Part II

Optimal strategies for coding efficiency in a segmentation-based coding system

Chapter 5

Basic optimal coding in a segmentation-based coding system

5.1 Introduction

In Part II, the general problem of finding the best representation for an image sequence is studied. The proposed solutions have been developed in the framework of segmentation-based coding systems. The focus is set on coding efficiency. The algorithms and methods developed in this work have been applied to a specific segmentation-based coding system, the so called SESAME [117, 16].

In Section 2.3 it was pointed out that coding systems decompose the input information into signal units that are coded separately. These signal units can be mutually dependent or independent. For instance, the rectangular blocs of a block based still image coder can be considered independent, in the sense that the way in which a particular unit is coded (i.e. the coding technique or quantizer choice) does not affect any other units. On the contrary, in video coding systems using motion compensation, the result of the coding of a particular signal unit (frame, region or block) depends on the choice of parameters that has been made in the signal unit used as a past reference for compensation.

To ensure optimality, the operational Rate-Distortion approach described in Section 2.3 has been used. This means that the search for optimality is restricted to a specific system, instead of the general search for optimality that is done when using the analytical or model based approach. While this choice is made for practical reasons, this implies that in the search for optimality the selected framework has a great impact on the performance of the coding system. This is, a system can be optimal on an operational Rate-Distortion sense and perform worse than a non-optimal system that uses a different framework [91].

A general solution to the problem of finding an optimal representation for an image sequence should take into account interdependencies where they exist. However, this is not

feasible in most cases because of the extreme complexity of this kind of approaches. Additionally, the delay introduced by this approach would not be acceptable in real time/interactive applications. In practice, such dependencies are ignored and the problem is tackled as a case of independent optimization.

In this work, both possibilities have been analyzed. In this chapter, a solution for the assumption of independent coding will be proposed. Then, Chapter 7 will tackle the problem of how dependent optimization can be handled in a practical way. A comparison between the two approaches will be given, in order to show how the increased complexity of the dependent model trades off for better performance.

Part II is organized as follows: After this introduction, in Section 5.2, a formal description of the independent optimization problem in its various formulations is given. Moreover, an algorithm that can deal with this problem is presented.

Then, in Chapter 6, a complete video coding system using the proposed algorithm is presented. The basic form of the coding system (i.e. without neither scalability nor content-based functionalities, the main goal being coding efficiency) is described, paying special attention to the optimization algorithm itself.

5.2 Problem Formulation

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

It is to be noted that, when coding a source (a video source in our case), there is always a data partitioning step. This is, the coding process does not take the signal as a whole, but a preliminary decomposition into smaller units is performed. Then, these smaller units are coded separately. When it comes to decide how to distribute a given bit budget among these signal units in a optimal way, the particular structure of this decomposition must be taken into account. It is clear that different coding systems can make different partitionings of the input source. Additionally, although these signal units are coded separately, this does not mean that they are independent. In fact, the decisions that are taken in the coding of a signal unit can affect the coding of other signal units (examples of this statement are motion compensated coding or pyramidal coding).

Let us examine the encoder framework in order to define properly the problem to be solved.

- **Segmentation-based coding system**: As it has been stated before, the problem is being analyzed into the framework of segmentation-based systems. This means that the optimization process not only has to find the optimal encoding parameters for each region, but even the shape of the regions; that is, the structure of the partition itself. Another important assumption is that regions are the smallest units subject to the optimization process. That is, the encoder has to be capable of adjusting parameters separately for each region.

- **Cost and distortion**: In this work, the strategy followed is to actually encode the image data in order to compute rate and distortion. Other approaches exist based on a modelization of the R-D curves [60]. This second approach sacrifices accuracy for a much lower computational cost. In our work, the coexistence of multiple coding techniques makes this approach impractical, because the modelization of the R-D curves for different techniques may not be comparable and may lead to meaningless results.

  To compute the cost of each region, the following elements are taken into account: its texture coefficients, one motion vector, the contour description of the region and some overhead information (See Section 6.2).

  Any additive measure of distortion can be used. Squared error has been used as it gives good results. Additionally, it is widely used and permits an easy comparison with other approaches.

The optimization problem can be tackled in many ways. One is to ensure that each frame is optimal *by itself*, this is, without taking into account the influence of the decisions made in the coding of previous frames. 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. This second approach can improve the coding results at the expense of increasing the computing complexity.

5.2.1 Independent optimization

In this case the objective is to find an optimal partition and the best distribution of the available bit budget between the resulting regions, ignoring the temporal and spatial dependencies between different frames that may exist. These dependencies are ignored to limit the computational complexity of the algorithm. The optimization is carried out within each frame of the sequence.

Let $\mathcal{P}$ be the set of possible partitions that are to be studied for optimality, $\mathcal{Q}$ the set of available texture coding techniques, $P \in \mathcal{P}$ a given image partition, $Q \in \mathcal{Q}$ a choice of techniques/quantizers for each region of the given partition P, $R(Q,P)$ the number of bits necessary to code P with quantizers Q, and $D(Q,P)$ the associated distortion.

Then, the problem to be solved can be stated as:

$$\min_{P\in\mathcal{P},Q\in\mathcal{Q}} D(Q,P) \quad \text{where} \quad R(Q,P) \leq R_{budget} \tag{5.1}$$

This is a complex constrained optimization problem. Using Lagrangian relaxation, it can be converted to an equivalent unconstrained problem [28, 124]:

$$\min_{Q\in\mathcal{Q},P\in\mathcal{P}} J(Q,P) \tag{5.2}$$

where J represents the Lagrangian cost that results from merging rate and distortion through the Lagrange multiplier $\lambda \geq 0$.

$$J(Q,P) = D(Q,P) + \lambda R(Q,P), \tag{5.3}$$

For a given partition P, this problem can be solved by an iterative fast convex search algorithm. To find the optimal partition together with the bit allocation, a variation of the technique discussed in [105] is used [82]. The process is started by computing a set of hierarchical partitions (Partition Tree), ordered from finer to coarser (see Figure 5.1). The hierarchy comes from the fact that finer partitions contain the coarser ones, so that a tree-like structure can be defined with parent-children relationships between regions on consecutive levels. Then, a dynamic programming algorithm is used to compare the Lagrangian cost of

each 'father' region with the cost of its 'children' regions, and depending on the result, a decision is made (see Figure 5.1) on whether to keep the father or the children regions.

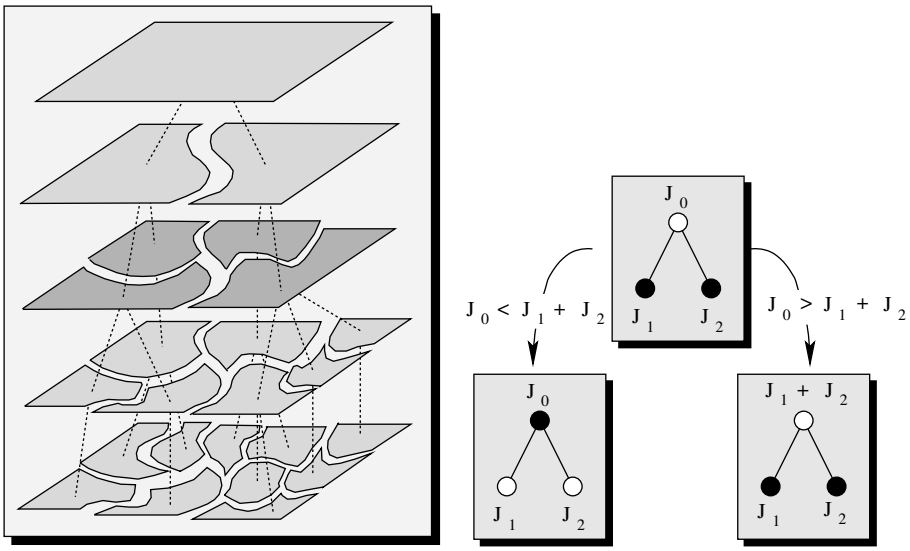


Figure 5.1: Partition Tree and local decision

The definition of the optimum λ can be done with a gradient search algorithm. The algorithm starts with a very high value λ_U (10^{20}) and a very low value λ_L (0). For both values of λ, a local analysis is performed and, for each region in each level, the Lagrangian J(λ) for each coding technique is computed. In each case, the technique giving the minimum J(λ) is considered as the optimum one for this region and stored. Then, in order to define the best partition, a bottom-up local analysis of the Partition Tree is done. Starting from the lowest level, the Lagrangian of each node is compared with the sum of Lagrangian's of the nodes that emanate from it. If the cost of the 'father' node is less than or equal to the sum of the costs of the 'children' nodes, the tree is pruned at this level. Otherwise, we look at the next level. This procedure is applied recursively until all the tree has been analyzed. The additivity in rate and in distortion is mandatory.

This pruning criterion is shown in figure 5.1. It can be formulated as follows:

$$\text{prune if} \quad (D_{child_1} + \cdots + D_{child_N}) + \lambda(R_{child_1} + \cdots + R_{child_N}) > (D_{parent} + \lambda R_{parent}) \quad (5.4)$$

That is, if the Lagrange cost of the 'father' node is lower than the sum of Lagrange costs of the 'children' nodes, this node becomes active. Its associated region in the Partition Tree is selected to belong to the resulting partition. The 'children' nodes and all nodes that are below them are deactivated. If the Lagrange cost of the 'father' node is larger than the sum of Lagrange cost of the 'children' nodes, these nodes are marked as active and the 'father'

node remains deactivated. In a rate-distortion sense, it is better to split the region and to code the resulting regions.

This process is performed for both λ_U and λ_L. Each value of λ defines a partition and an associated coding rate R. If R_U or R_L is equal to the budget, the process ends. Otherwise, a new value for λ is computed and the process is iterated until a value of λ is found such that the cost of the resulting partition is less or equal than R_{budget}. The algorithm can be stated as:

- Find $\lambda_L > \lambda_U$ such that: $R_T(\lambda_L) \leq R_{budget} \leq R_T(\lambda_U)$.
- Apply the pruning criterion of Eq. 5.4 from full-depth tree to root. An optimal partition for the current λ is found.
- If $R(\lambda_L) = R_{budget}$ or $R(\lambda_U) = R_{budget}$ the solution is found and the algorithm stops. Otherwise, a new value of λ is computed:

$$\lambda_n \leftarrow \frac{|D_T(\lambda_L) - D_T(\lambda_U)|}{|R_T(\lambda_L) - R_T(\lambda_U)|} \tag{5.5}$$

  if $R(\lambda_n) le R_{budget}$ then $\lambda_L = \lambda_n$. Otherwise, $\lambda_U = \lambda_n$

The two last steps are iterated until $R(\lambda_n) = R_{budget}$ or until no further reduction in the quantity $R(\lambda) - R_{budget}$ is possible. The algorithm converges very fast. Usually, five to ten iterations are enough to obtain the optimal results. At the end of the algorithm, the result is a valid partition and the choice of coding techniques for each region that minimizes the Lagrange cost of the current frame.

The algorithm can also work at constant-quality and variable bit rate. This is, a target quality value that must be reached with the minimum coding cost. If the Lagrange cost is defined as J'(λ) = R + λ D, the same optimization algorithm can be used.

Chapter 6

Description of the basic coding system

The objective of this section is to give an overview of the coding strategy, and to show the structure and details for the basic version (i.e. without neither scalability nor content-based functionalities) of the SESAME video coder, which is used to demonstrate the optimization algorithm proposed in the previous section.

A first version of this algorithm was proposed within the framework of MPEG-4 under the name of SESAME. A detailed description of the algorithm and of its hardware complexity can be found in [16].

The basic points to take into account about the basic coding system are:

- It is a segmentation-based system. Segmentation involves both spatial and motion homogeneity criteria.
- It uses motion compensation for both the texture and the partition in order to obtain an efficient representation. That is, a prediction for both the partition and the texture of the current frame is created using the last coded frame and motion information. The objective is to exploit temporal redundancy.
- The time evolution of each region is defined (time tracking). This will prove to be essential to properly address content-based functionalities.
- The results are optimal (or near-optimal) for the given framework in a rate-distortion sense. The definition of the coding strategy involves a global optimization of the partition as well as of the coding technique/quality level for each region.

The block diagram corresponding to the encoding process is presented in Figure 6.1.

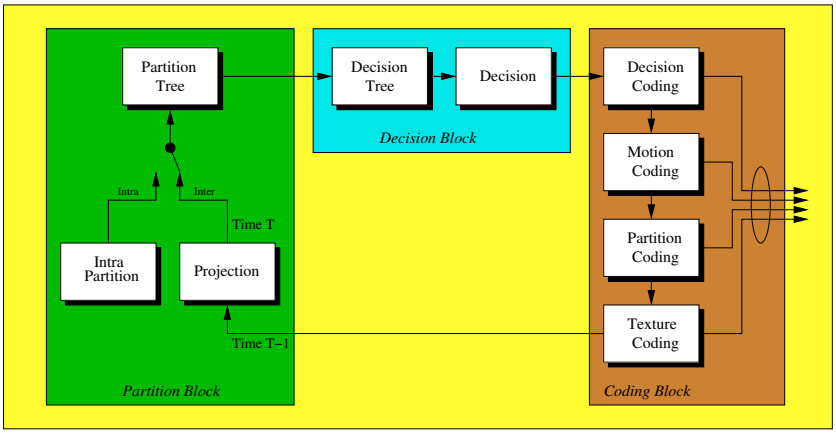


Figure 6.1: Scheme of the segmentation-based coder

The encoding process relies on three main blocks: Partition block, Decision block and Coding block.

The *Partition block* constructs for each frame of the sequence the universe of regions out of which the Decision block will construct the final partition. This universe of regions is constructed so that they represent the image with various levels of resolution, from finer to coarser. The universe of regions has to offer to the Decision block the possibility to introduce new regions that may appear in the scene or to eliminate past regions that are no longer needed. For coding efficiency reasons, this set of regions should be related to the partition of the previous frame. This will be also a key factor to the ability of the algorithm to track objects.

The *Decision block* makes the link between the analysis (partition block) and the synthesis (coding block) parts of the encoder. This block is responsible of determining the optimal (in a Rate-Distortion sense) bit allocation among the various types of information to be coded and transmitted (motion, partition, and texture). In a SBCS this involves: a) determining the optimal partition by selecting regions out of a set of possibilities (provided by the Partition block) b) selecting the appropriate texture coding technique and quality level (defined by the quantizer choice) to use in each of the resulting regions.

The *Coding block* is responsible of encoding the various types of information necessary to reconstruct the sequence at the decoder, i.e. motion, partition and texture information, as well as the coding strategy information.

In the sequel, the various processing steps are further explained and discussed.

6.1 Partition block

The optimization algorithm presented in Section 5.2 follows an operational rate-distortion approach. This means that the decision algorithm will make a choice among a set of given possibilities. The objective of the Partition block is to construct a universe of regions, representing the image with various levels of detail, that will be provided to the Decision block to be used as a basis to select the final partition. This is an *analysis* step, involving the search and the characterization of the features present in the current scene (size and position of regions, motion).

There are various considerations to be made in the way this set of partitions must be constructed: In the first place, to improve coding efficiency, the regions must be related to the ones in the partition of the previous encoded frame. This way, both the partition and the texture of the current frame can be predicted accurately, using the last coded frame and motion information. The result is that the cost of coding the compensation error decreases. In this process, the time evolution of each region along the sequence is defined (time tracking). This temporal link will enable the algorithm to track objects when dealing with content-based functionalities. This time evolution is ensured by deriving the universe of regions at time T from the *projected partition*, this is, a prediction for the partition at time T constructed by applying motion information to the partition at time T-1 and adapting the result to the spatial information in frame T.

To limit the complexity of the Decision process, the set of regions must be of reasonable size and constructed in a hierarchical way. The size of the universe of regions is mainly a computational cost consideration, because the cardinality of this set has a high impact on the processing time of the whole process. The hierarchical structure will enable the decision algorithm to adequately compare regions at the different resolution levels so that the optimal trade-off between quality and cost can be globally determined by means of a series of local decisions (see Figure 5.1).

Another point to take into account is that this universe of regions must enable the decision algorithm to introduce fluctuations with respect to the projected partition. This is necessary because in generic sequences new objects can appear at any moment, while some objects may disappear. Moreover, for coding efficiency reasons, it may be useful to merge regions that have a similar motion (and thus can be compensated together) to limit the number of regions and thus avoid the extra cost of additional motion and shared contours.

To meet the above requirements, the approach that has been used is a two steps process, represented in Figure 6.1 by the *Projection* and the *Partition Tree* blocks: First, in the *Projection* block, a projected partition is created by applying motion information to the partition of the previously coded frame #(n-1). This is, the partition of the previous coded frame is adapted to the data of the current frame. The objective of the projection is to relate

regions or objects present in the current frame with their references in the previous frame. The result is a partition for the current frame #(n) (the Projected Partition). This Projected Partition is the basis for the current frame partition. It provides the time evolution of the regions in the previous partition, i.e., it tracks the regions over each frame of the sequence.

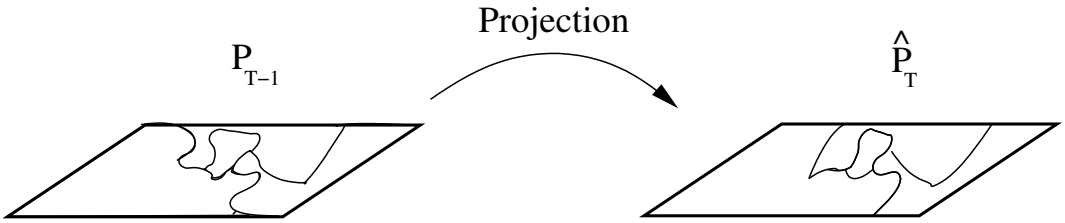


Figure 6.2: Example of projection

The projection is performed in two steps. First, motion is estimated between the original frames I_{T-1} and I_T, using a classical block-matching algorithm, and the previous image and its partition are motion compensated ($I_{T-1} \rightarrow \hat{I}_T$ and $P_{T-1} \rightarrow \hat{P}_T$) [119]. Then, the definition of regions into the current frame is achieved by a 3D watershed algorithm [79] that extends the compensated past regions (called markers) into the current frame [94]. That is, previous regions are projected into the current image and are used as markers in order to drive the segmentation process. In this process, motion information is used in the marker projection step and spatial information is used in the region growing process that extends the markers into the current frame. Some areas in the image may not be clearly assigned to one region or another after the marker projection step. These are called *uncertainty areas*. Pixels $\{p_i\}$ in the uncertainty areas are assigned to regions $\{R_j\}$ in an order defined by a distance function that takes into account three different types of information: texture, contour complexity and deformation.

$$Cost(p_i, R_j) = \alpha_1 dist_t(p_i, R_j) + \alpha_2 dist_c(p_i, R_j) + \alpha_3 dist_d(p_i, R_j) \tag{6.1}$$

where $dist_t$ measures the difference from the gray level value of p_i with respect to the mean value of R_j, $dist_c$ is related to the increase in the contour complexity of R_j if p_i is assigned to it, and $dist_d$ measures the deformation of R_j with respect to the projected marker if p_i is added to it. The use of this cost function provides a good stability of the labels through the time domain, and therefore allows an efficient tracking of the objects [65].

To avoid problems due to the spatial inhomogeneity in merged regions (with motion homogeneity), the projection is carried out using a double partition. This is, instead of projecting directly the previous partition, the regions in P_{T-1} are first re-segmented using spatial criteria. This results in a finer partition, P^R_{T-1}, with regions that are spatially homogeneous. This partition is motion compensated and the result $\hat{P}^R_T$ is a fine representation of the current

image.

This partition has too many regions to be practical as a projected partition, so a merging step is carried out. Only the contours in $\hat{P}_T^R$ that are associated to contours present in P_{T-1} are preserved. This is done by re-labeling $\hat{P}_T^R$ using the fact that the algorithm keeps the information of the origin in P_{T-1} of each region in P_{T-1}^R.

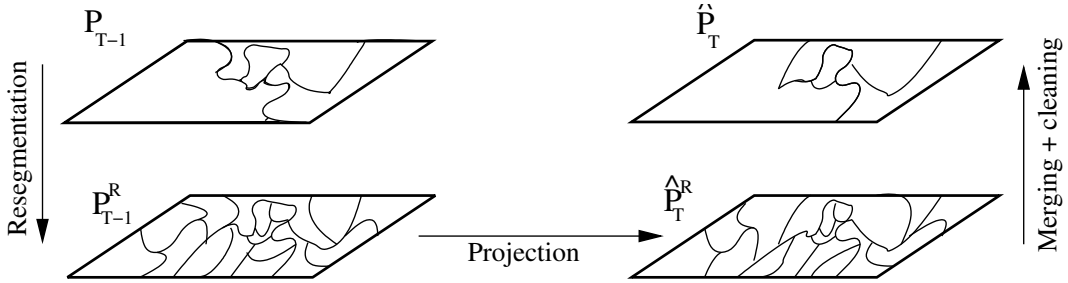


Figure 6.3: Double partition projection process.

However, the projection and re-labeling of the fine partition does not ensure that an actual partition is obtained. Indeed, unconnected regions may have the same label after re-labeling. This problem is solved by a cleaning step that consists in keeping the largest connected component for each label and removing the other components. The pixels of the resulting uncertainty areas are then assigned to the neighboring regions by means of a 2D watershed algorithm.

Note that the Projected Partition may not be optimal. Besides, the possibility to introduce fluctuations with respect to the partition of the previous frame (introducing or eliminating regions) is severely restricted because the scene is represented at only one resolution level.

This leads to a second step, that consists on constructing a universe of regions out of this Projected Partition. These regions will be candidates to form the final partition for the current frame. This universe of regions is created by splitting or merging regions present in the Projected Partition. Merging is done on a motion similarity basis, and splitting is performed by re-segmenting this Projected Partition on a spatial basis [97, 82].

The result is a set of hierarchical partitions, called Partition Tree, formed by the Projected Partition plus a set of partitions that are generated from this Projected Partition (See Figure 6.4).

The partitions are constructed by means of two different procedures (See Figure 6.5): The upper levels of the tree, above the projected partition, consist on larger regions. These coarse partitions are generated by merging regions from the projected partition. The criterion used to merge regions is motion similarity. The aim is to merge neighboring regions which have a similar motion and thus can be compensated with the same motion parameters. Merged regions are more efficient in terms of coding cost, because they require only one set of motion

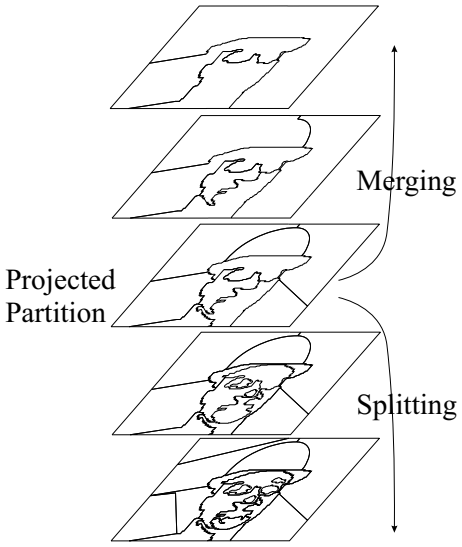


Figure 6.4: Partition Tree

parameters, one set of texture parameters, and there are less contours to code. However, this leads to an increase in the distortion. For the intra-frame mode, the merging of the upper levels cannot be based on a motion similarity criterion. Instead, a gray level similarity criterion is used.

Prior to the actual merging, motion is estimated for each region at a given level using a parametric model [22, 135]. Usually an affine model (defined by a polynomial of order one in x and y) is used, but it is also possible to use other models (translational or quadratic). The estimation itself is done by differential methods [120]. The result is a set of parameters that gives the time evolution of a region from frame T-1 to frame T. A backward estimation is used. Experimental tests show that for this framework, the affine model is a good compromise between quality, computing and coding costs.

When estimating the motion for regions in levels above the projected partition, there is a region correspondence problem that affects the motion estimation process. This is because these regions are constructed by merging regions in the projected partition. The estimation process computes the motion parameters for each region by looking for the region in the previous coded partition (P_{T-1}) with the same label value. In this case, there is more than one region (and more than one label) in P_{T-1} related to the region whose motion is being estimated, and thus, part of the region reference in P_{T-1} is not used in the estimation, leading to inaccurate results. This is solved by using a merged version of P_{T-1} in the estimation (see Figure 6.6).

Once the motion parameters for a given level are known, a merging cost is computed

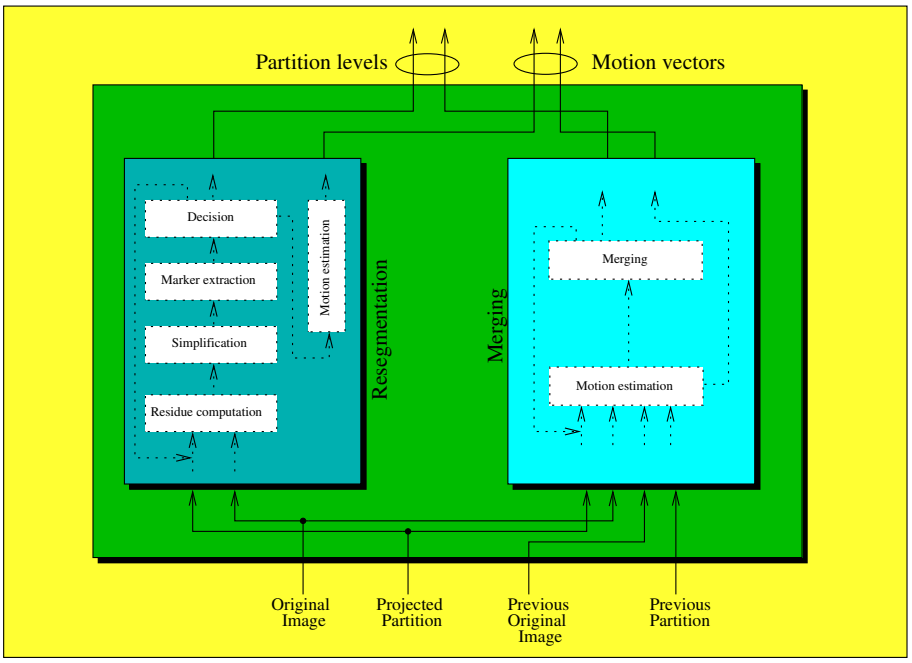


Figure 6.5: Construction of the Partition Tree

for every couple of neighboring regions. Then, the couples of neighboring regions with the smallest merging costs are merged until the required number of regions for the next level are obtained.

The lower levels of the tree, below the projected partition, are constructed by successively re-segmenting the projected partition. A constrained morphological segmentation is used to ensure that the contours of previous regions are preserved [118]. Regions of a given level are split to produce the regions of the next lower level. Each level has to refine the segmentation of the previous level by introducing new significant regions. The procedure is purely spatial: it does not take into account any motion information. As a consequence, the contours of new objects are based on spatial discontinuities.

The regions in the lower levels can represent new objects appearing in the scene or objects with several components that have been previously merged together but are starting to undergo different motions.

Motion parameters estimation is performed for all regions in the tree. These motion parameters will be used for inter-mode coding of the texture of these regions.

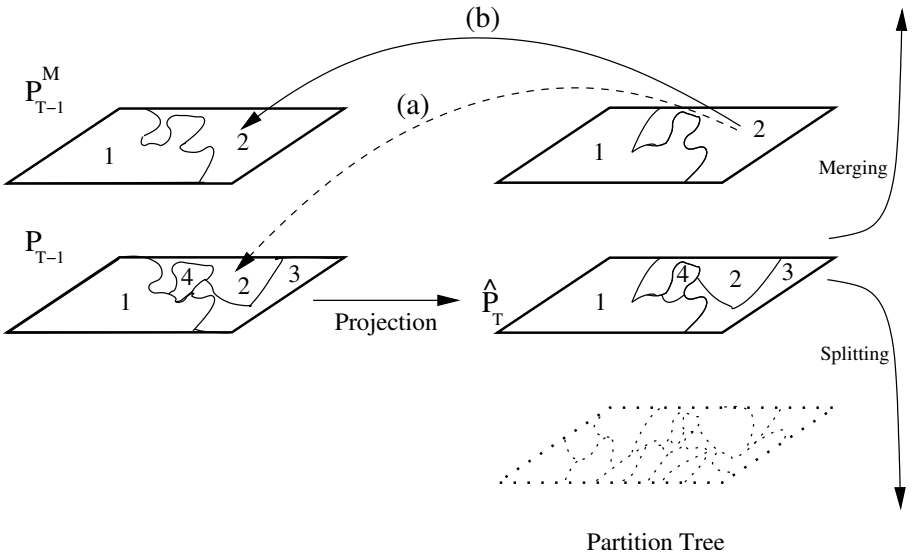


Figure 6.6: Estimation of the upper levels of the Partition Tree. If the estimation indicated by (a) is performed, region with label 3 in P_{T-1} is not used for the estimation/compensation, resulting in poor accuracy. If estimation indicated by (b) is used, all pixels related with this regions in P_{T-1} are merged in one region, and therefore the estimation/compensation is better.

6.2 Decision: Choice of the partition and coding strategy

The function of the *Decision* step is to select a set of regions that form a valid partition of the current frame, and the appropriate texture coders so that the coded image is optimal in a rate distortion sense (See Figure 6.7).

The fact that the contours of higher levels of the Partition Tree are preserved in the lower levels allows us to define the regions in the Partition Tree as a hierarchical structure, where each region in a given level is connected with the regions in the next lower level that originate from it in a father/children relationship.

6.2.1 Creation of the Decision Tree

The Decision Tree (not to be mistaken for the classification concept that shares the same name) is a data structure with the same hierarchical structure of the Partition Tree plus information that will allow to define the coding strategy. Each region of the Partition Tree is represented by a node in the Decision Tree [107, 82]. The Decision Tree should also convey the information about the coding cost (Rate) and quality (Distortion) of all possible texture coding techniques. Every node has associated a list of the rates and distortions that result from the coding of the corresponding region with each of the available coding choices (See

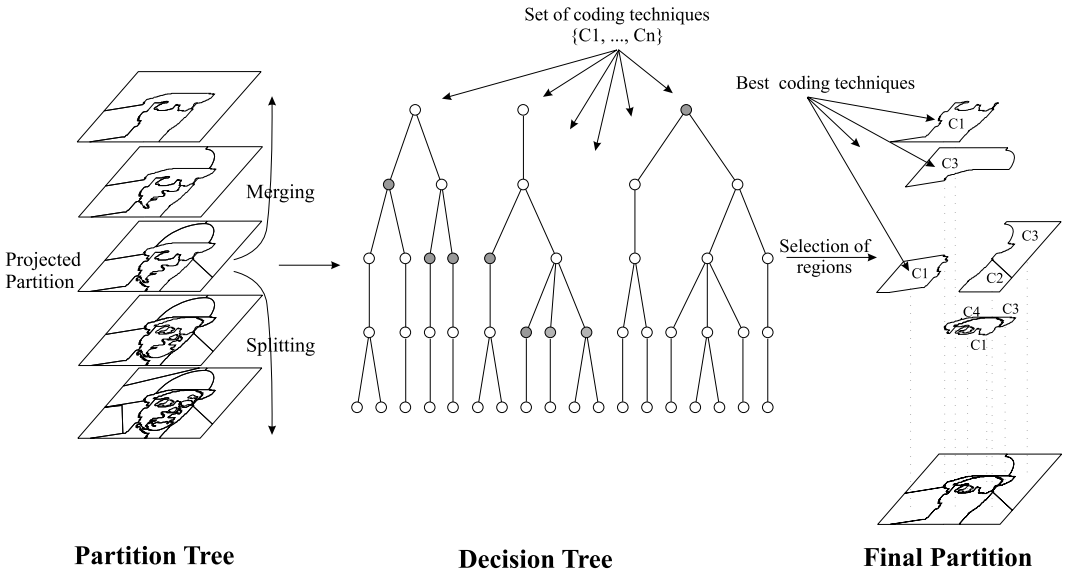


Figure 6.7: Decision Process

Figure 6.8).

This structure allows to compare the Lagrangian cost of a region with the sum of the costs of the branches emanating from that region for a given λ. To preserve the additivity of the rate and distortion among the levels of the tree, the contours of the parent regions must not change in the re-segmentation steps.

The texture of each region in each level is coded by all the available coding techniques, and the resulting rates and distortions are stored in the nodes of the Decision Tree. In this work, various coding techniques have been used: a shape-adaptive 2-D transform [127, 46], a generalized orthogonal transform [35], a region based wavelet decomposition [5, 16], and the gray level average of the regions, which is a special case of the previous techniques. Various quantization levels are available for each technique.

This approach gives a great flexibility, because if a new coding technique is developed, it is possible to put it into competition with the previous ones, without modification of the decision algorithm. All kind of coding techniques appropriated for dealing with regions can be used. The simultaneous use of various coding techniques can deal efficiently with different kinds of textures.

The coding depends on whether the current frame is sent on intra-frame or in inter-frame mode:

- Intra-frame mode: The texture of each region of the current frame is coded separately. Then, contours are coded by means of a chain-code algorithm.

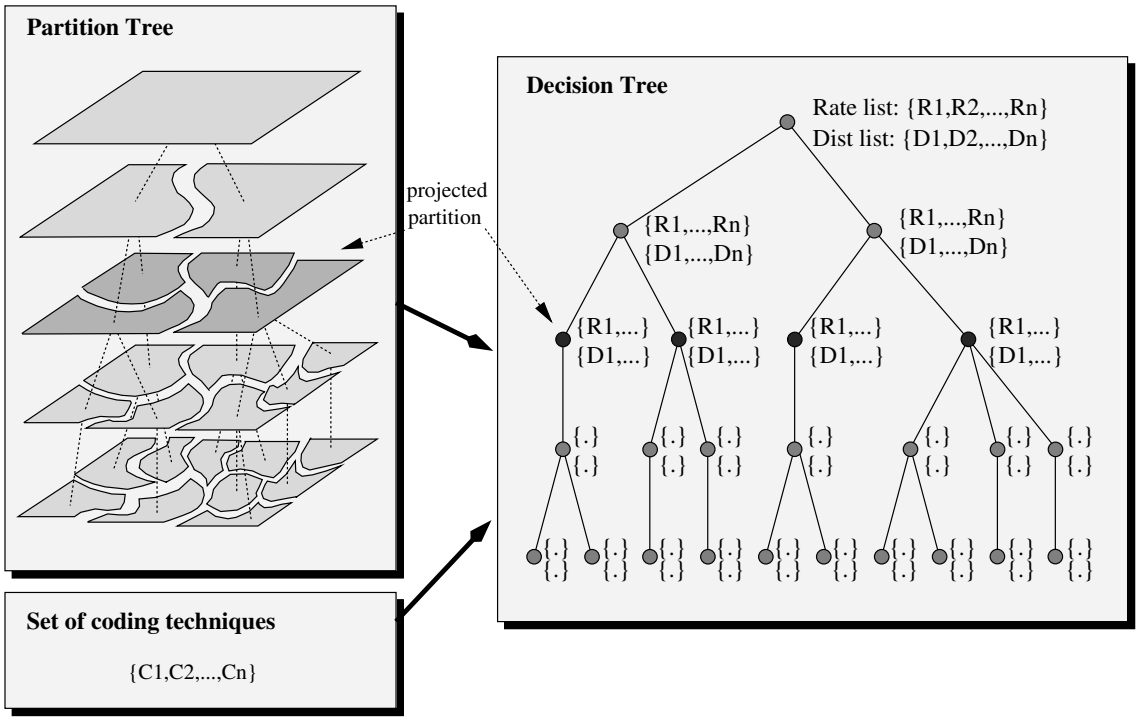


Figure 6.8: Construction of the Decision Tree

- Inter-frame mode: For each region in the frame there are two choices: Coding of the prediction error after motion compensation of the texture of each region, or coding of the original texture information as in intra-frame mode. While the coding of the prediction error is generally more efficient, the possibility to code the original texture information of the region may result in a better coding of new regions appearing in the current frame. Then, contours are coded by means of a modified chain-code algorithm [74]. Motion-compensation followed by error coding of the contours may result in a further reduction of the bit-rate [113].

The distortion measure that has been used is the squared error due to its simplicity. Anyway, any distortion measure that is additive in space can be used. Additivity in space means that if a region is split, the sum of the distortions of the resulting regions is equal to the distortion measured on the original region. In the optimization step, to prune the Decision Tree a comparison is performed between the Lagrange cost of each node and the sum of Lagrange costs of all its child nodes is. This requires a distortion measure compatible with this comparison.

The rate is a measure of the global coding cost in bits of each region. For each region, the total coding cost is the sum of the costs of the texture coefficients, motion parameters (in

inter-frame mode), and contour of the region. To take into account the effect of the entropy coder, the cost of texture coding is estimated by measuring the entropy of the coefficients. The cost of the motion coding is also given by entropy estimation of the model parameters. The cost of the contour coding is considered to be proportional to the perimeter of the region. Our tests show that these estimations give a very good approximation of the final costs.

6.2.2 Optimization Algorithm and Decision coding

The optimization algorithm will determine the optimal subtree of the Decision Tree together with the optimal coding model for each node of the tree. The resulting subtree must define a valid partition for the current frame. The algorithm approach is based on the Lagrange-multiplier method. See Section 5.2.1 for the problem formalization and a detailed description of the optimization algorithm.

The decision process has to decide about :

- The final partition, based on the projection and the Partition Tree.
- The best coding method for each region.

As seen in Section 5.2.1, an iterative approach is used, including both a global optimization that uses a convex search algorithm to determine λ^*, as well as a bottom-up dynamic programming local analysis of the Decision Tree in order to define the best partition.

The result of this decision step is the final partition and the information that will allow the receiver to know which texture coding technique has been used for each region.

This last information is called Decision in the sequel. It is used by the texture coding and decoding blocks to know how each region should be processed.

Concerning the first point, the final partition is the partition that has to be coded and transmitted. In order to help the coding process, the decision also gives some information on the origin of each region. In fact, the regions belonging to the final partition may come either from the projected partition (Projected regions), from the levels of the Partition Tree above the projected partition (Merged regions) or from the levels below the projected partition (Segmented regions). This piece of information is going to be sent to the receiver for the following reasons:

In SESAME, the texture-compensation can be done in two modes. One of them (called "with restriction") tries to solve the problem of uncovered areas. In this mode, a pixel is compensated only if it belongs at time T and T-1 to regions having the same label. Areas where no compensation has been performed correspond to uncovered background and a different prediction (mainly spatial prediction) is used. Now, in order to efficiently use this mode of compensation, the labels of the final partition should be carefully assigned. For the projected

regions, the label is simply the label as defined by the projection. The problem arises for Merged and Segmented regions. Indeed, if we consider that they are all new regions with new labels, the compensation with restriction will not assign any value for them. As a result, the texture of Merged and Segmented regions will always be sent in intra mode. To avoid this situation, SESAME sends Merging orders and Splitting information.

Merging orders are a set of numbers saying to the coder and the decoder the origin at time T-1 of merged regions at time T. For example, the region X at time T has been created by the union of the set of regions $\{x_i\}$ at time T-1. Both the emitter and the receiver will apply the same merging to the regions $\{x_i\}$ of the previously coded partition. From this point, the texture compensation with restriction will be able to use the pixels of the union of $\{x_i\}$ at time T-1 to predict pixels of region X at time T.

The Splitting information is not an order that implies some specific transformation. It simply defines that the set of regions $\{x_i\}$ of the partition at time T comes from the region X at time T-1. During the texture compensation, pixels coming from X are allowed to predict pixels of all regions $\{x_i\}$.

The Merging orders are also useful for the coding of the partition. Indeed, the partition coding relies on motion compensation. Before motion compensating the previous coded partition, its regions are merged according to the instructions contained in the Merging orders.

Finally, besides being useful for the coding of partition and texture, the Splitting and Merging information are a way to trace the history of the partition and the transformation of the regions.

6.3 Optimization of the algorithm for very low bit-rates

In the approach described up to here, every frame of the sequence is given the same bit budget. This results in frames with variable quality, so problems can arise when working at very low bit rates. Some frames may have very poor quality (scene changes, complex motion). To solve this problem, a very simple method can be used to give more bits to the frames with low quality.

The optimization works basically on a fixed nominal budget, but a minimum quality threshold is defined. If the coded image defined by the decision step does not reach this minimum quality, the budget is increased by steps of 5%, and the Decision step is performed again. This procedure is stopped when the decision has found the optimum strategy:

- The distortion is minimal.
- The budget is at least equal to the nominal budget.
- The signal to noise ratio is everywhere above a given threshold.

This way, the frame has better quality, and it may be possible to use it for the compensation of the next frame. This improves the prediction of the future frames, so they can be coded with less bits. The coding budget for each frame is not constant, but the sequence overall bit rate is very homogeneous, as it will be shown in Section 6.6.

6.4 Coding of the partition information

Partition coding in SESAME involves three different types of information: shape (contour of the regions), position, and the label of each region. The goals are good accuracy, reduced coding cost and preservation of the time evolution of the regions to enable tracking.

Although the coding strategy relies on a general approach that does not impose any specific contour representation, lossless or near lossless techniques are preferred. This is because a) the human visual system is very sensible to errors in the position of contours, and b) when coding the contours of partitions with multiple regions, lossy techniques may present problems in narrow regions, that can lead to changes in the partition topology in the decoding process.

6.4.1 Structure of the partition coding process

To improve the coding efficiency, the coding of the partition relies on motion compensation [112, 113]. A similar process as the one used for texture is used for partitions, that are represented with an image of labels. The process is depicted in Figure 6.9.

First, intra regions are encoded (this is, regions that the Decision block has selected for coding in intra mode). Then, the partition of the previous frame (P^*_{T-1}) is compensated using the motion information computed in the creation of the Partition Tree (a motion vector for each region, estimated in a backward mode). The result is a prediction for the partition of the current frame ($\hat{P}_T$). The prediction error is then computed by comparing this prediction with the actual partition created in the Decision step. A simplification of the prediction error is performed before the actual encoding. The resulting encoded partition is stored as a reference for the coding of future frames (P^*_T).

The encoding of intra-frame regions may be done with almost any coding technique (lossy or lossless). In this implementation, the modified chain code described in [74] has been used (See Section 6.4.2). Additional information has to be sent to the receiver to determine which regions are transmitted in intra-frame mode. With this information, a binary mask is created that will be used in the partition compensation step.

Note that texture and contour motions may not be the same. The motion of pixels inside a region (texture motion) may not be equivalent to the motion of the region shape. Only in the case of rigid foreground regions both motions are equivalent. But in the case of background regions, the modifications in their contours are defined by the motion of the

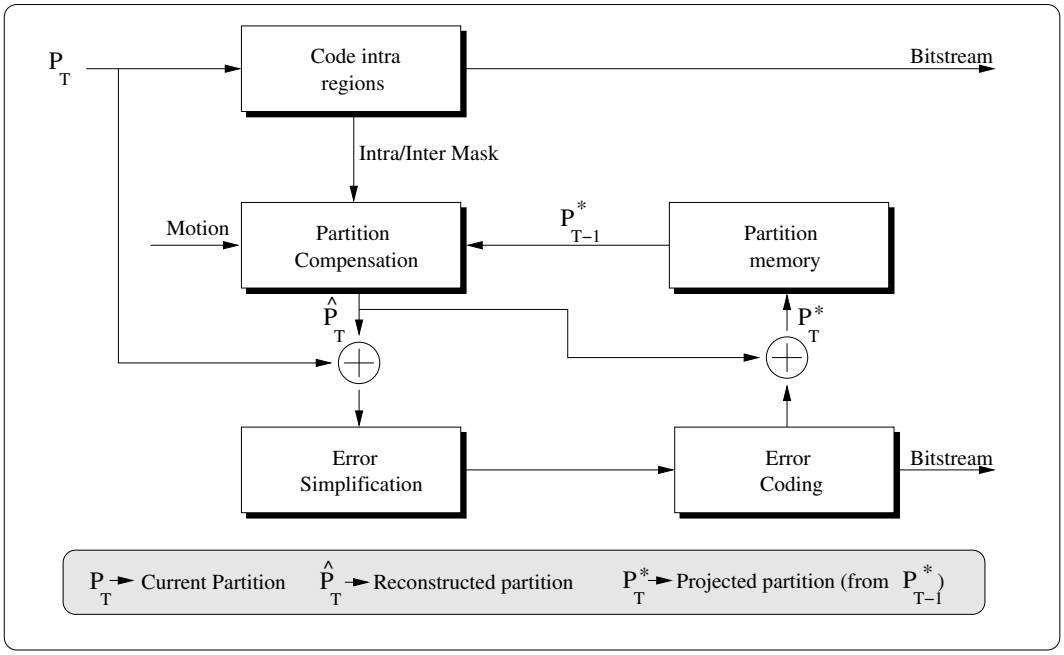


Figure 6.9: Motion compensation loop for partition coding

regions in its foreground. In this case, the use of additional information called *order* [96], that defines which regions are considered to be in the foreground, transmitted with the motion parameters, allows to solve the possible problems created by this situation.

The use of texture motion information plus order information is a quite efficient solution for compensating both texture and partition because the order information necessary to compensate the partition only represents a small amount of information. The reader is referred to [113, 116] for the detailed description of order estimation algorithms. Although there is the possibility to use two different sets of motion vectors for texture and contour as in MPEG-4 [48], the single set plus order information approach is better suited in segmentation-based systems where the amount of contour information is more important than in the MPEG case.

Once the partition has been compensated, and thus, a prediction of the current partition is available, inter-frame regions will be encoded. The process starts by computing the prediction error that can be then simplified, introducing losses in the coding process (See Figure 6.10). The coding of the simplified prediction error is done in a differential way. Indeed, the receiver already knows the contours of the compensated partition and therefore, only new contours have to be sent. Coding techniques similar to the ones used in the coding of intra-frame regions can be used.

The final step is to send the label of each region of the error partition, by using an appropriate strategy. The label information has to be restored on the receiver in order to be

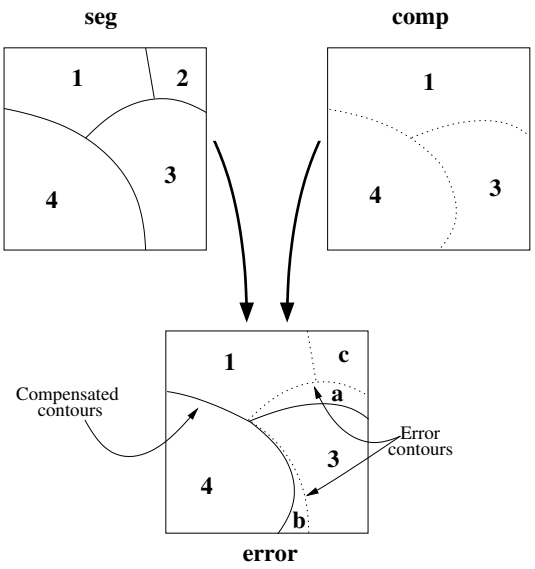


Figure 6.10: Error partition is constructed with the compensated partition *comp* and the current partition *seg*. Only new contours (the ones with the dotted line) are encoded. The appropriate labels for the error regions a, b and c must be also sent. In this example, the correct labels are a = 3, b = 4, and c = 2.

able to define and to track the time evolution of the various regions.

6.4.2 Contour coding with Chain Code

Chain Code techniques [30, 116] allow lossless coding of image partitions. Regions are represented by their boundaries and the coding process consists on tracking and encoding the boundaries. The process used to encode a partition is:

- Define an appropriate boundary representation. This usually leads to constructing a 'contour' image.
- Using the fact that two consecutive boundary points in a discrete grid are neighbors, encode the movements from one contour point to another, starting from a given initial point (or various), until the complete contour is tracked.

The contour of a given partition can be represented using an hexagonal contour grid [74]. In this case, a one to one relationship between partitions and their boundary representation can be achieved. In this type of representation, each pair of neighbor pixels in the partition is separated by a contour grid site. This contour grid site is labeled as *active* if the associated pair of pixels have different labels, otherwise it is labeled as *inactive*.

This concept is illustrated in the first example of Figure 6.11, where circular and line segment elements represent sites from the partition and contour grids, respectively.

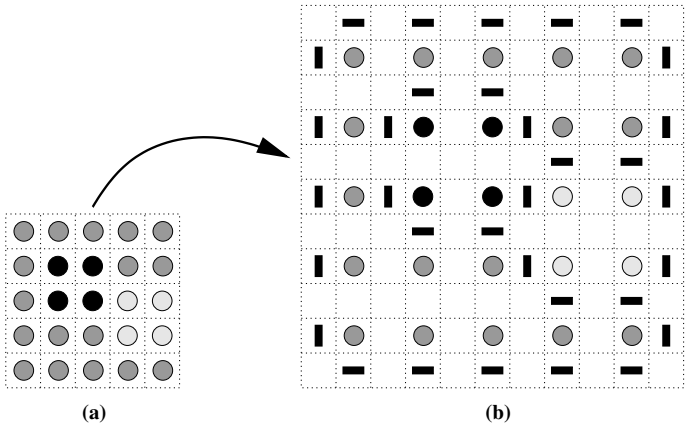


Figure 6.11: Relationship between partition and contour grid sites

Encoding of the contours is done by specifying the set of movements necessary to track the complete set of active contour grid sites. Shape and position information are jointly coded by introducing the location information in the chain code itself. The neighborhood system is 6-connected and, therefore, there are 6 basic movements. Nevertheless, as a derivative chain code is used [30], the amount of possible movements reduces to 3: turn right, turn left and straight ahead (r,l,s) (See Figure 6.12).

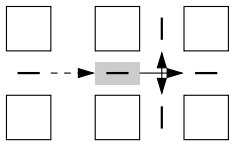


Figure 6.12: Movements in a hexagonal grid for DCC.

The intersections of contours are encoded 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.

Triple points are encoded by introducing a new symbol, say M, in the chain itself. Only a subset of the triple points are actually necessary to describe the whole set of contours. These triple points are called *active* triple points and only active triple points are marked in the chain by the symbol M.

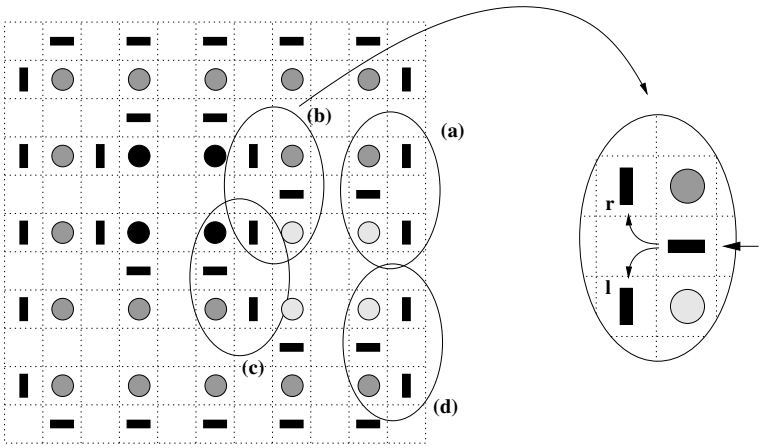


Figure 6.13: Example of triple points. The ellipses mark the triple points in the example partition. The right figure shows a detail of a triple point.

Regions that have no contact with other regions do not produce any triple point and have to be directly addressed by the coordinates of their initial points.

In the inter-frame case, differential coding of the contours is used. That is, as the decoder already knows the contours that correspond to the compensated partition, only the new contours have to be sent. The contour segments are encoded by sending the initial point (always located on already known contours) and the set of movements describing the segment. Note that it is not necessary to send the ending point. The decoder can detect an end of segment when an already known contour is reached by the new contour.

6.5 Coding of the texture

Once the partition has been coded, the texture of the resulting regions is encoded by using the techniques selected by the Decision step. The goal is to encode the texture and color information inside each region of an image partition.

The images are coded using the reconstructed partition. Inter-frame and intra-frame region-based coding techniques are used. The motion information computed in the creation of the Partition Tree is used to compensate the texture of the regions in the previous coded frame to build a prediction for the current image. In this case, the information to encode consists in the resulting compensation error (Inter-frame mode). This reduces the coding cost.

The regions where this prediction is poor or not possible (new regions appearing in the current frame or regions with motion not handled by the underlying model) are coded by themselves, without references to previous frames (Intra-frame mode). In this case the infor-

mation to encode is formed by the original pixel color values.

Motion compensation of the texture is used to build the best possible predicted image so that the error to be coded is minimal. This process relies on the following data: past and current partitions, motion information and last coded frame.

To improve the quality of the compensated images, a restricted motion compensation is used. This means that each pixel is compensated using the motion parameters of its region only if there is a correspondence between the label the prediction comes from and the current label. In the areas left empty due to this restriction, the texture is obtained by extrapolation of the compensated texture in a region by region basis [119]. This is important for uncovered areas, as only the texture information from the region this area belongs to is used for extrapolation.

The algorithm flexibility allows the combination of several texture coding region-based techniques. In this work, four techniques have been used: Shape-adaptive DCT [127, 46], region-based wavelets [5, 16], approximation onto orthogonal bases [35] and padding [51].

Al these techniques can work in intra- and inter-frame modes, with several levels of quantization for the texture coefficients.

6.6 Experimental results

Experimental results are presented for different test sequences in QCIF (176 x 144 pixels) and CIF (352 x 288 pixels) formats. The sequences have been recorded at a frame-rate of 30 Hz except for *Dancer* at 25 Hz. Only the first frame of these sequences has been coded in intra-frame mode. The first frame of each sequence can be seen in Figure 6.14.

- *Akiyo* is a typical 'head and shoulders' sequence, with small motion and a large static background.
- *Foreman* is a sequence with camera zoom, pan and several parts with complex motion, as well as occlusions and important scene changes.
- *News* is a sequence with a foreground nearly static and a moving background.
- *Weather* has a complex but static background that dominates the scene and the foreground motion is low at the beginning of the sequence but increases to the end of the sequence.
- *Mother & daughter* has a static background and moderate, low complexity motion at the foreground.
- *Dancer* is a composite sequence, with a moving background with new information entering the frame and a foreground with a very complex motion.

Figure 6.14: First frames of the test sequences. From left to right, top to bottom: *Akiyo*, *Foreman*, *News*, *Weather*, *Mother and daughter* and *Dancer*

Let us start by an overview of the Partition Tree creation and the Decision algorithm. Figures 6.15 and 6.16 show the partitions that define the Partition Tree for frames #180 and 149 of Foreman and News sequences respectively, as well as the final partitions found by the optimization algorithm.

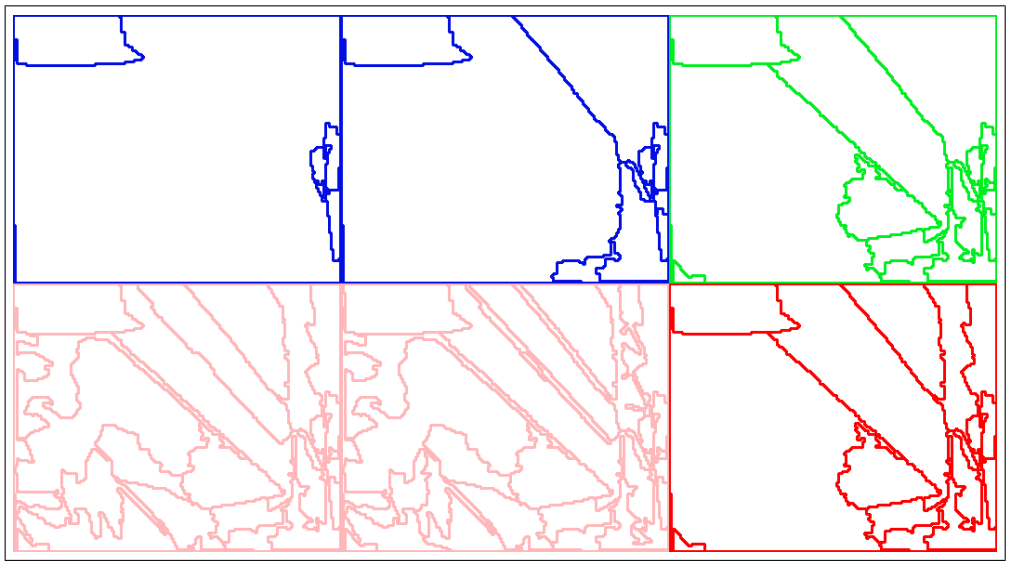


Figure 6.15: Segmentation Tree for frame #180 of the Foreman sequence. Merging levels are plotted in blue, the projection level in green and re-segmentation levels in purple. The final partition is represented in red.

In both cases, the partition tree is composed by the projected partition, two merging levels and two splitting levels. The final partition is represented in the bottom-right image in red color.

In the example given in Figure 6.15, the derived final partition is very similar to the projected partition (represented in green color) and thus, the algorithm do succeed to maintain a high degree of temporal region coherence. The only differences with respect to the projected partition are in the right part of the image, and are due to new regions necessary to represent the information entering the scene due to camera pan. In the News sequence, most of the image is almost static except for a region with strong motion (the screen with the ballerina), so the optimization algorithm spends most of the available bit budget on this region by assigning a low value for the texture quantizer. The rest of the regions are almost static and thus, can be adequately temporally predicted, so they can be given a higher value of the texture quantizer. In this case, temporal region coherence can only be maintained for the remaining regions.

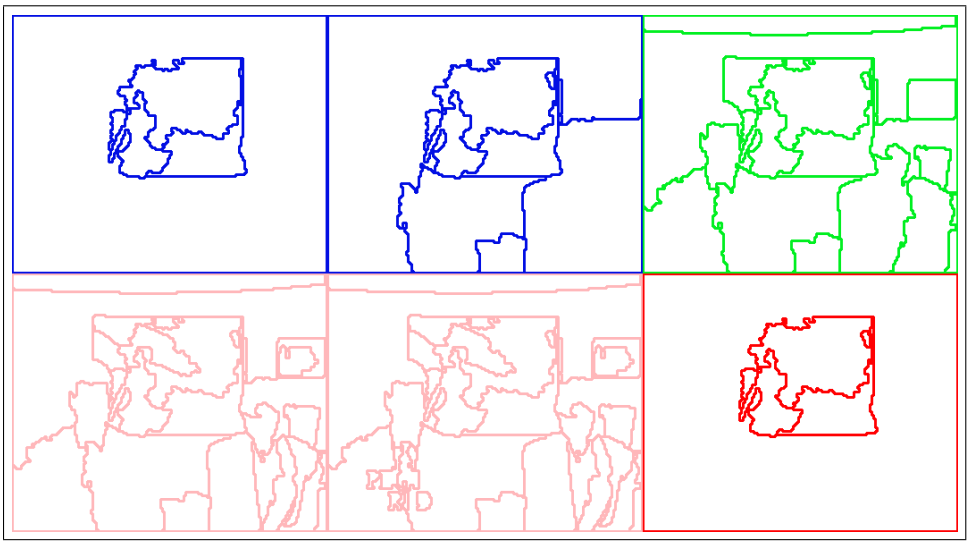


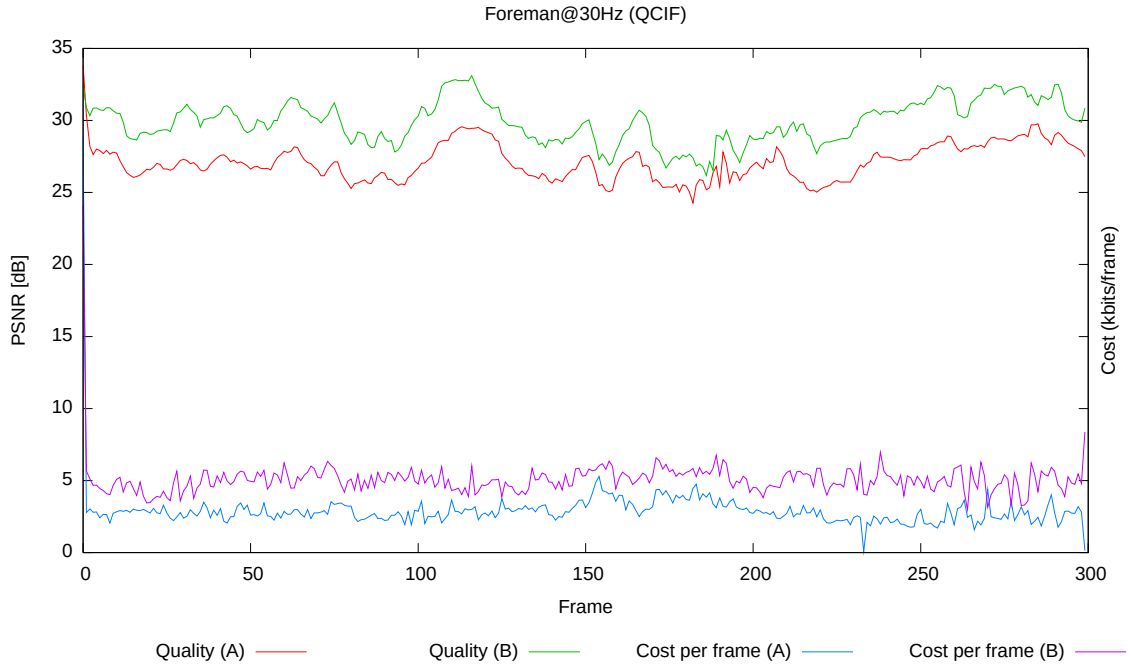
Figure 6.16: Segmentation Tree for frame #147 of the News sequence. Merging levels are plotted in blue, the projection level in green and re-segmentation levels in purple. The final partition is represented in red.

Figure 6.17 (a) shows the evolution of PSNR and 'per-frame' coding cost through all the frames of the *Foreman* sequence, encoded at 82 kbps (A) and at 152 kbps (B). Figure 6.17 (b) presents the same results for the *Dancer* sequence, encoded at 338 kbps (A) and at 542 kbps (B).

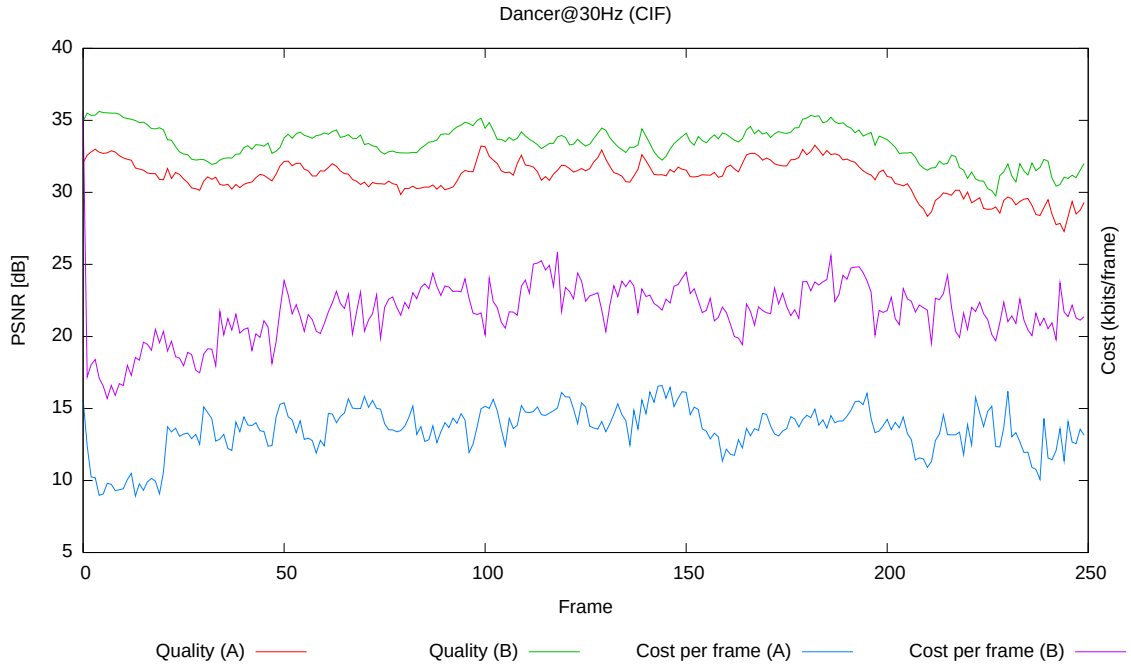
Note that the rate is not strictly constant because the set of techniques/quantizers can not always ensure that an exact optimal solution exists for the constrained problem. In these cases, the solution selected is the one that gives the bitrate closer to the given budget. Exact bit allocation could be achieved by choosing always the solution just below the given budget and feeding the remaining bits to a greedy algorithm. Another simpler and possibly more useful solution would be to use a buffer at the output of the encoder.

An optimization algorithm aimed at constant quality could be implemented by simply altering the Lagrangian function to minimize. Instead of this solution, to avoid frames with low quality, the SESAME encoder allows selecting a minimum quality for the coded frames. If the available bit-budget for a given frame does not suffice to provide this minimum quality, it is increased for this frame until the desired quality is reached (See Section 6.3).

The results for the bit allocation among the different types of information (decision, motion, contour and texture) are summarized on Table 6.1. Information related to the allocation (that is, technique/quantizer for each region and splitting/merging information) takes usually around 1% or less of the bitstram. Texture information logically dominates in the bitstream.



(a)



(b)

Figure 6.17: Evolution of PSNR and cost per frame along the Foreman and Dancer sequences (the same scale is used on left and right axes)

Sequence	Decision	Motion	Contour	Texture	Total (bits)	PSNR [dB]
Foreman (qcif)	1.0%	5.8%	17.5%	75.7%	2761.0	27.0
Mother & Daughter (qcif)	1.0%	2.8%	7.1%	89.1%	2294.8	31.1
News (qcif)	0.5%	2.8%	6.5%	90.2%	2142.4	29.0
Weather (qcif)	0.5%	3.6%	8.7%	87.2%	2322.1	25.9
Dancer (cif)	1.0%	8.3%	38.5%	52.2%	13536.9	31.1

Table 6.1: Coding results for different test sequences at 30Hz. (PSNR and cost in bits averaged per image)

The *Decision* algorithm adaptively makes different allocations depending on the bit-rate and the characteristics of the sequence. At low bit rates, the amount of information used for contour and motion is more significant than at high bit rates. Additionally, in sequences with complex motion and scene changes motion and contour information are more relevant. This can be seen by comparing the results for *Foreman* (complex motion, scene changes) and *Mother & daughter* (simpler motion, no scene changes) sequences.

Figure 6.18 compares the performance of SESAME with the MoMuSys implementation of MPEG-4 [85] and the H.264/AVC Reference Software Encoder, JM 8.2 [15]. H.264 has been selected because it represents the state of the art for coding efficiency, although has no provision for content-based functionalities. MPEG-4 is the only established standard with support for content-based coding. The results show that, in terms of coding efficiency, the H.264 reference encoder is much superior to the other two systems due to its new improvements. The MoMuSys encoder also provides results superior to SESAME.

The main reason for this difference in raw coding efficiency between the SESAME encoder and both standard codecs is due to the largest maturity of block-based techniques. Nevertheless, it has to be pointed out that the goal of this thesis has been to demonstrate the ability of an object-based video coding system to provide content-based functionalities rather than to compete in coding efficiency with the newest standards. In order to improve the coding efficiency of the SESAME encoder, advanced prediction techniques such as those implemented in MPEG-4 and H.264 should be implemented. The motion estimation techniques used in MPEG-4 and H.264 are much more complex and efficient than the one used in our work; half-pixel or quarter-pixel refinement, advanced prediction, unrestricted search are used in these schemes to increase the prediction efficiency. In contrast, in this work a simple integer motion search for regions has been used for the sake of simplicity. Texture coding techniques are also much simpler than the ones used in MPEG-4 of H.264, without enhancements such as DC/AC prediction and without the fine tuning present in the mentioned standards.

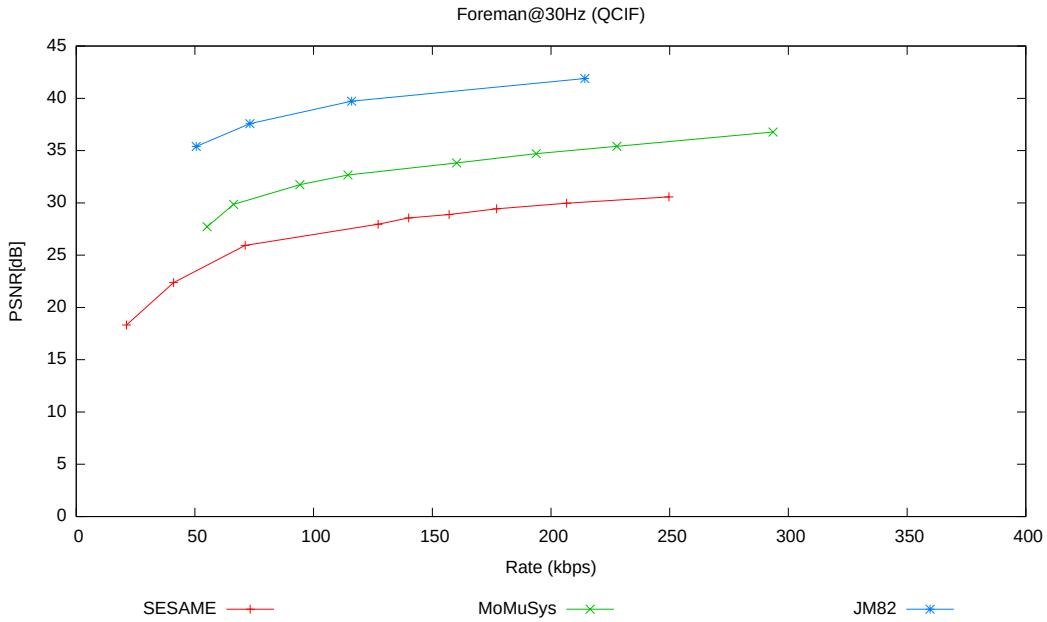


Figure 6.18: Comparison of RD performance between SESAME, MoMuSys and JM 8.2 encoders. Values are for the Foreman sequence (QCIF) at 30 Hz. Values of rate and distortion are per frame averages over 300 frames

This being said, the proposed scheme brings specific features for content-based functionalities that are not provided by MPEG-4 (and, of course, neither are they by H.264): In SESAME there is separate syntax in the bitstream, with independent access to shape, motion and texture fields; motion vectors are region-based, which gives a better representation of object motion and facilitates object tracking. On the contrary, MPEG-4 is a block-based method and uses a combined syntax, where data are encoded on a block by block basis, including motion and shape information, restricting separate access to the different information fields, thus difficulting content-based functionalities.

Figures 6.19, 6.20 and 6.21 show the decoded images for various frames along the video sequences *Akiyo*, *Mother and daughter*, *News* and *Weather*. The coding artifacts are very similar to those found in MPEG-4 because the texture coding technique used is a simplified version of the 'padding' technique used there. The decision algorithm usually selects partitions with a low number of regions. In all the examples given in the following examples (QCIF) less than 10 regions are used, and most of partitions are composed of two of three regions. Often, there is only one region covering all the frame (see for example the partition in the fourth row in Figure 6.20). This is because contours are expensive and because the texture coding techniques being used are better suited for rectangular regions. The algorithm only selects to introduce arbitrarily shaped regions in the cases where there is motion present and, therefore,

the quality prediction may be inaccurate. For example, in Figure 6.19, the static background is never segmented, and regions appear always in the body and head of the anchorwoman. The same behavior can be observed on the other results presented in Figures 6.21 and 6.20

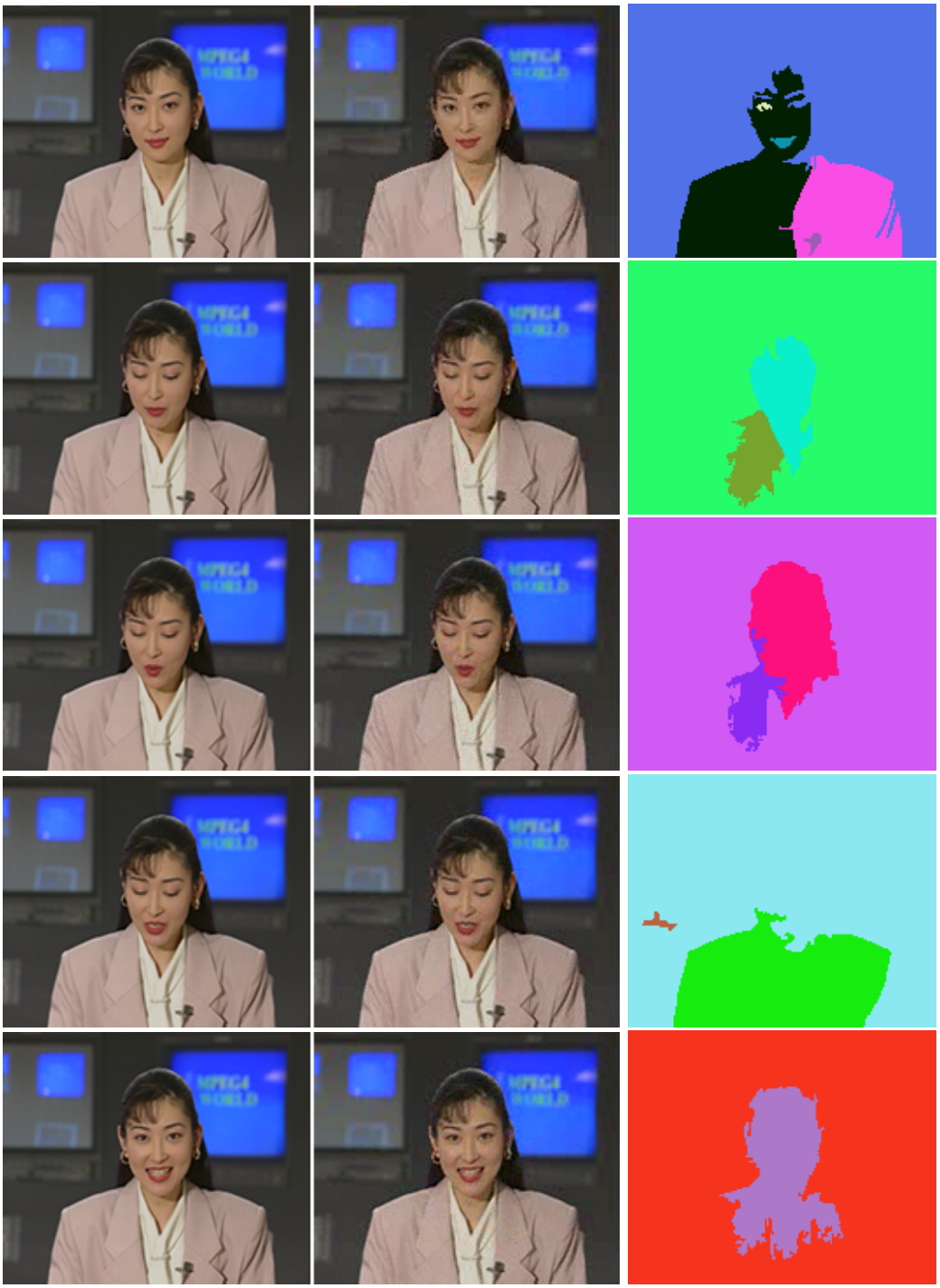


Figure 6.19: Akiyo, 10Hz@62kbps, frame #0 (intra) and #60, #120, #180 and #240 (inter)

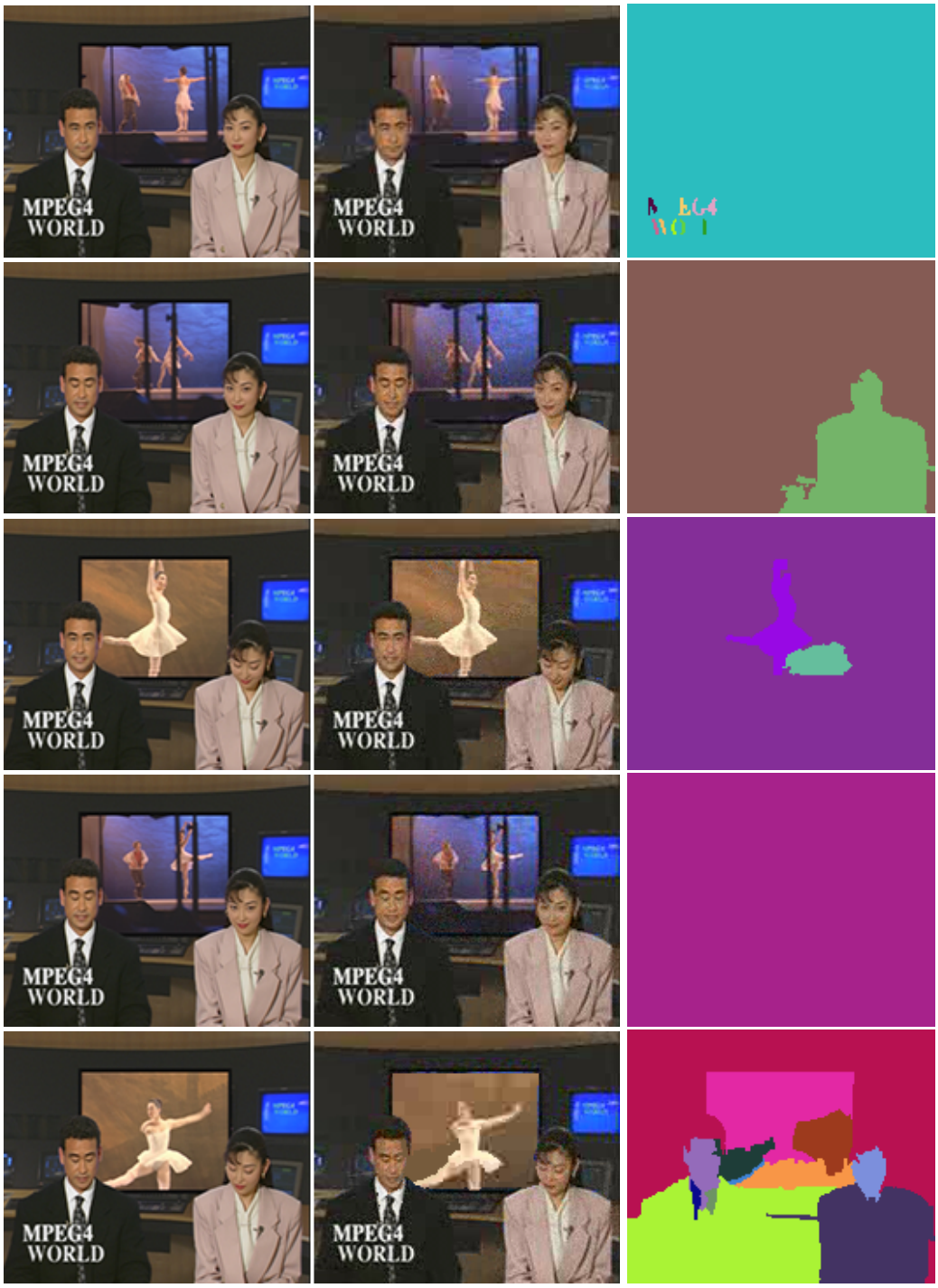


Figure 6.20: News, 15Hz@64kbps, frames #0 (intra) and #60, #120, #180 and #240 (inter)

Figure 6.21: Mother and daughter, 10Hz@64kbps, frames #0, #60, #120, #180 and #240