130

ITU Telecommunication Standardization Sector
Study Group 15, Working Party 15/1
Expert's Group on Very Low Bitrate Video Telephony

LBC-97-029
February 1996

Anlage D 7
i.Sa. Google / VSL
Nichtigkeitsklage vom 8. April 2015
Bundespatentgericht München
Quinn Emanuel LLP

Title: **Proposal for Advanced Video Coding**
Source: **Nokia Research Center[1]**
Purpose: **Proposal**

Introduction

This document presents a low bit rate video coding scheme developed at Nokia Research Center. The simulation results reported in this document indicate that the scheme consistently achieves higher coding efficiency than the H.263 video coder. This contribution is intended as a continuation of the proposal presented in the document LBC-96-91 at the July 1996 London ITU-T meeting.

The Appendix of this document contains a detailed description of the proposed video coder. It gives a brief description of novel elements of the coder as well as a description of the bitstream syntax.

Video coder

The video coder presented in this contribution contains three major elements distinguishing it from the VM codec which prove to be the source of its improved coding efficiency. These elements are:

1. Rough segmentation of the video frame into arbitrary shaped regions composed of 4-connected 8x8 pixel blocks. The segmentation allows compact encoding of motion vector fields and can be described with relatively few bits.
2. Motion compensation scheme utilizing the above mentioned segmentation and a quadratic motion field model which enables very accurate prediction. Motion fields are compactly encoded using 2-D separable orthogonal polynomials.
3. Powerful VQ and Multi-Shape DCT based scheme utilizing spatial properties of the prediction frame for efficient coding of the residual error.

Coding Results

The performance of the proposed coder has been compared to the H.263 video coder. The Telenor implementation of H.263 (TMN 1.6c) with Advanced Prediction Mode and Unrestricted Motion Vectors was used in all the comparisons. MPEG-4 test sequences of Class A and Class B in QCIF resolution were used in the simulations.

Both schemes operated at a fixed frame rate and with a fixed value of the quantiser parameter. In all simulations the Nokia coder used the same first INTRA frame as H.263.

In the first experiment the objective quality of reconstructed sequences at approximately equal bit rates was compared. Average[2] Peak-Signal-to-Noise Ratio (PSNR) was used as the measure of quality. Simulation results are collected in Table 1.

[1] contact person: Contact person: M Karczewicz
email: marta.karczewicz@research.nokia.fi

[2] obtained by averaging PSNRs of particular frames.

Sequence	Picture size	QP Intra	NOKIA			H.263			PSNR improvement (Y/C) [dB]
			QP	Bit rate [kbps]	PSNR(Y/C) [dB]	QP	Bit rate [kbps]	PSNR (Y/C) [dB]	
Mthr&Dtr	qcif	8	15	10.14	**34.5 / 40.5**	12	10.26	**33.5 / 39.4**	**1.0 / 1.1**
Hall Objects	qcif	8	13	10.22	**33.7 / 39.1**	15	10.21	**32.4 / 38.5**	**1.3 / 0.6**
Container	qcif	8	13	9.88	**33.3 / 38.8**	15	10.00	**31.0 / 37.7**	**2.3 / 1.1**
Akiyo	qcif	4	7	9.77	**38.6 / 42.3**	9	10.28	**35.9 / 40.9**	**2.7 / 1.4**
Mthr&Dtr	qcif	4	7	20.68	**37.2 / 42.5**	8	21.10	**35.7 / 41.2**	**1.5 / 1.3**
Silent Voice	qcif	8	10	24.19	**33.5 / 37.8**	10	24.30	**32.6 / 37.2**	**0.9 / 0.6**
Container	qcif	4	7	24.07	**36.4 / 41.0**	9	23.85	**33.9 / 39.7**	**2.5 / 1.3**
News	qcif	8	12	24.91	**33.8 / 37.8**	13	24.86	**31.7 / 36.6**	**2.1 / 1.2**
Foreman	qcif	8	10	47.03	**34.3 / 38.8**	10	47.43	**32.3 / 38.0**	**2.0 / 0.8**
Coastguard	qcif	8	13	49.12	**30.4 / 41.3**	13	49.51	**29.5 / 40.6**	**0.9 / 0.7**

Table 1: Comparison of reconstruction PSNRs at equal bit rates

Results in Table 1 shows that the Nokia coder achieves 0.9 - 2.7 dB higher reconstruction PSNR than H.263.

The purpose of the second experiment was to find out the bit rate reductions achieved by the Nokia coder when compared to H.263 at approximately equal reconstruction PSNRs. As before, all the simulations were obtained using fixed values of quantiser parameters. The results of simulations are collected in Table 2.

As can be seen in Table 2 the Nokia coder enables bit rate reductions between 33 and 53 %. Experiments not shown in this document confirm similar improvements over H.263 when PB frames are used in both schemes. It was also noted that improvements over H.263 tends to be higher when the quality of the first frame improves.

Sequence	QP Intra	NOKIA			H.263			Bitrate reduction [%]	PSNR (Y/C) diff. [dB]
		QP	Bit rate [kbps]	PSNR (Y/C) [dB]	QP	Bit rate [kbps]	PSNR (Y/C) [dB]		
Akiyo	4	7	**11.11**	39.0 / 42.3	5	**23.55**	39.0 / 42.2	**53**	**0.0 / 0.1**
Container	8	9	**14.14**	34.4 / 39.3	7	**28.48**	34.5 / 39.4	**50**	**-0.1 /-0.1**
Mthr&Dtr	8	11	**12.19**	35.1 / 40.8	8	**18.23**	35.2 / 40.4	**33**	**-0.1 / 0.4**
Mthr&Dtr	4	7	**26.66**	38.1 / 42.6	5	**44.67**	38.1 / 42.7	**40**	**0.0 /-0.1**
Foreman	8	8	**55.70**	35.2 / 39.6	6	**87.64**	35.1 / 40.0	**36**	**0.1 / -0.4**

Table 2: Bit rate reductions achieved by Nokia coder.

Conclusions

The video coder presented in this proposal is shown to achieve higher coding efficiency than H.263 video coder. The objective improvements achieved by the Nokia coder range from 0.9 to 2.7 dB which translates to bit rate savings between 33 and 53%. In the light of above facts we believe that the codec has a good potential to be a basis for development of the ITU-T Long-term Videotelephony Standard H.263L.

NOKIA
RESEARCH CENTER

APPENDIX

DESCRIPTION OF NOKIA VERY LOW BIT RATE VIDEO CODER
version 3.0

1. Introduction

This document describes a scheme for compression of video at very low bit rates developed in Nokia Research Center. The primary objective of the scheme is to achieve higher coding efficiency when compared to H.263 video coder and MPEG-4 Verification Model. The video coder described in this document includes a number of novel elements which contribute to its improved performance. These elements are:

- segmentation of coded frames into regions obtained by quadtree splitting and then merging.
- very accurate motion compensated prediction utilizing 2-D orthogonal polynomials for encoding of motion vector fields.
- multiple choice coding scheme applied to motion compensated prediction error.

In order to facilitate fair comparison with H.263, the coder described in this document has been designed taking into account similar algorithmic delay constraints as those imposed on H.263 codec. Also the bitstream syntax was kept, whenever possible, similar to H.263. The major syntactic difference is the adoption of *Region Layer* in place of H.263 *Macroblock Layer*. Frame segmentation into regions is defined by segmentation information contained in the *Picture Layer* of the bitstream. The syntax of the coder allows straightforward extension to support arbitrarily-shaped frame segmentation.

Unless otherwise specified in this document the video source is assumed to in the QCIF format (176 x 144 pixels for luminance, 88 x 72 pixels for chrominance). The coder support also other source formats.

2. Coder Description

2.1 Video Source Coding Algorithm

The schematic diagram of the encoder is shown in Figure 2-1. The main novel elements of the coder are the motion field encoding scheme and the prediction error coding scheme shown in the figure as Inter Coding which utilizes motion compensated prediction frame (as shown in Figure 2-1) to improve coding efficiency.

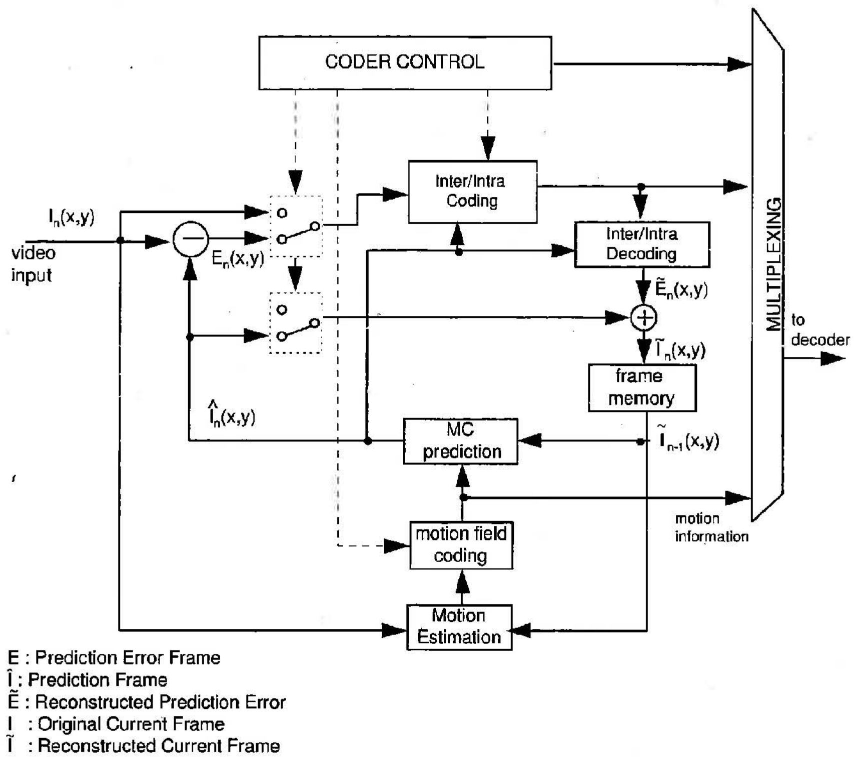


Figure 2-1 Schematic diagram of the encoder.

2.1.1 Definitions

Reference frame: The most recently reconstructed frame. Denoted $\tilde{I}_{n-1}$

Current frame: The frame that is being coded at the current moment. Denoted I_n.

Prediction frame: Motion compensated prediction of the current frame. Denoted $\hat{I}_n$.

I type frame: The frames which are coded independently of the reference frame.

P type frame: The frames coded by motion compensated prediction.

PB type frame:. A PB-frame consists of two frames being coded as one unit. The name PB comes from the name of picture types in Recommendation H.263.

Region: A semi-arbitrary shaped segment in the current frame. The term 'semi-arbitrary shaped' refers to the convention that the building blocks of the regions are *n x n* pel blocks. In the current implementation, *n=8*.

UNCHANGED type region: The regions which are found to have very little temporal change. These regions are coded by copying the corresponding area from the reference frame.

INTRA type region: The regions which are decoded independently from the reference frame.

INTER type region: Regions coded by motion compensated prediction using the reference frame. Information for such regions includes motion information and coded prediction error.

NO MOTION INTER type region: The difference between the current frame and the reference frame is coded. Information for such regions includes only coded prediction error.

INTER-B type region: (This type of region can occur only in B type frames of a PB unit). Region which is coded by bidirectional motion compensated prediction using both the reference frame and the reconstructed P-frame.

Motion vector: A pair of numbers $[\Delta x(x,y), \Delta y(x,y)]$ where $\Delta x(x,y)$ and $\Delta y(x,y)$ are the values of horizontal and vertical displacements of a pel at location *(x, y)*, respectively.

Motion vector field: A set of motion vectors of all pels in a INTER type region.

Motion model: A parametric formula describing values of motion vectors. Motion model used in Nokia coder is a second order polynomial model. (See Section 2.3.1 for details)

Motion coefficients: Coefficients needed to reconstruct motion for a region, as defined by the underlying motion model. The Nokia coder uses motion model with 12 coefficients.

2.1.2 Coding modes

The coding mode defines the region type in the current frame. Each frame type has its own set of coding modes. The coding mode decision is sent to the decoder.

If the current frame is P type, its regions can be of type UNCHANGED, INTER or INTRA. In UNCHANGED mode, no further information is sent. In INTER mode, the motion coefficients and the prediction error for the region are sent. In INTRA mode, the contents of the region are coded as such.

For PB-type frames, the region types are determined independently for the P- and B-frames. For the P-component, the possible region types are UNCHANGED, INTER-B or INTRA. In UNCHANGED mode, no further information is sent. In INTER-B mode, the motion coefficients and the prediction error for the region are sent. In INTRA mode both motion coefficients and the contents of the region are sent. For B-component, there are 8 possible region types. These types are determined by independently indicating usage of the following three coding elements:

- the differential correction to the motion parameters inherited from the region in P-component (similar to delta motion vectors in PB-frames of H.263),
- the prediction error information,
- backward prediction.

The coding of INTRA-type frames is adopted from Recommendation H.263 [3].

2.2 Frame Segmentation

Throughout this section the *current frame* is assumed to consist of luminance component only. The division of the *current frame* into segments is described in two steps: splitting (SPLIT bits) and merging (MERGE bits). The SPLIT and MERGE bitstream can describe any segmentation consisting of combination of *8 x 8* pel blocks.

The decoder starts with a fixed partitioning of the *current frame* into *32 x 32* pel segments (*initial segmentation*) shown by the solid lines in Figure 2-2[3]. The received SPLIT bits indicate which of those segments should be further divided into smaller segments. The received MERGE bits indicate how the resulting split segments should be recombined to form the final frame segmentation

The encoder starts the segmentation from the *32 x 32* pel regions. Motion of these regions is estimated and each region not satisfying a normalized prediction error criterion is split into smaller blocks. Splitting and motion estimation for regions proceeds recursively until the measure of prediction error for the region falls below a given threshold, or until the minimum size of the region (*8x8* pixels) is reached. The next step in the encoder is the merging during which some of the neighboring blocks are combined together using the rate-distortion criterion. This two-stage process can result in an segmentation consisting of regions which are combinations of 4-connected *8 x 8* pel blocks (Figure 2-5).

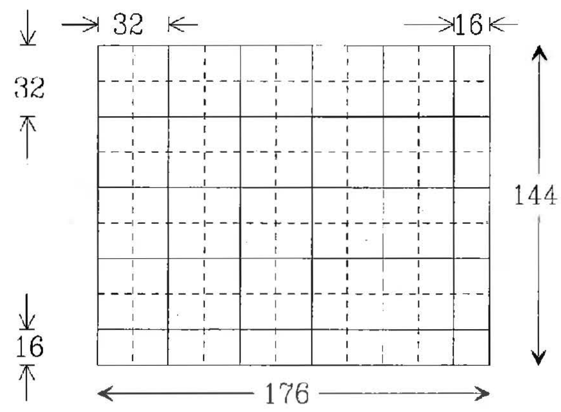


Figure 2-2. Initial segmentation of the coded frame.

[3] QCIF resolution frames cannot be fully divided into *32 x 32* blocks which is why the initial partitioning of the QCIF frame includes *16 x 32, 32 x 16* and *16 x 16* blocks on the frame boundaries.

2.2.1 Splitting

Split information is sent from the encoder to the decoder as two sequences of bits (SPLIT bits). Bits of the first sequence indicate splitting of all the regions in the initial segmentation which are greater than *16x16* pel (i.e. *32x32, 32x16,* and *16x32*) into four or two *16x16* regions as shown by the dashed lines in Figure 2-2. We refer to this operation as *first split level.* An example of the regions resulting from the first split is shown in Figure 2-3.

The bits of the second sequence indicate which of the *16x16* regions present in the frame segmentation after performing the first split level should be split into four *8x8* regions. We refer to this operation as *second split level.* An example of the allowed partitioning for the second split is shown in Figure 2-3. Figure 2-4 shows an example frame segmentation after two levels of split.

Each of the resulting two SPLIT bit sequences is coded using an entropy coded run-length technique described in Section 4.1.3.

2.2.2 Merging

The next step in building the segmentation is merging of regions produced by the above splitting procedure. In this step, the encoder merges those neighboring regions which satisfy a prediction error criterion for the hypothetical region that would result from merging those two neighboring regions. We refer to this merging algorithm as *Motion Assisted Merging*. The algorithm is described in [2].

The encoder informs the decoder about the merge/not-merge decisions using MERGE bits which are inserted in the bitstream right after the SPLIT bits. MERGE bits refer to an *adjacency graph,* which is initialized at the beginning of the transmission and is being updated in the meantime according to the following rules:

1 Initialize *adjacency graph*:

- 1.1 Assign an unique label to each of the split segments: scan the segmentation image from left to right - top to bottom with a step of 8x8 pel block, every time a new segment is encountered assign a new label to it, which is incremented by one comparing to the previous label.
- 1.2 Associate with each segment array of labels of neighboring segments using 4-connectivity rule[4]. These arrays are initially sorted according to increasing index of segment labels.

2 Construct MERGE bit sequence by parsing the adjacency diagram from the segment with lowest index to the segment with highest index, and parsing the array of neighbors from start to end. Generate bits for merge/not merge decisions only for those neighbors having a higher segment index than the index of the segment being processed. 0 indicates that the two segments remain intact, 1 indicates that the two segments are merged.

3 Every time two segments are merged update the *adjacency graph* before proceeding with encoding/decoding subsequent MERGE bits.

- 3.1 The whole area of the merged segment should be labeled with the lower label. The higher label should be removed from the *adjacency graph* completely.
- 3.2 Adjacency relations must be updated. Specifically, if the segment labeled i is merged with the segment labeled j, where $i<j$, do the following for each of the indices k in the array of neighbors of segment j, parsing the array from start to end:

  If k is a common neighbor of i and j,

  proceed to the next k

  else

  concatenate the index k to the end of neighbor array of segment i.

Figure 2-5 shows a possible segment layout after merging.

[4] In this context, 4-connectivity rule means that two segments are neighbors if they have at least a line segment as their common border

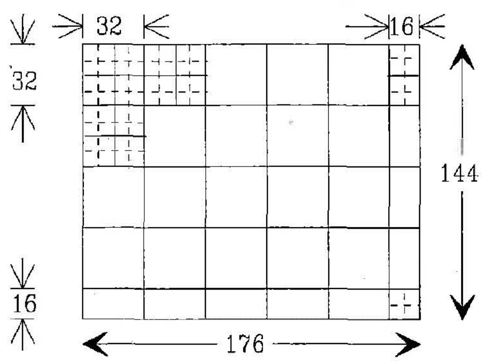


Figure 2-3. Example region boundaries after the first level split. Dashed lines indicate candidates for the second level split.

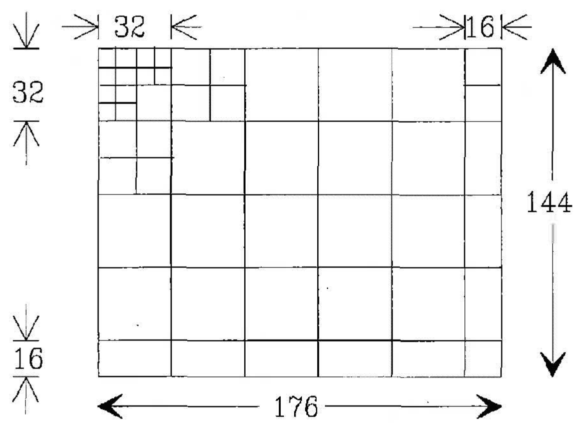


Figure 2-4. Example region boundaries after the second level split

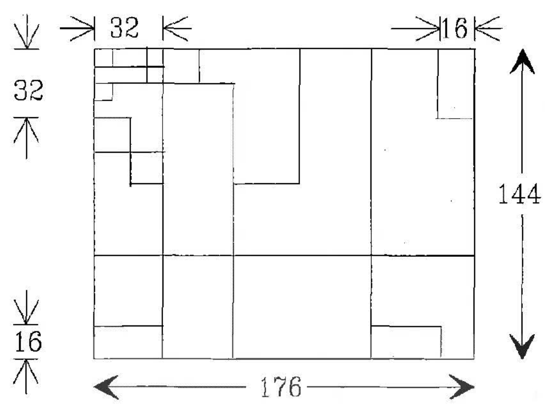


Figure 2-5. Example regions after merging

2.3 Motion Field Coding

In this document only decoder specific features of motion vector field coding are described. Detailed description of motion estimation and encoding (*Motion Assisted Merging* and *Coefficient Removal*) can be found in [1,2].

Motion compensated prediction of a region resulting from the segmentation process described above is performed according to the following equation:

$$\hat{I}_n(x,y) = \tilde{I}_{n-1}[x+\Delta x(x,y), y+\Delta y(x,y)] \qquad (2\text{-}1)$$

where $\hat{I}_n$ is the prediction frame and $\tilde{I}_{n-1}$ is the previous reconstructed frame (reference frame) and the pair of numbers $[\Delta x(x,y), \Delta y(x,y)]$ is a *motion vector* of the pel in location *(x, y)*.

The chrominance motion vectors are calculated by evaluating the luminance motion vector field in the half-pel position corresponding to the location of this chrominance pel and by dividing the resulting motion vector by two.

2.3.1 Motion model

The motion field of each INTER region is represented by a set of 12 *motion coefficients*. These coefficients are produced in the encoder by the motion estimation and motion field encoding blocks in Figure 2-1. The relation between motion coefficients and the actual values of motion vectors in a given point of a region is defined by the following parametric model:

$$\begin{aligned}\Delta x(x,y) &= \tilde{c}_1 f_1(x,y) + \tilde{c}_2 f_2(x,y) + \tilde{c}_3 f_3(x,y) + \tilde{c}_4 f_4(x,y) + \tilde{c}_5 f_5(x,y) + \tilde{c}_6 f_6(x,y) \\ \Delta y(x,y) &= \tilde{c}_7 f_1(x,y) + \tilde{c}_8 f_2(x,y) + \tilde{c}_9 f_3(x,y) + \tilde{c}_{10} f_4(x,y) + \tilde{c}_{11} f_5(x,y) + \tilde{c}_{12} f_6(x,y)\end{aligned} \qquad (2\text{-}2)$$

where functions $f_j(\cdot)$ (j=1,2, ...,6) are called *motion field basis functions* and *(x, y)* are integer pixel coordinates in a system with the origin in the left-upper corner of the frame.

The motion model is based on 6 basis functions and the same model is used for horizontal and vertical displacements. The basis functions are obtained by orthonormalizing the basis function set *{1, x, y, xy, x^2, y^2}* to the bounding box of the region (example shown in Figure 2-6).

Hence the form of the motion field basis functions is uniquely determined by the size of the bounding box of the region and can be constructed in the decoder after the SPLIT and MERGE bits are received. The two dimensional (2-D) basis functions $f_j(\cdot)$ are built as a tensor product of two sequences of one dimensional (1-D) discrete orthonormal polynomials:

$$g_r(x) = \sum_{j=0}^{r} \alpha_{r,j} x^j, \quad r = 0,1,2 \quad \text{orthonormal on the interval } [x_{\min}, x_{\max}]$$

$$h_r(y) = \sum_{j=0}^{r} \beta_{r,j} y^j, \quad r = 0,1,2 \quad \text{orthonormal on the interval } [y_{\min}, y_{\max}] \qquad (2\text{-}3)$$

Functions $f_j(\cdot)$ are built as follows:

$$\begin{aligned}&f_1(x,y) = g_0(x)h_0(y) \quad f_2(x,y) = g_0(x)h_1(y) \quad f_3(x,y) = g_1(x)h_0(y) \\ &f_4(x,y) = g_1(x)h_1(y) \quad f_5(x,y) = g_0(x)h_2(y) \quad f_6(x,y) = g_2(x)h_0(y)\end{aligned} \qquad (2\text{-}4)$$

The coefficients of the polynomial $g_r(x)$, with $L = x_{\max} - x_{\min}$ are given by

$$\begin{aligned}
\alpha_{0,0} &= \sqrt{\frac{1}{L+1}} \\
\alpha_{1,0} &= \sqrt{\frac{3L}{(L+1)(L+2)}} + 2x_{\min}\sqrt{\frac{3}{L(L+1)(L+2)}} \\
\alpha_{1,1} &= -2\sqrt{\frac{3}{L(L+1)(L+2)}} \\
\alpha_{2,0} &= \sqrt{\frac{5(L-1)L}{(L+1)(L+2)(L+3)}} + 6x_{\min}\sqrt{\frac{5L}{(L-1)(L+1)(L+2)(L+3)}} + 6x_{\min}^2\sqrt{\frac{5}{(L-1)L(L+1)(L+2)(L+3)}} \\
\alpha_{2,1} &= -6\sqrt{\frac{5L}{(L-1)(L+1)(L+2)(L+3)}} - 12x_{\min}\sqrt{\frac{5}{(L-1)L(L+1)(L+2)(L+3)}} \\
\alpha_{2,2} &= 6\sqrt{\frac{5}{(L-1)L(L+1)(L+2)(L+3)}}
\end{aligned} \qquad (2\text{-}5)$$

The respective coefficients $\beta_{r,j}$ are calculated as coefficients $\alpha_{r,j}$ using the above formulas by replacing $x_{\min}$ by $y_{\min}$ and setting $L = y_{\max} - y_{\min}$.

The choice of orthogonal polynomials for basis functions was motivated by the observation that coefficients corresponding to such polynomials are less sensitive to quantization than coefficients corresponding to basis functions $\{1, x, y, xy, x^2, y^2\}$. Two options were considered initially:

- polynomials orthonormalized with respect to shape of the region (using some orthogonalization method) or
- separable polynomials orthonormalized with respect to the bounding box of the region calculated according. to formulas (2-4).

It was found that the latter motion model provides equally good performance[5] as the former one. The separable model was chosen since its basis functions are given by analytic formulas and can be computed with relatively low computational complexity.

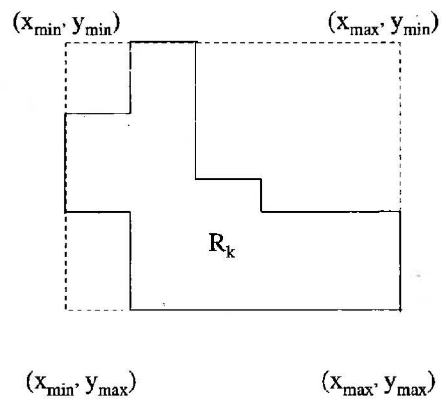


Figure 2-6. Bounding box for the region R_k

2.3.2 Motion coefficient scaling and quantization

2.3.2.1 Motion coefficient scaling

Scaling of motion coefficients enables to vary the bit allocation between segments having different sizes, but the same size of the bounding box. Scaling operation itself does not affect neither the bit allocation nor the precision of motion estimation. If there was no quantization of motion coefficients, scaling would not affect the codec performance at all. It is the quantization of scaled motion coefficients, which dedicates more bits (assuming that the number of motion coefficients are equal) to larger segments among those having the same size of the bounding box.

Motion coefficients of a segment are scaled according to the ratio between the size of the bounding box and the size of the segment. Let us denote the number of pixels in the segment by P, and the number of pixels in the bounding box by P_{box}. The scaling factor *scale* equals to:

$$scale = \frac{P_{box}}{P} \tag{2-6}$$

Each of the motion coefficients of the segment is divided by the scaling factor *scale*. In order to keep the validity of (2-2) it is necessary that the value of each of the basis functions (2-4) at each pixel location in the segment is multiplied by the same scaling factor *scale*.

2.3.2.2 Motion coefficient quantization

A uniform scalar quantizer is applied to each of the non-zero motion coefficients. The quantizer step size is predefined as STEP=3. The quantization of a coefficient $\tilde{c}_j$ corresponding to orthonormal basis functions is done according to the formula:

[5] measured in bits needed to achieve a given prediction error.

$$\text{LEVEL} = \tilde{c}_j \ // \ \text{STEP}, \tag{2-7}$$

where // denotes division followed by rounding operation.

Each of the non-zero motion coefficients $\tilde{c}_j$ is reconstructed from the transmitted LEVEL. The following formula describes the inverse quantization process of a given coefficient resulting in a reconstructed coefficient value $\hat{c}_j$:

$$\hat{c}_j = \text{sign(LEVEL)} * |\text{LEVEL}| * \text{STEP} \tag{2-8}$$

The maximum encodable value of |LEVEL| is 1130. Whenever |LEVEL| exceeds the maximum encodable value it is clipped.

Encoder transmits scaled motion coefficients, so before proceeding to build the *prediction frame* in the decoder it is necessary to scale the basis functions as described in the previous section.

2.3.3 Motion coefficient coding

The motion field of an INTER type region is described by 12 motion coefficients, 6 of which relate to horizontal and 6 to vertical components of motion vectors (2-2). Since the amount and type of motion in the sequences tends to vary, the method for coding motion coefficients was designed to enable adaptation of the complexity of the motion model to the motion in the sequence.

The motion information for a region consists of two elements: *selection information* and quantized coefficients. Selection information contains two 6-tuples of bits: MCP_x and MCP_y. Each bit of the 6-tuple MCP_x corresponds to one coefficient of the horizontal displacement function $\Delta x(x, y)$ whereas bits of MCP_y correspond to coefficients of the vertical displacement function $\Delta y(x, y)$ of the region. The role of these bits is to indicate whether the corresponding coefficient has a nonzero value. Thus selection information identifies which motion coefficients are transmitted to the decoder.

Note that this structure of information allows varying the complexity of the motion model between regions, frames, and sequences. This enables the usage in the encoder of *Coefficient Removal Algorithm* to determine the importance of a given coefficient for the result of the prediction and to determine which coefficients need to be transmitted for a particular region. The algorithm is described in [2]. However, other methods for estimating and selecting the motion coefficients can be used in the encoder.

The 6-tuples in the selection information as well as the values of the quantized non-zero coefficients are Huffman coded as described in Section 4.2.4.

The coder uses a PB-frame mode adapted to operate on arbitrary shaped regions and to utilize a polynomial motion model. If the PB-frame mode is used, then, as in H.263, additional differential motion coefficients can be sent for a region to improve prediction of the region in the B-frame. The implementation of PB-frame mode is described in detail in Section 3.

2.3.4 Image interpolation

Since values of motion vectors can have non-integer values, motion compensated prediction requires evaluating the luminance and chrominance values at non-integer locations *(x, y)* in the reference frame $\tilde{I}_{n-1}$. The interpolation of luminance and chrominance values is done by cubic convolution interpolation using pixel luminance values in the *4x4* neighborhood [4].

For a frame of size *M x N*, the pixels have coordinates $x_j = 0,1,\ldots,M-1$ and $y_k = 0,1,\ldots,N-1$. Let (x_j, y_k) be such that $x_j \le x < x_{j+1}$ and $y_k \le y \le y_{k+1}$. The cubic convolution interpolation in the point *(x, y)* is defined as:

$$\tilde{I}_{n-1}(x,y) = \sum_{l=-1}^{2} \sum_{m=-1}^{2} c_{j+l,k+m} u\left(x - x_{j+l}\right) u\left(y - y_{k+m}\right) \tag{2-9}$$

where 1-D interpolation function is defined as:

$$u(s)=\begin{cases}\frac{3}{2}|s|^3-\frac{5}{2}|s|^2+1 & 0<|s|<1\\ -\frac{1}{2}|s|^3+\frac{5}{2}|s|^2-4|s|+2 & 1<|s|<2\\ 0 & 2<|s|\end{cases} \qquad (2\text{-}10)$$

For pixels within the frame boundaries, the c_{jk}'s are given by $c_{jk}=\tilde{I}_{n-1}(x_j,y_k)$. Luminance values c_{jk} residing outside the frame which are needed for interpolation are obtained by duplicating the luminance values of pixels on the boundary of the frame.

2.4 Multi-Transform Prediction Error Coding

2.4.1 Overview

The Multi-Transform Prediction Error Coding technique used in this coder is based on the observation that in typical video sequences the prediction error (residual error after motion compensation) is concentrated near the contours of the moving objects. Knowledge of the localization of prediction error can be used to improve coding efficiency by using transforms with better localization properties.

The exact location of contours of moving objects is not known in general. However, the locations can be approximately determined by finding edges and other discontinuities in a video frame. For this purpose, the coder utilizes the prediction frame which is known both to the encoder and the decoder (after receiving motion coefficients). The improved coding efficiency of the system is due to the fact that properties (location, directionality, etc.) of the prediction error signal at a given location can be inferred from properties of the prediction frame $\hat{I}_n$ in the same location. Thus the decoder can anticipate what coding technique(s) the encoder will choose for coding of prediction error pattern in this block.

In the proposed coder both the encoder and the decoder include a classifier which analyses spatial properties (location of discontinuities and their directionality) of the prediction frame $\hat{I}_n$. The above information is used to switch between a multitude of coding methods. The decision on the best methods is made by an optimization procedure based on rate-distortion performance. The selection information is transmitted to the decoder as a variable length codeword. Among the possible methods there are Multi-Shape DCTs, extrapolation and Entropy-Constrained VectorQuantization (ECVQ).

2.4.2 8x8 Classification

The criterion for classification of an *8x8* block, using the prediction frame $\hat{I}_n$, is the location of the areas with high variance of pixel values. Each *8x8* block is divided into *4 4x4* quadrants. For each quadrant *B* its variance is calculated:

$$\mathrm{var}_B=\sum_{(i,j)\in B}\left[\hat{I}_n(i,j)-\mu\right]^2 \qquad (2\text{-}11)$$

where μ is the average of the pixel values in quadrant *B*. A quadrant is said to be active when its variance is larger than *256*. According to the number and location of active quadrants *8x8* block is classified into one of *6* classes shown in Figure 2-9. The prediction and prediction error blocks are rotated after active quadrants are established. The rotation can be 90°, 180°, 270°. After the rotation the active quadrant(s) of the corresponding *8x8* prediction block should coincide with those shown in Figure 2-7. All rotations are counter-clockwise.

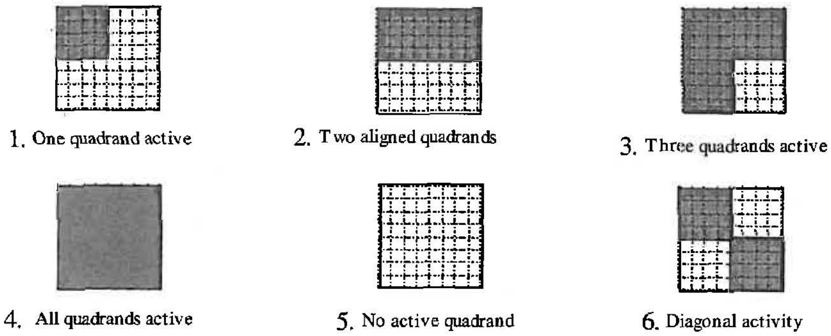


Figure 2-7: The 8x8 classes

2.4.3 4x4 Classification

4x4 blocks are classified using as a criterion their variance and spatial directionality in the prediction frame and operational rate-distortion point.

2.4.3.1 Variance Classification

The variance of a *4x4* block *B* (denoted as var_B) is calculated using Equation (2-11). According to the value of the variance *4x4* block is classified as:

- *Low Variance Block* if $var_B < 1300$,
- *High Variance Block* if $var_B \geq 1300$.

2.4.3.2 Rate Classification

As an indication of the operational rate-distortion point the quantizer value QUANT is used. Information about the quantizer value is sent to the decoder for each region.

To reflect the operational rate-distortion point, each *4x4* block is classified according to the quantizer value QUANT used for a region which contains *4x4* block under consideration. Blocks with $QUANT \leq 9$ are classified as *High Rate Blocks*, blocks with $QUANT > 13$ are classified as *Low Rate Blocks*, and blocks with $9 < QUANT \leq 13$ are classified as *Medium Rate Blocks*.

2.4.3.3 4x4 Directionality Classifier

Given a *4x4* prediction block the edge content in four directions is calculated:

$$H = \frac{1}{3x4}\sum_{i=1}^{3}\sum_{j=1}^{4}\left|\hat{I}_n(i+1,j) - \hat{I}_n(i,j)\right|, \qquad V = \frac{1}{3x4}\sum_{i=1}^{4}\sum_{j=1}^{3}\left|\hat{I}_n(i,j+1) - \hat{I}_n(i,j)\right|$$

$$D_2 = \frac{1}{3x3}\sum_{i=1}^{3}\sum_{j=1}^{3}\left|\hat{I}_n(i+1,j+1) - \hat{I}_n(i,j)\right| \qquad D_1 = \frac{1}{3x3}\sum_{i=1}^{3}\sum_{j=1}^{3}\left|\hat{I}_n(i+1,j) - \hat{I}_n(i,j+1)\right|$$

The coordinates *i* and *j* of the pixels are given with respect to the upper left corner of the *4x4* block under consideration.

The smallest value m_1, and the second smallest value m_2 in the set $\left(\frac{H}{V}, \frac{V}{H}, \frac{D_1}{D_2}, \frac{D_2}{D_1}\right)$ are found. Next the *4x4* block is assigned to one of the following classes:

- *Horizontal*
- *Vertical,*
- *Diagonal-1,*
- *Diagonal-2,*
- *Texture,*
- *Flat.*

The classification is presented in the form of pseudo-code:

```
// Selection of Horizontal and Vertical Classes.
if( (m1 < 0.65) && (m2 ≥1.5*m1) )
{
        if(m1 == H/V ) CLASS=VERTICAL; // Selecting VERTICAL Class.
        if(m1 == V/H ) CLASS = HORIZONTAL; // Selecting HORIZONTAL Class.
}

// Selecting Diagonal Classes.
if(m1 < 0.65)
{
        if(m1 == D1/D2 ) CLASS = DIAGONAL-2; // Selecting class DIAGONAL-2.
        if(m1 == D2/D1 ) CLASS = DIAGONAL-1; // Selecting class DIAGONAL-1.
        if( (m1 == H/V ) || (m1 == V/H ) )
                if( (m2 < 0.75) || (m2 < 1.5m1)
                {
                        if(m2 == D1/D2 ) CLASS = DIAGONAL-2; // Selecting class DIAGONAL-2.
                        if(m2 == D2/D1 ) CLASS = DIAGONAL-1; // Selecting class DIAGONAL-1.
                }
}

// Selecting Texture and Flat Classes.
if(m1 ≥0.65)
{
        if(max(H,V,D1,D2) < 0.9)
                CLASS = FLAT; // Selecting FLAT class.
        else
                CLASS = TEXTURE; // Selecting TEXTURE class.
}

if(0.5 ≤m1 < 0.65)
{
        if( (m2 ≥0.75) || (m2 ≥ 1.5m1) )
                if( (m1 == H/V ) || (m1 == V/H )
                        if(max(H,V,D1,D2) < 0,9)
                                CLASS = FLAT, // Selecting FLAT class.
                        else
                                CLASS = TEXTURE; // Selecting TEXTURE class.
}
```

In the next step, the prediction error quadrants which are assigned to the *Vertical* class are rotated to be classified as *Horizontal* and *Diagonal-2* prediction error quadrants are rotated to be classified as *Diagonal-1* (all rotations are counter-clockwise). Therefore using directionality of the prediction error block as a criterion the *4x4* block can be classified as

- *Horizontal (Hor)*
- *Digonal-1 (Di1),*
- *Texture (Tex),*
- *Flat (Flat).*

Using as criterion the variance, spatial directionality and operational rate-distortion point as class features, there are 18 classes to which a *4x4* block can be assigned.

2.4.4 Coding Methods

2.4.4.1 8x8 Coding Methods

The coder includes *26* methods. Each of these methods is applied to pixels marked gray in Figure 2-8. Throughout the rest of this document we will refer to these methods with the enumeration shown in Figure 2-8.

For each method the coded area is a cluster or combination of clusters of the same size. For example in method 7-5 the coded area is a combination of *2 4x4* clusters. There are only *5* different cluster sizes: *8x8, 4x8, 8x4, 3x8 and 4x4.* Clusters of size *8x8, 4x8, 8x4* and *3x8* are coded using 2-D separable Discrete Cosine Transforms (DCT) of the corresponding size, i.e., *8x8, 4x8, 8x4* and *3x8* DCT, respectively. *4x4* clusters are coded using one *of 4x4 Coding Methods* which will be described in the next section. Among *4x4* coding methods there is *4x4* DCT. The assignment of zigzag scanning methods and VLC tables for coefficients to DCTs used in the coder are shown in Table 3. Note also that sizes *4x8* and *8x4* can use the same VLC table. The quantization and dequantization of coefficients in all the transforms are done in the same way as for INTER DCT coefficients in Recommendation H.263.

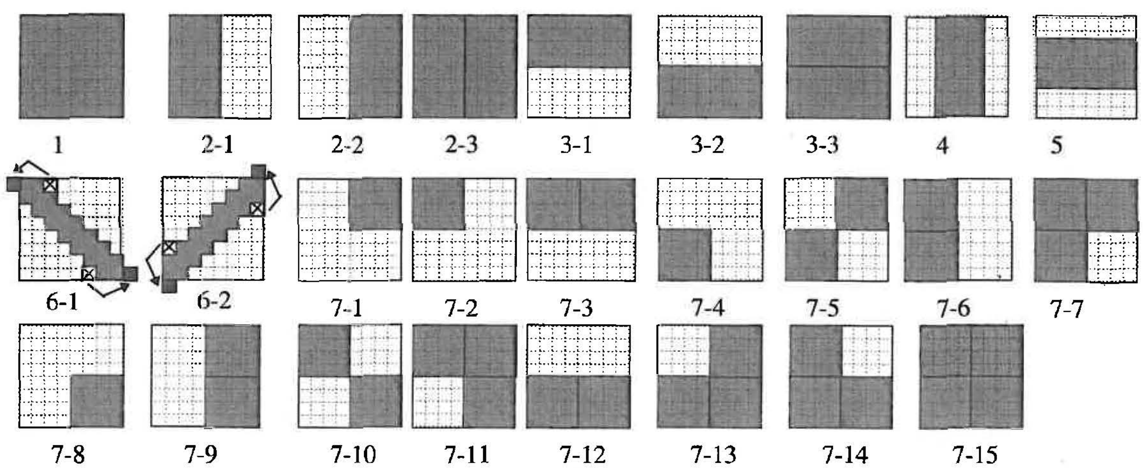


Figure 2-8: Location of 26 coding methods in an *8x8* block. Pixels belonging to a given method are marked with gray.

TABLE 3

Summary of DCT techniques used for coding of prediction error.

Transform no:	Size of transform	Coefficient scanning order	VLC table
1	*8 x 8*	as in Recommendation H.263	as in Recommendation H.263
6	*3 x 8*	TABLE 26	TABLE 21
2-#, 3-#,4,5	*4 x 8, 8 x 4*	TABLE 24	TABLE 19
7-#	*4 x 4*	TABLE 25	TABLE 20

2.4.4.2 4x4 Coding Methods

There are following *4x4 Coding Methods:*

1. *4x4 DCT.*
2. *4x4 Low Energy Horizontal ECVQ (LHor)*
3. *4x4 High Energy Horizontal ECVQ (HHor).*
4. *4x4 Low Energy Vertical ECVQ (LVer) - 4x4 Low Energy Horizontal ECVQ* is applied to the block after rotating the block 90° counter clockwise.
5. *4x4 High Energy Vertical ECVQ (HVer) - 4x4 High Energy Horizontal ECVQ* is applied to the block after rotating the block 90° counter clockwise
6. *4x4 Low Energy Diagonal-1 ECVQ (LDi1).*
7. *4x4 High Energy Diagonal-1 ECVQ (HDi1).*
8. *4x4 Low Energy Texture ECVQ (LTex).*
9. *4x4 High Energy Texture ECVQ (HTex).*
10. *4x4 Flat ECVQ (Flat).*
11. *4x4 Low Energy Diagonal-2 ECVQ (LDi2) - 4x4 Low Energy Diagonal-1 ECVQ* is applied to the block after rotating the block 90° counter clockwise.
12. *Extrapolation* of the quadrant (*Ext*). The median value of 9 neighboring pixels of the quadrant in the predicted 8x8 block is calculated. The reconstruction value of all the pixels of the quadrant is set to this value.

To train low energy ECVQ codebooks prediction error blocks with energy between *500* and *4000* are used. To train high energy ECVQ codebooks prediction error blocks having energy above *2000* are used. The energy of a *4x4* block is defined as the sum of squared pixel values in the block.

For training of ECVQ codebook of certain directionality only *4x4* prediction error blocks having this directionality are used. To determine directionality of prediction error block *4x4 Directionality Classifier* described in Section 2.4.3.3 is applied to this block. For example to obtain horizontal codebook we used:

- *4x4* prediction error blocks classified as horizontal,
- *4x4* prediction error blocks classified as vertical rotated 90° counter clockwise.

2.4.5 Method Selection

There is a set of *8x8 Coding Methods* associated with each *8x8* class. The assignment of methods to classes of blocks is given in Table 17. To each of the methods selected for a given class there is assigned *8x8* method selection number MTYPE8. If a 7-# method is to be used for a particular *8x8* block, then each coded *4x4* block can use one of the *4x4 Coding Methods* available for *4x4* class to which this block is assigned. The class for *4x4* block is determined using the corresponding *4x4* prediction block and the value of QUANT. Given the class of the *4x4* block Table 18 shows the available *4x4 Coding Methods* and the corresponding *4x4* method selection numbers MTYPE4.

In encoder all the methods in the set assigned to a class of a given *8x8* block are tested. The lagrangian value is used as a selection criterion

lagrangian = square_error + lambda*number_of_bits.

The method selection algorithm for an *8x8* block consists of the following steps:

1 Determine the class of the 8x8 block using the corresponding prediction block.

2 Determine those *4x4* quadrants that can be coded using 7-# methods assigned to that class.

  2.1 For each of the *4x4* quadrants that can be coded

    2.1.1 Determine the class of the *4x4* block using the corresponding prediction block.

    2.1.2 Using each possible method for that *4x4* class, encode the *4x4* block and calculate its lagrangian.

    2.1.3 Choose the 4x4 method for each quadrant with the smallest lagrangian.

  2.2 Calculate the lagrangian of the tested *7-#* methods. The lagrangian is calculated by adding the lagrangians of the selected *4x4* methods for the quadrants which are coded for the considered *7-#* method.

3 For each of the rest of the available methods of the class.

  3.1 Calculate the lagrangian of the tested method.

4 Compare the lagrangians of all the available methods of the class.

5 Choose the method with the smallest lagrangian.

The 8x8 method selection number MTYPE8 corresponding to the chosen transform is transmitted to the decoder as a variable length codeword. If one of *7-#* method was selected the 4x4 method selection numbers MTYPE4 for all coded *4x4* blocks are sent as variable length codewords.

2.4.6 Decoding process.

In order to decode the coded prediction error in a given block, the decoder performs classification of the corresponding *8x8* block in the prediction frame $\hat{I}_n$ as described in 2.4.2. Knowing the class of the block the decoder can interpret the MTYPE8 field as defined in Table 17. If MTYPE8 indicates a combination of *4x4* methods the decoder will use the *4x4* classification as described in 2.4.3 and hence it will be able to interpret MTYPE4 (as defined in Table 18), which would be the next field in the bitstream.

2.5 Chrominance coding

Chrominance is coded using 2-D DCT. The mode and the size of a chrominance block depends on the mode of the luminance region in the corresponding location. The chrominance frame is divided into *8 x 8* blocks. The mode of the corresponding *16x16* Y block is used for the *8x8* chrominance block and the coding of such chrominance blocks is the same as in Rec. H.263 [3]. When the corresponding *16x16* Y block spans regions with different modes (so called JUNCTION block), each *4x4* quadrant of such a chrominance block is handled separately. This situation is explained in Figure 2-9. A *4x4* UNCHANGED chrominance block is copied, and a *4x4* INTER or NO MOTION INTER chrominance block is coded using *4x4* DCT. For *4x4* INTRA chrominance blocks in a *8x8* chrominance block (maximum 3), a joint (average) mean value (DC value) is calculated. The quantization of this joint DC value depends on the number of *4x4* INTRA chrominance blocks in a *8x8* chrominance block:

Number of 4x4 blocks	Quantization	Number of bits	Dequantization
1	$DC_Q = \left(\sum_{i=1}^{16} p_i\right) / 64$	6	$DC_{DQ} = 4DC_Q$
2	$DC_Q = \left(\sum_{i=1}^{32} p_i\right) / 64$	7	$DC_{DQ} = 4DC_Q$
3	$DC_Q = \left(\sum_{i=1}^{48} p_i\right) / 48$	8	$DC_{DQ} = DC_Q$

The content of the *4x4* blocks after removing the mean is coded using *4x4* DCT. The VLC and scanning order are the same for both *4x4* INTRA and INTER blocks and are given in Table 27 and in Table 25, respectively.

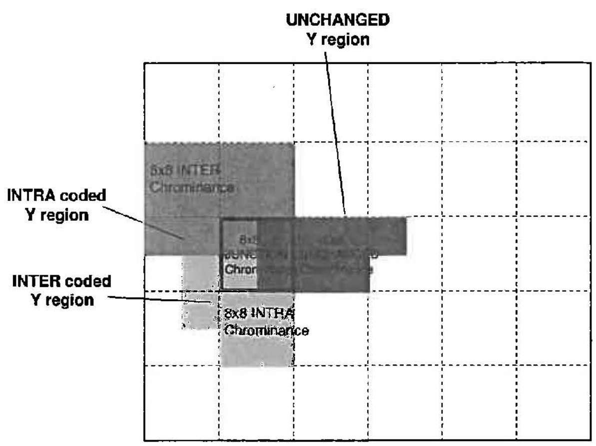


Figure 2-9 Location of 4 INTRA coded chrominance blocks with respect to location of INTRA coded region in luminance (gray). Positions of chrominance blocks are marked with dashed lines.

3. *PB-frame Mode*

3.1 Introduction

This section describes the PB-frame mode in Nokia coder. A PB-frame consists of two pictures being coded as one entity. The name PB-frame comes from the name of picture types in Recommendation H.263 to indicate that PB modes used in the presented coder share a number of common features with PB-frames of H.263. Thus a PB-frame consists of one P-picture which is predicted from the previous decoded P-picture and one B-picture which is predicted both from the previous decoded P-picture and the P-picture currently being decoded. The prediction process is illustrated in Figure 3-1

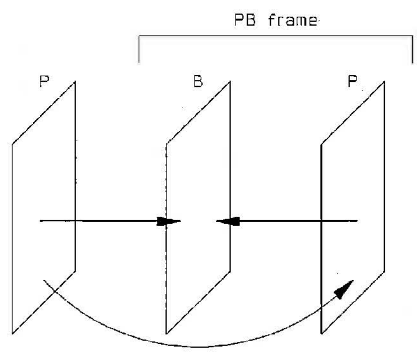
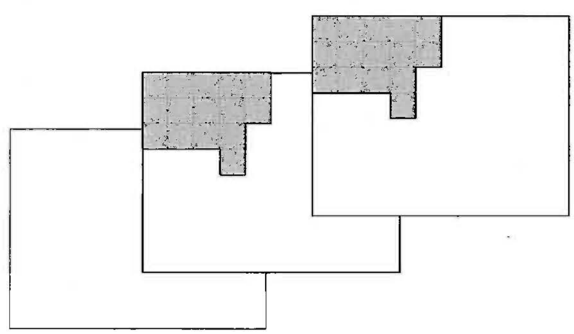


Figure 3-1 Prediction in PB-frame mode

Throughout this section we use the same convention as in Recommendation H.263 regarding the distinction between forward and backward motion compensated prediction. Hence forward prediction refers to situation when reference frame is earlier in time than the frame being coded, whereas backward prediction refers to situation when reference frame is later in time than the frame being coded.

3.2 Region Layer

B-frame is segmented into regions in identical manner as the P-frame using SPLIT and MERGE bits for this PB-frame. Hence a region of PB-frame is formed by taking *8x8* blocks of P-frame and the blocks of B-frame in the corresponding locations as shown in Figure 3-2.



Previous P B frame P frame

Figure 3-2 Location of *8x8* blocks in a PB-region (shown shaded).

First the data for the region of P-frame is transmitted, then the data for the corresponding region of B-frame. The possible modes for B-region are signaled by a VLC word determining values of the following three flags:

1. BMCC indicating that MCC prediction error coding is present for B-region,
2. BMVF indicating the use of backward motion vector field for prediction of the B-region (unlike in PB-frame mode of H.263 where backward prediction is used unconditionally),

3. DMVF indicating that differential motion vector field coefficients are present for the refinement of the forward motion vector field for B-frames.

3.3 PB-frames and INTRA regions

For an INTRA coded region in PB-frame, motion vector field coefficients are transmitted but used only to predict the region in B-frame. Coding mode INTRA of a region has the following meaning in the context of PB-frame:

- The P-region is INTRA coded .
- The corresponding B-region is INTER-B coded.
- The corresponding B-region is predicted using either bidirectional prediction (if BMVF flag is 1) or only forward prediction (if BMVF flag is 0).

3.4 Calculation of motion vector field for the B-picture

The vectors for the B-picture are calculated as follows. Assume we have a motion vector field *MVF* to be used in the P-picture (MVF represents a vector field for one region in the P-frame). For prediction of the B-picture we need either both forward and backward motion vector fields MVF_F and MVF_B or only forward motion vector field MVF_F. The forward and backward motion vector components are derived from *MVF*. The forward motion vector field MVF_F can be enhanced by a differential motion vector field given by MVF_D (if DMVF flag of the region mode is set to 1).

Values of motion vectors *MVF* are calculated using values of decoded motion coefficients $\tilde{c}_j$ (see also Section 2.3.1) using equation:

$$MVF(x,y) = \sum_{j=1}^{12} \tilde{c}_j \cdot f_j(x,y) \qquad (3\text{-}1)$$

Then MVF_F and MVF_D components are given by the following formulas:

$$MVF_D(x,y) = \sum_{j=1}^{12} \tilde{b}_j \cdot f_j(x,y) \qquad MVF_F(x,y) = \sum_{j=1}^{12} (\tilde{c}_j / 2) \cdot f_j(x,y) + MVF_D(x,y) \qquad (3\text{-}2)$$

Whenever the DMVF flag is set to 0 then no data for $\tilde{b}_j$ coefficients is transmitted and MVF_D is 0. When DMVF flag is set to 1 then data for the coefficients $\tilde{b}_j$ is transmitted in the same manner as the $\tilde{c}_j$ coefficients are transmitted for P-frames with the exception that the VLC tables are different (Section 4.2.3).

Bidirectional prediction inside a region in B-frame involves first identification of positions of pixels which are to be predicted bidirectionally, and second finding backward motion vectors MVF_B for such pixels. A bidirectionally predicted pixel is found by taking a pixel *(p, q)* in P region, and calculating its motion vector *MVF(p, q)*. The noninteger coordinates *(x',y')* in B-frame indicated by the motion vector *0.5 * MVF(p, q)* are quantized to closest integer-valued pixel position *(x, y)*. Then the pixel in position *(x, y)* is bidirectionally predicted with a backward motion vector:

$$MVF_B(x,y) = -0.5 * MVF(p,q) \qquad (3\text{-}3)$$

and a forward motion vector $MVF_F(x, y)$ as in the above equations.

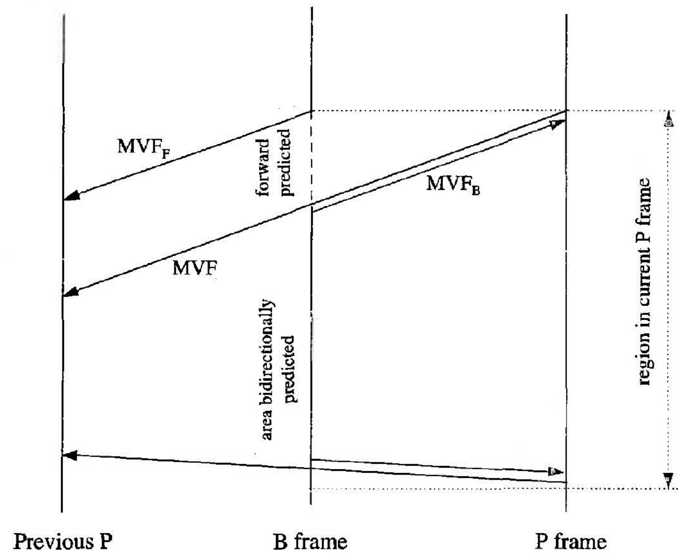


Figure 3-3 Prediction in PB-frame mode

If in the above procedure a pixel *(x, y)* in the region in B-frame is backward predicted twice (as the result of rounding of coordinates *(x',y')*) always the first prediction is used.

3.5 Prediction of a B-region in PB-frame

The following procedure applies for luminance as well as chrominance blocks. First, the forward and backward vectors are calculated. It is assumed that the P-region (luminance and chrominance) is first decoded and reconstructed. This region is called P_{REC}. Based on P_{REC} and the previous P-frame, the prediction for the B-region is calculated.

The prediction of the B-region has two modes that are used for different parts of the region:

- The prediction for the bidirectionally predicted pixels in the B-region is obtained as the average of the forward prediction with MVF_F relative to the previous decoded picture, and the backward prediction using MVF_B relative to P_{REC}. The average is calculated by dividing the sum of the two predictions by two (division followed by truncation).
- For all other pixels, only forward prediction using MVF_F relative to the previous decoded picture is used.

The prediction for chrominance is done in the same way it is done for the P-regions.

3.6 Reconstruction of a B-region in PB-frame

Once the prediction for B-region is calculated the prediction error is applied. This is done as in the INTER-B coded regions in the P-frame with the exception that a quantizer parameter for region in B-frame (BQUANT) is obtained by scaling the quantizer parameter for P-frame (PQUANT) as indicated by PBQUANT parameter (Section 4.1.6). Note that, the updated QPs of each region (with RDQUANT) is also reflected to the B-frame with the same scale.

4. Syntax and Semantics

The video bitstream is arranged in a hierarchical structure consisting of three layers. These layers are:

- Picture layer,
- Region layer,
- Block layer.

The syntax diagram is shown in Figure 4-1. Variable Length Codes are shown in round blocks while Fixed Length Codes are shown in angled blocks. Abbreviations and semantics are defined in later sections.

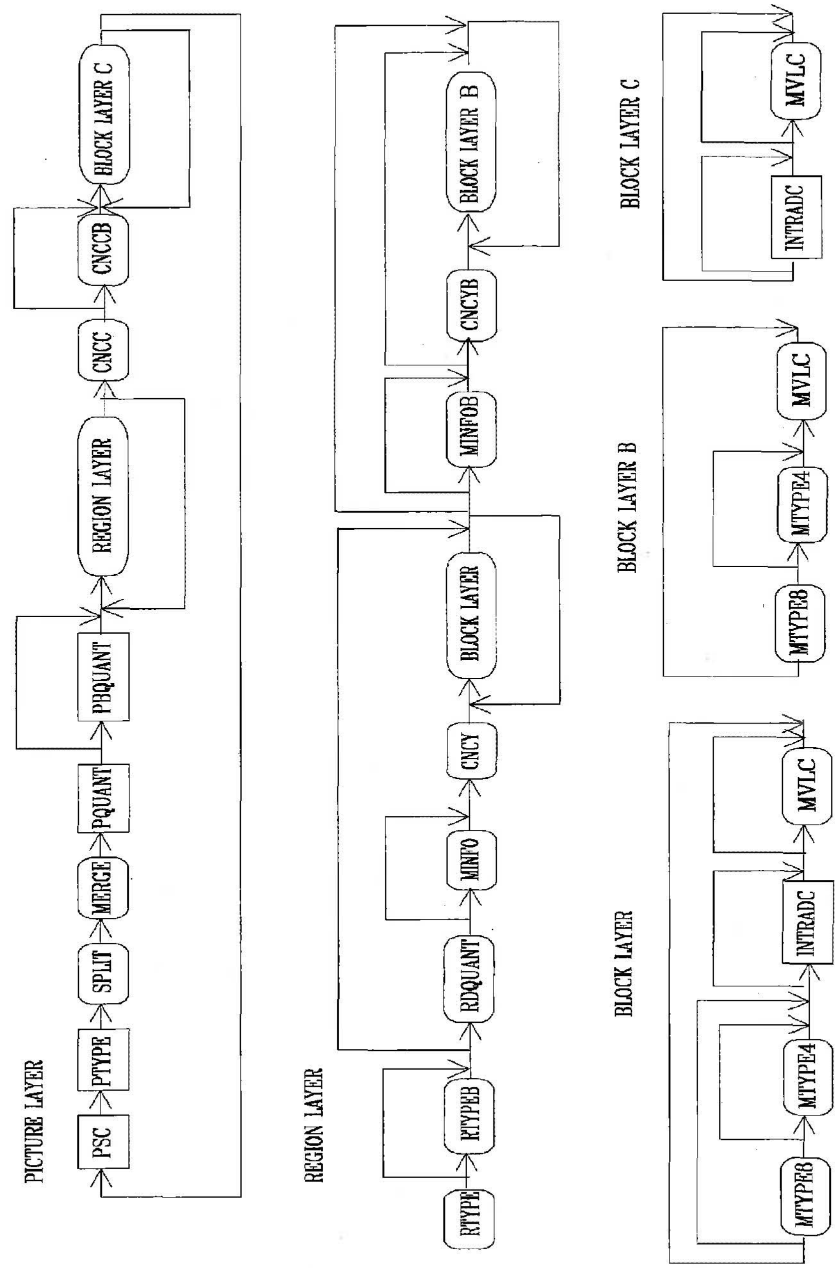


Figure 4-1 Syntax diagram for the video bitstream

4.1 Picture Layer

Data for each picture consists of a picture header followed by data for regions. The structure is shown in Figure 4-2. PBQUANT and CNCCB is present only if PTYPE indicates 'PB-frame'.

PSC	PTYPE	SPLIT	MERGE	PQUANT	PBQUANT	REGION LAYER	CNCC	CNCCB	BLOCK LAYER C

Figure 4-2. Structure of picture layer

4.1.1 Picture Start Code (PSC) (22 bits)

PSC is a word of 22 bits. Its value is 0000 0000 0000 0000 1 00000.

4.1.2 Picture Type Information (PTYPE) (3 bits)

Information about the complete picture is given by this fixed length word consisting of 3 bits. The first bit is 1 if the picture is in QCIF format, and it is 0 if the picture is in CIF format. The second bit is 1 if the PB-frame mode is on, in which case the next bit is DON'T CARE. Otherwise the third bit specifies that the picture is an INTRA frame if it is 1, or an INTER frame if it is 0.

4.1.3 Split Information (SPLIT) (Variable Length)

At the first split level a bit is assigned to each of the regions in the initial segmentation greater than *16 x 16* pel area, i.e., *32x32*, *32x16* and *16x32* regions. The bit value of 1 indicates splitting of the region into smaller *16 x 16* regions whereas the value 0 indicates that the region is left intact. The first sequence of bits is produced by concatenating bits corresponding to the regions scanned row-wise (left to right, top to bottom). Solid lines in Figure 2-3 show the an example of region boundaries after the first level split.

The bits of the second sequence refer to *16 x 16* regions present in the frame segmentation after performing the first split level. The second sequence assigns one bit to each of the *16 x 16* regions. The regions are scanned row-wise (left to right, top to bottom). The solid lines in Figure 2-4 indicate of region boundaries after the second split level. Both sequences of split bits are run-length coded. Separate code tables are used for runs of ones and runs of zeros. This approach requires the first bit of the whole sequence to be transmitted separately at the beginning of the Huffman coded bit sequence. This first bit needs to be included in the length of the first run of the bit sequence. The Huffman codes are given in Table 4 for one runs and Table 5 for zero runs.

After the escape code for one runs a codeword of fixed length of three bits follows. This codeword denotes the run lengths exceeding the maximum Huffman encodable run by one to seven bits. This is achieved by employing the binary representation with the least significant bit as first. Since the maximum run length can not be limited beforehand (to any small number) the codeword of '111' is reserved to indicate that there is another three bits long codeword following up. Finally the three bits long codeword indicates that either there is one to seven additional bits in the run or that there are more than seven bits remaining in the run, and that the number of this bits will be determined by the next codeword. Iteration of this procedure till the end of the run can describe any length of the run. The zero runs exceeding the maximum encodable run length are coded in the same way, but the length of the codeword equals to four instead of three bits.

Table 4

VLC codes of runs of ones for split information

RUN	Number of bits	Code
1	1	1
2	2	01
3	3	001
4	4	0001
5	5	00001
6	6	000001
ESCAPE	6	000000

Table 5

VLC codes of runs of zeros for split information

RUN	Number of bits	Code
1	2	10
2	2	11
3	3	011
4	3	010
5	4	0011
6	4	0001
7	5	00000
8	5	00001
ESCAPE	4	0010

4.1.4 Merge Information (MERGE) (Variable Length)

The merging bits are inserted into the bitstream uncompressed. The exact amount and meaning of these bits are known to the decoder which is assumed to have the same region adjacency graph as the encoder describing adjacency relations of the regions after splitting.

4.1.5 Quantizer Information (PQUANT) (5 bits)

A fixed length codeword of 5 bits which indicates the quantizer value QUANT to be used for the picture. The codewords are the natural binary representations of the values of QUANT which range from 1 to 31. The quantizer step size is 2*QUANT.

4.1.6 Quantization information for B-pictures (PBQUANT) (2 bits)

PBQUANT is present if PTYPE indicates 'PB-frame'. With PB-frames QUANT is used for the P-region, while for the B-region a different quantization parameter BQUANT is used. PBQUANT indicates the relation between QUANT and BQUANT as defined in TABLE 6. In this table, "/" means division by truncation. BQUANT ranges from 1 to 31; if the value for BQUANT resulting from TABLE 6 is greater than 31, it is clipped to 31.

TABLE 6

PBQUANT codes and relation between QUANT and BQUANT

PBQUANT	BQUANT
00	(5xQUANT)/4
01	(6xQUANT)/4
10	(7xQUANT)/4
11	(8xQUANT)/4

4.1.7 Coded/NotCoded Information for Chrominance (CNCC) (Variable Length) and Coded/NotCoded Information for Chrominance of B Frames (CNCCB) (Variable Length)

During prediction error coding, two bit arrays of same size are generated for U and V components of the whole frame. The array has '1' for a coded *8x8* or *4x4* Intra/Inter block, and '0' for not coded ones (UNCHANGED blocks are skipped). For *8x8* INTRA blocks not coded indicates that there is no coded AC coefficients, while for *4x4* INTRA blocks it indicates that there is neither AC nor DC coefficients, since for such junction blocks, the run-length scanning starts from DC coefficient. Scanning is row-wise (even inside a JUNCTION block).These two arrays are merged into one using the prefix-free code in TABLE 7.

TABLE 7

VLC for joining U&V Coded/Not Coded bitstreams

CODE	U	V
0	0	0
10	1	1
110	1	0
111	0	1

This joint array is coded using Truncated Hufman Coding on Run Lengths. The Huffman table is truncated at every 10^{th} symbol. That is n^{th} symbol is coded by *(n-1)/10* truncation symbols and the Huffman code for the $(\mathrm{mod}_{10}(n\text{-}1)+1)^{th}$ symbol. Symbols and the corresponding Huffman codes are as defined in TABLE 8 :

TABLE 8

VLC table for CNCC and CNCCB symbols

Index	Symbol	Number of bits	VLC code
1	1	2	11
2	01	4	1010
3	001	5	10110
4	0001	5	10010
5	00001	5	10001
6	000001	5	10000
7	0000001	6	101111
8	00000001	6	101110
9	000000001	6	100111
10	0000000001	6	100110
--	Truncation	1	0

4.2 Region Layer

Data for each region consists of motion information followed by data for blocks. The structure is shown in Figure 4-3. Each region contains one or more *8x8* blocks. RTYPEB, together with MINFOB, CNCYB and BLOCK LAYER B, contains B-frame information and they are skipped if PTYPE does not indicate PB-frame mode. If the PTYPE indicates P-frame and RTYPE indicates INTRA or NO MOTION INTERregion MINFO is skipped. All RDQUANT, MINFO, CNCY, BLOCK LAYER are skipped if RTYPE indicates an UNCHANGED region. The case for MINFOB, CNCYB and BLOCK LAYER B will be described in the section for RTYPEB.

RTYPE	RDQUANT	RTYPEB	MINFO	CNCY	BLOCK LAYER	MINFOB	CNCYB	BLOCK LAYER B

Figure 4-3 Structure of Region Layer

4.2.1 Region Type Information (RTYPE) (Variable Length)

RTYPE indicates which of the four types UNCHANGED, INTER, NO MOTION INTER and INTRA a region of a predicted frame can be. Symbols and the corresponding Huffman codes are as defined in TABLE 9.

TABLE 9

VLC table for RTYPE symbols

Index	Symbol	Number of bits	Code
1	INTER	1	0
2	NO MOTION INTER	2	10
3	UNCHANGED	3	110
4	INTRA	3	111

4.2.2 Region Type Information for B Frame Region (RTYPEB) (Variable Length)

Region type for a region in B-frame encodes the following three flags. "x" denotes that the flag is set.

BMCC - when the flag is set MCC VLCs for the region are transmitted.

BMVF - when the flag is set area of the region is bidirectionally predicted, otherwise only forward prediction is used

DMVF - when the flag is set Delta motion coefficients for the region are transmitted.

Assignment of status of these flags to VLC codes is shown in TABLE 10

TABLE 10.
VLC table for RTYPEB symbols

Index	BMCC	BMVF	DMVF	Number of bits	Code
0				6	000001
1	x			6	000000
2			x	4	0001
3	x		x	4	0011
4		x		1	1
5	x	x		5	00001
6		x	x	2	01
7	x	x	x	4	0010

4.2.3 Quantization Parameter Offset for Inter and Intra Regions (RDQUANT) (Variable Length)

The QUANT used for an Inter or Intra type region can have an offset of ±1 units with respect to the QUANT of the picture. Note that, if the region belongs to a PB-Frame, B frame is also affected from the offset as much as PBQUANT field indicates. The VLC used can be found in TABLE 6.

TABLE 11
RDQUANT codes

PDQUANT	OFFSET
1	0
01	+1
00	-1

4.2.4 Motion Information (MINFO), Motion Information for B-region (MINFOB) (Variable length)

Motion Information of P region (MINFO) and Motion Information of B-region (MINFOB), i.e., differential motion coefficients both utilize the following syntax:

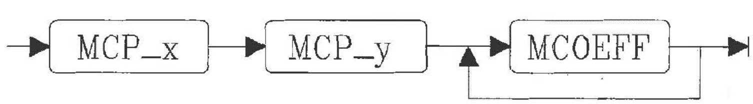


Figure 4-4 Bitstream syntax for motion coefficients

MCP_x and MCP_y are variable bit codewords each denoting one possible combination of six bits: $b_1,\ldots,b_6$. MCP_x refers to the x-component $(c_1,\ldots,c_6)$ and MCP_y to the y-component $(c_7,\ldots,c_{12})$ of the motion field. The bit b_j of MCP_x equal to 1 indicates a non-zero value of the quantized motion coefficient $\tilde{c}_j$. Value 0 of b_j indicates a zero value of $\tilde{c}_j$ after quantization. The bit b_j of MCP_y equal to 1 indicates a non-zero value of the quantized motion coefficient $\tilde{c}_{j+6}$. Value 0 of b_j indicates a zero value of $\tilde{c}_{j+6}$ after quantization. Both MCP_x and MCP_y coded using the same Huffman code. Codes for P-frames are given in TABLE 12, and codes for B-frames are given in TABLE 15. MCOEFF contains information only about non-zero motion coefficients. The motion coefficients are transmitted in increasing order of indices each of them quantized as described in Section 2.3.2. The values of LEVEL are Huffman coded. Codes for P-frames are given in TABLE 13, and codes for B-frames are given in TABLE 14. The last bit of the code denoted as 's' corresponds to the sign of the LEVEL value: 0 - positive, 1 - negative.

TABLE 12

VLC table for Motion Coefficient Patterns MCP_x and MCP_y for P-frames

Index	MC Pattern	No. of bits	VLC Code	Index	MC Pattern	No. of bits	VLC Code
1	000000	3	111	33	000001	6	001101
2	100000	4	1101	34	100001	5	10010
3	010000	6	001110	35	010001	8	00001101
4	110000	4	1010	36	110001	6	010100
5	001000	6	010000	37	001001	7	0001110
6	101000	4	1100	38	101001	5	01110
7	011000	7	0010000	39	011001	8	00001010
8	111000	5	10000	40	111001	6	010110
9	000100	7	0010101	41	000101	8	00000111
10	100100	6	010010	42	100101	7	0010001
11	010100	7	0001101	43	010101	9	000000011
12	110100	6	001111	44	110101	7	0010110
13	001100	7	0011001	45	001101	8	00001111
14	101100	6	011001	46	101101	6	010111
15	011100	7	0010010	47	011101	8	00001100
16	111100	5	10001	48	111101	5	01111
17	000010	7	0011000	49	000011	9	000000010
18	100010	6	010101	50	100011	8	00001001
19	010010	7	0010111	51	010011	9	000000101
20	110010	6	010011	52	110011	7	0001010
21	001010	8	00010000	53	001011	8	00000110
22	101010	6	010001	54	101011	7	0001001
23	011010	8	00010001	55	011011	8	00000100
24	111010	5	01101	56	111011	6	011000
25	000110	9	000000001	57	000111	9	000000100
26	100110	7	0010011	58	100111	8	00001000
27	010110	8	00000011	59	010111	10	0000000000
28	110110	7	0001111	60	110111	7	0001011
29	001110	8	00000101	61	001111	10	0000000001
30	101110	7	0001100	62	101111	7	0010100
31	011110	8	00001110	63	011111	8	00001011
32	111110	5	10011	64	111111	4	1011

TABLE 13

Huffman codes of MCOEFF for P-frames

|LEVEL|	Number of bits	Code	|LEVEL|	Number of bits	Code
1	5	1000s	23	8	0001001s
2	4	111s	24	8	0000111s
3	4	110s	25	9	00001000s
4	4	101s	26	9	00001010s
5	5	1001s	27	9	00001101s
6	5	0111s	28	9	00001100s
7	5	0110s	29	9	00001011s

8	6	01010s	30	9	00001001s
9	6	01011s	31	9	00000111s
10	6	01001s	32	9	00000110s
11	6	00111s	33	10	000000000s
12	6	00110s	34	10	000001000s
13	7	001000s	35	10	000001001s
14	7	001010s	36	10	000000111s
15	7	001011s	37	10	000000110s
16	7	001001s	38	10	000000101s
17	7	000111s	39	10	000000100s
18	8	0001010s	40	10	000000011s
19	8	0001000s	41	10	000000010s
20	8	0001101s	42	10	000000001s
21	8	0001100s	ESC1*)	5	01000
22	8	0001011s	ESC2**)	8	00000101

*) This row corresponds to an 11 bit long word consisting of 5 bits escape code and 6 bits offset covering the interval of LEVEL absolute values from 43 to 106

**) This row corresponds to a 18 bit long word consisting of 8 bits escape code and 10 bits offset covering the interval of LEVEL absolute values from 107 to 1130

TABLE 14

Huffman codes of MCOEFF for B-frames

|LEVEL|	Number of bits	Code
1	2	1s
2	3	01s
3	4	001s
4	5	0001s
5	6	00001s
6	7	000001s
7	9	00000010s
8	9	00000011s
9	9	00000001s
10	9	00000000s

TABLE 15

VLC table for Motion Coefficient Patterns MCP_x and MCP_y for B-frames

Index	MC Pattern	No. of bits	VLC code	Index	MC Pattern	No. of bits	VLC code
1	000000	2	11	33	000001	4	1010
2	100000	4	1011	34	100001	6	010011
3	010000	5	10010	35	010001	7	0010000
4	110000	6	010101	36	110001	7	0010111
5	001000	5	10001	37	001001	6	011001
6	101000	6	011000	38	101001	6	010001
7	011000	7	0011110	39	011001	8	00001111
8	111000	6	011101	40	111001	7	0011000
9	000100	5	10000	41	000101	7	0011010
10	100100	6	011100	42	100101	7	0011111
11	010100	6	011011	43	010101	8	00000111
12	110100	7	0010110	44	110101	8	00000001
13	001100	7	0100001	45	001101	8	00000000

14	101100	7	0100000	46	101101	7	0010010
15	011100	8	00001001	47	011101	7	0010001
16	111100	6	010110	48	111101	6	010010
17	000010	5	01111	49	000011	7	0001111
18	100010	6	010100	50	100011	7	0001110
19	010010	7	0011101	51	010011	8	00001100
20	110010	7	0011100	52	110011	7	0001101
21	001010	7	0011011	53	001011	8	00001010
22	101010	6	011010	54	101011	7	0001100
23	011010	8	00001011	55	011011	8	00001000
24	111010	7	0011001	56	111011	7	0001011
25	000110	8	00001101	57	000111	8	00000110
26	100110	8	00001110	58	100111	8	00000101
27	010110	8	00010000	59	010111	8	00000100
28	110110	7	0010011	60	110111	7	0001010
29	001110	8	00010001	61	001111	8	00000010
30	101110	7	0010101	62	101111	7	0001001
31	011110	8	00000011	63	011111	7	0010100
32	111110	6	010111	64	111111	5	10011

4.2.5 Coded/NotCoded Information for B Frame Region Luminance(CNCYB) (Variable Length) AND Coded/NotCoded Information for P Region Luminance (CNCY) (Variable Length)

This section defines a bitstream consisting of variable length coded overhead information for the luminance component (Y) of a region. The not coded bitstream contains 1 bit per each *8x8* Y block, scanned in alternating row order starting from the left top corner of the region, indicating whether any bits were used to code information in the block. If the block is of type INTER, its being not coded indicates that no bits are spent for any DCT coefficient. If it is of type INTRA only the AC coefficients are of concern as the transmission of 8 bits for DC coefficient is obligatory.

As a variable length code, Huffman coding with escape to fixed length coding option is utilized on the runs of zeros and ones of the stream. These two tables are used in turns. Runs of 0s longer than 13 bits are coded using ESCAPE code followed by fixed length codewords. Runs of 1s are coded in this manner when length of the run exceeds 9. The fixed length codeword size is chosen to be the minimum number of bits required to express if the rest of the bitstream was a single run. The first bit of the coded stream always holds the value of the first bit in the not coded stream and it is also included in the first run. The symbols and the corresponding Huffman codes are as defined in TABLE 16.

TABLE 16

VLC table for CNCY symbols

Index	Symbol	Number of bits	Code	Index	Symbol	Number of bits	Code
1	0	2	11	1	1	1	1
2	00	2	01	2	11	2	01
3	000	3	100	3	111	3	000
4	0000	3	000	4	1111	4	0010
5	00000	4	0010	5	11111	5	00110
6	000000	5	10110	6	111111	6	001110
7	0000000	6	101111	7	1111111	8	00111111
8	00000000	5	10100	8	11111111	8	00111110
9	000000000	6	101010	--	ESCAPE	7	0011110
10	0000000000	7	1011101				
11	00000000000	7	1010111				
12	000000000000	7	1011100				
13	0000000000000	7	1010110				
--	ESCAPE	4	0011				

4.3 Block Layer, Block Layer B and Block Layer C

4.3.1 Block layer

The bitstream structure of Block Layer is shown in Figure 4-5.

MTYPE8	MTYPE4	INTRADC	MVLC

Figure 4-5 Structure of BLOCK LAYER

For not coded blocks, the whole block layer is skipped if it belongs to INTER or NO MOTION INTER region, and only INTRADC field is decoded if it belongs to an INTRA region. For INTRA region blocks, MTYPE8 and MTYPE4 fields are skipped. For INTER and NO MOTION INTER region blocks INTRADC field is skipped. If MTYPE8 indicates an *8x8* coding method which does not involve a *4x4* coding method, then MTYPE4 field is skipped.

4.3.2 8x8 Method Selection Number (MTYPE8) and 4x4 Method Selection Number (MTYPE4) (Variable Length)

Encoding or decoding of both MTYPE8 and MTYPE4 requires the knowledge of the class of the corresponding *8x8* or *4x4* block. The *8x8* class is determined by the classification algorithm described in Section 2.4.2, using only the prediction frame. When the class of the *8x8* block is determined, MTYPE8 is interpreted and the selected *8x8* method is retrieved. The assignment of methods to classes of *8x8* blocks and corresponding VLCs are shown in Table 17 below.

If MTYPE8 indicates the usage of *4x4* method, then there exists the MTYPE4 field to indicate which *4x4* method is used for each coded quadrant. To interpret MTYPE4 classification of the corresponding *4x4* prediction block is required. The *4x4* class is determined by the classification algorithm described in Section 2.4.3, using the prediction frame and the QUANT value. Assignment of *4x4 Coding Methods* to classes of *4x4* blocks and corresponding MTYPE4 VLCs are given in Table 18. The MTYPE4 field contain VLC codes of methods to be used at one to four quadrants. These quadrants are indicated in MTYPE8 and scanned in row wise order both in encoder and decoder.

Table 17

Available methods and corresponding MTYPE8 Huffman Codes for each *8x8* class.

1 Quadrant		2 Quadrants		3 Quadrants		All Quadrants		No Quadrants		Diagonal	
Method	VLC	Method	VLC	Method	VLC	Method	VLC	Method	VLC	Method	VLC
1	11	1	11	1	11	1	11	1	1	1	11
7-2	10	3-1	10	7-2	101	7-15	101	3-2	011	7-2	101
3-1	011	7-2	011	3-1	100	7-3	100	3-1	010	7-8	100
2-1	010	7-1	010	2-1	011	7-12	011	2-2	0011	2-1	011
3-2	001	7-3	0011	7-15	0101	7-6	0101	2-1	0010	3-1	0101
2-2	000	3-2	0010	7-1	0100	7-9	0100	7-2	00011	3-2	0100
		5	0001	7-4	0011	5	0011	7-4	00010	2-2	0011
		7-15	0000	7-6	0010	3-3	0010	7-8	00001	7-14	0010
				7-3	0001	4	0001	7-1	00000	7-12	0001
				6-2	0000	2-3	0000			6-1	0000

TABLE 18

Available methods and corresponding MTYPE4 Huffman Codes of each *4x4* class.

Low Rate						Medium Rate						High Rate					
Low Variance			High Variance			Low Variance			High Variance			Low Variance			High Variance		
Hor	Di-1	Flat	Hor	Dil	Tex	Hor	Di-1	Flat	Hor	Di-1	Tex	Hor	Di-1	Flat	Hor	Di-1	Tex
HHor 1	HDi1 10	Ext 1	HHor 1	HDi1 1	HTex 1	Flat 1	Ext 11	Flat 1	LHor 11	LDi1 1	LTex 11	Flat 1	Flat 1	Flat 11	LHor 11	LDi1 1	LTex 11
Ext 01	Ext 11	HHor 01	HTex 01	HTex 011	HDi1 01	Ext 011	Flat 10	Ext 01	HTex 10	HDi1 011	HTex 10	LHor 01	LDi1 01	Ext 10	LTex 10	LTex 01	LDi1 10
HDi1 001	Flat 011	HDi1 001	HDi1 001	HVer 010	HVer 001	LHor 010	HDi1 01	HDi1 0011	LTex 011	HTex 010	LDi1 01	Ext 0011	Ext 0011	LDi2 011	Flat 01	Flat 001	DCT 01
DCT 000	HVer 010	DCT 0001	DCT 0001	HHor 001	DCT 0001	HTex 001	HTex 001	HTex 0010	HHor 010	LTex 001	HVer 001	LDi1 0010	LTex 0010	LDi1 010	DCT 001	DCT 0001	LVer 001
	HHor 001	HVer 0000	Ext 0000	DCT 0001	HHor 00001	HHor 0001	DCT 000	DCT 0001	LDi1 001	DCT 0001	DCT 0001	LTex 0001	DCT 0001	LTex 001	LDi1 000	HTex 00001	Flat 0001
	DCT 000			Ext 0000	Ext 00000	HDi1 00001		HHor 00001	DCT 0001	Ext 0000	HHor 00001	DCT 0000	HTex 0000	DCT 000		HDi1 00000	LHor 0000
						DCT 00000		HVer 00000	Ext 0000		Ext 00000						

4.3.3 INTRADC (Fixed Length) and VLC for Coding Methods (MVLC) (Variable Length)

DC coefficient for INTRA blocks (INTRADC) has a fixed length of 8 bits and the coding is the same as in Recommendation H.263 [3]. The field, MVLC, may contain coded coefficients of Multi-Shape DCTs or entropy codes of labels of Directional ECVQs. The coding of Multi-Shape DCT coefficients is similar to coding of DCT coefficients in Recommendation H.263, i.e., 2-D block of quantized transform coefficients is scanned to form a 1-D list. Symbols are formed by scanning the nonzero coefficients in the list. Symbol is determined by three parameter: LAST indicates whether current coefficient is the last non-zero coefficient in the list (1 if the coefficient is last nonzero coefficient in the list, 0 otherwise), RUN which indicates the number of successive zeros preceding the coded coefficient, and LEVEL which is the absolute value of the coded coefficient. Symbols (combinations of LAST, RUN, LEVEL) are coded using VLC codes in TABLE 19, Table 20, and Table 21. The 8 x 8 transforms coefficients are coded the same way as in Recommendation H.263 [3]. The last bit "s" denotes the sign of the coefficient, "0" for positive and "1" for negative.

Symbols (LAST, RUN, LEVEL) not contained in VLC tables are coded with a 21 to 22 bit word consisting of 8 bits ESCAPE code, 1 bit LAST, 4 (for 4 x 4 DCT) or 5 (4 x 8, 8 x 4 and 3 x 8 DCTs) bits RUN and 8 bits LEVEL. The codes for RUN are given in Table 22 and for LEVEL in Table 23.

The scanning order of coefficients Multi-Shape DCTs are presented in, Table 24, Table 25, and Table 26. The scanning order of 8 x 8 transform is the same as in Recommendation H.263 [3].

TABLE 19

The VLC CODEs for coefficients of 4 x 8 and 8x4 DCT.

Ind.	LAST	RUN	LEVEL	BITS	VLC CODE	Ind.	LAST	RUN	LEVEL	BITS	VLC CODE
0	0	0	1	4	111s	41	1	0	4	12	00000000111s
1	0	0	2	6	01011s	42	1	1	1	5	1000s
2	0	0	3	8	0001111s	43	1	1	2	10	000010000s
3	0	0	4	9	00001010s	44	1	1	3	13	000000000010s
4	0	0	5	10	000001001s	45	1	2	1	5	1001s
5	0	0	6	11	0000001100s	46	1	2	2	9	00001101s
6	0	0	7	13	000000000000s	47	1	2	3	13	000000000001s
7	0	0	8	12	00000001011s	48	1	3	1	5	1010s
8	0	1	1	5	1011s	49	1	3	2	10	000001010s
9	0	1	2	8	0001011s	50	1	4	1	6	01100s
10	0	1	3	10	000001011s	51	1	4	2	11	0000001111s
11	0	1	4	12	00000000110s	52	1	5	1	6	01110s
12	0	2	1	6	01000s	53	1	5	2	12	00000001101s
13	0	2	2	9	00010011s	54	1	6	1	6	01111s
14	0	2	3	11	0000001000s	55	1	6	2	10	000001111s
15	0	3	1	6	01010s	56	1	7	1	6	01101s
16	0	3	2	9	00010010s	57	1	7	2	12	00000001111s
17	0	3	3	12	00000000100s	58	1	8	1	6	01001s
18	0	4	1	7	001100s	59	1	8	2	13	000000000100s
19	0	4	2	11	0000001010s	60	1	9	1	7	001101s
20	0	5	1	7	001110s	61	1	9	2	13	000000000111s
21	0	5	2	11	0000001011s	62	1	10	1	7	001011s
22	0	6	1	8	0001101s	63	1	10	2	13	000000000110s
23	0	6	2	11	0000001110s	64	1	11	1	7	001111s
24	0	7	1	8	0010001s	65	1	11	2	13	000000000011s
25	0	7	2	12	00000000101s	66	1	12	1	8	0010010s
26	0	8	1	8	0010101s	67	1	13	1	8	0010000s
27	0	8	2	13	000000000101s	68	1	14	1	8	0001100s
28	0	9	1	9	00010001s	69	1	15	1	8	0010011s
29	0	10	1	9	00001111s	70	1	16	1	8	0010100s
30	0	11	1	9	00001011s	71	1	17	1	9	00001100s
31	0	12	1	10	000001100s	72	1	18	1	9	00010000s
32	0	13	1	10	000001101s	73	1	19	1	9	00001110s
33	0	14	1	10	000001110s	74	1	20	1	10	000010001s
34	0	15	1	11	0000001101s	75	1	21	1	10	000010010s
35	0	16	1	12	00000001110s	76	1	22	1	10	000010011s
36	0	17	1	12	00000001100s	77	1	23	1	11	0000001001s
37	0	18	1	12	00000001010s	78	1	24	1	12	00000001000s
38	1	0	1	4	110s	79	1	25	1	12	00000001001s
39	1	0	2	8	0001110s	80	ESCAPE			7	0001010
40	1	0	3	10	000001000s						

TABLE 20

The VLC CODEs for 4 x 4 transform.

Ind.	LAST	RUN	LEVEL	BITS	VLC CODE	Ind.	LAST	RUN	LEVEL	BITS	VLC CODE
0	0	0	1	5	0110s	19	1	1	1	4	100s
1	0	0	2	7	000100s	20	1	1	2	9	00001010s
2	0	0	3	9	00000100s	21	1	2	1	4	101s
3	0	0	4	11	0000000110s	22	1	2	2	11	0000000010s
4	0	1	1	5	0101s	23	1	3	1	4	110s
5	0	1	2	9	00000101s	24	1	3	2	11	0000000100s
6	0	2	1	6	00100s	25	1	4	1	5	0111s
7	0	2	2	10	000000100s	26	1	4	2	11	0000000101s
8	0	3	1	7	000111s	27	1	5	1	5	0100s
9	0	3	2	12	00000000000s	28	1	6	1	6	00101s
10	0	4	1	7	000101s	29	1	7	1	6	00111s
11	0	5	1	8	0000111s	30	1	8	1	6	00110s
12	0	6	1	9	00000011s	31	1	9	1	7	000110s
13	0	7	1	9	00001011s	32	1	10	1	8	0000110s
14	0	8	1	11	0000000011s	33	1	11	1	9	00000111s
15	0	9	1	10	000000101s	34	1	12	1	9	00001000s
16	0	10	1	11	0000000111s	35	1	13	1	11	0000000001s
17	1	0	1	4	111s	36	1	14	1	12	00000000001s
18	1	0	2	9	00000110s	37	ESCAPE			8	00001001

TABLE 21

The VLC CODEs for 3 x 8 transform.

Ind.	LAST	RUN	LEVEL	BITS	VLC CODE	Ind.	LAST	RUN	LEVEL	BITS	VLC CODE
0	0	0	1	5	0110s	24	1	1	1	3	11s
1	0	0	2	9	00001101s	25	1	1	2	10	000000111s
2	0	1	1	6	00110s	26	1	2	1	5	0101s
3	0	1	2	9	00001100s	27	1	2	2	10	000000101s
4	0	2	1	8	0001101s	28	1	3	1	5	0111s
5	0	3	1	8	0001110s	29	1	4	1	5	0100s
6	0	4	1	8	0001111s	30	1	5	1	7	001001s
7	0	5	1	8	0010001s	31	1	6	1	6	00111s
8	0	6	1	9	00001110s	32	1	7	1	7	001010s
9	0	7	1	9	00010011s	33	1	8	1	8	0001011s
10	0	8	1	10	000001011s	34	1	9	1	9	00001001s
11	0	9	1	10	000001100s	35	1	10	1	9	00001010s
12	0	10	1	10	000001001s	36	1	11	1	9	00001011s
13	0	11	1	9	00010000s	37	1	12	1	8	0010000s
14	0	12	1	9	00010001s	38	1	13	1	8	0001010s
15	0	13	1	9	00010010s	39	1	14	1	8	0001100s
16	0	14	1	10	000001000s	40	1	15	1	9	00001000s
17	0	15	1	10	000000110s	41	1	16	1	9	00000111s
18	0	16	1	10	000000100s	42	1	17	1	10	000000010s
19	0	17	1	10	000000011s	43	1	18	1	7	001011s
20	0	18	1	10	000000001s	44	1	19	1	11	0000000000s

21	0	19	1	10	000001101s	45	1	20	1	11	0000000001s
22	1	0	1	3	10s	46	ESCAPE			8	00001111
23	1	0	2	10	000001010s						

TABLE 22

Fixed Length CODEs for RUN.

Index	RUN	CODE for *4x4* DCT	CODE for 4x8, 3x8, 8x4 DCTs
0	0	0000	00000
1	1	0001	00001
2	2	0010	00010
.		.	.
.		.	.
15	15	1111	01111
16	16	FORBIDDEN	10000
.		.	
.		.	
31	31	FORBIDDEN	11111

TABLE 23

Fixed Length CODEs for LEVEL.

Index	LEVEL	CODE
-	-128	FORBIDDEN
0	-127	1000 0001
.	.	.
125	-2	1111 1110
126	-1	1111 1111
-	0	FORBIDDEN
127	1	0000 0001
128	2	0000 0010
.	.	
253	127	0111 1111

TABLE 24

The scanning order of 8 x 4 transform (The scanning order of the 4 x 8 transform is the same).

0	3	4	7	8	11	14	22
1	5	10	13	15	19	23	27
2	9	12	16	18	21	25	29

6	17	20	24	26	28	30	31

TABLE 25

The scanning order of 4 x 4 transform.

0	2	4	9
1	5	7	12
3	6	10	14
8	11	13	15

TABLE 26

The scanning order of 3 x 8 transform.

0	1	4
2	5	20
3	19	21
6	18	22
7	17	23
8	16	15
9	12	14
10	11	13

4.3.4 Block layer B

The structure is shown in Figure 4-6. Since B-Frames do not have INTRA type regions, there is no INTRADC field. The whole Block Layer B is skipped if the block is not coded. If the *8x8* coding method indicated in MTYPE8 does not involve a *4x4* coding method, MTYPE4 field is skipped. The coding methods and VLC tables are same as those described in section 4.3.1,

MTYPE8	MTYPE4	MVLC

Figure 4-6 Structure of BLOCK LAYER B

4.3.5 Block layer C

Coding of chrominance blocks is discussed in detail in Section 2.5. Coding of *8x8* chrominance blocks is the same as in Recommendation H.263. The structure is shown in Figure 4-7. If the PB-frame mode is ON, the block layer C includes the chrominance prediction error information of the B-frame also. INTRADC field is discarded for chrominance blocks of a B-frame. If the *8x8* block is a JUNCTION block MVLC contains the DCT coefficient VLCs of coded quadrants in a row scan order. If *4x4* INTRA blocks exist among the quadrants of the JUNCTION block, then INTRADC field contains the joint DC value of all the *4x4* INTRA blocks.

INTRADC	MVLC

Figure 4-7 Structure of BLOCK LAYER C

The VLC codes for *4 x 4* DCT coefficients for INTER, NO MOTION INTER and INTRA chrominance blocks are same and they are shown in Table 27.

Table 27

The VLC CODEs for 4 x 4 Transform for all Coded Chrominance Blocks.

Ind.	LAST	RUN	LEVEL	BITS	VLC CODE	Ind.	LAST	RUN	LEVEL	BITS	VLC CODE
0	0	0	1	4	000s	42	1	0	5	8	0010110s
1	0	0	2	6	10001s	43	1	0	6	10	101110011s
2	0	0	3	7	100110s	44	1	0	7	9	00101000s
3	0	0	4	8	1001110s	45	1	0	8	9	10000111s
4	0	0	5	8	0010111s	46	1	0	9	12	00101010101s
5	0	0	6	9	10010111s	47	1	0	10	10	100000000s
6	0	0	7	9	10000001s	48	1	0	11	8	0111010s
7	0	0	8	10	001111011s	49	1	0	12	8	1011101s
8	0	0	9	10	100000001s	50	1	0	13	7	001001s
9	0	0	10	10	001010100s	51	1	0	14	6	00110s
10	0	0	11	10	001010110s	52	1	0	15	7	101111s
11	0	0	12	11	1000011010s	53	1	0	16	7	101100s
12	0	0	13	11	1000011001s	54	1	0	17	7	101101s
13	0	0	14	11	1011100000s	55	1	0	18	9	10011111s
14	0	0	15	10	101110001s	56	1	0	19	9	00101001s
15	0	0	16	10	001111101s	57	1	0	20	9	10000010s
16	0	0	17	11	1001011001s	58	1	0	21	15	10111001000001s
17	0	0	18	11	0010101011s	59	1	0	22	14	1011100100011s
18	0	0	19	11	1000001111s	60	1	0	23	12	00111111111s
19	0	0	20	12	10000011101s	61	1	1	1	5	1010s
20	0	0	21	12	00111111100s	62	1	1	2	8	0111011s
21	0	1	1	7	001000s	63	1	1	3	11	1011100101s
22	0	1	2	9	10000101s	64	1	1	4	14	1011100100001s
23	0	1	3	10	100101000s	65	1	1	5	16	101110010000001s
24	0	1	4	10	100000110s	66	1	1	6	16	101110010000000s
25	0	1	5	11	0011111101s	67	1	1	7	14	1011100100010s
26	0	1	6	13	100000111001s	68	1	1	8	13	100000111000s
27	0	1	7	12	00111111000s	69	1	1	9	12	10000110000s
28	0	1	8	12	00111111110s	70	1	2	1	5	0110s
29	0	1	9	11	0010101111s	71	1	2	2	9	10010101s
30	0	1	10	12	00111111101s	72	1	2	3	11	0010101110s
31	0	1	11	12	00111111001s	73	1	3	1	7	001110s
32	0	2	1	9	10000100s	74	1	3	2	11	0011110100s
33	0	2	2	11	1001011000s	75	1	4	1	7	011100s
34	0	2	3	11	1011100001s	76	1	5	1	9	10011110s
35	0	3	1	10	100101001s	77	1	6	1	9	00111100s
36	0	4	1	11	0011110101s	78	1	7	1	10	100101101s
37	0	5	1	12	10111001001s	79	1	8	1	11	1000011011s
38	1	0	1	3	11s	80	1	9	1	12	10000110001s
39	1	0	2	4	010s	81	1	10	1	12	00101010100s
40	1	0	3	6	01111s	82	ESCAPE	0	0	9	001111100s
41	1	0	4	7	100100s						

E-130

5. References

The following references provide supplementary information in regard to concepts and techniques described in this document :

[1] M. Karczewicz, J. Nieweglowski, and P. Haavisto, "Motion compensating video coding using polynomial motion vector field models," accepted to *Image Communication Journal Special Issue on MPEG-4*.

[2] "Core Experiment on Motion Compensated Prediction Using Quadtree Segmentation and Polynomial Motion Fields," contribution m1189 to ISO/IEC JTC1/SC29/WG11, October 1996.

[3] ITU-T Recommendation H.263 (1995): "Video coding for low bitrate communication".

[4] R.G.Keys "Cubic convolution interpolation for digital image processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol 29, no. 6, 1981, pp. 1153-1160.