Part III

Optimal strategies for content-based functionalities in a segmentation-based coding system

Chapter 7

Scalable optimal coding in a segmentation-based coding system

7.1 Introduction

In this chapter, the problem of video coding optimization is tackled in the framework of a system that can address content-based functionalities. The focus will be in the various types of content-based scalability and object tracking.

In a scalable coding system, the optimization problem to solve is far more complex because the enhancement layers have coding dependencies on the base layer. The solution implies finding the optimal partition and the set of quantizers for both the base and the enhancement layers. Moreover, due to the coding dependencies of the enhancement layer with respect to the base layer, the partition and the set of quantizers of the enhancement layer will depend on the decisions made on the base layer.

While the basic system presented in Chapter 5 uses an independent optimization approach, in this extended system two different optimization algorithms are presented: The first one is an independent algorithm that considers each frame or scalability layer *by itself*, without taking into account dependencies with other frames/scalability layers. The second one is a dependent optimization algorithm that considers groups of scalability layers (or groups of frames) as a whole, and performs the optimization step globally on these groups.

This chapter is organized as follows: First, a formal description of the scalable coding optimization problem in its various formulations is given. An algorithm to deal with this problem in its independent and dependent formulations is detailed. Then, a complete video coding system using the proposed algorithm is presented. This system has two different modes of operation: an unsupervised mode, aimed at coding efficiency, without content-based functionalities, and a supervised mode, where the enhancement layer can deal with content-based functionalities.

7.2 Problem Formulation

The purpose of this section is to give a formal description of the problem of *coding a video source with several scalability layers, such that each layer has associated a fixed bit-rate or fixed quality*. This problem is analyzed within the framework of segmentation-based video coding systems, and the purpose is to find the best possible solutions on an operational Rate-Distortion sense.

This problem will be faced as an extension of the problem tackled in Section 5.2. The main differences that arise from this new framework are:

- **Content-based Scalability**: The system must deal with content-based scalability. This is, all the bit budget available for a given enhancement layer is spent on the coding of the selected object(s).

  First, for the base layer, a multi-resolution set of partitions, that are organized in a tree-like structure, is built. This set of partitions is used as a basis to select the final partition. The base layer is intended to provide a basic representation of the current frame so the selected objects may not be handled there differently from the rest of the frame. To be able to refine a given object (content-based scalability) in the enhancement layer, a subset of this partition-tree is used to construct the enhancement-layer partition (See Figure 7.1). Note that this means that the existing dependencies respect to the base layer are not only in the texture coding step, but also in the partition construction and coding steps as well.

  Another point to have in mind is that, to enable the system to be able to deal with semantic objects instead of regions, it is necessary to introduce some restrictions on the construction of the partition (see Section 8.11). This aspect does not have a high impact on the optimization problem, as it does not affect the structure of the optimization algorithm and the only effects are a slight performance loss due to the fact that it is possible that some R-D working points are unreachable due to these restrictions.

- **Object Tracking**: The system must also support other content-based functionalities, such as object tracking.

  The tracking functionality affects the structure of the coding algorithm in the sense of restricting the periodicity of intra coded frames that are used to refresh the information at the receiver site. The effect of these intra coded frames is to break the temporal link between the regions and objects in the successive frames, thus avoiding a proper tracking of these regions and objects. If intra coded frames are to be used, an extra region-matching step can be used in order to relate regions in the intra coded frame with regions in the previous frame.

The optimization problem can be tackled in many ways. One is to ensure that every scalability layer in each frame is optimal *by itself*; this is, without taking into account the influence of the decisions taken in the coding of previous frames or previous layers. This independent optimization approach can only provide local optimality, but it is used in practice for simplicity reasons. Another way to solve this optimization problem is to consider the dependencies introduced by the motion compensation step or by the scalability algorithm. This second approach can improve the coding results at the expense of increasing the computing complexity.

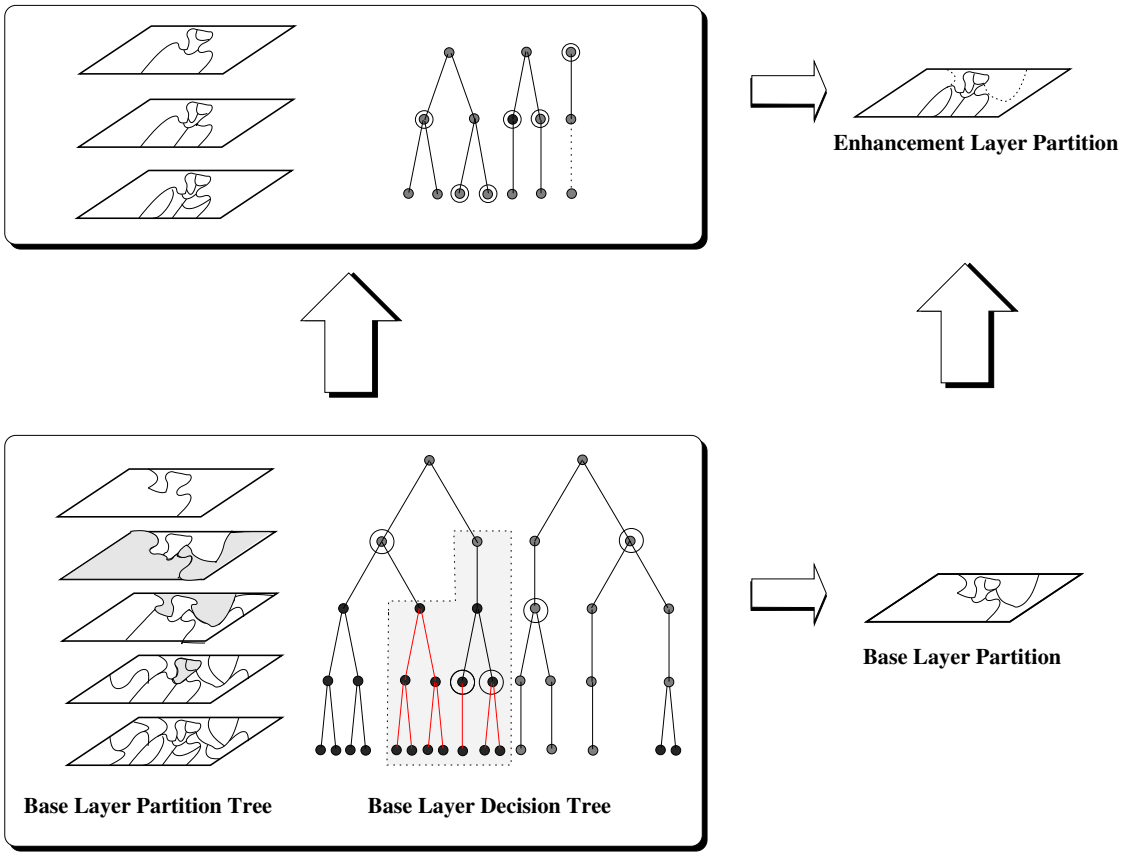

Figure 7.1: Scalable problem. In the solution adopted in this work, the base layer Decision Tree is pruned, and the branches are used to build a new Decision Tree for the enhancement layer. Only the nodes and branches representing the regions of the selected object are used.

7.2.1 Independent optimization

In this case, each scalability layer is optimized separately. This means that a bit budget R_{budget_i} is given for each layer L_i. For each layer, the problem is identical to the one described in Section 5.2. For each scalability layer, the constrained problem to solve is:

$$\min_{P_i \in \mathcal{P}_i, Q_i \in \mathcal{Q}} D_i(P_i, Q_i) \quad \text{where} \quad R_i(P_i, Q_i) \leq R_{budget_i} \tag{7.1}$$

and the respective unconstrained problems is:

$$\min_{Q_i \in \mathcal{Q}, P_i \in \mathcal{P}} J_i(P_i, Q_i) \quad \text{where} \quad J_i(P_i, Q_i) = D_i(P_i, Q_i) + \lambda_i R_i(P_i, Q_i) \tag{7.2}$$

The same bit-allocation algorithm presented in Section 5.2 can be used separately for the different scalability layers. In short, the process is to construct first the base layer using the same process as in the non-scalable mode, optimizing for the base layer bit-budget, and then, create a new Decision Tree for the enhancement layer as a subset of the base layer Partition Tree. That is, once the base layer partition has been created, the base layer Decision Tree is pruned to form the enhancement layer Decision Tree. Then, the optimization algorithm is used to optimize for the enhancement layer bit budget. As dependencies between scalability layers are ignored this optimization is performed separately from the optimization process in the base layer. This process can be iterated for all the enhancement layers.

Complete details of these processes are given in Chapter 8.

7.2.2 Dependent optimization

Motion compensation based coding systems with scalability support usually have temporal and spatial dependencies. In the case of segmentation-based coding systems, temporal dependency arises from the fact that the partition for the current frame depends on the partition of the last frame (partition projection). In the same way, the quality of the reconstructed image depends on the quantizer levels used in the current and previous frames (motion compensation). The spatial dependencies can be found in the construction of the scalability layers, where both the partition and the quality of a given layer depend on the previous coded layers.

The dependent problem can be stated as follows: For two sets of regions (for example, two objects, two full frames or two scalability layers), where D_1, R_1, Q_1, P_1 and D_2, R_2, Q_2, P_2 represent the distortion, bit-rate, quantizers and partition of each set of regions respectively, the global optimization problem to solve can be stated as:

$$\min_{P_1, P_2 \in \mathcal{P}, Q_1, Q_2 \in \mathcal{Q}} [D_1(P_1, Q_1) + D_2(P_1, P_2, Q_1, Q_2)]$$

$$\text{where} \quad R_1(P_1, Q_1) + R_2(P_1, P_2, Q_1, Q_2) \leq R_{budget} \tag{7.3}$$

and by Lagrangian relaxation, can be converted into:

$$\min_{P_1,P_2,Q_1,Q_2} [J_1(P_1, Q_1) + J_2(P_1, P_2, Q_1, Q_2)] \tag{7.4}$$

In [104] a similar formulation was used to solve two dependent bit allocation problems: one with temporal dependencies — the optimal coding for a full MPEG GOP —, and the other with spatial dependencies — a case of pyramidal coding.

In the first case, coding optimality is achieved by selecting the appropriated texture quantization parameters (at the frame level) so that the sum of the distortion of the frames of the GOP is minimal for a given bit budget. Temporal dependencies introduced by the motion compensation process are addressed by organizing the dependency tree in the form of a trellis. The 'states' of the trellis represent the quantization choices for the *independently coded* I frames, while the 'branches' denote the quantizer choices associated with B frames (in this formulation, P frames are omitted). The trellis is populated with the Lagrangian costs (for a fixed λ) associated with the quantizers for each frame. The optimal quantizer choice for each frame is given by the optimal path across the trellis, this is, the one with minimum total cost across all trellis paths. The independently coded I frames decouple the B frame pairs one from another, breaking temporal dependency at periodic intervals. The Viterbi algorithm [29] is used to find the minimum cost path through the trellis. The monotonicity properties of the R-D curves of the dependent components (frames) are exploited to formulate pruning conditions to eliminate suboptimal points. This allows to obtain optimal and near-optimal solutions in a fast way.

Although this problem is similar to the one addressed in this thesis, with similar formulation and goals, the approach followed in [104] is not feasible because in the case of a SBCS the segmentation process must be also considered to obtain optimal results. That is, in addition to the selection of the quantizer level for each region, the optimal partition for the image has to be constructed. This implies computing every possible partition (from the set of regions present in the Partition Tree) for each scalability layer, and then coding all the resulting regions with the available set of texture coding techniques/quantizer levels. The complexity of this approach grows exponentially with the number of frames, the number of regions and the number of quantizer choices.

Moreover, in the PSNR problem (optimization of scalability layers, see Section 8.7) an additional constraint is commonly introduced to ensure that the quality of the base layer reaches a minimum, leading to a new and more complex problem:

$$\min_{P_1,P_2\in\mathcal{P},Q_1,Q_2\in\mathcal{Q}}[D_1(P_1,Q_1)+D_2(P_1,P_2,Q_1,Q_2)$$

$$\text{where}\quad R_1(P_1,Q_1)+R_2(P_1,P_2,Q_1,Q_2)\leq R_{budget}$$

$$\text{and}\quad D_1\leq D_{min} \tag{7.5}$$

By Lagrangian relaxation, it can be converted into:

$$\min_{P_1,P_2,Q_1,Q_2}[J_1(P_1,Q_1)+J_2(P_1,P_2,Q_1,Q_2)]$$

$$\text{with}\quad D_1\leq D_{min} \tag{7.6}$$

The support for content-based functionalities has implications on the handling of the temporal dependency issues. It is clear that it is not feasible to consider the temporal dependencies for a whole video sequence, since this would lead to an exponentially complex problem, impossible to solve in practice. In [104], this issue is handled using the fact that MPEG I frames are coded independently, and thus, the temporal dependencies are broken at regular intervals at these frames. However, in the work presented in this thesis, the support for content-based functionalities relies on the region-tracking approach based on a projection of the last coded partition. In this case, the use of intra-frame coding at regular intervals would break region tracking, thus preventing content-based functionalities. However, if the process to generate the set of partitions (projection and construction of the Partition Tree) is carried out as usual on the encoder side, but all the texture and contour coding is done in intra-frame mode, the inter-frame dependencies can be ignored. This way, the support for content-based functionalities is not affected and intra-mode frames can be used to break the temporal dependencies at regular intervals.

Another factor to take into account is the computational complexity of this dependent optimization approach is very high. As mentioned before, it grows exponentially with the number of frames/scalability layers, the number of regions and the number of quantizer choices. This results in the practical impossibility to jointly optimize more than two units (frames or scalability layers). To avoid having too many intra-mode frames just to break the dependencies, different approaches have been followed depending on the type of scalability being used:

- **Spatial and PSNR scalability**: Only the dependencies between the different scalability layers for a given frame are considered. Inter-frame dependencies are ignored.

This means that only the *spatial* dependencies between the base and enhancement layers will be taken into account.

- **Temporal scalability**: Only the dependencies between the base layer and the associated enhancement layers are considered. Dependencies between consecutive base layers are ignored. Dependencies only exist in the *temporal* domain.

The decision to ignore inter-frame dependencies should not affect the genericity of the optimization algorithm since the base layers could be coded in intra-frame mode and a region-matching step could be used to preserve the temporal link between the regions in different frames necessary to deal with content-based functionalities. Developing a region-matching algorithm would be somewhat off-topic in this thesis, where the purpose is to propose a dependent optimization algorithm and to compare its performance with the independent optimization approach.

As it will be shown in Chapter 8, to demonstrate the dependent optimization algorithm, sequences are coded in inter-frame mode (intra-frame coding is used only for the first frame of the sequence). Temporal dependencies between the layers of different frames are ignored in a first approach for simplicity.

The proposed dependent optimization algorithm is adapted from [104], where it was applied to the case of pyramid-based multi-resolution coding. It is based on the fact that, within each frame or scalability layer, all blocs are independent and thus, operate at a constant slope at optimality. For the first frame or layer, the solutions on the convex hull are examined by sweeping the value of the Lagrange multiplier λ_1 from zero to infinity. For each value of λ_1, the Lagrange optimization process is used on the first frame or scalability layer to find a partition of the image, with the optimal quantizer choice for each region. If the resulting rate satisfies that $R_{1,\lambda_{1,i}} \leq R_{budget}$, an optimization process is performed on the second frame or scalability layer in order to minimize its distortion for the bit budget $R_{2,i} = R_{budget} - R_{1,i}$, with a Lagrange multiplier λ_2. At the end of the process, the values that minimize $D_1 + D_2$ are chosen.

The algorithm can be stated as:

Step 1: Starting at $\lambda_{1,0} = 0$, find $R_{1,\lambda_{1,0}}$ and $D_{1,\lambda_{1,0}}$. If $R_{1,\lambda_{1,0}} \leq R_{budget}$, optimize the second frame or scalability layer for the rate $R_{2,0} = R_{budget} - R_{1,0}$ by minimizing the corresponding Lagrangian $J_2(\lambda_{2,0})$. Store the resulting values of $R_{1,0}, D_{1,0}, R_{2,0}, D_{2,0}$. Otherwise, sweep $\lambda_{1,j}$ until a feasible value is found.

Step 2: Repeat the previous step by sweeping $\lambda_{1,j}$ toward infinity, until $D_{1,\lambda_{1,j}} \geq D_{1max}$. The monotonicity properties ensure that for larger values of $\lambda_{1,j}$, the resulting solutions will not satisfy the restriction stated by equation 7.6 and therefore they should not be further

analyzed.

Step 3: Choose the solution that minimizes $D_1 + D_2$.

This process is summarized in Table 7.1. The framed row shows the optimal working point, where $D_1 + D_2$ is minimum. It is easy to see that this algorithm can also be used in the more general case where the total distortion to be minimized is a weighted average of the distortions of the different layers.

$$\min_{P_1,P_2\in\mathcal{P},Q_1,Q_2\in\mathcal{Q}} [\omega_1 D_1(P_1,Q_1) + \omega_2 D_2(P_1,P_2,Q_1,Q_2)] \tag{7.7}$$

The computational complexity of this approach is still high, though feasible for groups of two frames or scalability layers.

The complete algorithm is detailed in Chapter 8.

Base layer	**Enhancement layer**	
$\lambda_{1,0} = 0 \Rightarrow R_{1,0}, D_{1,0} \longrightarrow$	$R_{2,0} = R_{budget} - R_{1,0} \Rightarrow \lambda_{2,0}, D_{2,0}$	$D_{1,0} + D_{2,0}$
$\lambda_{1,1} = \lambda_{1,0} + 1 \triangle \lambda \Rightarrow R_{1,1}, D_{1,1} \longrightarrow$	$R_{2,1} = R_{budget} - R_{1,1} \Rightarrow \lambda_{2,1}, D_{2,1}$	$D_{1,1} + D_{2,1}$
...		
$\boldsymbol{\lambda_{1,j} = \lambda_{1,0} + j \triangle \lambda \Rightarrow R_{1,j}, D_{1,j} \longrightarrow}$	$\boldsymbol{R_{2,j} = R_{budget} - R_{1,j} \Rightarrow \lambda_{2,j}, D_{2,j}}$	$\boldsymbol{D_{1,j} + D_{2,j}}$
...		
$\lambda_{1,N} = \lambda_{1,0} + N \triangle \lambda \Rightarrow R_{1,N}, D_{1,N} \longrightarrow$	$R_{2,N} = R_{budget} - R_{1,N} \Rightarrow \lambda_{2,N}, D_{2,N}$	$D_{1,N} + D_{2,N}$

Table 7.1: Progression of dependent optimization process

The typical stair-shaped dependency of the solution rate $R^*(\lambda)$ on the Lagrange multiplier λ (See Figure 7.2) suggests a more efficient way of performing the optimization. Instead of testing all values of λ, it would be enough to concentrate in the singular points, that is, only those values of λ for which more than one solution exists.

The process is to start with a λ_i. To find λ_{i+1} the value of λ is increased in $\delta\lambda$ so that $\lambda_{i+1} = \lambda_i + \delta\lambda$. If $R(\lambda_{i+1}) \neq R(\lambda_i)$ then, a new value of λ is computed as:

$$\lambda_j \leftarrow \frac{|D(\lambda_i) - D(\lambda_{i+1})|}{|R(\lambda_i) - R(\lambda_{i+1})|} \tag{7.8}$$

If $R(\lambda_j) < R(\lambda_i)$ then $\lambda_{i+1} \leftarrow \lambda_j$

If $R(\lambda_j) = R(\lambda_i)$ then $\lambda_i \leftarrow \lambda_j$

The process is iterated until $\lambda_i = \lambda_{i+1}$. This is the singular value nearer to λ_i. The next singular value is computed by iterating this simple process from the beginning.

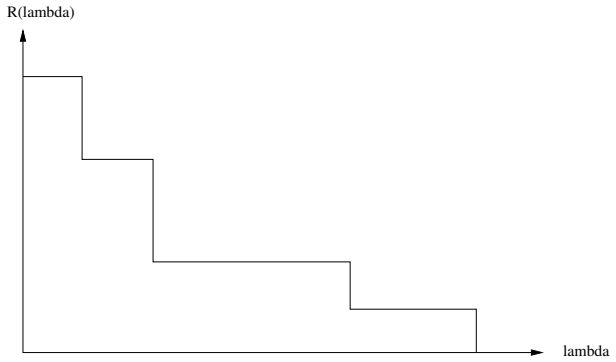

Figure 7.2: Typical dependency of the solution rate $R^*(\lambda)$ on the Lagrange multiplier λ

Chapter 8

Description of the extended coding system: Scalability

8.1 Introduction

The optimization algorithms presented in section 7.2 are applied to an existing segmentation based video coding system (known as SESAME) that has been extended to support content-based functionalities. This extended system will be referred to as XSESAME. In XSESAME, the base layer is intended to provide a basic representation of the current frame. The enhancement layers are used to improve the coding of selected regions (coding efficiency mode, non content-based) or objects (object functionalities mode, content-based). Full-frame scalability (i.e. non content-based) is viewed as a special case of content-based scalability with an object covering the complete frame. As it was stated in Section 7.2, the goal is the efficient representation, using a region-based paradigm, of a video sequence with several scalability layers and allowing content-based functionalities. We will restrict from now on to two scalability layers for simplicity. This does not restrict the generality of the approach, because all given procedures can be easily extended to support more scalability layers.

This Chapter is structured as follows: First, a global view of the structure of the encoder is given in Section 8.2. A detailed explanation of the various blocs in the scheme follows in Sections 8.3 (Projection block), 8.4 (construction of the base layer), 8.5 (construction of the enhancement layer), 8.9 (partition coding) and 8.10 (texture coding). Section 8.6 describes the two operating modes defined for the encoder: coding efficiency mode and object functionalities mode. Section 8.7 shows the implementation of the different scalability types (PSNR, spatial and temporal) currently supported by the coding system. Details on how the dependent and independent optimization strategies are implemented are given in Section 8.8. Section 8.11 shows an example of content-based functionalities (object tracking) integrated into the encoder.

8.2 Structure of the encoder

The structure of the XSESAME system is an extension of the structure of the basic system presented in Section 5. A block diagram that summarizes the various parts of the system is presented in Figure 8.1. This system is used to demonstrate the two different optimization algorithms: dependent and independent one.

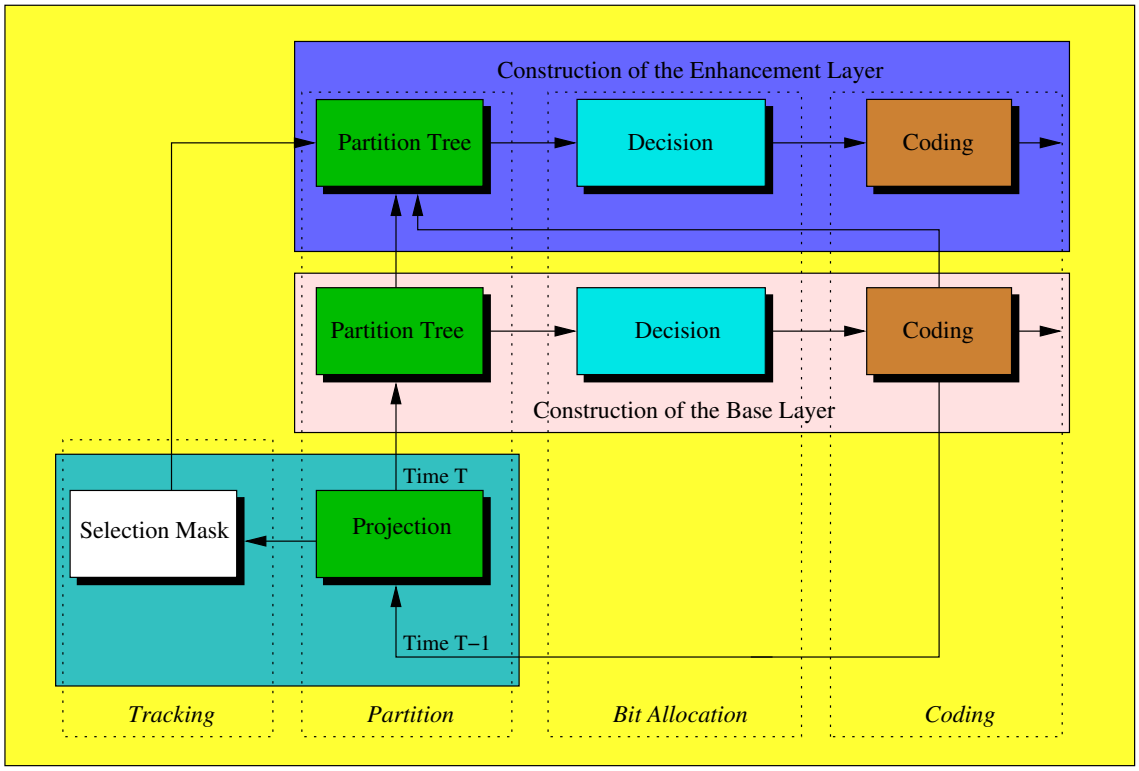

Figure 8.1: Scheme of the scalable encoder.

The extended coding algorithm can be defined by three main steps:

1. **Projection**. Partitioning the current frame into regions and definition of meaningful objects. These regions and objects are related to the ones in previously coded frames.

2. **Construction of the base layer**. In this step a basic representation of the whole image is generated.

3. **Construction of the enhancement layer(s)**. This is done by selecting the regions or objects in the scene that are to be coded in the enhancement layer(s).

There are two region selection modes: *Object functionalities* (supervised) and *Coding efficiency* (unsupervised), that are discussed in Section 8.6 and three scalable modes: *PSNR*, *spatial* and *temporal*, that are discussed in Section 8.7. All combinations of region selection and scalable modes are permitted.

In the following, the different steps are further developed in order to provide a complete description of the system.

8.3 Projection

The projection step requires a few changes with respect to the projection step in the basic coding system. In this step the partition of the previous coded frame is adapted to the data of the current frame. The result is a partition of the current frame that is not definitive but a starting point to construct the final optimal partition. The objective of the projection is to relate regions or objects present in the current frame to their references in the previous frame.

An improved tracking algorithm is used which allows a better automatic tracking of objects along the sequence. Spatial and motion information are used in both the marker selection and the partition creation [71]. More details of this process are given in Section 8.11.

Since the coding system must handle content-based functionalities, the contours of meaningful objects must be introduced in the Partition Tree. This way, the decision algorithm is able to select the appropriate regions that compose the object in the enhancement layer (as it will be shown in the sequel, the enhancement layer's Partition Tree is created as a subset of the base layer's Partition Tree).

The problem is that, if the base layer partition of the previous frame is directly projected, these contours may not appear in the projected partition. That is because the function of the base layer is only to give a basic representation of the image. Thus, no meaningful objects are necessarily defined in it and the contours of the objects may not be present in the previous frame encoded base layer (they are only ensured to appear in the enhancement layers).

For this reason, it is necessary to inject the contours of the object — as defined in the previous frame — into the coded base layer *prior* to the projection step (only at the encoder side). These contours will be present in the projected partition, and thus, in the middle and lower levels of the base layer Partition Tree. This way, the decision algorithm will be able to freely select regions to form the base layer (that may or may not reflect the object contours) and later, it will be ensured that the universe of regions has the appropriate elements so that the object can be represented at the enhancement layer.

However, the mere injection of the object contours are not sufficient because, if content-based functionalities are to be provided, label stability in the enhancement layer between successive frames of the sequence must be ensured. Label stability is achieved when the same

region receives the same label along the successive frames of the sequence. It is necessary for coding efficiency because, this way, the system will be able to use inter-frame texture coding at the enhancement layer (See Section 8.7). The injection of the contours of the partition does not allow region tracking in the enhancement layers because it does not solve the problem of region matching between the successive frames enhancement layers.

The adopted solution has been to construct two versions of the projected partition. The first one (called from now on *base projected partition* or simply *projected partition*) is created by projecting the previous coded base layer partition. The second one (called from now on *enhanced projected partition*) is created by projecting a union of the partitions of all the scalability layers. (See Figure 8.2)

The base projected partition will be used for the projected level of the Partition Tree and to construct the merging levels. The enhanced projected partition will be used for the construction of the re-segmented levels, and contains a) the contour of the objects, and b) information of the labels of the regions in the enhancement layers that will be useful for maintaining label coherence (See Figure 8.3 in Section 8.4). The two versions of the projected partitions must be constructed in a way that ensures that the common contours are identical. This could be done, for example, by using a look-up table (LUT) that relates the labels in the previous frame decoded base layer partition (rec_L0 in the figure) and the labels in the union of the scalability layers partition (rec_L0L1 in the figure).

However, the projection algorithm can, in some situations, introduce new regions. These new regions would not be listed in the LUT and therefore undefined regions in the base projected partition may appear. To solve this situation, both the rec_L0 and rec_L0L1 are projected, and the LUT is constructed using these projections. As the contours of these two partitions do not necessarily match, the construction of the LUT is done by relating the regions with better matching. Once the LUT is constructed, it can be used together with the enhanced projected partition to rebuild the base projected partition.

Note that the fact that these contours appear in the base layer Partition Tree does not mean that they will appear in the base layer final partition. The optimization algorithm can choose to select regions from the above levels of the PT that do not contain these contours.

The imposition of regions from the enhancement layers into the projected partition will result in new regions that have no reference in the previous base layer coded partition. As a result, the contours of these regions are to be coded always in intra-layer mode (if they are selected to appear in the final partition). The texture of these new regions can be coded in intra- or inter-frame mode. If content-based functionalities are to be considered, it is necessary to decide whether or not the new regions belong to the selected object (See Section 8.11).

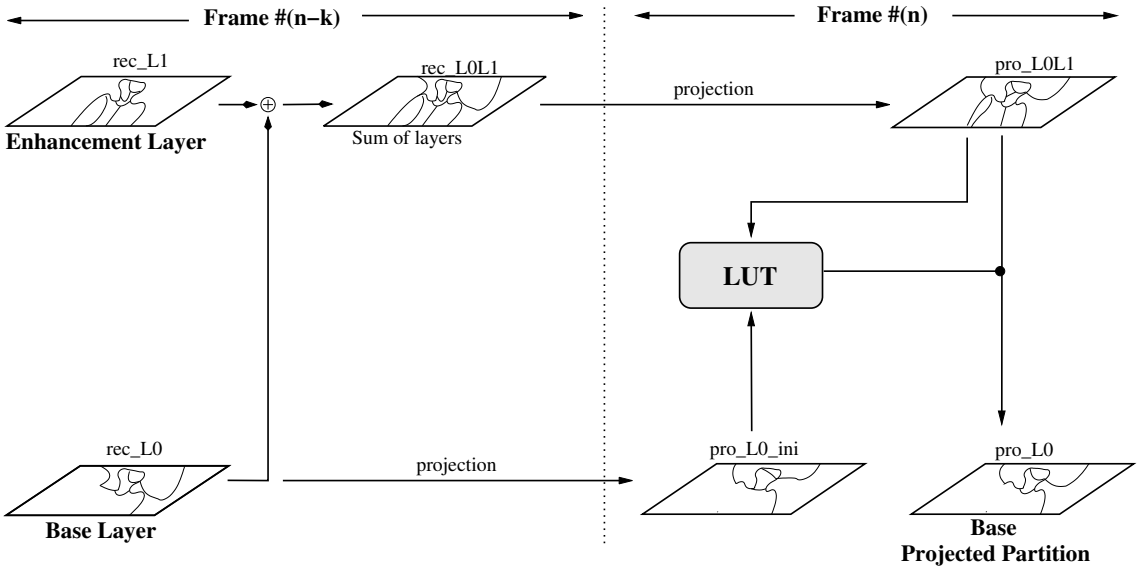

Figure 8.2: Creation of the two projected partitions. In the figure, the suffixes L0 and L1 stand for base and enhancement layers, respectively. Thus rec_L0 is the base layer decoded partition, while pro_L0L1 is the projection of the sum of the base and enhancement layer partitions (enhanced projected partition). pro_L0 is constructed from pro_L0L1 and a LUT.

8.3.1 Definition and handling of Video Objects

In the previous discussion, it is assumed that the VOs are available at the encoder side. Let us see how VOs are handled in the extended coding system. This handling comprises initial definition, representation and updating.

The definition of VOs can be done in two manners: One possibility is to take the problem of defining the VOs out of the design of the coding system. In this case, the shape of each VO should be previously computed by means of an external method, and provided to the encoder by means of a binary mask for each frame. This mask will be used in the projection step to identify the regions belonging to the object. There is no updating step as the mask is externally provided.

The second possibility is to select the object interactively in the first frame of the sequence and then automatically construct the selection mask for the successive frames by tracking the selected object along the sequence. The updating of the mask from frame to frame is performed by the encoder. This tracking capability will be further discussed in Section 8.11.

8.4 Construction of the base layer

The construction of the base layer requires modifications with respect to the process used in the coding of the full frame in the non scalable case. From the projected partition, a set of hierarchical partitions (*Partition Tree*) is constructed. In this case, two versions of the projected partition are used (See Figure 8.3) in order to introduce the object in the Partition Tree and to maintain label coherency in the enhancement layer.

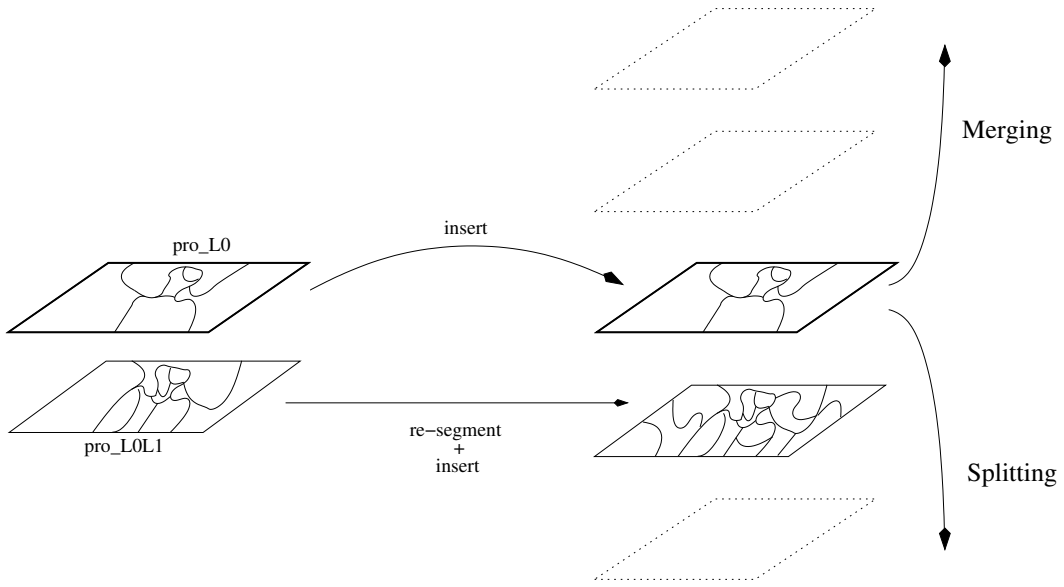

Figure 8.3: Construction of the Partition Tree from two versions of the projected partition. pro_L0 is the base projected partition. It is used as a projection level and to build the merging levels. pro_L0L1 is the enhanced projected partition. It is not introduced directly in the Partition Tree, but used to build the re-segmented levels.

The base projected partition is injected without modification into the Partition Tree. It is used to create the merging levels, by using the same motion homogeneity approach as in the non-scalable case (see Section 6.1). That is, first motion is estimated for each region at a given level and then, a merging cost is computed for every couple of neighboring regions. The couples of neighboring regions with the smallest merging costs are merged until the required number of regions for the next level are obtained. This procedure is iterated for all the merging levels.

To introduce the information (contours and labels of the regions) conveyed by the enhancement layer partition in the Partition Tree, the re-segmented levels are constructed using the enhanced projected partition instead of the base projected partition. This partition contains the contours of both base and enhancement layers as it is constructed by merging previous frame base and enhancement layer final partitions. While the enhanced projected partition

could be injected directly into the Partition Tree, a different approach has been used. The enhanced projected partition is re-segmented and the resulting partition is the one that is injected into the Partition Tree. The reason for this is that this way the results are more coherent with the re-segmentation process used in the splitting levels.

The re-segmentation is carried out using a size criterion for the first levels and a contrast criterion for the last level. At each level, there is a filtering process with an intensity that decreases progressively at the successive partition levels and that determines the number of regions required at each given level. Experimental tests show that the enhanced projected partition is usually under-segmented when compared with the first re-segmentation level if this were constructed using the usual criterion (size or contrast). In this case, by re-segmenting the enhanced projected partition, the number of regions in the first re-segmented level will be similar to the one that will result if were constructed from the base projected partition.

At the decoder side, the projected partition can be constructed by the base layer decoder. It will be used there to label the base layer decoded partition. The second version of the projected partition can only be constructed by the enhancement layer decoder, and it is used to label the enhancement layer decoded partition. Details on how the enhancement layer regions are labeled are given in Section 8.11.

Each region in every level of the Partition Tree is encoded by means of all the available coding techniques/quality levels, and the resulting data is stored in a hierarchical structure called Decision Tree. The data stored in the *Decision Tree* is used by the R-D optimization algorithm stated in Section 7.2 to make a decision that leads to an optimal representation of the image for a given bit budget. That is, a decision on which regions from the different levels will form the final partition, and a decision on the optimal texture coding technique and quantizer choice for each region.

Note that the Partition Tree constructed in this step contains the contour of the object(s) to be coded in the enhancement layer. That is because in the projection step, these contours have been introduced in the projected partition, which is the initial point from which the Partition Tree is created. However, in the construction of the base layer no restriction is imposed to preserve the object contours.

It is important to remember that, as the Partition Tree is derived from the projected partition, the temporal coherence between the partitions of successive frames is preserved, and thus, the system can address content-based functionalities.

8.4.1 Balancing the Partition Tree

To ensure that the resolution of the Partition Tree is well balanced, that is, with the different partition levels evenly distributed from fine to coarse, a simple modification in the Partition Tree construction step is used. It helps in the cases where the projected partition has too

many or too few regions. The new approach consists on altering the symmetry of the Partition Tree depending on the number of regions on the projected partition (See Figure 8.4).

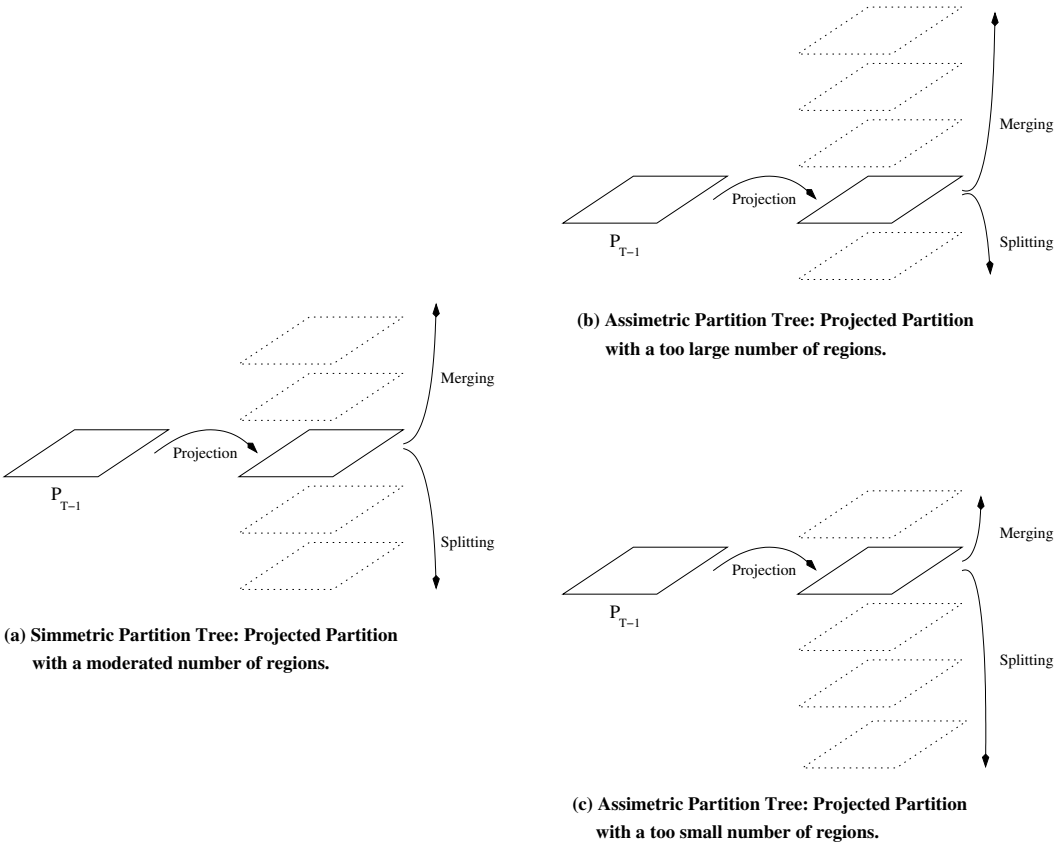

Figure 8.4: Adjusting the Partition Tree structure depending on the number of regions in the projected partition.

If the projected partition is over-segmented, with a large number of regions, more merging levels are used at the expense of re-segmentation levels. If the projected partition is too coarse, with a small number of regions, more re-segmentation levels are used at the expense of merging levels. A fixed mapping, based only on the number of regions of the projected partition, is used to derive the number of merging/splitting levels from the number of regions in the projection level. If num_regs_pro is the number of regions in the projected partition, then:

$$1 \leq num\_regs\_pro \leq 3 \longrightarrow 0\ merging,\ 1\ projected,\ 4\ splitting\ levels$$
$$3 < num\_regs\_pro \leq 15 \longrightarrow 1\ merging,\ 1\ projected,\ 3\ splitting\ levels$$
$$15 < num\_regs\_pro \leq 40 \longrightarrow 2\ merging,\ 1\ projected,\ 2\ splitting\ levels$$
$$40 < num\_regs\_pro \longrightarrow 3\ merging,\ 1\ projected,\ 0\ splitting\ levels$$

The goal is to stabilize the total number of regions in the Partition Tree, so that the decision algorithm has a broad range of options to select when constructing the final partition. This is specially necessary because the injection of the contours of meaningful objects in the projection step can increase the number of regions at the projection level.

8.5 Construction of the enhancement layer

For the enhancement layer, the Partition Tree is not derived from the projected partition, but it is constructed as a subset of the base layer Partition Tree (See Figure 8.5). Let us give the justification for this mode of operation: As it has been seen in Chapter 5, the Partition Tree is a set of partitions that represents the image with various levels of resolution. When the decision algorithm decides the final base layer partition, regions from different levels are selected. The purpose of the enhancement layer is to enhance the coding of selected regions or objects. Due to the segmentation-based nature of the algorithm, this can be done in two ways: improving the coding of the texture or refining the partition. The later means to select regions from lower levels of the Partition Tree to form the final enhancement layer partition. This can be achieved by constructing the new Partition Tree taking only the regions that lie under the ones selected to form the base layer Partition Tree.

This can easily be done using the Decision Tree because this structure represents in a compact way the hierarchical structure of the Partition Tree. After the selection of the optimal partition for the base layer, there are regions in the Partition Tree that represent the image with a finer level of detail than the one used in the base layer. These regions can be used to create the enhancement layer Partition Tree. The process is to prune the base layer Decision Tree at the level of the regions belonging to the base layer final partition, and create a new Decision Tree with the pruned branches. From the new Decision Tree, the corresponding Partition Tree can be build easily.

No re-segmentation step is used in this process since the unused branches are taken. If the decision algorithm has selected a region in the lowest level of the tree for the base layer, this region is used in all levels of the enhancement layer Decision Tree. Therefore, such a region can not be further re-segmented (although its coding quality can be improved) in the enhancement levels. Nodes with only one branch in Figure 8.5 represent this situation, for example node 2. In this case, the 'child' region has the same shape as the 'father' region.

It exists a relation between the target quality for a give layer and the 'deepness' (the Partition Tree level) of the regions forming its final partition. If the tree is well balanced, the regions forming the base layer will be usually taken from the coarse and middle levels of the Partition Tree, thus allowing the refinement of the regions in the enhancement layer by selecting regions from the finer levels.

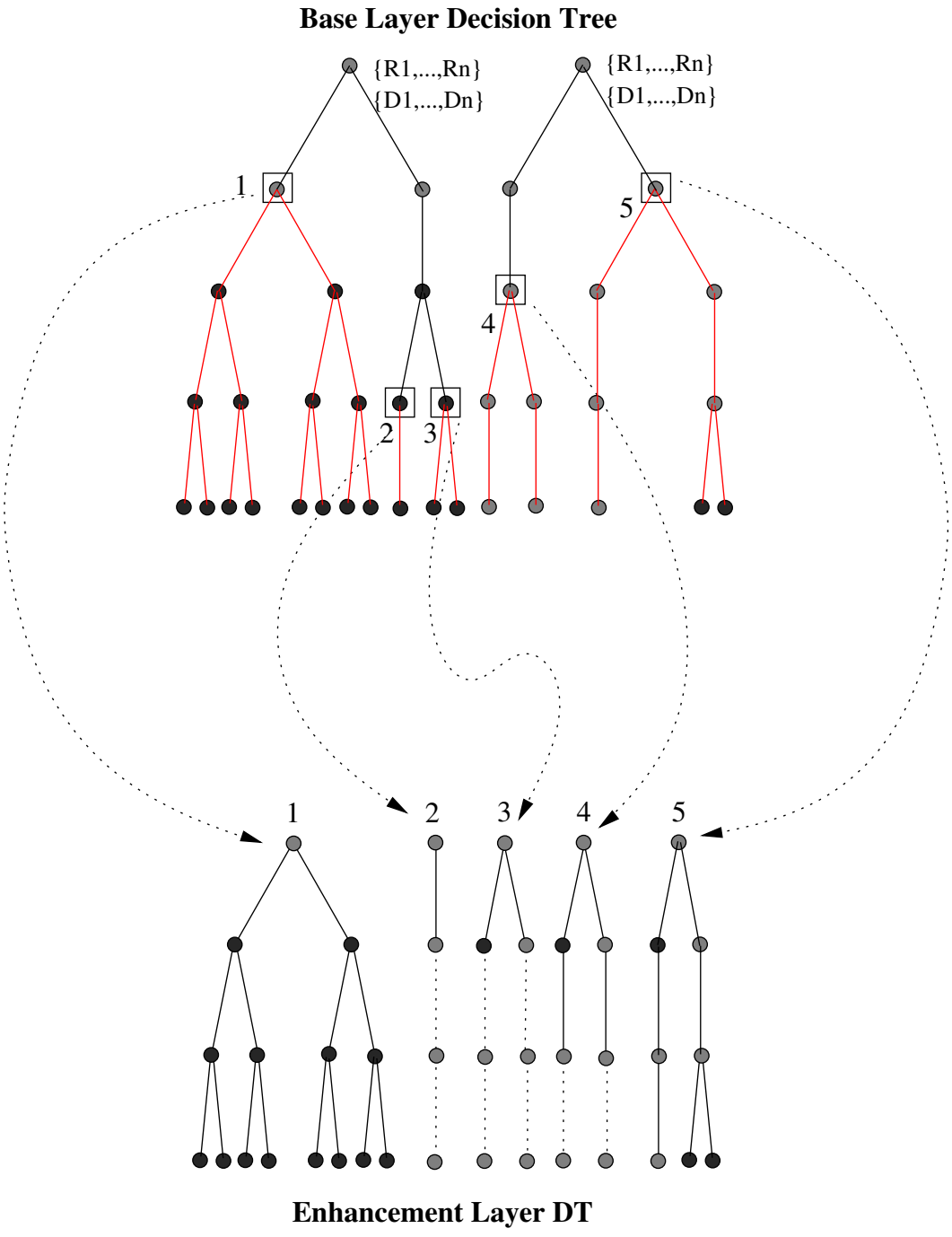

Figure 8.5: Construction of the enhancement layer Decision Tree by pruning the base layer Decision Tree. The square boxes mark the regions forming the base layer partition.

Note that only the structure of the base layer Decision Tree (the relationship between 'father' and 'children' regions) is used. The rate-distortion data stored in the base layer Decision Tree can not be used and must be re-computed for each region to adapt to the new data being coded. That is, all the nodes of the Decision Tree must be re-populated with the R-D data corresponding to: a) the coding of the error between the original image and the decoded base layer (in PSNR and spatial scalability); b) the coding of the error between the original image and the compensation of the enhancement layer of the previous coded frame.

Then, the R-D optimization algorithm (see Section 7.2) is applied to find the optimal representation for this layer. For each type of scalability, specific adaptations in the structure of the encoder are necessary (See Section 8.7).

In all types of scalability, it is possible to define Video Objects in a supervised way by giving to a region or set of regions a semantic meaning. This allows to select the objects that are to be placed on the enhancement layers (*object scalability*). This way, an object can be defined in the first frame and tracked across the video sequence in the enhancement layer, ensuring that all the available bit budget is spent only on the refinement of this object.

8.6 Region selection modes

The coding system presented in this thesis has two different modes of operation. One aimed at providing content-based functionalities at the enhancement layer and the other aimed at coding efficiency, without content-based functionalities. The process used to build the enhancement layer final partition is different for each mode. The construction of the enhancement layer final partition consists on selecting the appropriate regions from the Partition Tree. The region selection criterion can be chosen by the decision algorithm, without external interferences, in order to obtain the best results in a R-D sense, or alternatively, can be externally influenced in order to prioritize a set of regions with a semantic meaning. In this section these two approaches are presented.

8.6.1 Coding efficiency mode: Unsupervised

The objective is to enhance the coding already done in the base layer by distributing the bit budget among chosen regions of the new enhancement layer Partition Tree. In this case, it is the optimization algorithm itself who decides which regions are to be sent in a given layer on a rate-distortion sense basis, without user interaction of any kind.

The process is the following: first, a new Partition Tree and Decision Tree are constructed for the enhancement layer. These trees are subsets of the base layer Partition and Decision trees. As it is shown in Figure 8.6, the decision algorithm selects regions from the base layer Partition Tree (gray regions in the Partition Tree and circled nodes in the Decision Tree).

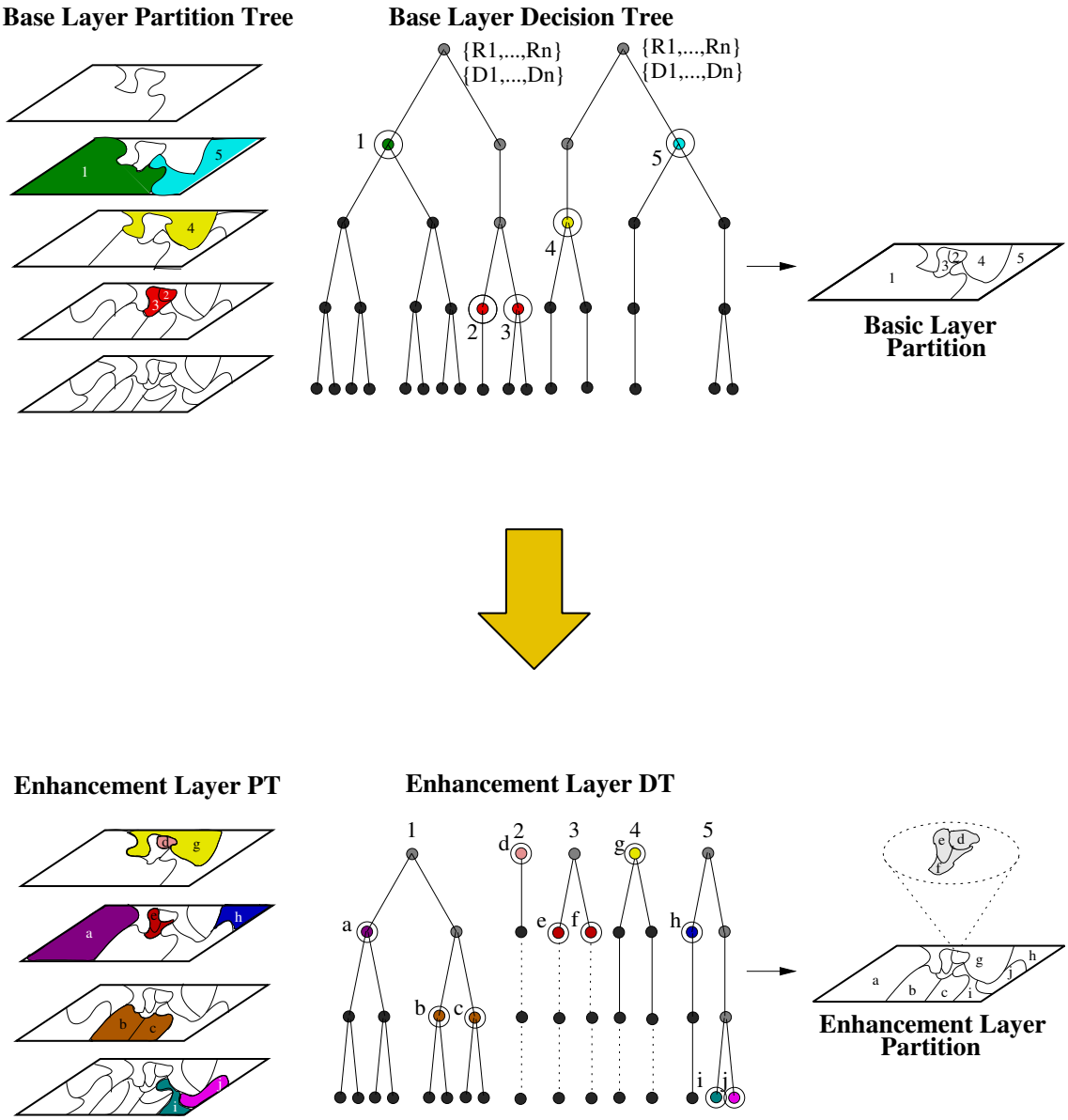

Figure 8.6: Coding efficiency mode: Example with two layers. (The usual representation of "base layer at the bottom/enhancement layer at the top of the figure" is reversed here to better show the process).

The regions selected (labeled with numbers 1-5 in the figure) form the final partition for the base layer. Once the base layer is encoded, a new tree is created with the branches of the base layer Decision Tree, pruned at the level corresponding to the selected regions (See also Figure 8.5). All the nodes of the Decision Tree must be re-populated with the R-D data, as was explained in Section 8.5. Note that the construction of the enhancement layer is done only by splitting regions present in the base layer, no merging is allowed.

The new Partition Tree represents the whole frame using only the finer parts of the old Partition Tree. The decision algorithm uses this new tree to select the set of regions (labeled with letters a-j) that will form the enhancement layer partition.

We have to point out that, in this mode, regions may not have any semantic meaning, because the criterion that is used for its definition is coding efficiency.

8.6.2 Object functionalities mode: Supervised

In this mode, the key point is the possibility to provide external control or interactivity over the objects that are to be placed on the enhancement layers. Let's recall that the notion of object differs from the notion of region. Regions are the basic high-level entities defined by a segmentation step according to a homogeneity criterion and without any semantic meaning. Objects are entities with a semantic meaning, defined by an analysis process. In this work, an object can have an internal structure and be composed of several regions. This internal structure results from the segmentation-based nature of the coding algorithm.

The Partition Tree is a structure composed of regions organized in a hierarchical way. To define semantic objects, a selection mask is used. This mask marks which regions form the given object. Masks can be externally provided for each frame or, given a selection mask for the first frame of the sequence, the regions under the mask can be tracked along the sequence by means of an appropriate algorithm [71], so that the masks in the following frames are automatically generated. It is this capability of marking and tracking specific objects which gives the algorithm its strength to handle content-based functionalities.

The process to create the enhancement layer trees differs from the same process in the coding efficiency mode, since here it is restricted to select regions belonging entirely to the object. This is, the region selection algorithm is restricted to the branches that originate in regions covered by the selection mask (See Figure 8.7). This way, the resulting Partition Tree represents only a portion of the image, that is, the area corresponding to the selected object.

Note that the injection of the regions of the enhancement layer in the construction of the Partition Tree (See Section 8.4), using the enhanced projected partition, guarantees that a set of regions whose external contours match those of the object will exist in the Partition Tree.

In the example in Figure 8.7, the final partition for the base layer is composed of five

regions. Among these regions, two do not overlap the selection mask (regions 4 and 5), two are inside (regions 2, and 3), and one is only partly covered by the mask (region 1). In the creation of the enhancement layer Decision Tree, the branches representing regions outside the mask are discarded (4, 5), the branches corresponding to regions completely under the selection mask (2,3) are taken directly as in the coding efficiency mode. In the case of branches representing regions covered in part by the selection mask, a search is done in the lower levels of the tree until a region is found that is completely under the selection mask (1') (Such regions will be always present in the Partition Tree because contours of the selection mask have been injected in the Projection step). The corresponding sub-branch is selected to form part of the new Decision Tree.

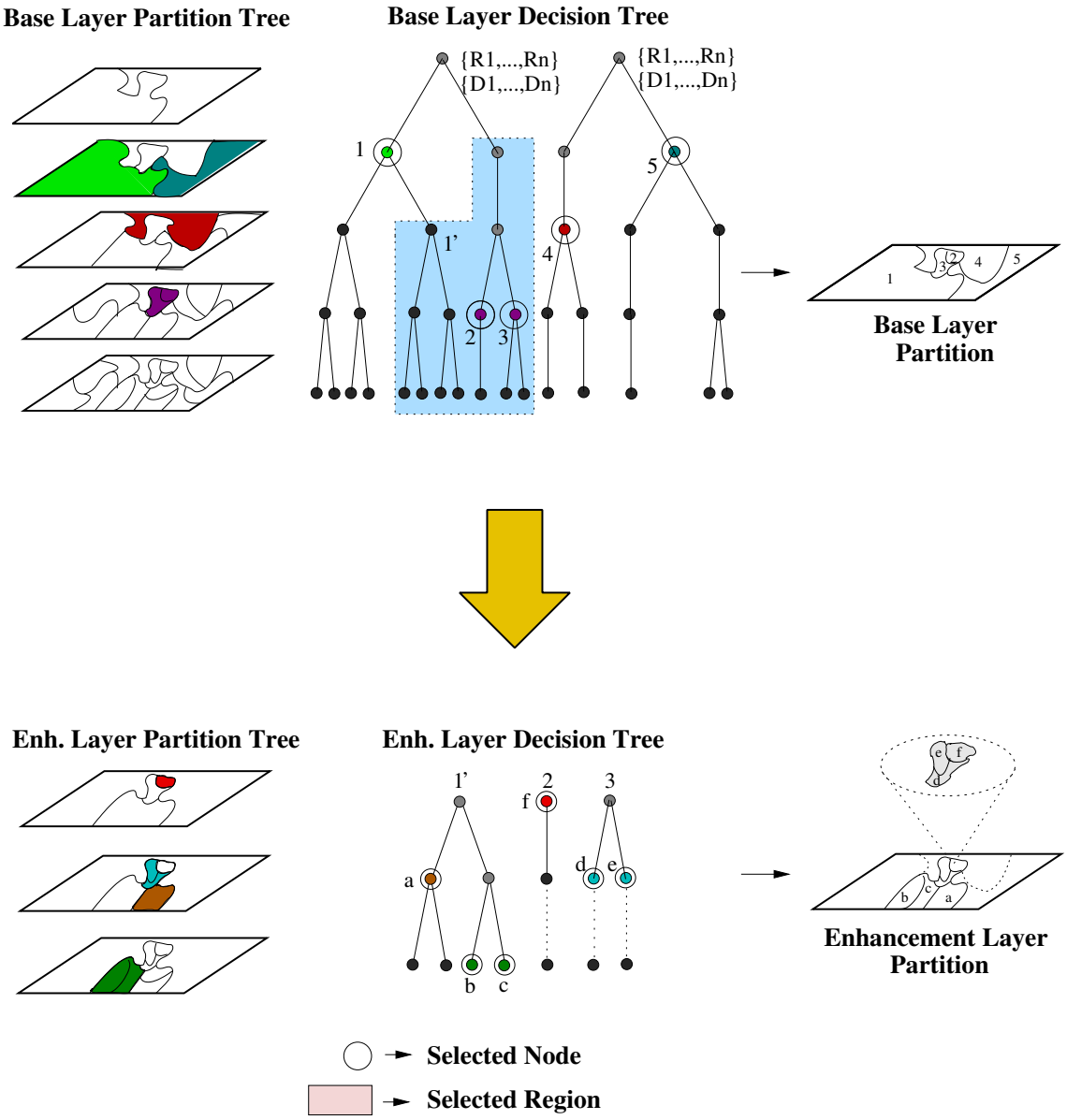

Figure 8.7: Object functionalities mode: Example with two layers. (The usual representation of "base layer at the bottom/enhancement layer at the top of the figure" is reversed here to better show the process).

8.7 Scalable modes

In this section, the algorithmic structure and the implementation details for the different scalable modes are presented.

8.7.1 PSNR scalability

In this mode, the enhancement layer improves the coding of the current frame by refining the texture of the base layer.

In PSNR scalability, a new Partition Tree is constructed taking as basis the final base layer coded partition. This tree is formed by the regions or objects already present in the base layer Partition Tree that have not been used in the base layer. This way, the additional layers can enhance the coding of the current frame by introducing new regions, or by refining the texture of regions already present in the base layer.

For the texture coding, the algorithm uses a pyramidal approach [102, 11] combined with motion compensated coding of the previous frame (See Figure 8.8).

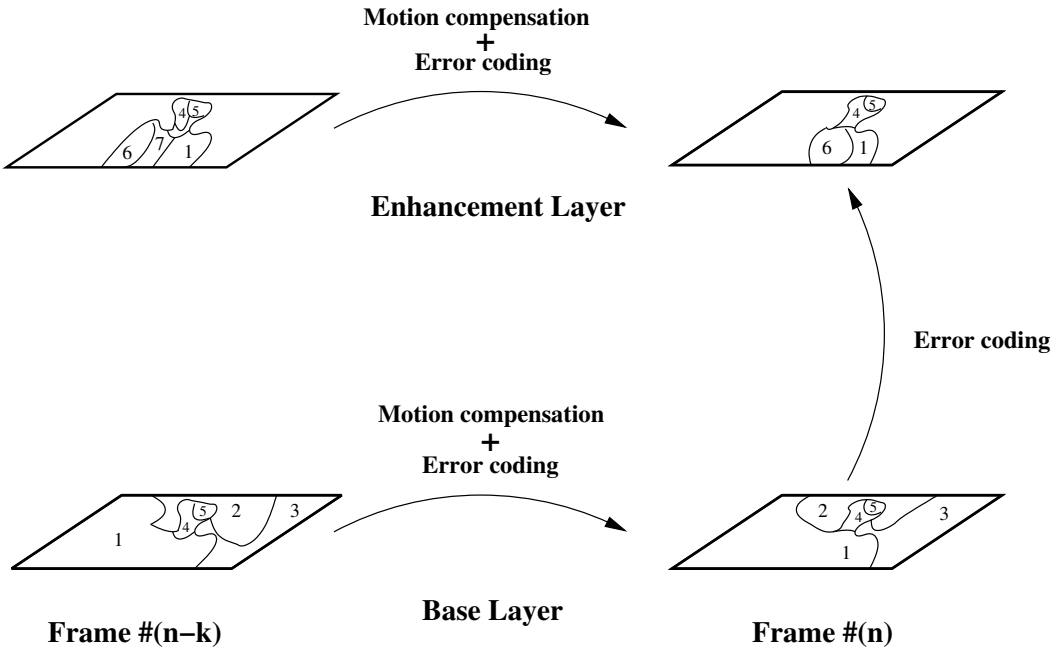

Figure 8.8: Texture compensation modes. Texture and contours in the enhancement layer can be predicted from the previous frame enhancement layer or from the current frame base layer. In the figure, this is done for object scalability (See Section 8.6.2)

Refinement of the texture of the regions is done by coding two different types of information:

- The residue between the original image and the decoded base layer.
- The prediction error between the original image and the motion compensated previous frame enhancement layer.

Then, both techniques are put into competition and, for each region, the best coding method is selected by using an appropriate optimization algorithm (See Section 7.2). This dual approach allows the efficient representation of different types of regions. The motion compensation technique will normally be selected in the case of regions already present in the last coded frame that can be efficiently predicted. The pyramidal approach is usually selected for new regions that may appear in the scene and for regions that can not be correctly motion compensated.

In a segmentation-based coding system, new regions can be introduced in the enhancement layer in order to improve the coding efficiency. These new regions are constructed by splitting the ones already present in the base layer, and should provide better homogeneity, leading to an improved coding efficiency. In Object functionalities mode, the enhancement layer introduces and refines the Video Objects. In this case, the segmentation process used in the construction of the hierarchy must be constrained by the shape of the Video Objects.

Note that the use of a prediction of the enhancement layer for texture coding introduces new temporal dependencies in the process. As it was stated in Section 7.2, in this work inter-frame dependencies are ignored in the optimization process, even in the dependent optimization mode.

> **Implementation notes**: The process of correctly labeling regions in the enhancement layer is important because the coding efficiency can be improved by maintaining the temporal coherency between the labels of the regions in the enhancement layer. This way, the texture motion compensation plus error coding process shown in Figure 8.8 can be properly implemented.
>
> In the enhancement layer, regions originate from those in the base layer in a 'father/children' relationship. Every region in the enhancement layer has a unique reference in the base layer, and several regions can have the same reference. In intra-frame mode, among all the children of a same father, the largest one keeps the father label, and the others are labeled with increasing values, starting at the highest value of labels in the base layer. In inter-frame mode, the projection of the union of all scalability layers is used as a reference to label the regions of the scalability layers, in the same way the base projection is used to label the base layer.

8.7.2 Spatial scalability

In Spatial scalability (see Figure 8.9) the scalability layers are coded at different resolution levels. In the base layer, a sub-sampled version of the original frame is coded. The coding residue is obtained from the original image at full resolution and from an up-sampled version of the reconstructed base layer.

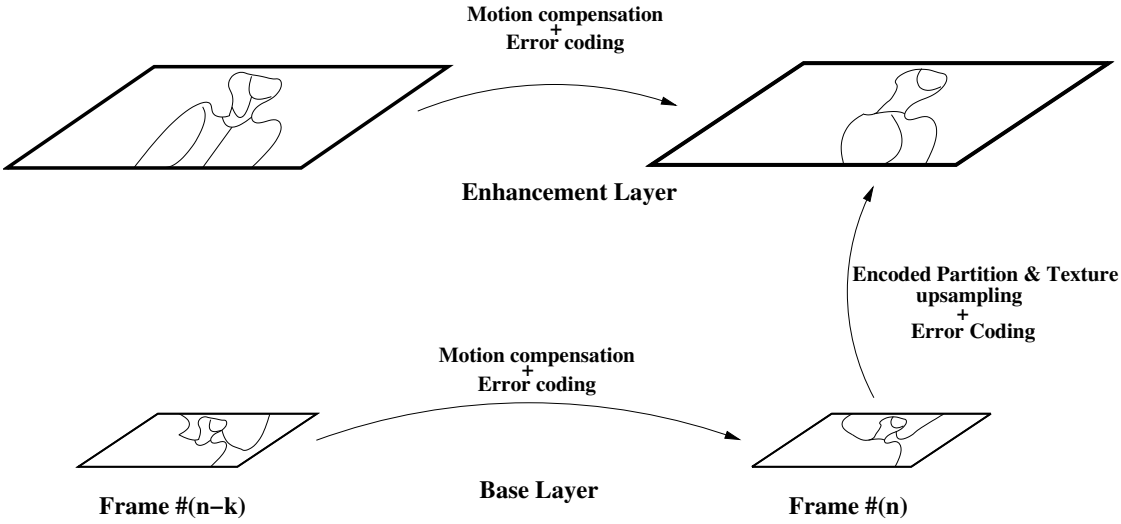

Figure 8.9: Spatial scalability. Texture and contours in the enhancement layer can be predicted from the previous frame enhancement layer or from an up-sampled version of the current frame base layer.

In spatial scalability, the construction of the new Partition Tree is done as in PSNR scalability. The difference is that here the enhancement layer represents the image with increased resolution. For this reason, the partitions that form the new Partition Tree must be properly scaled. That is easily performed because in our implementation, the Partition Tree is represented by the finest partition and a look-up table that defines the regions that are merged in the successive levels. In this case, to define the new Partition Tree at the new resolution it is enough to up-sample the finest partition.

The double projection approach described in Section 8.3 is performed as follows: as now the base and enhancement layer partitions have different resolution, the previous frame base layer partition (rec_L0 in Figure 8.10) is up-sampled and merged with the enhancement layer previous layer partition. The result (rec_L0L1 in the figure) is projected into the current frame at full resolution (pro_L0L1). At the same time, rec_L0 is also projected, and the result (pro_L0_ini) is up-sampled and compared with pro_L0L1 in order to build the LUT. The final base projected partition is created by applying the information in the LUT to a down-sampled

version of the enhanced projected partition.

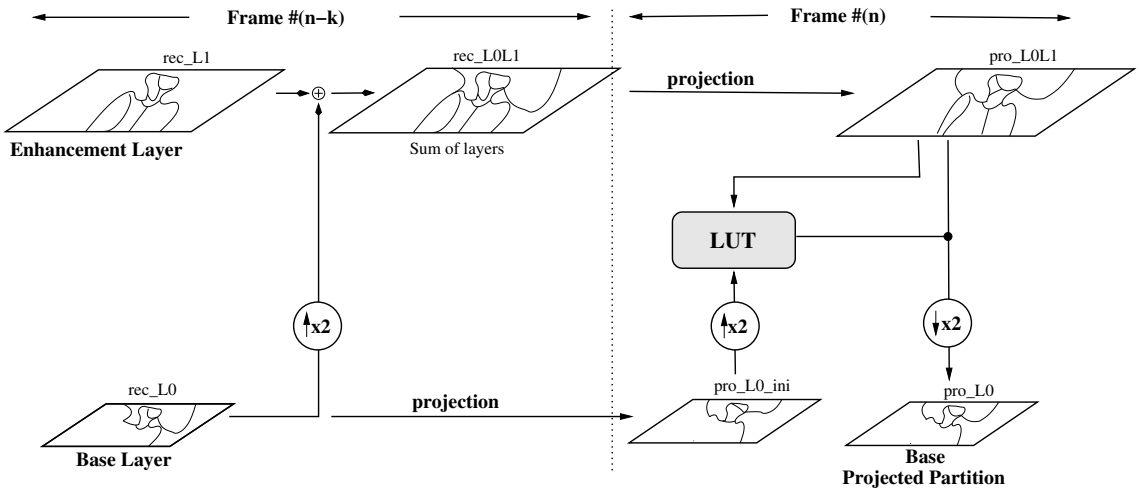

Figure 8.10: Creation of the two projected partitions. In the figure, the suffixes L0 and L1 stand for base and enhancement layers, respectively. Thus rec_L0 is the base layer decoded partition, while pro_L0L1 is the projection of the sum of the base and enhancement layer partitions (enhanced projected partition). pro_L0 is constructed from pro_L0L1 and a LUT.

Implementation notes: The process to up-sample both the finest partition of the Partition Tree and the final base layer partition needs some further considerations. If the contour-coding technique used in the base layer is lossy, the shape of the regions in the finest level of the Partition Tree (not contour-coded) and in the base layer decoded partition may be slightly different. The differences can lead to errors in the labeling of the enhancement layer partition. To avoid that, the following process is used to ensure coherency between up-sampled partitions:

- Up-sample (x2) the base layer decoded partition using nearest-neighbor interpolation (4 connectivity).
- Up-sample (x2) the finest level of the Partition Tree. Bilinear interpolation (4 connectivity) is used in the interior of the regions. Border pixels are not given a label, and are left as uncertainty regions.
- A 2D Watershed algorithm, constrained by the up-sampled base layer decoded partition, is used to give a label to uncertainty pixels.

Another important point to take into account is that the partition down-sampling process used to create the base layer projected partition could alter the topology of the partitions in the general case. This is the case, for instance,

when a region is shaped as chunks connected by thin lines. An image containing such regions cannot be down-sampled directly because if thin connections were eliminated completely by the down-sampling process, partition connectivity would be lost. To avoid this problem, the original partition will be modified by using a filtering step (see Figure 8.11), consisting on removing small regions and thin elongations in the partition, is performed during the projection step. Re-labeling is used to deal with regions that are broken into non-connected regions.

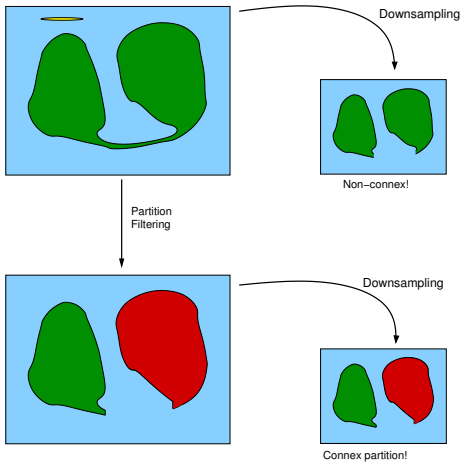

Figure 8.11: Partition filtering.

As the filtering can replace the topology of the partition, it is used only during the partition definition step. For instance, it is applied in the projection step as well as in the construction of the re-segmented levels during the definition of the Partition Tree.

8.7.3 Temporal scalability

In temporal scalability (See Figure 8.13), the same process used to construct the base layer is also applied to construct the enhancement layer: That is, projection of the coded base layer partition, construction of the new Partition Tree and finally decision and coding of the texture and the partition. In this case, the new Partition Tree is not a subset of the previous one as it is in the PSNR and spatial scalability cases. The reason is that, in this case, the base and enhancement scalability layers represent two different frames of the sequence. The regions and objects can modify its shape and position from frame to frame (See Figure 8.12). Thus, they can not be represented by the same set of hierarchical partitions.

In object functionalities mode all the available bit-budget is used to code the selected

object. The background is therefore not coded and is taken without modification from the base layer. If the selected object has changed its shape or position in the enhancement layer (frame #(n+k)) with respect to the base layer (frame #(n)), an uncovered background zone will appear (see Figure 8.12). This zone cannot be filled with data from the base layer. In this case, a hole will be left on the background.

This is not a problem in the coding efficiency mode of temporal scalability because in this case the whole frame is coded in both the base and enhancement layers. Of course, in PSNR and spatial scalability this problem does no appear because in these cases, the base and the enhancement layers represent the same frame and thus, the shape and position of the selected object in both layers coincide.

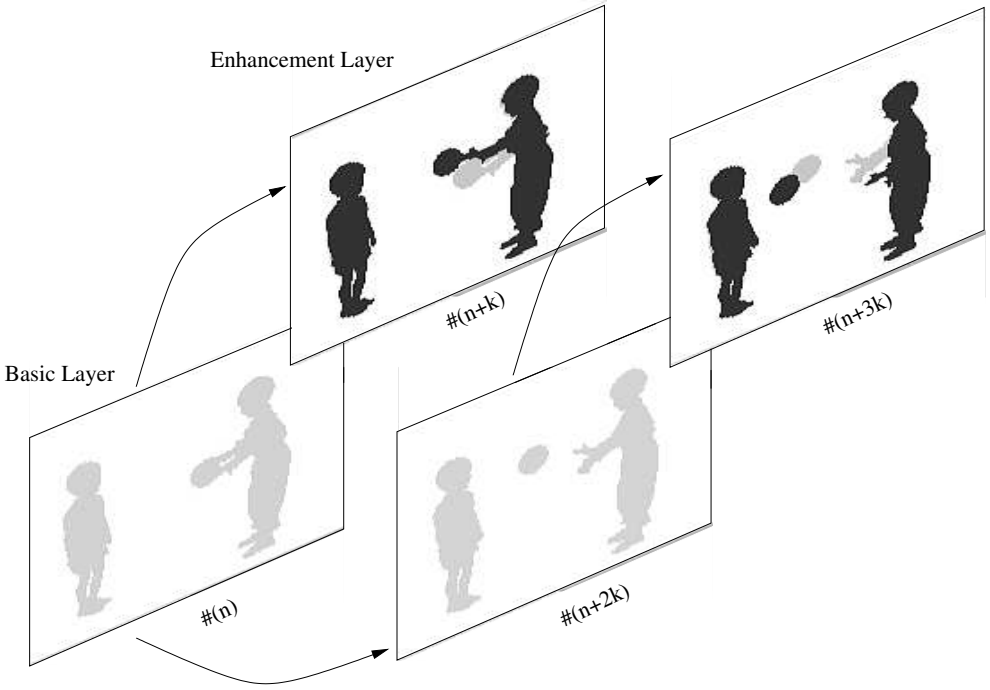

Figure 8.12: Uncovered background in Temporal Scalability: The gray zones in the enhancement layer mark the uncovered areas that must be filled.

Uncovered background can be dealt by detecting the uncovered background zones, which are then filled by motion compensating the texture of the neighboring regions. This way, only a few motion vectors are used to code these non selected areas and almost all the available bit budget can be spent on the coding of the object itself (See a more detailed explanation in *Implementation notes*, later in this Section).

Figure 8.13 shows the links and dependencies between the different frames at different scalability layers. In this scheme, the reference is always the most recent frame belonging to the same or lower scalability layer level.

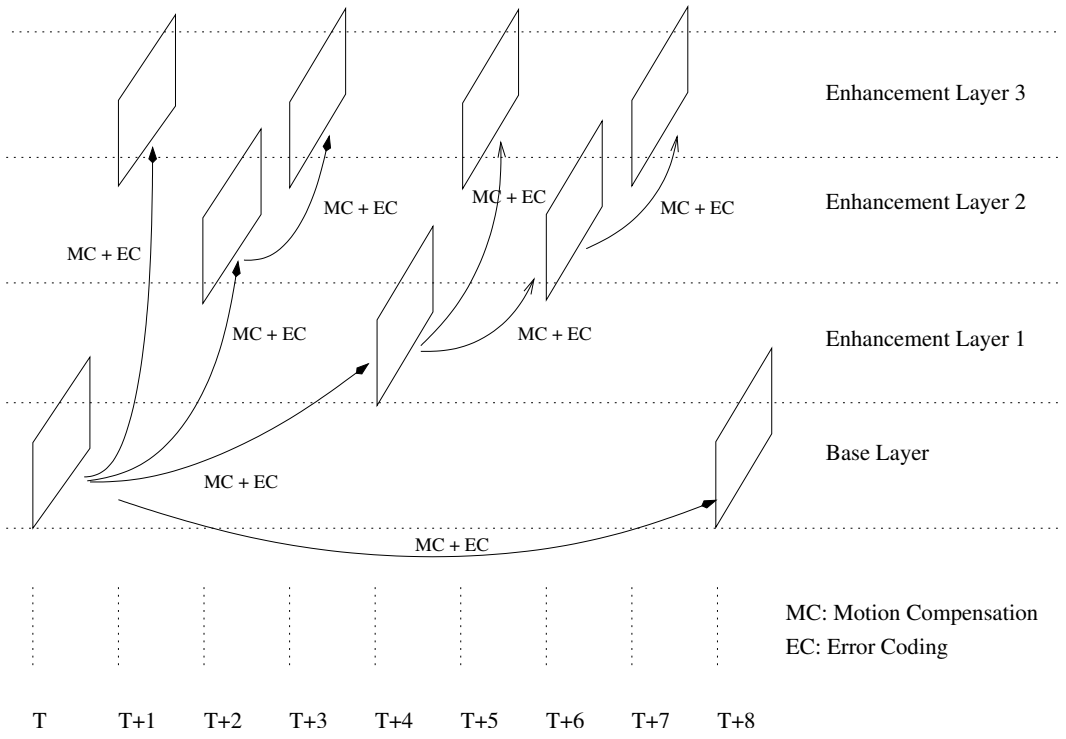

Figure 8.13: Temporal scalability layer dependencies. Example with four scalability layers. The arrows represent the references for texture/contour compensation.

Implementation notes: The process to deal with uncovered background zones has been implemented as follows: Let's be *mask* and *prev_mask* the selection masks marking the object of interest in the current and previous frames respectively. Lets *ori_old* and *rec_old* be the previous frame original and reconstructed images, and *uncov_mask* be a binary mask representing the uncovered background region. Then:

The uncovered regions are detected by comparing *mask* and *prev_mask*. The uncovered background is formed by pixels belonging to *prev_mask* that do not belong to *mask*.

Area filtering is used to eliminate small regions in the *uncov_mask*. Then, motion estimation is performed on the uncovered zones, so that a motion vector is computed that allows compensation of the texture in this zone from the data in the reconstructed previous frame base layer.

Pixels belonging to the filtered small regions are interpolated from neighboring pixel values.