Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG

Geneva, Switzerland, January 29-February 1, 2002

Document **JVT-B118r2**
File: JVT-B118r2.doc
Generated: 2002-03-15

Title: Working Draft Number 2, Revision 2 (WD-2)

Status: Approved Output Document

Contact: Thomas Wiegand
Heinrich Hertz Institute (HHI), Einsteinufer 37, D-10587 Berlin, Germany
Tel: +49 - 30 - 31002 617, Fax: +49 - 030 - 392 72 00, wiegand@hhi.de

Purpose: Information

This document presents the Working Draft Number 2 (***WD-2***) released by the Joint Video Team (JVT). The JVT is being formed by ITU-T SG16 Q.6 (VCEG) and ISO/IEC JTC 1/SC 29/WG 11 (MPEG). This document is a description of a reference coding method to be used for the development of a new video compression method called ***JVT Coding*** as ITU-T Recommendation (H.26L) and ISO/IEC JTC1 Standard (MPEG-4, Part 10).

The following changes are included relative to WD-1:

- Deblocking filter JVT-B011 (general usefulness)
- CACM+ CABAC JVT-B036 (general usefulness with CABAC)
- SI Frames JVT-B055 (use for streaming, random access, error recovery)
- Intra Prediction JVT-B080 (put in software as configurable feature, use it in common conditions, seek complexity analysis)
- Interlace frame/field switch at picture level from JVT-B071 (with field coding as the candidate baseline interlace-handling design)
- MB partition alteration VCEG-O17 (general usefulness)
- CABAC eff. improve JVT-B101 (general usefulness with CABAC)
- Normative picture number update behavior from JVT-B042 (error resilience)
- Enhanced GOP concept description in interim file format non-normative appendix JVT-B042
- Exp-Golomb VLC JVT-B029 (general usefulness without CABAC)
- Bitstream NAL structure with start code and emulation prevention part of JVT-B063 (use for bitstream environments)
- Transform JVT-B038 (general usefulness)
- Extension of quant range (general usefulness)

The following changes are adopted, but no text has been submitted to the editor for inclusion into this document. Please note that all adoptions are tentative unless being fully integrated into this document and the software.

- Encoding Motion Search Range JVT-B022 (lower complexity method not for common conditions testing)
- Robust reference frame selection JVT-B102

Additionally, the formula in the encoder test model for the determination of the Lagrange parameter was adapted by the editor to obtain a working description.

Non-editorial changes between JVT-B118r1 and JVT-B118r2:

- Another correction of TABLE 7
- Picture type (Ptype) extended for signaling S-frames

CONTENTS

1 Scope

This document is a description of a reference coding method to be used for the development of a new video compression ITU-T Recommendation (H.26L) and ISO Standard (MPEG-4, Part 10). The basic configuration of the algorithm is similar to H.263 and MPEG-4, Part 2.

2 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

AC coefficient:	Any transform coefficient for which the frequency in one or both dimensions is non-zero.
MH-field picture:	A field structure MH-Picture.
MH-frame picture:	A frame structure MH-Picture.
MH-picture:	A predictive-coded picture: A picture that is coded using up to two motion compensated prediction signals from past and/or future reference fields or frames.
backward compatibility:	A newer coding standard is backward compatible with an older coding standard if decoders designed to operate with the older coding standard are able to continue to operate by decoding all or part of a bitstream produced according to the newer coding standard.
backward motion vector:	A motion vector that is used for motion compensation from a reference frame or reference field at a later time in display order.
backward prediction:	Prediction from the future reference frame (field).
bitstream (stream):	An ordered series of bits that forms the coded representation of the data.
bitrate:	The rate at which the coded bitstream is delivered from the storage medium to the input of a decoder.
block:	An N-row by M-column matrix of samples, or NxM transform coefficients (source, quantised or dequantised).
bottom field:	One of two fields that comprise a frame. Each line of a bottom field is spatially located immediately below the corresponding line of the top field.
byte aligned:	A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.
byte:	Sequence of 8-bits.
channel:	A digital medium that stores or transports a bitstream constructed according to this specification.
chrominance format:	Defines the number of chrominance blocks in a macroblock.
chrominance component:	A matrix, block or single sample representing one of the two colour difference signals related to the primary colours in the manner defined in the bitstream. The symbols used for the chrominance signals are Cr and Cb.
coded MH-frame:	A MH-frame picture or a pair of MH-field pictures.
coded frame:	A coded frame is a coded I-frame, a coded P-frame or a coded MH-frame.
coded I-frame:	An I-frame picture or a pair of field pictures, where the first field picture is an I-picture and the second field picture is an I-picture or a P-picture.
coded P-frame:	A P-frame picture or a pair of P-field pictures.
coded picture:	A coded picture is made of a picture header, the optional extensions immediately following it, and the following picture data. A coded picture may be a coded frame or a coded field.
coded video bitstream:	A coded representation of a series of one or more pictures as defined in this specification.
coded order:	The order in which the pictures are transmitted and decoded. This order is not necessarily the same as the display order.
coded representation:	A data element as represented in its encoded form.

coding parameters:	The set of user-definable parameters that characterise a coded video bitstream. Bitstreams are characterised by coding parameters. Decoders are characterised by the bitstreams that they are capable of decoding.
component:	A matrix, block or single sample from one of the three matrices (luminance and two chrominance) that make up a picture.
compression:	Reduction in the number of bits used to represent an item of data.
constant bitrate coded video:	A coded video bitstream with a constant average bitrate.
constant bitrate:	Operation where the bitrate is constant from start to finish of the coded bitstream.
data element:	An item of data as represented before encoding and after decoding.
data partitioning:	A method for dividing a bitstream into two separate bitstreams for error resilience purposes. The two bitstreams have to be recombined before decoding.
DC coefficient:	The transform coefficient for which the frequency is zero in both dimensions.
transform coefficient:	The amplitude of a specific basis function.
decoder input buffer:	The first-in first-out (FIFO) buffer specified in the video buffering verifier. 2
decoder:	An embodiment of a decoding process.
decoding (process):	The process defined in this specification that reads an input coded bitstream and produces decoded pictures or audio samples.
dequantisation:	The process of rescaling the quantised transform coefficients after their representation in the bitstream has been decoded and before they are presented to the inverse transform.
digital storage media (DSM):	A digital storage or transmission device or system.
transform:	Either the transform or the inverse discrete cosine transform. The transform is an invertible, discrete orthogonal transformation.
display aspect ratio:	The ratio height/width (in SI units) of the intended display.
display order:	The order in which the decoded pictures are displayed. Normally this is the same order in which they were presented at the input of the encoder.
display process:	The (non-normative) process by which reconstructed frames are displayed.
editing:	The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in this specification.
encoder:	An embodiment of an encoding process.
encoding (process):	A process, not specified in this specification, that reads a stream of input pictures or audio samples and produces a valid coded bitstream as defined in this specification.
fast forward playback:	The process of displaying a sequence, or parts of a sequence, of pictures in display-order faster than real-time.
fast reverse playback:	The process of displaying the picture sequence in the reverse of display order faster than real-time.
field:	For an interlaced video signal, a “field” is the assembly of alternate lines of a frame. Therefore an interlaced frame is composed of two fields, a top field and a bottom field.
field-based prediction:	A prediction mode using only one field of the reference frame. The predicted block size is 16x16 luminance samples. Field-based prediction is not used in progressive frames.
field period:	The reciprocal of twice the frame rate.

field (structure) picture:	A field structure picture is a coded picture with picture_structure is equal to "Top field" or "Bottom field".
flag:	A variable which can take one of only the two values defined in this specification.
forbidden:	The term "forbidden" when used in the clauses defining the coded bitstream indicates that the value shall never be used. This is usually to avoid emulation of start codes.
forced updating:	The process by which macroblocks are intra-coded from time-to-time to ensure that mismatch errors between the inverse transform processes in encoders and decoders cannot build up excessively.
forward compatibility:	A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.
forward motion vector:	A motion vector that is used for motion compensation from a reference frame or reference field at an earlier time in display order.
forward prediction:	Prediction from the past reference frame (field).
frame:	A frame contains lines of spatial information of a video signal. For progressive video, these lines contain samples starting from one time instant and continuing through successive lines to the bottom of the frame. For interlaced video a frame consists of two fields, a top field and a bottom field. One of these fields will commence one field period later than the other.
frame-based prediction:	A prediction mode using both fields of the reference frame.
frame period:	The reciprocal of the frame rate.
frame (structure) picture:	A frame structure picture is a coded picture with picture_structure is equal to "Frame".
frame rate:	The rate at which frames are be output from the decoding process.
future reference frame (field):	A future reference frame(field) is a reference frame(field) that occurs at a later time than the current picture in display order.
frame reordering:	The process of reordering the reconstructed frames when the coded order is different from the display order. Frame reordering occurs when MH-frames are present in a bitstream. There is no frame reordering when decoding low delay bitstreams.
group of pictures:	tbd (to be defined :o)
header:	A block of data in the coded bitstream containing the coded representation of a number of data elements pertaining to the coded data that follow the header in the bitstream.
interlace:	The property of conventional television frames where alternating lines of the frame represent different instances in time. In an interlaced frame, one of the field is meant to be displayed first. This field is called the first field. The first field can be the top field or the bottom field of the frame.
I-field picture:	A field structure I-Picture.
I-frame picture:	A frame structure I-Picture.
I-picture, intra-coded picture:	A picture coded using information only from itself.
intra coding:	Coding of a macroblock or picture that uses information only from that macroblock or picture.
level :	A defined set of constraints on the values which may be taken by the parameters of this specification within a particular profile. A profile may contain one or more levels. In a different context, level is the absolute value of a non-zero coefficient (see "run").
luminance component:	A matrix, block or single sample representing a monochrome representation of the signal and related to the primary colours in the manner defined in the bitstream. The

	symbol used for luminance is Y.
Mbit:	1 000 000 bits
macroblock:	The four 8 by 8 blocks of luminance data and the two (for 4:2:0 chrominance format), four (for 4:2:2 chrominance format) or eight (for 4:4:4 chrominance format) corresponding 8 by 8 blocks of chrominance data coming from a 16 by 16 section of the luminance component of the picture. Macroblock is sometimes used to refer to the sample data and sometimes to the coded representation of the sample values and other data elements defined in the macroblock header of the syntax defined in this part of this specification. The usage is clear from the context.
motion compensation:	The use of motion vectors to improve the efficiency of the prediction of sample values. The prediction uses motion vectors to provide offsets into the past and/or future reference frames or reference fields containing previously decoded sample values that are used to form the prediction error.
motion estimation:	The process of estimating motion vectors during the encoding process.
motion vector:	A two-dimensional vector used for motion compensation that provides an offset from the coordinate position in the current picture or field to the coordinates in a reference frame or reference field.
non-intra coding:	Coding of a macroblock or picture that uses information both from itself and from macroblocks and pictures occurring at other times.
opposite parity:	The opposit parity of top is bottom, and vice versa.
P-field picture:	A field structure P-Picture.
P-frame picture:	A frame structure P-Picture.
P-picture, predictive-coded picture:	A picture that is coded using motion compensated prediction from past reference fields or frame.
parameter:	A variable within the syntax of this specification which may take one of a range of values. A variable which can take one of only two values is called a flag.
parity (of field):	The parity of a field can be top or bottom.
past reference frame (field):	A past reference frame(field) is a reference frame(field) that occurs at an earlier time than the current picture in display order.
picture:	Source, coded or reconstructed image data. A source or reconstructed picture consists of three rectangular matrices of 8-bit numbers representing the luminance and two chrominance signals. A “coded picture” is defined in 3.21. For progressive video, a picture is identical to a frame, while for interlaced video, a picture can refer to a frame, or the top field or the bottom field of the frame depending on the context.
picture data:	In the VBV operations, picture data is defined as all the bits of the coded picture, all the header(s) and user data immediately preceding it if any (including any stuffing between them) and all the stuffing following it, up to (but not including) the next start code, except in the case where the next start code is an end of sequence code, in which case it is included in the picture data.
prediction:	The use of a predictor to provide an estimate of the sample value or data element currently being decoded.
prediction error:	The difference between the actual value of a sample or data element and its predictor.
predictor:	A linear combination of previously decoded sample values or data elements.
profile:	A defined subset of the syntax of this specification.
progressive:	The property of film frames where all the samples of the frame represent the same instances in time.
quantised transform coefficients:	transform coefficients before dequantisation. A variable length coded representation of quantised transform coefficients is transmitted as part of the coded video bitstream.

quantiser scale:	A scale factor coded in the bitstream and used by the decoding process to scale the dequantisation.
random access:	The process of beginning to read and decode the coded bitstream at an arbitrary point.
reconstructed frame:	A reconstructed frame consists of three rectangular matrices of 8- bit numbers representing the luminance and two chrominance signals. A reconstructed frame is obtained by decoding a coded frame.
reconstructed picture:	A reconstructed picture is obtained by decoding a coded picture. A reconstructed picture is either a reconstructed frame (when decoding a frame picture), or one field of a reconstructed frame (when decoding a field picture). If the coded picture is a field picture, then the reconstructed picture is the top field or the bottom field of the reconstructed frame.
reference field:	A reference field is one field of a reconstructed frame. Reference fields are used for forward and backward prediction when P-pictures and MH-pictures are decoded. Note that when field P-pictures are decoded, prediction of the second field P- picture of a coded frame uses the first reconstructed field of the same coded frame as a reference field.
reference frame:	A reference frame is a reconstructed frame that was coded in the form of a coded I-frame or a coded P-frame. Reference frames are used for forward and backward prediction when P-pictures and MH-pictures are decoded.
reordering delay:	A delay in the decoding process that is caused by frame reordering.
reserved:	The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for JVT defined extensions.
sample aspect ratio:	(abbreviated to SAR). This specifies the distance between samples. It is defined (for the purposes of this specification) as the vertical displacement of the lines of luminance samples in a frame divided by the horizontal displacement of the luminance samples. Thus its units are (metres per line) ÷ (metres per sample)
side information:	Information in the bitstream necessary for controlling the decoder. 16x8 prediction — A prediction mode similar to field-based prediction but where the predicted block size is 16x8 luminance samples.
run:	The number of zero coefficients preceding a non-zero coefficient, in the scan order. The absolute value of the non-zero coefficient is called “level”.
saturation:	Limiting a value that exceeds a defined range by setting its value to the maximum or minimum of the range as appropriate.
skipped macroblock:	A macroblock for which no data is encoded.
slice:	A series of macroblocks.
Source (input):	Term used to describe the video material or some of its attributes attributes before encoding.
spatial prediction:	prediction derived from the same decoded frame.
start codes [system and video]:	32-bit codes embedded in that coded bitstream that are unique. They are used for several purposes including identifying some of the structures in the coding syntax.
stuffing (bits, bytes):	Code-words that may be inserted into the coded bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream which would otherwise be lower than the desired bitrate.
temporal prediction:	prediction derived from reference frames or fields other than those defined as spatial prediction
top field:	One of two fields that comprise a frame. Each line of a top field is spatially located immediately above the corresponding line of the bottom field.
variable bitrate:	Operation where the bitrate varies with time during the decoding of a coded bitstream.

variable length coding (VLC):	A reversible procedure for coding that assigns shorter code-words to frequent events and longer code-words to less frequent events.
video buffering verifier (VBV):	A hypothetical decoder that is conceptually connected to the output of the encoder. Its purpose is to provide a constraint on the variability of the data rate that an encoder or editing process may produce.
video complexity verifier (VCV):	Ok… just kidding.
video sequence:	The highest syntactic structure of coded video bitstreams. It contains a series of one or more coded frames.
XYZ profile decoder:	decoder able to decode bitstreams conforming to the specifications of the XYZ profile (with XYZ being any of the defined Profile names).
zig-zag scanning order:	A specific sequential ordering of the transform coefficients from (approximately) the lowest spatial frequency to the highest.

3 Source Coder

3.1 Picture formats

The image width and height of the source data are restricted to be multiples of 16. At the moment only colour sequences using 4:2:0 chrominance sub-sampling are supported.

This specification describes coding of video that contains either progressive or interlaced frames, which may be mixed together in the same sequence.

A frame of video contains two fields, the top field and the bottom field, which are interleaved. The first (i.e., top), third, fifth, etc. lines of a frame are the top field lines. The second, fourth, sixth, etc. lines of a frame are the bottom field lines. A top field picture consists of only the top field lines of a frame. A bottom field picture consists of only the bottom field lines of a frame.

The two fields of an interlaced frame are separated in time by a field period (which is half the time of a frame period). They may be coded separately as two field pictures or together as a frame picture. A progressive frame should always be coded as a single frame picture. However, a progressive frame is still considered to consist of two fields (at the same instant in time) so that other field pictures may reference the sub-fields of the frame.

The vertical and temporal sampling positions of samples in interlaced frames are shown in Figure 1. The vertical sampling positions of the chrominance samples in a top field of an interlaced frame are specified as shifted up by 1/4 luminance sample height relative to the field-sampling grid in order for these samples to align vertically to the usual position relative to the full-frame sampling grid. The vertical sampling positions of the chrominance samples in a bottom field of an interlaced frame are specified as shifted down by 1/4 luminance sample height relative to the field-sampling grid in order for these samples to align vertically to the usual position relative to the full-frame sampling grid. The horizontal sampling positions of the chrominance samples are specified as unaffected by the application of interlaced field coding.

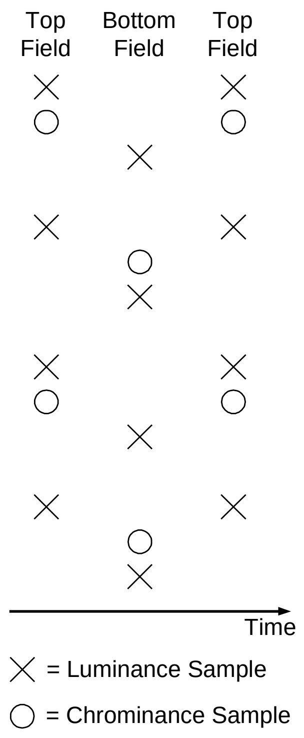


FIGURE 1

Vertical and temporal sampling positions of samples in interlaced frames

3.2 Subdivision of a picture into macroblocks

Pictures (both frame and field) are divided into macroblocks of 16x16 pixels. For instance, a QCIF picture is divided into 99 macroblocks as indicated in FIGURE 1. A number of consecutive macroblocks in coding order (see FIGURE 1) can be organized in slices. Slices represent independent coding units in a way that they can be decoded without referencing other slices of the same frame.

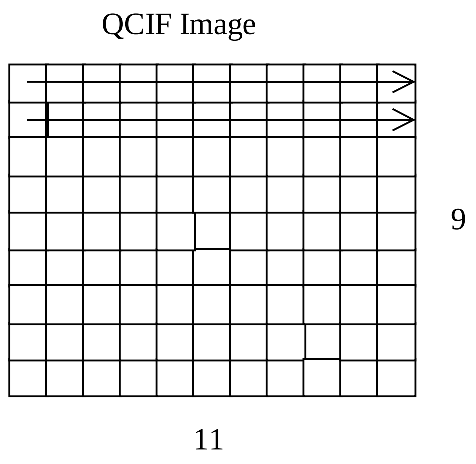


FIGURE 1

Subdivision of a QCIF picture into 16x16 macroblocks

3.3 Order of the bitstream within a macroblock

FIGURE 2 and FIGURE 3 indicate how a macroblock or 8x8 sub-block is divided and the order of the different syntax elements resulting from coding a macroblock.

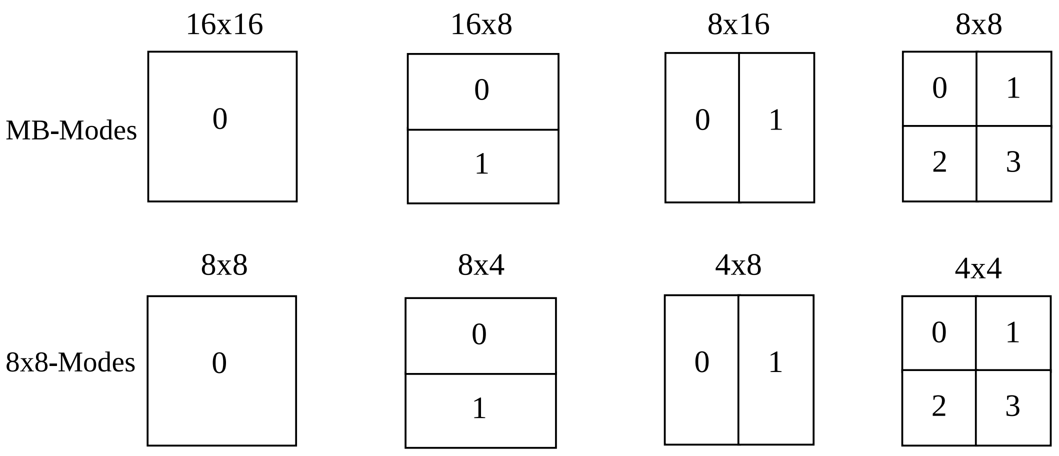


FIGURE 2

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

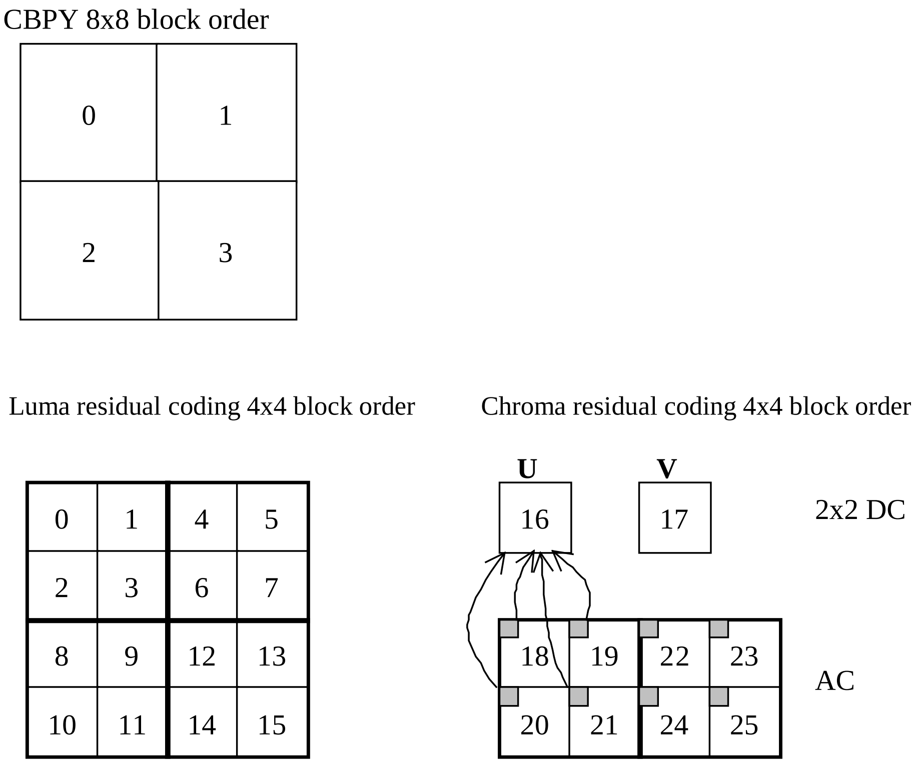


FIGURE 3

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

4 Syntax and Semantics

4.1 Organization of syntax elements (NAL concept)

The Video Coding Layer (VCL) is defined to efficiently represent the content of the video data, and the Network Adaptation Layer (NAL) is defined to format that data and provide header information in a manner appropriate for conveyance by the higher level system. The data is organized into data packets, each of which contains an integer number of bytes. These data packets are then transmitted in a manner defined by the NAL.

Any sequence of bits to be carried by an NAL can be formatted into a sequence of bytes in a manner defined as a Raw Byte Sequence Payload (RBSP), and any RBSP can be encapsulated in a manner that prevents emulation of byte stream start code prefixes in a manner defined defined as an encapsulated byte sequence payload (EBSP). These two formats are described below in this section.

4.1.1 Raw Byte Sequence Payload (RBSP)

A raw byte sequence payload (RBSP) is defined as an ordered sequence of bytes that contains a string of data bits (SODB). A SODB is defined as an ordered sequence of bits, in which the left-most bit is considered to be the first and most significant bit (MSB) and the right-most bit is considered to be the last and least significant bit (LSB). The RBSP contains the SODB in the following form:

1. If the SODB is null, the RBSP is also null.
2. Otherwise, the RBSP shall contain the SODB in the following form:
    a. The first byte of the RBSP shall contain the (most significant, left-most) eight bits of the SODB; the next byte of the RBSP shall contain the next eight bits of the SODB, etc.; until fewer than eight bits of the SODB remain.
    b. The final byte of the RBSP shall have the following form:
        i. The first (most significant, left-most) bits of the final RBSP byte shall contain the remaining bits of the SODB, if any,
        ii. The next bit of the final RBSP byte shall consist of a single packet stop bit (PSB) having the value one ('1'), and
        iii. Any remaining bits of the final RBSP byte, if any, shall consist of byte-alignment stuffing bits (BASB's) having the value zero ('0').

Note that the last byte of a RBSP can never have the zero (0x00).

If the boundaries of the RBSP are known, the decoder can extract the SODB from the RBSP by concatenating the bits of the bytes of the RBSP and discarding the last (least significant, right-most) bit having the value one ('1') and any following (less significant, farther to the right) bits that follow it having the value zero ('0').

The means for determining the boundaries of the RBSP depend on the NAL.

4.1.2 Encapsulated Byte Sequence Payload (EBSP)

A encapsulated byte sequence payload (EBSP) is defined as an ordered sequence of bytes that contains the raw byte sequence payload (RBSP) in the following form:

1. If the RBSP contains fewer than three bytes, the EBSP shall be the same as the RBSP.
2. Otherwise, the EBSP shall contain the RBSP in the following form:
    a. The first byte of the EBSP shall contain the first byte of the RBSP,
    b. The second byte of the EBSP shall contain the second byte of the RBSP,
    c. Corresponding to each subsequent byte of the RBSP, the EBSP shall contain one or two subsequent bytes as follows:
        i. If the last two previous bytes of the EBSP are both equal to zero (0x00) and if the next byte of the RBSP is either equal to one (0x01) or equal to 255 (0xFF), the EBSP shall contain two bytes of data that correspond to the next byte of the

RBSP. The first of these two bytes shall be an emulation prevention byte (EPB) equal to 255 (0xFF), and the second of these two bytes shall be equal to the next byte of the RBSP.

ii. Otherwise, the EBSP shall contain one next byte of data corresponding to the next byte of the RBSP. This byte shall be equal to the next byte of the RBSP.

The format of the EBSP prevents the three-byte start code prefix (SCP) equal to 0x00, 0x00, 0x01 from occurring within an EBSP. A decoder shall extract the RBSP from the EBSP by removing and discarding each byte having the value 255 (0xFF) which follows two bytes having the value zero (0x00) within an EBSP. The means for determining the boundaries of an EBSP is specified by the NAL.

Note that the effect of concatenating two or more RBSP's and then encapsulating them into an EBSP is the same as first encapsulating each individual RBPS and then concatenating the result (because the last byte of an RBSP is never zero). This allows the association of individual EBSP's to video data packets to depend on the NAL without affecting the content of the EBSP. For example, high-level header data can be placed in its own EBSP and this same EBSP content can be carried in several alternative ways, depending on the particular NAL design in use, including the possibilities of:

- Sending it within a low-level video packet with associated lower-level data for independent packet decoding, or
- Sending it in a separate higher-level header packet, or
- Sending it in an external "out-of-band" reliable channel.

4.1.3 Payload Type Indicator Byte (PTIB)

The type of data carried in a data packet is indicated by a payload type indicator byte (PTIB). The table below contains the values defined for the PTIB.

PTIB Value	Meaning
0x01	Sequence Data: Payload contains configuration information for sequence
0x02	Sequence SEI: Payload contains SEI for sequence
0x03	Random Access Point Data: Payload TBD
0x04	Random Access Point SEI: Payload contains SEI for random access point
0x05	Picture Data: Null payload
0x06	Picture SEI: Payload contains SEI for picture
0x07	Non-Partitioned Slice Data: Payload is one EBSP containing slice header data and non-data-partitioned video data for the slice
0x05	Slice Mode and MV Data: Payload is one EBSP containing slice header data and all other non-coefficient data for the slice
0x06	Slice Intra Coefficient Data: Payload is one EBSP containing slice header data and all intra coefficients for the slice
0x07	Slice Inter Coefficient Data: Payload is one EBSP containing slice header data and all non-intra coefficients for the slice
0x08	End of Sequence Data: Null payload
0x09	End of Sequence SEI: Payload contains SEI for end of sequence

[*Ed. Note:* Further work needed to finalize format of PTIB and define the types of data to follow.]

4.1.4 Slice Mode

Tbd.

4.1.5 Data Partitioning Mode

Data Partitioning re-arranges the symbols in 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 partition. The partition to fetch from is determined through the decoder's state machine, according to the syntax diagram discussed in section 4.4.

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	Luminance AC Coefficients
6	TYPE_COEFF_C	Chrominance 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.

4.2 Syntax diagrams

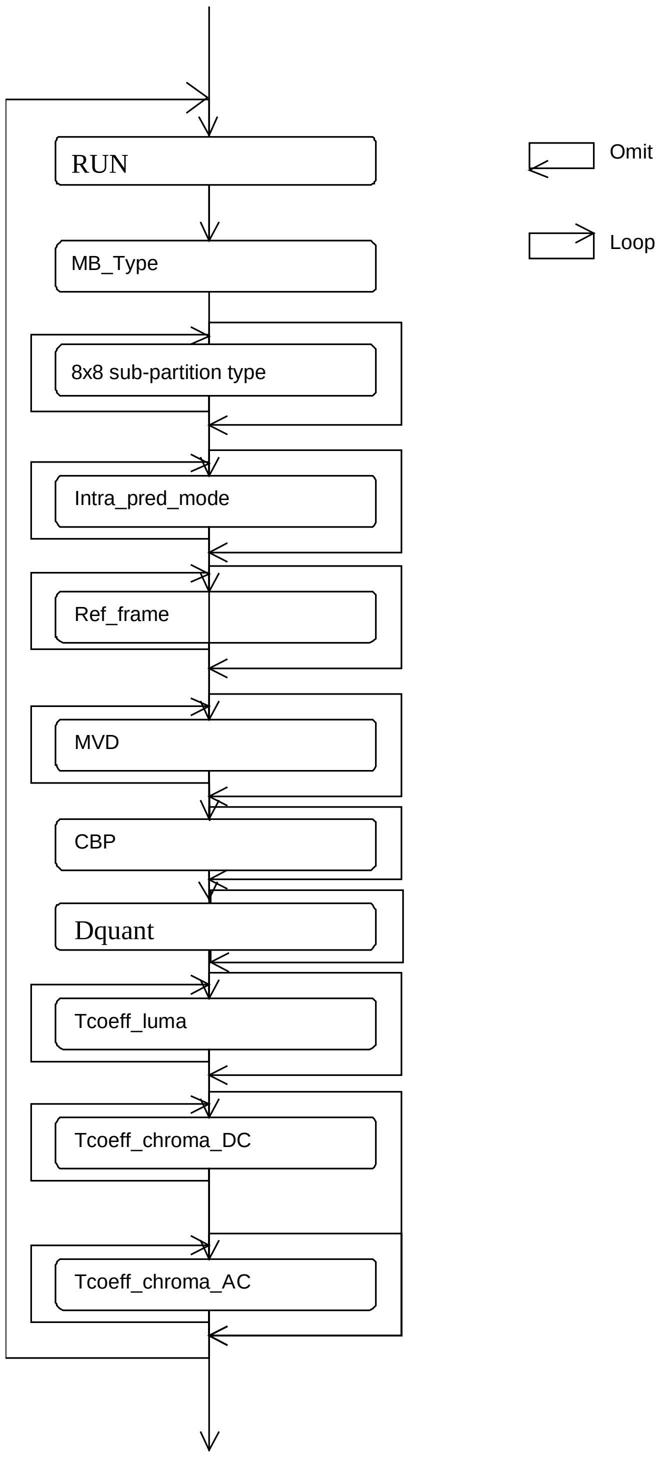


FIGURE 4

Syntax diagram for all the elements in the macroblock layer

4.3 Slice Layer

Editor: The slice layer needs a significant amount of work together with the NAL concept. The Picture sync and picture type codewords are present to ease the VCL development.

The information on the slice layer is coded depending on the NAL type. The coding, the order and even the presence of some fields may differ between the different NALs.

4.3.1 Slicesync

A slice sync may be inserted as the first codeword if it is needed by the NAL. If UVLC codes are used for the transmission of the slice level it should be a 31 bit long (L = 31) codeword with the INFO part set to zero.

4.3.2 Temporal reference (TRType/TR)

Two codewords. Temporal reference type (TRType) indicates how TR is transmitted

TRType = 0	Absoltue TR
TRType <> 0	Error

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.

4.3.3 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.
Code_number =5:	SP picture with prediction from the most recent decoded picture only.
Code_number =6:	SP 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 =7:	SI picture

4.3.4 Picture structure (PSTRUCT)

Code_number =0:	Progressive frame picture.
Code_number =1:	Top field picture.
Code_number =2:	Bottom field picture.
Code_number =3:	Interlaced frame picture, whose top field precedes its bottom field in time.
Code_number =4:	Interlaced frame picture, whose bottom field precedes its top field in time.

Note that when top field and bottom field pictures are coded for a frame, the one that is decoded first is the one that occurs first in time.

4.3.5 Size information

A series of up to three codewords. The first codeword indicates a size change. If set to zero the size is unchaged otherwise (if set to one) it is followed by two codewords containing the new width and height.

4.3.6 Reference Picture ID

4.3.7 First MB in slice

The number of the first macroblock contained in this slice.

4.3.8 Slice QP (SQP)

Information about the quantizer QUANT to be used for luminance for the picture. (See under Quantization concerning QUANT for chrominance). The 6-bit representation is the natural binary representation of the value of QP+12, which range from 0 to 47 (QP ranges from -12 to +39). QP+12 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: $QUANT_{H.263}(QP) \approx QP0(QP) = 2QP/6$. QP0() will be used later for scaling purposes when selecting prediction modes. Negative values of QP correspond to even smaller step sizes, as described below under quantization.

4.3.9 SP Slice QP

For SP frames the SP slice QP is transmitted, using the same scale described above.

4.3.10 Number of macroblocks in slice

For CABAC entropy coding the number of macroblocks contained in the slice is transmitted.

4.3.11 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.

4.3.12 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.

4.3.13 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 1

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

4.3.13.1 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.

4.3.13.2 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.

4.3.14 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 RPS 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.

4.3.14.1 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).

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, or
- 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

TABLE 2

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

4.3.14.2 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”. 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).

4.3.14.3 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”.

4.3.14.4 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.

4.3.15 Supplemental Enhancement Information

Supplemental enhancement information (SEI) is encapsulated into chunks of data separate from coded slices, for example. It is up to the network adaptation layer to specify the means to transport SEI chunks. Each SEI chunck may contain one or more SEI messages. Each SEI message shall consist of a SEI header and SEI payload. The SEI header starts at a byte-aligned position from the first byte of a SEI chunk or from the first byte after the previous SEI message. The SEI header consists of two codewords, both of which consist of one or more bytes. The first codeword indicates the SEI payload type. Values from 00 to FE shall be reserved for particular payload types, whereas value FF is an escape code to extend the value range to yet another byte as follows:

```
payload_type = 0;
for (;;) {
  payload_type += *byte_ptr_to_sei;
  if (*byte_ptr_to_sei < 0xFF)
    break;
  byte_ptr_to_sei++;
}
```

The second codeword of the SEI header indicates the SEI payload size in bytes. SEI payload size shall be coded similarly to the SEI payload type.

The SEI payload may have a SEI payload header. For example, a payload header may indicate to which picture the particular data belongs. The payload header shall be defined for each payload type separately.

4.4 Macroblock layer

Following the syntax diagram for the macroblock elements, the various elements are described.

4.4.1 Number of Skipped Macroblocks (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.

If PSTRUCT indicates a frame, then the skipped macroblock is formed by copying the collocated picture material from the last decoded frame, which either was decoded from a frame picture or is the union of two decoded field pictures. If PSTRUCT indicates a field, then the skipped macroblock is formed by copying the collocated material from the last decoded field of the same parity (top or bottom), which was either decoded from a field picture or is part of the most recently decoded frame picture.

[Editor: this needs alignment with the RPS.]

For a B macroblock skip means direct mode without coefficients. RUN indicates the number of skipped macroblocks in an inter- or B-picture before a coded macroblock. If a picture or slice ends with one or more skipped macroblocks, they are represented by an additional RUN which counts the number of skipped macroblocks.

4.4.2 Macro block type (MB_Type)

Refer to TABLE 7. There are different MB-Type tables for Intra and Inter frames.

4.4.2.1 Intra Macroblock modes

Intra 4x4	Intra coding as defined in sections **Error! Reference source not found.** to **Error! Reference source not found.**.
Imode, nc, AC	See definition in section 0. These modes refer to 16x16 intra coding.

4.4.2.2 Inter Macroblock Modes

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. 16x8)	The macroblock is predicted from a past picture with block size NxM. For the macroblock modes 16x16, 16x8, and 8x16, a motion vector is provided for each NxM block. If N=M=8, for each 8x8 sub-partition an additional codeword is transmitted which indicates in which mode the corresponding sub-partition is coded (see section 4.4.3). 8x8 sub-partitions can also be coded in intra 4x4 mode. Depending on N,M and the 8x8 sub-partition modes there may be 1 to 16 sets of motion vector data for a macroblock.
Intra 4x4	4x4 intra coding.

Code numbers from 6 and upwards represent 16x16 intra coding.

4.4.3 8x8 sub-partition modes

NxM (eg. 8x4)	The corresponding 8x8 sub-partition is predicted from a past picture with block size NxM. A motion vector is transmitted for each NxM block. Depending on N and M, up to 4 motion vectors are coded for an 8x8 sub-partition, and thus up to 16 motion vectors are transmitted for a macroblock.

Intra The 8x8 sub-partition is coded in intra 4x4 mode.

4.4.4 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 Q from neighboring blocks may already be decoded and may be used for prediction. When pixels E-H are not available, whether because they have not yet been decoded, are outside the picture or outside the current independent slice, the pixel value of D is substituted for pixels E-H. When pixels M-P are not available, the pixel value of L is substituted for pixels M-P.

```
Q A B C D E F G H
I|a b c d|
J|e f g h|
K|i j k l|
L|m n o p|
M
N
O
P
```

FIGURE 5

Syntax diagram for the Picture header.

4.4.4.1.1 Prediction for Intra-4x4 mode for luminance blocks

For the luminance signal, there are 9 intra prediction modes labeled 0 to 8. Mode 0 is ‘DC-prediction’ (see below). The other modes represent directions of predictions as indicated below.

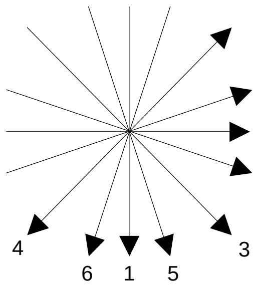


FIGURE 6

Syntax diagram for the Picture header.

Mode 0: DC prediction

Generally all pixels are predicted by (A+B+C+D+I+J+K+L)//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.

Mode 1: 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.

Mode 2: 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.

Mode 3: Diagonal Down/Right prediction

This mode is used only if all A,B,C,D,I,J,K,L,Q are inside the picture. This is a 'diagonal' prediction.

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

Mode 4: Diagonal Down/Left prediction

This mode is used only if all A,B,C,D,I,J,K,L,Q are inside the picture. This is a 'diagonal' prediction.

a is predicted by	(A + 2B + C + I + 2J + K + 4) / 8;
b, e are predicted by	(B + 2C + D + J + 2K + L + 4) / 8;
c, f, i are predicted by	(C + 2D + E + K + 2L + M + 4) / 8;
d, g, j, m are predicted by	(D + 2E + F + L + 2M + N + 4) / 8;
h, k, n are predicted by	(E + 2F + G + M + 2N + O + 4) / 8;
l, o are predicted by	(F + 2G + H + N + 2O + P + 4) / 8;
p is predicted by	(G + H + O + P + 2) / 4;

Mode 5: Vertical-Left prediction

This mode is used only if all A,B,C,D,I,J,K,L,Q are inside the picture. This is a 'diagonal' prediction.

a, j are predicted by	(Q + A + 1) / 2;
b, k are predicted by	(A + B + 1) / 2;
c, l are predicted by	(B + C + 1) / 2;
d is predicted by	(C + D + 1) / 2;
e, n are predicted by	(I + 2Q + A + 2) / 4;
f, o are predicted by	(Q + 2A + B + 2) / 4;
g, p are predicted by	(A + 2B + C + 2) / 4;
h is predicted by	(B + 2C + D + 2) / 4;
i is predicted by	(Q + 2I + J + 2) / 4;
m is predicted by	(I + 2J + K + 2) / 4;

Mode 6: Vertical-Right prediction

This mode is used only if all A,B,C,D,I,J,K,L,Q are inside the picture. This is a 'diagonal' prediction.

a is predicted by	(2A + 2B + J + 2K + L + 4) / 8;
b, i are predicted by	(B + C + 1) / 2;
c, j are predicted by	(C + D + 1) / 2;
d, k are predicted by	(D + E + 1) / 2;
l is predicted by	(E + F + 1) / 2;
e is predicted by	(A + 2B + C + K + 2L + M + 4) / 8;
f, m are predicted by	(B + 2C + D + 2) / 4;
g, n are predicted by	(C + 2D + E + 2) / 4;
h, o are predicted by	(D + 2E + F + 2) / 4;
p is predicted by	(E + 2F + G + 2) / 4;

Mode 7: Horizontal-Up prediction

This mode is used only if all A,B,C,D,I,J,K,L,Q are inside the picture. This is a 'diagonal' prediction.

a is predicted by	(B + 2C + D + 2I + 2J + 4) / 8;
b is predicted by	(C + 2D + E + I + 2J + K + 4) / 8;
c, e are predicted by	(D + 2E + F + 2J + 2K + 4) / 8;
d, f are predicted by	(E + 2F + G + J + 2K + L + 4) / 8;
g, i are predicted by	(F + 2G + H + 2K + 2L + 4) / 8;
h, j are predicted by	(G + 3H + K + 3L + 4) / 8;
l, n are predicted by	(L + 2M + N + 2) / 4;
k, m are predicted by	(G + H + L + M + 2) / 4;
o is predicted by	(M + N + 1) / 2;
p is predicted by	(M + 2N + O + 2) / 4;

Mode 8: Horizontal-Down prediction

This mode is used only if all A,B,C,D,I,J,K,L,Q are inside the picture. This is a 'diagonal' prediction.

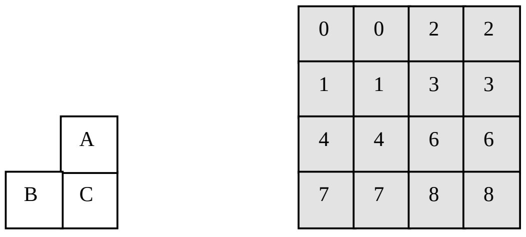
a, g are predicted by	(Q + I + 1) / 2;
b, h are predicted by	(I + 2Q + A+ 2) / 4;
c is predicted by	(Q + 2A + B+ 2) / 4;
d is predicted by	(A + 2B + C+ 2) / 4;
e, k are predicted by	(I + J + 1) / 2;
f, l are predicted by	(X + 2I + J+ 2) / 4;
I, o are predicted by	(J + K + 1) / 2;
j, p are predicted by	(I + 2J + K+ 2) / 4;
m is predicted by	(K + L + 1) / 2;
n is predicted by	(J + 2K + L + 2) / 4;

Coding of Intra 4x4 prediction modes

Since each of the 4x4 luminance 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 **Error! Reference source not found.**a. 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. . When an adjacent block is coded by 16x16 intra mode, prediction mode is “mode 0: DC_prediction”; when it is coded in inter mode, prediction mode is “mode 0: DC_prediction” in the usual case and “outside” in the case of constrained intra update. This ordering is listed in **Error! Reference source not found.**.

For each prediction mode of A and B a list of 9 numbers is given. Example: Prediction mode for A and B is 2. The string 2 8 7 1 0 6 4 3 5 indicates that mode 2 is also the most probable mode for block C. Mode 8 is the next most probable one etc. In the bitstream there will for instance be information that Prob0 = 1 (see TABLE 8) indicating that the next most probable mode shall be used for block C. In our example this means Intra prediction mode 8. Use of '–' in the table indicates that this instance cannot occur because A or B or both are outside the picture.

For more efficient coding, information on intra prediction of two 4x4 luminance blocks are coded in one codeword (Prob0 and Prob1 in TABLE 8). The order of the resulting 8 codewords is indicated in **Error! Reference source not found.**b.



a

FIGURE 7

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 3

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

B/A	Outside	0	1	2	3
Outside	0--------	01-------	10-------	---------	---------
0	02-------	021648573	125630487	021876543	021358647

1	---------	102654387	162530487	120657483	102536487
2	20-------	280174365	217683504	287106435	281035764
3	---------	201385476	125368470	208137546	325814670
4	---------	201467835	162045873	204178635	420615837
5	---------	015263847	152638407	201584673	531286407
6	---------	016247583	160245738	206147853	160245837
7	---------	270148635	217608543	278105463	270154863
8	---------	280173456	127834560	287104365	283510764

B/A	4	5	6	7	8
Outside	---------	---------	---------	---------	---------
0	206147583	512368047	162054378	204761853	208134657
1	162045378	156320487	165423078	612047583	120685734
2	287640153	215368740	216748530	278016435	287103654
3	421068357	531268470	216584307	240831765	832510476
4	426015783	162458037	641205783	427061853	204851763
5	125063478	513620847	165230487	210856743	210853647
6	640127538	165204378	614027538	264170583	216084573
7	274601853	271650834	274615083	274086153	278406153
8	287461350	251368407	216847350	287410365	283074165

4.4.4.1.2 Prediction for Intra-16x16 mode for luminance

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 Luminance macroblock. We have 4 different prediction modes:

<u>Mode 0 (vertical)</u>

Pred(i,j) = P(i,-1), i,j=0..15

<u>Mode 1 (horizontal)</u>

Pred(i,j) = P(-1,j), i,j=0..15

<u>Mode 2 (DC prediction)</u>

Pred(i,j) = $(\sum_{i=0}^{15}(P(-1,i)+P(i,-1)))/32$ i,j=0..15,

where only the average of 16 pixels are used when the other 16 pixels are outside the picture or slice. If all 32 pixels are outside the picture, the prediction for all pixels in the block is 128.

<u>Mode 3 (Plane prediction)</u>

Pred(i,j) = max(0, min(255, (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

$$H = \sum_{i=1}^{8} ix(P(7+i,-1) - P(7-i,-1))$$

$$V = \sum_{j=1}^{8} jx(P(-1,7+j) - P(-1,7-j))$$

<u>Residual coding</u>

The residual coding is based on 4x4 transform. But similar to coding of chrominance 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.

<u>Coding of mode information for Intra-16x16 mode</u>

See TABLE 7. 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 chrominance (see 4.4.7)

4.4.4.1.3 Prediction in intra coding of chrominance blocks

For chrominance prediction there is only one mode. No information is therefore needed to be transmitted. The prediction is indicated in the figure below. The 8x8 chrominance 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)

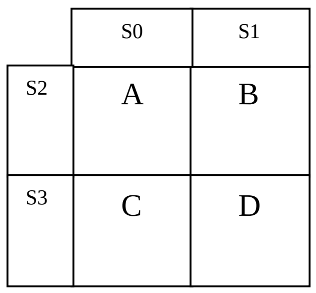


FIGURE 8

Prediction of Chrominance blocks.

4.4.5 Reference picture (Ref_picture)

If PTYPE indicates possibility of prediction from more than one previous decoded picture, the exact picture to be used must be signalled. This is done according to the following tables. If PSTRUCT indicates that the current picture is a frame picture, then the reference picture is a previous frame in the reference buffer that was either encoded as a single frame picture or a frame that was encoded as two field pictures and have been re-constructed as a frame. Thus for frames the following table gives the reference frame:

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

The reference parameter is transmitted for each 16x16, 16x8, or 8x16 block. If the macroblock is coded in 8x8 mode, the reference frame parameter is coded once for each 8x8 sub-partition unless the 8x8 sub-partition is transmitted in intra mode. If the UVLC is used for entropy coding and the macroblock type is indicated by codeword 4 (8x8, ref=0), no reference frame parameters are transmitted for the whole macroblock.

If PSTRUCT indicates that the current picture is a field picture, then the reference picture is either a previous field in the reference buffer that was separately encoded as a field picture or a previous field that is half of a frame that was encoded as a frame picture. Note that for a P field picture, forward prediction from field 1 to field 2 in the same frame is allowed. For the purpose of determining REF_picture, a unique reference field number is assigned to each reference field in the reference field (frame) buffer according to its distance from the current field, as shown in Figures 3a and 3b, modified by the fact that smaller code numbers are given to the fields of the same field parity as the current field. Thus for fields the following table gives the reference field:

Code_number	Reference field
0	the last decoded previous field of the same parity (2 fields back)
1	the last decoded previous field of the opposite parity (1 field back)
2	same parity field 4 fields back
3	opposite parity field 3 fields back
..	..

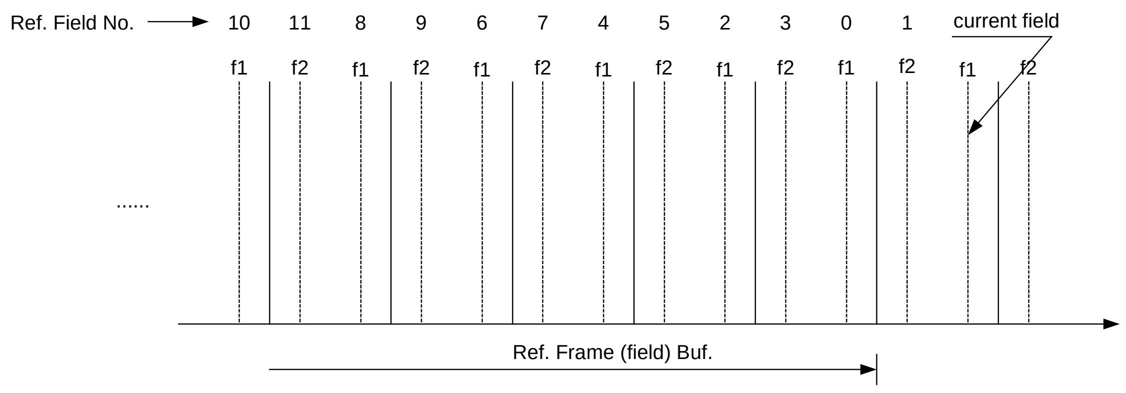


FIGURE 3a

Reference picture number assignment when the current picture is the *first* field coded in a frame. Solid-line is for frames and dotted-line for fields. f1 stands for field 1 and f2 for field 2.

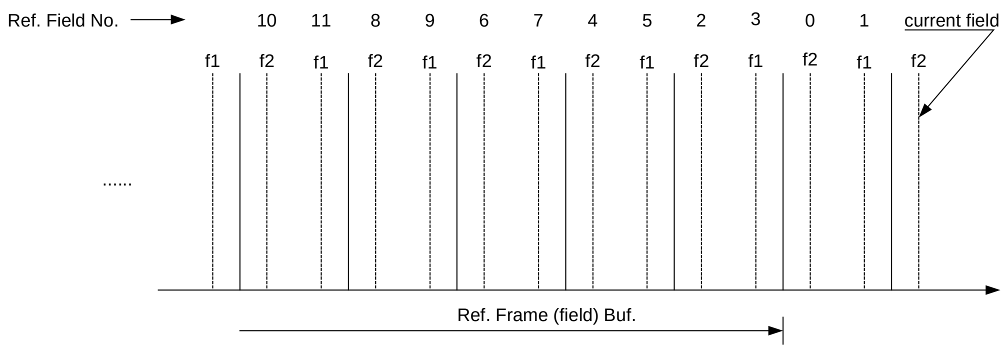


FIGURE 3b

Reference picture number assignment when the current picture is the *second* field coded in a frame. Solid-line is for frames and dotted-line for fields. f1 stands for field 1 and f2 for field 2.

4.4.6 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.

4.4.6.1 Prediction of vector components

With exception of the 16x8 and 8x16 block shapes, "median prediction" (see 4.4.6.1.1) is used. In case the macroblock may be classified to have directional segmentation the prediction is defined in 4.4.6.1.2.

4.4.6.1.1 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

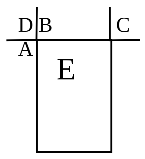


FIGURE 9

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.

4.4.6.1.2 Directional segmentation prediction

If the macroblock where the block to be predicted is coded in 16x8 or 8x16 mode, 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

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

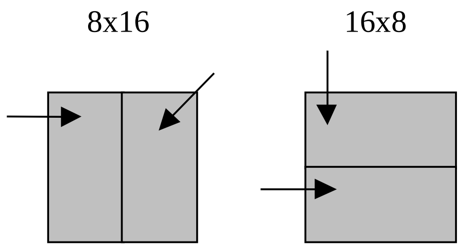


FIGURE 10

Directional segmentation prediction

4.4.6.2 Chrominance vectors

Chrominance vectors are derived from the Luminance vectors. Since chrominance has half resolution compared to luminance, the chrominance vectors are obtained by a division of two:

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

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

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

4.4.7 Coded Block Pattern (CBP)

The CBP contains information of which 8x8 blocks - Luminance and chrominance - 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 luminance 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 chrominance we define 3 possibilities:

nc=0:	no chrominance coefficients at all.
nc=1	There are nonzero 2x2 transform coefficients. All chrominance AC coefficients = 0. Therefore we do not send any EOB for chrominance AC coefficients.
nc=2	There may be 2x2 nonzero coefficients and there is at least one nonzero chrominance 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 chrominance 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.

4.4.8 Change of Quantizer Value (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 -26 to +26, which enables the QP to be changed to any value in the range [-12..39].

$\text{QUANT}_{new} = -12 + \text{modulo}_{52}(\text{QUANT}_{old} + \text{Dquant} + 64)$ (also known as "arithmetic wrap")

5 Decoding Process

5.1 Slice Decoding

5.1.1 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.

5.1.1.1 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.

5.1.1.2 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 indicated parameter set in the latest received slice or data partition signals the required picture number update behavior, the decoder shall operate as follows. The default picture index order shall be updated as if pictures corresponding to missing picture numbers were inserted to the multi-picture buffer using the “Sliding Window” buffering type. An index corresponding to a missing picture number is called an “invalid” index. The decoder should infer an unintentional picture loss if any “invalid” index is referred to in motion compensation or if an “invalid” index is re-mapped.

If the indicated parameter set in the latest received slice or data partition does not signal the required picture number update behavior, the decoder should infer an unintentional picture loss 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.

In case of an unintentional picture loss, the decoder may invoke some concealment process. If the required picture number update behavior was indicated, the decoder may replace the picture corresponding to an “invalid” index with an error-concealed one and remove the “invalid” indication. If the required picture number update behavior was not indicated, the decoder may insert an error-concealed picture into the multi-picture buffer assuming the “Sliding Window” buffering type. 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.

5.1.1.3 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.

5.1.1.4 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 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).

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.

5.2 Motion Compensation

5.2.1 Fractional pixel accuracy

Note that when PSTRUCT indicates a field picture, then all of the interplations for sub-pel motion are based solely on pixels from just the reference field indicated by REF_picture; the pixels from the other field of the containing frame play no role at all.

5.2.1.1 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.

5.2.1.2 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).

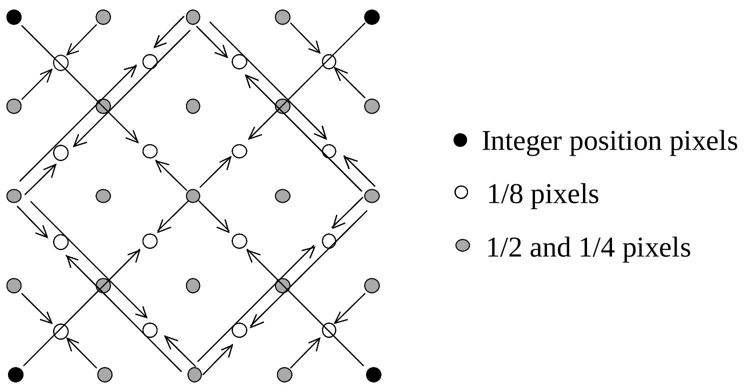


FIGURE 11

Diagonal interpolation for 1/8 pel interpolation.

5.3 Transform Coefficient Decoding

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.

5.3.1 Transform for Blocks of Size 4x4 Samples

Instead of a discrete cosine transform (DCT), an integer transform with a similar coding gain as a 4x4 DCT is used. The transformation of input pixels X={x_{00} … x_{33}} to output coefficients Y is defined by:

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix}$$

Multiplication by two can be performed either through additions or through left shifts, so that no actual multiplication operations are necessary. Thus, we say that the transform is multiplier-free.

For input pixels with 9-bit dynamic range (because they are residuals from 8-bit pixel data), the transform coefficients are guaranteed to fit within 16 bits, even when the second transform for DC coefficients is used. Thus, all transform operations can be computed in 16-bit arithmetic. In fact, the maximum dynamic range of the transforms coefficients fills a range of only 15.2 bits; this small headroom can be used to support a variety of different quantization strategies (which are outside the scope of this specification).

The inverse transformation of normalized coefficients Y’={y'_{00} … y'_{33}} to output pixels X’ is defined by:

$$X' = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} y'_{00} & y'_{01} & y'_{02} & y'_{03} \\ y'_{10} & y'_{11} & y'_{12} & y'_{13} \\ y'_{20} & y'_{21} & y'_{22} & y'_{23} \\ y'_{30} & y'_{31} & y'_{32} & y'_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$$

Multiplications by ½ are actually performed via right shifts, so that the inverse transform is also multiplier-free. The small errors introduced by the right shifts are compensated by a larger dynamic range for the data at the input of the inverse transform.

The transform and inverse transform matrices above have orthogonal basis functions. Unlike the DCT, though, the basis functions don't have the same norm. Therefore, for the inverse transform to recover the original pixels, appropriate normalization factors must be applied to the transform coefficients before quantization and after dequantization. Such factors are absorbed by the quantization and dequantization scaling factors described below.

By the above exact definition of the inverse transform, the same operations will be performed on coder and decoder side. Thus we avoid the usual problem of "inverse transform mismatch".

5.3.1.1 Transform for Blocks of Size 4x4 Samples Containing DC vales for Luminance

To minimize the dynamic range expansion due to transformations (and thus minimize rounding errors in the quantization scale factors), a simple scaled Hadamard transform is used. The direct transform is defined by:

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_{D00} & x_{D01} & x_{D02} & x_{D03} \\ x_{D10} & x_{D11} & x_{D12} & x_{D13} \\ x_{D20} & x_{D21} & x_{D22} & x_{D23} \\ x_{D30} & x_{D31} & x_{D32} & x_{D33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) // 2$$

where the symbol // denotes division with rounding to the nearest integer:

$$a // 2^b = \operatorname{sign}(a) \times \left[\left(\operatorname{abs}(a) + 2^{b-1} \right) >> b \right]$$

The inverse transform for the 4x4 luminance DC coefficients is defined by:

$$X_{QD} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} y_{QD00} & y_{QD01} & y_{QD02} & y_{QD03} \\ y_{QD10} & y_{QD11} & y_{QD12} & y_{QD13} \\ y_{QD20} & y_{QD21} & y_{QD22} & y_{QD23} \\ y_{QD30} & y_{QD31} & y_{QD32} & y_{QD33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

5.3.2 Transform for Blocks of Size 2x2 Samples (DC vales for Chrominance)

With the low resolution of chrominance 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 chrominance. The 2 dimensional 2x2 transform procedure is illustrated below. DC0,1,2,3 are the DC coefficients of 2x2 chrominance blocks.

DC0	DC1	Two dimensional 2x2 transform $\Rightarrow$	DDC(0,0)	DDC(1,0)
DC2	DC3		DDC(0,1)	DDC(1,1)

Definition of transform:

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

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

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

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

Definition of inverse transform:

DC0 = (DCC(0,0)+ DCC(1,0)+ DCC(0,1)+ DCC(1,1))

DC1 = (DCC(0,0)- DCC(1,0)+ DCC(0,1)- DCC(1,1))

DC2 = (DCC(0,0)+ DCC(1,0)- DCC(0,1)- DCC(1,1))

DC3 = (DCC(0,0)- DCC(1,0)- DCC(0,1)+ DCC(1,1))

5.3.3 Zig-zag Scanning and Quantization

5.3.3.1 Simple Zig-Zag Scan

Except for Intra coding of luminance 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:

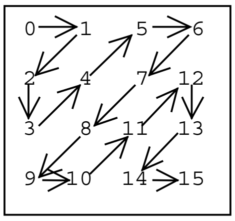


FIGURE 12

Simple zig-zag scan.

5.3.3.2 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.

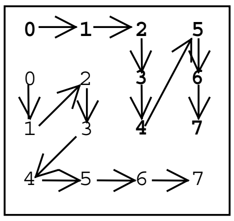


FIGURE 13

Double zig-zag scan.

5.3.3.3 Quantization

The quantization and dequantization processes shall perform usual quantization and dequantization, as well as take care of the necessary normalization of transform coefficients. As specified in sub-subsection 4.4.8, 52 different QP values are used, from -12 to +39. The quantization and dequantization equations are defined such that the equivalent step size doubles for every increment of 6 in QP. Thus, there is an increase in step size of about 12% from one QP to the next. There is no “dead zone” in the quantization process. For the QP in the range [0…31] the corresponding step size range is about the same as that for H.263. With the range [-12…39], the smallest step size is about four times smaller than in H.263, allowing for very high fidelity reconstruction, and the largest is about 60% larger than in H.263, allowing for more rate control flexibility.

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

QPluma:	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
QPchroma:	17	17	18	19	20	20	21	22	22	23	23	24	24	25	25	25	26	26	26	27	27	27	27

with QPchroma = QPluma if QPluma < 17.

5.3.3.3.1 Quantization of 4x4 luminance or chrominance coefficients

Since the step size doubles for every increment of 6 in QP, a periodic quantization table is used. Thus, our indices into quantization coefficient tables depend only on (QP+12)%6 = QP%6 (where the % symbol means the modulus operator), and the quantization and de-quantization formulas depend both on QP%6 and (QP+12)/6. In that way, the table size is minimized.

Quantization is performed according to the following equation:

$$Y_Q(i,j) = \left[Y(i,j) \cdot Q\left((QP+12)\%6, i, j\right) + f \right] / 2^{15+(QP+12)/6}, \quad i,j = 0,\ldots,3$$

where Y are the transformed coefficients, Y_Q are the corresponding quantized values, Q(m,i,j) are the quantization coefficients listed below, and |f| is in the range 0 to $2^{14+(QP+12)/6}/2$, with f having the same sign as the coefficient that is being quantized. Recall that QP+12 is the value signaled to the decoder, as discussed in Subsection **Error! Reference source not found.**. Note that while the intermediate value inside square brackets in the equation above has a 32-bit range, the final value Y_Q is guaranteed to fit in 16 bits.

5.3.3.3.2 Dequantization of 4x4 luminance or chrominance coefficients

Dequantization is performed according to the following equation:

$$Y'(i,j) = Y_Q(i,j) \cdot R\left((QP+12)\%6, i, j\right) << \left(QP+12\right)/6, \quad i,j = 0,\ldots,3$$

where the R(m,i,j) are the dequantization coefficients listed below. After dequantization, the inverse transform specified above is applied. Then the final results are normalized by:

$$X(i,j) = \left(X'(i,j) + 2^5 \right) >> 6$$

The dequantization formula can be performed in 16-bit arithmetic, because the coefficients R(m,i,j) are small enough. Furthermore, the de-quantized coefficients have the maximum dynamic range that still allows for the inverse transform to be performed in 16-bit arithmetic. Thus, the decoder needs only 16-bit arithmetic for dequantization and inverse transform.

5.3.3.3.3 Quantization of 4x4 luminance DC coefficients

Quantization is performed according to the following equation:

$$Y_Q(i,j) = \left[Y(i,j) \cdot Q\left((QP+12)\%6, i, j\right) + f \right] / 2^{16+(QP+12)/6}, \quad i,j = 0,\ldots,3$$

5.3.3.3.4 Dequantization of 4x4 luminance DC coefficients

The order of inverse transform and de-quantization of DC coefficients is changed, that is, the inverse transform is performed first, then dequantization. This achieves the best possible dynamic range during inverse transform computations.

After the inverse transform, dequantization is performed according to the following equation:

$$X_D(i,j) = \left[X_{QD}(i,j) \cdot R\left((QP+12)\%6, 0, 0\right) \right] / 2^{4-(QP+12)/6}, \quad i,j = 0,\ldots,3$$

5.3.3.3.5 Quantization of 2x2 chrominance DC coefficients

Quantization is performed according to the following equation:

$$Y_Q(i,j) = \left[Y(i,j) \cdot Q\left((QP+12)\%6, i, j\right) + 2f \right] / 2^{16+(QP+12)/6}, \quad i,j = 0,\ldots,3$$

5.3.3.3.6 Dequantization of 2x2 chrominance DC coefficients

Again the order of inverse transform and de-quantization of DC coefficients is changed, that is, the inverse transform is performed first, then dequantization. This achieves the best possible dynamic range during inverse transform computations.

After the inverse transform, dequantization is performed according to the following equation:

$$X_D(i,j) = \left[X_{QD}(i,j) \cdot R\left((QP+12)\%6,0,0\right) \right] / 2^{3-(QP+12)/6}, \quad i,j = 0,\ldots,3$$

5.3.3.3.7 Quantization and dequantization coefficient tables

The coefficients Q(m,i,j) and R(m,i,j), used in the formulas above, are defined in pseudo code by:

```
Q[m][i][j] = quantMat[m][0] for (i,j) = {(0,0),(0,2),(2,0),(2,2)},
Q[m][i][j] = quantMat[m][1] for (i,j) = {(1,1),(1,3),(3,1),(3,3)},
Q[m][i][j] = quantMat[m][2] otherwise.

R[m][i][j] = dequantMat[m][0] for (i,j) = {(0,0),(0,2),(2,0),(2,2)},
R[m][i][j] = dequantMat[m][1] for (i,j) = {(1,1),(1,3),(3,1),(3,3)},
R[m][i][j] = dequantMat[m][2] otherwise.

quantMat[6][3] = {{13107, 5243, 8066}, {11916, 4660, 7490}, {10082, 4194, 6554},
{9362, 3647, 5825}, {8192, 3355, 5243}, {7282, 2893 , 4559}};

dequantMat[6][3] = {{10, 16, 13}, {11, 18, 14}, {13, 20, 16}, {14, 23, 18}, {16, 25,
20}, {18, 29, 23}};
```

5.3.3.4 Scanning and Quantization of 2x2 chrominance 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'.

Chrominance 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 TABLE 7.

5.3.4 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

5.4 Deblocking Filter

After the reconstruction of a macroblock a conditional filtering of this macroblock is taking place, that effects the boundaries of the 4x4 block structure. Filtering is done on a macroblock level. In a first step the 16 pel of the 4 vertical edges (horizontal filtering) of the 4x4 raster are filtered. After that, the 4 horizontal edges (vertical filtering) follow. This process also affects the boundaries of the already reconstructed macroblocks above and to the right of the current macroblock. Frame edges are not filtered. Note, that intra prediction of the current macro block takes place on the unfiltered content of the already decoded neighbouring macroblocks. Depending on the implementation, these values have to be stored before filtering.

Note that when PSTRUCT indicates a field picture, then all calculations for the deblocking filter are based solely on pixels from just the current field; the pixels from the other field of the containing frame play no role at all.

5.4.1 Content Dependent thresholds

For the purpose of filtering each 4x4 block edge in a reconstructed macroblock a filtering **Boundary Strength Bs** is assigned for luma:

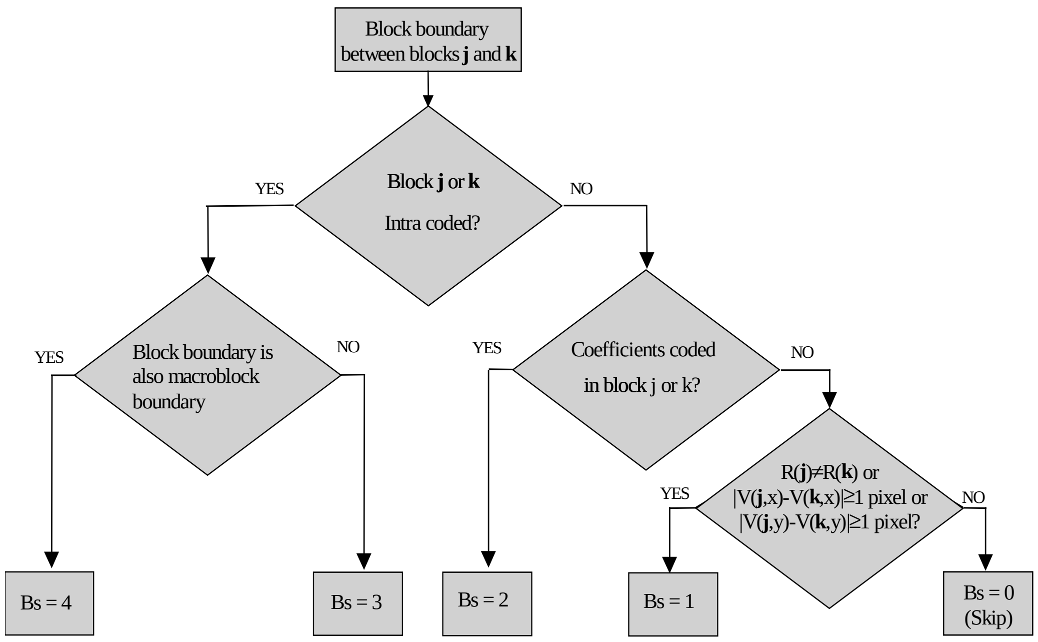

Block boundary between blocks **j** and **k**
Block **j** or **k** Intra coded?
YES
NO
Block boundary is also macroblock boundary
YES
NO
Coefficients coded in block j or k?
YES
NO
R(**j**)≠R(**k**) or |V(**j**,x)-V(**k**,x)|≥1 pixel or |V(**j**,y)-V(**k**,y)|≥1 pixel?
YES
NO
Bs = 4
Bs = 3
Bs = 2
Bs = 1
Bs = 0 (Skip)

FIGURE 14

Flow chart for determining boundary strength (Bs), for the block boundary between two neighbouring blocks j and k.

Block boundaries of chroma blocks always correspond to a block boundary of luma blocks. Therefore the corresponding **Bs** for luma is also used for chroma boundaries.

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

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

with the actual boundary between p1 and q1. In the default mode up to two pixels can be updated as a result of the filtering process on both sides of the boundary (that is at most p2, p1, q1, q2). Filtering across a certain 4x4 block boundary is skipped all together if the corresponding Bs is equal to zero. Sets of pixels across this edge are only filtered if the condition

$$Bs \neq 0 \quad \textbf{\textit{AND}} \quad |p_1 - q_1| < \alpha \quad \textbf{\textit{AND}} \quad |p_2 - p_1| < \beta \quad \textbf{\textit{AND}} \quad |q_2 - q_1| < \beta$$

is true. The QP dependant thresholds α and β can be found in table 5.

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
α	3	3	3	3	3	3	3	3	3	3	4	4	5	6	8	10	13	17	21	26	30	35	43	48	55	64	77	96	128	128	192	192
β	0	0	0	0	0	0	0	0	4	4	4	5	5	5	7	7	7	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15

TABLE 4

QP dependent threshold parameters α and β.

5.4.2 The Filtering Process

Two sorts of filter are defined. In the default case the following filter will be used for p_1 and q_1

$\Delta \quad = \quad Clip(-C, \quad C, \quad ((q_1 - p_1) << 2 \quad + \quad (p_2 - q_2) \quad + \quad 4) \quad >> \quad 3)$

$P_1 \quad = \quad Clip(\ 0, \quad 255, \quad (p_1 + \Delta))$

$Q_1 \quad = \quad Clip(\ 0, \quad 255, \quad (q_1 - \Delta))$

The two intermediate threshold variables

$a_p \quad = \quad |p3 - p1|$
$a_q \quad = \quad |q3 - q1|$
are used to decide whether p_2 and q_2 are filtered. These pixels are only processed for luminance.

If (Luma && $a_p < \beta$) p_2 is filtered by:

$P_2 \quad = \quad p_2 + Clip(\ -C0, \quad C0, \ (p_3 + P_1 \ - \ p_2 << 1) \ >> \ 1)$

If (Luma && $a_q < \beta$) q_2 is filtered by:

$Q_2 \quad = \quad q_2 \ + \ Clip(-C0, \quad C0, \ (q_3 + Q_1 \ - \ q_2 << 1) \ >> \ 1)$

where Clip() denotes a clipping function with the parameters Clip(Min, Max, Value) and the clipping thesholds:

$C0 \quad = \quad ClipTable[\ QP, \ Bs \ ]$ *(see TABLE 5)*

C is set to *C0* and than incremented by one if p_2 will be filtered and again by one if q_2 will be filtered.

TABLE 5

QP and Bs dependent Clipping thresholds

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
ClipTable[QP,1]	0	0	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	4	4
ClipTable[QP,2]	0	0	0	0	0	0	0	0	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3	4	4	4	4	5	5	6	6
ClipTable[QP,3]	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	3	3	3	3	4	4	4	5	5	6	6	6	8	8	9	10
ClipTable[QP,4]	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	3	3	3	3	4	4	4	5	5	6	6	6	8	8	9	10

5.4.3 Stronger filtering for intra coded blocks

In case of intra coding and areas in the picture with little texture the subjective quality can be improved by stronger filtering across block boundaries (that is: *not* 4x4 block boundaries). This stronger filtering is performed if:

$(Bs == 4) \quad \textbf{AND} \quad (a_p < \beta) \quad \textbf{AND} \quad (a_q < \beta) \quad \textbf{AND} \quad (1 \ < \ |p_1 - q_1| \ < \ QP / 4)$

In this case filtering is performed with the equations shown below.

$$P_1 = (p_3 + 2*p_2 + 2*p_1 + 2*q_1 + q_2 + 4) \ / \ 8$$
$$P_2 = (p_4 + 2*p_3 + 2*p_2 + 2*p_1 + q_1 + 4) \ / \ 8$$
$$Q_1 = (p_2 + 2*p_1 + 2*q_1 + 2*q_2 + q_3 + 4) \ / \ 8$$
$$Q_2 = (p_1 + 2*q_1 + 2*q_2 + 2*q_3 + q_4 + 4) \ / \ 8$$

Only for luma p3 and q3 are filtered:

$$P_3 = (2*p_4 + 3*p_3 + 2*p_2 + p_1 + 4) \ / \ 8$$
$$Q_3 = (2*q_4 + 3*q_3 + 2*q_2 + q_1 + 4) \ / \ 8$$

5.5 Entropy Coding

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

```
        1
      0 1 x0
    0 0 1 x1 x0
```

```
  0 0 0 1 x2 x1 x0
0 0 0 0 1 x3 x2 x1 x0
  .................
```

where xn take values 0 or 1. We will sometimes refer to a codeword with its length in bits (L = 2n-1) and INFO = xn,…x1, x0. Notice that the number of bits in INFO is n-1 bits. 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)

TABLE 6

Code number and Codewords in explicit form

Code_number	Code word
0	1
1	0 1 0
2	0 1 1
3	0 0 1 0 0
4	0 0 1 0 1
5	0 0 1 1 0
6	0 0 1 1 1
7	0 0 0 1 0 0 0
8	0 0 0 1 0 0 1
9	0 0 0 1 0 1 0
10	0 0 0 1 0 1 1
......	. . . .

When L (L = 2N-1) and INFO is known, the regular structure of the table makes it easy to create a codeword. Similarly, a decoder may easily decode a codeword by reading in N bit prefix followed by N-1 INFO. 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 7 lists the connection between code number and most of the parameters used in the present coding method.

TABLE 7

Connection between codeword number and parameter values.

Code_ number	RUN	MB_Type		8x8 mode	MVD D-QUANT	CBP		Tcoeff_chroma_ DC[1]		Tcoeff_chroma _AC[1] Tcoeff_luma[1] Simple scan		Tcoeff_luma[1] Double scan	
		Intra	Inter			Intra	Inter	Level	Run	Level	Run	Level	Run
0	0	Intra4x4	16x16	8x8	0	47	0	EOB	-	EOB	-	EOB	-
1	1	0,0,0[2]	16x8	8x4	1	31	16	1	0	1	0	1	0
2	2	1,0,0	8x16	4x8	-1	15	1	-1	0	-1	0	-1	0
3	3	2,0,0	8x8	4x4	2	0	2	2	0	1	1	1	1
4	4	3,0,0	8x8 (ref=0)	Intra	-2	23	4	-2	0	-1	1	-1	1
5	5	0,1,0	Intra4x4		3	27	8	1	1	1	2	2	0
6	6	1,1,0	0,0,0[2]		-3	29	32	-1	1	-1	2	-2	0
7	7	2,1,0	1,0,0		4	30	3	3	0	2	0	1	2
8	8	3,1,0	2,0,0		-4	7	5	-3	0	-2	0	-1	2
9	9	0,2,0	3,0,0		5	11	10	2	1	1	3	3	0
10	10	1,2,0	0,1,0		-5	13	12	-2	1	-1	3	-3	0
11	11	2,2,0	1,1,0		6	14	15	1	2	1	4	4	0
12	12	3,2,0	2,1,0		-6	39	47	-1	2	-1	4	-4	0
13	13	0,0,1	3,1,0		7	43	7	1	3	1	5	5	0
14	14	1,0,1	0,2,0		-7	45	11	-1	3	-1	5	-5	0
15	15	2,0,1	1,2,0		8	46	13	4	0	3	0	1	3
16	16	3,0,1	2,2,0		-8	16	14	-4	0	-3	0	-1	3
17	17	0,1,1	3,2,0		9	3	6	3	1	2	1	1	4
18	18	1,1,1	0,0,1		-9	5	9	-3	1	-2	1	-1	4
19	19	2,1,1	1,0,1		10	10	31	2	2	2	2	2	1
20	20	3,1,1	2,0,1		-10	12	35	-2	2	-2	2	-2	1
21	21	0,2,1	3,0,1		11	19	37	2	3	1	6	3	1
22	22	1,2,1	0,1,1		-11	21	42	-2	3	-1	6	-3	1
23	23	2,2,1	1,1,1		12	26	44	5	0	1	7	6	0
24	24	3,2,1	2,1,1		-12	28	33	-5	0	-1	7	-6	0
25	25		3,1,1		13	35	34	4	1	1	8	7	0
26	26		0,2,1		-13	37	36	-4	1	-1	8	-7	0
27	27		1,2,1		14	42	40	3	2	1	9	8	0
28	28		2,2,1		-14	44	39	-3	2	-1	9	-8	0
29	29		3,2,1		15	1	43	3	3	4	0	9	0
30	30				-15	2	45	-3	3	-4	0	-9	0
31	31				16	4	46	6	0	5	0	10	0
32	32				-16	8	17	-6	0	-5	0	-10	0
33	33				17	17	18	5	1	3	1	4	1
34	34				-17	18	20	-5	1	-3	1	-4	1
35	35				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]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).

[2]16x16 based intra mode. The 3 numbers refer to values for (Imode,AC,nc) - see 0.

TABLE 8

Connection between codeword number and Intra Prediction Mode Probability

Code_ number	Prob0, Prob1[3]	Code_ number	Prob0, Prob1[3]	Code_ number	Prob0, Prob1[3]	Code_ number	Prob0, Prob1[3]
0	0,0	21	2,3	41	2,6	61	6,5
1	0,1	22	3,2	42	6,2	62	4,7
2	1,0	23	1,5	43	3,5	63	7,4
3	1,1	24	5,1	44	5,3	64	3,8
4	0,2	25	2,4	45	1,8	65	8,3
5	2,0	26	4,2	46	8,1	66	4,8
6	0,3	27	3,3	47	2,7	67	8,4
7	3,0	28	0,7	48	7,2	68	5,7
8	1,2	29	7,0	49	4,5	69	7,5
9	2,1	30	1,6	50	5,4	70	6,6
10	0,4	31	6,1	51	3,6	71	6,7
11	4,0	32	2,5	52	6,3	72	6,7
12	3,1	33	5,2	53	2,8	73	5,8
13	1,3	34	3,4	54	8,2	74	8,5
14	0,5	35	4,3	55	4,6	75	6,8
15	5,0	36	0,8	56	6,4	76	8,6
16	2,2	37	8,0	57	5,5	77	7,7
17	1,4	38	1,7	58	3,7	78	7,8
18	4,1	39	7,1	59	7,3	79	8,7
19	0,6	40	4,4	60	5,6	80	8,8
20	6,0						

[3] 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).

6 Context-based Adaptive Binary Arithmetic Coding (CABAC)

6.1 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.

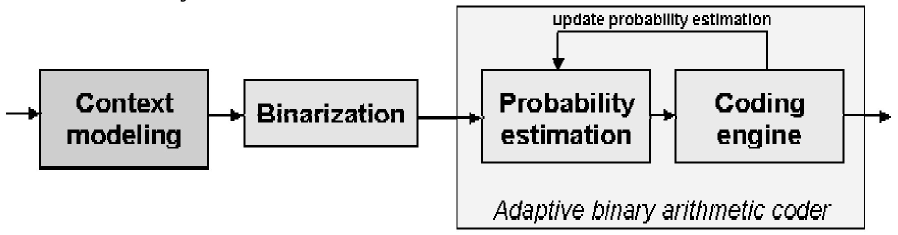


FIGURE 15

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 6.2 and Section 6.3. 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 6.6. 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 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).

6.2.1 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 6.6.

6.2.1.1 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 B[1] 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 + B*, which results in three 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 signalled. 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 6.6.

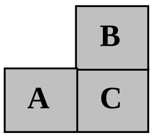


FIGURE 16

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

6.2.1.2 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 (see 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) + ((B==Skip)?0:1)* and by *ctx_mb_type_inter(C) = ((A==Direct)?0:1) + ((B==Direct)?0:1)* for B-frame coding. Thus, we obtain 3 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 6.6. 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 Section 6.6. Thus, a total number of 6 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.

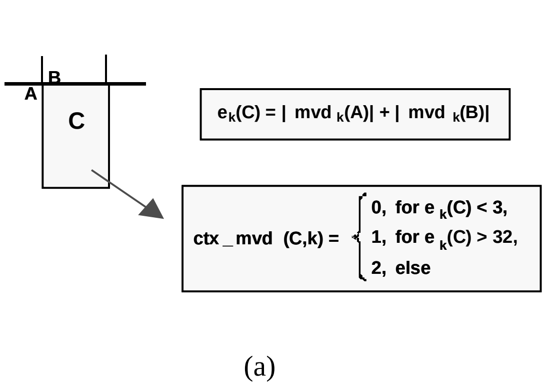
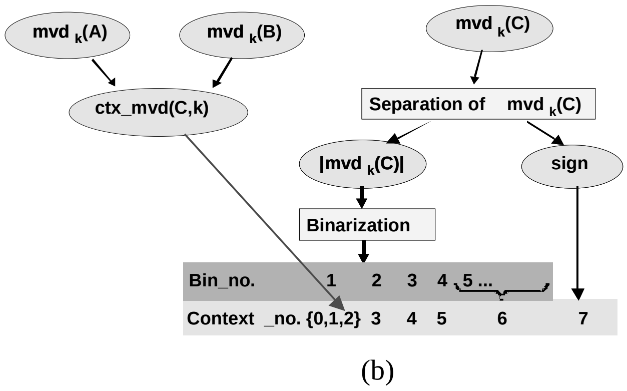

B
A
C
$e_k(C) = | mvd_k(A)| + | mvd_k(B)|$
$ctx\_mvd(C,k) = \begin{cases} 0, & \text{for } e_k(C) < 3, \\ 1, & \text{for } e_k(C) > 32, \\ 2, & \text{else} \end{cases}$
(a)


FIGURE 17

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

6.2.2 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:

$$ctx\_mvd(C,k) = \begin{cases} 0, e_k(C) < 3, \\ 1, e_k(C) > 32, \\ 2, else. \end{cases}$$

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 of the modulus, we have 4 additional models: one for the second, one for the third bin, one for the fourth, and one model for all remaining bins. In addition, the sign coding routine is provided with a separate model. This results in a total sum of 8 different models for each vector component.

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 6.6.

6.2.3 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 Section 4.4, a context model is built by using symbols A and B of the reference frame parameter belonging to the two neighboring macroblocks (see 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.

6.3 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.

6.3.1 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.

6.3.2 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 (see 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 described in Section 4.4.4. 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.

6.3.3 Context Models for Run/Level and Coeff_count

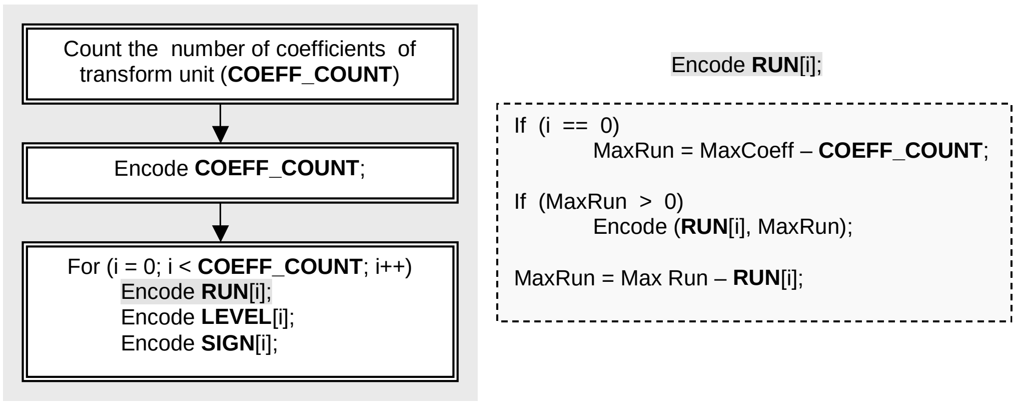
Coding of RUN and LEVEL is conditioned primarily on the scanning mode, the DC/AC block type, the luminance/chrominance, and the intra/inter macroblock decision. Thus, a total number of 8 different block types are given according to TABLE 9. In contrast to the UVLC coding mode, RUN and LEVEL are coded separately in the CABAC mode, and furthermore an additional coding element called COEFF_COUNT is introduced, which denotes the number of non-zero coefficients in a given block. The coefficients corresponding to a scanning mode are processed in the following way: First, the number of non-zero coefficients COEFF_COUNT is encoded. Then, in a second step, RUN, LEVEL and SIGN of all non-zero coefficients are encoded. FIGURE 18 illustrates this encoding scheme that is described in more detail in the following sections.

TABLE 9

Numbering of the different block types used for coding of RUN, LEVEL and COEFF_COUNT

Ctx_run_level_coeff_count	Block Type
0	Double Scan (Intra Luma only)
1	Single Scan (Inter Luma only)
2	Intra Luma 16x16, DC

3	Intra Luma 16x16, AC
4	Inter Chroma, DC
5	Intra Chroma, DC
6	Inter Chroma, AC
7	Intra Chroma, AC


Count the number of coefficients of transform unit (**COEFF_COUNT**)
Encode **COEFF_COUNT**;
Encode **RUN**[i];

```
For (i = 0; i < COEFF_COUNT; i++)
        Encode RUN[i];
        Encode LEVEL[i];
        Encode SIGN[i];
```

```
If (i == 0)
        MaxRun = MaxCoeff – COEFF_COUNT;

If (MaxRun > 0)
        Encode (RUN[i], MaxRun);

MaxRun = Max Run – RUN[i];
```

FIGURE 18

Coding scheme for transform coefficients

6.3.3.1 Context-based Coding of COEFF_COUNT Information

For capturing the correlations between COEFF_COUNTs of neighboring blocks appropriate context models are designed, which are only applied to the first binary decision (*bin*) in the unary binarization tree. More specifically, first the COEFF_COUNT is classified according the block type and then a local context for the first bin is built according to the rules specified in TABLE 10. The remaining bins are coded with two separate models; one for the second and the other for all remaining bins.

TABLE 10

Description of the context formation for the first bin for coding of COEFF_COUNT (CC denotes the COEFF_COUNT of the corresponding blocks)

Block Type	*Number of ctx's*	*Description of the context formation for the first bin*	*Comments*

0	3	ctx_coeff_count(A) =0;
		ctx_coeff_count(B) = (CC(A)==0) ? 1 : 2;
			A: upper scan path;
			B: lower scan path

1	4	ctx_coeff_count(C) = ((CC(A)==0)?0:1) + 2*((CC(B)==0)?0:1);	A,B,C as in **Error! Reference source not found.**
2	1	Only one ctx	

3	3	ctx_coeff_count(0) =0;
		ctx_coeff_count(i) = (CC(i-1)==0) ? 1 : 2; i={1,..,15}
			i denotes the number of the 4x4 block as in **FIGURE 3**

4,5,6,7	2	ctx_coeff_count(U) == 0;
		ctx_coeff_count(V) == 1;
			U, V : the different chroma parts

6.3.3.2 Context-based Coding of RUN Information

For encoding the RUN, the information of the initially encoded COEFF_COUNT and all previously encoded RUN elements is exploited, as shown in the pseudo-C code of FIGURE 18. Here, we make use of the fact that, if the maximum number of a RUN range is known in advance, the binarization can be restricted to a truncated code tree, which is explained in section 4.5.2.3.5. However, by using the information given by previously encoded RUNs the *MaxRun* counter can further be adapted on the fly, which may lead to potentially shorter binarized codewords of subsequent RUN elements to encode. In some cases, where, for instance, coefficients are aggregated at the end of the scan path, signaling of zero-valued RUN elements is completely omitted.

Context models for coding of RUN information depend on a threshold decision involving mostly the coefficient counter, *e.g.* in 4x4 single scan inter coding mode, the context model *ctx_run* for luma is given by *ctx_run = ((COEFF_COUNT) >= 4) ? 1 : 0.* For a more detailed description refer to Table 11. The idea behind this context design is that the COEFF_COUNT represents the activity of the given block, and that the probability distribution of the RUN symbol somehow depends on the activity of the block. First, the RUN is classified according the given block type, then a context *ctx_run* is chosen according to Table 11. For each context *ctx_run* 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 codeword related to RUN.

Table 11

Description of the context formation for RUNs
(the variable i denotes the coefficient counter ($i \in \{0,..,(COEFF\_COUNT-1)\}$)

Block Type	***Description of the context formation for the runs)***
0,3,6,7	ctx_run = ((COEFF_COUNT-i) >= 3) ? 1 : 0;
1	ctx_run = (COEFF_COUNT >= 4) ? 1 : 0;
2	ctx_run = ((COEFF_COUNT-i) >= 4) ? 1 : 0;
4,5	ctx_run = ((COEFF_COUNT-i) >= 2) ? 1 : 0;

6.3.3.3 Context-based Coding of LEVEL Information

LEVEL information is first separated into sign and magnitude. Then the magnitude and sign of LEVEL, are first classified based on the block type. For coding of *ABS_LEVEL=abs(LEVEL)* context models depending on the previously encoded or decoded *ABS_LEVEL* within a given block are applied. More specifically, the following context model is defined for bins of the magnitude *ABS_LEVEL:*

Context Model for ABS_LEVEL

```
if (bin_nr>3)
        bin_nr=3;
end
if (Prev_level>MAX_LEVEL)
        Prev_level=MAX_LEVEL;
end
context_nr = (bin_nr-1)*MAX_LEVEL+Prev_level,
```

where *MAX_LEVEL=3,* and where the *Prev_level* is initialized to zero at the beginning of each block and updated after each en-/decoding step by the most recently en-/decoded ABS_LEVEL value. Note that for double-scan, *Prev_level* is initialized at the beginning of each scan, i.e., twice per block.

6.4 Double Scan Always for CABAC Intra Mode

In the UVLC mode, the scan mode for intra coding depends on the QP value: for QP < 24 the double scan is used, whereas for QP≥24 the single scan mode is employed. In the contrast to that, CABAC entropy coding always uses the double scan mode, independently of the given QP value.

6.5 Context Modeling for Coding of Dquant

For a given macroblock, the value of Dquant is first mapped to a positive value using the arithmetic wrap as described in section 4.4.8. This wrapped value *C* is then coded conditioned on the corresponding Dquant *A* of the left neighboring macroblock, such that that *ctx_dquant(C)=(A !=0);* This results in 2 context models for the first bin. Two additional models are used for the second and all remaining bins of the binarized value *C*. Thus, a total sum of four context models are used for Dquant.

TABLE 11

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**	. .

6.6 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 12.

TABLE 12

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

P_MB_type	Binarization
0	0
1	1 0 0
2	1 0 1
3	1 1 0 0 0
4	1 1 0 0 1
5	1 1 0 1 0
6	1 1 0 1 1
7	1 1 1 0 0
8	1 1 1 0 1
9	1 1 1 1 0
Bin_no	**1 2 3 4 5**

B_MB_type	Binarization
0	0
1	1 0 0
2	1 0 1
3	1 1 0 0 0
4	1 1 0 0 1
5	1 1 0 1 0
6	1 1 0 1 1
7	1 1 1 0 0 0 0
.	.
17	1 1 1 1 0 1 0
Bin_no	**1 2 3 4 5 6 7**

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 14.

6.6.1 Truncated Binarization for RUN, COEFF_COUNT and Intra Prediction Mode

If the maximum number of a given syntax element is known in advance, the binarization can be restricted to a truncated code tree. Suppose the alphabet size of a specific syntax element is fixed, then for the unary binarization of the maximum value of this syntax element the terminating "1" can be omitted. This mechanism applies for RUN, COEFF_COUNT and Intra Prediction Mode.

6.7 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 "Arithmetic Coding Revisited", Moffat, Neal, Witten. ACM Transactions on Information Systems, 16(3):256-294, July 1998.

[Editor: This must be improved by the area assistant editor.]

7 B-pictures

7.1 Introduction

Temporal scalability is achieved using bi-directionally predicted pictures, or B pictures. The use of B pictures is indicated in PTYPE. The B pictures are predicted from either or both the previous and subsequent reconstructed pictures to achieve improved coding efficiency as compared to that of P pictures. FIGURE 19 illustrates the predictive structure with two B pictures inserted between I/P pictures.

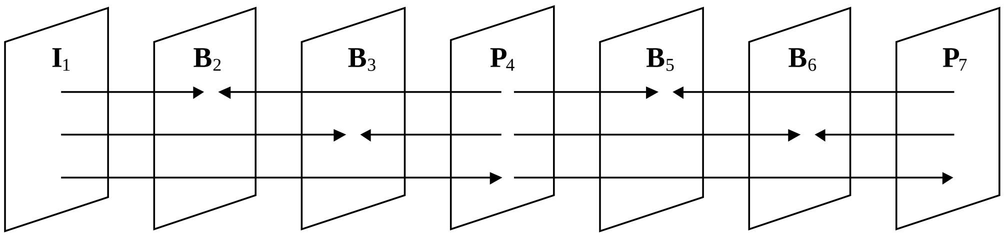


FIGURE 19

Illustration of B picture concept.

The location of B pictures in the bitstream is in a data-dependence order rather than in temporal order. Pictures that are dependent on other pictures shall be located in the bitstream after the pictures on which they depend. For example, as illustrated in Figure A.1, B_2 and B_3 are dependent on I_1 and P_4, and B_5 and B_6 are dependent on P_4 and P_7. Therefore the bitstream syntax order of the encoded pictures would be I_1, P_4, B_2, B_3, P_7, B_5, B_6, … However, the display order of the decoded pictures should be I_1, B_2, B_3, P_4, B_5, B_6, P_7, … The difference between the bitstream order of encoded pictures and the display order of decoded pictures will increase latency and memory to buffer the P pictures.

There is no limit to the number of B pictures that may be inserted between each I/P picture pair. The maximum number of such pictures may be signaled by external means (for example Recommendation H.245). The picture height, width, and pixel aspect ratio of a B picture shall always be equal to those of its temporally subsequent reference picture.

The B pictures support multiple reference frame prediction. The maximum number of previous reference frames that may be used for prediction in B pictures must be less than or equal to the number of reference frames used in the immediately following P frame, and it may be signaled by external means (for example Recommendation H.245). The use of this mode is indicated by PTYPE.

7.2 Macroblock modes and 8x8 sub-partition modes

There are five different prediction types supported by B pictures. They are direct, forward, backward, bi-directional and the intra prediction modes. Both direct mode and bi-directional mode are bi-directional prediction. The only difference is that the bi-directional mode uses separate motion vectors for forward and backward prediction, whereas the forward and backward motion vectors of the direct mode are derived from the motion vectors used in the corresponding macroblocks of the subsequent reference frame. In the direct mode, the same number of motion vectors as in the co-located macroblock of the temporal following reference picture is used. To calculate prediction blocks for the direct and bi-directional prediction mode, the forward and backward motion vectors are used to obtain appropriate blocks from reference frames and then these blocks are averaged by dividing the sum of the two prediction blocks by two.

Forward prediction means prediction from a previous reference picture, and backward prediction means prediction from a temporally subsequent reference picture.

The intra prediction means to encode the macroblock by using intra coding.

For B-frames, a similar tree-structured macroblock partitioning as in P-pictures is used. As in P-frames, one reference frame parameter is transmitted for each 16x16, 16x8, and 8x16 block as well as for each 8x8 sub-partition. Additionally, for each 16x16, 16x8, 8x16 block, and each 8x8 sub-partition, the prediction direction (forward, backward, bi-directional) can be chosen separately. For avoiding a separate code word to specify the prediction direction, the indication of the prediction direction is incorporated into the codewords for macroblock modes and 8x8 partitioning modes, respectively, as shown in the table Table 12 and Table 13. Moreover, an 8x8 sub-partition of a B-frame macroblock can also be coded in Direct mode.

Table 12: Macroblock modes for B-frames

Code number	Macroblock mode	1. block	2. block
0	Direct		
1	16x16	Forw.	
2	16x16	Backw.	
3	16x16	Bidirect.	
4	16x8	Forw.	Forw.
5	8x16	Forw.	Forw.
6	16x8	Backw.	Backw.
7	8x16	Backw.	Backw.
8	16x8	Forw.	Backw.
9	8x16	Forw.	Backw.
10	16x8	Backw.	Forw.
11	8x16	Backw.	Forw.
12	16x8	Forw.	Bidirect.
13	8x16	Forw.	Bidirect.
14	16x8	Backw.	Bidirect.
15	8x16	Backw.	Bidirect.
16	16x8	Bidirect.	Forw.
17	8x16	Bidirect.	Forw.
18	16x8	Bidirect.	Backw.
19	8x16	Bidirect.	Backw.
20	16x8	Bidirect.	Bidirect.
21	8x16	Bidirect.	Bidirect.
22	8x8(split)		
23	Intra4x4		
24 …	Intra16x16		

Table 13: Modes for 8x8 sub-partitions in B-frames

Code number	8x8 partition mode	Prediction
0	Direct	
1	8x8	Forw.
2	8x8	Backw.
3	8x8	Bidirect.
4	8x4	Forw.
5	4x8	Forw.
6	8x4	Backw.
7	4x8	Backw.
8	8x4	Bidirect.
9	4x8	Bidirect.
10	4x4	Forw.
11	4x4	Backw.

12	4x4	Bidirect.
13	Intra	

7.3 Syntax

Some additional syntax elements are needed for B pictures. The structure of B picture related fields is shown in FIGURE 20. On the Ptype, two picture types shall be added to include B pictures with and without multiple reference frame prediction. On the MB_type, different macroblock types are defined to indicate the different prediction types for B pictures. The fields of MVDFW, and MVDBW are inserted to enable bi-directional prediction. These fields replace the syntax element MVD.

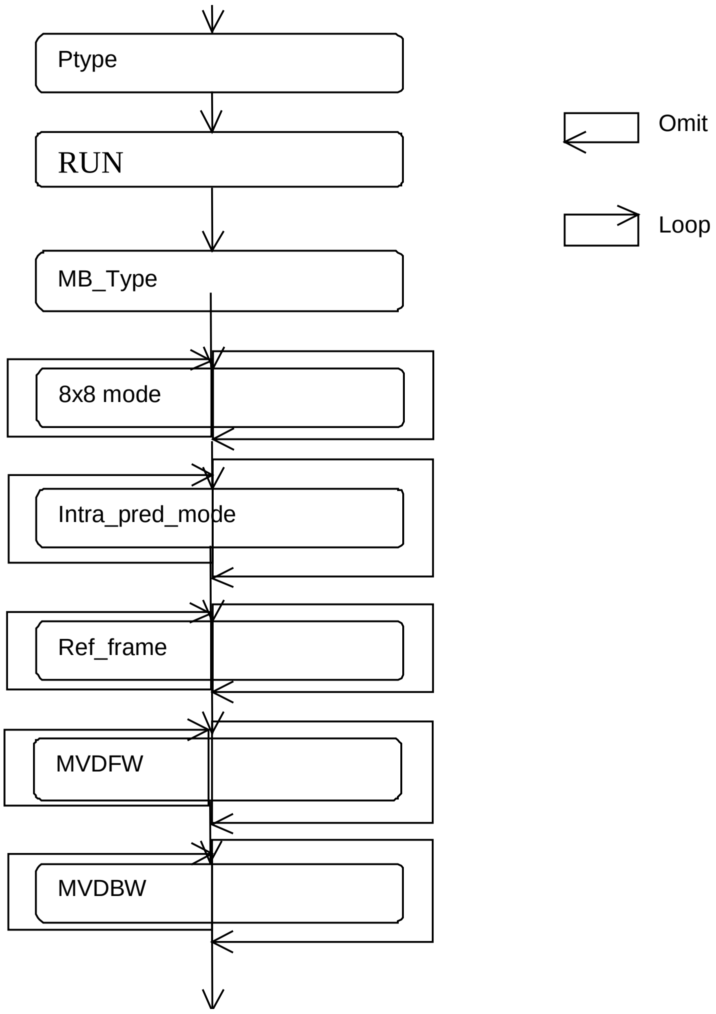


FIGURE 20

Syntax diagram for B pictures.*[Editor: Add PSTRUCT]*

7.3.1 Picture type (Ptype)), Picture Structure (PSTRUCT) and RUN

See section 3.1 for definition.

7.3.2 Macro block type (MB_type) and 8x8 sub-partition type

Table 12shows the macroblock modes for B pictures.

In “Direct” prediction type, no motion vector data is transmitted.

In NxM mode, each NxM block of a macroblock is predicted by using different motion vectors, reference frames, and prediction directions. As indicated in Table 12, three different macroblock modes that differ in their prediction directions exist for the 16x16 mode. For the 16x8 and 8x16 macroblock modi, 9 different combinations of the prediction directions are possible. If a macroblock is transmitted in 8x8 mode, an additional codeword for each 8x8 sub-partition indicates the decomposition of the 8x8 block as well as the chosen prediction direction (see Table 13).

.

The “Intra_4x4” and “Intra_16x16” prediction type indicates that the macroblock is encoded by intra coding with different intra prediction modes which are defined in the same manner as in section 4.4.4 and TABLE 7. No motion vector data is transmitted for the intra macroblocks or 8x8 sub-partition coded in intra mode.

7.3.3 Intra prediction mode (Intra_pred_mode)

As present, Intra_pred_mode indicates which intra prediction mode is used. Intra_pred_mode is present when Intra_4x4 prediction type is indicated in the MB_type or the 8x8 sub-partition type. The code_number is same as that described in the Intra_pred_mode entry of TABLE 7.

7.3.4 Reference Picture (Ref_picture)

At present, Ref_picture indicates the position of the reference picture in the reference picture buffer to be used for forward motion-compensated for current macroblock. The reference frame parameter is transmitted for each 16x16, 16x8, 8x16 block, and each 8x8 sub-partition if forward or bi-directional prediction is used. Ref_picture is present only when the Ptype signals the use of multiple reference frames. Decoded I/P pictures are stored in the reference picture buffer in first-in-first-out manner and the most recently decoded I/P picture is always stored at position 0 in the reference frame buffer. The code_number and interpretation for Ref_picture differs for PSTRUCT indicating field picture or frame picture and is the same as the definition for REF_picture in section 3.4.4.

Note that whenever PSTRUCT indicates that the current picture is a field picture and when the present MB_type indicates Backward_NxM or Bi-directional prediction type, the backward reference is the immediately following I or P field of the same field parity (in display order). If PSTRUCT indicates a frame, then the backward reference is the immediately following I or P frame (in display order).

TABLE 14

Code_number for ref_frame

Code_number	Reference frame
0	The most recent previous frame (1 frame back)
1	2 frames back
2	3 frames back
…	…

7.3.5 Motion vector data (MVDFW, MVDBW)

MVDFW is the motion vector data for the forward vector, if present. MVDBW is the motion vector data for the backward vector, if present. If so indicated by MB_type and/or 8x8 sub-partition type, vector data for 1-16 blocks are transmitted. The order of transmitted motion vector data is the same as thatindicated in FIGURE 2. For the code_number of motion vector data, please refer to TABLE 7.

7.4 Decoder Process for motion vector

7.4.1 Differential motion vectors

Motion vectors for forward, backward, or bi-directionally predicted blocks are differentially encoded. A prediction has to be added to the motion vector differences to get the motion vectors for the macroblock.

The predictions are formed in way similar to that described in section 4.4.6. The only difference is that forward motion vectors are predicted only from forward motion vectors in surrounding macroblocks, and backward motion vectors are predicted only from backward motion vectors in surrounding macroblocks.

If a neighboring macroblock does not have a motion vector of the same type, the candidate predictor for that macroblock is set to zero for that motion vector type.

7.4.2 Motion vectors in direct mode

In direct mode the same block structure as for the co-located macroblock in the temporally subsequent picture is used. For each of the sub-blocks the forward and backward motion vectors are computed as scaled versions of the corresponding vector components of the co-located macroblock in the temporally subsequent picture as described below.

If the multiple reference frame prediction is used, the forward reference frame for the direct mode is the same as the one used for the corresponding macroblock in the temporally subsequent reference picture. The forward and backward motion vectors for direct mode macroblocks are calculated differently depending on whether PSTRUCT and the reference are fields or frames. Also note that if the subsequent reference picture is an intra-coded frame or the reference macroblock is an intra-coded block, the motion vectors are set to zero. With possible adaptive switch of frame/field coding at picture level, a B-frame or its future reference frame can be coded in either frame structure or field structure. Hence, there can be four different combinations of frame or field coding for a pair of a MB in B and its collocated MB in the future reference. Calculations of the two MVs in direct mode are slightly different for the four cases.

Case 1: Both the current MB and its collocated are in frame mode

Both the current B and its future reference are in frame structure, as shown in Fig. 4. The forward reference is the frame pointed by the forward MV of the collocated MB and the backward reference is the immediate future reference frame of I or P. Two MVs (MV_F, MV_B) are calculated by (see Fig. 4)

$$MV_F = TR_B \cdot MV / TR_D$$
$$MV_B = (TR_B - TR_D) \cdot MV / TR_D$$

where TR_B is the temporal distance between the current B frame and the reference frame pointed by the forward MV of the collocated MB, and TR_D is the temporal distance between the future reference frame and the reference frame pointed by the forward MV of the collocated MB.

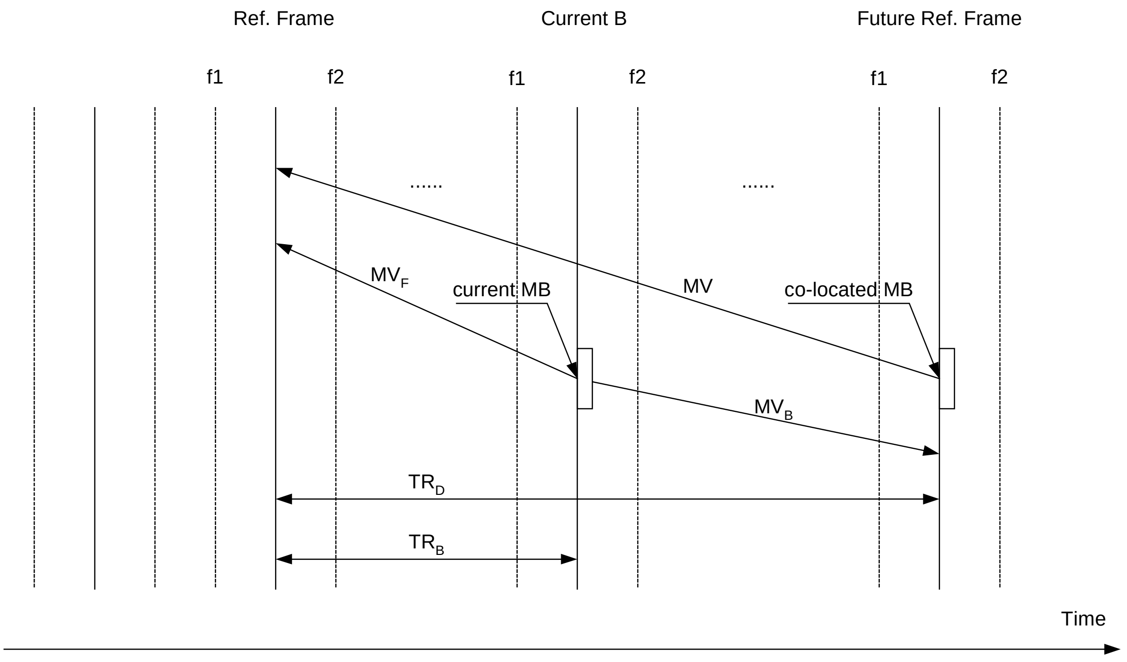


FIGURE 21

Both the current MB in B and its collocated MB in the future reference of I or P are in frame mode. Solid-line is for frames and dot-line for fields. f1 stands for field 1 and f2 for field 2.

Case 2: Both the current MB and its collocated MB are in field mode

Both the current B frame and its future reference are in field structure. Two MVs of direct mode MB in B field are calculated from the forward MV of the collocated MB in the future reference field of the same parity. For field 1, the forward MV of the collocated MB will always point to one of the previously coded I or P fields, as shown in Fig. 5. The forward reference field for the direct mode MB will be the same as pointed by the forward MV of the collocated MB, and the backward reference field will be field 1 of the future reference frame. The forward and backward MVs $(MV_{F,i}, MV_{B,i})$ for the direct mode MB are calculated as follows,

$$MV_{F,i} = TR_{B,i} \cdot MV_i / TR_{D,i}$$
$$MV_{B,i} = (TR_{B,i} - TR_{D,i}) \cdot MV_i / TR_{D,i}$$

where the subscript, i, is the field index (1 for the 1st field and 2 for the 2nd field), and MV_i is the forward MV of the collocated MB in field i of the future reference frame. $TR_{B,i}$ is the temporal distance between the current B field (i) and the reference field pointed by the forward MV of the collocated MB in field interval, $TR_{D,i}$ is the temporal distance between the future reference field (i) and the reference field pointed by the forward MV of the collocated MB in field interval.

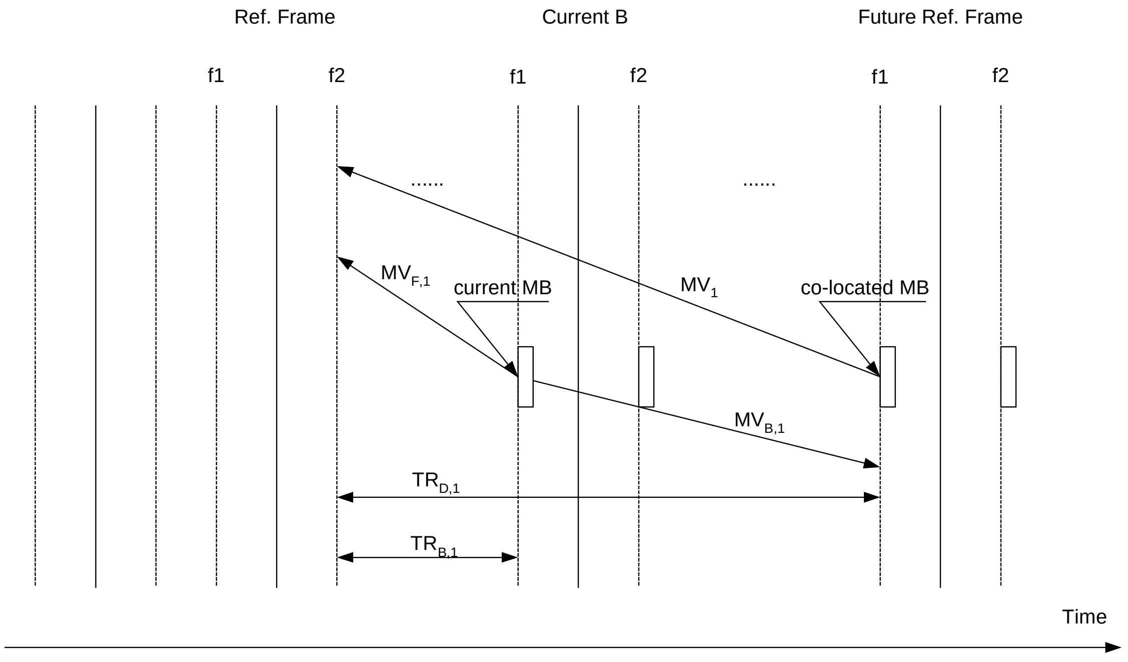


FIGURE 22

Both the current MB in B and its collocated MB in the future reference of I or P are in field mode. Solid-line is for frames and dot-line for fields. f1 stands for field 1 and f2 for field 2. The forward MV of the collocated MB in field 1 of the future reference always points to one of the previously coded I or P fields.

For field 2, the forward MV of the collocated MB may point to one of the previously coded I or P fields. Calculation of the forward and backward MVs therefore follows the same equation (2) above. However, it is also possible that the forward MV of the collocated MB in field 2 of the future reference frame points to field 1 of the same frame, as shown in Fig. 6. In this case, the two MVs $(MV_{F,2}, MV_{B,2})$ of the direct mode MB are calculated as follows:

$$MV_{F,2} = -TR_{B,2} \cdot MV_2 / TR_{D,2}$$
$$MV_{B,2} = -(TR_{B,2} + TR_{D,2}) \cdot MV_2 / TR_{D,2}$$

Note that both MVs $(MV_{F,2}, MV_{B,2})$ are now backward MVs, pointing to field 1 and field 2 of the future reference frame respectively.

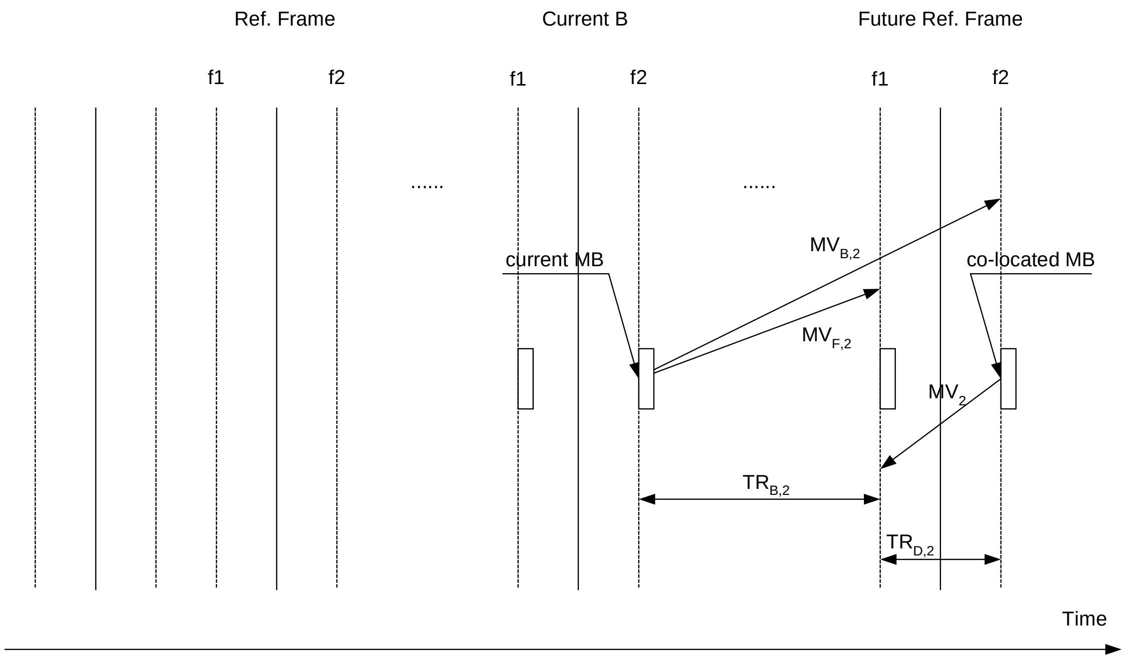


FIGURE 23

Both the current MB in B and its collocated MB in the future reference of I or P are in field mode. Solid-line is for frames and dot-line for fields. f1 stands for field 1 and f2 for field 2. The forward MV of the collocated MB in field 2 of the future reference frame may point to field 1 of the same frame.

Case 3:The current MB is field mode and its collocated MB in frame mode

The current B is in field structure while its future reference is in frame structure, as shown in Fig. 7. Let $y_{current\_MB}$ and $y_{collocated\_MB}$ be the notations of the vertical indexes of the current MB and its collocated MB respectively. Then, $y_{collocated\_MB} = 2y_{current\_MB}$. Two MVs of direct mode MB are calculated from the forward MV of the collocated MB in the future reference frame as follows

$$MV_{F,i} = TR_{B,i} \cdot MV / TR_D$$
$$MV_{B,i} = (TR_{B,i} - TR_D) \cdot MV / TR_D$$

Since the collocated MB is frame coded, MV in (4) is derived by dividing the associated frame-based MV by 2 in vertical direction. Forward reference field of direct mode MB is the same parity field in the reference frame used by its collocated MB. Backward reference field is the same parity field of the future reference frame.

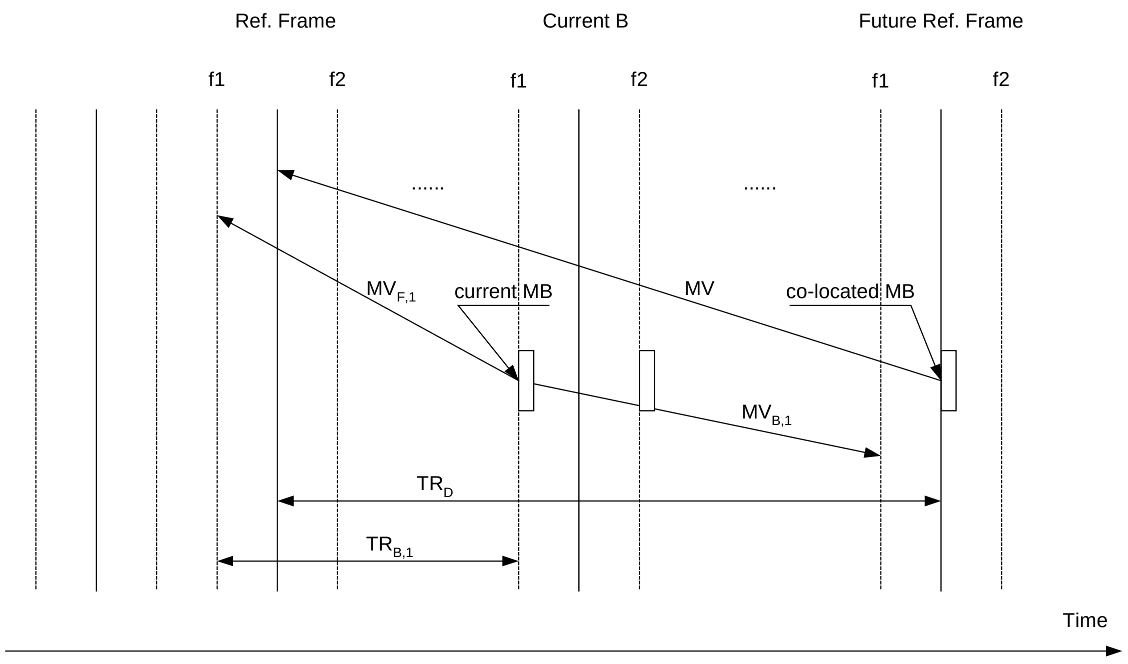


FIGURE 24

The current MB in B is in field mode while its collocated MB in the future reference of I or P in frame mode. Solid-line is for frames and dot-line for fields. f1 stands for field 1 and f2 for field 2.

Case 4: The current MB is in frame mode while its collocated MB in fieldmode

The current B is in frame structure while its future reference is in field structure, as shown in Fig. 8. Let $y_{current\_MB}$ and $y_{collocated\_MB}$ be the notations of the vertical indexes of the current MB and its collocated MB respectively. Then, $y_{collocated\_MB} = y_{current\_MB} / 2$. The two fields of the collocated MB in the future reference may be coded in different modes. Since field 1 of the future reference is temporally close to the current B, the collocated MB in field 1 of the future reference provides certain basis for the current MB. Hence, the collocated MB in field 1 of the future reference is used in calculating the MVs and determining the references for direct mode. Two frame-based MVs (MV_F, MV_B) of direct mode MB are calculated as follows

$$MV_F = TR_B \cdot MV_1 / TR_{D,1}$$
$$MV_B = (TR_B - TR_{D,1}) \cdot MV_1 / TR_{D,1}$$

where MV_1 is derived by doubling the field-based MV of the collocated MB in field 1 of the future reference in vertical direction, TR_B is the temporal distance between the reference frame of one of whose fields is pointed by the forward MV of the collocated MB in field 1 of the future reference and the current B frame in field interval, and $TR_{D,1}$ is the temporal distance between field 1 of the future reference and the reference field pointed by the forward MV of the collocated MB in field 1 of future reference. Forward reference frame of the direct mode MB is the frame of one of whose fields is pointed by the forward field MV of the collocated MB in field 1 of the future reference frame. Backward reference frame is the future reference frame.

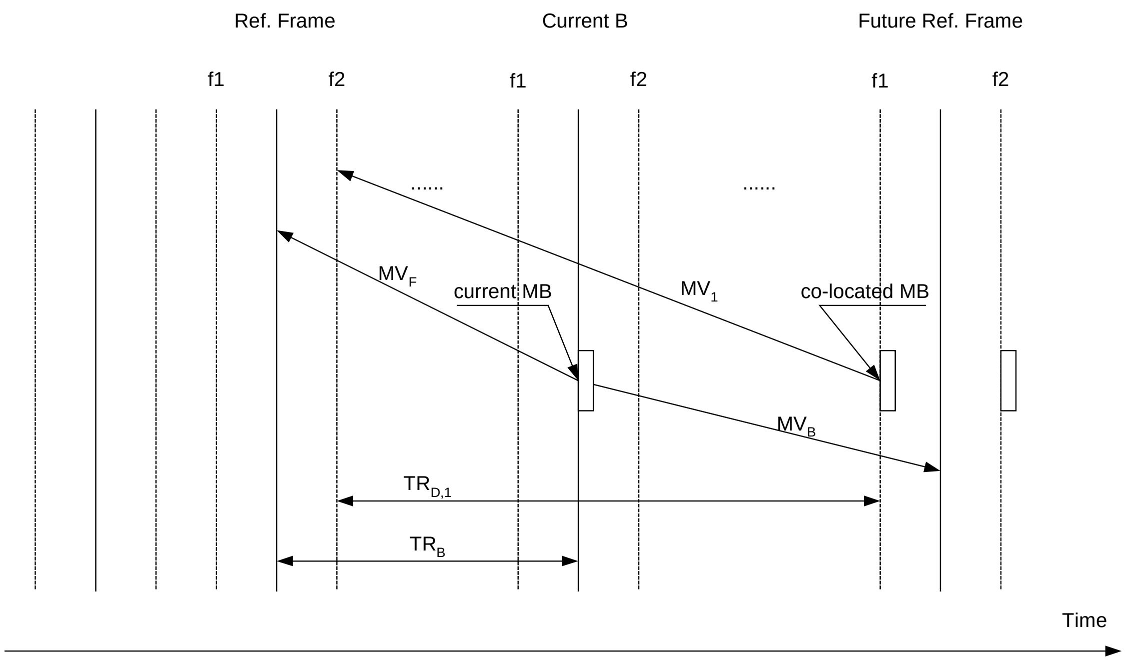


FIGURE 25

The current MB in B is in frame mode while its collocated MB in the future reference of I or P is in field mode. Solid-line is for frames and dot-line for fields. f1 stands for field 1 and f2 for field 2.

8 SP-pictures

There are two types of S-pictures, namely SP-pictures and SI-pictures. SP-pictures make use of motion-compensated predictive coding to exploit temporal redundancy in the sequence similar to P-pictures and SI-pictures make use of spatial prediction similar to I-pictures. Unlike P-pictures, however, SP-picture coding allows identical reconstruction of a frame even when different reference frames are being used. SI-picture, on the other hand, can identically reconstruct a corresponding SP-picture. These properties of S-pictures provide functionalities for bitstream switching, splicing, random access, VCR functionalities such as fast-forward and error resilience/recovery.

8.1 Syntax

The use of S-pictures is indicated by PTYPE. If PTYPE indicates an S-picture, quantization parameter PQP is followed by an additional quantization parameter SPQP in the slice header, see Section **Error! Reference source not found.**. The rest of the syntax elements for SP-pictures are the same as those in Inter-frames. Similarly, the rest of the syntax elements for SI-pictures are the same as those in I-pictures.

8.1.1 Picture type (Ptype) and RUN

See Section **Error! Reference source not found.** for definition.

8.1.2 Macro block type (MB_Type)

MB_Type indicates the prediction mode and the block size used to encode each macroblock. There are different MB types for SP and SI-pictures.

8.1.3 Macroblock modes for SI-pictures

The MB types for SI-pictures are similar to those of I-pictures with an additional MB type, called SIntra 4x4. TABLE 15 depicts the relationship between the code numbers and MB_Type for SI-pictures.

SIntra 4x4	SI Mode.
Intra 4x4	4x4 Intra coding.
Imode, nc, AC	See definition in Section **Error! Reference source not found.**. These modes refer to 16x16 intra coding.

TABLE 15

MB Type for SI-Pictures

Code_ number	MB_Type (SI-pictures)
0	SIntra 4x4
1	Intra 4x4
2	0,0,0[1]
3	1,0,0
4	2,0,0
5	3,0,0
6	0,1,0
7	1,1,0
8	2,1,0
9	3,1,0
10	0,2,0
11	1,2,0
12	2,2,0
13	3,2,0
14	0,0,1
15	1,0,1
16	2,0,1
17	3,0,1
18	0,1,1
19	1,1,1
20	2,1,1
21	3,1,1
22	0,2,1

23	1,2,1
24	2,2,1
25	3,2,1

[1] 16x16 based intra mode. The 3 numbers refer to values for (Imode,AC,nc) - see **Error! Reference source not found.**.

8.1.4 Macroblock Modes for SP-pictures

The MB types for SP-pictures are identical to those of P-pictures, see Section **Error! Reference source not found.** and TABLE 7. However, all of the inter mode macroblocks, i.e., Skip and NxM, in SP frames refer to SP mode.

8.1.5 Intra Prediction Mode (Intra_pred_mode)

Intra_pred_mode is present when Intra 4x4 or SIntra 4x4 prediction types are indicated by the MB_Type. Intra_pred_mode indicates which intra prediction mode is used for a 4x4 block in the macroblock. The code_number for SIntra 4x4 is the same as that described in the Intra_pred_mode entry of TABLE 7.

Coding of Intra 4x4 prediction modes

Coding of the intra prediction modes for SIntra 4x4 blocks in SI-pictures is identical to that in I-pictures, i.e., the intra prediction modes of the neighboring blocks are taken into account as described in Section 3.4.3.1.1. Coding of the intra prediction modes for Intra 4x4 blocks in SI-pictures differs from I-picture coding if the neighboring block is coded in SIntra 4x4 mode. In this case the prediction mode of the neighboring Sintra 4x4 block is treated to be “mode 0: DC_prediction”.

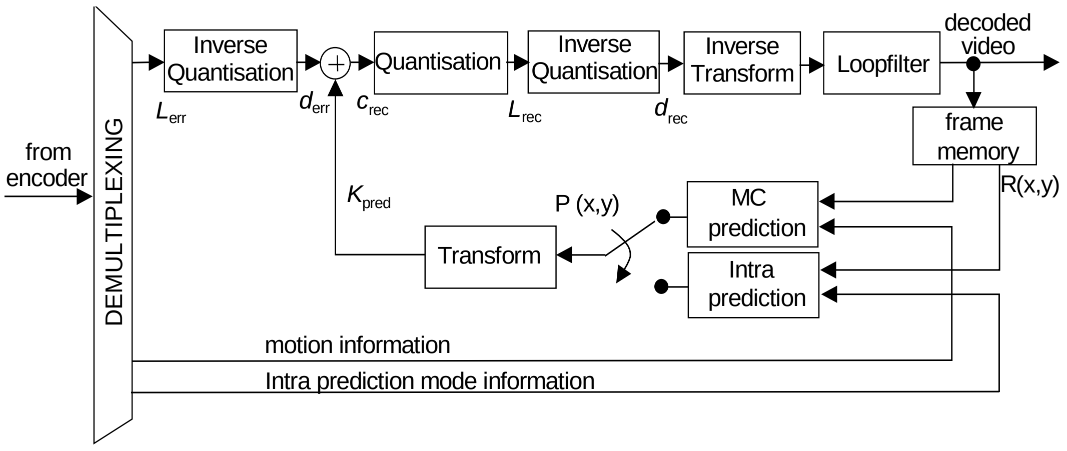


FIGURE 26

A generic block diagram of S-picture decoder

8.2 S-frame decoding

A video frame in SP-picture format consists of blocks encoded in either Intra mode (Intra 4x4, Intra 8x8 or Intra 16x16 Modes) or in SP mode (Skip or NxM). Similarly, SI-pictures consist of macroblocks encoded in either Intra mode or in SI mode (SIntra 4x4). Intra macroblocks are decoded identically to those in I- and P-pictures. FIGURE 26 depicts a generic S-picture decoding for SI and SP modes. In SP mode, the prediction P(x,y) for the current macroblock of the frame being decoded is formed by the motion compensated prediction block in the identical way to that used in P-picture decoding. In SI mode, the prediction P(x,y) is formed by intra prediction block in the identical way to that used in I-picture decoding. After forming the predicted block P(x,y), the decoding of SI and SP-macroblocks follows the same steps. Firstly, the received prediction error coefficients denoted by L_{err} are dequantized using quantization parameter PQP and the results are added to the transform coefficients K_{pred} of predicted block which are found by applying forward transform, see Section **Error! Reference source not found.**, to the predicted block. The resulting sum is quantized with the quantization parameter SPQP. These steps are combined in a single step as follows:

$$Y_{QREC}(i,j) = \frac{Y_{PRED}(i,j) \cdot Q((QP2+12)\%6, i, j) + Y_{QERR}(i,j) \cdot F(QP1, QP2, i, j) + f}{2^{15+(QP2+12)/6}}, \quad i, j = 0,\ldots,3$$

where

$$F(QP1, QP2, i, j) = \frac{Q((QP2+12)\%6, i, j) \cdot 2^{15+(QP2+12)/6} + Q((QP1+12)\%6, i, j)/2}{Q((QP1+12)\%6, i, j)}$$

and the values of f and the coefficients Q(m,i,j) are defined in Subsection 5.3.3.3. Here QP_1 is signaled by PQP value and the QP_2 is signaled by the additional QP parameter SPQP. Notice that when QP1=QP2, then the calculation of Y_{QREC} reduces simply to the sum of the received level and the level found by quantizing the predicted block coefficient. The coefficients Y_{QREC} are then dequantized using $QP=QP_2$ and inverse transform is performed for these dequantized levels, as defined in Subsections 0 and 5.3.1, respectively. Finally, after the reconstruction of a macroblock, filtering of this macroblock takes place as described in Section 8.2.2.

8.2.1 Decoding of DC values of Chrominance

The decoding of chrominance components for SP- and SI-macroblocks is similar to the decoding of luminance components described in Section 8.2. AC coefficients of the chrominance blocks are decoded following the steps described in the earlier section 8.2 where the quantization parameters PQP and SPQP are changed according to the relation between the quantization values of luminance and chrominance, as specified in Section **Error! Reference source not found.**. As described in Section **Error! Reference source not found.**, the coding of DC coefficients of chrominance components includes an additional 2x2 transform. Similarly, for SP and SI-macroblocks, an additional 2x2 transform is applied to the DC coefficients of the predicted 4x4 chrominance blocks and the steps described in Section 8.2 are applied to the 2x2 transform coefficients.

8.2.2 Deblocking Filter

When applying deblocking filter for macroblocks in S-frames, all macroblocks are treated as Intra macroblocks as described in Section **Error! Reference source not found.**.

9 Hypothetical Reference Decoder

9.1 Purpose

The hypothetical reference decoder (HRD) is a mathematical model for the decoder, its input buffer, and the channel. The HRD is characterized by the channel's peak rate R (in bits per second), the buffer size B (in bits), and the initial decoder buffer fullness F (in bits). These parameters represent levels of resources (transmission capacity, buffer capacity, and delay) used to decode a bit stream.

A closely related object is the leaky bucket (LB), which is a mathematical constraint on a bit stream. A leaky bucket is characterized by the bucket's leak rate R_1 (in bits per second), the bucket size B_1 (in bits), and the initial bucket fullness $B_1^{-}F_1$ (in bits). A given bit stream may be constrained by any number of leaky buckets $(R_1,B_1,F_1),\ldots,(R_N,B_N,F_N)$, $N\geq 1$. The LB parameters for a bit stream, which are encoded in the bit stream header, precisely describe the minimum levels of the resources R, B, and F that are sufficient to guarantee that the bit stream can be decoded.

9.2 Operation of the HRD

The HRD input buffer has capacity B bits. Initially, the buffer begins empty. At time t_{start} it begins to receive bits, such that it receives $S(t)$ bits through time t. $S(t)$ can be regarded as the integral of the instantaneous bit rate through time t. The instant at which $S(t)$ reaches the initial decoder buffer fullness F is identified as the decoding time t_0 of the first picture in the bit stream. Decoding times t_1, t_2, t_3, …, for subsequent pictures (in bit stream order) are identified relative to t_0, per Section 9.3. At each decoding time t_i, the HRD instantaneously removes and decodes all d_i bits associated with picture i, thereby reducing the decoder buffer fullness from b_i bits to $b_i - d_i$ bits. Between time t_i and t_{i+1}, the decoder buffer fullness increases from $b_i - d_i$ bits to $b_i - d_i + [S(t_{i+1}) - S(t_i)]$ bits. That is, for $i \geq 0$,

$$b_0 = F$$

$$b_{i+1}= b_i - d_i + [S(t_{i+1}) - S(t_i)].$$

The channel connected to the HRD buffer has peak rate R. This means that unless the channel is idle (whereupon the instantaneous rate is zero), the channel delivers bits into the HRD buffer at instantaneous rate R bits per second.

9.3 Decoding Time of a Picture

The decoding time t_i of picture i is equal to its presentation time τ_i, if there are no B pictures in the sequence. If there are B pictures in the sequence, then $t_i = \tau_i - m_i$, where $m_i = 0$ if picture i is a B picture; otherwise m_i equals $\tau_i - \tau_i'$, where τ_i' is the presentation time of the I or P picture that immediately precedes picture i (in presentation order). If there is no preceding I or P picture (i.e., if $i = 0$), then $m_i = m_0 = t_1 - t_0$. The presentation time of a picture is determinable from its temporal reference and the frame rate.

9.4 Schedule of a Bit Stream

The sequence (t_0,d_0), (t_1,d_1), (t_2,d_2), … is called the schedule of a bit stream. The schedule of a bit stream is intrinsic to the bit stream, and completely characterizes the instantaneous coding rate of the bit stream over its lifetime. A bit stream may be pre-encoded, stored to a file, and later transmitted over channels with different peak bit rates to decoders with different buffer sizes. The schedule of the bit stream is invariant over such transmissions.

9.5 Containment in a Leaky Bucket

A leaky bucket with leak rate R_1, bucket size B_1, and initial bucket fullness B_1–F_1 is said to contain a bit stream with schedule (t_0,d_0), (t_1,d_1), (t_2,d_2), … if the bucket does not overflow under the following conditions. At time t_0, d_0 bits are inserted into the leaky bucket on top of the B_1–F_1 bits already in the bucket, and the bucket begins to drain at rate R_1 bits per second. If the bucket empties, it remains empty until the next insertion. At time t_i, $i \geq 1$, d_i, bits are inserted into the bucket, and the bucket continues to drain at rate R_1 bits per second. In other words, for $i \geq 0$, the state of the bucket just prior to time t_i is

$b_0 = B_1 – F_1$

$b_{i+1} = \max\{0, b_i + d_i – R_1(t_{i+1}–t_i)\}$.

The leaky bucket does not overflow if $b_i + d_i \le B_1$ for all $i \ge 0$.

Equivalently, the leaky bucket contains the bit stream if the graph of the schedule of the bit stream lies between two parallel lines with slope R_1, separated vertically by B_1 bits, possibly sheared horizontally, such that the upper line begins at F_1 at time t_0, as illustrated in the figure below. Note from the figure that the same bit stream is containable in more than one leaky bucket. Indeed, a bit stream is containable in an infinite number of leaky buckets.

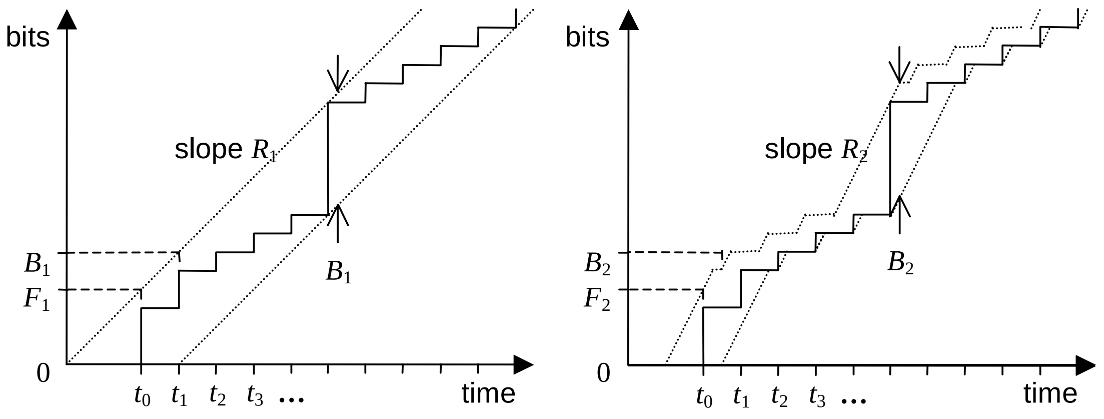


FIGURE 27

Illustration of the leaky bucket concept.

If a bit stream is contained in a leaky bucket with parameters (R_1,B_1,F_1), then when it is communicated over a channel with peak rate R_1 to a hypothetical reference decoder with parameters $R=R_1$, $B=B_1$, and $F=F_1$, then the HRD buffer does not overflow or underflow.

9.6 Bit Stream Syntax

The header of each bit stream shall specify the parameters of a set of $N \ge 1$ leaky buckets, $(R_1,B_1,F_1),\ldots,(R_N,B_N,F_N)$, each of which contains the bit stream. In the current Test Model, these parameters are specified in the first $1+3N$ 32-bit integers of the Interim File Format, in network (big-endian) byte order:

$N, R_1, B_1, F_1, \ldots, R_N, B_N, F_N$.

The R_n shall be in strictly increasing order, and both B_n and F_n shall be in strictly decreasing order.

These parameters shall not exceed the capability limits for particular profiles and levels, which are yet to be defined.

9.7 Minimum Buffer Size and Minimum Peak Rate

If a bit stream is contained in a set of leaky buckets with parameters (R_1,B_1,F_1), …, (R_N,B_N,F_N), then when it is communicated over a channel with peak rate R, it is decodable (i.e., the HRD buffer does not overflow or underflow) provided $B \ge B_{min}(R)$ and $F \ge F_{min}(R)$, where for $R_n \le R \le R_{n+1}$,

$$B_{min}(R) = \alpha B_n + (1 – \alpha)B_{n+1}$$

$$F_{min}(R) = \alpha F_n + (1 – \alpha)F_{n+1}$$

$$\alpha = (R_{n+1} – R) / (R_{n+1} – R_n).$$

For $R \le R_1$,

$$B_{min}(R) = B_1 + (R_1 – R)T$$

$$F_{min}(R) = F_1,$$

where $T = t_{L-1} - t_0$ is the duration of the bit stream (i.e., the difference between the decoding times for the first and last pictures in the bit stream). And for $R \geq R_N$,

$$B_{min}(R) = B_N$$
$$F_{min}(R) = F_N .$$

Thus, the leaky bucket parameters can be linearly interpolated and extrapolated.

Alternatively, when the bit stream is communicated to a decoder with buffer size B, it is decodable provided $R \geq R_{min}(B)$ and $F \geq F_{min}(B)$, where for $B_n \geq B \geq B_{n+1}$,

$$R_{min}(B) = \alpha R_n + (1 - \alpha)R_{n+1}$$
$$F_{min}(B) = \alpha F_n + (1 - \alpha)F_{n+1}$$
$$\alpha = (B - B_{n+1}) / (B_n - B_{n+1}).$$

For $B \geq B_1$,

$$R_{min}(B) = R_1 - (B - B_1)/T$$
$$F_{min}(B) = F_1.$$

For $B \leq B_N$, the stream may not be decodable.

In summary, the bit stream is guaranteed to be decodable in the sense that the HRD buffer does not overflow or underflow, provided that the point (R,B) lies on or above the lower convex hull of the set of points $(0,B_1+R_1T)$, (R_1,B_1), …, (R_N,B_N), as illustrated in the figure below. The minimum start-up delay necessary to maintain this guarantee is $F_{min}(R) / R$.

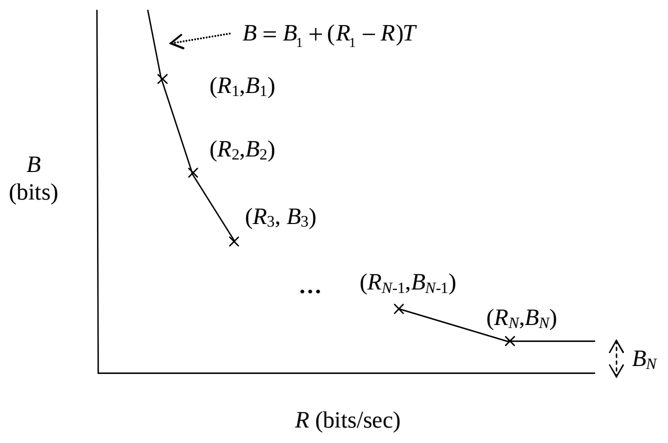


FIGURE 28

Illustration of the leaky bucket concept.

A compliant decoder with buffer size B and initial decoder buffer fullness F that is served by a channel with peak rate R shall perform the tests $B \geq B_{min}(R)$ and $F \geq F_{min}(R)$, as defined above, for any compliant bit stream with LB parameters $(R_1,B_1,F_1),\ldots,(R_N,B_N,F_N)$, and shall decode the bit stream provided that $B \geq B_{min}(R)$ and $F \geq F_{min}(R)$.

9.8 Encoder Considerations (informative)

The encoder can create a bit stream that is contained by some given N leaky buckets, or it can simply compute N sets of leaky bucket parameters after the bit stream is generated, or a combination of these. In the former, the encoder enforces the N leaky bucket constraints during rate control. Conventional rate control algorithms enforce only a single leaky bucket constraint. A rate control algorithm that simultaneously enforces N leaky bucket constraints can be obtained by running a conventional rate control algorithm for each of the N leaky bucket constraints, and using as the current quantization parameter (QP) the maximum of the QPs recommended by the N rate control algorithms.

Additional sets of leaky bucket parameters can always be computed after the fact (whether rate controlled or not), from the bit stream schedule for any given Rn, from the iteration specified in Section 9.5.

Appendix I Non-normative Encoder Recommendation

I.1 Motion Estimation and Mode Decision

I.1.1 Low-complexity mode

I.1.1.1 Finding optimum prediction mode

Both for intra prediction and motion compensated prediction, a similar loop as indicated in **Error! Reference source not found.** is run through. The different elements will be described below.

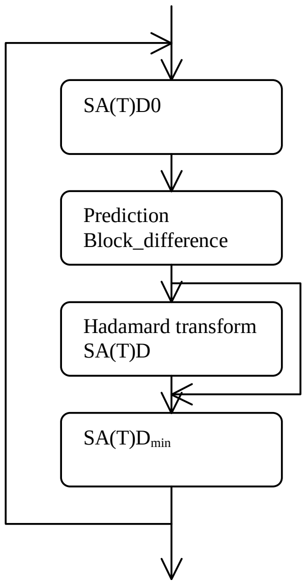


FIGURE 29

Loop for prediction mode decision

I.1.1.1.1 SA(T)D0

The SA(T)D to be minimised is given a 'bias' value SA(T)D0 initially in order to favour prediction modes that need few bits to be signalled. This bias is basically a parameter representing bit usage times QP_0(QP)

Intra mode decision:	SA(T)D0 = QP_0(QP)xOrder_of_prediction_mode (see above)
Motion vector search:	SA(T)D0 = QP_0(QP)x(Bits_to_code_vector + 2xcode_number_of_ref_frame)

In addition there are two special cases:

- For motion prediction of a 16x16 block with 0 vector components, 16xQP_0(QP) is subtracted from SA(T)D to favour the skip mode.
- For the whole intra 4x4 macroblock, 24xQP_0(QP) is added to the SA(T)D before comparison with the best SA(T)D for inter prediction. This is an empirical value to prevent using too many intra blocks.

For flat regions having zero motion, B pictures basically fail to make effective use of zero motion and instead are penalized in performance by selecting 16x16 intra mode. Therefore, in order to prevent assigning 16x16 intra mode to a region with little details and zero motion, SA(T)D of direct mode is subtracted by 16xQP_0(QP) to bias the decision toward selecting the direct mode.

The calculation of SA(T)D0 at each mode is as follows.

- Forward prediction mode :
  SA(T)D0 = QP_0(QP) x (2xcode_number_of_Ref_frame + Bits_to_code_MVDFW)
- Backward prediction mode :
  SA(T)D0 = QP_0(QP) x Bits_to_code_MVDBW

- Bi-directional prediction mode :
  SA(T)D0 = QP_0(QP) x (2xcode_number_of_Ref_frame + Bits_to_code_forward_Blk_size + Bits_to_code_backward_Blk_size + Bits_to_code_MVDFW + Bits_to_code_MVDBW)
- Direct prediction mode :
  SA(T)D0 = – 16 x QP_0(QP)
- Intra 4x4 mode :
  SA(T)D0 = 24 x QP_0(QP)
- Intra 16x16 mode :
  SA(T)D0 = 0

I.1.1.1.2 Block_difference

For the whole block the difference between the original and prediction is produced

Diff(i,j) = Original(i,j) - Prediction(i,j)

I.1.1.1.3 Hadamard transform

For integer pixel search (see below) we use SAD based on Diff(i,j) for decision. Hence no Hadamard is done and we use SAD instead of SATD.

$$SAD = \sum_{i,j} \left| Diff\,(i,j) \right|$$

However, since we will do a transform of Diff(i,j) before transmission, we will do a better optimisation if a transform is done before producing SAD. Therefore a two dimensional transform is performed in the decision loop for selecting intra modes and for fractional pixel search (see below). To simplify implementation, the Hadamard transform is chosen in this mode decision loop. The relation between pixels and basis vectors (BV) in a 4 point Hadamard transform is illustrated below (not normalized):

```
     Pixels →
B    1     1    1    1
V    1     1   -1   -1
↓    1    -1   -1    1
     1    -1    1   -1
```

This transformation is performed horizontally and vertically and result in DiffT(i,j). Finally SATD for the block and for the present prediction mode is produced.

$$SATD = (\sum_{i,j} \left| DiffT\,(i,j) \right|) / 2$$

I.1.1.1.4 Mode decision

Choose the prediction mode that results in the minimum value $SA(T)D_{min}$ = min(SA(T)D+SA(T)D0).

I.1.1.2 Encoding on macroblock level

I.1.1.2.1 Intra coding

When starting to code a macroblock, intra mode is checked first. For each 4x4 block, full coding indicated in **Error! Reference source not found.** is performed. At the end of this loop the complete macroblock is intra coded and a $SATD_{intra}$ is calculated.

I.1.1.2.2 Table for intra prediction modes to be used at the encoder side

Error! Reference source not found. gives the table of intra prediction modes according to probability of each mode to be used on the decoder side. On the encoder side we need a sort of inverse table. Prediction modes for A and B are known as in **Error! Reference source not found.**. For the encoder we have found a Mode that we want to signal with an ordering number in the bitstream (whereas on the decoder we receive the order in the bitstream and want to convert this to a mode). **Error! Reference source not found.** is therefore the relevant table for the encoder. Example: Prediction mode for A and B is 2. The string in **Error! Reference source not found.** is 2 1 0 3 4 5. This indicates that prediction mode 0 has order 2 (third most probable). Prediction mode 1 is second most probable and prediction mode 2 has order 0 (most probable) etc. As in **Error! Reference source not found.** '–' indicates that this instance can not occur because A or B or both are outside the picture.

TABLE 16

Prediction ordering to be used in the bitstream as a function of prediction mode (see text).

B\A	outside	0	1	2	3	4	5
outside	0-----	021---	102---	120---	012---	012---	012---
0	0---12	025314	104325	240135	143025	035214	045213
1	0---12	014325	102435	130245	032145	024315	015324
2	0---12	012345	102345	210345	132045	032415	013245
3	0---12	135024	214035	320154	143025	145203	145032
4	1---02	145203	125403	250314	245103	145203	145302
5	1---20	245310	015432	120534	245130	245301	135420

I.1.1.2.3 Inter mode selection

Next motion vector search is performed for motion compensated prediction. A search is made for all 7 possible block structures for prediction as well as from the 5 past decoded pictures. This result in 35 combinations of blocksizes and reference frames. For B-frames, the motion search is also conducted for the temporal following reference picture to obtain backward motion vectors.

I.1.1.2.4 Integer pixel search

The search positions are organised in a 'spiral' structure around the <u>predicted</u> vector (see vector prediction). The numbering from 0 and upwards for the first positions are listed below:

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

A parameter MC_range is used as input for each sequence. To speed up the search process, the search range is further reduced:

- Search range is reduced to: Range = MC_range/2 for all block sizes except 16x16 in prediction from the most recent decoded picture.
- The range is further reduced to: Range = Range/2 for search relative to all older pictures.

After Range has been found, the centre for the spiral search is adjusted so that the (0,0) vector position is within the search range. This is done by clipping the horizontal and vertical positions of the search centre to ±Range.

I.1.1.2.5 Fractional pixel search

Fractional pixel search is performed in two steps. This is illustrated below where capital letters represent integer positions, numbers represent ½ pixel positions and lower case letters represent ¼ pixel positions.

```
A       B       C
    1   2   3
D   4   E   5   F
      a b c
    6 d 7 e 8
      f g h
    G        H        I
```

Assume that the integer search points to position E. Then ½ pixel positions 1,2,3,4,5,6,7,8 are searched. Assume that 7 is the best position. Then the ¼ pixel positions a,b,c,d,e,f,g,h are searched. (Notice that by this procedure a position with 'more low pass filtering' – see 5.2.1 - is automatically checked). If motion compensation with 1/8 pixel accuracy is used, an additional sub-pixel refinement step is performed in the described way. After fractional pixel search has been performed for the complete macroblock, the SATD for the whole macroblock is computed: $\mathrm{SATD}_{\mathrm{inter}}$.

I.1.1.2.6 Decision between intra and inter

If $\mathrm{SATD}_{\mathrm{intra}} < \mathrm{SATD}_{\mathrm{inter}}$ intra coding is used. Otherwise inter coding is used.

I.1.2 High-complexity mode

I.1.2.1 Motion Estimation

For each block or macroblock the motion vector is determined by full search on integer-pixel positions followed by sub-pixel refinement.

I.1.2.1.1 Integer-pixel search

As in low-complexity mode, the search positions are organized in a spiral structure around a prediction vector. The full search range given by MC_range is used for all INTER-modes and reference frames. To speed up the search process , the prediction vector of the 16x16 block is used as center of the spiral search for all INTER-modes. Thus the SAD values for 4x4 blocks can be pre-calculated for all motion vectors of the search range and than used for fast SAD calculation of all larger blocks. The search range is not forced to contain the (0,0)-vector.

I.1.2.1.2 Fractional pixel search

The fractional pixel search is performed as in the low-complexity case.

I.1.2.1.3 Finding the best motion vector

The integer-pixel motion search as well as the sub-pixel refinement returns the motion vector that minimizes

$$J(\mathbf{m}, \lambda_{MOTION}) = SA(T)D(s, c(\mathbf{m})) + \lambda_{MOTION} \cdot R(\mathbf{m} - \mathbf{p})$$

with $\mathbf{m} = (m_x, m_y)^T$ being the motion vector, $\mathbf{p} = (p_x, p_y)^T$ being the prediction for the motion vector, and λ_{MOTION} being the Lagrange multiplier. The rate term $R(\mathbf{m}\text{-}\mathbf{p})$ represents the motion information only and is computed by a table-lookup. The rate is estimated by using the universal variable length code (UVLC) table, even if CABAC is used as entropy coding method. For integer-pixel search, SAD is used as distortion measure. It is computed as

$$SAD(s,\ c(\mathbf{m})) = \sum_{x=1,\ y=1}^{B,\ B} \left| s[x,y] - c[x - m_x, y - m_y] \right|, \qquad B = 16, 8 \ or \ 4 .$$

with *s* being the original video signal and *c* being the coded video signal. In the sub-pixel refinement search, the distortion measure SATD is calculated after a Hadamard transform (see section **Error! Reference source not found.**). The Lagrangian multiplier λ_{MOTION} is given by

$$\lambda_{MODE,P} = \sqrt{0.85 \cdot 2^{QP/3}}$$

for I- and P-frames and

$$\lambda_{MODE,B} = \sqrt{4 \cdot 0.85 \cdot 2^{QP/3}}$$

for B-frames, where QP is the macroblock quantization parameter.

I.1.2.1.4 Finding the best reference frame

The determination of the reference frame *REF* and the associated motion vectors for the *NxM* inter modes is done after motion estimation by minimizing

$$J(REF \mid \lambda_{MOTION}) = SATD(s, c(REF, \mathbf{m}(REF))) + \lambda_{MOTION} \cdot (R(\mathbf{m}(REF) - \mathbf{p}(REF)) + R(REF)).$$

The rate term $R(REF)$ represents the number of bits associated with choosing REF and is computed by table-lookup using UVLC.

I.1.2.1.5 Finding the best prediction direction for B-frames

The determination of the prediction direction *PDIR* for the *NxM* inter modes in B-frames is done after motion estimation and reference frame decision by minimizing

$$J(PDIR \mid \lambda_{MOTION}) = SATD(s, c(PDIR, \mathbf{m}(PDIR))) + \lambda_{MOTION} \cdot (R(\mathbf{m}(PDIR) - \mathbf{p}(PDIR)) + R(REF(PDIR)))$$

I.1.2.2 Mode decision

I.1.2.2.1 Macroblock mode decision

The macroblock mode decision is done by minimizing the Lagrangian functional

$$J(s, c, MODE \mid QP, \lambda_{MODE}) = SSD(s, c, MODE \mid QP) + \lambda_{MODE} \cdot R(s, c, MODE \mid QP)$$

where QP is the macroblock quantizer, λ_{MODE} is the Lagrange multiplier for mode decision, and $MODE$ indicates a mode chosen from the set of potential prediction modes:

I-frame: $MODE \in \{INTRA4x4, INTRA16x16\}$,

P-frame: $MODE \in \left\{\begin{matrix} INTRA4x4, INTRA16x16, SKIP, \\ 16x16, 16x8, 8x16, 8x8 \end{matrix}\right\}$,

B-frame: $MODE \in \left\{\begin{matrix} INTRA4x4, INTRA16x16, DIRECT, \\ 16x16, 16x8, 8x16, 8x8 \end{matrix}\right\}$.

Note that the $SKIP$ mode refers to the 16x16 mode where no motion and residual information is encoded. SSD is the sum of the squared differences between the original block s and its reconstruction c given as

$$SSD(s, c, MODE \mid QP) = \sum_{x=1,y=1}^{16,16} (s_Y[x,y] - c_Y[x,y,MODE \mid QP])^2 + \sum_{x=1,y=1}^{8,8} (s_U[x,y] - c_U[x,y,MODE \mid QP])^2 + \sum_{x=1,y=1}^{8,8} (s_V[x,y] - c_V[x,y,MODE \mid QP])^2,$$

and $R(s, c, MODE \mid QP)$ is the number of bits associated with choosing $MODE$ and QP including the bits for the macroblock header, the motion, and all DCT blocks. $c_Y[x, y, MODE \mid QP]$ and $s_Y[x, y]$ represent the reconstructed and original luminance values; c_U, c_V and s_U, s_V the corresponding chrominance values.

The Lagrangian multiplier λ_{MODE} is given by

$$\lambda_{MODE,P} = 0.85 \cdot 2^{QP/3}$$

for I- and P-frames and

$$\lambda_{MODE,B} = 4 \cdot 0.85 \cdot 2^{QP/3}$$

for B-frames, where QP is the macroblock quantization parameter.

I.1.2.2.2 8x8 mode decision

The mode decision for 8x8 sub-partitions is done similar to the macroblock mode decision by minimizing the Lagrangian functional

$$J(s,c,MODE\,|\,QP,\lambda_{MODE}) = SSD(s,c,MODE\,|\,QP) + \lambda_{MODE}\cdot R(s,c,MODE\,|\,QP)$$

where QP is the macroblock quantizer, λ_{MODE} is the Lagrange multiplier for mode decision, and $MODE$ indicates a mode chosen from the set of potential prediction modes:

P-frame: $$MODE \in \left\{\begin{matrix} INTRA\,4x4, \\ 8x8,\ 8x4,\ 4x8,\ 4x4 \end{matrix}\right\}$$

B-frame: $$MODE \in \left\{\begin{matrix} INTRA\,4x4,\ DIRECT, \\ 8x8,\ 8x4,\ 4x8,\ 4x4 \end{matrix}\right\}.$$

I.1.2.2.3 INTER 16x16 mode decision

The *INTER16x16* mode decision is performed by choosing the INTER16x16 mode which results in the minimum SATD value.

I.1.2.2.4 INTER 4x4 mode decision

For the *INTRA4x4* prediction, the mode decision for each 4x4 block is performed similar to the macroblock mode decision by minimizing

$$J(s,c,IMODE\,|\,QP,\lambda_{MODE}) = SSD(s,c,IMODE\,|\,QP) + \lambda_{MODE}\cdot R(s,c,IMODE\,|\,QP)$$

where QP is the macroblock quantizer, λ_{MODE} is the Lagrange multiplier for mode decision, and $IMODE$ indicates an intra prediction mode:

$$IMODE \in \{DC,\ HOR, VERT,\ DIAG,\ DIAG\_RL,\ DIAG\_LR\}.$$

SSD is the sum of the squared differences between the original 4x4 block luminance signal s and its reconstruction c, and $R(s,c,IMODE\,|\,QP)$ represents the number of bits associated with choosing *IMODE*. It includes the bits for the intra prediction mode and the DCT-coefficients for the 4x4 luminance block. The rate term is computed using the UVLC entropy coding, even if CABAC is used for entropy coding.

I.1.2.3 Algorithm for motion estimation and mode decision

The procedure to encode one macroblock s in a I-, P- or B-frame in the high-complexity mode is summarized as follows.

1. Given the last decoded frames, λ_{MODE}, λ_{MOTION}, and the macroblock quantizer QP
2. Choose intra prediction modes for the *INTRA 4x4* macroblock mode by minimizing
   $J(s,c,IMODE\,|\,QP,\lambda_{MODE}) = SSD(s,c,IMODE\,|\,QP) + \lambda_{MODE}\cdot R(s,c,IMODE\,|\,QP)$
   with $IMODE \in \{DC,\ HOR, VERT,\ DIAG,\ DIAG\_RL,\ DIAG\_LR\}$.
3. Determine the best $INTRA16x16$ prediction mode by choosing the mode that results in the minimum SATD.
4. For each 8x8 sub-partition

   Perform motion estimation and reference frame selection by minimizing

   SSD + λ Rate(MV, REF)

B-frames: Choose prediction direction by minimizing

SSD + λ Rate(MV(PDIR), REF(PDIR))

Determine the coding mode of the 8x8 sub-partition using the rate-constrained mode decision, i.e. minimize

SSD + λ Rate(MV, REF, Luma-Coeff, block 8x8 mode)

Here the SSD calculation is based on the reconstructed signal after DCT, quantization, and IDCT.

5. Perform motion estimation and reference frame selection for 16x16, 16x8, and 8x16 modes by minimizing
$$J(REF,\mathbf{m}(REF)\,|\,\lambda_{MOTION}) = SA(T)D(s,c(REF,\mathbf{m}(REF))) + \lambda_{MOTION}\cdot(R(\mathbf{m}(REF)-\mathbf{p}(REF)) + R(REF))$$
for each reference frame and motion vector of a possible macroblock mode.
6. B-frames: Determine prediction direction by minimizing
$$J(PDIR\,|\,\lambda_{MOTION}) = SATD(s,c(PDIR,\mathbf{m}(PDIR))) + \lambda_{MOTION}\cdot(R(\mathbf{m}(PDIR)-\mathbf{p}(PDIR)) + R(REF(PDIR)))$$

7. Choose the macroblock prediction mode by minimizing
$$J(s,c,MODE\,|\,QP,\lambda_{MODE}) = SSD(s,c,MODE\,|\,QP) + \lambda_{MODE}\cdot R(s,c,MODE\,|\,QP),$$
given QP and λ_{MODE} when varying $MODE$. $MODE$ indicates a mode out of the set of potential macroblock modes:

I-frame: $MODE \in \{INTRA4x4,\ INTRA16x16\}$,

P-frame: $MODE \in \left\{\begin{matrix} INTRA4x4,\ INTRA16x16,\ SKIP, \\ 16x16,\ 16x8,\ 8x16,\ 8x8 \end{matrix}\right\}$,

B-frame: $MODE \in \left\{\begin{matrix} INTRA4x4,\ INTRA16x16,\ \ DIRECT, \\ 16x16,\ 16x8,\ 8x16,\ 8x8 \end{matrix}\right\}$.

The computation of $J(s,c,SKIP\,|\,QP,\lambda_{MODE})$ and $J(s,c,DIRECT\,|\,QP,\lambda_{MODE})$ is simple. The costs for the other macroblock modes are computed using the intra prediction modes or motion vectors and reference frames, which have been estimated in the previous steps.

I.2 Quantization

$$Y_Q(i,j) = \left[Y(i,j)\cdot Q((QP+12)\%6,i,j) + f\right] / 2^{15+(QP+12)/6}$$

I.3 Elimination of single coefficients in inter macroblocks

I.3.1 Luminance

With the small 4x4 blocks, it may happen that for instance a macroblock has only one nonzero coefficient with |Level| =1. This will probably be a very expensive coefficient and it could have been better to set it to zero. For that reason a procedure to check single coefficients have been implemented for inter luma blocks. During the quantization process, a parameter **Single_ctr** is accumulated depending on Run and Level according to the following rule:

- If Level = 0 or (|Level| = 1 and Run > 5) nothing is added to Single_ctr.
- If |Level| > 1, 9 is added to Single_ctr.
- If |Level| = 1 and Run < 6, a value T(Run) is added to Single_ctr. where T(0:5) =(3,2,2,1,1,1)

- If the accumulated Single_ctr for a 8x8 block is less than 4, all coefficients of that luma block are set to zero. Similarly, if the accumulated Single_ctr for the whole macroblock is less than 6, all coefficients of that luma macroblock are set to zero.

I.3.2 Chrominance

A similar method to the one for luma is used. **Single_ctr** is calculated similarly for each chroma component, but for AC coefficients only and for the whole macroblock.

If the accumulated **Single_ctr** for each chroma component of a macroblock is less than 7, all the AC chroma coefficients of that component for the whole macroblock are set to zero.

I.4 S-Pictures

I.4.1 Encoding of secondary SP-pictures

This section suggests an algorithm that can be used to create a secondary SP-picture with identical pixel values to another SP-picture (called the target SP-picture). The secondary SP-picture is typically used for switching from one bitstream to another and thus has different reference frames for motion compensation than the target SP-picture.

Intra-type macroblocks from the target SP-picture are copied into the secondary SP-picture without alteration. SP-macroblocks from the target SP-pictures will be replaced by the following procedure: Let L_{rec} be the quantized coefficients, see Section 8.2, that are used to reconstruct the SP-block. Define the difference levels L_{err} by:

$$L_{err} = L_{rec} - L_P$$

where L_P is the quantized transform coefficients of the predicted block from any of the SP coding modes. Then a search over all the possible SP modes is performed and the mode resulting in the minimum number of bits is selected.

I.4.2 Encoding of SI-pictures

The following algorithm is suggested to encode an SI-picture such that the reconstruction of it is identical to that of an SP-picture. SP-pictures consist of intra and SP-macroblocks. Intra-type macroblocks from SP-pictures are copied into SI-pictures with minor alteration. Specifically, the MB_Type is altered to reflect Intra Modes in SI-pictures, i.e., MB_Type is incremented by one, See TABLE 15. The SP-macroblocks from SP-pictures will be replaced by SIntra 4x4 modes. Let L_{rec} be the quantized coefficients, see Section 8.2, that are used to reconstruct the SP-block. Define the difference levels L_{err} by:

$$L_{err} = L_{rec} - L_I$$

where L_I is the quantized transform coefficients of the predicted block from any of the intra prediction modes. Then a search over the possible intra prediction modes is performed and the mode resulting in the minimum number of bits is selected.

I.5 Encoding with Anticipation of Slice Losses

Low delay video transmission may lead to losses of slices. The decoder may then stop decoding until the next I picture or P picture may conduct a concealment, for example as explained in Appendix IV, and continue decoding. In the latter case, spatio-temporal error propagation occurs if the concealed picture content is referenced for motion compensation. There are various means to stop spatio-temporal error propagation including the usage of multiple reference pictures and Intra coding of macroblocks. For the latter case, a Lagrangian mode selection algorithm is suggested as follows.

Since transmission errors occur randomly, the decoding result is also a random process. Therefore, the average decoder distortion is estimated to control the encoder for a specified probability of packet losses *p*. The average decoding result is obtained by running *N* complete decoders at the encoder in parallel. The statistical process of loosing a slice is assumed to be independent for each of the *N* decoders. The slice loss process for each decoder is also assumed to be i.i.d. and a certain slice loss probability *p* is assumed to be known at the encoder. Obviously for large *N* the decoder gets a very good estimate of the average decoder distortion. However, with increasing *N* a linear increase of storage and decoder complexity in the encoder is incurred. Therefore, this method might not be practical in real-time encoding processes and complexity and memory efficient algorithms are currently under investigation.

To encode a macroblock in a P picture, the set of possible macroblock types is given as

$\mathcal{S}_{MB}$ = { SKIP, INTER_16x16, INTER_16x8, INTER_8x16, INTER_8x8, INTRA_16x16}

For each macroblock the coding mode *m'* is selected according to

$$m' = \min_{m \in \mathcal{S}_{MB}} \{D_m + \lambda R_m\}$$

with D_m being the distortion in the current macroblock when selecting macroblock mode *m* and R_m being the corresponding rate, i.e. the number of bits. For the COPY_MB and all INTER_MxN types, the distortion D_m is computed as

$$D_m = \frac{1}{N}\sum_{n=1}^{N}\sum_{i}(f_i - \hat{f}_{i,n,m}(p))^2$$

with f_i being the original pixel value at position *i* within the macroblock and $\hat{f}_{i,n,m}$ being the reconstructed pixel value at position *i* for coding macroblock mode *m* in the simulated decoder *n*. The distortion for the INTRA macroblocks remains unchanged. Since the various reconstructed decoders also contain transmission errors, the Lagrangian cost function for the COPY_MB and all INTER_MxN types increases making INTRA_NxN types more popular.

The λ parameter for mode decision depends on the quantization parameter *q* as follows

$$\lambda = (1 - p)0.85 \cdot 2^{QP/3}.$$

Appendix II Network Adaptation Layer

II.1 Byte Stream NAL Format

The byte stream NAL format is specified for use by systems that transmit JVT video as an ordered stream of bytes, such as ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems or ITU-T Rec. H.320 systems.

The byte stream NAL format consists of the sequence of video packets, each of which is prefixed as follows:

1. Optionally (at the discretion of the encoder if not prohibited by a system-level specification), any number of zero-stuffing bytes (ZSB) having the value zero.
2. A three-byte start code prefix (SCP), consisting of two bytes that are both equal to zero (0x00), followed by one byte having the value one (0x01).
3. The payload type indicator byte (PTIB),
4. One or more EBSP's then follows as determined by the PTIB.

Any number of zero-stuffing bytes (ZSB) may optionally (at the discretion of the encoder if not prohibited by a system-level specification) follow the last video packet in the byte stream.

The location of each SCP can be detected by scanning for the three-byte SCP value pattern in the byte stream.

The first EBSP starts immediately after the SCP and PTIB, and the last EBSP ends with the last non-zero byte prior to the next SCP.

II.2 IP NAL Format

This section covers the Network Adaptation Layer for non-managed, best effort IP networks using RTP [RFC1889] as the transport. The section will likely end up in the form of a standard's track RFC covering an RTP packetization for H.26L.

The NAL takes the information of the Interim File Format as discussed in section **Error! Reference source not found.** and converts it into packets that can conveyed directly over RTP. It is designed to be able to take advantage of more than one virtual transport stream (either within one RTP stream by unequal packet content protection currently discussed in the IETF and as Annex I of H.323, or by using several RTP streams with network or application layer unequal error protection).

In doing so, it has to

- arrange partitions in an intelligent way into packets
- eventually split/recombine partitions to match MTU size constraints
- avoid the redundancy of (mandatory) RTP header information and information in the video stream
- define receiver/decoder reactions to packet losses. Note: the IETF tends more and more to do this in a normative way, whereas in the ITU and in MPEG this is typically left to implementers. Issue has to be discussed one day. Current document provides information without makeing any assumptions about that.

II.2.1 Assumptions

Any packetization scheme has to make some assumptions on typical network conditions and constraints. The following set of assumptions have been used in earlier Q.15 research on packetization and are deemed to be still valid:

- MTU size: around 1500 bytes per packet for anything but dial-up links, 500 bytes for dial-up links.

- Packet loss characteristic: non-bursty (due to drop-tail router implementations, and assuming reasonable pacing algorithms (e.g. no bursting occurs at the sender).
- Packet loss rate: up to 20%

II.2.2 Combining of Partitions according to Priorities

In order to allow unequal protection of more important bits of the bitstream, exactly two packets per slice are generated (see Q15-J-53 for a detailed discussion). Slices should be used to ensure that both packets meet the MTU size constraints to avoid network splitting/recombining processes.

The ’First’ packet contains the following partitions:

- TYPE_HEADER
- TYPE_MBHEADER
- TYPE_MVD
- TYPE_EOS

The ’Second’ packet is assembled using the rest of the partitions

• TYPE_CBP	Coded Block Patterm
• TYPE_2x2DC	2x2 DC Coefficients
• TYPE_COEFF_Y	Luminance AC Coefficients
• TYPE_COEFF_C	Chrominance AC Coefficients

This configuration allows decoding the first packet independently from the second (although not vice versa). As the first packet is more important both because motion information is important for struction [for what?] and because the ’First’ packet is necessary to decode the ’Second’, UEP should be used to protect the ’First’ packet s higher.

II.2.3 Packet Structure

Each packet consists of a fixed 32 bit header a series of Part-of-Partition structures (POPs).

The packet header contains information

Bit 31	1 == This packet contains a picture header
Bit 30	1 == This packet contains a slice header
Bits 25..29	Reserved
Bits 10-24	StartMB. It is assumed that no picture has more than 2**14 Macroblocks
Bits 0..9	SliceID. It is assumed that a picture does not have more than 1024 slices

Note: The PictureID (TR) can be easily reconstructed from the RTP Timestamp and is therefore not coded again.

Note: The current software reconstructs QP and Format out of the VLC coded Picture/Slice header symbols. This is architecturally not nice and should be changed, probably by deleting these two values from the interim File Format.

Note: This header is likely to change once bigger picture formats etc. come into play.

Each Part-of-Partition structure contains a header of 16 bits whose format is as follows

Bits 15..12	Data Type
Bits 11..0	Length of VLC-coded POP payload (in bits, starting byte-aligned, 0 indicates 4096 bits of payload)

The reasoning behind the introduction of POP packets lies in avoiding large fixed length headers for (typically) small partitions. See Q15-J-53.

II.2.4 Packetization Process

The packetization process converts the Interim File Format (or, in a real world system, data partitioned symbols arriving through a software interface) into packets. The following RTP header fields are used (see RFC1889 for exact semantics):

- Timestamp: is calculated according to the rules of RFC1889 and RFC2429 based on a 90 KHz timestamp.
- Marker Bit: set for the very last packet of a picture ('Second' packet of the last Slice), otherwise cleared.
- Sequence Number is increased by one for every generated packet, and starts with 0 for easier debugging (this in contrast to RFC1889, where a random initialization is mandatory for security purposes).
- Version (V): 2
- Padding (P): 0
- Extension (X): 0
- Csourcecount (CC): 0
- Payload Type (PT): 0 (This in contrast to RFC1889 where 0 is forbidden)

The RTP header is followed by the payload, which follows the packet structure of section II.2.3.

The RTP packet file, used as the input to packet loss simulators and similar tools (note that this format is identical to the ones used for IP-related testing during the H.263++ project, so that the loss simulators, error paytterns etc, can be re-used):

Int32	size of the following packet in bytes
[]byte	packet content, starting with the RTP header.

II.2.5 De-packetization

The De-packetization process reconstructs a file in the Interim File Format from an RTP packet file (that is possible subject to packet losses). This task is straightforward and reverse to the packetization process. (Note that the QP and Format fields currently have to be reconstructed using the VLC-coded symbols in the TYPE_HEADER partition. This bug in the Interim File Format spec should be fixed some time).

II.2.6 Repair and Error Concealment

In order to take advantage of potential UEP for the 'First' packet and the ability of the decoder to reconstruct data where CBP/coefficient information was lost, a very simple error concealment strategy is used. This strategy repairs the bitstream by replacing a lost CBP partition with CBPs that indicate no coded coefficients. Unfortunately, the CBP codewords for Intra and Inter blocks are different, so that such a repair cannot be done context-insensitive. Instead of (partly) VLC-decoding the CBP partition in the NAL module in order to insert the correct type of CBP symbol in the (lost) partition, the decoder itself can be changed to report the appropriate CBP symbols saying "No coefficients" whenever the symbol fetch for a CBP symbol returns with the indication of a lost/empty partition.

Appendix III Interim File Format

III.1 General

Note: The intent of JVT is to move toward use of the ISO media file format as the defined method for JVT video content storage. This Appendix defines an interim file format that can be used until encapsulation of JVT video content into ISO media file format has been specified.

A file is self-contained.

A file consists of boxes, whose structure is identical to boxes of ISO/IEC 14496-1:2001 (ISO media file format).

A box may contain other boxes. A box may have member attributes. If a box contains attributes and other boxes, boxes shall follow the attribute values.

The attribute values in the boxes are stored with the most significant byte first, commonly known as network byte order or big-endian format.

A number of boxes contain index values into sequences in other boxes. These indexes start with the value 0 (0 is the first entry in the sequence).

The Syntactic Description Language (SDL) of ISO/IEC 14496-1:2001 is used to define the file format. In addition to the existing basic data types, the UVLC elementary data type is defined in this document. It shall be used to carry variable-length bit-fields that follow the JVT UVLC design.

Unrecognized boxes should be skipped and ignored.

III.2 File Identification

The File Type Box is the first box of the file. JVT files shall be identified from a major Brand field equal to 'jvt '.

The preferred file extension is '.jvt'.

III.3 Box

III.3.1 Definition

Boxes start with a header, which gives both size and type. The header permits compact or extended size (32 or 64 bits). Most boxes will use the compact (32-bit) size. The size is the entire size of the box, including the size and type header, fields, and all contained boxes. This facilitates general parsing of the file.

III.3.1.1 Syntax

```
aligned(8) class box (unsigned int(32) boxType)
    unsigned int(32) size;
    unsigned int(32) type = boxType;

    if (size==1) {
        unsigned int(64) largesize;
    } else if (size==0) {
        // box extends to end of file
    }
}
```

III.3.1.2 Semantics

- size is an integer that specifies the number of bytes in this box, including all its fields and contained boxes; if size is 1 then the actual size is in the field largesize; if size is 0, then this box is the last one in the file, and its contents extend to the end of the file (normally only used for an Alternate Track Media Box)

- type identifies the box type; standard boxes use a compact type, which is normally four printable characters, to permit ease of identification, and is shown so in the boxes below.

III.4 Box Order

An overall view of the normal encapsulation structure is provided in the following table.

The table shows those boxes that may occur at the top-level in the left-most column; indentation is used to show possible containment. Thus, for example, an Alternate Track Header Box (atrh) is found in a Segment Box (segm).

Not all boxes need be used in all files; the mandatory boxes are marked with an asterisk (*). See the description of the individual boxes for a discussion of what must be assumed if the optional boxes are not present.

There are restrictions in which order the boxes shall appear in a file. See the box definitions for these restrictions.

TABLE 17

Box types.

ftyp				*	**Error! Reference source not found.**	File Type Box, identifies the file format
jvth				*	**Error! Reference source not found.**	File Header Box, file-level meta-data
cinf					**Error! Reference source not found.**	Content Info Box, describes file contents
atin				*	**Error! Reference source not found.**	Alternate Track Info Box, describes characteristics of tracks
prms				*	**Error! Reference source not found.**	Parameter Set Box, enumerated set of frequently changing coding parameters

segm				*	**Error! Reference source not found.**	Segment Box, contains meta- and media data for a defined period of time
	atrh			*	**Error! Reference source not found.**	Alternate Track Header Box, meta-data for a track
		pici		*	III.5.8	Picture Information Box, meta-data for individual pictures
		layr			III.5.9	Layer Box, meta-data for a layer of pictures
			sseq		III.5.10	Sub-Sequence Box, meta-data for a sub-sequence within a layer
	swpc				**Error! Reference source not found.**	Switch Picture Box, identifies pictures that can be used to switch between tracks.
	atrm			*	III.5.11	Alternate Track Media Box, media data for a track

III.5 Box Definitions

III.5.1 File Type Box

III.5.1.1 Definition

Box Type:	\`ftyp’
Container:	File
Mandatory:	Yes
Quantity:	Exactly one

A media-file structured according to the ISO media file format specification may be compatible with more than one detailed specification, and it is therefore not always possible to speak of a single ‘type’ or ‘brand’ for the file. This box identifies a JVT file in a similar fashion without claiming compatibility with the ISO format. However, it enables other file readers to identify the JVT file type. It must be placed first in the file.

III.5.1.2 Syntax

```
aligned(8) class FileTypeBox aligned(8) extends box(‘ftyp’) {
    unsigned int(32) majorBrand = ‘jvt ’;
    unsigned int(16) jmMajorVersion;
    unsigned int(16) jmMinorVersion;
    unsigned int(32) compatibleBrands[];   // to end of the box
}
```

III.5.1.3 Semantics

This box identifies the specification to which this file complies.

majorBrand is a brand identifier for the interim JVT file format. Only ‘jvt ’ shall be used for majorBrand, as the file format is not compatible with any other format.

jmMajorVersion and jmMinorVersion define the version of the standard working draft the file complies with. For example, JM-1 files shall have jmMajorVersion equal to 1 and jmMinorVersion equal to 0.

compatibleBrands is a list, to the end of the box, of brands. Should only include the entry ‘jvt ’.

Note: As the interim JVT file format is based on the ISO media file format, it might be appropriate to allow a combination of many ISO media file format based file types into the same file. In such a case, the majorBrand might not be equal to ‘jvt ‘ but ‘jvt ‘ should be one of the compatibleBrands. As this option was not discussed in the Pattaya meeting, it is not reflected in the current specification of the interim JVT file format (this document).

III.5.2 File Header Box

III.5.2.1 Definition

Box Type:	\`jvth’
Container:	File
Mandatory:	Yes
Quantity:	One or more

This box must be placed as the second box of the file.

The box can be repeated at any position of the file when no container box is open. A File Header Box identifies a random access point to the file. In other words, no data prior to a selected File Header Box is required to parse any of the succeeding data. Furthermore, any segment can be parsed without a forward reference to any of the data succeeding the particular segment.

III.5.2.2 Syntax

```
aligned(8) class fileHeaderBox extends box(‘jvth’) {
    unsigned int(8) majorVersion = 0x00;
    unsigned int(8) minorVersion = 0x00;
    unsigned int(32) timescale;
    unsigned int(32) numUnitsInTick;
    unsigned int(64) duration;
    unsigned int(16) pixAspectRatioX;
    unsigned int(16) pixAspectRatioY;
    unsigned int(16) maxPicId;
    unsigned int(8) numAlternateTracks;
    unsigned int(2) numBytesInPayloadCountMinusOne;
    unsigned int(2) numBytesInPictureOffsetMinusTwo;
    unsigned int(2) numBytesInPictureDisplayTimeMinusOne;
    unsigned int(2) numBytesInPictureCountMinusOne;
    unsigned int(2) numBytesInPayloadSizeMinusOne;
}
```

III.5.2.3 Semantics

majorVersion and minorVersion indicate the version of the file format. This specification defines the format for version 0.1 (majorVersion.minorVersion). Version numbering is independent of working draft document and joint model software as well as the version of the standard | recommendation. This allows parsers interpret the high-level syntax of the files, even if decoding of a file according to the indicated joint model or standard version was not supported.

timescale is the number of time units which pass in one second. For example, a time coordinate system that measures time in sixtieths of a second has a time scale of 60.

numUnitsInTick is the number of time units according to timescale that correspond to one clock tick. A clock tick is the minimum unit of time that can be presented in the file. For example, if the clock frequency of a video signal is (30 000) / 1001 Hz, timescale should be 30 000 and numUnitsInTick should be 1001.

duration is an integer that declares length of the file (in the indicated timescale). Value zero indicates that no duration information is available.

pixAspectRatioX and pixAspectRatioY define the pixel geometry, calculated by pixAspectRatioX / pixAspectRatioY. Value zero in either or both of the attributes indicate an unspecified pixel aspect ratio.

maxPicId gives the maximum value for the picture identifier.

numAlternateTracks gives the number of alternative encodings of the same source. Typically each encoding is targeted for different bit-rate. Each file shall contain at least one track.

numBytesInPayloadCountMinusOne indicates the number of bytes that are needed to signal the maximum number of payloads in any picture. For example, numBytesInPayloadCountMinusOne equal to zero indicates that one byte is needed to signal the number of payloads, and the maximum number of payloads is 255.

numBytesInPictureOffsetMinusTwo indicates the number of bytes that are needed to signal picture offsets. For example, numBytesInPictureOffsetMinusTwo equal to zero indicates that the offsets are two-byte integer values with a range of –32768 to 32767.

numBytesInPictureDisplayTimeMinusOne indicates the number of bytes that are needed to signal picture display time offsets.

numBytesInPictureCountMinusOne indicates the number of bytes that are needed to signal the maximum number of pictures in a segment.

numBytesInPayloadSizeMinusOne indicates the number of bytes to signal the maximum payload size in bytes.

III.5.3 Content Info Box

III.5.3.1 Definition

Box Type:	\`cinf´
Container:	File
Mandatory:	No
Quantity:	Zero or more

This box gives information about the content of the file.

The box can be repeated at any position of the file when no container box is open.

III.5.3.2 Syntax

```
aligned(8) class contentInfoBox extends box(‘cinf’) {
    unsigned int(64) creationTime;
    unsigned int(64) modificationTime;

    unsigned int(8) titleNumBytes;
    if (titleNumBytes)
        unsigned int(8)[titleNumBytes] title;

    unsigned int(8) authorNumBytes;
    if (authorNumBytes)
        unsigned int(8)[authorNumBytes] author;

    unsigned int(8) copyrightNumBytes;
    if (copyrightNumBytes)
        unsigned int(8)[copyrightNumBytes] copyright;
    unsigned int(16) descriptionNumBytes;
    if (descriptionNumBytes)
        unsigned int(8)[descriptionNumBytes] description;

    unsigned int(16) URINumBytes;
    if (URINumBytes)
        unsigned int(8)[URINumBytes] URI;
}
```

III.5.3.3 Semantics

creationTime is an integer that declares the creation time of the presentation (in seconds since midnight, Jan. 1, 1904).

modificationTime is an integer that declares the most recent time the presentation was modified (in seconds since midnight, Jan. 1, 1904.

titleNumBytes gives the number of bytes in title.

title, if present, contains the title of the file coded according to ISO/IEC 10646-1 UTF-8.

authorNumBytes gives the number of bytes in author.

author, if present, contains the author of the source or the encoded representation in the file coded according to ISO/IEC 10646-1 UTF-8.

copyrightNumBytes gives the number of bytes in copyright.

copyright shall be used only to convey intellectual property information regarding the source or the encoded representation in the file. copyright is coded according to ISO/IEC 10646-1 UTF-8.

descriptionNumBytes gives the number of bytes in description.

description shall be used only to convey descriptive information associated with the information contents of the file. description is coded according to ISO/IEC 10646-1 UTF-8.

URINumBytes gives the number of bytes in URI.

URI contains a uniform resource identifier (URI), as defined in IETF RFC 2396. URI is coded according to ISO/IEC 10646-1 UTF-8. URI shall be used to convey any related information to the file.

III.5.4 Alternate Track Info Box

III.5.4.1 Definition

Box Type:	‘atin’
Container:	File
Mandatory:	Yes
Quantity:	One or more.

This box specifies the characteristics of alternate tracks. The box shall precede the first Segment Box. The box can be repeated at any position of the file when no container box is open.

III.5.4.2 Syntax

```
aligned(8) class alternateTrackInfo {
    unsigned int(16) displayWindowWidth;
    unsigned int(16) displayWindowHeight;
    unsigned int(16) maxSDUSize;
    unsigned int(16) avgSDUSize;
    unsigned int(32) avgBitRate;
}
```

```
aligned(8) class alternateTrackInfoBox
    extends box('atin') {
    (class alternateTrackInfo) trackInfo[numAlternateTracks];
}
```

III.5.4.3 Semantics

displayWindowWidth and displayWindowHeight declare the preferred size of the rectangular area on which video images are displayed. The values are interpreted as amount of pixels.

An SDU is defined as the payload and the payload header. maxSDUSize gives the size in bytes of the largest SDU of the track. avgSDUSize gives the average size of the SDU over the entire track. Value zero in either attribute indicates that no information is available.

avgBitRate gives the average bit-rate in bits/second over the entire track. Payloads and payload headers taken into account in the calculation.

III.5.5 Parameter Set Box

III.5.5.1 Definition

Box Type:	'prms'
Container:	File
Mandatory:	Yes
Quantity:	One or more

This box specifies a parameter set.

Parameter sets can be repeated in the file to allow random access. A parameter set is uniquely identified within a file based on parameterSetID. Decoders can infer a repetition of a parameter set if a set with the same parameterSetID has already appeared in a file. A redundant copy of a parameter set can safely be ignored.

III.5.5.2 Syntax

```
aligned(8) class parameterSetBox
    extends box('prms') {
    unsigned int(16) parameterSetID;
    unsigned int(8) profile;
    unsigned int(8) level;
    unsigned int(8) version;
    unsigned int(16) pictureWidthInMBs;
    unsigned int(16) pictureHeightInMBs;
    unsigned int(16) displayRectangleOffsetTop;
    unsigned int(16) displayRectangleOffsetLeft;
    unsigned int(16) displayRectangleOffsetBottom;
    unsigned int(16) displayRectangleOffsetRight;
    unsigned int(8) displayMode;
    unsigned int(16) displayRectangleOffsetFromWindowTop;
    unsigned int(16) displayRectangleOffsetFromWindowLeftBorder;
    unsigned int(8) entropyCoding;
    unsigned int(8) motionResolution;
    unsigned int(8) partitioningType;
    unsigned int(8) intraPredictionType;
    bit requiredPictureNumberUpdateBehavior;
};
```

III.5.5.3 Semantics

parameterSetId gives the identifier of the parameter set. The identifier shall be unique within a file.

profile defines the coding profile in use.

level defines the level in use within the profile.

version defines the version in use within the profile and the level.

pictureWidthInMBs and pictureHeightInMBs define the extents of the coded picture in macroblocks.

displayRectangleOffsetTop, displayRectangleOffsetLeft, displayRectangleOffsetBottom, and displayRectangleOffsetRight define the rectangle to be displayed from the coded picture. Pixel units are used.

displayMode defines the preferred displaying mode. Value zero indicates that the display rectangle shall be rescaled to fit onto the display window. No scaling algorithm is defined. Image shall be as large as possible, no clipping shall be applied, image aspect ratio shall be maintained, and image shall be centered in the display window. Value one indicates that the display rectangle shall be located as indicated in displayRectangleOffsetFromWindowTop and displayRectangleFromWindowLeftBorder. No scaling shall be done and clipping shall be applied to areas outside the display window. No fill pattern is defined for areas in the display window that are not covered by the display rectangle.

displayRectangleOffsetFromWindowTop and displayWindowOffsetFromWindowLeftBorder indicate the location of the top-left corner of the display rectangle within the display window. The values are given in pixels. The values are valid only if displayMode is one.

FIGURE 30 clarifies the relation of different display rectangle and window related attributes. The dashed rectangle of in the decoded picture represents the display rectangle, which is indicated by displayRectangleOffsetTop, displayRectangleOffsetLeft, displayRectangleOffsetBottom, and displayRectangleOffsetRight.

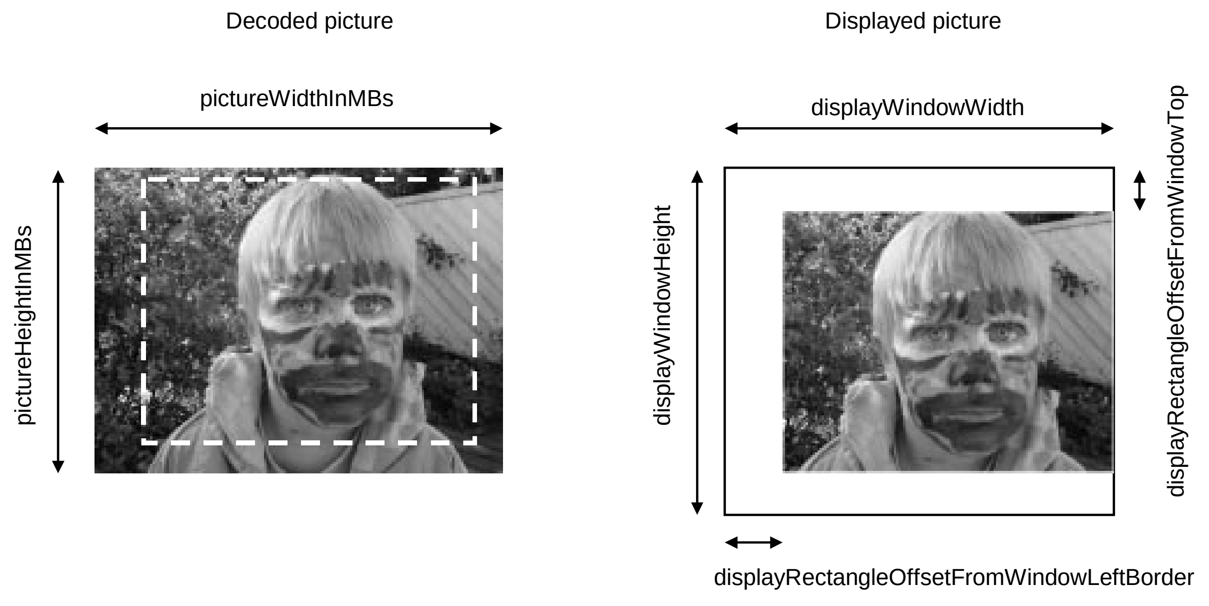


FIGURE 30

Relation of display window and rectangle attributes.

entropyCoding equal to zero stands for UVLC, whereas value one stands for CABAC.

motionResolution equal to zero stands for full-pixel motion resolution, one stands for half-pixel motion resolution, two stands for ¼-pixel motion resolution, and three stands for 1/8-pixel motion resolution.

partitioningType equal to zero stands for the single slice mode and one stands for the data partitioning mode.

intraPredictionType equal to zero stands for normal INTRA prediction, whereas one stands for the constrained INTRA prediction.

If requiredPictureNumberUpdateBehavior equals to one, decoder operation in case of missing picture numbers is normatively specified. Decoder operation is defined in section CROSS-REFERENCE TO BE ADDED.

III.5.6 Segment Box

III.5.6.1 Definition

Box Type:	\`segm'
Container:	File
Mandatory:	Yes
Quantity:	One or more

A segment box contains data from a certain period of time. Segments shall not overlap in time. Segments shall appear in ascending order of time in the file. A segment box is a container box for several other boxes.

III.5.6.2 Syntax

```
aligned(8) class SegmentBox extends Box('segm') {
     unsigned int(64) fileSize;
     unsigned int(64) startTick;
     unsigned int(64) segmentDuration;
}
```

III.5.6.3 Semantics

fileSize indicates the number of bytes from the beginning of the Segment Box to the end of the file. Value zero indicates that no size information is available. When downloading a file to a device with limited storage capabilities, fileSize can be used to determine if a file fits into the available storage space. In a progressive downloading service, fileSize, startTick, and duration (in the File Header Box) can be used to estimate the average bit-rate of the file including meta-data. This estimation can then be used to decide how much initial buffering is needed before starting the playback.

startTick indicates the absolute time of the beginning of the segment since the beginning of the presentation (time zero). Any time offsets within the segment are relative to startTick.

segmentDuration indicates the duration of the segment. Value zero indicates that no duration information is available.

III.5.7 Alternate Track Header Box

III.5.7.1 Definition

Box Type:	'atrh'
Container:	Segment Box ('segm')
Mandatory:	Yes
Quantity:	One or more

An alternate track represents an independent encoding of the same source as for the other alternate tracks. The Alternate Track Header Box contains meta-data for an alternate track. The boxes shall appear in the same order in all Segment Boxes and they can be indexed starting from zero. Each succeeding box is associated with an index one greater than the previous one. The index can be used to associate the box with a particular track and with the information given in the Alternate Track Info Box.

The Alternate Track Header Box is a container box including at least a Picture Information Box and optionally one or more Layer Boxes.

III.5.7.2 Syntax

```
aligned(8) class alternateTrackHeaderBox extends box('atrh') {
    unsigned int(8) numLayers;
}
```

III.5.7.3 Semantics

numLayers indicates the number of layers and Layer Boxes within the Alternate Track Header Box.

III.5.8 Picture Information Box

III.5.8.1 Definition

Box Type:	'pici'
Container:	Alternate Track Header Box ('atrh')
Mandatory:	Yes
Quantity:	One or more

The box contains an indication of the number of the pictures in the alternate track in this segment. In addition, the box contains picture information for each of these pictures. Picture information shall appear in ascending order of picture identifiers (in modulo arithmetic). In other words, picture information shall appear in coding/decoding order of pictures.

A picture information block contains a pointer to the coded representation of the picture. A picture is associated with a display time and with a number of so-called payloads.

A payload refers to a slice, a data partition, or a piece of supplemental enhancement information. A payload header refers to an equivalent definition as in VCEG-N72R1. For example, a payload header of a single slice includes the "first byte", an indication of the parameter set in use, and the slice header.

III.5.8.2 Syntax

```
aligned(8) class payloadInfo {
    unsigned int((numBytesInPayloadSizeMinusOne + 1) * 8) payloadSize;
    unsigned int(8) headerSize;
    unsigned int(4) payloadType;
    unsigned int(1) errorIndication;
    unsigned int(3) reserved = 0;
    if (payloadType == 0) { // single slice
        UVLC parameterSet;
        sliceHeader;
    }
    else if (payloadType == 1) { // partition A
        UVLC parameterSet;
        sliceHeader;
        UVLC sliceID;
    }
    else if (payloadType == 2 || payloadType == 3) { // Partition B or
C
        UVLC pictureID;
        UVLC sliceID;
    }
    else if (payloadType == 5) { // Supplemental enhancement
information
        // no additional codewords
    }
}

aligned(8) class pictureInfo {
     bit intraPictureFlag;
     bit syncPictureFlag;
     aligned(8) int((numBytesInPictureOffsetMinusTwo + 2) * 8)
pictureOffset;
     int((numBytesInPictureDisplayTimeMinusOne + 1) * 8)
pictureDisplayTime;
     if (numLayers) { // from AlternateTrackHeaderBox
          unsigned int(8) layerNumber;
          unsigned int(16) subSequenceIdentifier;
          if (syncPictureFlag) {
               unsigned int(8) originLayerNumber;
               unsigned int(16) originSubSequenceIdentifier;
          }
     }
     unsigned int((numBytesInPayloadCountMinusOne + 1) * 8)
numPayloads;
     (class payloadInfo) payloadData[numPayloads];
}

aligned(8) class PictureInformationBox extends Box('pici') {
     unsigned int((numBytesInPictureCountMinusOne + 1) * 8)
numPictures;
     (class pictureInfo) pictureData[numPictures];
}
```

III.5.8.3 Semantics

payloadInfo gives information related to a payload. payloadSize indicates the number of bytes in the payload (excluding the payload header). The value of headerSize is the number of bytes in the payload header, i.e., the number of bytes remaining in the structure. The rest of the data is defined in VCEG-N72R1.

pictureInfo gives information related to a picture.

intraPictureFlag is set to one, when the picture is an INTRA picture. The flag is zero otherwise.

If syncPictureFlag is one, the coded picture represents the same picture content than the previous or succeeding coded picture having syncPictureFlag equal to one. Any coded representation of a sync picture can be used to recover the reconstructed picture, and no noticeable difference on the reconstructed picture or any picture based on that should occur. Reconstructred pictures that are exactly equal can be reached with SP pictures. The picture number of the sync pictures shall be the same, and, if present, the layer number and sub-sequence identifier of the sync pictures shall be the same.

A picture pointer is maintained to point to the beginning of the latest picture in the corresponding Alternate Track Media Box. The pointer is relative to the beginning of the Alternate Track Media Box. pictureOffset gives the increment or the decrement (in bytes) for the picture pointer to obtain the coded data for the picture. Initially, before updating the pointer for the first picture of the alternate track in a segment, the picture pointer shall be zero.

pictureDisplayTime gives the time when the picture is to be displayed. It is assumed that the picture remains visible until the next picture is to be displayed. The value is relative to the corresponding value of the previous picture.

layerNumber and subSequenceIdentifier are present if numLayers in the Alternate Track Header Box is greater than zero. layerNumber and subSequenceIdentifier identify to which layer and sub-sequence the picture belongs. In case of a sync picture, originLayerNumber and originSubSequence indicate the sub-sequence, based on which the sync picture was created. Notice that the sync picture itself may reside in a different sub-sequence.

numPayloads indicates the number of payloads in the picture. payloadData is an array of payloadInfo structures signaling the characteristics of the payloads.

numPictures indicates the number of pictures in the track during the period of the segment. pictureData is an array of pictureInfo structures signaling the meta-data of the pictures.

III.5.9 Layer Box

III.5.9.1 Definition

Box Type:	‘layr’
Container:	Alternate Track Header Box (‘atrh’)
Mandatory:	No
Quantity:	Zero or more

This box defines the layer information of the pictures in the segment.

Layers can be ordered hierarchically based on their dependency on each other: The base layer is independently decodable. The first enhancement layer depends on some of the data in the base layer. The second enhancement layer depends on some of the data in the first enhancement layer and in the base layer and so on.

A layer number is assigned to layers. Number zero stands for the base layer. The first enhancement layer is associated with number one and each additional enhancement layer increments the number by one. Layer Boxes shall appear in ascending order of layer numbers starting from zero.

A Layer Box contains at least one Sub-Sequence Box.

III.5.9.2 Syntax

```
aligned(8) class LayerBox extends Box(‘layr’) {
    unsigned int(32) avgBitRate;
    unsigned int(32) avgFrameRate;
}
```

III.5.9.3 Semantics

avgBitRate gives the average bit-rate in bits/second of the layer within the segment. Payloads and payload headers taken into account in the calculation. Value zero means an undefined bit-rate.

avgFrameRate gives the average frame rate in frames/(256 seconds) of the layer within the segment. Value zero means an undefined frame rate.

III.5.10 Sub-Sequence Box

III.5.10.1 Definition

Box Type:	‘sseq’
Container:	Layer Box (‘layr’)
Mandatory:	Yes
Quantity:	One or more

This box defines the sub-sequence information of the pictures in a particular layer within a segment.

A sub-sequence shall not depend on any other sub-sequence in the same or in a more enhanced scalability layer. In other words, it shall only depend on one or more sub-sequences in one or more less enhanced scalability layers. A sub-sequence in the base layer can be decoded independently of any other sub-sequences.

A sub-sequence covers a certain period of time within the sequence. Sub-sequences within a layer and in different layers can partly or entirely overlap. A picture shall reside in one layer and in one sub-sequence only.

A sub-sequence identifier is assigned to sub-sequences. Sub-sequences within a particular layer in a segment shall have unique identifiers. If a sub-sequence continues in the next segment, it shall retain its identifier.

III.5.10.2Syntax

```
aligned(8) class dependencyInfo{
    unsigned int(8) layerNumber;
    unsigned int(16) subSequenceIdentifier;
}
aligned(8) class SubSequenceBox extends Box('sseq') {
    unsigned int(16) subSequenceIdentifer;
    bit continuationFromPreviousSegmentFlag;
    bit continuationToNextSegmentFlag;
    bit startTickAvailableFlag;
    aligned(32) unsigned int(64) ssStartTick;
    unsigned int(64) ssDuration;
    unsigned int(32) avgBitRate;
    unsigned int(32) avgFrameRate;
    unsigned int(16) numReferencedSubSequences;
    (class dependencyInfo) dependencyData[numReferencedSubSequences];
}
```

III.5.10.3Semantics

layerNumber and subSequenceIdentifier within the dependencyInfo class identify a sub-sequence that is used as a motion compensation reference for the current sub-sequence.

subSequenceIdentifier in the Sub-Sequence Box gives the identifier for the sub-sequence.

continuationFromPreviousSegmentFlag is equal to one, if the current sub-sequence continues from the previous segment.

continuationToNextSegmentFlag is equal to one, if the current sub-sequence continues in the next segment.

If startTickAvailableFlag equals to zero, the value of ssStartTick and ssDuration is undefined. Otherwise, ssStartTick indicates the start time of the sub-sequence relative to the start time of the segment. ssDuration indicates the duration of the sub-sequence. ssDuration equal to zero indicates an undefined duration.

avgBitRate gives the average bit-rate in bits/second of the sub-sequence within the segment. Payloads and payload headers taken into account in the calculation. Value zero means an undefined bit-rate.

avgFrameRate gives the average frame rate in frames/(256 seconds) of the sub-sequence within the segment. Value zero means an undefined frame rate.

numReferencedSubSequences gives the number of directly referenced sub-sequences. dependencyData is an array of dependencyInfo structures giving the identification information of the referenced sub-sequences.

III.5.11Alternate Track Media Box

III.5.11.1Definition

Box Type:	'atrm'
Container:	Segment Box ('segm')

Mandatory:	Yes
Quantity:	One or more

An alternate track represents an independent encoding of the same source as for the other alternate tracks. The Alternate Track Media Box contains the media-data for an alternate track and for the duration of the segment. The boxes shall appear in the same order in all Segment Boxes and they can be indexed starting from zero. Each succeeding box is associated with an index one greater than the previous one. The index can be used to associate the box with a particular track and with the information given in other track-related boxes.

Pictures can appear in the box in any order. This ensures that disposable pictures, such as conventional B pictures, can be located flexibly. Data for different pictures shall not overlap. Data for a picture consists of payloads, i.e., slices, data partitions, and pieces of supplemental enhancement information. Payloads shall appear in successive bytes, and the order of payloads shall be the same as in the Alternate Track Header Box.

III.5.11.2Syntax

```
aligned(8) class AlternateTrackMediaBox extends Box('atrm') {
}
```

III.5.12Switch Picture Box

III.5.12.1Definition

Box Type:	'swpc'
Container:	Segment Box ('segm')
Mandatory:	No
Quantity:	Zero or one

This box defines which pictures can be used to switch from an alternate track to another. Typically these pictures are SP pictures.

III.5.12.2Syntax

```
aligned(8) class uniquePicture {
    unsigned int(8) alternateTrackIndex;
    unsigned int((numBytesInPictureCountMinusOne + 1) * 8)
pictureIndex;
}

aligned(8) class switchPictureSet {
    unsigned int(8) numSyncPictures;
    (class uniquePicture) syncPicture[numSyncPictures];
}

aligned(8) class switchPictureBox extends Box('swpc') {
    unsigned int((numBytesInPictureCountMinusOne + 1) * 8)
numSwitchPictures;
    (class switchPictureSet) switchPicture[numSwitchPictures];
}
```

III.5.12.3Semantics

uniquePicture uniquely identifies a picture within this segment. It contains two attributes: alternateTrackIndex and pictureIndex. alternateTrackIndex identifies the alternate track where the picture lies, and pictureIndex gives the picture index in coding order.

switchPictureSet gives a set of pictures that represent the same picture contents and can be used to replace any picture in the set as a reference picture for motion compensation. numSyncPictures gives the number of pictures in the set. syncPicture is an array of uniquePicture structures indicating which pictures belong to the set.

numSwitchPictures indicates the number of picture positions that have multiple interchangeable representations. switchPicture is an array of switchPictureSet structures indicating the set of pictures that can be used interchangeably for each picture position.

Appendix IV Non-Normative Error Concealment

IV.1 Introduction

It is assumed that no erroneous or incomplete slices are decoded. When all received slices of a picture have been decoded, skipped slices are concealed according to the presented algorithms. In practice, record is kept in a macroblock (MB) based status map of the frame. The status of an MB in the status map is "Correctly received" whenever the slice that the MB is included in was available for decoding, "Lost" otherwise. After the frame is decoded if the status map contains "Lost" MBs, concealment is started.

Given the slice structure and MB-based status map of a frame, the concealment algorithms were designed to work MB-based. The missing frame area (pixels) covered by MBs marked as "Lost" in the status map are concealed MB-by-MB (16x16 Y pixels, 8x8 U, V pixels). After an MB has been concealed it is marked in the status map as "Concealed". The order in which "Lost" MBs are concealed is important as also the "Concealed", and not only the "Correctly received" MBs are treated as reliable neighbors in the concealment process whenever no "Correctly received" immediate neighbor of a "Lost" MB exists. In such cases a wrong concealment can result in propagation of this concealment mistake to several neighbor concealed MBs. The processing order chosen is to take the MB columns at the edge of the frame first and then move inwards column-by-column so to avoid a concealment mistake made in the usually "difficult" (discontinuous motion areas, large coded prediction error) center part of the frame propagate to the "easy" (continuous motion area, similar motion over several frames) side parts of the frame.

FIGURE 31 shows a snapshot of the status map during the concealment phase where already concealed MBs have the status of "Concealed", and the currently processed (concealed) MB is marked as "Current MB".

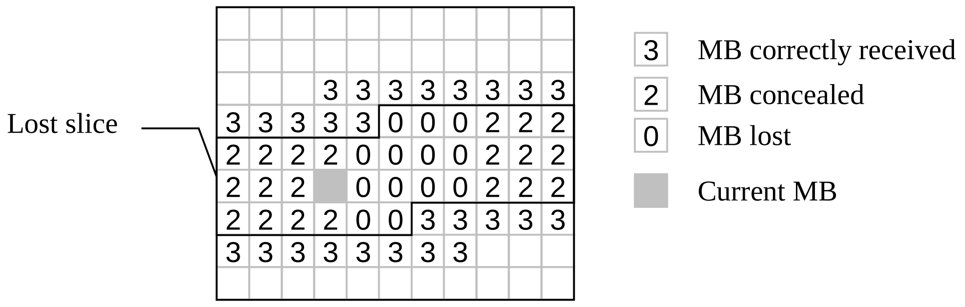


FIGURE 31

MB status map at the decoder

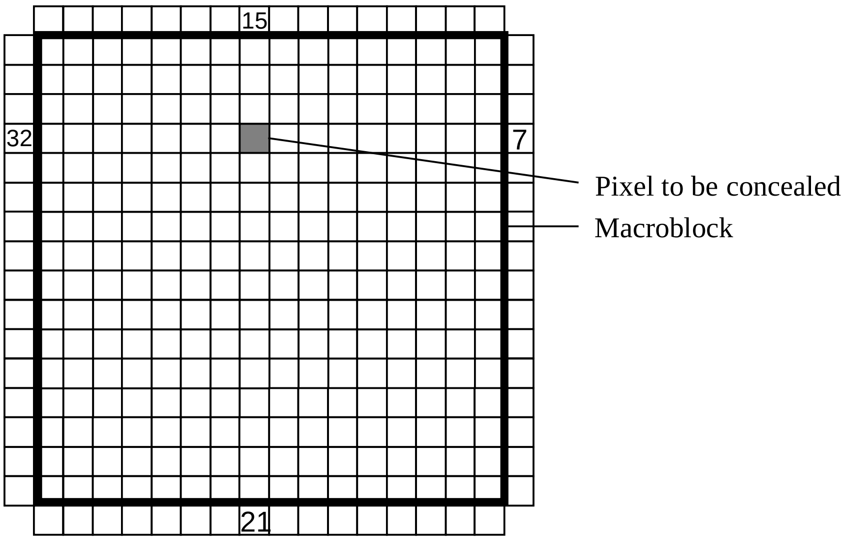
IV.2 INTRA Frame Concealment

Lost areas in INTRA frames have to be concealed spatially as no prior frame may resemble the INTRA frame. The selected spatial concealment algorithm is based on weighted pixel averaging presented in *A. K. Katsaggelos and N. P. Galatsanos (editors), "Signal Recovery Techniques for Image and Video Compression and Transmission", Chapter 7, P. Salama, N. B. Shroff, and E. J. Delp, "Error Concealment in Encoded Video Streams", Kluwer Academic Publishers, 1998.*

Each pixel value in a macroblock to be concealed is formed as a weighted sum of the closest boundary pixels of the selected adjacent macroblocks. The weight associated with each boundary pixel is relative to the inverse distance between the pixel to be concealed and the boundary pixel. The following formula is used:

Pixel value = $(\sum a_i \times (B-d_i)) / \sum(B-d_i)$

where a_i is the pixel value of a boundary pixel in an adjacent macroblock, B is the horizontal or vertical block size in pixels, and d_i is the distance between the destination pixel and the corresponding boundary pixel in the adjacent macroblock.



In

FIGURE 32, the shown destination pixel is calculated as follows

Pixel value = (15x(16-3) + 21x(16-12) + 32x(16-7) + 7x(16-8)) / (13 + 4 + 9 + 8) ≈ 18

Only "Correctly received" neighboring MBs are used for concealment if at least two such MBs are available. Otherwise, neighboring "Concealed" MBs are also used in the averaging operation.

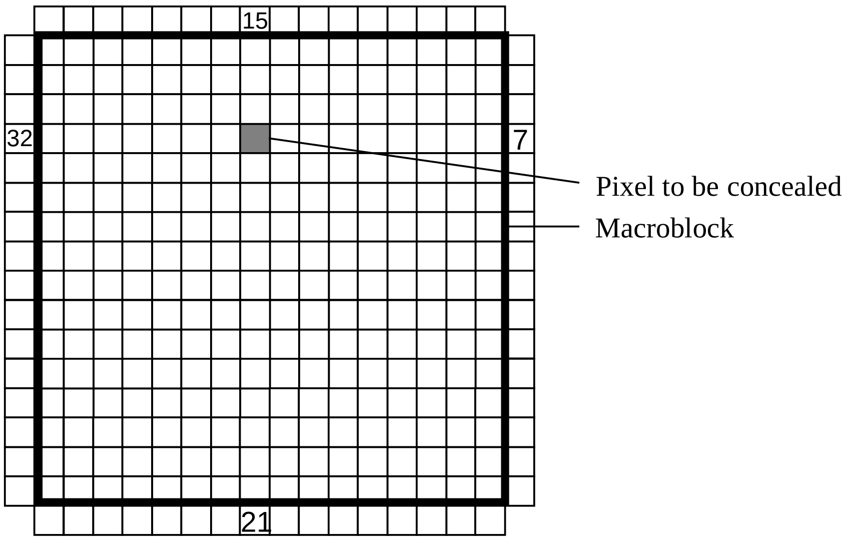


FIGURE 32

Spatial concealment based on weighted pixel averaging

IV.3 INTER and SP Frame Concealment

IV.3.1 General

Instead of directly operating in the pixel domain a more efficient approach is to try to "guess" the motion in the missing pixel area (MB) by some kind of prediction from available motion information of spatial or temporal neighbors. This "guessed" motion vector is then used for motion compensation using the reference frame. The copied pixel values give the final reconstructed pixel values for concealment, and no additional pixel domain operations are used. The presented algorithm is based on *W.-M. Lam, A. R. Reibman, and B. Liu, "Recovery of lost or erroneously received motion vectors," in Proc. ICASSP'93, Minneapolis, Apr. 1993, pp. V417–V420.*

IV.3.2 Concealment using motion vector prediction

The motion activity of the correctly received slices of the current picture is investigated first. If the average motion vector is smaller than a pre-defined threshold (currently ¼ pixels for each motion vector component), all the lost slices are concealed by copying from co-located positions in the reference frame. Otherwise, motion-compensated error concealment is used, and the motion vectors of the lost macroblocks are predicted as described in the following paragraphs.

The motion of a "Lost" MB is predicted from a spatial neighbor MB's motion relying on the statistical observation, that the motion of spatially neighbor frame areas is highly correlated. For example, in a frame area covered by a moving foreground scene object the motion vector field is continuous, which means that it is easy to predict.

The motion vector of the "Lost" MB is predicted from one of the neighbor MBs (or blocks). This approach assumes, that the motion vector of one of the neighbor MBs (or blocks) models the motion in the current MB well. It was found in previous experiments, that median or averaging over all neighbors' motion vectors does not give better results. For simplicity, in the current implementation the smallest neighbor block size that is considered separately as predictor is set to 8x8 Y pixels. The motion of any 8x8 block is calculated as the average of the motion of the spatially corresponding 4x4 or other shaped (e.g. 4x8) blocks.

The decision of which neighbor's motion vectors to use as prediction for the current MB is made based on the smoothness of the concealed (reconstructed) image. During this trial procedure the concealment pixel values are calculated using the motion vector of each candidate (motion compensated pixel values). The motion vector, which results in the smallest luminance change across block boundaries when the block is inserted into its place in the frame is selected. (see FIGURE 33). The zero motion vector case is always considered and this copy concealment (copy pixel values from the co-located MB in the reference frame) is evaluated similarly as the other motion vector candidates.

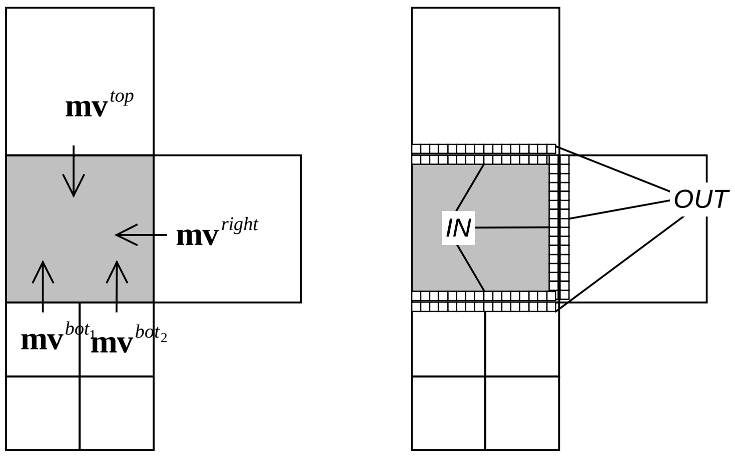


FIGURE 33

Selecting the motion vector for prediction

The winning predictor motion vector is the one which minimizes the side match distortion d_{sm}, which is the sum of absolute Y pixel value difference of the *IN*-block and neighboring *OUT*-block pixels at the boundaries of the current block:

$$\min_{dir\in\{top,bot,left,right\}}\arg\left\langle d_{sm}=\left(\sum_{j=1}^{N}\left|\hat{Y}(\mathbf{mv}^{dir})_j^{IN}-Y_j^{OUT}\right|\right)\Big/N\right\rangle$$

When "Correctly received" neighbor MBs exist the side match distortion is calculated only for them. Otherwise all the "Concealed" neighbor MBs are included in the calculation.

IV.3.3 Handling of Multiple reference frames

When multiple references are used, the reference frame of the candidate motion vector is used as the reference frame for the current MB. That is, when calculating the side match distortion d_{sm}, the IN-block pixels are from the reference frame of the candidate motion vector.

IV.4 B Frame Concealment

A simple motion vector prediction scheme according to the prediction mode of the candidate MB is used as follows:

If the prediction mode of the candidate MB is

- forward prediction mode, use the forward MV as the prediction the same way as for P frames.
- backward prediction mode, use the backward MV as the prediction.
- bi-directional prediction mode, use the forward MV as the prediction, and discard the backward MV.
- direct prediction mode, use the backward MV as the prediction.

Note that 1) Each MV, whether forward or backward, has its own reference frame. 2) An Intra coded block is not used as a motion prediction candidate.

IV.5 Handling of Entire Frame Losses

TML currently lacks H.263 Annex U type of reference picture buffering. Instead, a simple sliding window buffer model is used, and a picture is referred using its index in the buffer. Consequently, when entire frames are lost, the reference buffer needs to be adjusted. Otherwise, the following received frames would use wrong reference frames. To solve this problem, the reference picture ID is used to infer how many frames are lost, and the picture indices in the sliding window buffer are shifted appropriately.