by adding the compensated texture and the coded error. As can be seen in Fig. 10, all coding steps can

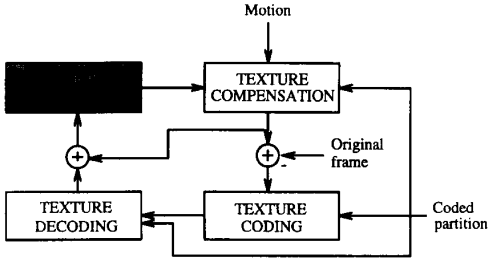


Fig. 10. Structure of the texture encoding.

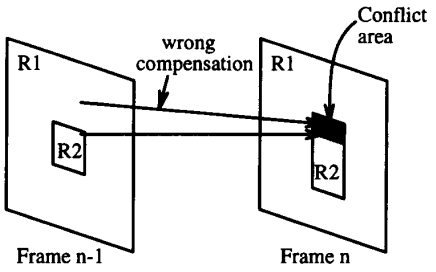


Fig. 11. Texture compensation in the case of conflict areas.

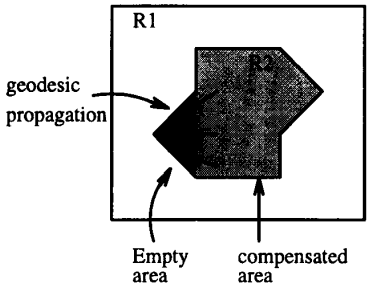


Fig. 12. Texture compensation in the case of empty areas.

be region-based since the partition has already been coded and the receiver knows the different regions.

For color sequences, the coding process is achieved separately on the luminance (Y) and the chrominance (U and V) signals. The only difference is the quality of the texture error coding which has a much higher accuracy in the case of luminance.

A. Texture Compensation

As discussed in the previous section, the motion information estimated on the contours is used to compensate both the contours and the texture. This approach may produce errors and inaccuracies in the compensated image, in particular, for background regions. However, with a simple model, like the translational model that is used here, these texture compensation errors are not very annoying.

The texture compensation is more complicated than the "classical" compensation because the motion estimation has been performed from frame $n-1$ toward frame n. Therefore, the projection of frame $n-1$ into frame n creates some empty areas, where no values have been assigned to the pixels, and some conflict areas, where more than one value is assigned. To assign a value to these pixels, the coded partition information can be used.

Fig. 11 illustrates a case of conflict area. Several pixels of frame $n-1$ are projected to the same location of frame n. As a result several gray-level values may be assigned to the pixel of frame n. However, since each region of frame $n-1$ can only be translated, each possible gray-level value corresponds to a specific region of frame $n-1$. Most of the time, one of these regions in frame $n-1$ corresponds to the actual region of the pixel in frame n. In Fig. 11, regions $R1$ and $R2$ of frame $n-1$ are in conflict but the pixels of the conflict area belong to region $R2$ in frame n. The conflict can be naturally solved by taking the gray-level value of the pixels of frame $n-1$ coming from the same region as the conflict pixel in frame n. In the example of Fig. 11, only pixels coming from region $R2$ are taken into consideration in the conflict area. If none of the possible regions of frame $n-1$ corresponds to the region of the conflict area, the area will be processed as an empty area. Finally, note that these conflict areas were also present during the compensation of contours. However, the coding of the contour error has solved these conflicts and the coded partition is used now as a reference to solve the texture conflicts.

Empty areas cannot be solved by the preceding technique because no gray-level candidates in frame $n-1$ can be found. In this case, the safest gray-level value to take is the value of the closest pixel of the same region in frame n. As illustrated by Fig. 12, this procedure can be seen as a propagation of the pixels gray-level values of the region into its empty areas. This is a *geodesic dilation* of the borders of the empty area in the empty area itself.

B. Texture Error Coding

After motion compensation, texture errors are coded. Texture errors come, on the one hand, from inaccuracies of the compensation (nontranslational motion, region deformation, background region, etc.) and, on the other hand, from the presence of new regions which cannot be compensated. Note that in intraframe mode, all regions are considered as new regions and are directly coded. A large number of techniques can be used for this purpose. In the following, a simple region-based transform technique is used [2].

Starting from a given basis of functions such as 2D polynomials or 2D cosine functions, an orthogonal basis is constructed for each region. The orthogonality condition means that the inner product of the realizations of the basis vectors *inside* the region is equal to zero. The orthogonalization algorithm is similar to the classical Graham–Schmitt algorithm. Once the orthogonal basis has been computed, the gray-level function representing the texture is projected onto this basis. This results in a set of coefficients that have to be coded and transmitted to the receiver. Note that the basis itself has not to be transmitted since it only relies on the regions' shape and can be computed in the receiver. The use of an orthogonal basis is important because it produces coefficients with a very compact probability density function. In fact, the

Table 1

Sequence	Mode	Motion bits	Contour bits	Texture bits	Total bits	bit rate at 5 Hz kb/s
Miss America	Intra-frame	0	4879	3950	8829	
Miss America	Inter-frame	220	2167	1238	3625	18
Carphone	Intra-frame	0	6528	3950	10478	
Carphone	Inter-frame	641	3228	1348	5217	26
Foreman	Intra-frame	0	6740	4052	10792	
Foreman	Inter-frame	529	3950	1941	6420	32

resulting set of coefficients have a distribution which is very similar to that obtained with block-based transforms. In particular, if the original basis relies on cosine functions, the transformed coefficients can be quantized and coded as DCT coefficients. The coding of coefficients involves a quantization, a zig-zag scanning of the coefficient space, and the coding of the runs of zeros and the amplitude of nonzero coefficients. Finally, note that the quantization of the coefficients can be easily controlled to precisely regulate the final bit stream.

Fig. 13 illustrates the texture compensation and coding. The first row presents the coded texture of frame $n-1$ and its compensation. In the compensation, the empty and conflict areas are respectively shown in black and white. As can be seen, these areas mainly correspond to transitions between objects and new regions. In particular, large black areas appearing in the window correspond to new regions. The second row of Fig. 13 gives the compensated frame after processing of empty/conflict areas and the compensation error. The major part of the error is related to new regions which cannot be predicted from frame $n-1$. Finally, the last row gives the coded error and the coded frame n.

Fig. 13. Example of texture compensation. First row: coded texture of frame $n-1$, compensated frame with empty areas in black and conflict areas in white.Second row: compensated frame with new and unresolved areas in white, compensation error. Third row: coded compensation error, reconstructed frame n.

VII. Results

To illustrate the approach described in this paper, the well known *Miss America*, *Carphone*, and *Foreman* sequences in QCIF format have been selected.

The *Miss America* sequence is segmented with a target number of contour points $\mathcal{CP}$ of 2500, whereas a value of $\mathcal{CP}$ of 4250 is used for the two remaining sequences. In intraframe mode, the four segmentation levels respectively produce approximately $\mathcal{CP}/4$, $2\mathcal{CP}/4$, $3\mathcal{CP}/4$, $\mathcal{CP}$ contour points. These parameters lead to an average number of regions of 60 for *Miss America*, 80 for *Carphone*, and 90 for *Foreman*. The segmentation is performed recursively at 25 Hz. This means that all frames are segmented. This point is important because, if the segmentation rate is decreased, small moving objects may be disconnected. They result in regions which are always processed as new regions and may have an influence on the overall bit stream.

Motion estimation, contour and texture coding are performed at 5 Hz. The search range for the motion estimation is fixed at $\pm$ 15 pixels. Finally, the texture coding relies on decomposition on a orthogonal basis of cosine functions. The number of basis vectors, that is of coefficients to code, for each region is equal to 25 for the luminance signal (Y) and to 4 for chrominance signals (U and V). The quantization of the coefficients is stronger in interframe mode than in intraframe.

Table 1 gives some details about the bit stream composition after entropy coding by a first order adaptive arithmetic coder. In all cases, 150 frames have been coded. The figures are averages.

Finally, Fig. 14 shows the original frames corresponding to frame numbers 10, 60, and 110 of each sequence. Fig. 15 presents the luminance of the coded sequences. These examples show that the approach described in this paper is appropriate for very low bit rate video coding. Bit rates lower than 32 kb/s can be obtained with reasonable quality. Moreover, no assumptions have been made about the sequence content.

VIII. Conclusion

A region-based coding algorithm for video sequences has been presented in this paper. This algorithm is mainly devoted to very low bit rate video coding applications. No

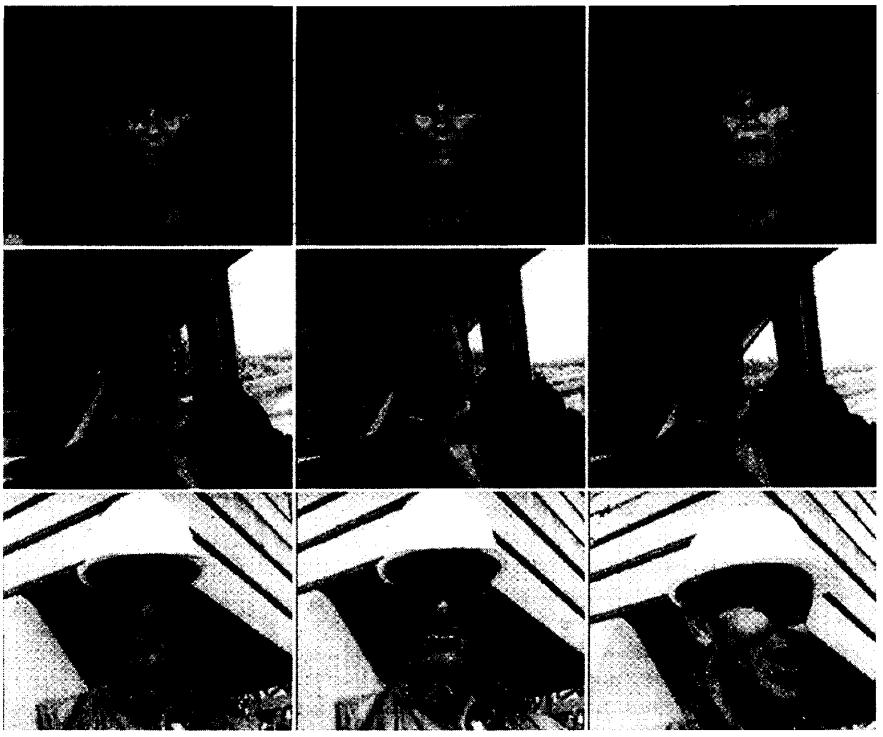


Fig. 14. Original frames (number 10, 60, and 110) of the sequences *Miss America*, *Carphone*, and *Foreman*.

assumption is made about the sequence content and, the algorithm structure leads to a scalable coding process able to give various levels of quality and bit rates. Several steps of the algorithm are controlled to regulate the bit stream.

The coding approach involves a time-recursive segmentation relying on the pixels' homogeneity, a region-based motion estimation, and motion compensated contour and texture coding. The segmentation process relies on morphological tools such as *connected operators* and *watersheds*. It deals with two possible criteria: Size and contrast, which are appropriate to extract the visually most important image components. Motion estimation is done after segmentation and is used for the coding of contour and texture. Finally, both contour and texture are coded by motion compensation predictive techniques. Morphological techniques have proved to be very useful for specific steps of contour coding and for texture compensation. As shown by several examples, this approach gives very interesting results for very low bit rate video coding.

APPENDIX

The goal of this section is to define the basic morphological tools discussed in the paper. A complete description of mathematical morphology can be found in [19], [20].

A. Morphological Operators and Filters

A large number of morphological tools relies on two basic sets of transformations known as *erosions* and *dilations*. In this paper, two sets of *erosions* and *dilations* are used. The first one deals with *erosion* and *dilation* with flat structuring element. If $f(x)$ denotes an input signal and M_n a window (or flat structuring element) of size n, the *erosion* and *dilation* by M_n are given by

$$\text{erosion: } \epsilon_n(f)(x) = \text{Min}\{f(x+y), y \in M_n\} \quad (5)$$

$$\text{dilation: } \delta_n(f)(x) = \text{Max}\{f(x-y), y \in M_n\}. \quad (6)$$

The second set of *erosions* and *dilations* involves geodesic transforms [5]. They are always defined with respect to a reference function r. The geodesic dilation of size one (that is the smallest size on the discrete space) is defined as the minimum between the dilation of size one of the original function f and the reference function r. The *geodesic erosion* is defined by duality:

$$\text{geodesic dilation of size 1: } \delta^1(f,r) = \text{Min}\{\delta_1(f), r\} \quad (7)$$

$$\text{geodesic erosion of size 1: } \epsilon^1(f,r) = -\delta^1(-f,-r). \quad (8)$$

Geodesic dilations and *erosions* of arbitrary size are defined by iterations. For example, the *geodesic dilation (erosion)* of infinite size, also called *reconstruction by dilation (by erosion)* is given by

$$\text{reconstruction by dilation: } \gamma^{\text{rec}}(f,r) = \delta^{\infty}(f,r) = (\cdots\delta^1(\cdots\delta^1(f,r)\ldots,r) \quad (9)$$

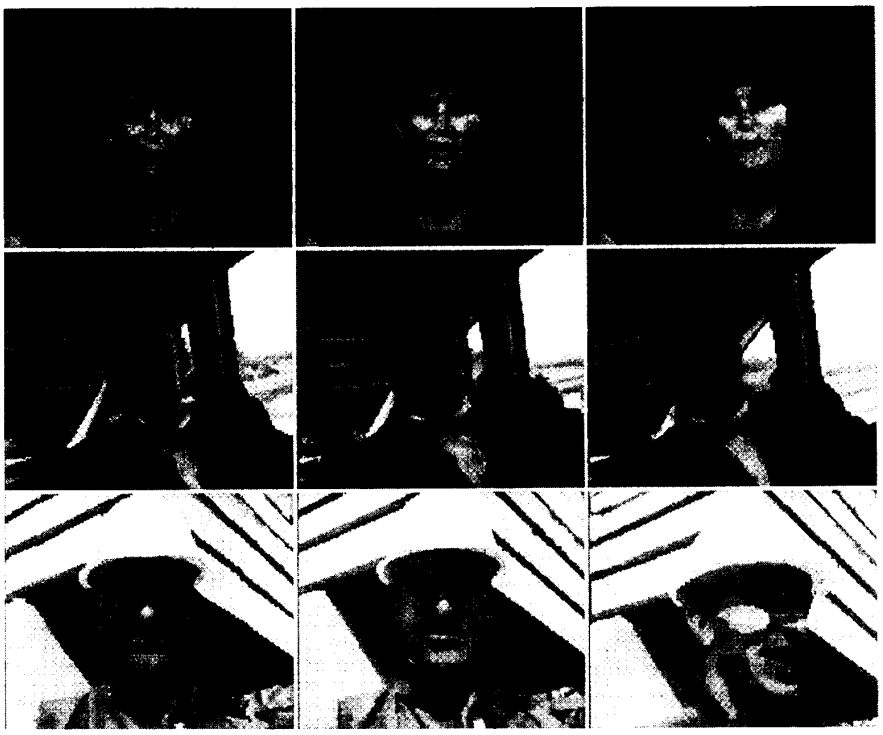


Fig. 15. Coded frames (number 10, 60, and 110) of the sequences *Miss America*, *Carphone*, and *Foreman*.

reconstruction by erosion:

$$\varphi^{\text{rec}}(f,r) = \epsilon^{\infty}(f,r) = (\cdots\epsilon^{1}(\cdots\epsilon^{1}(f,r)\ldots,r). \tag{10}$$

It has to mentioned that reconstruction processes can be implemented very efficiently by using queues which avoid any iterating process and lead to extremely fast algorithms [22].

Elementary *erosions* and *dilations* allow the definition of morphological filters such as the *morphological opening* and *closing*:

$$\text{morphological opening: } \gamma_n(f) = \delta_n(\epsilon_n(f)) \tag{11}$$

$$\text{morphological closing: } \varphi_n(f) = \epsilon_n(\delta_n(f)). \tag{12}$$

A *morphological opening* (resp. *closing*) simplifies the original signal by removing the bright (resp. dark) components that do not fit within the structuring element. It is a size-oriented simplification. If the simplification has to deal with both bright and dark elements, an *open_close* $\gamma_n(\varphi_n(f))$ or a *close_open* $\varphi_n(\gamma_n(f))$ has to be used. None of these filters are self-dual, but in practice they approximately remove the same kind of information. These filters can be used as simplification tools before segmentation, but they do not allow a perfect preservation of the contour information [16]. In order to improve the contour preservation properties, filters by reconstruction should be used.

The most popular filter by reconstruction is the *opening by reconstruction* of erosion $\gamma^{\text{rec}}(\epsilon_n(f), f)$. Of course, by duality, a closing can be defined: $\varphi^{\text{rec}}(\delta_n(f), f)$. These filters have a size-oriented simplification effect on the signal but preserve the contour information.

Finally, the h-maxima and h-minima operators are used for contrast-oriented simplification. They can also be defined in terms of reconstruction. If h is a constant,

$$h - \max(f) = \gamma^{\text{rec}}(f-h, f) \tag{13}$$

$$h - \min(f) = \varphi^{\text{rec}}(f+h, f). \tag{14}$$

Filters by reconstruction belong to the class of *connected operators* that have been introduced in [21], [18]. Among the set of theoretical results given in [21], let us focus on the following ones: *Connected operators* fundamentally have the property of interacting with the signal by producing *flat zones*. A *flat zone* is a connected component of the image where the gray-level value is constant. Note that a *flat zone* may be reduced to a single point. Moreover, consider a family of connected operators $\{\psi_\lambda\}$ depending on a parameter λ (for example, a family of *open_close by reconstruction* depending on the size of the structuring element). Assume that the family has the following property, called a *pyramidal* property in [21], [18]

$$\text{for any } \lambda \geq \mu, \exists \nu \text{ such that } \psi_\lambda = \psi_\nu(\psi_\mu)$$

$$\text{that is, for any } f, \psi_\lambda(f) = \psi_\nu(\psi_\mu(f)). \tag{15}$$

Intuitively, this property means that, the knowledge of $\psi_\mu(f)$ is sufficient to compute all $\psi_\lambda(f)$ for $\lambda \geq \mu$. With these assumptions, it can be shown that *the set of flat zones produced by ψ_λ are either preserved or merged when λ increases*. In other words, the contours of the flat regions are either conserved or removed by the filter. This very strong property explains the simplification effect as well as the contour preservation feature of these filters. As a result, *connected filters* are extremely useful for segmentation.

B. Morphological Gradients

In morphology, three gradients are generally used:

$$\text{Morphological gradient: } g(f) = \delta_1(f) - \epsilon_1(f) \quad (16)$$

$$\text{Gradient by erosion: } g^-(f) = f - \epsilon_1(f) \quad (17)$$

$$\text{Gradient by dilation: } g^+(f) = \delta_1(f) - f. \quad (18)$$

All gradients are positive, but the first one is symmetrical with respect to the contour position whereas the two remaining ones are not. In [14], it was mentioned that the use of the gradient results in a loss of information. In particular, if the original signal involves transitions, its gradient is either biased (gradient by erosion or dilation) or thick (2 pixels). In the case of still images, this phenomenon is not extremely annoying. In the case of moving images, the use of the gradient results in a much larger loss of information [17] because its thickness depends on the objects' motion.

ACKNOWLEDGMENT

Besides the authors, the main participants of the *MORPHECO* project are L. Bouchard (Philips Research Lab., Paris, texture coding), P. Brigger (EPFL, contour coding), J. R. Casas (UPC, texture coding), T. Gasull (UPC, contour coding), B. Marcotegui (CMM, segmentation), F. Marqués (UPC, contour coding), M. Pardàs (UPC, segmentation), and O. Ribes (CMM, contour coding).

REFERENCES

[1] M. Eden and M. Kocher, "On the performance of contour coding algorithm in the context of image coding. Part 1: Contour segment coding," *EURASIP, Signal Process.*, vol. 8, pp. 381–386, 1985.

[2] M. Gilge, T. Engelhardt, and R. Mehlan, "Coding of arbitrarely shaped image segments based on a generalized orthogonal transform," *EURASIP, Image Commun.*, vol. 1, no. 2, pp. 153–180, Oct. 1989.

[3] C. Gu and M. Kunt, "Contour simplification and motion compensated coding," to be published in *EURASIP, Signal Process.: Image Commun.*, special issue for very low bitrate coding, 1995.

[4] M. Kunt, A. Ikonomopoulos, and M. Kocher, "Second generation image coding techniques," *Proc. IEEE*, vol. 73, pp. 549–575, Apr. 1985.

[5] C. Lantuejoul and F. Maisonneuve, "Geodesic methods in image analysis," *Patt. Recog.*, vol. 17, no. 2, pp. 117–187, 1984.

[6] F. Marqués, J. Sauleda, and T. Gasull, "Shape and location coding for contour images," in *Picture Coding Symp.*, pp. 18.6.1–18.6.2, Lausanne, Switzerland, Mar. 1993.

[7] F. Meyer, "Color image segmentation," in *4th Int. Conf. on Image Process. and its Appl.*, Maastricht, The Netherlands, May 1992, pp. 303–304.

[8] ——, "Morphological image segmentation for coding," in *1st Workshop on Math. Morph. and its Applic. to Signal Process.*, J. Serra and P. Salembier, Eds., pp. 46–51, Barcelona, Spain, May 1993.

[9] F. Meyer and S. Beucher, "Morphological segmentation," *J. Visual Commun. and Image Represent.*, vol. 1, no. 1, pp. 21–46, Sept. 1990.

[10] H. G. Musmann, M. Hötter, and J. Ostermann, "Object-oriented analysis-synthesis coding of moving images," *Signal Process., Image Commun.*, vol. 1, no. 2, pp. 117–138, Oct. 1989.

[11] M. Pardàs and P. Salembier, "Time-recursive segmentation of image sequences," in *EUSIPCO '94, VII Europe. Signal Process. Conf.*, Edinburgh, UK, Sept. 1994, pp. 18–21.

[12] S. Rajala, M. Civanlar, and W. Lee, "Video data compression using three-dimensional segmentation based on HVS properties," in *Int. Conf. on Acoust., Speech and Signal Process.*, pp. 1092–1095, New York, 1988.

[13] P. Salembier, "Multi-criterion segmentation for image coding," in *1st Workshop on Math. Morph. and its Appl. to Signal Process.*, J. Serra and P. Salembier, Eds., Barcelona, Spain, May 1993, pp. 40–45.

[14] ——, "Morphological multiscale segmentation for image coding," *EURASIP Signal Process.*, vol. 38, no. 3, pp. 359–386, Sept. 1994.

[15] P. Salembier, C. Gu, M. Pardàs, and M. Kunt, "Very low bit rate video coding using morphological segmentation and contour/texture motion compensation," in *IEEE 12th Int. Conf. on Patt. Recog.*, Jerusalem, Israel, Oct. 1994.

[16] P. Salembier and M. Kunt, "Size-sensitive multiresolution decomposition of images with rank order based filters," *EURASIP, Signal Process.*, vol. 27, no. 2, pp. 205–241, May 1992.

[17] P. Salembier and M. Pardàs, "Hierarchical morphological segmentation for image sequence coding," *IEEE Trans. Image Process.*, vol. 3, pp. 639–651, Sept. 1994.

[18] P. Salembier and J. Serra, "Flat zones filtering, connected operators and filters by reconstruction," *IEEE Trans. Image Process.*, vol. 3, Aug. 1994.

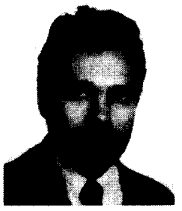
[19] J. Serra, *Image Analysis and Mathematical Morphology*. New York: Academic, 1982.

[20] ——, *Image Analysis and Mathematical Morphology, Vol II: Theoretical Advances*. New York: Academic, 1988.

[21] J. Serra and P. Salembier, "Connected operators and pyramids," in *SPIE Image Algebra and Math. Morph.*, vol. 2030, pp. 65–76, San Diego, CA, July 1993.

[22] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Patt. Anal. and Mach. Intell.*, vol. 39, pp. 1845–1855, Dec. 1991.

[23] P. Willemin, T. Reed, and M. Kunt, "Image sequence coding by split and merge," *IEEE Trans. Commun.*, vol. 39, 1845–1855, Dec. 1991.



Philippe Salembier received a degree from the Ecole Polytechnique, Paris, France, in 1983, and a degree from the Ecole Nationale Supérieure des Télécommunications, Paris, in 1985. He received the Ph.D. from the Swiss Federal Institute of Technology in 1991. He was a Postdoctoral Fellow at Harvard Robotics Laboratory, Cambridge, MA, in 1991.

From 1985 to 1989 he worked at the Laboratoires d'Electronique Philips, Limeil-Brevannes, France, in the field of telecommunication applied to television signals. In 1989 he joined the Swiss Federal Institute of Technology in Lausanne (EPFL) to work on image processing, specifically digital modulations and signal processing for HDTV. In 1991, he joined the Polytechnic University of Catalonia, Barcelona, Spain, where he teaches digital signal processing. His current research interests include image and sequence coding, image modeling, segmentation problems, texture analysis, mathematical morphology, and nonlinear filtering.

Luis Torres received a degree in telecommunication engineeering from the Telecommunications School of the Polytechnic University of Catalonia, Barcelona, in 1977, and the Ph.D. degree in electrical engineering from the University of Wyoming in 1986.

He is an Associate Professor at the Polytechnic University of Catalonia, Barcelona, Spain, where he teaches television systems and image processing courses. He is currently responsible for several projects in very low bit-rate video coding applications working toward the MPEG 4 standard. His main interests are in image coding, statistical coding, multiresolution schemes, and object-based systems.

Fernand Meyer was born in 1952. He received the engineering degree from the Ecole des Mines des Paris in 1975, and the Ph.D. in 1979.

He has been with the Centre de Morphologie Mathematique (CMM) of the Ecole des Mines des Paris since 1975, where he is now in charge of the coding activities and of the applications of image processing to medical imaging. His research interests are in skeletons, morphological segmentation, algorithmic design, and 3D mathematical morphology.

Chuang Gu was born in Shanghai, China, on March 23, 1965. He received the B.S. and M.S. degrees in computer science from Fudan University in 1986 and 1989, respectively.

During 1989–1990 he worked in the Computer Aided Design Center at Fudan University. From 1991 to 1992 he was a Research Fellow of the European Organization for High Energy Physics (CERN). In late 1992 he joined the the Signal Processing Laboratory at the Swiss Federal Institute of Technology (EPFL) as a Research Assistant. His primary research interests include mathematical morphology, image and image sequence coding, and computer vision. He is the author of more than 20 research publications, and is currently preparing his Ph.D. dissertation on very low bit-rate video coding techniques.