ITU - Telecommunications Standardization Sector
STUDY GROUP 16
Video Coding Experts Group (VCEG)

Document VCEG-N83 d1
Filename: VCEG-N83d1.doc
Generated: 21 Dec 2001

Question: Q.6/SG16(VCEG)

Source: Thomas Wiegand
Heinrich Hertz Institute (HHI)
Einsteinufer 37
D-10587 Berlin, Germany

Tel: +49 - (0)30 - 31002 617
Fax: +49 - (0)30 - 392 72 00
Email: wiegand@hhi.de

Title: H.26L Test Model Long-Term Number 9 (TML-9) draft0.

Purpose: Test model

This d0 version of TML-9 includes the following changes from TML-8 (as adopted or announced in Santa Barbara):

- Leaky bucket measurement [VCEG-N58r1]
- Fast loop-filter algorithm [VCEG-N08]
- Advanced Error Concealment [VCEG-N63]
- Filtering outside of the picture [VCEG-N19]
- Fast sub-pel interpolation method [VCEG-N31]
- Robust Lagrangian Coder Control [VCEG-N38,VCEG-N39,VCEG-N50]
- Supplemental enhancement information [VCEG-N60]
- Clarification of B-picture coding
- SP-frame Specification
- Generalized ERPS syntax [VCEG-O10]

CONTENTS

1 Scope

This document is a description of a reference coding method to be used for the development of a new compression method ITU-T recommendation- H.26L. The basic configuration of the algorithm is similar to H.263.

Some of the differences from H.263 are:

- Only one regular VLC or context-based adaptive binary arithmetic coding is used for symbol coding
- ¼-sample or 1/8-sample accuracy used for motion prediction
- A number of different block sizes are used for motion prediction
- Residual coding is based on 4x4 blocks and a integer transform is used
- Multiple reference frames may be used for prediction

The first part of the document describes the coding method by mainly defining the decoder actions. Towards the end, 'test model issues' relevant for the encoder to be used as reference for the development of the standard will be covered. This split should make it easy at a later stage to split the document into what is relevant for the standard and a test model. However, at the moment we find it preferable to have one combined document.

~~2~~

~~3~~2 Organization of source- and compressed data

~~3.1~~2.1 Picture formats

At the moment only the QCIF and CIF formats are included in the model [Not true of the software anymore – the document needs to be updated about this (but not relevant to testing at QCIF and CIF resolutions)].

~~3.2~~2.2 Subdivision of a picture into macroblocks

A CIF picture is divided into 18x22 = 396 macroblocks. Similarly a QCIF picture is divided into 99 macroblocks as indicated in FIGURE 1. At the moment there are no other layers in the described model. [Slice layers are supported in the software, but not fully reflected in this document.]

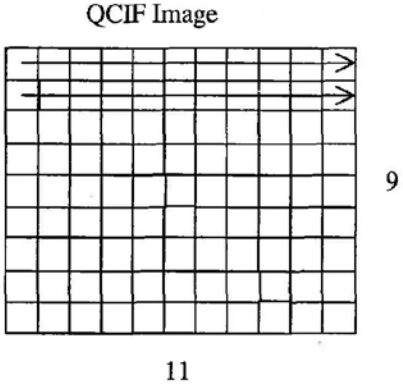


FIGURE 1

Subdivision of a QCIF picture into 16x16 macroblocks

~~3.3~~2.3 Order of the bitstream within a macroblock

FIGURE 2 and FIGURE 3 indicate how a macroblock is divided and the order of the different syntax elements resulting from coding a macroblock.

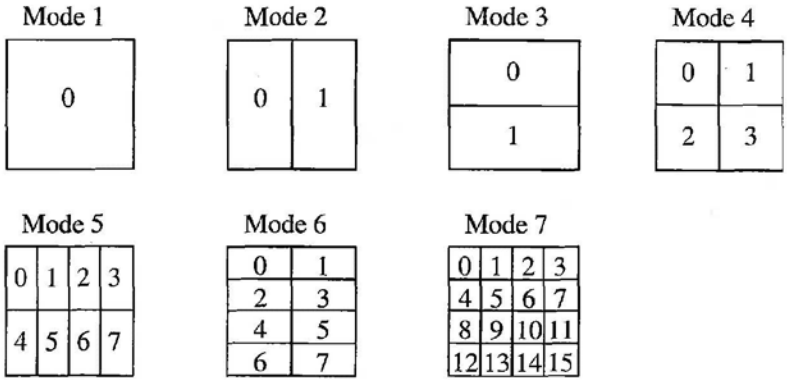
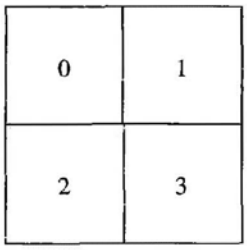


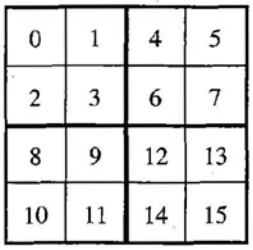
FIGURE 2

Numbering of the vectors for the different blocks depending on the inter mode. For each block the horizontal component comes first followed by the vertical component

CBPY 8x8 block order



Luma residual coding 4x4 block order



Chroma residual coding 4x4 block order

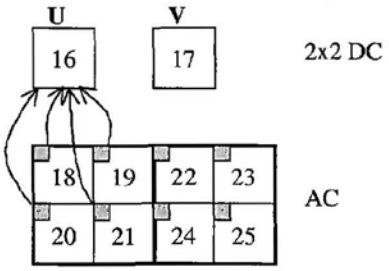


FIGURE 3

Ordering of blocks for CBPY and residual coding of 4x4 blocks

~~3.4~~2.4 Syntax

~~3.4.1~~2.4.1 Syntax diagram

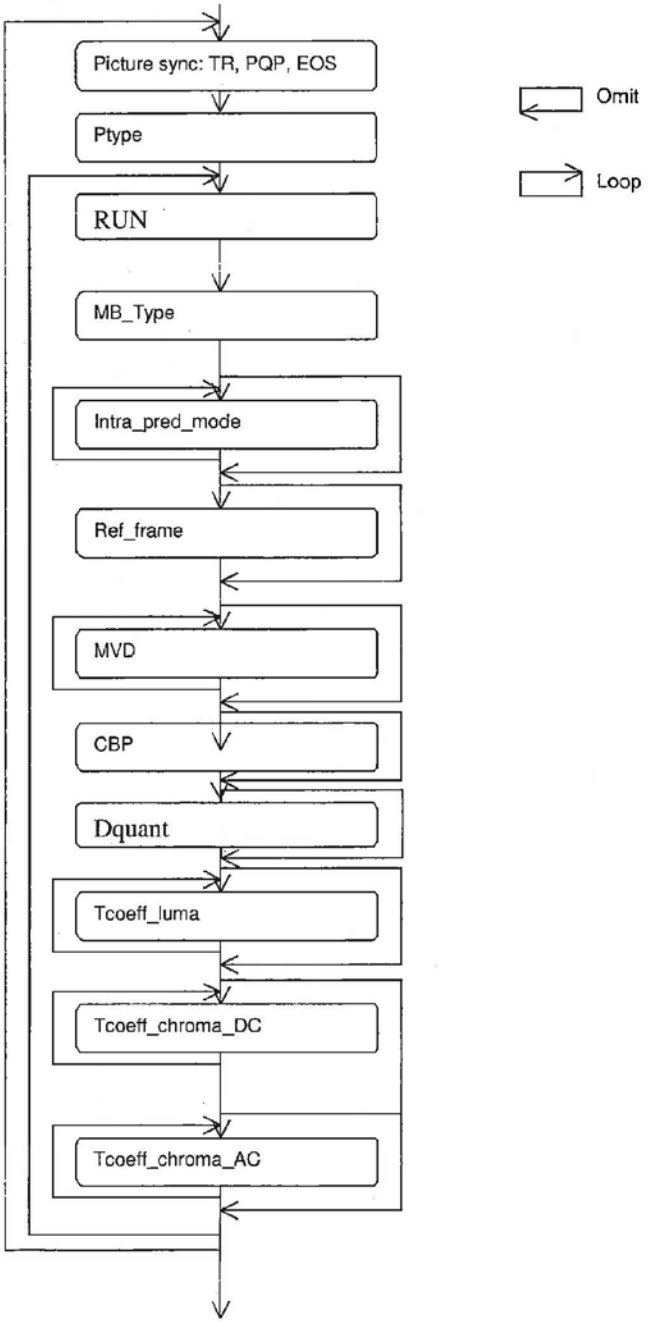


FIGURE 4

Syntax diagram for all the elements in the video bitstream

43 Description of syntax elements

4.13.1 Picture sync

The first codeword in a picture is 31 bits long (L = 31). It works as a picture sync, but it also contains an INFO part that has 15 bits. These bits are used for:

TR(8 bits) Temporal reference. The value of TR is formed by incrementing its value in the temporally-previous reference picture header by one plus the number of skipped or non-reference pictures at the picture clock frequency since the previously transmitted one.

PQP(5 bits) Picture QP: Information about the quantizer QUANT to be used for luma for the picture. (See under Quantization concerning QUANT for chroma). The 5 bit representation is the natural binary representations of the values of QP which range from 0 to 31. QP is a pointer to the actual quantization parameter QUANT to be used. (See below under quantization). The range of quantization value is still about the same as for H.263, 1-31. An approximate relation between the QUANT in H.263 and QP is: QUANTH.263(QP) ≈ QP0(QP) = 2QP/6 . QP0() will be used later for scaling purposes when selecting prediction modes.

Formats 0 indicates QCIF, 1 indicates CIF

EOS 0 for picture header. 1 indicates End Of Sequence

The sketch below indicates where the different bits are located within INFO.

8b: TR	5b: PQP	1b: Formats	1b: EOS

FIGURE 5

Syntax diagram for the Picture header.

4.23.2 Picture type (Ptype)

Code_number =0: Inter picture with prediction from the most recent decoded picture only.

Code_number =1: Inter picture with possibility of prediction from more than one previous decoded picture. For this mode information reference picture for prediction must be signalled for each macroblock.

Code_number =2: Intra picture.

Code_number =3: B picture with prediction from the most recent previous decoded and subsequent decoded pictures only.

Code_number =4: B picture with possibility of prediction from more than one previous decoded picture and subsequent decoded picture. When using this mode, information reference frame for prediction must be signalled for each macroblock.

4.33.3 RUN

A macroblock is called skipped if no information is sent. In that case the reconstruction of an inter macroblock is made by copying the collocated picture material from the last decoded frame. (See separate definition for B-pictures). RUN indicates the number of skipped macroblocks in an inter- or B-picture before a coded picture. If the last MB of the frame is not coded an additional codeword is added which points to a MB outside the picture.

4.43.4 Macro block type (MB_Type)

Refer to TABLE 5. There are different MB-Type tables for Intra and Inter frames.

~~4.4.1~~3.4.1 Intra

Intra 4x4	Intra coding as defined in sections 0 to 3.5.8.
Imode, nc, AC	See definition in section 3.5.9.3. These modes refer to 16x16 intra coding.

~~4.4.2~~3.4.2 Inter

Skip	No further information about the macroblock is transmitted. A copy of the colocated macroblock in the most recent decoded picture is used as reconstruction for the present macroblock.
NxM (eg. 8x4)	The macroblock is predicted from a past picture with block size NxM. For each NxM block motion vector data is provided. Depending on N and M there may be 1 to 16 sets of motion vector data for a macroblock.
Intra 4x4	4x4 intra coding.

Code numbers from 9 and upwards represent 16x16 intra coding.

~~4.5~~3.5 Intra prediction mode (Intra_pred_mode)

Even in Intra mode, prediction is always used for each sub block in a macroblock. A 4x4 block is to be coded (pixels labeled a to p below). The pixels A to I from neighboring blocks are already decoded and may be used for prediction.

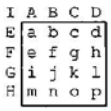


FIGURE 6

Syntax diagram for the Picture header.

There are 6 intra prediction modes labeled 0 to 5. Mode 0 is 'DC-prediction' (see below). The other modes represent directions of predictions as indicated below.

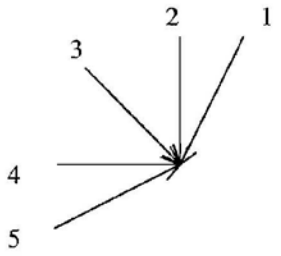


FIGURE 7

Syntax diagram for the Picture header.

~~4.5.1~~3.5.1 Mode 0: DC prediction

Generally all pixels are predicted by (A+B+C+D+E+F+G+H)//8. If four of the pixels are outside the picture, the average of the remaining four is used for prediction. If all 8 pixels are outside the picture the prediction for all pixels in the block is 128. A block may therefore always be predicted in this mode.

~~4.5.2~~3.5.2 Mode 1: Vertical/Diagonal Prediction

This mode is used only if all A,B,C,D are inside the picture.

a is predicted by: (A+B)/2

e is predicted by	B
b,i are predicted by	(B+C)/2
f,m are predicted by	C
c,j are predicted by	(C+D)/2
d,g,h,k,l,n,o,p are predicted by	D

~~4.5.3~~3.5.3 Mode 2: Vertical prediction

If A,B,C,D are inside the picture, a,e,i,m are predicted by A, b,f,j,n by B etc.

~~4.5.4~~3.5.4 Mode 3: Diagonal prediction

This mode is used only if all A,B,C,D,E,F,G,H,I are inside the picture. This is a 'diagonal' prediction.

m is predicted by:	(H+2G+F)//4
i,n are predicted by	(G+2F+E)//4
e,j,o are predicted by	(F+2E+I)//4
a,f,k,p are predicted by	(E+2I+A)//4
b,g,l are predicted by	(I+2A+B)//4
c,h are predicted by	(A+2B+C)//4
d is predicted by	(B+2C+D)//4

~~4.5.5~~3.5.5 Mode 4: Horizontal prediction

If E,F,G,H are inside the picture, a,b,c,d are predicted by E, e,f,g,h by F etc.

~~4.5.6~~3.5.6 Mode 5: Horizontal/Diagonal prediction

This mode is used only if all E,F,G,H are inside the picture.

a is predicted by:	(E+F)/2
b is predicted by	F
c,e are predicted by	(F+G)/2
f,d are predicted by	G
i,g are predicted by	(G+H)/2
h,j,k,l,m,n,o,p are predicted by	H

~~4.5.7~~3.5.7 Prediction of chroma blocks

For chroma prediction there is only one mode. No information is therefore needed to be transmitted. The prediction is indicated in the figure below. The 8x8 chroma block consists of 4 4x4 blocks A,B,C,D. S0,1,2,3 are the sums of 4 neighbouring pixels.

If S0, S1, S2, S3 are all inside the frame:

A = (S0 + S2 + 4)/8

B = (S1 + 2)/4

C = (S3 + 2)/4

D = (S1 + S3 + 4)/8

If only S0 and S1 are inside the frame:

A = (S0 + 2)/4

B = (S1 + 2)/4

C = (S0 + 2)/4

D = (S1 + 2)/4

If only S2 and S3 are inside the frame:

A = (S2 + 2)/4

B = (S2 + 2)/4

C = (S3 + 2)/4

D = (S3 + 2)/4

If S0, S1, S2, S3 are all outside the frame: A = B = C = D = 128

(Note: This prediction should be considered changed)

	S0	S1
S2	A	B
S3	C	D

~~4.5.8~~3.5.8 Coding of Intra prediction modes

Since each of the 4x4 luma blocks shall be assigned a prediction mode, this will require a considerable number of bits if coded directly. We have therefore tried to find more efficient ways of coding mode information. First of all we observe that the chosen prediction of a block is highly correlated with the prediction modes of adjacent blocks. This is illustrated in FIGURE 8a. When the prediction modes of A and B are known (including the case that A or B or both are outside the picture) an ordering of the most probable, next most probable etc. of C is given. This ordering is listed in

TABLE 1. For each prediction mode of A and B a list of 5 numbers is given. Example: Prediction mode for A and B is 2. The string 2 1 0 3 4 5 indicates that mode 2 is also the most probable mode for block C. Mode 1 is the next most probable one etc. In the bitstream there will for instance be information that Prob0 = 1 (see TABLE 5) indicating that the next most probable mode shall be used for block C. In our example this means Intra prediction mode 1. Use of '–' in the table indicates that this instance can not occur because A or B or both are outside the picture.

For more efficient coding, information on intra prediction of two 4x4 luma blocks are coded in one codeword (Prob0 and Prob1 in TABLE 5). The order of the resulting 8 codewords is indicated in FIGURE 8b.

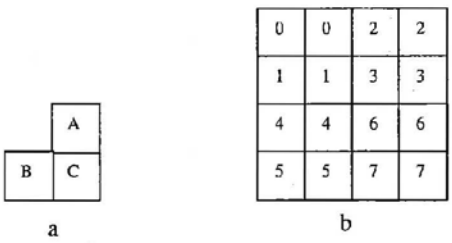


FIGURE 8

a) Prediction mode of block C shall be established. A and B are adjacent blocks. b) order of intra prediction information in the bitstream

TABLE 1

Prediction mode as a function of ordering signalled in the bitstream (see text)

B\A	outside	0	1	2	3	4	5

outside	0-----	021---	102---	201---	012---	012---	012---
0	045---	041352	104325	230415	304215	043152	043512
1	045---	014325	102435	203145	032145	041325	014352
2	045---	012345	102345	210345	302145	042135	013245
3	045---	304152	310425	231054	304215	403512	305412
4	405---	403512	401532	240351	430512	403512	405312
5	504---	540312	015432	201453	530412	450312	504132

~~4.5.9~~3.5.9 Intra mode based on 16x16 macroblocks (16x16 intra mode)

This intra mode is particularly suitable for regions with little details, also referred to as 'flat' regions.

~~4.5.9.1~~3.5.9.1 Prediction modes

Assume that the block to be predicted has pixel locations 0 to 15 horizontally and 0 to 15 vertically. We use the notation P(i,j) where i,j = 0..15. P(i,-1), i=0..15 are the neighboring pixels above the block and P(-1,j), j=0..15 are the neighboring pixels to the left of the block. Pred(i,j) i,j = 0..15 is the prediction for the whole luma macroblock. We have 4 different prediction modes:

- IMODE = 0 (vertical)
  Pred(i,j) = P(i,-1), i,j=0..15
- IMODE = 1 (horizontal)
  Pred(i,j) = P(-1,j), i,j=0..15
- IMODE = 2 (DC prediction)
  $\text{Pred(i,j)} = (\sum_{i=0}^{15}(P(-1,i)+P(i,-1)))/32 \quad i,j=0..15$
- IMODE = 3 (Plane prediction)
  Pred(i,j) = (a + bx(i-7) + cx(j-7) +16)/32
  Where:
  a = 16x(P(-1,15) + P(15,-1))
  b = 5x(*H*/4)/16
  c = 5x(*V*/4)/16
  $H = \sum_{i=1}^{8} ix(P(7+i,-1)+P(7-i,-1))$
  $V = \sum_{j=1}^{8} jx(P(-1,7+j)+P(-1,7-j))$

~~4.5.9.2~~3.5.9.2 Residual coding

The residual coding is based on 4x4 transform. But similar to coding of chroma coefficients, another 4x4 transform to the 16 DC coefficients in the macroblock are added. In that way we end up with an overall DC for the whole MB which works well in flat areas.

Since we use the same integer transform to DC coefficients, we have to perform additional normalization to those coefficients, which implies a division by 676. To avoid the division we performed normalization by $49/2^{15}$ on the encoder side and $48/2^{15}$ on the decoder side, which gives sufficient accuracy.

Only single scan is used for 16x16 intra coding.

To produce the bitstream, we first scan through the 16 'DC transform' coefficients. There is no 'CBP' information to indicate no coefficients on this level. If AC = 1 (see below) ac coefficients of the 16 4x4 blocks are scanned. There are 15 coefficients in each block since the DC coefficients are included in the level above.

~~4.5.9.3~~3.5.9.3 Signalling of mode information for 16x16 intra coding

See TABLE 5. Three parameters have to be signaled. They are all included in MB-type.

Imode:	0,1,2,3
AC:	0 means there are no ac coefficients in the 16x16 block. 1 means that there is at least one ac coefficient and all 16 blocks are scanned.
nc:	CBP for chroma (see 3.8)

~~4.6~~3.6 Reference frame (Ref_frame)

If PTYPE indicates possibility of prediction from more than one previous decoded picture, the exact frame to be used must be signalled. This is done according to the following table.

Code_number	Reference frame
0	The last decoded frame (1 frame back)
1	2 frames back
2	3 frames back
..	..

~~4.7~~3.7 Motion Vector Data (MVD)

If so indicated by MB_type, vector data for 1-16 blocks are transmitted. For every block a prediction is formed for the horizontal and vertical components of the motion vector. MVD signals the difference between the vector component to be used and this prediction. The order in which vector data is sent is indicated in FIGURE 2. Motion vectors are allowed to point to pixels outside the reference frame. If a pixel outside the reference frame is referred to in the prediction process, the nearest pixel belonging to the frame (an edge or corner pixel) shall be used. All fractional pixel positions shall be interpolated as described below. If a pixel referred in the interpolation process (necessarily integer accuracy) is outside of the reference frame it shall be replaced by the nearest pixel belonging to the frame (an edge or corner pixel). Reconstructed motion vectors shall be clipped to +-19 integer pixels outside of the frame.

~~4.7.1~~3.7.1 Fractional pixel accuracy

~~4.7.1.1~~3.7.1.1 1/4 luminance sample interpolation

The pixels labeled as 'A' represent original pixels at integer positions and other symbols represent the pixels to be interpolated

A	d	**b**	d	**A**
e	h	f	h	
b	g	**c**	g	**b**
e	h	f	i	
A		**b**		**A**

The interpolation proceeds as follows

1 The 1/2 sample positions labeled as 'b' are obtained by first applying 6-tap filter (1,-5,20,20,-5,1) to nearest pixels at integer locations in horizontal or vertical direction. The resulting intermediate value *b* is divided by 32, rounded to the nearest integer and clipped to lie in the range [0, 255].

2 The 1/2 sample position labeled as 'c' is obtained using 6-tap filtering (1,-5,20,20,-5,1) of intermediate the values *b* of the closest 1/2 sample positions in vertical or horizontal direction. The obtained value is divided by 1024, rounded to the nearest integer and clipped to lie in the range [0, 255].

3 The 1/4 sample positions labeled as 'd', 'g', 'e' and 'f' are obtained by averaging (with truncation to the nearest integer) the two nearest samples at integer or 1/2 sample position using d=(A+b)/2, g=(b+c)/2, e=(A+b)/2, f=(b+c)/2.

4 The 1/4 sample positions labeled as 'h' are obtained by averaging (with truncation to nearest integer) the two nearest pixels 'b' in diagonal direction.

5 The pixel labeled 'i' (the "funny" position) is computed as $(A_1+A_2+A_3+A_4+2)/4$ using the four nearest original pixels.

~~4.7.1.2~~3.7.1.2 1/8 Pel Luminance Samples Interpolation

The pixels labeled as 'A' represent original pixels at integer positions and other symbols represent 1/2, 1/4 and 1/8 pixels to be interpolated.

A	d	**b^1**	d	**b^2**	d	**b^3**	d	**A**
d	e	d	f	d	f	d	e	
b^1	d	**c^{11}**	d	**c^{12}**	d	**c^{13}**	d	
d	f	d	g	d	g	d	f	
b^2	d	**c^{21}**	d	**c^{22}**	d	**c^{23}**	d	
d	f	d	g	d	g	d	e	
b^3	d	**c^{31}**	d	**c^{32}**	d	**c^{33}**	d	
d	e	d	f	d	f	d	e	
A								**A**

The interpolation proceeds as follows

1 The 1/2 and 1/4 sample positions labeled as 'b' are obtained by first calculating intermediate values '*b*' using 8 tap filtering of nearest pixels at integer locations in horizontal or vertical direction. The final value of 'b' is calculated as (*b*+128)/256 and clipped to lie in the range [0, 255]. The 1/2 and 1/4 sample positions labeled as 'c' are obtained by 8 tap filtering of intermediate values *b* in horizontal or vertical direction, dividing the result of filtering by 65536, rounding it to the nearest integer and clipping it to lie in the range [0, 255]. The filter tap values depend on pixel position and are listed below:

'b', 'c' position	filter tabs
1/4	(-3, 12, -37, 229, 71, -21, 6, -1)
2/4	(-3, 12, -39, 158, 158, -39, 12, -3)
3/4	(-1, 6, -21, 71, 229, -37, 12, -3)

2 The 1/8 sample positions labeled as 'd' are calculated as the average (with truncation to the nearest integer) of the two closest 'A', 'b' or 'c' values in the horizontal or vertical direction.

3 The 1/8 sample positions labeled as 'e' are calculated by averaging with truncation of two 1/4 pixels labeled as 'b^1', which are the closest in the diagonal direction *[Editor: what about the e-position column 8, row 6 ?]*. The 1/8 sample positions labeled as 'g' are calculated via $(A+3c^{22}+2)/4$ and the 1/8 sample positions marked as 'f' are calculated via $(3b^1+b^1+2)/4$ (pixel 'b^2' closer to 'f' is multiplied by 3).

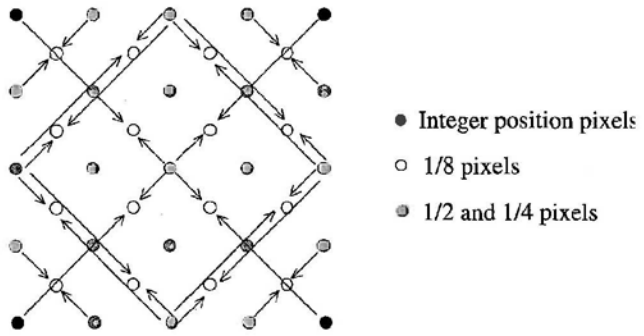


FIGURE 9

Diagonal interpolation for 1/8 pel interpolation.

~~4.7.2~~3.7.2 Prediction of vector components

In all macroblocks with square motion vector blocks (16x16, 8x8, 4x4) "median prediction" (see 3.7.2.1) is used. In case the macroblock may be classified to have directional segmentation the prediction is defined in 3.7.2.2.

~~4.7.2.1~~3.7.2.1 Median prediction

In the figure below the vector component E of the indicated block shall be predicted. The prediction is normally formed as the median of A, B and C. However, the prediction may be modified as described below. Notice that it is still referred to as "median prediction"

A The component applying to the pixel to the left of the upper left pixel in E

B The component applying to the pixel just above the upper left pixel in E

C The component applying to the pixel above and to the right of the upper right pixel in E

D The component applying to the pixel above and to the left of the upper left pixel in E

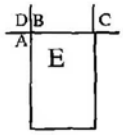


FIGURE 10

Median prediction of motion vectors.

A, B, C, D and E may represent motion vectors from different reference pictures. As an example we may be seeking prediction for a motion vector for E from the last decoded picture. A, B, C and D may represent vectors from 2, 3, 4 and 5 pictures back. The following substitutions may be made prior to median filtering.

- If A and D are outside the picture, their values are assumed to be zero and they are considered to have "different reference picture than E".
- If D, B, C are outside the picture, the prediction is equal to A (equivalent to replacing B and C with A before median filtering).
- If C is outside the picture or still not available due to the order of vector data (see FIGURE 2), C is replaced by D.

If any of the blocks A, B, C, D are intra coded they count as having "different reference frame. If one and only one of the vector components used in the median calculation (A, B, C) refer to the same reference picture as the vector component E, this one vector component is used to predict E.

~~4.7.2.2~~3.7.2.2 Directional segmentation prediction

If the macroblock where the block to be predicted belongs to has a directional segmentation (vector block size of 8x16, 16x8, 8x4 or 4x8) the prediction is generated as follows (refer to figure below and the definitions of A, B, C, E above):

Vector block size 8x16:

Left block:	A is used as prediction if it has the same reference picture as E, otherwise "Median prediction" is used
Right block:	C is used as prediction if it has the same reference picture as E, otherwise "Median prediction" is used

Vector block size 16x8:

Upper block:	B is used as prediction if it has the same reference picture as E, otherwise "Median prediction" is used
Lower block:	A is used as prediction if it has the same reference picture as E, otherwise "Median prediction" is used

Vector block size 8x4:

For white blocks: "Median prediction" is used

For shaded blocks: A is used as prediction

Vector block size 4x8:

For white blocks: "Median prediction" is used

For shaded blocks: B is used as prediction

If the indicated prediction block is outside the picture, the same substitution rules are applied as in the case of median prediction.

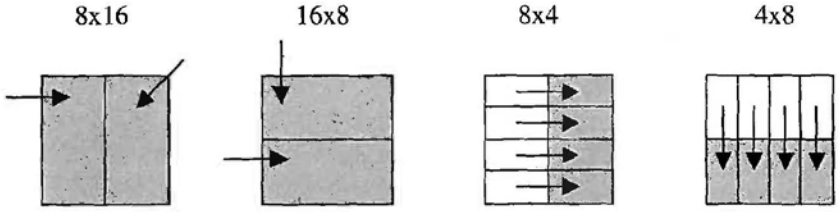


FIGURE 11

Directional segmentation prediction

~~4.7.3~~3.7.3 Chroma vectors

Chroma vectors are derived from the luma vectors. Since chroma has half resolution compared to luma, the chroma vectors are obtained by a division of two:

Croma_vector = Luma_vector/2 - which means that the chroma vectors have a resolution of 1/8 pixel.

Due to the half resolution, a chroma vector applies to 1/4 as many pixels as the luma vector. For example if the luma vector applies to 8x16 luma pixels, the corresponding chroma vector applies to 4x8 chroma pixels and if the luma vector applies to 4x4 luma pixels, the corresponding chroma vector applies to 2x2 chroma pixel.

For fractional pixel interpolation for chroma prediction, bilinear interpolation is used. The result is rounded to the nearest integer.

~~4.8~~3.8 Coded Block Pattern (CBP)

The CBP contains information of which 8x8 blocks - luma and chroma - contain transform coefficients. Notice that an 8x8 block contains 4 4x4 blocks meaning that the statement '8x8 block contains coefficients' means that 'one or more of the 4 4x4 blocks contain coefficients'. The 4 least significant bits of CBP contain information on which of 4 8x8 luma blocks in a macroblock contains nonzero coefficients. Let us call these 4 bits CBPY. The ordering of 8x8 blocks is indicated in FIGURE 3. A 0 in position n of CBP (binary representation) means that the corresponding 8x8 block has no coefficients whereas a 1 means that the 8x8 block has one or more non-zero coefficients.

For chroma we define 3 possibilities:

nc=0:	no chroma coefficients at all.
nc=1	There are nonzero 2x2 transform coefficients. All chroma AC coefficients = 0. Therefore we do not send any EOB for chroma AC coefficients.
nc=2	There may be 2x2 nonzero coefficients and there is at least one nonzero chroma AC coefficient present. In this case we need to send 10 EOBs (2 for DC coefficients and 2x4=8 for the 8 4x4 blocks) for chroma in a macroblock.

The total CBP for a macroblock is: **CBP = CBPY + 16xnc**

The CBP is signalled with a different codeword for Inter macroblocks and Intra macroblocks since the statistics of CBP values are different in the two cases.

~~4.9~~3.9 Dquant

Dquant contains the possibility of changing QUANT on the macroblock level. It does not need to be present for macroblocks without nonzero transform coefficients. More specifically Dquant is present for non-skipped macroblocks:

- If CBP indicates that there are nonzero transform coefficients in the MB **or**
- If the MB is 16x16 based intra coded

The value of Dquant may range from -16 to +16 which enables the QP to be changed to any value in the range 0-31.

$QUANT_{new} = modulo_{32}(QUANT_{old} + Dquant + 32)$ (also known as "arithmetic wrap")

~~5~~4 Transform and inverse transform

This section defines all elements related to transform coding and decoding. It is therefore relevant to all the syntax elements 'Tcoeff' in the syntax diagram.

~~5.1~~4.1 4x4 block size

Instead of DCT, an integer transform with basically the same coding property as a 4x4 DCT is used. The transformation of the pixels a,b,c,d into 4 transform coefficients A,B,C,D is defined by:

```
A = 13a + 13b + 13c + 13d
B = 17a +  7b -  7c - 17d
C = 13a - 13b - 13c + 13d
D =  7a - 17b + 17c -  7d
```

The inverse transformation of transform coefficients A,B,C,D into 4 pixels a',b',c',d' is defined by:

```
a' = 13A + 17B + 13C +  7D
b' = 13A +  7B - 13C - 17D
c' = 13A -  7B - 13C + 17D
d' = 13A - 17B + 13C -  7D
```

The relation between a and a' is: a' = 676a. This is because the expressions defined above contain no normalisation. Normalisation will be performed in the quantization/dequantisation process and a final shift after inverse quantization.

The transform/inverse is performed both vertically and horizontally in the same manner as in H.263. By the above exact definition of inverse transform, the same operations will be performed on coder and decoder side which means that we have no 'inverse transform mismatch'.

~~5.2~~4.2 2x2 transform/inverse transform of chroma DC coefficients

With the low resolution of chroma it seems to be preferable to have larger blocksize than 4x4. Specifically the 8x8 DC coefficient seems very useful for better definition of low resolution chroma. The 2 dimensional 2x2 transform procedure is illustrated below. DC0,1,2,3 are the DC coefficients of 2x2 chroma blocks.

DC0	DC1	Two dimensional 2x2 transform ⇒	DDC(0,0)	DDC(1,0)
DC2	DC3		DDC(0,1)	DDC(1,1)

Definition of transform:

DCC(0,0) = (DC0+DC1+DC2+DC3)/2

DCC(1,0) = (DC0-DC1+DC2-DC3)/2

DCC(0,1) = (DC0+DC1-DC2-DC3)/2

DCC(1,1) = (DC0-DC1-DC2+DC3)/2

Definition of inverse transform:

DC0 = (DCC(0,0)+ DCC(1,0)+ DCC(0,1)+ DCC(1,1))/2
DC1 = (DCC(0,0)- DCC(1,0)+ DCC(0,1)- DCC(1,1))/2
DC2 = (DCC(0,0)+ DCC(1,0)- DCC(0,1)- DCC(1,1))/2
DC3 = (DCC(0,0)- DCC(1,0)- DCC(0,1)+ DCC(1,1))/2

~~5.3.~~4.3.3 Zig-zag Scanning and quantization

~~5.3.1~~4.3.1 Simple zig-zag scan

Except for Intra coding of luma with QP<24, simple scan is used. This is basically zig-zag scanning similar to the one used in H.263. The scanning pattern is:

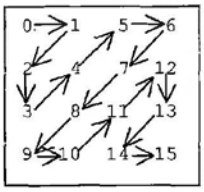


FIGURE 12

Simple zig-zag scan.

~~5.3.2~~4.3.2 Double zig-zag scan

When using the VLC defined above, we use a one bit code for EOB. For Inter blocks and Intra with high QP the probability of EOB is typically 50% which is well matched with the VLC. In other words this means that we on average have one non-zero coefficient per 4x4 block in addition to the EOB code (remember that a lot of 4x4 blocks only have EOB). On the other hand, for Intra coding we typically have more than one non-zero coefficient per 4x4 block. This means that the 1 bit EOB becomes inefficient. To improve on this the 4x4 block is subdivided into two parts that are scanned separately and with one EOB each. The two scanning parts are shown below – one of them in bold.

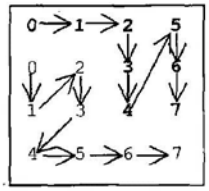


FIGURE 13

Double zig-zag scan.

~~5.3.3~~4.3.3 Quantization

The quantization/dequantization process shall perform 'normal' quantization/dequantization as well as take care of the fact that transform operations are kept very simple and therefore do not contain normalization of transform coefficients. 32 different QP values are used. They are arranged so that there is an increase of step size of about 12% from one QP to the next. Increase of QP by 6 means that the step size is about doubled. There is no 'dead zone' in the quantization process and the total range of step size from smallest to largest is about the same as for H.263.

The QP signalled in the bitstream applies for luma quantization/dequantization. This could be called QPluma. For chroma quantization/dequantization a different value - QPchroma - is used. The relation between the two is:

QPluma 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

QPchroma 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 17 18 19 20 20 21 22 22 23 23 24 24 25 25

When QP is used in the following we mean QPluma or QPchroma depending on what is appropriate.

Two arrays of numbers are used for quantization/dequantization.

A(QP=0,..,31)
620,553,492,439,391,348,310,276,246,219,195,174,155,138,123,110,98,87,78,69,62,55,49,44,39,35,31,
27,24,22,19,17

B(QP=0,..,31)
3881,4351,4890,5481,6154,6914,7761,8718,9781,10987,12339,13828,15523,17435,19561,21873,24552,
27656,30847,34870,38807,43747,49103,54683,61694,68745,77615,89113,100253,109366,126635,
141533

The relation between A() and B() is: A(QP)xB(QP)x6762 = 240.

It is assumed that a coefficient K is quantized in the following way:

LEVEL = (KxA(QP) + fx220)/220 |f| is in the range (0-0.5) and f has the same sign as K.

Dequantization:

K' = LEVELxB(QP)

After inverse transform this results in pixel values that are 220 too high. A shift of 20 bits (with rounding) is therefore needed on the reconstruction side. The definition of transform and quantization is designed so that no overflow will occur with use of 32 bit arithmetic.

~~5.3.4~~4.3.4 Scanning and quantization of 2x2 chroma DC coefficients

DDC() is quantized (and dequantized) separately resulting in LEVEL, RUN and EOB. The scanning order is: DCC(0,0), DCC(1,0), DCC(0,1), DCC(1,1). Inverse 2x2 transform as defined above is then performed after dequantization resulting in dequantized 4x4 DC coefficients: DC0' DC1' DC2' DC3'.

Chroma AC coefficients (4x4 based) are then quantized similarly to before. Notice that there are only 15 AC coefficients. The maximum size of RUN is therefore 14. However, for simplicity we use the same relation between LEVEL, RUN and Code no. As defined for 'Simple scan' in **Error! Reference source not found.**.

~~5.4~~4.4 Use of 2-dimensional model for coefficient coding.

In the 3D model for coefficient coding (see H.263) there is no good use of a short codeword of 1 bit. On the other hand, with the use of 2D VLC plus End Of Block (EOB) (as used in H.261, H.262) and with the small block size, 1 bit for EOB is usually well matched to the VLC.

Furthermore, with the fewer non-zero coefficients per block, the advantage of using 3D VLC is reduced.

As a result we use a 2D model plus End Of Block (EOB) in the present model. This means that an event to be coded (RUN, LEVEL) consists of:

RUN	which is the number of zero coefficients since the last nonzero coefficient.
LEVEL	the size of the nonzero coefficient
EOB	signals that there are no more nonzero coefficients in the block

~~5.5~~4.5 Deblocking Filter

After the reconstruction of a macroblock a conditional filtering of this macroblock takes place, that effects the *boundaries* of the 4x4 block structure. For this purpose each 4x4 luminance block in a reconstructed macroblock is assigned a filtering **Strength**, which has the following value:

TABLE 2

Assignment of filtering strength.

4x4 block condition	Strength
Macro block is INTRA-coded, or member of SP-frame	2
Macro block is non INTRA, but has nonzero coefficients	1
Else	0

Filtering across 4x4 block boundary takes place if:

- Strength ≠ 0 on at least one of the sides of the boundary, or
- the absolute difference between one of the motion vector components of the two adjacent blocks is at least one integer pixel (four ¼ pixels). For macro blocks in B-frames which are predicted "bidirectional" or "direct", this condition has to be valid for at least one of the involved vectors, or
- the adjacent motion vectors refer to different reference frames

Filtering takes place on a macroblock level. In a first step the 16 pel of the 4 vertical edges (horizontal filtering) of the 4x4 raster are filtered. Than the 4 horizontal edges (vertical filtering) follow. Note, that this process also affects the boundaries of the already reconstructed macroblocks above and to the right of the current macroblock.

Block boundaries of chroma blocks always correspond to a block boundary of luma blocks. Therefore corresponding **Strength** and **motion vector difference** for luma are also used for chroma boundaries. That is, for every 4x4 luma block two 2x2 chroma blocks are processed.

~~5.5.1~~4.5.1 Picture Content Dependant Parameters for Filtering

The set of eight pixels across a 4x4 block horizontal or vertical boundary is denoted as

$p_4,p_3,p_2,p_1 \mid q_1,q_2,q_3,q_4$

FIGURE 14

Illustration of de-blocking filtering.

with the actual boundary between p_1 and q_1. Up to two pixels can be updated as a result of the filtering process on both sides of the boundary (that is at most p_2, p_1, q_1, q_2). The number of updated pixels on both sides of the boundary depends on corresponding activity parameters a_p and a_q. They are limited by the so called overall activity parameter *n*:

$n = \min(3, 4 - |p_1-q_1| * \alpha(QP) / 128)$

Activity parameter a_p is equal to the smallest integer $i>0$ not larger than n for which the inequality $|p_1-p_i+1| \geq \beta(QP)$ holds. If no $i < n$ satisfies the condition, a_p is set to n. a_q is obtained correspondingly. For the quantizer dependant threshold parameters α and β see table 1 below.

TABLE 3

QP dependent threshold parameters α and β.

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α	128	128	128	128	128	128	128	128	128	128	122	96	75	59	47	37
β	0	0	0	0	0	0	0	0	3	3	3	4	4	4	6	6
QP	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
α	29	23	18	15	13	11	9	8	7	6	5	4	3	3	2	2
β	6	7	8	8	9	9	10	10	11	11	12	12	13	13	14	14

~~5.5.2~~4.5.2 Filtering Process

There are 3 conditions, that determine which of the boundary pixels are filtered:

- If $(a_p > 1)$ and $(a_q > 1)$ the pixels p_1 and q_1 are filtered by:

$$\Delta = \text{Clip}(-c, c, ((q_1 - p_1) << 2 + (p_2 - q_2) + 4) >> 3)$$
$$P_1 = \text{Clip}(0, 255, (p_1+\Delta))$$
$$Q_1 = \text{Clip}(0, 255, (q_1-\Delta))$$

- If $(a_p == 3)$ p_2 is filtered by:

$$\Delta = \text{Clip}(-c_p, c_p, (p_3+p_1 - p_2<<1) >> 1)$$
$$P_2 = \text{Clip}(0, 255, (p_2+\Delta))$$

- If $(a_q == 3)$ q_2 is filtered by:

$$\Delta = \text{Clip}(-c_q, c_q, (q_3+q_1 - q_2<<1) >> 1)$$
$$Q_2 = \text{Clip}(0, 255, (q_2+\Delta))$$

Here Clip() denotes a clipping function with the parameters Clip(Min, Max, Value) with the clipping constants:

$$c_p = \text{ClipTbl}(QP, \text{Strength}_p)$$
$$c_q = \text{ClipTbl}(QP, \text{Strength}_q)$$
$$c = (c_p + c_p + a_p + a_q) / 2$$

TABLE 4

ClipTbl (QP, Strength):

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
ClipIbb(qp,0)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ClipTbl(qp,1)	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	5	5	
ClipTbl(qp,2)	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5	5	5	7	8	9	

~~4.4.1~~4.5.3 Stronger filtering for intra coded macroblocks

In case of intra coding and areas in the picture with little texture the subjective quality can be improved by stronger filtering across macroblock boundaries (that is: *not* 4x4 block boundaries). This stronger filtering is performed only if one or both of the adjacent macroblocks are intra coded and $a_p = 3$ and $a_q = 3$ and $1 < |p_1-q_1| < QP / 4$.

In this case luminance filtering is performed using the formulas shown below. For chrominance pixels p_3 to q_3, are not filtered.

$$P_1 = (25*p_3 + 26*p_2 + 26*p_1 + 26*q_1 + 25*q_2 + 64) / 128$$
$$P_2 = (25*p_4 + 26*p_3 + 26*p_2 + 26*P_1 + 25*q_1 + 64) / 128$$
$$P_3 = (26*p_4 + 51*p_3 + 26*P_2 + 25*P_1 + 64) / 128$$
$$Q_1 = (25*p_2 + 26*p_1 + 26*q_1 + 26*q_2 + 25*q_3 + 64) / 128$$
$$Q_2 = (25*p_1 + 26*Q_1 + 26*q_2 + 26*q_3 + 25*q_4 + 64) / 128$$
$$Q_3 = (25*Q_1 + 26*Q_2 + 51*q_3 + 26*q_4 + 64) / 128$$

Note, that when filtering across the next 4x4 block boundary, pixel q_3 may get overwritten. To compensate the effect Q_3 is stored and used as P_2 when filtering across the next 4x4 block boundary.

~~6~~5 Entropy Coding

~~6.1~~5.1 Universal Variable Length Coding (UVLC)

In the default entropy coding mode, a universal VLC is used to code all syntax. The table of codewords may be written in the following compressed form.

$$
\begin{array}{c}
1 \\
0\ x_0\ 1 \\
0\ x_1\ 0\ x_0\ 1 \\
0\ x_2\ 0\ x_1\ 0\ x_0\ 1 \\
0\ x_3\ 0\ x_2\ 0\ x_1\ 0\ x_0\ 1 \\
\ldots\ldots\ldots\ldots\ldots\ldots
\end{array}
$$

where x_n take values 0 or 1. We will sometimes refer to a codeword with its length in bits (L) and INFO = x_n .. x_1 x_0 . Notice that the number of bits in INFO is L/2 (division by truncation). The codewords are numbered from 0 and upwards. The definition of the numbering is:

Code_number = $2^{L/2}$ + INFO -1 (L/2 use division with truncation. INFO = 0 when L = 1)

Some of the first code numbers and codewords are written explicitly in the table below. As an example, for the code number 5, L = 5 and INFO = 10 (binary) = 2 (decimal)

Code number	Codewords in explicit form
0	1
1	0 0 1
2	0 1 1
3	0 0 0 0 1
4	0 0 0 1 1
5	0 1 0 0 1
6	0 1 0 1 1
7	0 0 0 0 0 0 1
8	0 0 0 0 0 1 1
9	0 0 0 1 0 0 1
10	0 0 0 1 0 1 1
11	0 1 0 0 0 0 1
..	. . . .

When L and INFO is known, the regular structure of the table makes it easy to create a codeword bit by bit. Similarly, a decoder may easily read bit by bit until the last "1" which gives the end of the codeword. L and INFO is then readily available. For each parameter to be coded, there is a conversion rule from the parameter value to the code number (or L and INFO). TABLE 5 lists the connection between code number and most of the parameters used in the present coding method.

TABLE 5

Connection between codeword number and parameter values.

Code_ number	RUN	MB_Type		Intra_pred_mode[1]		MVD DQUANT	CBP		Tcoeff_chroma_DC[2]		Tcoeff_chroma_AC[2] Tcoeff_luma[2] Simple scan		Tcoeff_luma[2] Double scan	
		Intra	Inter	Prob0	Prob1		Intra	Inter	Level	Run	Level	Run	Level	Run
0	0	Intra4x4	16x16	0	0	0	47	0	EOB	-	EOB	-	EOB	-
1	1	0,0,0[3]	16x8	1	0	1	31	16	1	0	1	0	1	0
2	2	1,0,0	8x16	0	1	-1	15	1	-1	0	-1	0	-1	0
3	3	2,0,0	8x8	0	2	2	0	2	2	0	1	1	1	1
4	4	3,0,0	8x4	1	1	-2	23	4	-2	0	-1	1	-1	1
5	5	0,1,0	4x8	2	0	3	27	8	1	1	1	2	2	0
6	6	1,1,0	4x4	3	0	-3	29	32	-1	1	-1	2	-2	0
7	7	2,1,0	Intra4x4	2	1	4	30	3	3	0	2	0	1	2
8	8	3,1,0	0,0,0[3]	1	2	-4	7	5	-3	0	-2	0	-1	2
9	9	0,2,0	1,0,0	0	3	5	11	10	2	1	1	3	3	0
10	10	1,2,0	2,0,0	0	4	-5	13	12	-2	1	-1	3	-3	0
11	11	2,2,0	3,0,0	1	3	6	14	15	1	2	1	4	4	0
12	12	3,2,0	0,1,0	2	2	-6	39	47	-1	2	-1	4	-4	0
13	13	0,0,1	1,1,0	3	1	7	43	7	1	3	1	5	5	0
14	14	1,0,1	2,1,0	4	0	-7	45	11	-1	3	-1	5	-5	0
15	15	2,0,1	3,1,0	5	0	8	46	13	4	0	3	0	1	3
16	16	3,0,1	0,2,0	4	1	-8	16	14	-4	0	-3	0	-1	3
17	17	0,1,1	1,2,0	3	2	9	3	6	3	1	2	1	1	4
18	18	1,1,1	2,2,0	2	3	-9	51	9	-3	1	-2	1	-1	4
19	19	2,1,1	3,2,0	1	4	10	10	31	2	2	2	2	2	1
20	20	3,1,1	0,0,1	0	5	-10	12	35	-2	2	-2	2	-2	1
21	21	0,2,1	1,0,1	1	5	11	19	37	2	3	1	6	3	1
22	22	1,2,1	2,0,1	2	4	-11	21	42	-2	3	-1	6	-3	1
23	23	2,2,1	3,0,1	3	3	12	26	44	5	0	1	7	6	0
24	24	3,2,1	0,1,1	4	2	-12	28	33	-5	0	-1	7	-6	0
25	25		1,1,1	5	1	13	35	34	4	1	1	8	7	0
26	26		2,1,1	5	2	-13	37	36	-4	1	-1	8	-7	0
27	27		3,1,1	4	3	14	42	40	3	2	1	9	8	0
28	28		0,2,1	3	4	-14	44	39	-3	2	-1	9	-8	0
29	29		1,2,1	2	5	15	1	43	3	3	4	0	9	0
30	30		2,2,1	3	5	-15	2	45	-3	3	-4	0	-9	0
31	31		3,2,1	4	4	16	4	46	6	0	5	0	10	0
32	32			5	3	-16	8	17	-6	0	-5	0	-10	0
33	33			5	4	17	17	18	5	1	3	1	4	1
34	34			4	5	-17	18	20	-5	1	-3	1	-4	1
35	35			5	5	18	20	24	4	2	3	2	2	2
36	36					-18	24	19	-4	2	-3	2	-2	2
37	37					19	6	21	4	3	2	3	2	3
38	38					-19	9	26	-4	3	-2	3	-2	3
39	39					20	22	28	7	0	2	4	2	4
40	40					-20	25	23	-7	0	-2	4	-2	4
41	41					21	32	27	6	1	2	5	2	5
42	42					-21	33	29	-6	1	-2	5	-2	5
43	43					22	34	30	5	2	2	6	2	6
44	44					-22	36	22	-5	2	-2	6	-2	6
45	45					23	40	25	5	3	2	7	2	7
46	46					-23	38	38	-5	3	-2	7	-2	7
47	47					24	41	41	8	0	2	8	11	0
	..					..			..	..	..	..	..	..

[1] Prob0 and Prob1 defines the Intra prediction modes of two blocks relative to the prediction of prediction modes (see details in the section for Intra coding).

[2] For the entries above the horizontal line, the table is needed for relation between code number and Level/Run/EOB. For the remaining Level/Run combination there is a simple rule. The Level/Run combinations are assigned a code number according to the following priority: 1) sign of Level (+ -) 2) Run (ascending) 3) absolute value of Level (ascending).

[3] 16x16 based intra mode. The 3 numbers refer to values for (Imode,AC,nc) - see 3.5.9.3.

~~6.2~~5.2 Context-based Adaptive Binary Arithmetic Coding (CABAC)

~~6.2.1~~5.2.1 Overview

The entropy coding method of context-based adaptive binary arithmetic coding (CABAC) has three distinct elements compared to the default entropy coding method using a fixed, universal table of variable length codes (UVLC):

- *Context modeling* provides estimates of conditional probabilities of the coding symbols. Utilizing suitable context models, given inter-symbol redundancy can be exploited by switching between different probability models according to already coded symbols in the neighborhood of the current symbol to encode.
- *Arithmetic codes* permit non-integer number of bits to be assigned to each symbol of the alphabet. Thus the symbols can be coded almost at their entropy rate. This is extremely beneficial for symbol probabilities much greater than 0.5, which often occur with efficient context modeling. In this case, a variable length code has to spend at least one bit in contrast to arithmetic codes, which may use a fraction of one bit.
- *Adaptive* arithmetic codes permit the entropy coder to adapt itself to non-stationary symbol statistics. For instance, the statistics of motion vector magnitudes vary over space and time as well as for different sequences and bit-rates. Hence, an adaptive model taking into account the cumulative probabilities of already coded motion vectors leads to a better fit of the arithmetic codes to the current symbol statistics.

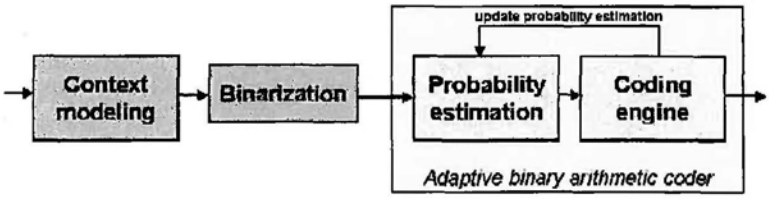


FIGURE 15

Generic block diagram of CABAC entropy coding scheme

Next we give a short overview of the main coding elements of the CABAC entropy coding scheme as depicted in FIGURE 15. Suppose a symbol related to an arbitrary syntax element is given, then, in a first step, a suitable model is chosen according to a set of past observations. This process of constructing a model conditioned on neighboring symbols is commonly referred to as *context modeling* and is the first step in the entropy coding scheme. The particular context models that are designed for each given syntax model are described in detail in Section 5.2.2 and Section 5.2.3. If a given symbol is non-binary valued, it will be mapped onto a sequence of binary decisions, so-called *bins*, in a second step. The actual *binarization* is done according to a given binary tree, as specified in Section 5.2.4. Finally, each binary decision is encoded with the *adaptive binary arithmetic coding* (AC) engine using the *probability estimates*, which have been provided either by the context modeling stage or by the binarization process itself. The provided models serve as a probability estimation of the related bins. After encoding of each bin, the related model will be updated with the encoded binary symbol. Hence, the model keeps track of the actual statistics.

~~6.2.2~~5.2.2 Context Modeling for Coding of Motion and Mode Information

In this section we describe in detail the context modeling of our adaptive coding method for the syntax elements macroblock type (MB_type), motion vector data (MVD) and reference frame parameter (Ref_frame).

~~6.2.2.1~~5.2.2.1 Context Models for Macroblock Type

We distinguish between MB_type for intra and inter frames. In the following, we give a description of the context models which have been designed for coding of the MB_type information in both cases. The subsequent process of mapping a non-binary valued MB_type symbol to a binary sequence in the case of inter frames will be given in detail in section 5.2.4.

~~6.2.2.1.1~~5.2.2.1.1 Intra Pictures

For intra pictures, there are two possible modes for each macroblock, i.e. Intra4x4 and Intra16x16, so that signalling the mode information is reduced to transmitting a binary decision. Coding of this binary decision for a given macroblock is performed by means of context-based arithmetic coding, where the context of a current MB_type C is build by using the MB_types A and B1 of neighboring macroblocks (as depicted in FIGURE 16) which are located in the causal past of the current coding event C. Since A and B are binary decisions, we define the actual context number *ctx_mb_type_intra(C)* of C by *ctx_mb_type_intra(C) = A + 2*B*, which results in four different contexts according to the 4 possible combinations of MB_type states for A and B.

In the case of MB_type Intra16x16, there are three additional parameters related to the chosen intra prediction mode, the occurrence of significant AC-coefficients and the coded block pattern for the chrominance coefficients, which have to be signaled. In contrast to the current test model, this information is not included in the mode information, but is coded separately by using distinct models as described in Section 5.2.3.

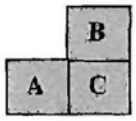


FIGURE 16

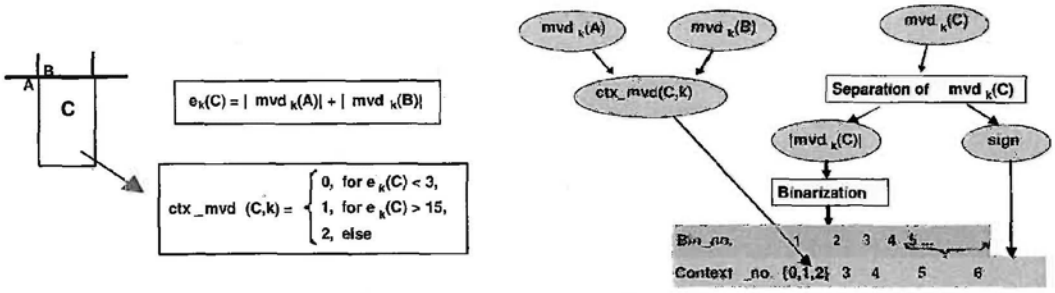
Neighboring symbols A and B used for conditional coding of a current symbol C.

~~6.2.2.1.2~~5.2.2.1.2 P- and B-Pictures

Currently there a 10 different macroblock types for P-frames and 18 different macroblock types for B-frames, provided that the additional information of the 16x16 Intra mode is not considered as part of the mode information. Coding of a given MB_type information C is done similar to the case of intra frames by using a context model which involves the MB_type information A and B of previously encoded (or decoded) macroblocks (cp. FIGURE 16). However, here we only use the information whether the neighboring macroblocks of the given macroblock are of type Skip (P-frame) or Direct (B-frame), such that the actual context number *ctx_mb_type_inter(C)* is given in C-style notation for P-frame coding by *ctx_mb_type_inter(C) = ((A==Skip)?0:1) + 2*((B==Skip)?0:1)* and by *ctx_mb_type_inter(C) = ((A==Direct)?0:1) + 2*((B==Direct)?0:1)* for B-frame coding. Thus, we obtain 4 different contexts, which, however, are only used for coding of the first bin of the binarization ***b(C)*** of C, where the actual binarization of C will be performed as outlined in Section 5.2.4. For coding the second bin, a separate model is provided and for all remaining bins of ***b(C)*** two additional models are used as further explained in Sect. 5.2.3. Thus, a total number of 7 different models are supplied for coding of macroblock type information relating to P-and B-frames.

---

1 For mathematical convenience the meaning of the variables A, B and C is context dependent.



(a) (b)

FIGURE 17

Illustration of the encoding process for a given residual motion vector component mvdk(C) of a block C: (a) Context selection rule. (b) Separation of mvdk(C) into sign and magnitude, binarization of the magnitude and assignment of context models to bin_nos.

~~6.2.2.2~~5.2.2.2 Context Models for Motion Vector Data

Motion vector data consists of residual vectors obtained by applying motion vector prediction. Thus, it is a reasonable approach to build a model conditioned on the local prediction error. A simple measure of the local prediction error at a given block C is given by evaluating the L1-norm $e_k(A,B) = |mvd_k(A)| + |mvd_k(B)|$ of two neighboring motion vector prediction residues $mvd_k(A)$ and $mvd_k(B)$ for each component of a motion vector residue $mvd_k(C)$ of a given block, where A and B are neighboring blocks of block C, as shown in FIGURE 17 (a). If one of the neighboring blocks belongs to an adjacent macroblock, we take the residual vector component of the leftmost neighboring block in the case of the upper block B, and in the case of the left neighboring block A we use the topmost neighboring block. If one of the neighboring blocks is not available, because, for instance, the current block is at the picture boundary, we discard the corresponding part of e_k. By using e_k, we now define a context model *ctx_mvd(C,k)* for the residual motion vector component $mvd_k(C)$ consisting of three different context models:

$$ctx\_mvd(C,k) = \begin{cases} 0, e_k(C) < 3, \\ 1, e_k(C) > 15, \\ 2, else. \end{cases}$$

For the actual coding process, we separate $mvd_k(C)$ in sign and modulus, where only the first bin of the binarization of the modulus $|mvd_k(C)|$ is coded using the context models *ctx_mvd(C,k)*. For the remaining bins, we have three additional models: two for the second and the third bin and a third model for all remaining bins. In addition, the sign coding routine is provided with a separate model. This results in a total sum of 7 different models for each vector component (see FIGURE 17).

In the case of B-frame coding, an additional syntax element has to be signaled when the bi-directional mode is chosen. This element represents the block size (Blk_size), which is chosen for forward or backward motion prediction. The related code number value ranges between 0 and 6 according to the 7 possible block shapes in FIGURE 2. Coding of Blk_size is done by using the binarization of the P_MB_type as described in Section 5.2.3.

~~6.2.2.3~~5.2.2.3 Context Models for Reference Frame Parameter

If the option of temporal prediction from more than one reference frame is enabled, the chosen reference frame for each macroblock must be signaled. Given a macroblock and its reference frame parameter as a symbol C according to the definition in Sect. 3, a context model is built by using symbols A and B of the reference frame parameter belonging to the two neighboring macroblocks (cp. FIGURE 16). The actual context number of C is then defined by *ctx_ref_frame(C) = ((A==0)?0:1) + 2*((B==0)?0:1)*, such that *ctx_ref_frame(C)* indicates one of four models used for coding of the first bin of the binary equivalent

b(C) of C. Two additional models are given for the second bin and all remaining bins of *b(C)*, which sums up to a total number of six different models for the reference frame information.

~~6.2.3~~5.2.3 Context Modeling for Coding of Texture Information

This section provides detailed information about the context models used for the syntax elements of coded block pattern (CBP), intra prediction mode (IPRED) and (RUN, LEVEL) information.

~~6.2.3.1~~5.2.3.1 Context Models for Coded Block Pattern

Except for MB_type Intra16x16, the context modeling for the coded block pattern is treated as follows. There are 4 luminance CBP bits belonging to 4 8x8 blocks in a given macroblock. Let C denote such a Y-CBP bit, then we define *ctx_cbp_luma(C) = A + 2*B*, where A and B are Y-CBP bits of the neighboring 8x8 blocks, as depicted in FIGURE 16. The remaining 2 bits of CBP are related to the chrominance coefficients. In our coding approach, these bits are translated into two dependant binary decisions, such that, in a first step, we send a bit *cbp_chroma_sig* which signals whether there are significant chrominance coefficients at all. The related context model is of the same kind as that of the Y-CBP bits, i.e. *ctx_cbp_chroma_sig(C) = A + 2*B*, where A and B are now notations for the corresponding *cbp_chroma_sig* bits of neighboring macroblocks. If *cbp_chroma_sig = 1* (non-zero chroma coefficients exist), a second bit *cbp_chroma_ac* related to the significance of AC chrominance coefficients has to be signalled. This is done by using a context model conditioned on the *cbp_chroma_ac* decisions A and B of neighboring macroblocks, such that *ctx_cbp_chroma_AC(C) = A + 2*B*. Note, that due to the different statistics there are different models for Intra and Inter macroblocks, so that the total number of different models for CBP amounts to 2*3*4=24. For the case of MB_type Intra16x16, there are three additional models, one for the binary AC decision and two models for each of the two chrominance CBP bits.

~~6.2.3.2~~5.2.3.2 Context Models for Intra Prediction Mode

In Intra4x4 mode, coding of the intra prediction mode C of a given block is conditioned on the intra prediction mode of the previous block A to the left of C (cp. FIGURE 16). In fact, it is not the prediction mode number itself which is signaled and which is used for conditioning but rather its predicted order similar as it is desribed in Sect. 3.5.8. There are 6 different prediction modes and for each mode, two different models are supplied: one for the first bin of the binary equivalent of C and the other for all remaining bins. Together with two additional models for the two bits of the prediction modes of MB_type Intra16x16 (in binary representation), a total number of 14 different models for coding of intra prediction modes is given.

~~6.2.3.3~~5.2.3.3 Context Models for Run/Level

Coding of (RUN, LEVEL) pairs is conditioned on the scanning mode, the DC/AC block type, the luminance/chrominance, and the intra/inter macroblock decision. Thus, a total number of 9 different block types are given according to TABLE 6, which, in turn, results in 9 different contexts. In contrast to the current test model, RUN and LEVEL are coded separately in our coding approach, as described in the following two subsections.

TABLE 6

Numbering of the different context models used for coding of RUN and LEVEL

Ctx_run_level	Block Type
0	Double Scan
1	Single Scan, Inter
2	Single Scan, Intra
3	Intra16x16, DC
4	Intra16x16, AC
5	Chroma, DC, Inter

6	Chroma, DC, Intra
7	Chroma, AC, Inter
8	Chroma, AC, Intra

~~6.2.3.3.1~~5.2.3.3.1 Context-based Coding of LEVEL Information

For a given block C, the LEVEL information is first separated into sign and magnitude. According to its context *ctx_run_level(C)* four different models are chosen, where one model is used for coding of the sign information and the remaining 3 models are used for the first, the second and all remaining bins of the binarization of LEVEL. If LEVEL≠0 (EOB), the corresponding RUN is coded in a subsequent coding routine, as described in the following section.

~~6.2.3.3.2~~5.2.3.3.2 Context-based Coding of RUN Information

For each context *ctx_run_level(C)* two separate models are provided for the coding of RUN; one model for the first bin and the second model for all remaining bins of the binary sequence related to RUN.

TABLE 7

Binarization by means of the unary code tree

Code symbol	Binarization						
0	1						
1	0	1					
2	0	0	1				
3	0	0	0	1			
4	0	0	0	0	1		
5	0	0	0	0	0	1	
6	0	0	0	0	0	0	1
...	.	.	.	.	.	.	.
Bin_no.	1	2	3	4	5	6	7

~~6.2.4~~5.2.4 **Binarization of Non-Binary Valued Symbols**

A non-binary valued symbol will be decomposed into a sequence of binary decisions. Except for the MB_type syntax element we use the binarization given by the unary code tree in TABLE 7.

TABLE 8

(a) Binarization for P-frame MB_type and (b) for B-frame MB_type

P_MB_type	Binarization
0	0
1	100
2	101
3	11000
4	11001
5	11010
6	11011
7	11100
8	11101
9	11110
Bin_no	12345

B_MB_type	Binarization
0	0
1	100
2	101
3	11000
4	11001
5	11010
6	11011
7	1110000
.	.
17	1111010
Bin_no	1234567

For the binary decomposition of the MB_type symbols of P- or B-frames, which are of limited range (0 ... 9) or (0 ... 17) respectively, an alternative binarization is used, which is shown in TABLE 8.

~~6.2.5~~5.2.5 Adaptive Binary Arithmetic Coding

At the beginning of the overall encoding of a given frame the probability models associated with all 126 different contexts are initialized with a pre-computed start distribution. For each symbol to encode the frequency count of the related binary decision is updated, thus providing a new probability estimate for the next coding decision. However, when the total number of occurrences of a given model exceeds a pre-defined threshold, the frequency counts will be scaled down. This periodical rescaling exponentially weighs down past observations and helps to adapt to the non-stationarity of a source. The binary arithmetic coding engine used in our presented approach is a straightforward implementation similar to that given in Witten et al., "Arithmetic Coding for Data Compression", *Comm. of the ACM*, 30 (6), 1987, pp.520-541.

~~7~~6 Data Partitioning and Interim File Format

The traditional TML bit stream syntax simply concatenates the VLC coded symbols to form a bit stream, with the exception of the possible byte alignment of the picture (and slice?) header. Data Partitioning re-arranges the symbols in such a way that all symbols of one data type (e.g. DC coefficients, macroblock headers, motion vectors) that belong to a single slice are collected in one VLC coded bitstream that starts byte aligned. Decoders can process such a partitioned data streams by fetching symbols from the correct patition. The partition to fetch from is determined through the decoder's state machine, according to the syntax diagram discussed in section 2.4

In order to cleanly divide the concept of data partitioning (which is useful in all error prone environments) from network specifica, an interim file format is used that stores the compressed, VLC coded bits of each partition along with easy-to-process, uncompressed header information. It should be emphasized that this file format is not intended to be transmitted – it rather forms a clean interface between network specific and network unspecific syntax elements. Adaptation layers for IP and highly bit error prone H.223-based networks are discussed later in this document.

~~7.1~~6.1 Data Partitioning

Data Partitioning is implemented by concatenating all VLC coded symbols of one data type and one slice (or full picture if slices are not used). At the moment, for a few partitions as indicated below contain data of more than one data type that are so closely related that a finer diversion seems to be fruitless. The following data types are currently defined:

0	TYPE_HEADER	Picture or Slice Headers (Note 1)
1	TYPE_MBHEADER	Macroblock header information (Note 2)
2	TYPE_MVD	Motion Vector Data
3	TYPE_CBP	Coded Block Patterm
4	TYPE_2x2DC	2x2 DC Coefficients
5	TYPE_COEFF_Y	Luma AC Coefficients
6	TYPE_COEFF_C	Chroma AC Coefficients
7	TYPE_EOS	End-of-Stream Symbol

Note 1: TYPE_HEADER encompasses all Picture/Slice header information

Note 2: TYPE_MBHEADER encompasses The MB-Type, Intra-Prediction mode and Reference Frame ID.

~~8~~7 Multi-Frame Buffering Syntax

[NB: This chapter needs work and is currently being refined.]

The syntax is altered in slice layer. The following fields are coded using UVLC codewords.

~~8.1~~7.1 Reference Picture Selection Flags (RPSF)

RPSF indicates which type of back-channel messages are needed by the encoder.

~~8.2~~7.2 Picture Number (PN)

PN shall be incremented by 1 for each coded and transmitted picture, in modulo MAX_PN operation, relative to the PN of the previous stored picture. For non-stored pictures, PN shall be incremented from the value in the most temporally-recent stored picture which precedes the non-stored picture in bitstream order.

The PN serves as a unique ID for each picture stored in the multi-picture buffer within MAX_PN coded and stored pictures. Therefore, a picture cannot be kept in the buffer after more than MAX_PN-1 subsequent coded and stored pictures unless it has been assigned a long-term picture index as specified below. The encoder shall ensure that the bitstream shall not specify retaining any short-term picture after more than MAX_PN-1 subsequent stored pictures. A decoder which encounters a picture number on a current picture having a value equal to the picture number of some other short-term stored picture in the multi-picture buffer should treat this condition as an error.

~~8.3~~7.3 Reference Picture Selection Layer (RPSL)

RPSL can be signaled with the following values:

Code number 0: The RPS layer is not sent,

Code number 1: The RPS layer is sent.

If RPSL is not sent, the default buffer indexing order presented in the next subsection shall be applied. RPS layer information sent at the slice level does not affect the decoding process of any other slice.

If RPSL is sent, the buffer indexing used to decode the current slice and to manage the contents of the picture buffer is sent using the following code words.

~~8.4~~7.4 Re-Mapping of Picture Numbers Indicator (RMPNI)

RMPNI is present in the RPS layer if the picture is a P, or B picture. RMPNI indicates whether any default picture indices are to be re-mapped for motion compensation of the current slice – and how the re-mapping of the relative indices into the multi-picture buffer is to be specified if indicated. If RMPNI indicates the presence of an ADPN or LPIR field, an additional RMPNI field immediately follows the ADPN or LPIR field.

A picture reference parameter is a relative index into the ordered set of pictures. The RMPNI, ADPN, and LPIR fields allow the order of that relative indexing into the multi-picture buffer to be temporarily altered from the default index order for the decoding of a particular slice. The default index order is for the short-term pictures (i.e., pictures which have not been given a long-term index) to precede the long-term pictures in the reference indexing order. Within the set of short-term pictures, the default order is for the pictures to be ordered starting with the most recent buffered reference picture and proceeding through to the oldest reference picture (i.e., in decreasing order of picture number in the absence of wrapping of the ten-bit picture number field). Within the set of long-term pictures, the default order is for the pictures to be ordered starting with the picture with the smallest long-term index and proceeding up to the picture with long-term index equal to the most recent value of MLIP1−1.

For example, if the buffer contains three short-term pictures with short-term picture numbers 300, 302, and 303 (which were transmitted in increasing picture-number order) and two long-term pictures with long-term picture indices 0 and 3, the default index order is:

default relative index 0 refers to the short-term picture with picture number 303,

default relative index 1 refers to the short-term picture with picture number 302,

default relative index 2 refers to the short-term picture with picture number 300,

default relative index 3 refers to the long-term picture with long-term picture index 0, and

default relative index 4 refers to the long-term picture with long-term picture index 3.

The first ADPN or LPIR field that is received (if any) moves a specified picture out of the default order to the relative index of zero. The second such field moves a specified picture to the relative index of one, etc. The set of remaining pictures not moved to the front of the relative indexing order in this manner shall retain their default order amongst themselves and shall follow the pictures that have been moved to the front of the buffer in relative indexing order.

If there is not more than one reference picture used, no more than one ADPN or LPIR field shall be present in the same RPS layer unless the current picture is a B picture. If the current picture is a B picture and more than one reference picture is used, no more than two ADPN or LPIR fields shall be present in the same RPS layer.

Any re-mapping of picture numbers specified for some slice shall not affect the decoding process for any other slice.

In a P picture an RMPNI "end loop" indication is followed by RPBT.

Within one RPS layer, RMPNI shall not specify the placement of any individual reference picture into more than one re-mapped position in relative index order.

TABLE 9

RMPNI operations for re-mapping of reference pictures

Code Number	Re-mapping Specified
0	ADPN field is present and corresponds to a negative difference to add to a picture number prediction value
1	ADPN field is present and corresponds to a positive difference to add to a picture number prediction value
2	LPIR field is present and specifies the long-term index for a reference picture
3	End loop for re-mapping of picture relative indexing default order

~~8.4.17~~.4.1 Absolute Difference of Picture Numbers (ADPN)

ADPN is present only if indicated by RMPNI. ADPN follows RMPNI when present. The code number of the UVLC corresponds to ADPN – 1. ADPN represents the absolute difference between the picture number of the currently re-mapped picture and the prediction value for that picture number. If no previous ADPN fields have been sent within the current RPS layer, the prediction value shall be the picture number of the current picture. If some previous ADPN field has been sent, the prediction value shall be the picture number of the last picture that was re-mapped using ADPN.

If the picture number prediction is denoted PNP, and the picture number in question is denoted PNQ, the decoder shall determine PNQ from PNP and ADPN in a manner mathematically equivalent to the following:

```
if (RMPNI == '1') {   // a negative difference
  if (PNP - ADPN < 0)
    PNQ = PNP - ADPN + MAX_PN;
  else
    PNQ = PNP - ADPN;
}else{              // a positive difference
  if (PNP + ADPN > MAX_PN-1)
    PNQ  = PNP + ADPN - MAX_PN;
```

```
  else
    PNQ   = PNP + ADPN;
}
```

The encoder shall control RMPNI and ADPN such that the decoded value of ADPN shall not be greater than or equal to MAX_PN.

As an example implementation, the encoder may use the following process to determine values of ADPN and RMPNI to specify a re-mapped picture number in question, PNQ:

```
DELTA = PNQ - PNP;
if (DELTA < 0) {
  if (DELTA < -MAX_PN/2-1)
    MDELTA = DELTA + MAX_PN;
  else
    MDELTA = DELTA;
}else{
  if(DELTA > MAX_PN/2)
    MDELTA = DELTA - MAX_PN;
  else
    MDELTA = DELTA;
}

ADPN = abs(MDELTA);
```

where abs() indicates an absolute value operation. Note that the code number of the UVLC corresponds to the value of ADPN – 1, rather than the value of ADPN itself.

RMPNI would then be determined by the sign of MDELTA.

~~8.4.2~~7.4.2 Long-term Picture Index for Re-Mapping (LPIR)

LPIR is present only if indicated by RMPNI. LPIR follows RMPNI when present. LPIR is transmitted using UVLC codewords. It represents the long-term picture index to be re-mapped. The prediction value used by any subsequent ADPN re-mappings is not affected by LPIR.

~~8.5~~7.5 Reference Picture Buffering Type (RPBT)

RPBT specifies the buffering type of the currently decoded picture. It follows an RMPNI "end loop" indication when the picture is not an I picture. It is the first element of the ERPS layer if the picture is an I picture. The values for RPBT are defined as follows:

Code number 0: Sliding Window,

Code number 1: Adaptive Memory Control.

In the "Sliding Window" buffering type, the current decoded picture shall be added to the buffer with default relative index 0, and any marking of pictures as "unused" in the buffer is performed automatically in a first-in-first-out fashion among the set of short-term pictures. In this case, if the buffer has sufficient "unused" capacity to store the current picture, no additional pictures shall be marked as "unused" in the buffer. If the buffer does not have sufficient "unused" capacity to store the current picture, the picture with the largest default relative index among the short-term pictures in the buffer shall be marked as "unused". In the sliding window buffering type, no additional information is transmitted to control the buffer contents.

In the "Adaptive Memory Control" buffering type, the encoder explicitly specifies any addition to the buffer or marking of data as "unused" in the buffer, and may also assign long-term indices to short-term pictures. The current picture and other pictures may be explicitly marked as "unused" in the buffer, as specified by the encoder. This buffering type requires further information that is controlled by memory management control operation (MMCO) parameters.

~~8.5.17~~.5.1 Memory Management Control Operation (MMCO)

MMCO is present only when RPBT indicates "Adaptive Memory Control", and may occur multiple times if present. It specifies a control operation to be applied to manage the multi-picture buffer memory. The MMCO parameter is followed by data necessary for the operation specified by the value of MMCO, and then an additional MMCO parameter follows – until the MMCO value indicates the end of the list of such operations. MMCO commands do not affect the buffer contents or the decoding process for the decoding of the current picture – rather, they specify the necessary buffer status for the decoding of subsequent pictures in the bitstream. The values and control operations associated with MMCO are defined in Table 2.

If MMCO is Reset, all pictures in the multi-picture buffer (but not the current picture unless specified separately) shall be marked "unused" (including both short-term and long-term pictures). A Reset Operation shall be present in the first picture in the bitstream.

The picture height and width shall not change within the bitstream except within a picture containing a Reset MMCO command.

A "stored picture" does not contain an MMCO command in its RPS layer which marks that (entire) picture as "unused". If the current picture is not a stored picture, its RPS layer shall not contain any of the following types of MMCO commands:

- An Reset MMCO command,
- Any MMCO command which marks any other picture (other than the current picture) as "unused" that has not also been marked as "unused" in the ERPS layer of a prior stored picture,
- Any MMCO command which assigns a long-term index to a picture that has not also been assigned the same long-term index in the ERPS layer of a prior stored picture, or
- Any MMCO command which marks sub-picture areas of any picture as "unused" that have not also been marked as "unused" in the ERPS layer of a prior stored picture.

TABLE 10

Memory Management Control Operation (MMCO) Values

Code Number	Memory Management Control Operation	Associated Data Fields Following
0	End MMCO Loop	None (end of ERPS layer)
1	Mark a Short-Term Picture as "Unused"	DPN
2	Mark a Long-Term Picture as "Unused"	LPIN
3	Assign a Long-Term Index to a Picture	DPN and LPIN
4	Specify the Maximum Long-Term Picture Index	MLIP1
5	Reset	None

~~8.5.1.17~~.5.1.1 Difference of Picture Numbers (DPN)

DPN is present when indicated by MMCO. DPN follows MMCO if present. DPN is transmitted using UVLC codewords and is used to calculate the PN of a picture for a memory control operation. It is used in order to assign a long-term index to a picture, mark a short-term picture as "unused", or mark sub-picture areas of a short-term picture as "unused". If the current decoded picture number is PNC and the decoded UVLC code number is DPN, an operation mathematically equivalent to the following equations shall be used for calculation of PNQ, the specified picture number in question:

```
if (PNC - DPN < 0)
  PNQ = PNC - DPN + MAX_PN;
else
  PNQ = PNC - DPN;
```

Similarly, the encoder may compute the DPN value to encode using the following relation:

```
if (PNC - PNQ < 0)
  DPN = PNC - PNQ + MAX_PN;
else
  DPN = PNC - PNQ;
```

For example, if the decoded value of DPN is zero and MMCO indicates marking a short-term picture as "unused", the current decoded picture shall be marked as "unused" (thus indicating that the current picture is not a stored picture).

~~8.5.1.2~~7.5.1.2 Long-term Picture Index (LPIN)

LPIN is present when indicated by MMCO. LPIN specifies the long-term picture index of a picture. It follows DPN if the operation is to assign a long-term index to a picture. It follows MMCO if the operation is to mark a long-term picture as "unused" or to mark sub-picture areas of a long-term picture as "unused".

~~8.5.1.3~~7.5.1.3 Maximum Long-Term Picture Index Plus 1 (MLIP1)

MLIP1 is present if indicated by MMCO. MLIP1 follows MMCO if present. If present, MLIP1 is used to determine the maximum index allowed for long-term reference pictures (until receipt of another value of MLIP1). The decoder shall initially assume MLIP1 is "0" until some other value has been received. Upon receiving an MLIP1 parameter, the decoder shall consider all long-term pictures having indices greater than the decoded value of MLIP1 – 1 as "unused" for referencing by the decoding process for subsequent pictures. For all other pictures in the multi-picture buffer, no change of status shall be indicated by MLIP1.

~~8.6~~7.6 Multi-Picture Decoder Process

The decoder stores the reference pictures for inter-picture decoding in a multi-picture buffer. The decoder replicates the multi-picture buffer of the encoder according to the reference picture buffering type and any memory management control operations specified in the bitstream. The buffering scheme may also be operated when partially erroneous pictures are decoded.

Each transmitted and stored picture is assigned a Picture Number (PN) which is stored with the picture in the multi-picture buffer. PN represents a sequential picture counting identifier for stored pictures. PN is constrained, using modulo MAX_PN arithmetic operation. For the first transmitted picture, PN should be "0". For each and every other transmitted and stored picture, PN shall be increased by 1. If the difference (modulo MAX_PN) of the PNs of two consecutively received and stored pictures is not 1, the decoder should infer a loss of pictures or corruption of data. In such a case, a back-channel message indicating the loss of pictures may be sent to the encoder.

Besides the PN, each picture stored in the multi-picture buffer has an associated index, called the default relative index. When a picture is first added to the multi-picture buffer it is given default relative index 0 – unless it is assigned to a long-term index. The default relative indices of pictures in the multi-picture buffer are modified when pictures are added to or removed from the multi-picture buffer, or when short-term pictures are assigned to long-term indices.

The pictures stored in the multi-picture buffers can also be divided into two categories: long-term pictures and short-term pictures. A long-term picture can stay in the multi-picture buffer for a long time (more than MAX_PN-1 coded and stored picture intervals). The current picture is initially considered a short-term picture. Any short-term picture can be changed to a long-term picture by assigning it a long-term index according to information in the bitstream. The PN is the unique ID for all short-term pictures in the multi-picture buffer. When a short-term picture is changed to a long-term picture, it is also assigned a long-term picture index (LPIN). A long-term picture index is assigned to a picture by associating its PN to an LPIN. Once a long-term picture index has been assigned to a picture, the only potential subsequent use of the long-term picture's PN within the bitstream shall be in a repetition of the long-term index assignment. The PNs of the long-term pictures are unique within MAX_PN transmitted and stored pictures. Therefore, the PN of a long-term picture cannot be used for assignment of a long-term index after MAX_PN-1 transmitted subsequent stored pictures. LPIN becomes the unique ID for the life of a long-term picture.

PN (for a short-term picture) or LPIN (for a long-term picture) can be used to re-map the pictures into re-mapped relative indices for efficient reference picture addressing.

~~8.6.1~~7.6.1 Decoder Process for Short/Long-term Picture Management

The decoder may have both long-term pictures and short-term pictures in its multi-picture buffer. The MLIP1 field is used to indicate the maximum long-term picture index allowed in the buffer. If no prior value of MLIP1 has been sent, no long-term pictures shall be in use, i.e. MLIP1 shall initially have an implied value of "0". Upon receiving an MLIP1 parameter, a new MLIP1 shall take effect until another value of MLIP1 is received. Upon receiving a new MLIP1 parameter in the bitstream, all long-term pictures with associated long-term indices greater than or equal to MLIP1 shall be considered marked "unused". The frequency of transmitting MLIP1 is out of the scope of this Recommendation. However, the encoder should send an MLIP1 parameter upon receiving an error message, such as an Intra request message.

A short-term picture can be changed to a long-term picture by using an MMCO command with an associated DPN and LPIN. The short-term picture number is derived from DPN and the long-term picture index is LPIN. Upon receiving such an MMCO command, the decoder shall change the short-term picture with PN indicated by DPN to a long-term picture and shall assign it to the long-term index indicated by LPIN. If a long-term picture with the same long-term index already exists in the buffer, the previously-existing long-term picture shall be marked "unused". An encoder shall not assign a long-term index greater than MLIP1–1 to any picture. If LPIN is greater than MLIP1–1, this condition should be treated by the decoder as an error. For error resilience, the encoder may send the same long-term index assignment operation or MLIP1 specification message repeatedly. If the picture specified in a long-term assignment operation is already associated with the required LPIN, no action shall be taken by the decoder. An encoder shall not assign the same picture to more than one long term index value. If the picture specified in a long-term index assignment operation is already associated with a different long-term index, this condition should be treated as an error. An encoder shall only change a short-term picture to a long-term picture within MAX_PN transmitted consecutive stored pictures. In other words, a short-term picture shall not stay in the short-term buffer after more than MAX_PN-1 subsequent stored pictures have been transmitted. An encoder shall not assign a long-term index to a short-term picture that has been marked as "unused" by the decoding process prior to the first such assignment message in the bitstream. An encoder shall not assign a long-term index to a picture number that has not been sent.

~~8.6.2~~7.6.2 Decoder Process for Reference Picture Buffer Mapping

The decoder employs indices when referencing a picture for motion compensation on the macroblock layer. In pictures other than B pictures, these indices are the default relative indices of pictures in the multi-picture buffer when the fields ADPN and LPIR are not present in the current slice layer as applicable, and are re-mapped relative indices when these fields are present. In B pictures, the first one or two pictures (depending on BTPSM) in relative index order are used for backward prediction, and the forward picture reference parameters specify a relative index into the remaining pictures for use in forward prediction. (needs to be changed)

The indices of pictures in the multi-picture buffer can be re-mapped onto newly specified indices by transmitting the RMPNI, ADPN, and LPIR fields. RMPNI indicates whether ADPN or LPIR is present. If ADPN is present, RMPNI specifies the sign of the difference to be added to a picture number prediction value. The ADPN value corresponds to the absolute difference between the PN of the picture to be re-mapped and a prediction of that PN value. The first transmitted ADPN is computed as the absolute difference between the PN of the current picture and the PN of the picture to be re-mapped. The next transmitted ADPN field represents the difference between the PN of the previous picture that was re-mapped using ADPN and that of another picture to be re-mapped. The process continues until all necessary re-mapping is complete. The presence of re-mappings specified using LPIR does not affect the prediction value for subsequent re-mappings using ADPN. If RMPNI indicates the presence of an LPIR field, the re-mapped picture corresponds to a long-term picture with a long-term index of LPIR. If any pictures are not re-mapped to a specific order by RMPNI, these remaining pictures shall follow after any pictures having a re-mapped order in the indexing scheme, following the default order amongst these non-re-mapped pictures.

If the decoder detects a missing picture, it may invoke some concealment process, and may insert an error-concealed picture into the multi-picture buffer. Missing pictures can be identified if one or several picture numbers are missing or if a picture not stored in the multi-picture buffer is indicated in a transmitted ADPN or LPIR. Concealment may be conducted by copying the closest temporally preceding picture that is available in the multi-picture buffer into the position of the missing picture. The temporal order of the short-term pictures in the multi-picture buffer can be inferred from their default relative index order and PN fields. In addition or instead, the decoder may send a forced INTRA update signal to the encoder by external means (for example, Recommendation H.245), or the decoder may use external means or back-channel messages (for example, Recommendation H.245) to indicate the loss of pictures to the encoder. A concealed picture may be inserted into the multi-picture buffer when using the "Sliding Window" buffering type. If a missing picture is detected when decoding a Slice, the concealment may be applied to the picture as if the missing picture had been detected at the picture layer.

~~8.6.3~~7.6.3 Decoder Process for Multi-Picture Motion Compensation

Multi-picture motion compensation is applied if the use of more than one reference picture is indicated. For multi-picture motion compensation, the decoder chooses a reference picture as indicated using the reference frame fields on the macroblock layer. Once, the reference picture is specified, the decoding process for motion compensation proceeds as described in the TML document (TODO: correct reference).

~~8.6.4~~7.6.4 Decoder Process for Reference Picture Buffering

The buffering of the currently decoded picture can be specified using the reference picture buffering type (RPBT). The buffering may follow a first-in, first-out ("Sliding Window") mode. Alternatively, the buffering may follow a customized adaptive buffering ("Adaptive Memory Control") operation that is specified by the encoder in the forward channel.

The "Sliding Window" buffering type operates as follows. First, the decoder determines whether the picture can be stored into "unused" buffer capacity. If there is insufficient "unused" buffer capacity, the short-term picture with the largest default relative index (i.e. the oldest short-term picture in the buffer) shall be marked as "unused". The current picture is stored in the buffer and assigned a default relative index of zero. The default relative index of all other short-term pictures is incremented by one. The default relative indices of all long-term pictures are incremented by one minus the number of short-term pictures removed.

In the "Adaptive Memory Control" buffering type, specified pictures or sub-picture areas may be removed from the multi-picture buffer explicitly. The currently decoded picture, which is initially considered a short-term picture, may be inserted into the buffer with default relative index 0, may be assigned to a long-term index, or may be marked as "unused" by the encoder. Other short-term pictures may also be assigned to long-term indices. The buffering process shall operate in a manner functionally equivalent to the following: First, the current picture is added to the multi-picture buffer with default relative index 0, and the default relative indices of all other pictures are incremented by one. Then, the MMCO commands are processed:

If MMCO indicates a reset of the buffer contents , all pictures in the buffer are marked as "unused" except the current picture (which will be the picture with default relative index 0 since a buffer reset must be the first MMCO command as required by subclause AAA).

If MMCO indicates a maximum long-term index using MLIP1, all long-term pictures having long-term indices greater than or equal to MLIP1 are marked as "unused" and the default relative index order of the remaining pictures are not affected.

If MMCO indicates that a picture is to be marked as "unused" in the multi-picture buffer and if that picture has not already been marked as "unused", the specified picture is marked as "unused" in the multi-picture buffer and the default relative indices of all subsequent pictures in default order are decremented by one.

If MMCO indicates the assignment of a long-term index to a specified short-term picture and if the specified long-term index has not already been assigned to the specified short-term picture, the specified short-term picture is marked in the buffer as a long-term picture with the specified long-term index. If another picture is already present in the buffer with the same long-term index as the specified long-term

index, the other picture is marked as "unused". All short-term pictures that were subsequent to the specified short-term picture in default relative index order and all long-term pictures having a long-term index less than the specified long-term index have their associated default relative indices decremented by one. The specified picture is assigned to a default relative index of one plus the highest of the incremented default relative indices, or zero if there are no such incremented indices.