8.8 Independent vs. dependent optimization approaches

In the independent optimization mode, the bit-budget is fixed *a priori* for each scalability layer. In each case, the system obtains the best representation for the given bit budget. The iterative Lagrangian optimization process is performed only once per layer.

In the dependent optimization mode, the bit-budget is given globally for all the scalability layers. In this case, base and enhancement layer construction steps (Sections 8.4 and 8.5) are iterated several times, as stated by the algorithm presented in Section 7.2.

That is, instead of finding the λ_1^* that gives the optimal solution for a preset base layer bit-budget, the base layer is constructed for successive values of λ_1, starting at $\lambda_1 = 0$. For each value of λ_{1i}, a base layer is constructed and the corresponding rate and distortion values (R_{1i}, D_{1i}) are computed and stored. Then, for each value of λ_{1i}, the enhancement layer is constructed by using a target bit-rate of $R_{2i} = R_{budget} - R_{1i}$. The corresponding rate and distortion values (R_{2i}, D_{2i}) are computed and stored. At the end of this process, the stored values are used to find the optimal values λ_{1i}^* and λ_{2i}^* corresponding to the ones that minimize $D_{1i} + D_{2i}$.

8.9 Partition coding

Partition coding in the extended system follows the same approach than in the basic system (See Section 6.4) with some modifications. Three different types of information are involved: shape (contour of the regions), position, and the label of each region.

To improve the coding efficiency, the coding of the partition relies on constructing a prediction for the current partition and then coding the prediction error. The prediction error is then computed by comparing this prediction with the current partition created in the Decision step.

8.9.1 Structure of the partition coding process

Coding of the base layer partition is done basically in the same way as in the basic encoder. For the enhancement layer, the major difference with respect to the basic encoder is the possibility to choose between two possible references that can be used to build the prediction: the motion compensated previous frame enhancement layer partition (this is, an *inter-frame mode* as in the basic coder) and, in PSNR and spatial scalability, the base layer partition (*scalable mode*). Of course, an *intra-frame mode* is also used for intra-coded frames and for new regions in inter-coded frames.

When using the current frame base layer as a reference, common contours between base and enhancement layers will coincide *exactly*. This means that if a good part of the contours

in the enhancement layer coincide with contours in the base layer, this will make a very good reference.

But as the base layer partition is usually coarser, it is possible that only a reduced part (or none) of the contours in the enhancement layer are represented in the base layer. In this case, the contours of the base layer do not provide a correct prediction.

On the contrary, the shape of the compensated contours from the previous frame enhancement layer may be more similar to the shape of the current contours, but the accuracy on the contour position may be worse due to the imprecision on the compensation process, depending on the ability of the motion model to describe the actual motion of the objects. See Figure 8.14 for an example of both cases.

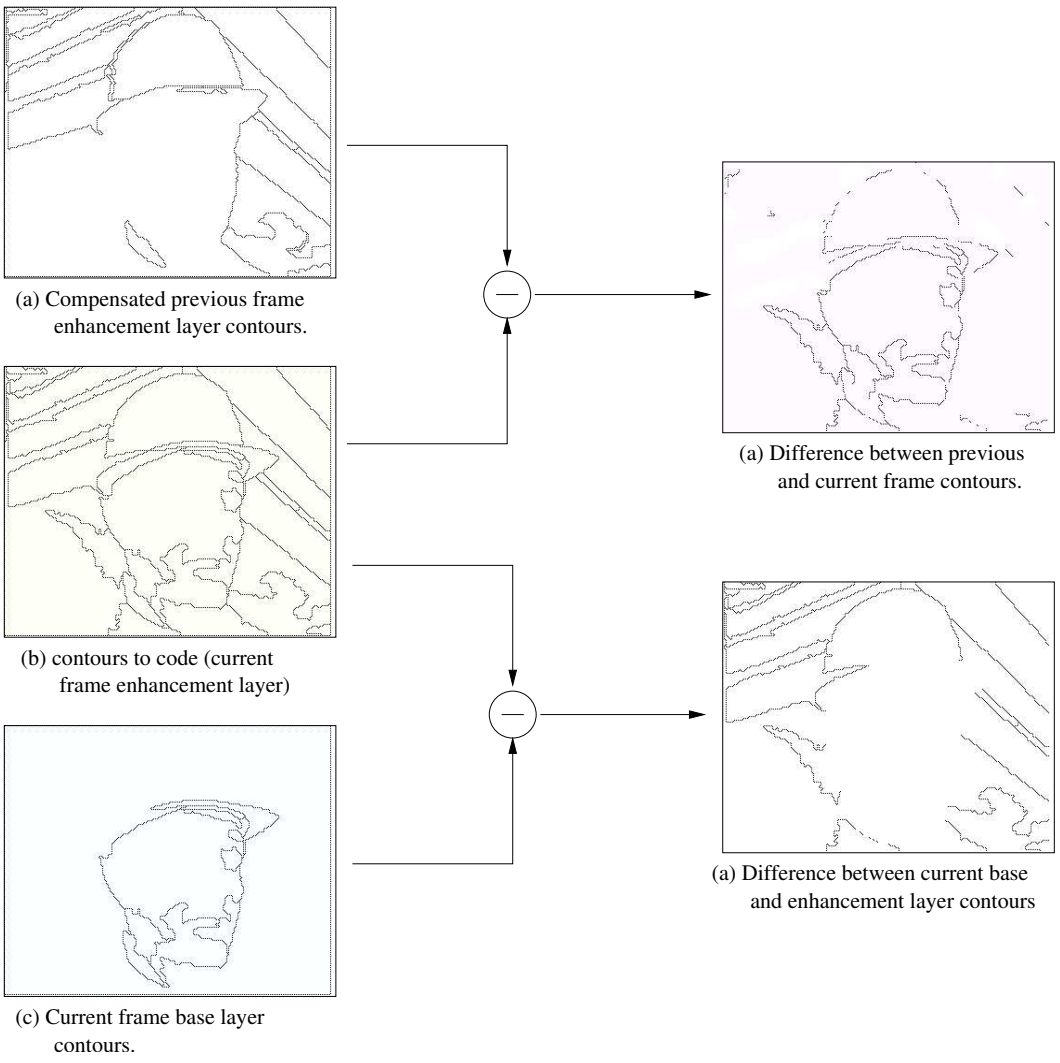

Figure 8.14: Comparison between inter-frame and intra-layer references for partition coding

To get the best reference in each case, a decision on which reference is to be used for each frame is taken by subtracting the contour image of the current frame enhancement layer partition from the contour images of previous frame enhancement layer partition and of the current frame base layer partition, respectively. The result providing more coincident points is kept.

Once the reference for contour coding is decided, the contour coding step is performed in the same way as in the basic system (See Section 6.4). The decoder must be informed of the type of contour reference selected at the encoder.

If the previous frame enhancement layer is selected as a reference, the coding of the partition is performed in the same way as in the non-scalable case.

If the base layer is taken as a reference, the process is simpler, because the contour compensation step can be skipped.

In both cases, labels are given using the enhanced projected partition. Details on how the enhancement layer regions are labeled are given in Section 8.11.

The following table summarizes the different partition coding modes available for the different coder profiles.

	Basic System (Only base layer)	Extended System			
		Base layer	Enhancement Layer		
			PSNR	Spatial	Temporal
Intra-frame mode	x	x	x	x	x
Inter-frame mode	x	x	x	x	x
Layer-Intra mode			x	x	

Table 8.1: Partition coding modes

8.9.2 Multi-Grid Chain Code

A new contour coding technique has been experimented in the extended coding system. In order to reduce the amount of information needed to code contours, a lossy system called Multi-Grid Chain Code [80, 32] (MGCC from now on) was studied.

MGCC is a quasi-lossless technique. This means that while it can provide increased efficiency over lossless techniques, it introduces few and controlled losses: losses are restricted to isolated pixels belonging to the boundary of the regions. This results in errors that are almost unnoticeable from a subjective quality point of view. Moreover, the encoding-decoding process ensures that the number of regions in the original and decoded partitions are the same.

The algorithm for MGCC developed in [36, 68, 69] has been used. This implementation

can efficiently represent general partitions in intra-frame mode. As the purpose was to use MGCC also for the inter-frame and scalable modes, these modes have been developed as a part of the work in this thesis.

A short description of the technique, as well as details of the three operation modes (intra-frame, inter-frame and scalable) are given in the sequel. In Section 8.9.3 a comparison between Chain Code and Multi-Grid Chain Code is given.

Intra-frame contour coding with MGCC

The same hexagonal contour grid as in the Chain Code technique described in Section 6.4 is used. In this case, instead of coding each individual contour site, MGCC defines a cell of 3x3 partition elements (5x5 contour elements) and uses a single movement to go through the cell (See Figure 8.15).

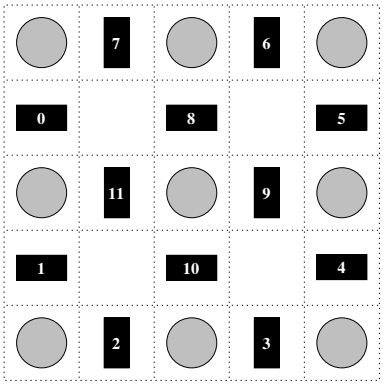

Figure 8.15: Structure of the MGCC cell. Starting by the input element indexed with a 0, any output element from the set {1 ... 7} can be reached going through the cell. Those contour elements inside the cell {8 ... 11} are not coded and introduce ambiguity in the coding process.

MGCC is a lossy technique because a symbol does not uniquely represent a contour configuration. As illustrated in Figure 8.16, a movement can correspond to two different contour configurations. Therefore, the central pixel can belong to two different regions. The decoder has to decide between two contour configurations. If the decision is incorrect, losses are introduced in the coding procedure.

As each cell has 7 possible output points, the size of the alphabet is greater than the one used for the chain code, however, the average step size is much larger and the probability of some symbols is much higher than the probability of other symbols and efficient entropy coding can be achieved.

Some movements represent larger steps than others. This is the case of symbols 3 or 4 with

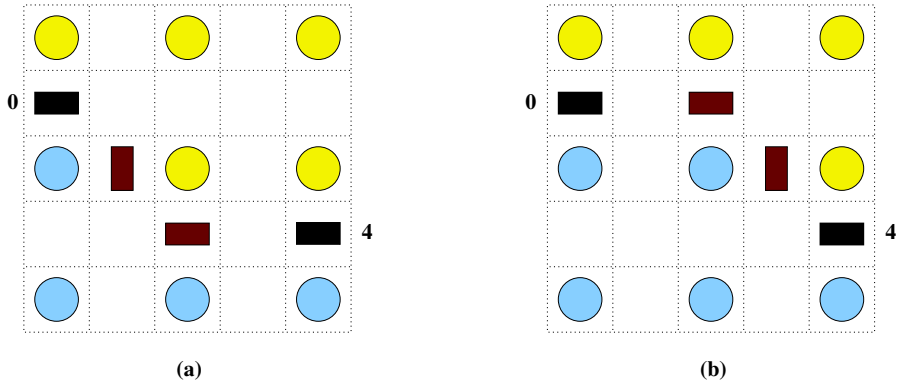

Figure 8.16: Two possible configurations for symbol 4.

respect to 1 or 7. Larger step sizes allow a more efficient representation because less symbols are needed. In order to increase the probability of large steps occurrence, two types of cells are used through the grid: *clockwise* (c) and *counterclockwise* (cc)(See Figure 8.17). This way, to characterize a cell, three parameters are necessary: its initial contour site, its type and its orientation. The orientation of a cell whose initial contour site is a horizontal contour element can be either *east* or *west* and, analogously, for a vertical contour element, it can be *north* or *south*. In order not to transmit additional information associated to the selection of the cell type, a prediction of the largest movement is made relying on the previous movement.

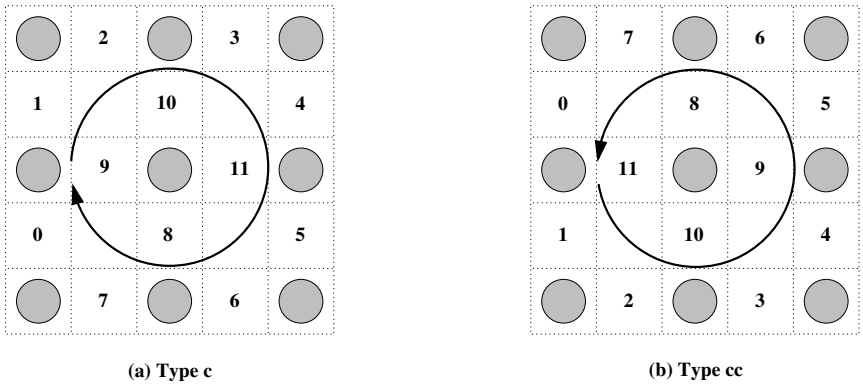

Figure 8.17: The two different cell types: c and cc

The contour is encoded by means of a contour tracking process: Starting from an initial point, cells are linked up to completing the contour. To link cells, the output contour site of the current cell becomes the input contour site of the following cell as indicated in Figure 8.19a. Contour sites inside the cell that are not linked with the initial contour are not

taken into account at this stage and will be encoded in a subsequent stage (See Figure 8.18).

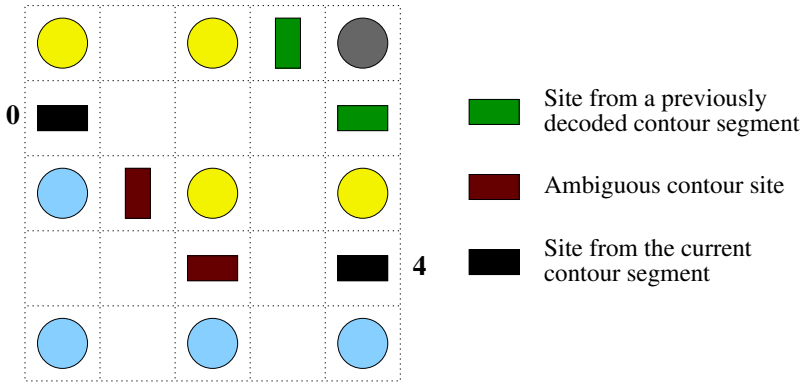

Figure 8.18: Outputs 5 and 6 are removed because they are not linked with the current contour segment being encoded (for instance, they may belong to a previously encoded contour segment)

The intersections of contours can be handled by using *triple points*. A triple point is a site in the contour grid that has at least three active neighbor contour sites (See Figure 6.13). Triple points can be used to locate the initial point of a new contour segment. Alternatively to the concept of triple point and given that the MGCC tracks the contour chaining cells, it is sometimes more natural to deal with the concept of cells with multiple outputs (See Figure 8.19b).

Cells with multiple outputs are encoded by introducing a new symbol, (M, say), into the chain itself. Priority is given to the output with a maximum length path. This ordering can help to eliminate ambiguities in the cell internal points. In the example in Figure 8.19b, only the represented configuration is compatible with the given set of symbols. Thus, no losses are introduced at the decoder.

With respect to previous implementations of the MGCC for general partitions [116, 67], the method used here [36, 68, 69] implies several improvements, while maintaining the coding efficiency.

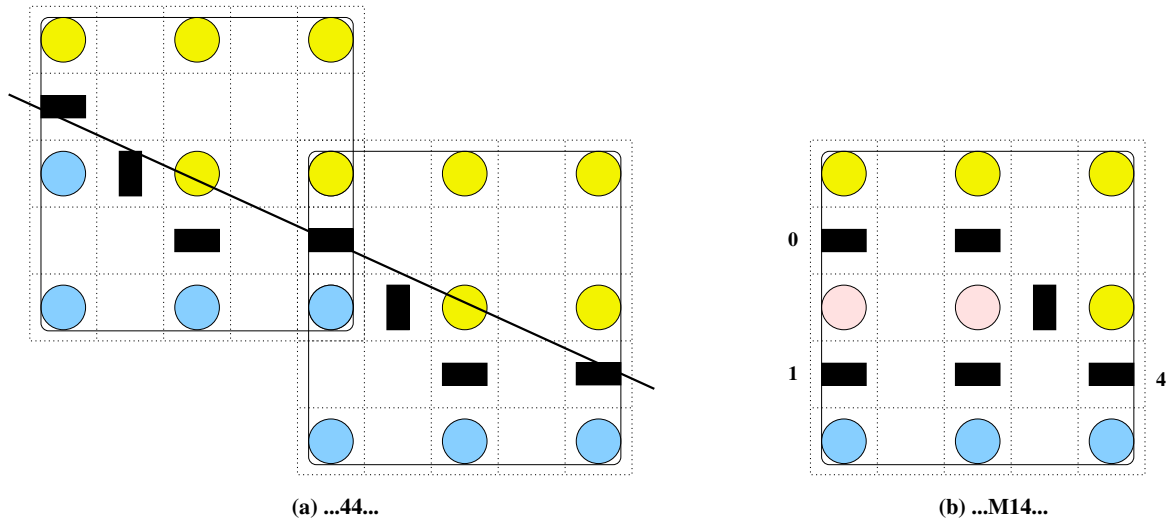

Figure 8.19: (a) Chaining MGCC cells. Each cell is represented by a symbol given by its exit point. In the figure, both cells are encoded using the symbol '4' (b) Representation of a triple point by using a cell with multiple outputs. The cell is represented by a set of symbols composed by all its exit points (in the figure, '1' and '4') preceded by a special symbol 'M'. In this example, output site 1 is called *primary* because it has the maximum length path, and output site 4 is called *secondary*

There are two basic problems in the coding process that can lead to incorrect results at the decoder site:

- The uncertainty in the central pixel of each cell may lead to inconsistencies in the decoded contours. That is, new regions may be created or regions be split so that the decoded contours are no longer consistent with the bitstream (See Figure 8.20). This problem arises since the encoding process is local (done cell by cell).

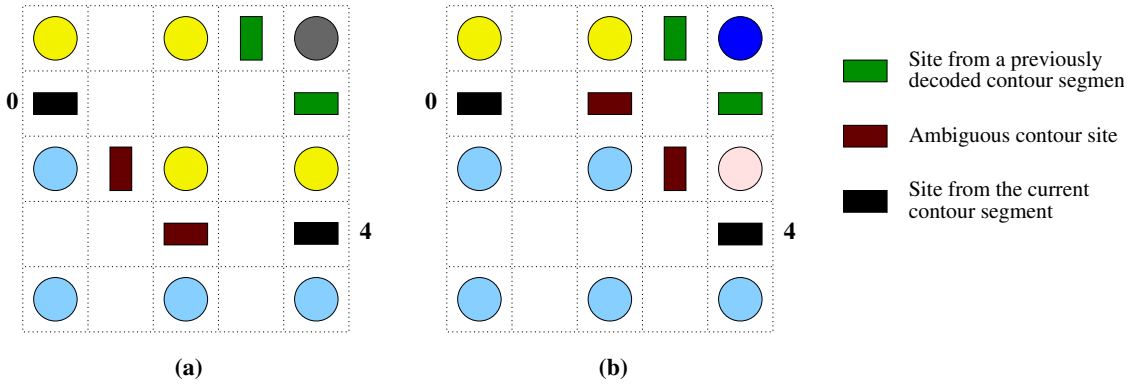

Figure 8.20: Yellow region in (a) is split in (b) because of the uncertainty in the internal contours of the cell.

- Detection of the end of a segment. As in the case of the Chain Code technique described in Section 6.4, the decoder detects an end of segment when an already known contour is reached by the new contour. However, in the MGCC, the final point of a segment can reach a contour that has been decoded as an ambiguous internal cell contour, and thus, can be shifted from its original position.

In [36, 69], both the encoder and the decoder use different marks for sure contour sites (input and output cell contour sites, and some internal cell contour sites in multiple output cells) and for unsure contour sites (the majority of the cell internal contour sites).

The decoder is divided into two main blocks: the Forward decoder and the Backward decoder. Both decoders work at cell level but, in the first case, the decoding is done following the order marked by the symbol chain (the contour tracking order in the encoder) and, in the second case, the inverse order is followed. Inconsistencies are solved by dividing the problem into two steps:

- The Forward decoder fixes all the contour points that are sure in the decoded partition (input points, output points and borders of the image), proposes links between them verifying that the proposed contours are consistent with the bitstream and detects local inconsistencies.

- The Backward decoder solves local as well as global inconsistencies relying only on the marks introduced in the previous step.

The problem in the detection of the end of a segment can be addressed by forcing the end of each segment on a sure point (an input, output or border point) of the previously decoded contours. This can be seen in Figure 8.21, where the closing point is located in one of the exit sites of the cell. Closing points can also be located in one of the internal cell sites.

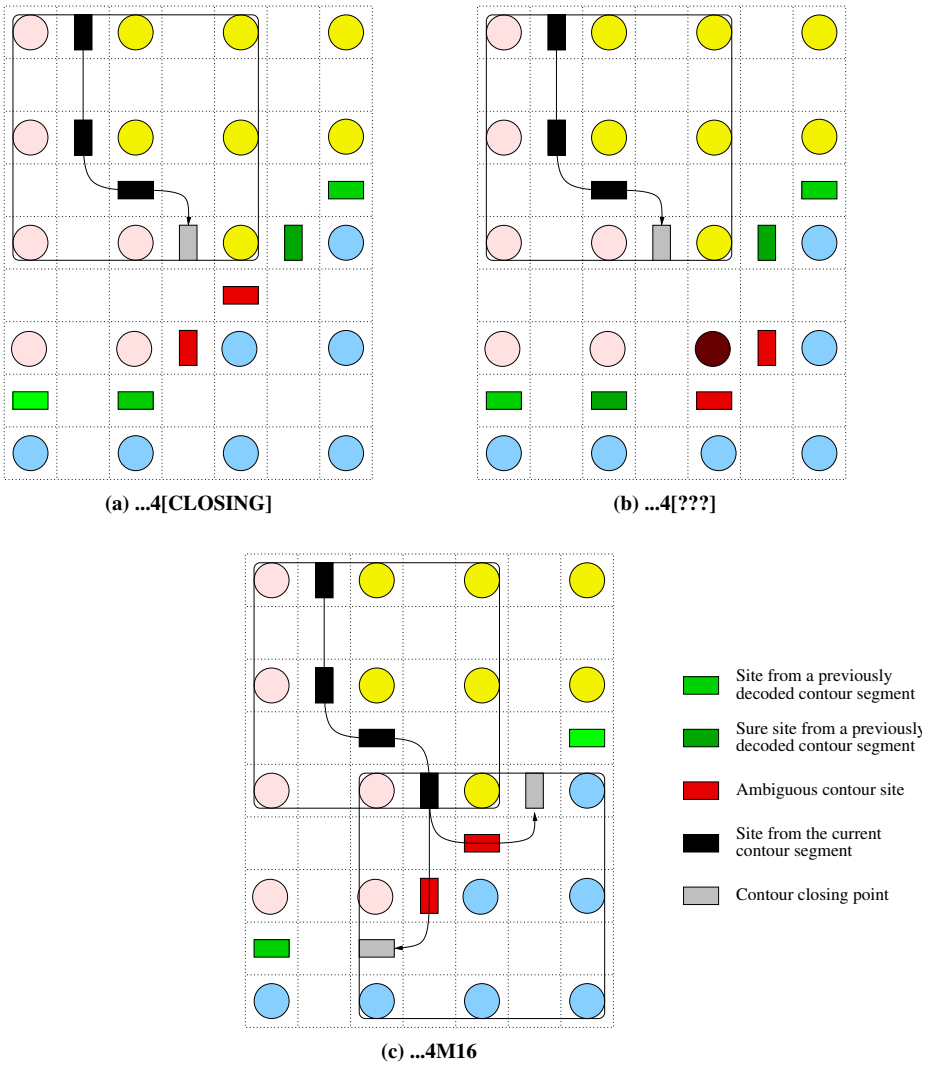

Figure 8.21: At the encoder (a) the symbol 4 would close the contour. However, the decoder may have shift the red contour sites as in (b), preventing the detection of the already decoded contours. The solution is to close always on a sure contour point, adding an extra cell if necessary, as in (c).

By encoding all the closing output points (even if they are already encoded in a previous contour segment), we can guarantee that no regions are merged. In Figure 8.21(c), outputs 1 and 6 are both closing output points and are sent in the symbol stream.

Scalable contour coding with MGCC

The previous implementation of the intra-mode MGCC coding technique can be extended to a scalable-mode coding approach. In this section, the case of having two layers (i.e.: base and enhancement layers) is discussed, but its extension to several layers is straightforward.

Given a partition, a sub-set of its contours may be sent separately, forming a coarse representation (base partition or layer). The remaining contour segments, up to forming the complete partition (enhancement partition or layer), may be sent in a posterior step. This is illustrated in Figure 8.22.

No additional symbols are included in the bitstream of the base layer to allow the possible decoding of the enhancement layer. Therefore, the intra-mode MGCC encoder proposed in Section 8.9.2 is used as well in the scalable mode.

Let's review some important points that apply in the scalable mode:

- Contour segments belonging to the enhancement partition are coded using the same technique as in the base partition.
- The initial points of the enhancement contour segments are directly located on known contours. These are all the contours marked as *sure* during the base layer coding process. Known contour elements are indexed and they are used as initial points.
- In the case of willing to represent the complete partition, the decoding of both layers is interleaved. This way, the base layer is first forward decoded, but its backward decoding is postponed.
- As previously, contour sites (either from the base or enhancement layer) inside the cell that are not linked with the current contour are not taken into account. In addition, contours belonging to the base partition (and thus already encoded) are used as initial or closing points of an enhancement contour segment. In the initial and final cells of a given contour, elements from the base layer may have to be encoded again because removing them can result in the decoder selecting a wrong configuration for the cell. (see discussion in Section 8.9.3).
- In the enhancement layer, the end of a contour segment relies on the contour elements from either the base or the enhancement layers previously marked as *sure* during the tracking process. If the output contour is that of highest priority, the end of the contour segment is reached. In addition, if the output contour is marked as *unsure* (from the

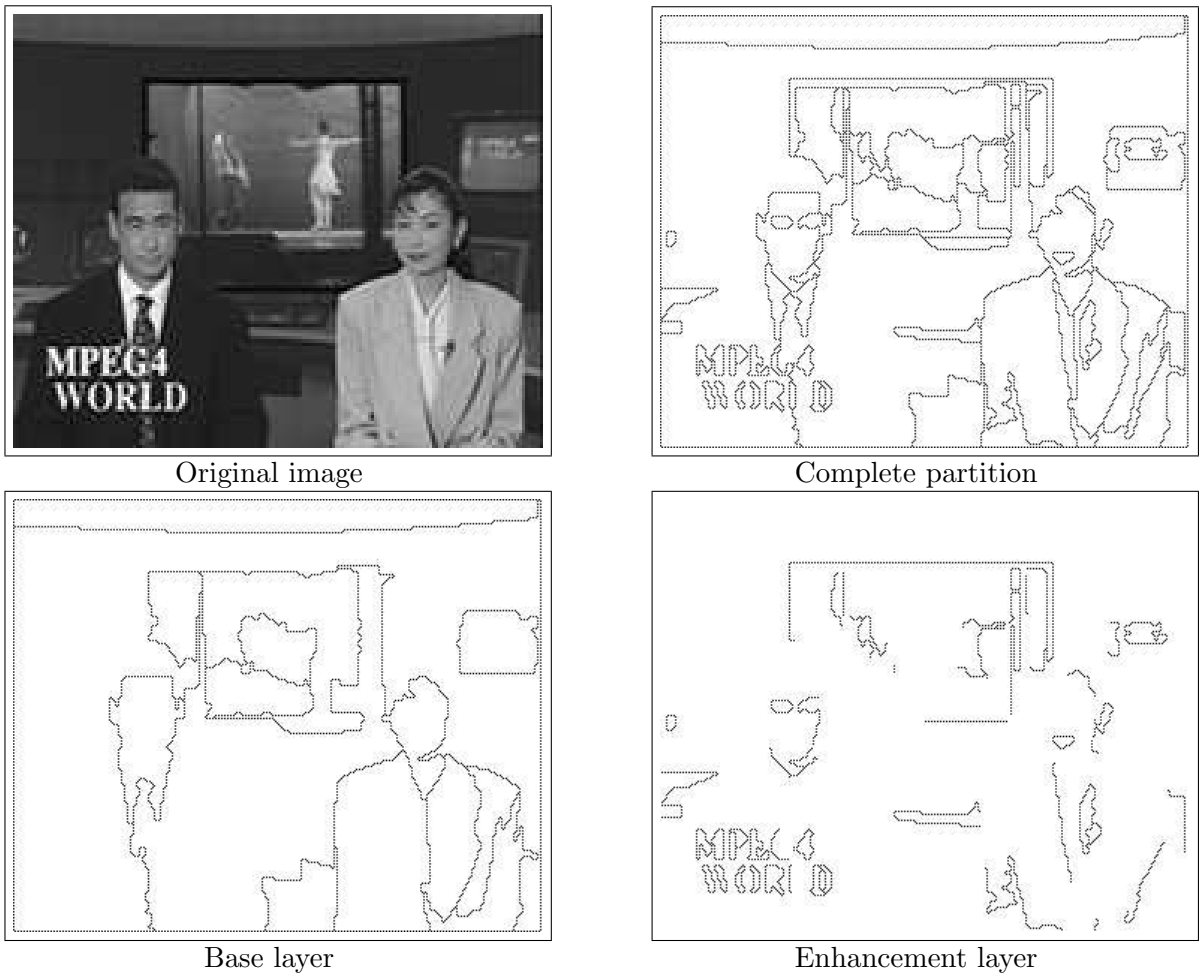

Figure 8.22: Example of scalable contour coding. The base layer can be decoded by itself. The contours sent at the enhancement layer are added to the base layer to form the complete partition for the enhancement layer.

base or the enhancement layers), but any of the following contour sites that can be reach from it is a *sure* one, the contour is ended as well.

- The end of a contour in the inverse tracking is detected when, from the last output contour, the cell created using the cell characteristics does not find a pixel marked with the previous index sure_input_enh. On turn, the end of partition is detected as in the intra-mode MGCC.

Inter-frame contour coding with MGCC

This section contains the details of the variations to be performed on the new implementation of the intra-mode MGCC encoder and decoder to cover the inter-mode coding approach.

The intra-mode MGCC coding approach can be used in the previous inter-mode technique to code the set of intra-regions as well as the additional contour segments forming the error of the compensated partition.

The set of intra-regions forms a partition that can be directly coded using the intra-mode MGCC. The only refinement that can be introduced in the decoding of the intra-regions deals with the uncertain pixels appearing in the center of some cells. This is the case of those cells associated to contours in the boundary between intra-regions and inter-regions. Those pixels are not directly assigned to any region in the intra-mode but they are analyzed in the inter-mode. This way, these pixels are treated as inter-mode pixels and, if one set of motion parameters suits them, they are assigned to the associated region. If there is no set of correct motion parameters, they are assigned to the intra-region.

Contours describing the compensation errors are coded relying on the previously decoded contours. This way, it is a simplified case of the scalable-mode MGCC. Both the transmitter and the receiver have the same information about the set of contours transmitted in intra-mode and the set of motion compensated contours. Thus, this information can play the role of the base layer in the scalable-mode MGCC and the compensation error contours that of the enhancement layer.

Nevertheless, the inter-mode MGCC is simpler than the scalable-mode as, here, the intra-mode contours and the motion compensated contours are perfectly known in the encoder and in the decoder. This way, initial contour segments can be associated to any previous contour point and not only to those marked as inout or belonging to the frame. The same policy can be followed when closing a contour.

8.9.3 Comparison between CC and MGCC

While the coding efficiency of MGCC (lossy) is theoretically superior to CC (lossless), the lossy nature of MGCC results in problems at the contour closing step that result in a performance degradation due to necessary side information. This overhead causes the inter and scalable MGCC technique to perform worse than CC.

This can be efficiently solved in the intra-frame mode. End-of-contour sites that are coded twice can be removed from the chain of symbols before the actual encoding is performed. This is done by storing along with each contour, its position in the chain of symbols. This information is only used at the encoder and thus, it is not sent in the bitstream. When the final site of a closing contour coincides with an already coded site (from a previous multiple cell with pending outputs), it is possible to "go back" in the chain of symbols and to remove the symbol associated with that output.

This approach is not feasible in the inter or enhancement modes when the duplicated contour symbol has been sent in a previous layer. In these cases, the decoding of the multiple

layers is interleaved (this is, all the layers are first forward decoded and the backward decoding does not take place until the end of the process).

In the inter or scalable modes, there exist cells containing contour sites from the current contour as well as contour sites from the reference contours. This results in cells with multiple outputs that are very expensive to code where some of the outputs represent contour sites that were already coded in a previous layer or in a previous frame. One strategy to deal with this situation would be not to consider the output points from the reference contour. However, this is not possible in most cases because the contour closing algorithm relies in sure points from the reference contour (See Figure 8.21).

Additionally, removing these reference contour output points in many cases confuses the decoder that can then select a configuration for the internal contour path that is incompatible with the actual contour being encoded. This will result in many cases in a change in the topology of the partition at the decoder and the original partition would not be recovered. Figure 8.23 shows an example of cell whose decoding is ambiguous, leading to possible partition topology changes at the decoder. If contour points from the reference contour are not sent, the cell in configuration (a) is encoded with a single symbol (3) but then the decoder can not difference between cells (a) and (b). If exit points from the reference contour are sent, the ambiguity at the decoder is eliminated but five symbols are needed to encode the cell (MM356).

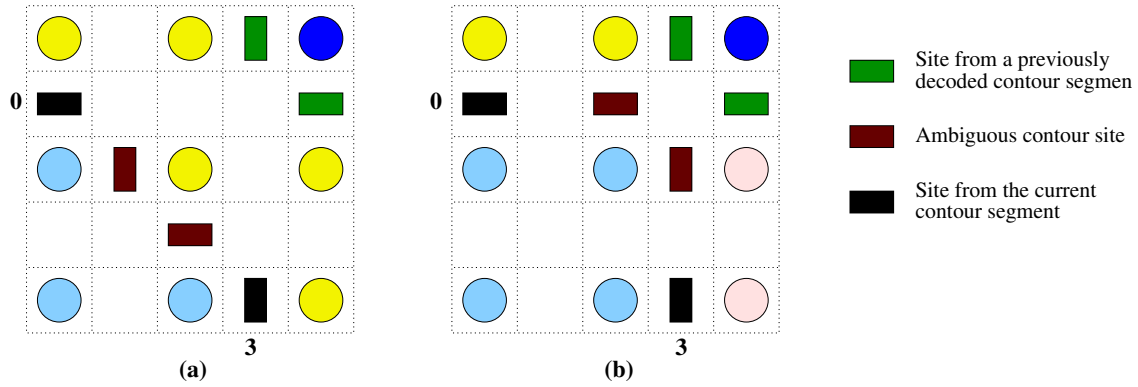

Figure 8.23: In the configuration of cell (a), if the exit points from the reference contour (5 and 6) are not encoded, the decoding is ambiguous between configuration (a) and (b).

Results comparing the intra and inter mode for both Extended CC and Multi-Grid CC are shown in Table 8.2. Scalable mode behaves much similarly to the Inter mode.

As it can be seen, the extra symbols result in an increase in the number of bits used to encode the contour. The conclusion is that it is better to keep with traditional CC in inter and scalable modes. MGCC can be used efficiently for intra mode frames.

	Intra mode		Inter mode	
	CC	MGCC	CC	MGCC
Akiyo (QCIF@30Hz) (25 regs/frame)	1.22 bpce	1.00 bpce	0.13 bpce	0.15 bpce
Akiyo (QCIF@30Hz) (2 regs/frame)	1.21 bpce	0.92 bpce	0.17 bpce	0.19 bpce
Foreman (QCIF@15Hz) (25 regs/frame)	1.33 bpce	1.07 bpce	0.54 bpce	0.56 bpce

Table 8.2: Comparative of the CC vs. MGCC contour coding techniques for Intra- and Inter-frame modes. Results show the average bits per contour element (bpce) after encoding the contours of 150 frames of each sequence.

8.10 Coding of the texture

The texture coding step in the extended system is very similar to the same step in the basic system (See Section 6.5). The principal difference is the introduction of a new coding mode (that will be called *layer intra* from now on), intended to code the error between the original and the base layer coded images. This new mode can be used in the spatial and PSNR scalability modes.

With the new *layer intra* mode, each region in the enhancement layer can be encoded with all the available coding techniques, each one in two modes: inter and layer intra. The *layer intra* mode can be selected when the motion of the region can not be accurately modeled and thus, the compensation error for the region is high.

In spatial scalability, the base and enhancement layers do not have the same resolution (the base layer is a sub-sampled representation of the original image). Thus, to compute the coding error the encoded base layer is up-sampled to the same resolution as the enhancement layer. Then, the error is computed by subtracting this up-sampled image from the original image at full resolution.

In temporal scalability the only difference with respect to the basic system is the handling of uncovered background in the enhancement layer. This point was discussed in Section 8.7.3.

The different texture coding modes are summarized in Figure 8.24.

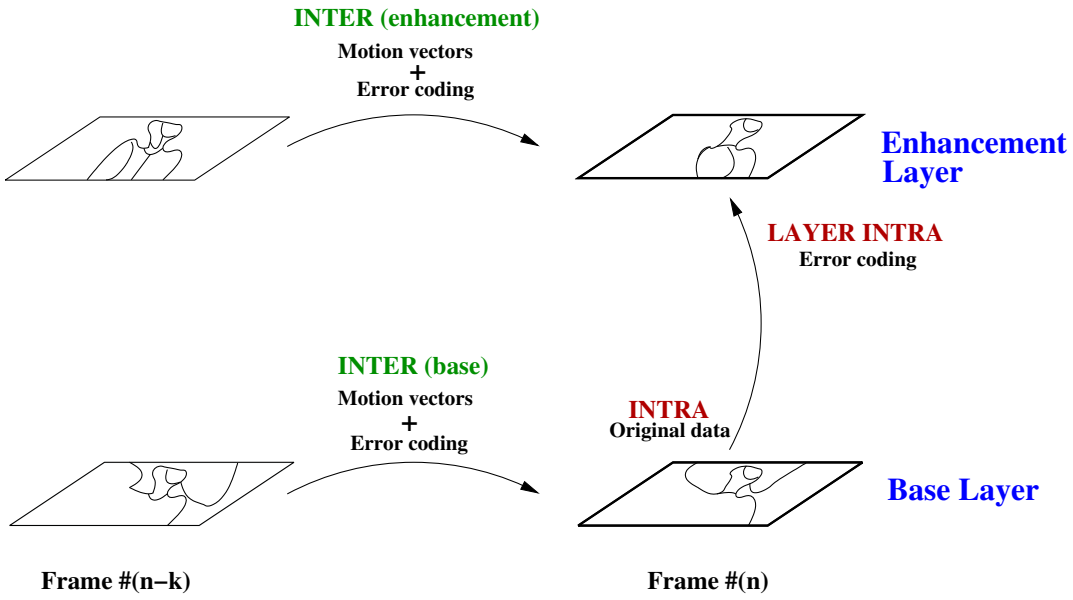

Figure 8.24: Different texture coding modes and information to be sent

8.11 Object Tracking

The possibility to track an object along a sequence is the key to properly define content-based functionalities. In the unsupervised case, tracking allows to relate the information between the objects in the previous frames with the current and future frames. This leads to the creation of Video Object representations; that is, to separately define the temporal evolution of an object.

Tracking an object along a sequence in a segmentation-based coding system is a complex task. The key problems to be solved are: identification, meaningful segmentation and region correspondence.

In the first place, the object must be defined. This can be done in very different ways. The simplest manner is by means of chroma-keying, if one has access to the creation process. Another common way to handle the object definition issue is to restrict the problem to the case of detecting and tracking moving objects [128, 84, 76, 90, 3, 78, 52]. Finally, objects can be defined in a supervised manner, by user's interaction [12, 54, 144]. In this work, it will be assumed that the object has been appropriately defined in the first frame of the sequence.

The second problem to be solved is meaningful segmentation. Segmentation based coding systems perform an adaptive partition of the image resulting in a set of regions with arbitrarily shaped contours. As one of the goals is coding efficiency, the partition consists on homogeneous regions that can be easily coded. Having the image decomposed into a set of regions, it makes sense to use or to adapt this representation to describe objects as sets of regions. The coding system already has to deal with regions of arbitrary shape, so the only modifications necessary are (a) to make sure that there is a set of regions that can describe appropriately the object, and (b) identify which regions in the partition are effectively forming the object.

The third problem to solve is region correspondence: Once the selected object has been properly identified and segmented in the first frame of the sequence, it is necessary to obtain a meaningful partition for each of the following frames of the sequence. Those partitions must reflect the contours of the object in the following frames, and it should be possible to establish the temporal relationship between regions in the two partitions. Classical object tracking techniques [39, 10, 135] use motion as main information. As a result, they may fail in tracking objects having components with different motions (e.g.: a whole human body with parts that may be even static) or objects that are part of a larger object presenting homogeneous motion (e.g.: tacking the publicity on a moving car). Moreover, direct segmentation of the motion information does not yield accurate boundaries. Object contours are more accurately defined relying on color than on motion information. Morphological techniques allow introducing color and texture information in the segmentation process to achieve accurate contour localization. The usual morphological approach is based on the concept of partition projection [72], already

introduced in Section 6.1. A partition of the current image is obtained by projecting the partition of the previous image into the current one. Motion between the previous and the current images is estimated and the previous partition is motion compensated. Projected regions are used as markers or seeds in the current image to conform its actual boundaries.

If the number of regions of the partition is low, projected markers are related to big and possibly complex (non-homogeneous) regions. In this case, the lack of homogeneity of the projected markers makes difficult their growing to conform to actual contours This problem can be avoided using an approach based on two partitions (see Section 6.1). The possibly inhomogeneous previous image partition is re-segmented using spatial criteria before projected. This finer partition results in more homogeneous projected markers that will adapt better to contours in current image. A merging step is performed to eliminate the excess of regions originated by the re-segmentation. This can be done because the relationship between regions in the original and re-segmented partitions is known.

Tracking objects in segmentation-based coding systems provides benefits but also poses specific problems: As the system is aimed at coding efficiency, the partition for each frame is constructed to ensure this goal. This may result in partitions that do not represent appropriately the object. For example, regions belonging to different objects (or to an object and the background) that share the same motion can be merged in a single region if they can be efficiently motion compensated using a single set of motion parameters. It must be ensured that the final partition for the coded frame contains the contours of the object to be tracked. This means to ensure that the contours of the object are present in the Partition Tree (See Section 8.3) and to restrict the optimization algorithm to take only regions under the selection mask (See Section 8.5).

In this work, the object tracking functionality is integrated in the scalable coding mode. That is, the object is only defined and tracked in the enhancement layer. The basic layer is devoted to provide a low bit-rate version of the image and it is only optimized for coding efficiency. Thus, the restriction in the Partition Tree that prevents the decision algorithm to merge regions belonging to the object and regions belonging to the background or other objects is only implemented in the enhancement layer. However, with some modifications, the algorithm can be applied to track an object in non-scalable modes.

In the sequel, the details about the techniques used to track objects along the sequence are presented, and experimental results are provided.

8.11.1 Motion estimation and partition compensation

The temporal link in the coding system is preserved by constructing the Partition Tree from the projected partition. This projected partition represents the temporal evolution of all existing regions from the last frame partition to the current frame. To perform this projection, the motion between two frames is estimated (using a block matching algorithm) and the

previous partition is motion compensated using the estimated motion. The use of more sophisticated motion estimation algorithms (e.g. region-based ones [22]) has proved to provide only slight improvements, while affecting dramatically the computational performance. In some coding applications (i.e. low bit-rate coding) is usual to skip the coding of some frames in order to decrease the final bit-rate. In such cases, motion estimation is performed between all consecutive frames and the final motion vector results from the concatenation of all the intermediate motion vectors. In these cases, a double searching area is proposed for the intermediate motion estimation steps so that the algorithm performs a prediction of the block trajectory. The idea is further illustrated in Figure 8.25, where the motion between frames #(n) and #(n-2) is computed.

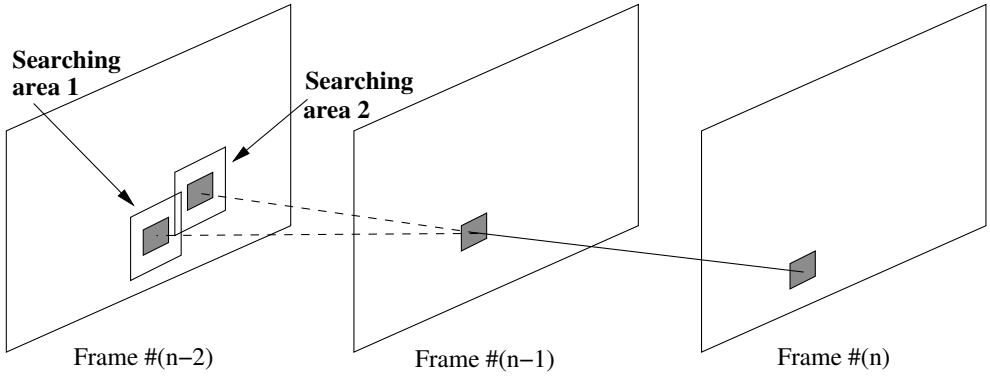

Figure 8.25: Recursive motion estimation: Motion between frames #(n) and #(n-1) is already estimated. In frame #(n-2), two searching areas are used. The first one is centered on the same position as the selected block was centered in frame #(n-1). The second one is centered in the position resulting from the extrapolation of the motion vector from frame #(n) to #(n-1).

The compensated regions used in this step are used as markers (*compensated markers*) giving an estimate of the region positions in the current image. In this work, compensated markers are validated by fitting them into a fine intra partition which represents the boundaries of the current image [70]. This approach increases both the spatial accuracy and the temporal stability of the regions.

If spatial or PSNR scalability are to be used, this process is done twice, first for the basic layer partition alone and then for the sum of the basic and enhancement layer partitions (See Section 8.3). This way, two related projected partitions are obtained, representing the evolution of regions in the basic layer and in the enhancement layers respectively.

8.11.2 Object mask creation

The object to track was defined in the fist frame partition by some appropriated method. As a result, a mapping between the object and one or more regions was obtained. A Look Up Table (LUT) stores which regions of the partition belong to the object. Then, as the system maintains the relationship between regions in successive frames (See Section 6.1), obtaining a representation of the object in the successive frames could be as simple as applying the information stored in the LUT to the projected regions.

However, as new regions can appear, the LUT must be updated through the tracking process. For these new texture regions a decision must be taken whether they belong or not to the object.

While many specific features could be used to check if new regions belong to the object, the one that ensures generic object tracking is the level of acceptable object deformation. Object variations can be due to motion or object deformation. The motion compensated object mask is used to analyze the new texture regions. This is, each new texture region is compared to the compensated object partition and if the new region is mostly (typically, 75%) contained in the compensated object mask, the new region is assumed to belong to the object. If objects are completely generic and can present holes, only regions on the boundary of the estimated object are checked.

Further refinements can be introduced by including *a priori* knowledge of the type of object to track. For instance, information about the number of connected components (this requires changes in the LUT) or about the smoothness of their shape (needs a filtering step) could be used.

8.11.3 Experimental results

In this section, the tracking algorithm is applied to different sequences in order to assess the system performance. The selected color video sequences are in QCIF format (176x144) at 30 Hz. They are selected because they represent different degrees of difficulties to the tracking algorithm. In each case, the mask for the first frame of the sequence defines the object that is to be tracked along the sequence. This mask is provided externally. The initial masks have been obtained by segmenting the image with a large number of regions and then, merging by hand the regions that define the object.

Figure 8.26 shows the results of tracking the head of the woman in the sequence *Mother and daughter*. The head presents rigid but non affine motion (there is some degree of rotation).

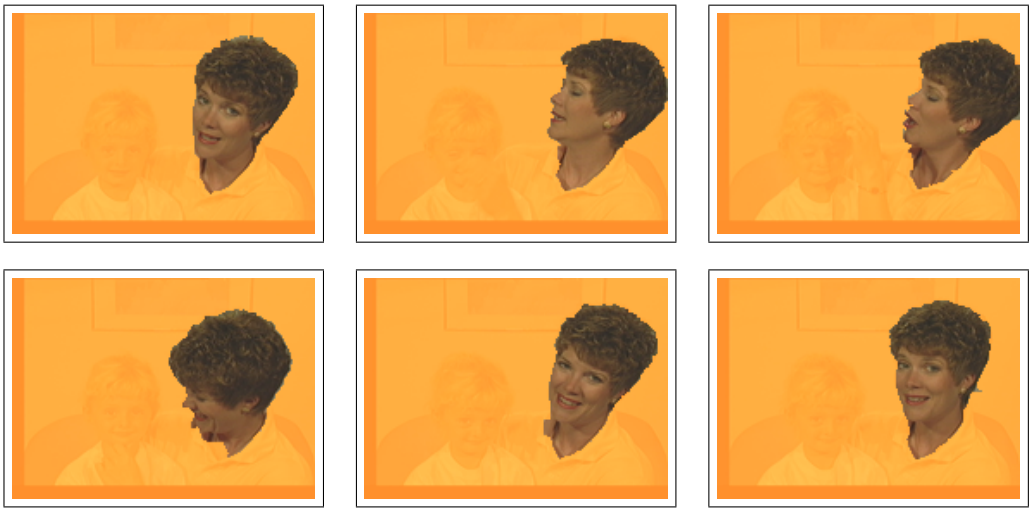

Figure 8.26: The tracking of *Mother and Daughter* frames #0, #60, #120, #180, #240 and #295

To show the ability of the algorithm to cope with objects with non-homogeneous motion, the next experiment shows the tracking of both mother and daughter in the same test sequence (see Figure 8.27). The head of the mother has different motion that those of its daughter. Moreover, the mother's hand appears and disappears in front of the object being tracked. Looking at the figure we can see that, as the hand enters the scene in front of the tracked object, it is included into the object. Later, in frame #120, the hand, now part of the object, is shadowing a portion of the background wall. When the hand disappears, this occluded background region is appropriately removed from the object.

To show the robustness of the algorithm, in this case the tracking has been done using non-consecutive frames: one of every five frames has been used, thus demonstrating the possibility to track objects even at very low bitrates.

Figure 8.27: The tracking of *Mother and Daughter* frames #0, #60, #120, #180, #240 and #295

Figure 8.28 shows the results of tracking the face in the *Foreman* sequence, coded at 30 Hz. This is a difficult sequence because the object to track presents complex motion. There is also camera motion, a new object entering the frame (the hand entering the scene in front of the face in frames #150 to #157) that occludes partially the object of interest, and at the end of the sequence, the camera pans and the object moves outside the frame and disappears. The algorithm is able to tackle with all these circumstances and correctly tracks the face even though it does not present spatial homogeneity and forms part of a larger object with homogeneous motion. When the hand enters the frame, it occludes partially the face of the foreman. After the hand moves out of the image, the algorithm can assign again to the object the regions being uncovered. When the object completely disappears from the scene, the tracking algorithm detects this fact and the tracking ends.

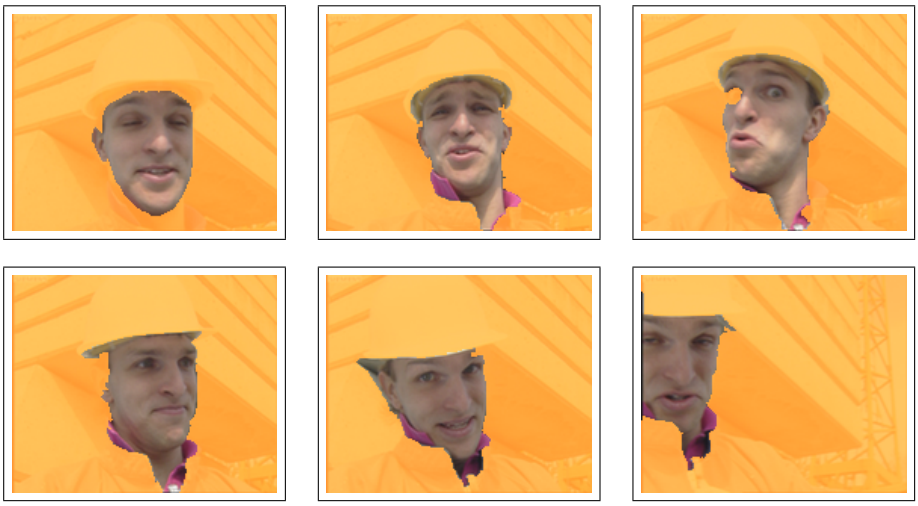

Figure 8.28: The tracking of *Foreman* frames #0, #60, #90, #120, #150 and #180

The *Akiyo* sequence presents small but non-rigid motion (see Figure 8.29). There are moving and still parts in the same object. Another problem is that the woman's hair has a color similar to the background. This causes small errors near its right eye, where part of the object is labeled as background. However, the rest of the hair is correctly separated from the background along all the frames of the sequence.

Figure 8.29: The tracking of *Akiyo* frames #0, #60, #120, #180, #240 and #295

If the tracking is limited to the head in the same sequence, results are slightly better (see Figure 8.30).

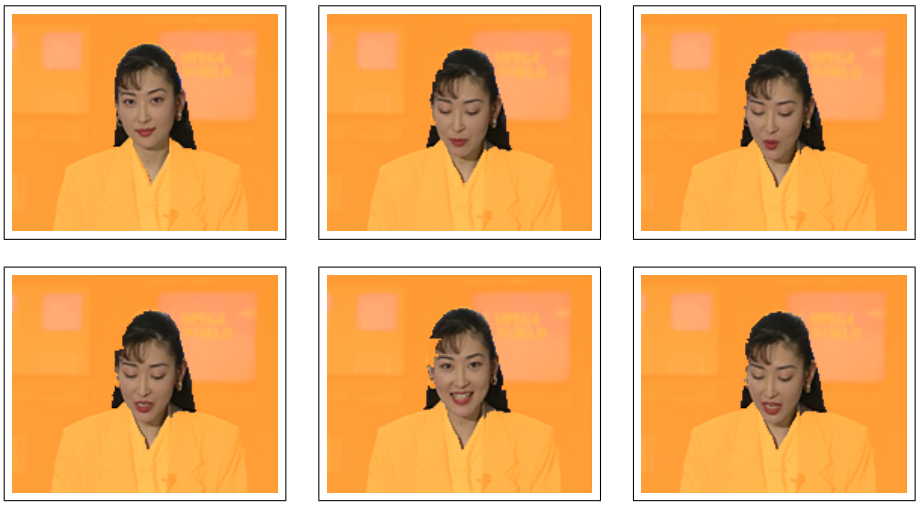

Figure 8.30: The tracking of *Akiyo* frames #0, #60, #120, #180, #240 and #298

The *Carphone* sequence in Figure 8.31 shows an error that the tracking algorithm presents in some situations. While it is able to correctly follow the movements of the head, a portion of the background is identified incorrectly as belonging to the object. The error occurs because at some point, a region belonging to the object and a region from the background having similar color can be incorrectly merged. Once this happens in one frame, the error is propagated to the following frames.

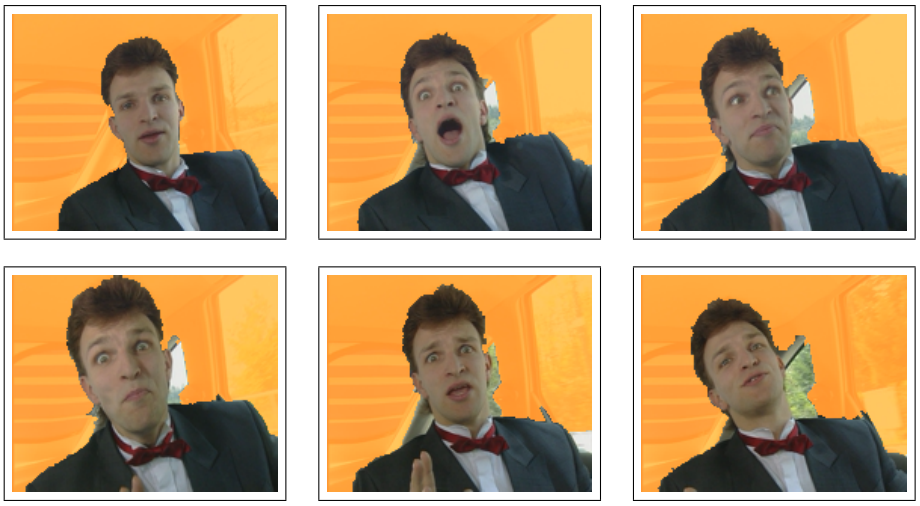

Figure 8.31: The tracking of *Carphone* frames #0, #60, #120, #180, #240 and #295

Chapter 9

Experimental results for the scalable coder

The purpose of this section is to show the operating mode and the performance of the encoder with the various supported types of scalability (spatial, PSNR and temporal, either in full frame or object mode). Results for different sequences at different bit-rates are presented. As the coding system is very complex, with many combinations of operational modes and scalability types, only the more meaningful results are presented here. More experimental results can be found at our web site [83].

The purpose of this work is to show the ability of a region-based video coding system to provide content-based functionalities rather than to compete in coding efficiency with the newest standards. Nevertheless, when possible, results are compared with an MPEG-4 encoder. The choice of the MPEG-4 standard is because it is an established industry standard and it has similar features (different scalability modes for full frame and arbitrary shaped objects) to the system being presented in this work so that a meaningful comparison is possible.

The first sections (9.1, 9.2 and 9.3) show results obtained with the independent optimization algorithm. This algorithm does not take into account dependencies between different frames or scalability layers and, thus, optimization is done separately for each frame and scalability layer. Section 9.4 shows the results obtained with the dependent optimization algorithm; that is, taking into account dependencies between frames and scalability layers, and performing the optimization step globally on those groups of frames/scalability layers.

9.1 PSNR scalability

Analysis of the Projection and the Partition Tree construction

Let us start with a review of the Projection step (See Section 8.3). Figure 9.1 shows the

details of the double projection construction (see Section 8.3) for frame #105 of the *News* sequence. The base layer projection is used to give the appropriate labels to the regions of the final base layer partition. Given that in XSESAME the base layer does not carry content-based functionalities, this projected partition may not contain all the contours of the object. The enhancement layer projection is used to build the splitting levels of the Partition Tree. To ensure that all relevant contours are present, it is constructed by projecting the sum of the base and enhancement layer partitions of the previous frame. If an external mask defining the object is provided for each frame, it is also used to drive the projection step.

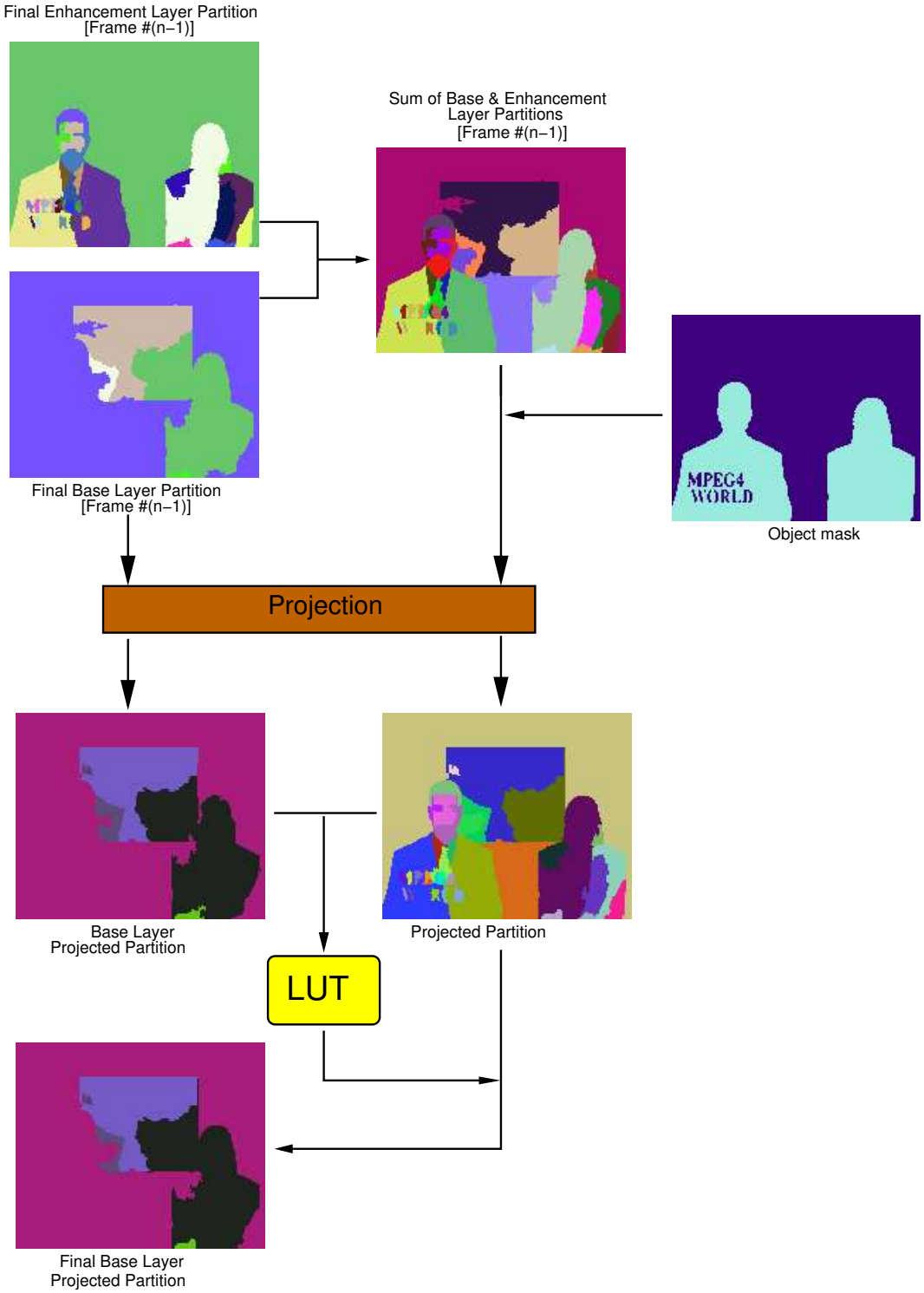

Figure 9.1: Partition compensation process in PSNR scalability

Note that, at first, the base and enhancement layer projections are constructed independently. As it can be seen in Figure 9.1, the two independent projection processes lead in most cases to partitions whose contours do not perfectly match. As contour coherence between all partitions is fundamental for the construction of the Partition Tree, a Look Up Table relating the regions in the two partitions is constructed, and the base layer projected partition is rebuild using this LUT.

Figure 9.2 shows the construction of the base and enhancement layer Partition Tree's for the frame #48 of the *Children* sequence. As it was shown in Section 8.5, the enhancement layer Partition Tree is created by re-using unused branches in the lower levels of the base layer Partition Tree.

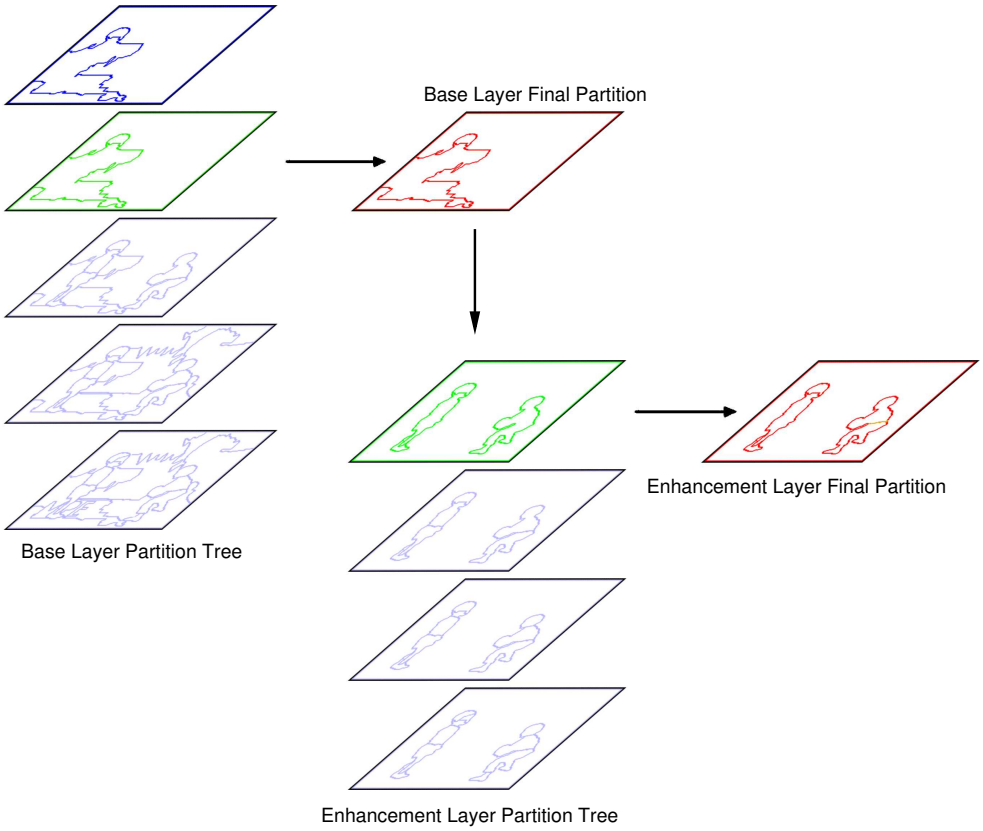

Figure 9.2: Base and enhancement layers PTs for the frame #48 of the Children sequence.

In the figure we can see that the base layer Partition Tree is asymmetric, that is, it has a different number of merging and splitting levels. This is due to the tree balancing algorithm described in Section 8.4.1. In this case, as the number of regions in the projected partition is small, more splitting levels are used at the expense of the merging levels. Note that, in this case, the base layer final partition coincides with the projected partition so it does not use

the extra 'region resolution' given by the additional splitting levels. However, these levels are essential in the enhancement layer Partition Tree to provide a broad choice of regions to the Decision algorithm.

Two examples of the final base and enhancement layer partitions are shown in Figure 9.3. In the first example (top row), the base layer does not contain all the contours of the object (the two children). In the enhancement layer, the necessary contours are introduced so only the selected objects are considered. Additionally, the regions defining the objects are refined by introducing new regions (the left child foot). In the second case (bottom row), the contours of the object (the two anchor persons) are already present in the base layer partition. The Decision algorithm has selected not to re-segment the regions in the base layer final partition to form the enhancement layer partition. Therefore, the PSNR improvement of the semantic object is achieved by refining its texture through residual error coding. The background segmentation differs in the two partitions because in the enhancement layer only regions inside the objects are taken into account.

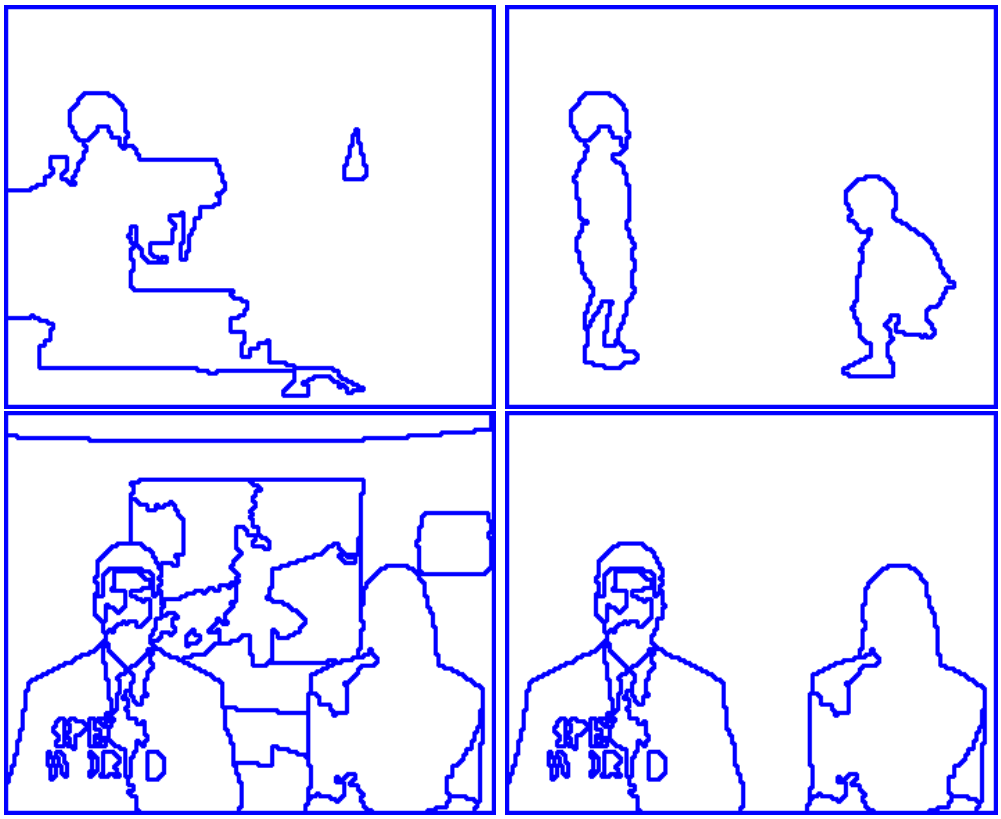

Figure 9.3: Final base and enhancement layer partitions for frame #45 of the News sequence (top row) and for #90 of the News sequence (bottom row)

As it can be seen in the first example (top row on Figure 9.3), base layer partitions do not necessarily convey semantic information. In this case, the goal is coding efficiency and so the resulting partition may or may not represent moving objects.

Results for PSNR scalability

An example of the performance of the layered coder in full frame scalability mode is shown in Figure 9.4. For the *Carphone* and *Akiyo* sequences (QCIF@30Hz), the PSNR at four different bit-rates is presented for the base and enhancement layers. The different performance between the curves for the base and enhancement layers show the bit-rate overhead inherent to layered scalability. This performance gap arises from the fact that in the base layer the actual image data is being encoded, while in the enhancement layer the data being sent is the residual error, a highly non-homogeneous signal.

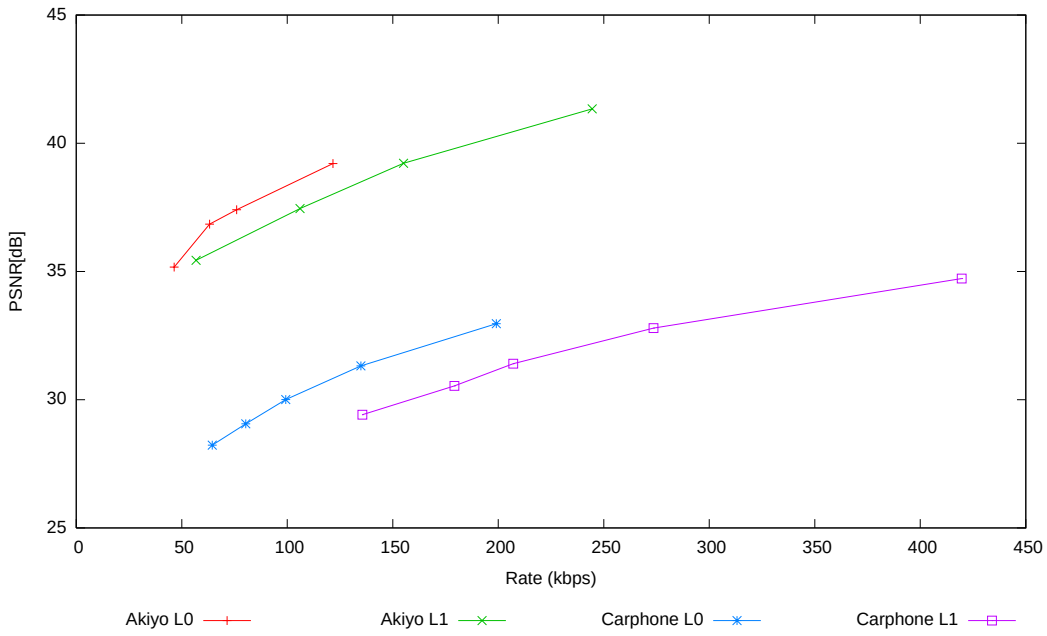

Figure 9.4: R-D curves for the Carphone and Akiyo sequences (QCIF@30Hz) using full frame scalability

Figure 9.5 shows the performance of the codec in object based scalability for the *Weather* sequence (QCIF@30Hz). PSNR values at four different bitrates are plotted for the base and enhancement layers. In object scalability mode, all the available bit-rate is devoted to code the regions corresponding to the selected object while the background is taken without changes from the base layer. The PSNR results are represented for the full enhancement layer and the selected object only.

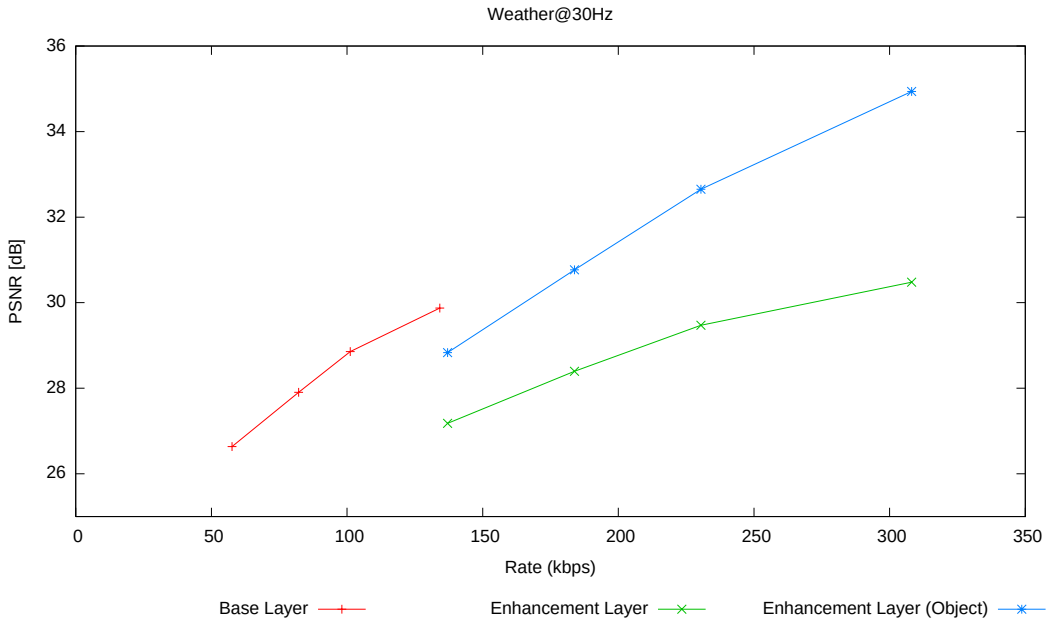

Figure 9.5: R-D curves for the Weather sequence (QCIF@30Hz) using object functionalities mode

Figure 9.6 compares the performance of the scalable encoder with the MoMuSys implementation of MPEG-4. Note that, while the raw performance of the segmentation-based encoder can not match the performance of the MoMuSys encoder (see a more detailed comment on this subject in Section 6.6), the proposed coding system is very well suited for content-based functionalities.

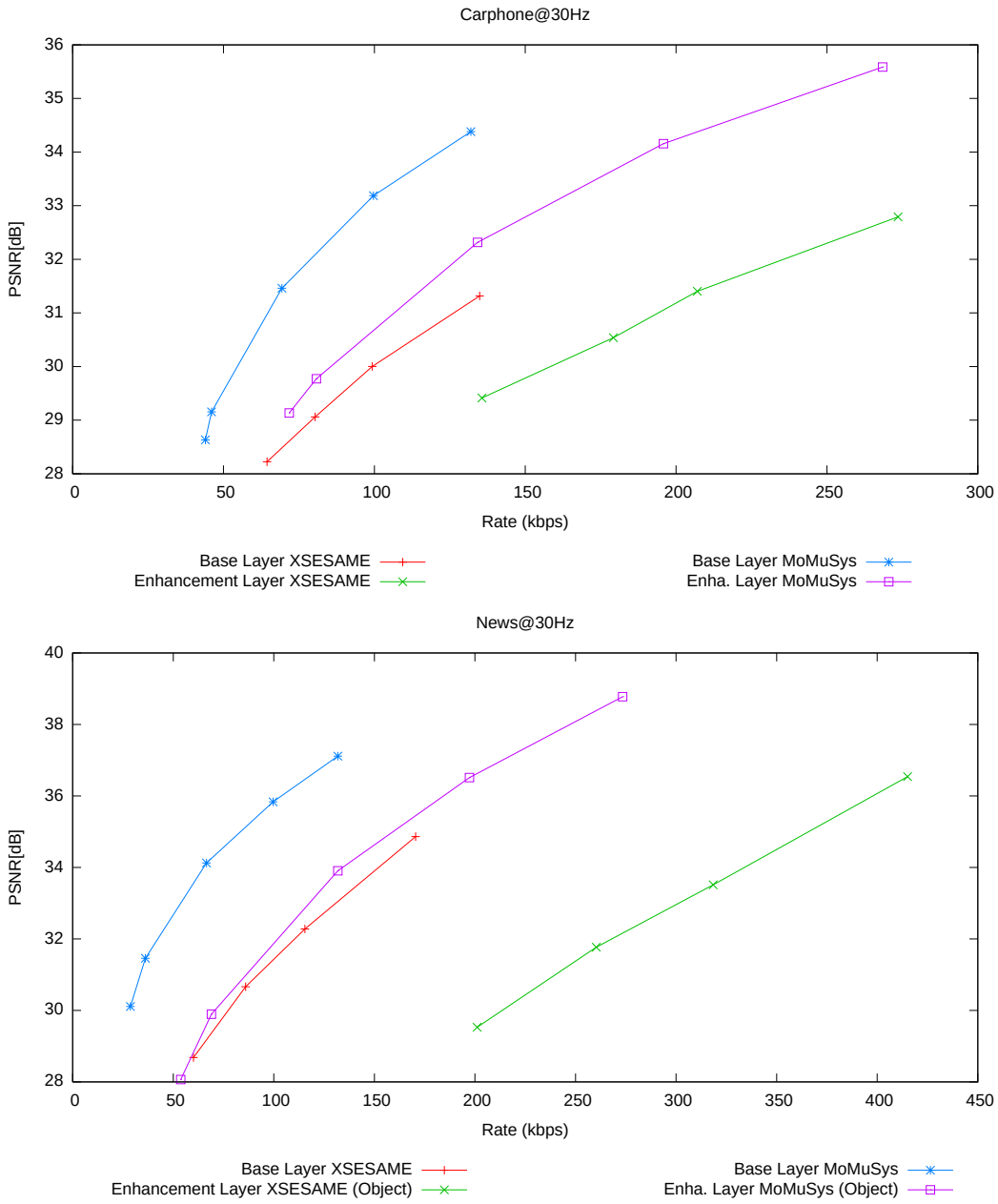

Figure 9.6: Comparison between XSESAME and MoMuSys coders for full frame and object functionalities modes in PSNR scalability

Layered scalability introduces bit-rate overhead with respect to single layer coding. That is, when coding the same sequence both in single layer mode and in layered scalability mode, the coding cost to reach the same decoded quality is higher in the scalable case. However, in the scalable case two layers at different qualities/resolution are obtained. Obtaining the same functionality in single layer mode requires encoding twice the same sequence and sending separately the unrelated streams. In this case, the cost of the multicast approach is always higher than the cost of the scalable encoder.

To measure such overhead, the same sequence has been encoded twice, once in single layer mode (no scalability) and once in scalable mode. The coding parameters are chosen so that the quality (PSNR) at the enhancement layer in the scalable coding is the same as the quality in non-scalable mode. Bit-rate overhead is obtained by comparing the resulting bitstreams necessary to encode the sequence in the two experiments. Numerical results are given in Table 9.1.

Sequence	Single Layer (no scalability)	Base Layer	Base+Enhanc. Layers	Overhead
News@10Hz [33.8dB]	92.4 kbps	65.6 kbps	130.6 kbps	41 %
News@15Hz [33.2dB]	98.1 kbps	67.8 kbps	129.8 kbps	32 %
News@30Hz [31.5dB]	97.3 kbps	62.2 kbps	127.4 kbps	31 %
Weather@10Hz [31.9 dB]	98.4 kbps	65.6 kbps	129.9 kbps	32 %
Weather@15Hz [29.5 dB]	89.7 kbps	67.8 kbps	126.9 kbps	41 %
Weather@30Hz [28.5 dB]	101.4 kbps	54.0 kbps	129.1 kbps	27 %

Table 9.1: PSNR scalability overhead for various test sequences coded at different frame-rates

The overhead figures are similar to those obtained with the MoMuSys encoder in similar tests (around 40%).

Figure 9.7 shows a graphical representation of these results. To complete the results, the figure also shows the cost of generating two separate streams (multicast) with the non-scalable coder. As it can be seen, while the scalable encoder suffers from appreciable bit-rate overhead, ranging from 10% to 40%, the total encoding cost is always less than the cost of multicast. The bit-rate overhead magnitude is comparable to the overhead in other layered-scalability encoders such as MPEG-4.

Figure 9.8 shows the evolution of PSNR through all the frames of the sequence. Note that the rate is not strictly constant because the set of techniques/quantizers can not always ensure that an exact optimal solution exists for the constrained problem. In these cases, the solution selected is the one that gives the bitrate closer to the given budget.

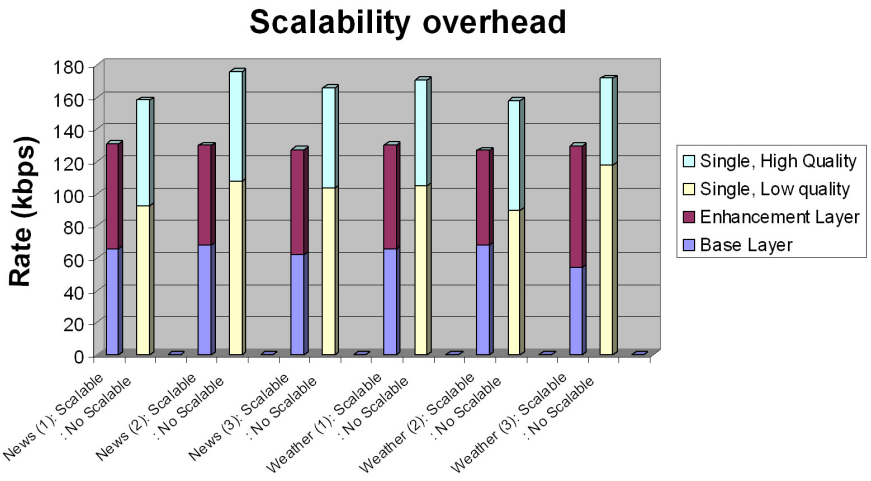

Figure 9.7: Comparison between single layer mode and PSNR scalable mode. PSNR at the enhancement layer in the scalable coding is the same as in non-scalable, high quality mode. Left columns show the cost of scalable coding; right columns show the cost of a multicast approach: Two independent streams, one of high quality and one of low quality.

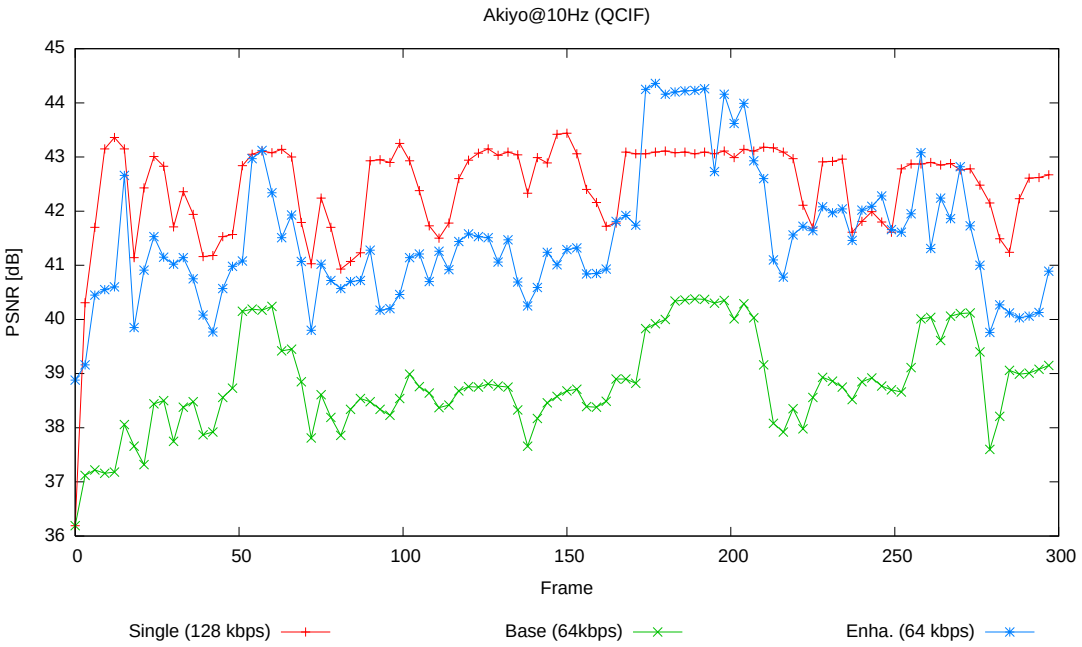

Figure 9.8: Comparison between PSNR full frame scalability and single layer coding

Examples of decoded images in the PSNR scalability mode can be seen in Figures 9.9 and 9.10 for full frame mode (*News* sequence, CIF@30Hz). The figure shows, from left to right, the original image and the base and enhancement layers. Coding is done using 256kbps in the base and enhancement layers.

Full frame mode - Summary of results: The behavior of the decision algorithm is similar to the single layer mode discussed in Section 6.6. Usually, partitions with a low number of regions are selected (even with only one region covering the entire frame), both in the base and in the enhancement layer. In many cases, the partition in the enhancement layer is the same as in the base layer and the improvement is achieved by refining the texture through residual error coding.

In some cases, the decision algorithm can select improving the enhancement layer by splitting regions in the base layer partition. This is the case of row #3 in Figure 9.9 and rows #2 and #3 in Figure 9.10. This is more common in sequences where motion is important.

To assess the use of the texture coding techniques, the total number of regions in all the frames of several test sequences has been analyzed. By sheer numbers, the most selected coding techniques (both in the base and enhancement layers) are the ones that do not involve residual error coding: that is, texture compensation only (in inter-frame mode) and copying directly region texture from the base layer (in layer intra mode). These techniques are typically used for small regions.

For the base layer, inter-frame techniques are always more selected than intra-frame techniques in all the test sequences analyzed (*News, Akiyo, Foreman, Mother and daughter, Carphone, Children, Weather, Dancer, Coastguard*). For small regions, occasionally the gray level average of the regions is selected. In sequences with a large motion or when the frame-rate is decreased, the use of intra-mode techniques becomes more noticeable.

In the enhancement layer for full frame mode, inter-frame techniques and layer intra techniques are used similarly, with a small advantage for inter-frame techniques. It is also interesting to note the effects of bit-rate on the number of regions, specially at the base layer. An increase in the bit-rate usually results in a decrease in the average number of regions per partition. This is because, at low bit-rates, techniques involving no residual error-coding are widely used (i.e. only compensation or copying texture from the base layer). These techniques allow the presence of many small regions due to its low cost. However, at high bit-rates, the low quality of these techniques is inadequate for the given R-D operating point. Obtaining good quality in small regions is then less effective from a R-D point of view than selecting larger regions.