RecA′(u,v) denotes the *final* reconstructed coefficient values for the block immediately above the current block. RecB′(u,v) denotes the *final* reconstructed coefficient values of the block immediately to the left of the current block.

The ability to use the reconstructed coefficient values for blocks A and B in the prediction of the coefficient values for block C depends on whether blocks A and B are in the same video picture segment as block C. A block is defined to be "in the same video picture segment" as another block only if the following conditions are fulfilled:

1) The relevant block is within the boundary of the picture, and

2) If not in Slice Structured mode (see Annex K), the relevant block is either within the same GOB or no GOB header is present for the current GOB, and

3) If in Slice Structured mode, the relevant block is within the same slice.

The block C to be decoded is predicted only from INTRA blocks within the same video picture segment as block C, as shown below.

If Prediction Mode = 0 is used (DC prediction only) and blocks A and B are both INTRA blocks within the same video picture segment as block C, then the DC coefficient of block C is predicted from the average (with truncation) of the DC coefficients of blocks A and B. If only one of the two blocks A and B is an INTRA block within the same video picture segment as block C, then the DC coefficient of only this one of the two blocks is used as the predictor for Prediction Mode = 0. If neither of the two blocks A and B are INTRA blocks within the same video picture segment as block C, then the prediction uses the value of 1024 as the predictor for the DC coefficient.

If Prediction Mode = 1 or 2 (vertical DC & AC or horizontal DC & AC prediction) and the referenced block (block A or block B) is not an INTRA block within the same video picture segment as block C, then the prediction uses the value of 1024 as the predictor for the DC coefficient and the value of 0 as the predictor for the AC coefficients of block C.

A process of "oddification" is applied to the DC coefficient in order to minimize the impact of IDCT mismatch errors. Certain values of coefficients can cause a round-off error mismatch between different IDCT implementations, especially certain values of the (0,0), (0,4), (4,0), and (4,4) coefficients. For example, a DC coefficient of $8k + 4$ for some integer k results in an inverse-transformed block having a constant value $k + 0.5$, for which slight errors can cause rounding in different directions on different implementations.

Define the clipAC() function to indicate clipping to the range –2048 to 2047. Define the clipDC() function to indicate clipping to the range 0 to 2047. Define the oddifyclipDC(x) function as:

```
If (x is even) {
    result = clipDC(x+1)
} else {
    result = clipDC(x)
}
```

The reconstruction for each INTRA prediction mode is then specified as follows, in which the operator "/" is defined as division by truncation:

Mode 0: DC prediction only.

```
RecC′(u,v) = clipAC( RecC(u,v) )          (u,v) ≠ (0,0), u = 0, …, 7, v = 0, …, 7.
If (block A and block B are both INTRA coded and are both in the same video picture segment as
block C) {
    tempDC = RecC(0,0) + ( RecA′(0,0)  + RecB′(0,0)  ) / 2
} else {
```

```
    If (block A is INTRA coded and is in the same video picture segment as block C) {
      tempDC = RecC(0,0) + RecA′(0,0)
    } else {
      If (block B is INTRA coded and is in the same video picture segment as block C) {
        tempDC = RecC(0,0) + RecB′(0,0)
      } else {
        tempDC = RecC(0,0) + 1024
      }
    }
  }
  RecC'(0,0) = oddifyclipDC( tempDC )
```

Mode 1: DC and AC prediction from the block above.

```
  If (block A is INTRA coded and is in the same video picture segment as block C) {
    tempDC = RecC(0,0) + RecA′(0,0)
    RecC′(u,0) = clipAC( RecC(u,0) + RecA′(u,0) )        u = 1, …, 7,
    RecC′(u,v) = clipAC( RecC(u,v) )                     u = 0, …, 7, v = 1, …, 7.
  } else {
    tempDC = RecC(0,0) + 1024
    RecC′(u,v) = clipAC( RecC(u,v) )                     (u,v) ≠ (0,0), u = 0, …,7, v = 0, …, 7
  }
  RecC′(0,0) = oddifyclipDC( tempDC )
```

Mode 2: DC and AC prediction from the block to the left.

```
  If (block B is INTRA coded and is in the same video picture segment as block C) {
    tempDC = RecC(0,0) + RecB′(0,0)
    RecC′(0,v) = clipAC( RecC(0,v) + RecB′(0,v) )        v = 1, …, 7,
    RecC′ (u,v) = clipAC( RecC(u,v) )                    u = 1, …, 7, v = 0, …, 7.
  } else {
    tempDC = RecC(0,0) + 1024
    RecC′(u,v) = clipAC( RecC(u,v) )                     (u,v) ≠ (0,0), u = 0, …, 7, v = 0, …, 7
  }
  RecC'(0,0) = oddifyclipDC( tempDC )
```

ANNEX J

Deblocking Filter mode

J.1 Introduction

This annex describes the use of an optional block edge filter within the coding loop. The main purpose of the block edge filter is to reduce blocking artifacts. The filtering is performed on 8 × 8 block edges. Motion vectors may have either 8 × 8 or 16 × 16 resolution (see J.2). The processing described in this annex applies only for the P-, I-, EP-, or EI-pictures or the P-picture part of an Improved PB-frame. (Possible filtering of B-pictures or the B-picture part of an Improved PB-frame is not a matter for standardization; however, some type of filtering is recommended for improved picture quality.) The capability of this mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

J.2 Relation to UMV and AP modes (Annexes D and F)

The use of the deblocking filter mode has similar effects on picture quality as Overlapped Block Motion Compensation (OBMC) as defined in Annex F when used alone. When both techniques are used together, a further improvement in picture quality can be obtained. The Advanced Prediction mode (see also Annex F) consists of three elements:

1) four motion vectors per macroblock as defined in F.2;

2) overlapped motion compensation for luminance as defined in F.3;

3) motion vectors over picture boundaries as defined in D.1.

In order for the Deblocking Filter mode to be able to provide maximum performance when complexity considerations may prevent use of the OBMC part of the Advanced Prediction mode, the Deblocking Filter mode includes the ability to use four motion vectors per macroblock and motion vectors over picture boundaries.

In summary, the three options defined in Annexes D, F and J contain the following five coding elements:

1) motion vectors over picture boundaries (D.1);

2) extension of motion vector range (D.2);

3) four motion vectors per macroblock (F.2);

4) overlapped motion compensation for luminance (F.3);

5) deblocking edge filter (J.3).

Table J.1 indicates which of the five elements are turned on depending on which of the three options defined in Annexes D, F and J are turned on.

Table J.1/H.263 – Feature Elements for UMV, AP, and DF modes

Unrestricted Motion Vector mode	Advanced Prediction mode	Deblocking Filter mode	Motion vectors over picture boundaries	Extension of motion vector range	Four motion vectors per macroblock	Overlapped motion compensation for luminance	Deblocking edge filter
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
OFF	OFF	ON	ON	OFF	ON	OFF	ON
OFF	ON	OFF	ON	OFF	ON	ON	OFF
OFF	ON	ON	ON	OFF	ON	ON	ON
ON	OFF	OFF	ON	ON	OFF	OFF	OFF
ON	OFF	ON	ON	ON	ON	OFF	ON
ON	ON	OFF	ON	ON	ON	ON	OFF
ON	ON	ON	ON	ON	ON	ON	ON

J.3 Definition of the deblocking edge filter

The filter operations are performed across 8 × 8 block edges at the encoder as well as on the decoder side. The reconstructed image data (the sum of the prediction and reconstructed prediction error) are clipped to the range of 0 to 255 as described in 6.3.2. Then the filtering is applied, which alters the picture that is to be stored in the picture store for future prediction. The filtering operations include an additional clipping to ensure that resulting pixel values stay in the range 0...255. No filtering is performed across a picture edge, and when the Independent Segment Decoding mode is in use, no

filtering is performed across slice edges when the Slice Structured mode is in use (see Annexes K and R) or across the top boundary of GOBs having GOB headers present when the Slice Structured mode is not in use (see Annex R). Chrominance as well as luminance data are filtered.

When the mode described in this annex is used together with the Improved PB-frames mode of Annex M, the backward prediction of the B-macroblock is based on the reconstructed P-macroblock (named P_{REC} in G.5) after the clipping operation but before the deblocking edge filter operations. The forward prediction of the B-macroblock is based on the filtered version of the previous decoded picture (the same picture data that are used for prediction of the P-macroblock).

The deblocking filter operates using a set of four (clipped) pixel values on a horizontal or vertical line of the reconstructed picture, denoted as A, B, C and D, of which A and B belong to one block called block1 and C and D belong to a neighbouring block called block2 which is to the right of or below block1. Figure J.1 shows examples for the position of these pixels.

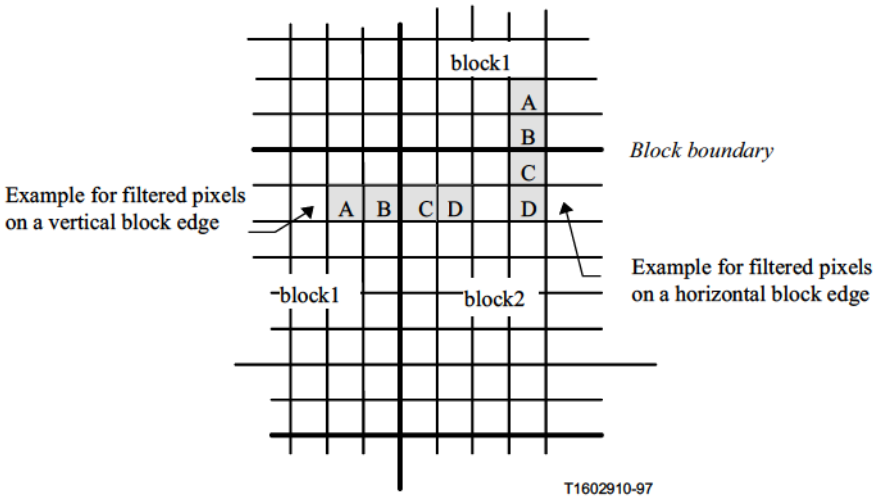


Figure J.1/H.263 – Examples of positions of filtered pixels

One or both of the following conditions must be fulfilled in order to apply the filter across a particular edge:

– Condition 1: block1 belongs to a coded macroblock (COD==0 || MB-type == INTRA); or

– Condition 2: block2 belongs to a coded macroblock (COD==0 || MB-type == INTRA).

If filtering is to be applied across the edge, A, B, C, and D shall be replaced by A1, B1, C1, D1 where:

B1 = clip(B + d1)

C1 = clip(C – d1)

A1 = A – d2

D1 = D + d2

d = (A–4B+4C–D) / 8

d1 = UpDownRamp(d, STRENGTH)

d2 = clipd1((A – D) / 4, d1/2)

UpDownRamp(x, STRENGTH) = SIGN(x) * (MAX(0, abs(x)–MAX(0, 2*(abs(x) – STRENGTH))))

STRENGTH depends on QUANT and determines the amount of filtering. The relation between STRENGTH and QUANT is given in Table J.2.

QUANT = quantization parameter used for block2 if block2 belongs to a coded macroblock, or

QUANT = quantization parameter used for block1 if block2 does not belong to a coded macroblock (but block1 does).

Table J.2/H.263 – Relationship between QUANT and STRENGTH of filter

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

The function clip(x) is defined according to 6.3.2, and the function clipd1(x, lim) clips x to the range ± abs(lim). The symbol "/" denotes division by truncation toward zero.

Figure J.2 shows how the value of d1 varies as a function of d. As a result, the filter has an effect only if d is smaller than 2*STRENGTH (and different from zero). This is to prevent the filtering of strong true edges in the picture content. However, if the Reduced-Resolution Update mode is in use, STRENGTH is set to infinity, and as a result, the value of d1 is always equal to the value of d (see Q.7.2).

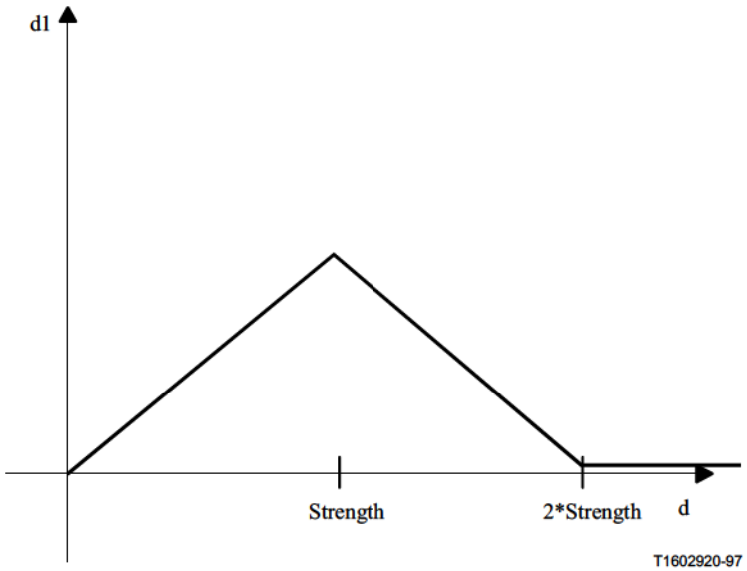


Figure J.2/H.263 – Parameter d1 as a function of parameter d for deblocking filter mode

The definition of d1 is designed to ensure that small mismatches between the encoder and decoder will remain small and will not build up over multiple pictures of a video sequence. This would be a problem, for example, with a condition that simply switches the filter on or off, because a mismatch of only ±1 for d could then cause the filter to be switched on at the encoder side and off at the decoder side, or vice versa.

Due to rounding effects, the order of edges where filtering is performed must be specified.

Filtering across horizontal edges:

Basically this process is assumed to take place first. More precisely, the pixels $\begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix}$ that are used in filtering across a horizontal edge shall not have been influenced by previous filtering across a vertical edge.

Filtering across vertical edges:

Before filtering across a vertical edge using pixels (A, B, C, D), all modifications of pixels (A, B, C, D) resulting from filtering across a horizontal edge shall have taken place.

Note that if one or more of the pixels (A, B, C, D) taking part in a filtering process are outside of a picture, no filtering takes place. Also, if the Independent Segment Decoding mode is in use (see Annex R) and one or more of the pixels (A, B, C, D) taking part in a filtering process are in different video picture segments (see I.3 concerning when a block is considered to be in the same video picture segment), no filtering is performed.

ANNEX K

Slice Structured mode

K.1 Introduction

This annex describes the optional Slice Structured mode of this Recommendation. The capability of this H.263 mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header. In order to facilitate optimal usage in a number of environments, this mode contains two submodes which are also capable of being signalled by external means (for example, Recommendation H.245). These two submodes are used to indicate whether or not rectangular slices will be used and/or whether the slices will be transmitted in sequential order or sent in an arbitrary order.

A slice is defined as a slice header followed by consecutive macroblocks in scanning order. An exception is for the slice that immediately follows the picture start code in the bitstream for a picture (which is not necessarily the slice starting with macroblock 0). In this case only part of the slice header is transmitted as described in K.2. The slice layer defines a video picture segment, and is used in place of the GOB layer in this optional mode. A slice video picture segment starts at a macroblock boundary in the picture and contains a number of macroblocks. Different slices within the same picture shall not overlap with each other, and every macroblock shall belong to one and only one slice.

This mode contains two submodes, which are signalled in the SSS field of the picture header:

1) The Rectangular Slice submode (RS): When RS is in use, the slice shall occupy a rectangular region of width specified by the SWI parameter of the slice header in units of macroblocks, and contains a number of macroblocks in scanning order within the rectangular region. When the Rectangular Slice submode is not in use, the SWI field is not present in the slice header and a slice contains a number of macroblocks in scanning order within the picture as a whole.

2) The Arbitrary Slice Ordering submode (ASO): When ASO is in use, the slices may appear in any order within the bitstream. When ASO is not in use, the slices must be sent in the (unique) order for which the MBA field of the slice header is strictly increasing from each slice to each subsequent slice in the picture.

Slice boundaries are treated differently from simple macroblock boundaries in order to allow slice header locations within the bitstream to act as resynchronization points for bit error and packet loss recovery and in order to allow out-of-order slice decoding within a picture. Thus, no data dependencies can cross the slice boundaries within the current picture, except for the Deblocking Filter mode which, when in use without the Independent Segment Decoding mode, filters across the boundaries of the blocks in the picture. However, motion vectors within a slice can cause data dependencies which cross the slice boundaries in the reference picture used for prediction purposes, unless the optional Independent Segment Decoding mode is in use.

The following rules are adopted to ensure that slice boundary locations can act as resynchronization points and to ensure that slices can be sent out of order without causing additional decoding delays:

1) The prediction of motion vector values are the same as if a GOB header were present (see 6.1.1), preventing the use of motion vectors of blocks outside the current slice for the prediction of the values of motion vectors within the slice.

2) The Advanced INTRA Coding mode (see Annex I) treats the slice boundary as if it were a picture boundary with respect to the prediction of INTRA block DCT coefficient values.

3) The assignment of remote motion vectors for use in overlapped block motion compensation within the Advanced Prediction mode also prevents the use of motion vectors of blocks outside the current slice for use as remote motion vectors (see F.3).

K.2 Structure of slice layer

The structure of the slice layer of the syntax is shown in Figure K.1 for all slices except the slice that immediately follows the picture start code in the bitstream for a picture. For the slice following the picture start code, only the emulation prevention bits (SEPB1, SEPB3, and conditionally SEPB2 as specified below), the MBA field and, when in the RS submode, also the SWI field are included.

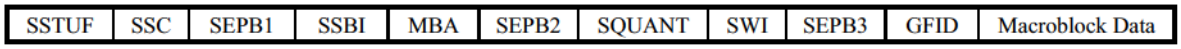


Figure K.1/H.263 – Structure of slice layer

Refer to 5.2.5 for GFID, and to 5.3 for description of Macroblock layer.

K.2.1 Stuffing (SSTUF) (Variable length)

A codeword of variable length consisting of less than 8 bits. Encoders shall insert this codeword directly before an SSC codeword whenever needed to ensure that SSC is byte aligned. If SSTUF is present, the last bit of SSTUF shall be the last (least significant) bit of a byte, so that the start of the SSC codeword is byte aligned. Decoders shall be designed to discard SSTUF. Notice that 0 is used for stuffing within SSTUF.

K.2.2 Slice Start Code (SSC) (17 bits)

A word of 17 bits. Its value is 0000 0000 0000 0000 1. Slice start codes shall be byte aligned. This can be achieved by inserting SSTUF before the start code such that the first bit of the start code is the first (most significant) bit of a byte. The slice start code is not present for the slice which follows the picture start code.

K.2.3 Slice Emulation Prevention Bit 1 (SEPB1) (1 bit)

A single bit always having the value of "1", which is included to prevent start code emulation.

K.2.4 Slice Sub-Bitstream Indicator (SSBI) (4 bits)

A codeword of length four bits which is present only when CPM = "1" in the picture header. SSBI indicates the Sub-Bitstream number for the slice for Continuous Presence Multipoint and Video Multiplex operation, as described in Annex C. The mapping from the value of SSBI to the Sub-Bitstream number is shown in Table K.1. SSBI is not present for the slice which follows the picture start code.

Table K.1/H.263 – Values of SSBI and associated Sub-Bitstream numbers

Sub-Bitstream number	SSBI field value	GN value emulated
0	1001	25
1	1010	26
2	1011	27
3	1101	29

K.2.5 Macroblock Address (MBA) (5/6/7/9/11/12/13/14 bits)

A codeword having a length that is dependent on the current picture size and whether the Reduced-Resolution Update mode is in effect (see Annex Q). The bits are the binary representation of the macroblock number of the first macroblock in the current slice as counted from the beginning of the picture in scanning order, starting with macroblock number 0 at the upper left corner. MBA uniquely identifies which macroblock in the picture the current slice starts with. Codeword lengths for this codeword are provided in Table K.2. For custom picture sizes, the field width is given by the first entry in the table that has an equal or larger number of macroblocks, and the maximum value is the number of macroblocks in the current picture minus one. When in the Reduced-Resolution Update mode, the relevant picture size is the lower resolution update picture size rather than the picture size indicated in the picture header (see Annex Q).

Table K.2/H.263 – Specification of MBA parameter

	Default		RRU mode	
Picture format	**Max value**	**Field width**	**Max value**	**Field width**
sub-QCIF	47	6	11	5
QCIF	98	7	29	6
CIF	395	9	98	7
4CIF	1583	11	395	9
16CIF	6335	13	1583	11
2048 × 1152	9215	14	2303	12

K.2.6 Slice Emulation Prevention Bit 2 (SEPB2) (1 bit)

A single bit always having the value "1", which is included under certain conditions to prevent start code emulation. For slices other than the one following the picture start code, SEPB2 is included only if the MBA field width is greater than 11 bits and CPM = "0" in the picture header, or if the MBA field width is greater than 9 bits and CPM = "1" in the picture header. For the slice following the picture start code, SEPB2 is included only if the Rectangular Slice submode is in use.

K.2.7 Quantizer Information (SQUANT) (5 bits)

A fixed length codeword of five bits which indicates the quantizer QUANT to be used for that slice until updated by any subsequent DQUANT. The codewords are the natural binary representations of the values of QUANT which, being half the step sizes, range from 1 to 31. SQUANT is not present for the slice which follows the picture start code.

K.2.8 Slice Width Indication in Macroblocks (SWI) (3/4/5/6/7 bits)

A codeword which is present only if the Rectangular Slice submode is active, and having a length which depends on the current picture size and whether the Reduced-Resolution Update mode is active, as specified in Table K.3. For custom picture sizes, the field width is given by the next standard format size which is equal or larger in width (QCIF, CIF, …), and the maximum value is the total number of macroblocks across the picture minus one. The last row of the table indicates the field width for picture sizes wider than 16CIF. SWI refers to the width of the current rectangular slice having its first macroblock (upper left) specified by MBA. The calculation of the actual slice width is given by

$$\text{Actual Slice Width} = \text{SWI} + 1$$

When in the Reduced-Resolution Update mode, the relevant picture size is that of the lower resolution picture size for the update information, rather than the picture size indicated in the picture header.

Table K.3/H.263 – Specification of SWI parameter

	Default		RRU mode	
Picture format	**Max value**	**Field width**	**Max value**	**Field width**
sub-QCIF	7	4	3	3
QCIF	10	4	5	3
CIF	21	5	10	4
4CIF	43	6	21	5
16CIF	87	7	43	6
1412...2048 pixels wide	127	7	63	6

K.2.9 Slice Emulation Prevention Bit 3 (SEPB3) (1 bit)

A single bit always having the value of "1" in order to prevent start code emulation.

ANNEX L

Supplemental Enhancement Information Specification

L.1 Introduction

This annex describes the format of the supplemental enhancement information sent in the PSUPP field of the picture layer of this Recommendation. The capability of a decoder to provide any or all of the enhanced capabilities described in this annex may be signalled by external means (for example Recommendation H.245). Decoders which do not provide the enhanced capabilities may simply discard any PSUPP information bits that appear in the bitstream. The presence of this supplemental enhancement information is indicated in PEI, and an additional PEI bit is inserted between every octet of PSUPP data, as described in 5.1.24 and 5.1.25.

In this annex, a distinction is made between the "decoded picture" and the "displayed picture". For purposes of this annex, the "displayed picture" is a picture having the same picture format as specified for the current picture by the picture layer of the video bitstream syntax. The "displayed picture" is constructed as described in this annex from the decoded picture, the prior displayed picture, the supplementary enhancement information described herein, and in some cases partly from a background picture which is externally controlled.

L.2 PSUPP format

The PSUPP data consists of a four-bit function type indication FTYPE, followed by a four-bit parameter data size specification DSIZE, followed by DSIZE octets of function parameter data, optionally followed by another function type indication, and so on. One function type indication value is defined as an escape code to provide for future extensibility to allow definition of more than fifteen different functions. A decoder which receives a function type indication which it does not support can discard the function parameter data for that function and then check for a subsequent

function type indication which may be supported. The defined FTYPE values are shown in Table L.1.

Table L.1/H.263 – FTYPE function type values

0	Reserved
1	Do Nothing
2	Full-Picture Freeze Request
3	Partial-Picture Freeze Request
4	Resizing Partial-Picture Freeze Request
5	Partial-Picture Freeze-Release Request
6	Full-Picture Snapshot Tag
7	Partial-Picture Snapshot Tag
8	Video Time Segment Start Tag
9	Video Time Segment End Tag
10	Progressive Refinement Segment Start Tag
11	Progressive Refinement Segment End Tag
12	Chroma Key Information
13	Reserved
14	Reserved
15	Extended Function Type

L.3 Do Nothing

No action is requested by the Do Nothing function. This function is used to prevent start code emulation. Whenever the last five or more bits of the final octet of the previous PSUPP octet are all zero and no additional PSUPP function requests are to be sent, the Do Nothing function shall be inserted into PSUPP to prevent the possibility of start code emulation. The Do Nothing function may also be sent when it is not required by rule expressed in the previous sentence. DSIZE shall be zero for the Do Nothing function.

L.4 Full-Picture Freeze Request

The full-picture freeze request function indicates that the contents of the entire prior displayed video picture shall be kept unchanged, without updating the displayed picture using the contents of the current decoded picture. The displayed picture shall then remain unchanged until the freeze picture release bit in the current PTYPE or in a subsequent PTYPE is set to 1, or until timeout occurs, whichever comes first. The request shall lapse due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of another full-picture freeze request prior to or upon expiration of the timeout period (e.g. repeating the request in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be zero for the full-picture freeze request function.

L.5 Partial-Picture Freeze Request

The partial-picture freeze request function indicates that the contents of a specified rectangular area of the prior displayed video picture should be kept unchanged, without updating the specified area of the displayed picture using the contents of the current decoded picture. The specified area of the displayed picture shall then remain unchanged until the freeze picture release bit in the current

PTYPE or in a subsequent PTYPE is set to 1, until a partial-picture freeze-release request affecting the specified area is received, until the source format specified in a picture header differs from that of previous picture headers, or until timeout occurs, whichever comes first. Any change in the picture source format shall act as a freeze release for all active partial-picture freeze requests. The request shall lapse due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical partial-picture freeze request prior to or upon expiration of the timeout period (e.g. repeating the request in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 4 for the partial-picture freeze request. The four octets of PSUPP that follow contain the horizontal and vertical location of the upper left corner of the frozen picture rectangle, and the width and height of the rectangle, respectively, using eight bits each and expressed in units of eight pixels. For example, a 24-pixel wide and 16 pixel tall area in the upper left corner of the video display is specified by the four parameters (0, 0, 3, 2).

L.6 Resizing Partial-Picture Freeze Request

The resizing partial-picture freeze request function indicates that the contents of a specified rectangular area of the prior displayed video picture should be resized to fit into a smaller part of the displayed video picture, which should then be kept unchanged, without updating the specified area of the displayed picture using the contents of the current decoded picture. The specified area of the displayed picture shall then remain unchanged until the freeze release bit in the current PTYPE or in a subsequent PTYPE is set to 1, until a partial-picture freeze-release request affecting the specified area is received, until the source format specified in a picture header differs from that of previous picture headers, or until timeout occurs, whichever comes first. Any change in the picture source format shall act as a freeze release for all active resizing partial-picture freeze requests. The request shall lapse due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of a partial-picture freeze request for the affected area of the displayed picture prior to or upon expiration of the timeout period (e.g. issuing a partial-picture freeze request in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 8 for the resizing partial-picture freeze request. The eight octets of PSUPP data that follow contain 32 bits used to specify the rectangular region of the affected area of the displayed picture, and then 32 bits used to specify the corresponding rectangular region of the affected area of the decoded picture. The width and height of the rectangular region in the decoded picture shall both be equal to 2^i times the width and height specified for the rectangular region in the displayed picture, where i is an integer in the range of 1 to 8. The location and size of each of these two rectangular regions is specified using the same format as such a region is specified in the partial-picture freeze request function.

L.7 Partial-Picture Freeze-Release Request

The partial-picture freeze-release request function indicates that the contents of a specified rectangular area of the displayed video picture shall be updated by the current and subsequent decoded pictures. DSIZE shall be equal to 4 for the partial-picture freeze-release request. The four octets of PSUPP data that follow specify a rectangular region of the displayed picture in the same format as such a region is specified in the partial-picture freeze request function.

L.8 Full-Picture Snapshot Tag

The full-picture snapshot tag function indicates that the current picture is labelled for external use as a still-image snapshot of the video content. DSIZE shall be equal to 4 for the full-picture snapshot

tag function. The four octets of PSUPP data that follow specify a snapshot identification number for external use.

L.9 Partial-Picture Snapshot Tag

The partial-picture snapshot tag function indicates that a specified rectangular area of the current picture is labelled for external use as a still-image snapshot of the video content. DSIZE shall be equal to 8 for the partial-picture snapshot tag function. The first four octets of PSUPP data that follow specify a snapshot identification number for external use, and the remaining four octets of PSUPP data that follow specify a rectangular region of the decoded picture in the same format as such a region is specified in the partial-picture freeze request function.

L.10 Video Time Segment Start Tag

The video time segment start tag function indicates that the beginning of a specified sub-sequence of video data is labelled as a useful section of video content for external use, starting with the current picture. The tagged sub-sequence of video data shall continue until stopped by the receipt of a matching video time segment end tag function or until timeout, whichever comes first. The tagged sub-sequence shall end due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical video time segment start tag function prior to or upon expiration of the timeout period (e.g. repeating the video time segment start tag function in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a video time segment identification number for external use.

L.11 Video Time Segment End Tag

The video time segment end tag function indicates that the end of a specified sub-sequence of video data is labelled as a useful section of video content for external use, ending with the previous picture. DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a video time segment identification number for external use.

L.12 Progressive Refinement Segment Start Tag

The progressive refinement segment start tag function indicates the beginning of a specified sub-sequence of video data which is labelled as the current picture followed by a sequence of zero or more pictures of refinement of the quality of the current picture, rather than as a representation of a continually moving scene. The tagged sub-sequence of video data shall continue until stopped by the receipt of a matching progressive refinement segment end tag function or until timeout, whichever comes first. The tagged sub-sequence shall end due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical progressive refinement segment start tag function prior to or upon expiration of the timeout period (e.g. repeating the progressive refinement start tag function in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a progressive refinement segment identification number for external use.

L.13 Progressive Refinement Segment End Tag

The progressive refinement segment end tag function indicates the end of a specified sub-sequence of video data which is labelled as an initial picture followed by a sequence of zero or more pictures

of the refinement of the quality of the initial picture, and ending with the previous picture. DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a progressive refinement identification number for external use.

L.14 Chroma Keying Information

The Chroma Keying Information Function (CKIF) indicates that the "chroma keying" technique is used to represent "transparent" and "semi-transparent" pixels in the decoded video pictures. When being presented on the display, "transparent" pixels are not displayed. Instead, a background picture which is either a prior reference picture or is an externally controlled picture is revealed. Semi-transparent pixels are displayed by blending the pixel value in the current picture with the corresponding value in the background picture. One octet is used to indicate the keying color value for each component (Y, C_B, or C_R) which is used for chroma keying. To represent pixels that are to be "semi-transparent", two threshold values, denoted as T_1 and T_2, are used. Let α denote the transparency of a pixel; $\alpha = 255$ indicates that the pixel is opaque, and $\alpha = 0$ indicates that the pixel is transparent. For other values of α, the resulting value for a pixel should be a weighted combination of the pixel value in the current picture and the pixel value from the background picture (which is specified externally). The values of α may be used to form an image that is called an "alpha map." Thus, the resulting value for each component may be:

$$[\alpha \cdot X + (255 - \alpha) \cdot Z] / 255$$

where X is the decoded pixel component value (for Y, C_B, or C_R), and Z is the corresponding pixel component value from the background picture.

The α value can be calculated as follows. First, the distance of the pixel color from the key color value is calculated:

$$d = A_Y(X_Y - K_Y)^2 + A_B(X_B - K_B)^2 + A_R - K_R)^2$$

in which X_Y, X_B, and X_R are the Y, C_B, and C_R values of the decoded pixel color, K_Y, K_B, and K_R are the corresponding key color parameters, and A_Y, A_B, and A_R. are keying flag bits which indicate which color components are used as keys. Once the distance d is calculated, the α value may be computed as specified in the following pseudo-code:

```
for each pixel
   if       (d<T1) then α = 0;
   else if  (d>T2) then α = 255;
   else α = [255·(d-T1)]/(T2-T1)
```

However, the precise method for performing the chroma keying operation in the decoder is not specified herein, since normative specification of the method is not needed for interoperability. The process described here is provided for illustration purposes in order to convey the intended interpretation of the data parameters.

Since the derived α value is simply a function of X_Y, X_B, and X_R, a Look-Up Table (LUT) can be built to achieve the above operation. Such an LUT has $2^{8\times N}$ entries corresponding to all pixel values, where N is the number of color components used as keys. Each entry in the LUT would then contain the corresponding α value.

DSIZE shall be in the range of 1 to 9 (inclusive) for chroma keying information, according to the amount of data sent with the CKIF. No more than one CKIF shall be sent with a picture.

The first octet following the DSIZE octet shall contain the representation order of the current picture – streams having a lower representation order are assumed to form the background picture for streams having a higher representation order.

If DSIZE is greater than one, the next octet after the representation order octet shall be used to send six flag bits defined as:

bit 1: A_Y: A flag bit indicating the presence of a K_Y key parameter for luminance Y values

bit 2: A_B: A flag bit indicating the presence of a K_B key parameter for chroma C_B values

bit 3: A_R: A flag bit indicating the presence of a K_R key parameter for chroma C_R values

bit 4: A_1: A flag bit indicating the presence of a T_1 threshold parameter for transparency

bit 5: A_2: A flag bit indicating the presence of a T_2 threshold parameter for opacity

bit 6: RPB: A flag bit indicating the use of the reference picture as a background picture

bit 7: reserved

bit 8: reserved

DSIZE shall be equal to 1 or shall be equal to 2 plus the number of flag bits among A_Y, A_B, and A_R which are set to 1, plus 2 times the number of bits among A_1 and A_2 which are set to 1. If DSIZE is greater than 1, then an additional octet shall be sent to specify the value of each color component for each of the flag bits A_Y, A_B, and A_R which are set to one and two additional octets shall be sent to specify each of the flagged threshold values among T_1 and T_2. These octets shall follow in the same order as the flag bits.

If DSIZE is equal to 1, or if all three keying color flag bits A_Y, A_B, and A_R, are zero, the keying flag bits A_Y, A_B, and A_R and keying colours K_Y, K_B, and K_R which were used for the previous keyed picture should also be used for the current picture. If no previous values have been sent for the video sequence, the default keying flag bits $A_Y = 1$, $A_B = 1$, and $A_R = 1$ and the default key colours $K_Y = 50$, $K_B = 220$, and $K_R = 100$ should be used as the previous values.

If DSIZE is equal to 1, or if both of the keying threshold flag bits A_1 and A_2 are zero, the keying threshold values T_1 and T_2 which were used for the previous keyed picture should also be used for the current picture. If no previous values have been sent for the video sequence, the default threshold values $T_1 = 48$ and $T_2 = 75$ should be used as the previous values.

In the portion where the pixels are "semi-transparent" (i.e. when $T_1 < d < T_2$), the decoded pixels typically contain the chroma key color in components where chroma keys are used. This may result in certain color artifacts. To address this problem, these pixel values may be adjusted before they are blended with the background color. Such a correction process may be applied to color components being used in the chroma key operation as indicated by the flag bits. The process is as follows:

$$X' = K + (T_2 / d)(X - K)$$

where X is the original decoded pixel component value, and X' is the corrected value.

Since the adjusted pixel values X'_Y, X'_B, and X'_R are functions of X_Y, X_B, and X_R, the color correction can be achieved by using an LUT. This LUT would have 2^{8N} entries corresponding to all pixel values, where N is the number of color components used as keys. Each entry would then contain the corresponding corrected values.

If the Reference Picture Background (RPB) flag bit is set to "1", this indicates that the temporally previous reference picture (prior to any Annex P resampling performed for the current picture) should be held as the (opaque) background of the current picture and of all subsequent chroma keyed pictures, until replaced by the arrival of another picture having an RPB flag set to "1". If the current picture has no temporally previous reference picture (i.e if the current picture is a picture of type

INTRA or EI), the picture to which the RPB flag bit refers is the picture which would normally have been the reference picture if the current picture were of type INTER or EP as appropriate. If the RPB flag bit is set to "0", this indicates that background should remain as previously controlled (either under external control or using a reference picture which was previously stored upon receipt of a prior picture having RPB set to "1").

The use of the chroma keying which is invoked by the issuance of the chroma keying information function shall start with the current picture and shall continue until a subsequent picture of type INTRA or EI occurs or until a timeout period expires, whichever comes first. The use of chroma keying shall end due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical chroma keying information function prior to or upon expiration of the timeout period (e.g. repeating the chroma keying information function in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). The encoder shall send sufficient information with the chroma keying information function for complete resynchronization to occur with each picture of type INTRA or EI and within each timeout interval (it shall not rely on using stored or default values of the key colours or thresholds).

L.15 Extended Function Type

The extended function type indication is used to signal that the following PSUPP octet contains an extended function. The usage of extended functions is reserved for the ITU to have a later ability to define a larger number of backward-compatible PSUPP data functions. DSIZE shall be equal to zero for the extended function type indication. In order to allow backward compatibility of future use of the extended function type indication, decoders shall treat the second set of four bits in the octet which follows the extended function type indication as a DSIZE value indicating the number of subsequent octets of PSUPP that are to be skipped for extended function parameter data, which may be followed by additional FTYPE indications.

ANNEX M

Improved PB-frames mode

M.1 Introduction

This annex describes an optional Improved PB-frames mode of this Recommendation. It is therefore considered to be advantageous to use the present Improved PB-frames mode instead of the PB-frames mode defined in Annex G. The capability of this mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

Most parts of this option are similar to the PB-frame option defined in Annex G.

To avoid confusion with B-pictures as defined in Annex O, the terms B-picture, B-macroblock, and B-block will not be used in this annex. Instead, we will use the notation B_{PB} to mean the "B-part" of an Improved PB-frame. When reference is made to Annex G, B-picture and B-block shall be read as B_{PB}-picture and B_{PB}-block.

The main difference between the PB-frames mode and the Improved PB-frames mode is that in the Improved PB-frames mode, the B_{PB}-macroblock has available a forward and a backward prediction mode in addition to the bidirectional prediction mode. In this annex, MVDB (when present) refers to a forward motion vector. (Note that, in Annex G, MVDB was used to refer to an enhancement of the

downscaled forward and backward vectors for bidirectional prediction, rather than a distinct forward motion vector.)

All the differences from Annex G are identified in this annex. When nothing is indicated, it means that the same procedure as described in Annex G is used.

M.2 B_{PB}-macroblock prediction modes

There are three different ways of coding a B_{PB}-macroblock. The different coding modes are signalled by the parameter MODB. The B_{PB}-macroblock coding modes are:

M.2.1 Bidirectional prediction

In the bidirectional prediction mode, prediction uses the reference pictures before and after the B_{PB}-picture (in the case of a sequence of Improved PB-frames, this means the P-picture part of the temporally previous Improved PB-frame and the P-picture part of the current Improved PB-frame). This prediction is equivalent to the prediction defined in Annex G when $MV_D = 0$. Notice that in this mode (and only in this mode), Motion Vector Data (MVD) of the PB-macroblock must be included even if the P-macroblock is INTRA coded. (Notice the difference between MVD – motion vector data – and MV_D – delta vector – defined in Annex G.)

M.2.2 Forward prediction

In the forward prediction mode, the vector data contained in MVDB are used for forward prediction from the previous reference picture (an INTRA or INTER picture, or the P-picture part of a PB or Improved PB-frame). This means that there is always only one 16×16 vector for the B_{PB}-macroblock in this prediction mode.

A simple predictor is used for coding of the forward motion vector. The rule for this predictor is that if the current macroblock is not at the far left edge of the picture or slice and the macroblock to the left has a forward motion vector, then the predictor of the forward motion vector for the current macroblock is set to the value of the forward motion vector of the block to the left; otherwise, the predictor is set to zero. The difference between the predictor and the desired motion vector is then VLC coded in the same way as vector data to be used for the P-picture (MVD).

Concerning motion vectors over picture boundaries defined in D.1, the described technique also applies for the forward B_{PB}-vector if this feature is in use (this applies for forward as well as for bidirectional prediction mode).

M.2.3 Backward prediction

In the backward prediction mode, the prediction of the B_{PB} macroblock is identical to P_{REC} (defined in G.5). No motion vector data is used for the backward prediction.

M.3 Calculation of vectors for bidirectional prediction of a the B-macroblock

In case bidirectional prediction is used, the scaled forward and backward vectors are calculated as described in Annex G when $MV_D = 0$.

M.4 MODB table

A new definition for MODB (replacing Table 11) is shown in Table M.1. It indicates the possible coding modes for a B_{PB}-block.

Table M.1/H.263 – MODB table for Improved PB-frames mode

Index	CBPB	MVDB	Number of bits	Code	Coding mode
0			1	0	Bidirectional prediction
1	x		2	10	Bidirectional prediction
2		x	3	110	Forward prediction
3	x	x	4	1110	Forward prediction
4			5	11110	Backward prediction
5	x		5	11111	Backward prediction
NOTE – The symbol "x" indicates that the associated syntax element is present.					

ANNEX N

Reference Picture Selection mode

N.1 Introduction

This annex describes the optional Reference Picture Selection mode of this Recommendation, which operates using a modified interframe prediction method called "NEWPRED." The capability of this H.263 mode is signalled by external means (for example, Recommendation H.245). The amount of additional picture memory accommodated in the decoder may also be signalled by external means to help the memory management at the encoder. This mode can use backward channel messages sent from a decoder to an encoder to inform the encoder which part of which pictures have been correctly decoded at the decoder. The use of this mode is indicated in the PLUSPTYPE field of the picture header. This mode has two back-channel mode switches which define whether a backward channel is used and what kind of messages are returned on that backward channel from the decoder, and has another submode defined in terms of the channel for the backward channel messages.

The two back-channel mode switches of this mode determine the type of messages sent on the back-channel, specifying whether ACK (Acknowledgment messages), or NACK (Non-Acknowledgment messages) are sent. Together, the two switches define four basic methods of operation:

1) NEITHER, in which no back-channel data is returned from the decoder to the encoder;
2) ACK, in which the decoder returns only acknowledgment messages;
3) NACK, in which the decoder returns only non-acknowledgment messages; and
4) ACK+NACK, in which the decoder returns both acknowledgment and non-acknowledgment messages.

The specific type of messages to be sent as outlined above is indicated in the picture header.

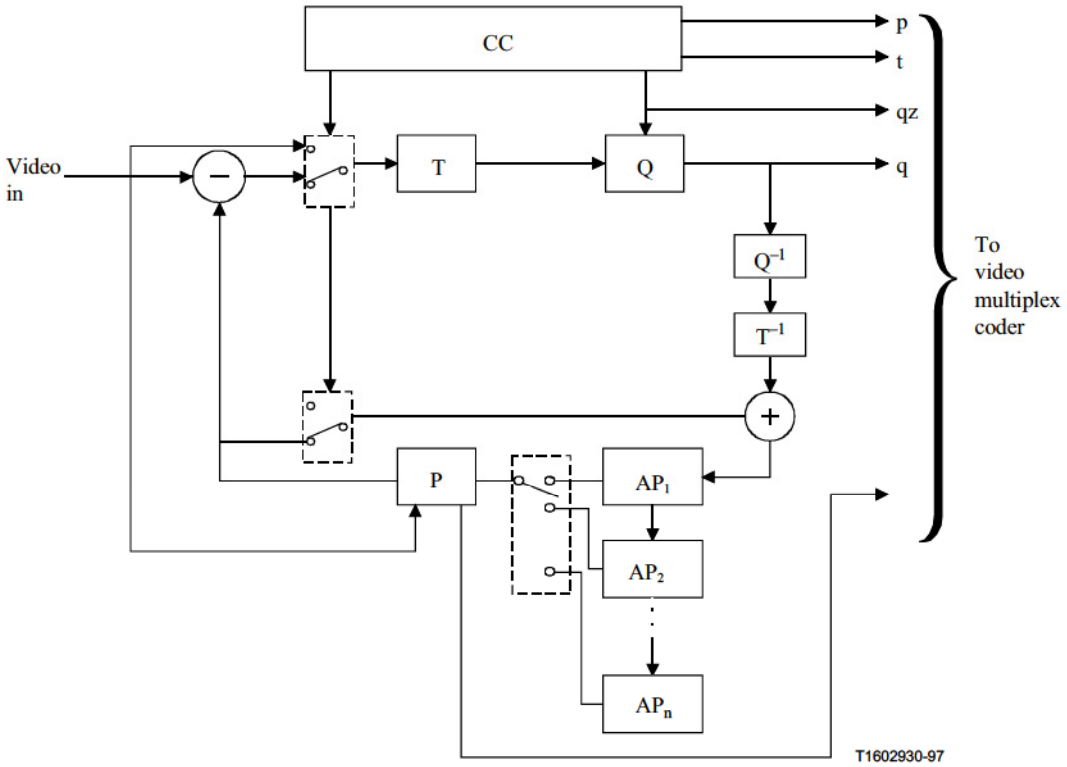
There are also two methods of operation in terms of the channel for backward channel messages:

1) Separate Logical Channel mode: This method of operation delivers back-channel data through a separate logical channel in the multiplex layer of the system, and
2) VideoMux mode: This method of operation delivers back-channel data for received video within the forward video data of a video stream of encoded data.

This annex specifies a syntax for the backward channel messages as well as for forward-channel data.

N.2 Video source coding algorithm

The source coder of this mode is shown in generalized form in Figure N.1. This figure shows a structure which uses a number of picture memories. The source coder may select one of the picture memories to suppress the temporal error propagation due to the inter-frame coding. The Independent Segment Decoding mode (see Annex R), which treats boundaries of GOBs with non-empty headers or slices as picture boundaries, can be used to avoid error propagation due to motion compensation across the boundaries of the GOBs or slices when this mode is applied to a smaller unit than a picture, such as a GOB or slice. The information to signal which picture is selected for prediction is included in the encoded bitstream. The strategy used by the encoder to select the picture to be used for prediction is out of the scope of this Recommendation.



T	Transform
Q	Quantizer
P	Picture Memory with motion compensated variable delay
AP	Additional Picture Memory
CC	Coding control
p	Flag for INTRA/INTER
t	Flag for transmitted or not
qz	Quantizer indication
q	Quantizing index for transform coefficients
v	Motion vector

Figure N.1/H.263 – Source coder for NEWPRED

N.3 Channel for back-channel messages

This mode has two methods of operation in terms of the type of channel for back-channel messages. One is the separate logical channel mode and the other is the videomux mode. The separate logical channel mode is a preferred mode and delivers the back-channel message defined in N.4.2 through the dedicated logical channel. The videomux mode is prepared for the system which cannot set up

the separate extra channel for the back-channel messages due to the restriction of the number of channels' combinations. The videomux mode delivers the back-channel messages through the same logical channel with the forward video data in the opposite direction.

N.3.1 Separate logical channel mode

The separate logical channel mode delivers back-channel messages through a dedicated logical channel opened only for the purpose of the back-channel messages. The association mechanism with the forward channel which delivers video data is provided by external means (for example, Recommendation H.245). Separate logical channel operation requires an external framing mechanism for synchronization of the messages within the back channel, as the back-channel syntax defined herein contains no synchronization flag words.

N.3.2 Videomux mode

The videomux mode delivers the back-channel messages through the same logical channel with the forward video data in the opposite direction. The syntax of the multiplexed bitstream is described in N.4.1. The back-channel messages may be inserted by using the Back-Channel message Indication (BCI) in the GOB or slice header.

N.4 Syntax

N.4.1 Forward channel

The syntax for the forward-channel data which conveys the compressed video signal is altered only in the Group of Blocks (GOB) or slice layer.

The syntax of the GOB layer is illustrated in Figure N.2. The fields of TRI, TR, TRPI, TRP, BCI, and BCM are added to Figure 9.

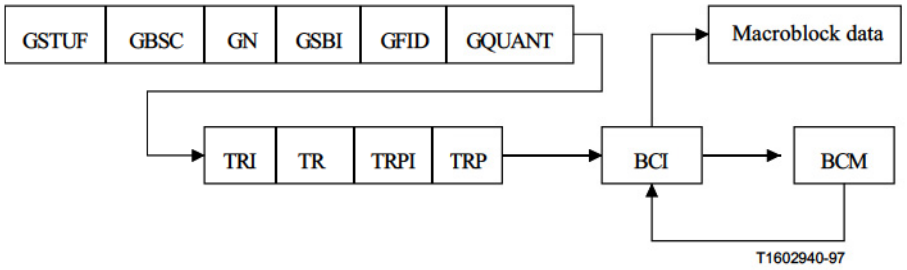


Figure N.2/H.263 – Structure of GOB layer for NEWPRED

When the optional Slice Structured mode (see Annex K) is in use, the syntax of the slice layer is modified in the same way as the GOB layer. The syntax is illustrated Figure N.3.

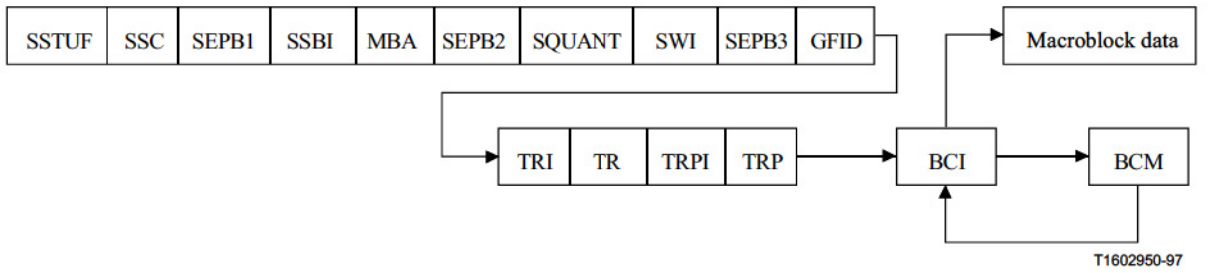


Figure N.3/H.263 – Structure of slice layer for NEWPRED

N.4.1.1 Temporal Reference Indicator (TRI) (1 bit)

TRI indicates whether or not the following TR field is present.

0: TR field is not present.

1: TR field is present.

N.4.1.2 Temporal Reference (TR) (8/10 bits)

When present, TR is an eight-bit number unless a custom picture clock frequency is in use, in which case it is a ten-bit number consisting of the concatenation of ETR and TR of the picture header.

N.4.1.3 Temporal Reference for Prediction Indicator (TRPI) (1 bit)

TRPI indicates whether or not the following TRP field is present.

0: TRP field is not present.

1: TRP field is present.

TRPI shall be equal to zero whenever the picture is an I- or EI-picture.

N.4.1.4 Temporal Reference for Prediction (TRP) (10 bits)

When present (as indicated in TRPI), TRP indicates the Temporal Reference which is used for prediction of the encoding, except in the case of B-pictures and the B-picture part of an Improved PB-frame. For B-pictures or the B-picture part of an Improved PB-frame, the picture having the temporal reference TRP is used for the prediction in the forward direction. (Prediction in the reverse-temporal direction always uses the immediately temporally subsequent picture.) TRP is a ten-bit number. If a custom picture clock frequency was not in use for the reference picture, the two MSBs of TRP are zero and the LSBs contain the eight-bit TR found in the picture header of the reference picture. If a custom picture clock frequency was in use for the reference picture, TRP is a ten-bit number consisting of the concatenation of ETR and TR from the reference picture header.

When TRP is not present, the most recent temporally previous anchor picture shall be used for prediction, as when not in the Reference Picture Selection mode. TRP is valid until the next PSC, GSC, or SSC.

N.4.1.5 Back-Channel message Indication (BCI) (Variable length)

This field contains one or two bits; when set to "1", it signals the presence of the following video Back-Channel Message (BCM) field. Otherwise, the field value is "01", which indicates the absence or the end of the video back-channel message field. Combinations of BCM and BCI may not be present, and may be repeated when present. BCI shall always be set to "01" if the videomux mode is not in use.

N.4.1.6 Back-Channel Message (BCM) (Variable length)

The Back-Channel Message having syntax as defined in N.4.2, which is present only if the preceding BCI field is set to "1".

N.4.2 Back-Channel Message (BCM) syntax

The syntax for the back-channel which conveys the acknowledgment/non-acknowledgment messages is illustrated in Figure N.4. This message is returned from a decoder to an encoder in order to tell whether a forward-channel data was correctly decoded or not.



Figure N.4/H.263 – Structure of Back-Channel Message (BCM) syntax for NEWPRED

N.4.2.1 Back-channel message Type (BT) (2 bits)

Back-channel message type indicates if the corresponding part of the encoded message is correctly decoded or not. Which type of message is required for the encoder is indicated in the picture header of the forward channel.

00: Reserved for future use.

01: Reserved for future use.

10: NACK. This indicates the erroneous decoding of the corresponding part of the forward-channel data.

11: ACK. This indicates the correct decoding of the corresponding part of the forward-channel data.

N.4.2.2 Unreliable Flag (URF) (1 bit)

The Unreliable Flag is set to 1 when a reliable value for TR or GN/MBA is not available to the decoder. (When BT is NACK, a reliable TR may not be available at the decoder.)

0: Reliable.

1: Unreliable.

N.4.2.3 Temporal Reference (TR) (10 bits)

Temporal reference contains the TR information of the video picture segment for which the ACK/NACK is indicated in the back-channel message.

NOTE – The meaning of the term "video picture segment" as used herein is defined in Annex R. If a custom picture clock frequency was not in use for the reference picture, the two MSBs of TR are zero and the LSBs contain the eight-bit TR found in the picture header of the reference picture. If a custom picture clock frequency was in use for the reference picture, TR is a ten-bit number consisting of the concatenation of ETR and TR from the reference picture header.

N.4.2.4 Enhancement Layer Number Indication (ELNUMI) (1 bit)

The enhancement layer number indication is "0" unless the optional Temporal, SNR and Spatial Scalability mode (Annex O) is used in the forward-channel data and some enhancement layers of the forward channel are combined on one logical channel and the back-channel message refers to an enhancement layer (rather than the base layer), in which case the enhancement layer number indication shall be "1".

N.4.2.5 Enhancement Layer Number (ELNUM) (4 bits)

Enhancement layer number is present if and only if ELNUMI is "1", in which case it contains the layer number of the enhancement layer referred to in the back-channel message.

N.4.2.6 BCPM (1 bit)

BCPM is "0" unless the CPM mode is used in the forward-channel data, in which case it is "1". If BCPM is "1", this indicates that BSBI is present.

N.4.2.7 Back-channel Sub-Bitstream Indicator (BSBI) (2 bits)

A fixed length codeword of 2 bits that is only present if BCPM is "1". The BSBI is the natural binary representation of the appropriate Sub-Bitstream number in the forward-channel data for which the ACK/NACK message is indicated in the back-channel message as described in 5.2.4 and in Annex C.

N.4.2.8 Back-channel Emulation Prevention Bit 1 (BEPB1) (1 bit)

A field which is present if and only if the videomux mode is in use. This field is always set to "1" to prevent a start code emulation.

N.4.2.9 GOB Number/Macroblock Address (GN/MBA) (5/6/7/9/11/12/13/14 bits)

A GOB number or macroblock address is present in this field. If the optional Slice Structured mode (see Annex K) is not in use, this field contains the GOB number of the beginning of the video picture segment for which the NACK/ACK message is indicated in the back-channel message. If the optional Slice Structured mode is in use, this field contains the macroblock address of the beginning of the slice for which the NACK/ACK message is indicated in the back-channel message. The length of this field is the length specified elsewhere in this Recommendation for GN or MBA.

N.4.2.10 Back-channel Emulation Prevention Bit 2 (BEPB2) (1 bit)

A field which is present if and only if the videomux mode is in use. This field is always set to "1" to prevent a start code emulation.

N.4.2.11 Requested Temporal Reference (RTR) (10 bits)

Requested temporal reference is present only if BT is NACK. RTR indicates the requested temporal reference of the GOB or slice associated with the NACK. Typically, it is the TR of the last correctly decoded video picture segment of the corresponding position at the decoder. If a custom picture clock frequency was not in use for the requested reference picture, the two MSBs of RTR are zero and the LSBs contain the eight-bit TR found in the picture header of the requested reference picture. If a custom picture clock frequency was in use for the requested reference picture, RTR is a ten-bit number consisting of the concatenation of ETR and TR from the requested reference picture header.

N.4.2.12 Stuffing (BSTUF) (Variable length)

This field is present if and only if the separate logical channel mode is in use and the back-channel message is the last in an external frame. BSTUF consists of a codeword of variable length consisting of zero or more bits of value "0". This field is only present at the end of an external frame.

N.5 Decoder process

The decoder of this mode may need an additional number of picture memories to store the correctly decoded video signals and their Temporal Reference (TR) information. The decoder uses the stored picture for which the TR is TRP as the reference picture for inter-frame decoding instead of the last decoded picture, if the TRP field exists in the forward-channel data. When the picture for which the

TR is TRP is not available at the decoder, the decoder may send a forced INTRA update signal to the encoder by external means (for example, Recommendation H.245). Unless a different frame storage policy is negotiated by external means, correctly decoded video picture segments shall be stored into memory for use as later reference pictures on a first-in, first-out basis as shown in Figure N.1 (except for B-pictures, which are not used as reference pictures), and video picture segments which are detected as having been incorrectly decoded should not replace correctly decoded ones in this memory area.

An Acknowledgment message (ACK) and a Non-Acknowledgment message (NACK) are defined as back-channel messages. An ACK may be returned when the decoder decodes a video picture segment successfully. NACKs may be returned when the decoder fails to decode a video picture segment, and may continue to be returned until the decoder gets the expected forward channel data which includes the requested TRP or an INTRA update. Which types of message shall be sent is indicated in the RPSMF field of the picture header of the forward-channel data.

In a usage scenario known as "Video Redundancy Coding", the Reference Picture Selection mode may be used by some encoders in a manner in which more than one representation is sent for the pictured scene at the same temporal instant (usually using different reference pictures). In such a case in which the Reference Picture Selection mode is in use and in which adjacent pictures in the bitstream have the same temporal reference, the decoder shall regard this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and shall decode and use the first such received picture while discarding the subsequent redundant picture(s).

ANNEX O

Temporal, SNR, and Spatial Scalability mode

This annex describes the optional mode of this Recommendation in support of Temporal, SNR, and Spatial Scalability. This mode may also be used in conjunction with error control schemes. The capability of this mode and the extent to which its features are supported is signaled by external means (for example, Recommendation H.245). The use of this mode is indicated in PLUSPTYPE.

O.1 Overview

Scalability allows for the decoding of a sequence at more than one quality level. This is done by using a hierarchy of pictures and enhancement pictures partitioned into one or more layers. There are three types of pictures used for scalability: B-, EI-, and EP-pictures, as explained below. Each of these has an enhancement layer number ELNUM that indicates to which layer it belongs, and a reference layer number RLNUM that indicates which layer is used for its prediction. The lowest layer is called the base layer, and has layer number 1.

Scalability is achieved by three basic methods: temporal, SNR, and spatial enhancement.

O.1.1 Temporal scalability

Temporal scalability is achieved using bidirectionally predicted pictures, or B-pictures. B-pictures allow prediction from either or both a previous and subsequent reconstructed picture in the reference layer. This property generally results in improved compression efficiency as compared to that of P-pictures. These B-pictures differ from the B-picture part of a PB- (or Improved PB-) frame (see Annexes G and M) in that they are separate entities in the bitstream: they are not syntactically intermixed with a subsequent P- (or EP-) picture.

B-pictures (and the B-part of PB- or Improved PB-frames) are not used as reference pictures for the prediction of any other pictures. This property allows for B-pictures to be discarded if necessary without adversely affecting any subsequent pictures, thus providing temporal scalability. Figure O.1 illustrates the predictive structure of P- and B-pictures.

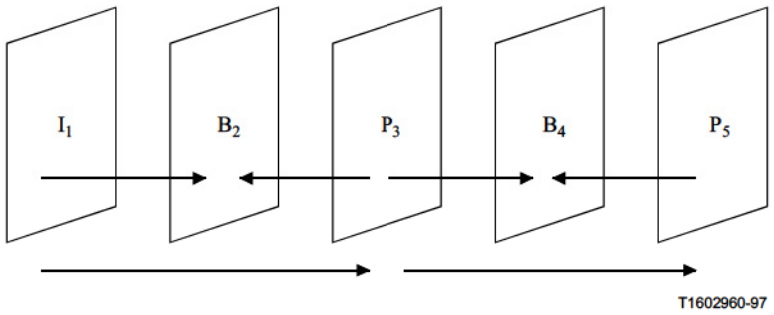


Figure O.1/H.263 – Illustration of B-picture prediction dependencies

The location of B-pictures in the bitstream is in a data-dependence order rather than in strict temporal order. (This rule is consistent with the ordering of other pictures in the bitstream, but for all picture types other than the B-picture, no such conflict arises between the data-dependence order and the temporal order.) For example, if the pictures of a video sequence were numbered 1, 2, 3, …, then the bitstream order of the encoded pictures would be I_1, P_3, B_2, P_5, B_4, …, where the subscript refers to the original picture number (as illustrated in Figure O.1).

There is no limit to the number of B-pictures that may be inserted between pairs of reference pictures in the reference layer (other than what is necessary to prevent temporal ambiguity from overflows of the temporal reference field in the picture header). However, a 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 layer picture.

Motion vectors are allowed to extend beyond the picture boundaries of B-pictures.

O.1.2 SNR scalability

The other basic method to achieve scalability is through spatial/SNR enhancement. Spatial scalability and SNR scalability are equivalent except for the use of interpolation as is described shortly. Because compression introduces artifacts and distortions, the difference between a reconstructed picture and its original in the encoder is (nearly always) a nonzero-valued picture, containing what can be called the coding error. Normally, this coding error is lost at the encoder and never recovered. With SNR scalability, these coding error pictures can also be encoded and sent to the decoder, producing an enhancement to the decoded picture. The extra data serves to increase the signal-to-noise ratio of the video picture, and hence, the term SNR scalability. Figure O.2 illustrates the data flow for SNR scalability. The vertical arrows from the lower layer illustrate that the picture in the enhancement layer is predicted from a reconstructed approximation of that picture in the reference (lower) layer.

If prediction is only formed from the lower layer, then the enhancement layer picture is referred to as an EI-picture. It is possible, however, to create a modified bidirectionally predicted picture using both a prior enhancement layer picture and a temporally simultaneous lower layer reference picture. This type of picture is referred to as an EP-picture or "Enhancement" P-picture. The prediction flow for EI- and EP-pictures is shown in Figure O.2. (Although not specifically shown in Figure O.2, an

EI-picture in an enhancement layer may have a P-picture as its lower layer reference picture, and an EP-picture may have an I-picture as its lower-layer enhancement picture.)

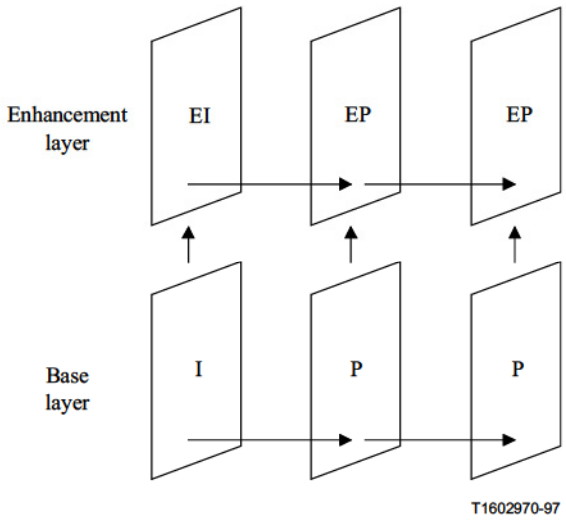


Figure O.2/H.263 – Illustration of SNR scalability

For both EI- and EP-pictures, the prediction from the reference layer uses no motion vectors. However, as with normal P-pictures, EP-pictures use motion vectors when predicting from their temporally prior reference picture in the same layer.

O.1.3 Spatial scalability

The third and final scalability method in the Temporal, SNR, and Spatial Scalability mode is spatial scalability, which is closely related to SNR scalability. The only difference is that before the picture in the reference layer is used to predict the picture in the spatial enhancement layer, it is interpolated by a factor of two either horizontally or vertically (1-D spatial scalability), or both horizontally and vertically (2-D spatial scalability). The interpolation filters for this operation are defined in O.6. For a decoder to be capable of some forms of spatial scalability, it may also need to be capable of custom picture formats. For example, if the base layer is sub-QCIF (128 × 96), the 2-D spatial enhancement layer picture would be 256 × 192, which does not correspond to a standard picture format. Another example would be if the base layer were QCIF (176 × 144), with the standard pixel aspect ratio of 12:11. A 1-D horizontal spatial enhancement layer would then correspond to a picture format of 352 × 144 with a pixel aspect ratio of 6:11. Thus a custom picture format would have to be used for the enhancement layer in these cases. An example which does not require a custom picture format would be the use of a QCIF base layer with a CIF 2-D spatial enhancement layer. Spatial scalability is illustrated in Figure O.3.

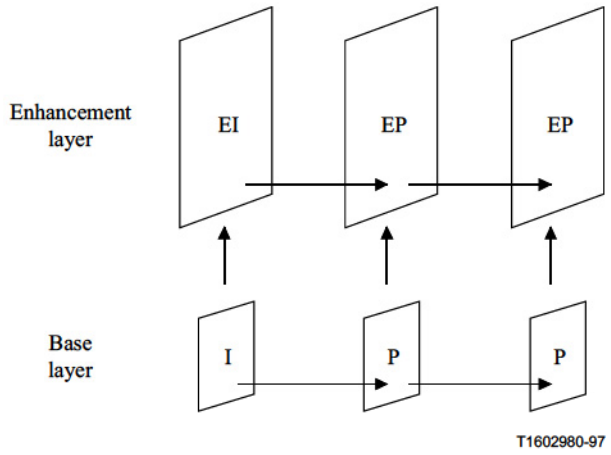


Figure O.3/H.263 – Illustration of spatial scalability

Other than requiring an upsampling process to increase the size of the reference layer picture prior to its use as a reference for the encoding process, the processing and syntax for a spatial scalability picture is functionally identical to that for an SNR scalability picture.

Since there is very little syntactical distinction between pictures using SNR scalability and pictures using spatial scalability, the pictures used for either purpose are called EI- and EP-pictures.

The picture in the base layer which is used for upward prediction in an EI- or EP-picture may be an I-picture, a P-picture, or the P-part of a PB- or Improved PB-frame (but shall not be a B-picture or the B-part of a PB- or Improved PB-frame).

O.1.4 Multilayer scalability

It is possible not only for B-pictures to be temporally inserted between pictures of types I, P, PB, and Improved PB, but also between pictures of types EI and EP, whether these consist of SNR or spatial enhancement pictures. It is also possible to have more than one SNR or spatial enhancement layer in conjunction with a base layer. Thus, a multilayer scalable bitstream can be a combination of SNR layers, spatial layers, and B-pictures. The size of a picture cannot decrease, however, with increasing layer number. It can only stay the same or increase by a factor of two in one or both dimensions. Figure O.4 illustrates a multilayer scalable bitstream.

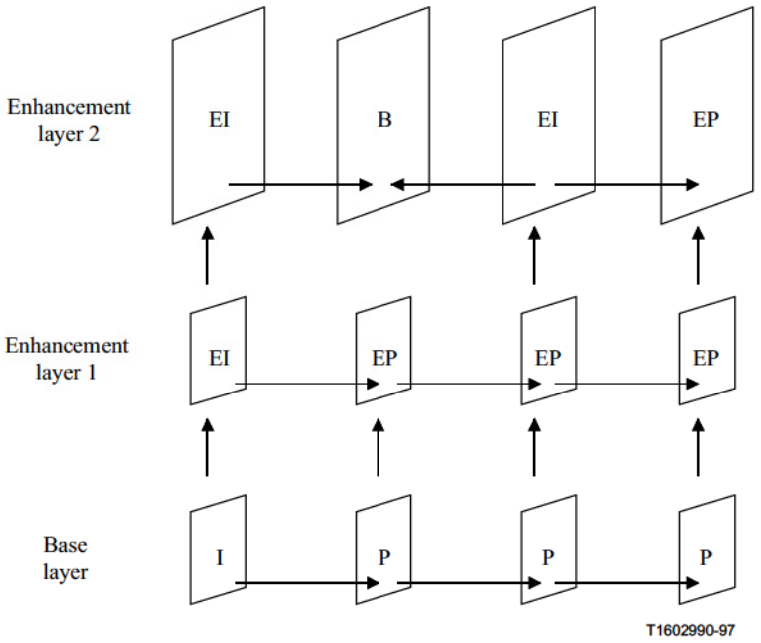


Figure O.4/H.263 – Illustration of multilayer scalability

In the case of multilayer scalability, the picture in a reference layer which is used for upward prediction in an EI- or EP-picture may be a I-, P-, EI-, or EP-picture, or may be the P-part of a PB- or Improved PB-frame in the base layer (but shall not be a B-picture or the B-part of a PB- or Improved PB-frame).

As with the two-layer case, B-pictures may occur in any layer. However, any picture in an enhancement layer which is temporally simultaneous with a B-picture in its reference layer must be a B-picture or the B-picture part of an PB- or Improved PB-frame. This is to preserve the disposable nature of B-pictures. Note, however, that B-pictures may occur in layers that have no corresponding picture in lower layers. This allows an encoder to send enhancement video with a higher picture rate than the lower layers.

The enhancement layer number and the reference layer number for each enhancement picture (B-, EI-, or EP-) are indicated in the ELNUM and RLNUM fields, respectively, of the picture header (when present). See the inference rules described in 5.1.4.4 for when these fields are not present. If a B-picture appears in an enhancement layer in which temporally surrounding SNR or spatial scalability pictures also appear, the Reference Layer Number (RLNUM) of the B-picture shall be the same as the Enhancement Layer Number (ELNUM).

The picture height, width, and pixel aspect ratio of a B-picture shall always be equal to those of its temporally subsequent reference layer picture.

O.2 Transmission order of pictures

Pictures which are dependent on other pictures shall be located in the bitstream after the pictures on which they depend.

The bitstream syntax order is specified such that for reference pictures (i.e. picture having types I, P, EI, or EP, or the P-part of PB or Improved PB), the following two rules shall be obeyed:

1) All reference pictures with the same temporal reference shall appear in the bitstream in increasing enhancement layer order (since each lower layer reference picture is needed to decode the next higher layer reference picture).

2) All temporally simultaneous reference pictures as discussed in item 1) above shall appear in the bitstream prior to any B-pictures for which any of these reference pictures is the first temporally subsequent reference picture in the reference layer of the B-picture (in order to reduce the delay of decoding all reference pictures which may be needed as references for B-pictures).

Then, the B-pictures with earlier temporal references shall follow (temporally ordered within each enhancement layer).

The bitstream location of each B-picture shall comply with the following rules:

1) Its bitstream location shall be after that of its first temporally subsequent reference picture in the reference layer (since the decoding of the B-picture generally depends on the prior decoding of that reference picture).

2) Its bitstream location shall be after that of all reference pictures that are temporally simultaneous with the first temporally subsequent reference picture in the reference layer (in order to reduce the delay of decoding all reference pictures which may be needed as references for B-pictures).

3) Its bitstream location shall precede the location of any additional temporally subsequent pictures other than B-pictures in its reference layer (since to allow otherwise would increase picture-storage memory requirements for the reference layer pictures).

4) Its bitstream location shall be after that of all EI- and EP-pictures that are temporally simultaneous with the first temporally subsequent reference picture.

5) Its bitstream location shall precede the location of all temporally subsequent pictures within its same enhancement layer (since to allow otherwise would introduce needless delay and increase picture-storage memory requirements for the enhancement layer).

Figure O.5 illustrates two allowable picture transmission orders given by the rules above for the layering structure shown therein (with numbers in dotted-line boxes indicating the bitstream order, separated by commas for the two alternatives).

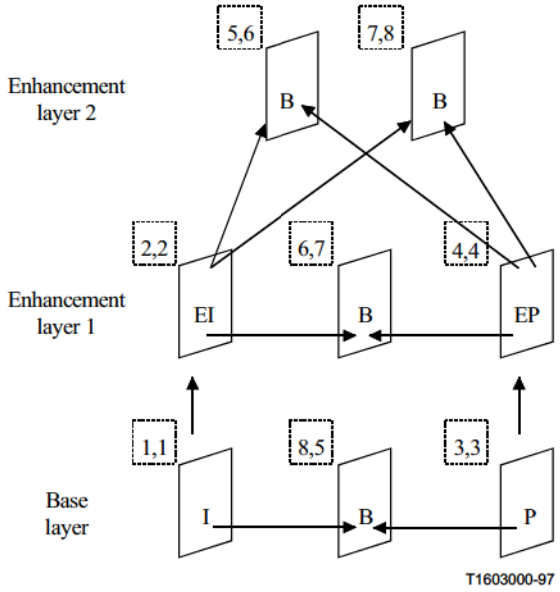


Figure O.5/H.263 – Example of picture transmission order

O.3 Picture layer syntax

The Enhancement Layer Number (ELNUM) (see 5.1.11) is always present in any enhancement (B, EI, or EP) picture and shall not be present in I- or P-pictures, or in PB- or Improved PB-frames. The Reference Layer Number (RLNUM) (see 5.1.12) is present for some enhancement pictures and is inferred for others, as described in 5.1.12.

There is exactly one base layer, and it has ELNUM and RLNUM equal to 1. The RLNUM gives the enhancement layer number of the forward and backward reference pictures for B-pictures, for the upward reference picture for EI- and EP-pictures. The reference pictures of the base layer may consist of I, PB, Improved PB, and P-pictures, none of which include ELNUM or RLNUM in the picture header (their implied values are 1).

For B-pictures, RLNUM must be less or equal to ELNUM, while for EI- and EP-pictures, RLNUM must be smaller than ELNUM.

ELNUM may be different from the layer number used at the system level. Since B-pictures have no other pictures dependent on themselves, they may even be put in a separate enhancement layer by system components external to this Recommendation (for example, Recommendations H.245 and H.223). Moreover, it is up to the implementor as to whether the enhancement pictures are sent in separate video channels or remain multiplexed together with the base layer pictures.

As stated in 5.1.4.5, the Deblocking Filter mode (see Annex J) does not apply within B-pictures. This is because B-pictures are not used for the prediction of any other pictures, and thus the application of a deblocking filter to these pictures is entirely a post-processing technique which is outside the scope of this Recommendation. However, the use of some type of deblocking filter for B-pictures is encouraged, and a filter such as that described in Annex J may in fact be a good design to use for that purpose.

The Temporal Reference (TR) (see 5.1.2) is defined exactly as for I- and P-pictures.

There shall be no TR_B (see 5.1.22) or DBQUANT (see 5.1.23) fields in the picture header of B-, EI- or EP-pictures.

O.4 Macroblock layer syntax

The macroblock layer syntax for B- and EP-pictures is the same, since each uses two reference pictures in a similar way. However, the interpretation varies slightly depending on the picture type. Figure O.6 indicates the B and EP syntax. The MBTYPE field indicates whether there is direct-mode prediction, forward prediction, backward/upward prediction, or bidirectional prediction. The MBTYPE is defined differently for B- and EP-pictures as described below.

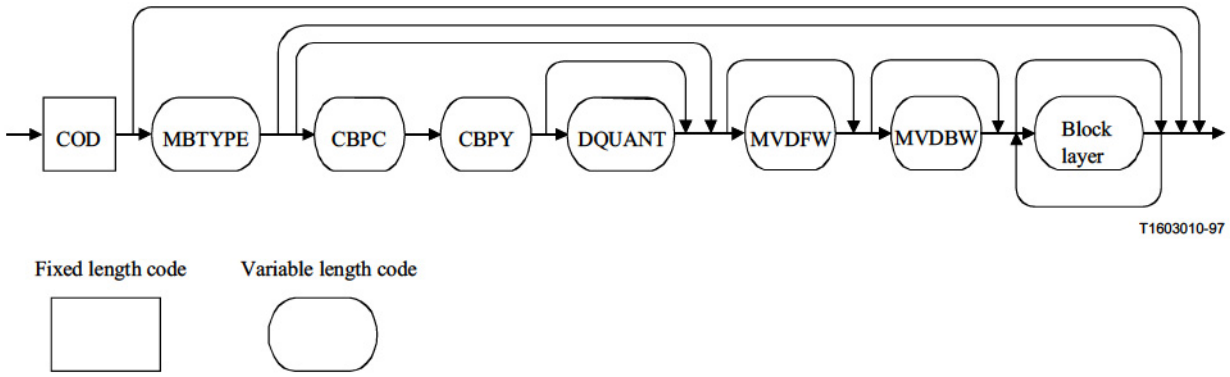


Figure O.6/H.263 – Macroblock syntax for EP- and B-pictures

The direct prediction mode is only available for B-pictures. It is a bidirectional prediction mode similar to the bidirectional mode in Improved PB-frames mode (Annex M). The only difference is that there is no restriction on which pixels can be backward predicted since the complete backward prediction image is known at the decoder. Bidirectional mode uses separate motion vectors for forward and backward prediction. In both direct and bidirectional mode, the prediction pixel values are calculated by averaging the forward and backward prediction pixels. The average is calculated by dividing the sum of the two predictions by two (division by truncation). In direct mode, when there are four motion vectors in the reference macroblock, then all four motion vectors are used as in Improved PB-frames mode (Annex M).

For B-pictures, forward prediction means prediction from a previous reference picture in the reference layer. Backward prediction means prediction from a temporally subsequent reference picture in the reference layer.

For EP-pictures, forward prediction means prediction from a previous EI- or EP-picture in the same layer, while upward prediction is used to mean prediction from the temporally simultaneous (possibly interpolated) reference picture in the reference layer. No motion vector is used for the upward prediction (which is syntactically in the same place as backward prediction for B-pictures), although one can be used for the forward prediction.

The macroblock syntax for EI-pictures is slightly different. As shown in Figure O.7, the MBTYPE and CBPC are combined into an MCBPC field. No forward prediction is used, only upward prediction from the temporally simultaneous reference picture in the reference layer. No motion vector is used.

In B- and EP-pictures, motion vectors over picture boundaries may be used as described in D.1 (although extension of the motion vector range as described in D.2 is only active if the Unrestricted Motion Vector mode is also in use).

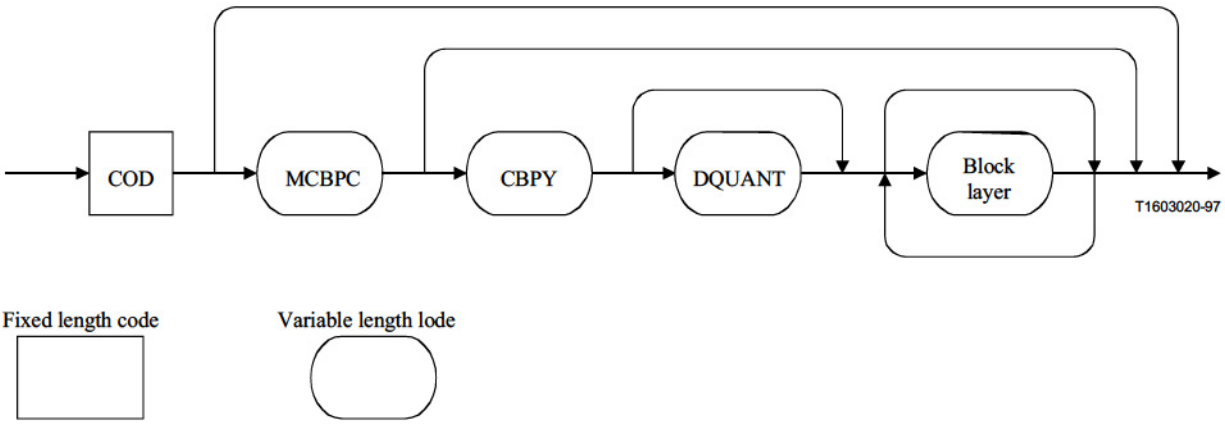


Figure O.7/H.263 – Macroblock syntax for EI-pictures

O.4.1 Coded macroblock indication (COD) (1 bit)

A bit which, when set to "0", signals that the macroblock is coded. If set to "1", no further information is transmitted for this macroblock and the macroblock is handled as "skipped" as described below.

O.4.2 MBTYPE/MCBPC (VLC)

There are different MBTYPE tables for B- and EP-pictures. For EI-pictures, there is instead a MCBPC table. Table O.1 is the MBTYPE table for B-pictures. Table O.2 is the MBTYPE table for EP-pictures. Table O.3 is the MCBPC table for EI-pictures.

For B-pictures, the "Direct (skipped)" prediction type indicates that neither MBTYPE, nor any data, is transmitted in the macroblock and that the decoder derives forward and backward motion vectors and the corresponding bidirectional prediction. This is signaled by the COD bit. The "Forward (no texture)", the "Backward (no texture)", and the "Bi-dir (no texture)" prediction types for a B-picture indicate forward, backward, and bi-directional prediction with no coefficients, and with one transmitted motion vector for forward and backward prediction, and two transmitted motion vectors for bidirectional prediction.

For EP-pictures, the "Forward (skipped)" prediction type indicates that no additional data is sent for the macroblock, so that the decoder should use forward prediction with a zero motion vector and no coefficients. The "Upward (no texture)" and "Bi-dir (no texture)" prediction types for an EP-picture indicate upward and bidirectional prediction with no coefficients, and with zero motion vector(s).

For EI-pictures, the "Upward (skipped)" prediction type indicates that no additional data is sent for the macroblock, so that the decoder should use upward prediction with a zero motion vector and no coefficients.

Table O.1/H.263 – MBTYPE VLC codes for B-pictures

Index	Prediction type	MVDFW	MVDBW	CBPC + CBPY	DQUANT	MBTYPE	Bits
–	Direct (skipped)					(COD = 1)	0
0	Direct			X		11	2
1	Direct + Q			X	X	0001	4
2	Forward (no texture)	X				100	3
3	Forward	X		X		101	3
4	Forward + Q	X		X	X	0011 0	5
5	Backward (no texture)		X			010	3
6	Backward		X	X		011	3
7	Backward + Q		X	X	X	0011 1	5
8	Bi-dir (no texture)	X	X			0010 0	5
9	Bi-dir	X	X	X		0010 1	5
10	Bi-dir + Q	X	X	X	X	0000 1	5
11	INTRA			X		0000 01	6
12	INTRA + Q			X	X	0000 001	7
13	Stuffing					0000 0000 1	9

Table O.2/H.263 – MBTYPE VLC codes for EP-pictures

Index	Prediction type	MVDFW	MVDBW	CBPC + CBPY	DQUANT	MBTYPE	Bits
–	Forward (skipped)					(COD = 1)	0
0	Forward	X		X		1	1
1	Forward + Q	X		X	X	001	3
2	Upward (no texture)					010	3
3	Upward			X		011	3
4	Upward + Q			X	X	0000 1	5
5	Bi-dir (no texture)					0001 0	5
6	Bi-dir	X		X		0001 1	5
7	Bi-dir + Q	X		X	X	0000 01	6
8	INTRA			X		0000 001	7
9	INTRA + Q			X	X	0000 0001	8
10	Stuffing					0000 0000 1	9

Table O.3/H.263 – MCBPC VLC codes for EI-pictures

Index	Prediction type	Code block pattern (56)	CBPY	DQUANT	MCBPC	Bits
–	Upward (skipped)				(COD = 1)	0
0	Upward	00	X		1	1
1	Upward	01	X		001	3
2	Upward	10	X		010	3
3	Upward	11	X		011	3
4	Upward + Q	00	X	X	0001	4
5	Upward + Q	01	X	X	0000 001	7
6	Upward + Q	10	X	X	0000 010	7
7	Upward + Q	11	X	X	0000 011	7
8	INTRA	00	X		0000 0001	8
9	INTRA	01	X		0000 1001	8
10	INTRA	10	X		0000 1010	8
11	INTRA	11	X		0000 1011	8
12	INTRA+Q	00	X	X	0000 1100	8
13	INTRA+Q	01	X	X	0000 1101	8
14	INTRA+Q	10	X	X	0000 1110	8
15	INTRA+Q	11	X	X	0000 1111	8
16	Stuffing				0000 0000 1	9

O.4.3 Coded Block Pattern for Chrominance (CBPC) (Variable length)

When present, CBPC indicates the coded block pattern for chrominance blocks as described in Table O.4. CBPC is present only for EP- and B-pictures, when its presence is indicated by MBTYPE (see Tables O.1 and O.2).

Table O.4/H.263 – CBPC VLC codes

Index	Code block pattern (56)	CBPC	Bits
0	00	0	1
1	01	10	2
2	10	111	3
3	11	110	3

O.4.4 Coded Block Pattern for Luminance (CBPY) (Variable length)

When present, CBPY indicates which blocks in the luminance portion of the macroblock are present. CBPY is present only when its presence is indicated by MBTYPE (see Tables O.1, O.2 and O.3). CBPY is coded as described in 5.3.5 and in Table 13. Upward predicted macroblocks in EI- and EP-pictures, bidirectional predicted macroblocks in EP-pictures, and INTRA macroblocks in EI-, EP-, and B-pictures use the CBPY definition for INTRA macroblocks, and other macroblock types in EI-, EP-, and B-pictures use the CBPY definition for INTER macroblocks.

O.4.5 Quantizer Information (DQUANT) (2 bits/Variable length)

DQUANT is used as in other picture macroblock types. See 5.3.6 and Annex T.

O.4.6 Motion vector data (MVDFW, MVDBW) (Variable length)

MVDFW is the motion vector data for the forward vector, if present. MVDBW is the motion vector data for the backward vector, if present (allowed only in B-pictures). The variable length codewords are given in Table 14, or in Table D.3 if the Unrestricted Motion Vector mode is used (see Annex D).

O.5 Motion vector decoding

O.5.1 Differential motion vectors

Motion vectors for forward, backward or bidirectionally predicted blocks are differentially encoded. To recover the macroblock motion vectors, a prediction is added to the motion vector differences. The predictions are formed in a similar manner to that described in 6.1.1, except 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. The same decision rules apply for the special cases at picture, GOB or slice borders as described in 6.1.1. If a neighboring macroblock does not have a motion vector of the same type (forward or backward), the candidate predictor for that macroblock is zero for that motion vector type.

O.5.2 Motion vectors in direct mode

For macroblocks coded in direct mode, no vector differences are transmitted. Instead, the forward and backward motion vectors are directly computed from the temporally consecutive P-vector as described in G.4 with the restriction that MV_D is always zero. These derived vectors are not used for prediction of other motion vectors.

O.6 Interpolation filters

The method by which a picture is interpolated for 2-D spatial scalability is shown in Figures O.8 and O.9. The first figure shows the interpolation for interior pixels, while the second one shows the interpolation close to picture boundaries. This is the same technique used in Annex Q and in some cases in Annex P. The method by which a picture is interpolated for 1-D spatial scalability is shown in Figures O.10 and O.11. Figure O.10 shows the interpolation for interior pixels in the horizontal direction. The interpolation for the vertical direction is analogous. Figure O.11 shows the interpolation for pixels at picture boundaries. Again, the interpolation for the vertical direction is analogous. Again, this is the same technique used in some cases in Annex P.

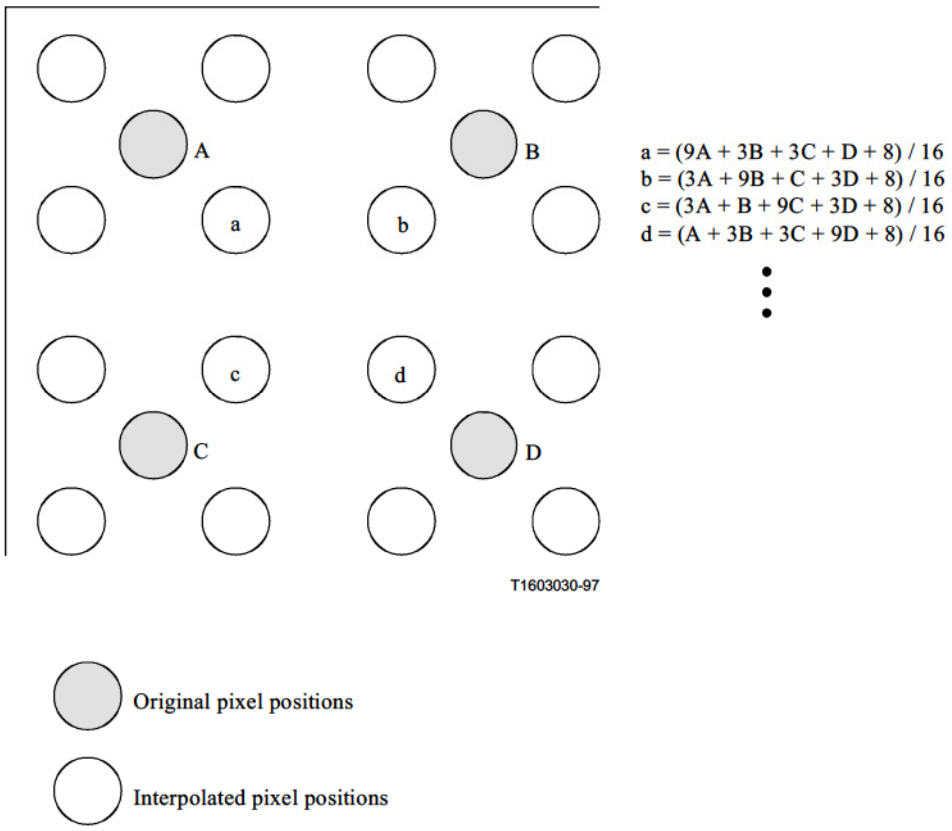


Figure O.8/H.263 – Method for interpolating pixels for 2-D scalability

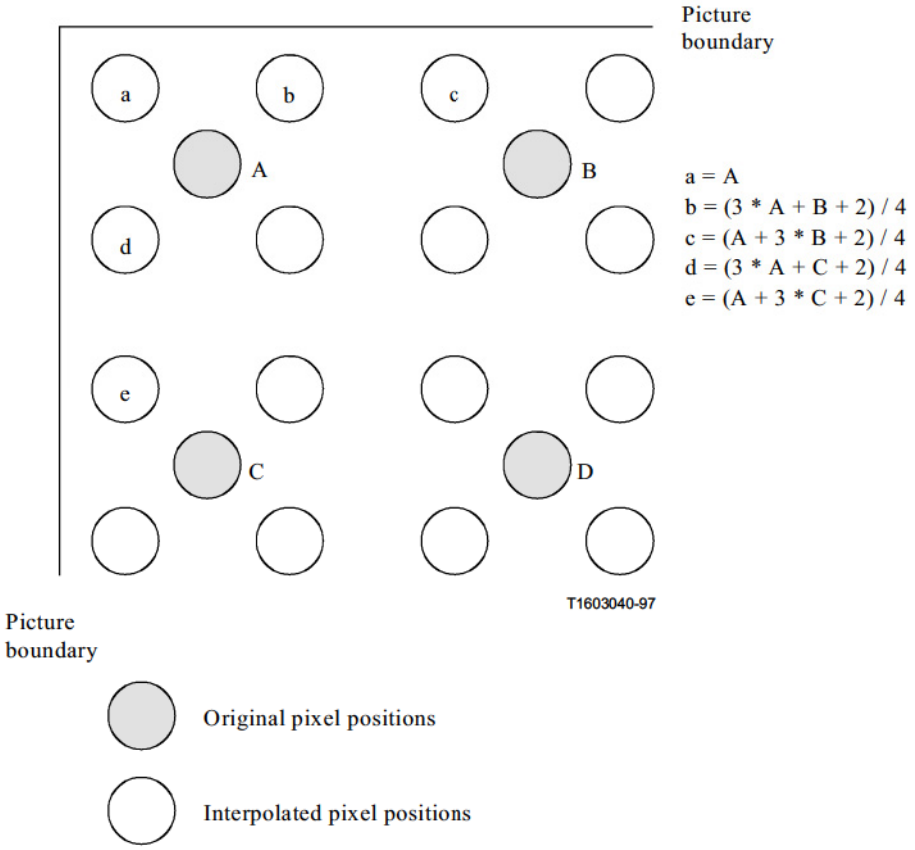


Figure O.9/H.263 – Method for 2-D interpolation at boundaries

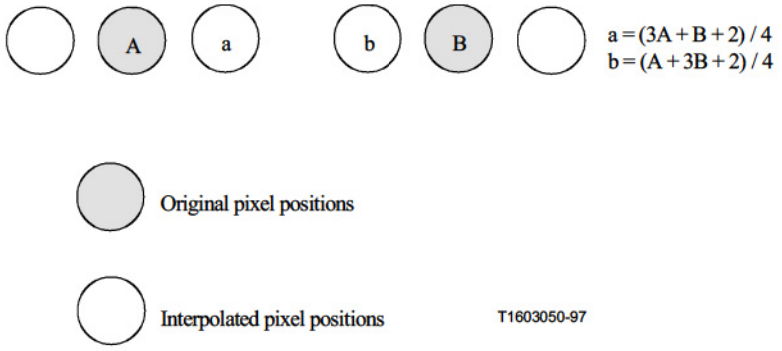


Figure O.10/H.263 – Method for interpolating pixels for 1-D scalability

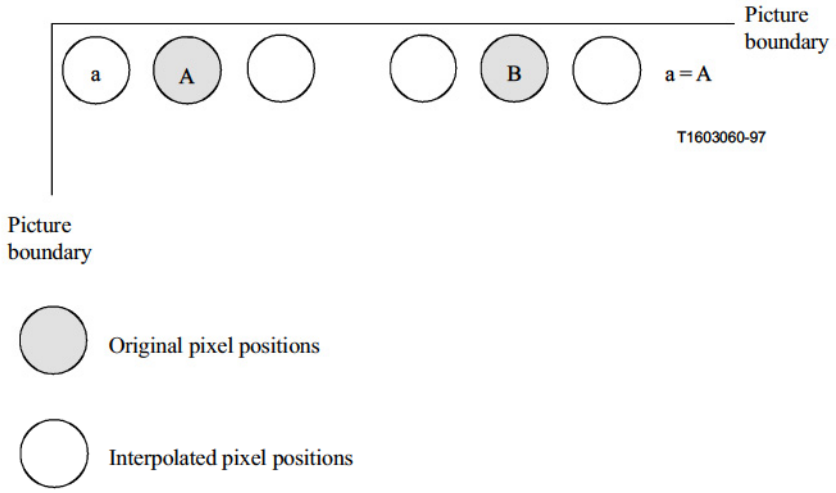


Figure O.11/H.263 – Method for 1-D interpolation at picture boundaries

ANNEX P

Reference picture resampling

P.1 Introduction

This annex describes the use and syntax of a resampling process which can be applied to the previous decoded reference picture in order to generate a "warped" picture for use in predicting the current picture. This resampling syntax can specify the relationship of the current picture to a prior picture having a different source format, and can also specify a "global motion" warping alteration of the shape, size, and location of the prior picture with respect to the current picture. In particular, the Reference Picture Resampling mode can be used to adaptively alter the resolution of pictures during encoding. A fast algorithm is used to generate bilinear interpolation coefficients. The capability to use this mode and the extent to which its features are supported is negotiated externally (for example, Recommendation H.245). This mode may be used in restricted scenarios that may be defined during capability negotiation (e.g. to support only factor-of-four picture resizing, to support only half-pixel resolution picture warping, or to support arbitrary image size resizing and displacements).

NOTE – The default image transformation between pictures of differing resolutions is defined to maintain the spatial alignment of the edges of the picture area and of the relative locations of the luminance and chrominance samples. This may have implications on the design of any resampling operations used for generating pictures of various resolutions at the encoder and for displaying pictures of various resolutions after decoding (particularly regarding shifts in spatial location caused by phase shifts induced in resampling). Also, since this mode can be used for dynamic adaptive picture resolution changes, operation with this mode may benefit from external negotiation to display the decoded picture at a higher resolution than its encoded picture size, in order to allow switching between encoded picture sizes without resizing the picture display.

If the Reference Picture Resampling bit in the PLUSPTYPE field is not set, but PLUSPTYPE is present and the picture is an INTER, B, or EP-picture or an Improved PB-frame, and the picture size differs from that of the temporally previous encoded picture, this condition invokes Reference Picture Resampling with warping parameters (see P.2.2) set equal to zero, fill mode (see P.2.3) set to *clip*, and displacement accuracy (see P.2.1) set to $\frac{1}{16}$-pixel accuracy. This causes the resampling process to easily act as a predictively-encoded change in picture resolution. In simple factor-of-four

resolution change cases such as transitions between CIF and 4CIF, the resampling process reduces to the same simple filter used for Spatial Scalability (Annex O) or for Reduced-Resolution Update (Annex Q), except for the application of rounding control.

If the picture is an EP-picture and the Reference Picture Resampling bit is set in the PLUSPTYPE field of the picture header of the reference layer, this bit shall also be set in the picture header of the EP-picture in the enhancement layer.

If a B-picture uses the Reference Picture Resampling mode, the resampling process shall be applied to the temporally previous anchor picture and not to the temporally subsequent one. The temporally previous anchor picture to which the resampling process is applied shall be the decoded picture (i.e. prior to any resampling applied by Reference Picture Resampling if this mode is also invoked for the subsequent reference picture). The temporally subsequent anchor picture shall have the same picture size as the B-picture.

If the Reference Picture Resampling mode is invoked for an Improved PB-frame, one set of warping parameters is sent and the resampled reference picture is used as the reference for both the B- and P-parts of the Improved PB-frame.

The Reference Picture Resampling is defined in terms of the displacement of the four corners of the current picture area. For the luminance field of the current picture with horizontal size H and vertical size V, four conceptual motion vectors $\underline{v}^{00}$, $\underline{v}^{H0}$, $\underline{v}^{0V}$, and $\underline{v}^{HV}$, are defined for the upper left, upper right, lower left, and lower right corners of the picture, respectively. These vectors describe how to move the corners of the current picture to map them onto the corresponding corners of the previous decoded picture, as shown in Figure P.1. The units of these vectors are the same as those in the reference picture grid. To generate a vector $\underline{v}(x,y)$ at some real-valued location (x, y) in the interior of the current picture, an approximation to bilinear interpolation is used, *i.e.* as in:

$$\underline{v}(x,y)=\left(1-\frac{y}{V}\right)\left[\left(1-\frac{x}{H}\right)\underline{v}^{00}+\left(\frac{x}{H}\right)\underline{v}^{H0}\right]+\left(\frac{y}{V}\right)\left[\left(1-\frac{x}{H}\right)\underline{v}^{0V}+\left(\frac{x}{H}\right)\underline{v}^{HV}\right]$$

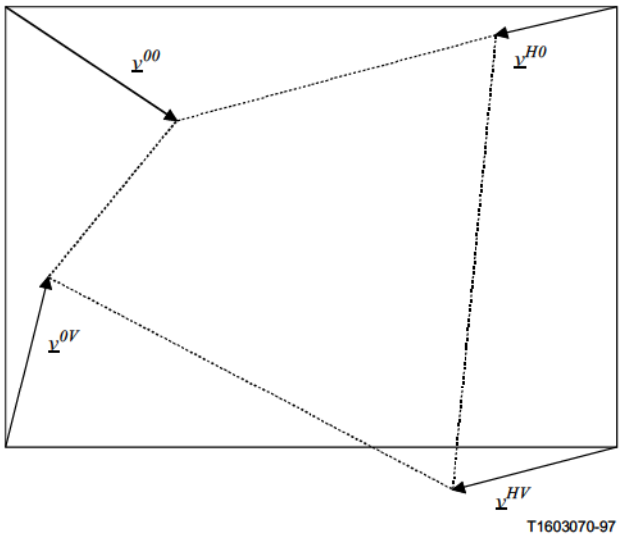


Figure P.1/H.263 – Example of conceptual motion vectors used for warping

The horizontal size H and vertical size V of the current picture, and the horizontal size H_R and vertical size V_R of the reference picture are those indicated by the picture header, regardless of

whether these values are divisible by 16 or not. If the picture width or height is not divisible by 16, the additional area shall be generated by adding pixels to the resampled picture using the same fill mode as used in the resampling process.

For the simplicity of description, the resampling vectors $\underline{r}^0$, $\underline{r}^x$, $\underline{r}^y$ and $\underline{r}^{xy}$ are defined as:

$$\underline{r}^0 = \underline{v}^{00}$$

$$\underline{r}^x = \underline{v}^{H0} - \underline{v}^{00}$$

$$\underline{r}^y = \underline{v}^{0V} - \underline{v}^{00}$$

$$\underline{r}^{xy} = \underline{v}^{00} - \underline{v}^{H0} - \underline{v}^{0V} + \underline{v}^{HV}$$

Using this definition, the equation for bilinear interpolation is rewritten as:

$$\underline{v}(x,y) = \underline{r}^0 + \left(\frac{x}{H}\right)\underline{r}^x + \left(\frac{y}{V}\right)\underline{r}^y + \left(\frac{x}{H}\right)\left(\frac{y}{V}\right)\underline{r}^{xy}$$

For warping purposes, it is assumed that the coordinates of the upper left corner of the picture area are $(x, y) = (0,0)$ and that each pixel has unit height and width, so that the centers of the pixels lie at the points $(x,y) = \left(i_L + \frac{1}{2}, j_L + \frac{1}{2}\right)$ for $i_L = 0, \ldots, H-1$ and $j_L = 0, \ldots, V-1$, where the L subscript indicates that i_L and j_L pertain to the luminance field. (As the pixel aspect ratio is usually constant or the conversion of the aspect ratio is performed by this resampling, there is no need to consider the actual pixel aspect ratio for these purposes.) Using this convention, the x- and y-displacements at the locations of interest in the luminance field of the reference picture are:

$$v_x(i_L, j_L) = \frac{1}{HV}\left[HVr_x^0 + \left(i_L + \frac{1}{2}\right)Vr_x^x + \left(j_L + \frac{1}{2}\right)Hr_x^y + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_x^{xy}\right]$$

$$v_y(i_L, j_L) = \frac{1}{HV}\left[HVr_y^0 + \left(i_L + \frac{1}{2}\right)Vr_y^x + \left(j_L + \frac{1}{2}\right)Hr_y^y + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_y^{xy}\right]$$

Because all positions and phases must be computed relative to the center of the upper-left corner pixel, which has the coordinate $(x, y) = \left(\frac{1}{2}, \frac{1}{2}\right)$, the quantities of primary interest are:

$$x_R(i_L, j_L) - \frac{1}{2} = \left(i_L + \frac{1}{2}\right) + v_x(i_L, j_L) - \frac{1}{2}$$

$$= \frac{1}{HV}\left[HVr_x^0 + \left(i_L + \frac{1}{2}\right)(HV + Vr_x^x) + \left(j_L + \frac{1}{2}\right)Hr_x^y + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_x^{xy}\right] - \frac{1}{2}$$

$$y_R(i_L, j_L) - \frac{1}{2} = \left(i_L + \frac{1}{2}\right) + v_y(i_L, j_L) - \frac{1}{2}$$

$$= \frac{1}{HV}\left[HVr_y^0 + \left(j_L + \frac{1}{2}\right)Vr_y^x + \left(j_L + \frac{1}{2}\right)(HV + Hr_y^y) + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_y^{xy}\right] - \frac{1}{2}$$

Once a location has been determined in the prior decoded reference picture using an approximation of these equations, bilinear interpolation as specified later in this annex shall be used to generate a value for the resampled pixel.

Each resampling vector can be decomposed into two components, with the first component describing the geometrical warping and the second component accounting for any difference in size between the predicted picture (horizontal size H and vertical size V) and the reference picture (horizontal size H_R and vertical size V_R). This decomposition is as follows:

$$\underline{v}^{00} = \underline{v}^{00}_{warp} + \underline{v}^{00}_{size} = \underline{v}^{00}_{warp} + (0,0)$$
$$\underline{v}^{H0} = \underline{v}^{H0}_{warp} + \underline{v}^{H0}_{size} = \underline{v}^{H0}_{warp} + (H_R - H, 0)$$
$$\underline{v}^{0V} = \underline{v}^{0V}_{warp} + \underline{v}^{0V}_{size} = \underline{v}^{0V}_{warp} + (0, V_R - V)$$
$$\underline{v}^{HV} = \underline{v}^{HV}_{warp} + \underline{v}^{HV}_{size} = \underline{v}^{HV}_{warp} + (H_R - H, V_R - V)$$

P.2 Syntax

Whenever the reference picture resampling bit is set in the PLUSTPYPE field of the picture header, the RPRP field of the picture header includes parameters which control the Reference Picture Resampling process. This includes a two-bit Warping Displacement Accuracy (WDA) field, may include eight warping parameters or one-bit warping parameter refinements, and includes a fill mode, as described in this subclause.

P.2.1 Warping Displacement Accuracy (WDA) (2 bits)

A two-bit warping displacement accuracy field WDA appears first in the RPRP field of the bitstream, and indicates the accuracy of the displacements for each pixel. A value of "10" indicates that the *x*- and *y*-displacements for each pixel are quantized to half-pixel accuracy. A value of "11" indicates that the displacements are quantized to $\frac{1}{16}$-pixel accuracy. The use of other values is reserved.

P.2.2 Warping parameters (Variable length)

When the Reference Picture Resampling parameters are sent for an INTER or B-picture, or an Improved PB-frame, eight warping parameters are included in the picture header using the variable length code (VLC) shown in Table D.3. For an EP-picture using SNR scalability, the warping parameters of the lower layer are used and no warping parameters are transmitted. If the Reference Picture Resampling bit is set in the PLUSPTYPE field of the picture header for an EP-picture using spatial scalability, the warping parameters of the lower layer are refined to the accuracy needed for the current layer by multiplying the warping parameter for each up-sampled dimension (warping parameters with subscript x and/or y) of the lower layer by two and adding the value of one additional bit that is sent in place of the associated warping parameter to define the least significant bit of the warping parameter.

The eight integer warping parameters (or their one-bit refinements) are sent in the following order:

$$w^0_x, w^0_y, w^x_x, w^x_y, w^y_x, w^y_y, w^{xy}_x, \text{ and } w^{xy}_y$$

When not a one-bit refinement, these parameters are sent in a similar manner as motion vector differences in the Unrestricted Motion Vector mode when PLUSPTYPE is present, using Table D.3 with no restriction on the range of the parameters (i.e. a range of –4095 to +4095). As when encoding motion vector difference pairs, an emulation prevention bit shall be added as needed after each pair of warping parameters is sent, such that if the all-zero codeword (the value +1 in half-pixel units) of Table D.3 is used for both warping parameters in the pair

(w_x^0 and w_y^0, w_x^x and w_y^y, or w_x^{xy} and w_y^{xy}) the pair of codewords is followed by a single bit equal to 1 to prevent start code emulation.

These eight warping parameters are interpreted as picture corner displacements relative to the displacements that would be induced by removing the resizing component of the resampling vectors. The warping parameters are scaled to represent half-pixel offsets in the luminance field of the current picture, and the range for the values of these paramenters is –4095 to +4095. The warping parameters are defined using the resampling vectors by the relations:

$$w_x^0 = 2r_x^0 \qquad w_y^0 = 2r_y^0$$

$$w_x^x = 2\left(r_x^x - (H_R - H)\right) \qquad w_y^x = 2r_y^x$$

$$w_x^y = 2r_x^y \qquad w_y^y = 2\left(r_y^y - (V_R - V)\right)$$

$$w_x^y = 2r_x^y \qquad w_y^{xy} = 2r_y^{xy}$$

P.2.3 Fill Mode (FILL_MODE) (2 bits)

For an INTER or B-picture or an Improved PB-frame, immediately following the VLC-coded warping parameters in the picture header are two bits which define the fill-mode action to be taken for the values of pixels for which the calculated location in the reference picture lies outside of the reference picture area. The meaning of these two bits is shown in Table P.1 and their location is shown in Figure P.2. For an EP-picture, the fill-mode action is the same as that for the reference layer, and the two fill-mode bits are not sent.

Table P.1/H.263 – Fill mode bits/action

fill-mode bits	fill action
0 0	*color*
0 1	*black*
1 0	*gray*
1 1	*clip*

If the fill mode is *clip*, the coordinates of locations in the prior reference picture are independently limited as in the Unrestricted Motion Vector mode so that pixel values outside of the prior reference picture area are estimated by extrapolation from the values of pixels at the image border. If the fill mode is *black*, luminance samples outside of the prior reference picture area are assigned a value of Y = 16 and chrominance samples are assigned a value of $C_B = C_R = 128$. If the fill mode is *gray*, luminance and chrominance values are assigned a value of $Y = C_B = C_R = 128$. If the fill mode is *color*, then additional fields are sent to specify a fill color, as described in the next subclause.

P.2.4 Fill Color Specification (Y_FILL, C_B_EPB, C_B_FILL, C_R_EPB, C_R_FILL) (26 bits)

If the fill mode is *color* and the picture is not an EP-picture, then the fill-mode bits are followed in the bitstream by three eight-bit integers, *Y_fill*, C_B*_ fill*, and C_R*_ fill*, which specify a fill color precisely. Between these three eight-bit integers are two emulation prevention bits (C_B_EPB and C_R_EPB) each of which is equal to 1. The format of this color specification, which is present only when the fill mode is *color*, is shown in Figure P.2. Each eight-bit integer field is sent using its

natural representation. For an EP-picture, the fill-mode action (and fill color) is the same as that for the reference layer, and the fill color specification is not sent.

FILL_MODE	Y_FILL	C_B_EPB	C_B_FILL	C_R_EPB	C_R_FILL

Figure P.2/H.263 – Format of fill mode and fill color specification data

P.3 Resampling algorithm

The method described in this subclause shall be mathematically equal in result to that used to generate the samples of the resampled reference picture. Using the integer warping parameters $w_x^0, w_y^0, w_x^x, w_y^x, w_x^y, w_y^y, w_x^{xy}$, and w_y^{xy} integer parameters u_x^{00}, u_y^{00}, u_x^{H0}, u_y^{H0}, u_x^{0V}, u_y^{0V}, u_x^{HV}, and u_y^{HV} which denote the x- and y-displacements at the corners of the luminance field in $\frac{1}{32}$-pixel accuracy (the actual displacements are obtained by dividing these values by 32) are defined as:

$$u_x^{00} = 16 w_x^0 \qquad u_y^{00} = 16 w_y^0$$

$$u_x^{H0} = 16\left(w_x^0 + w_x^x + 2(H_R - H)\right) \qquad u_y^{H0} = 16\left(w_y^0 + w_y^x\right)$$

$$u_x^{0V} = 16\left(w_x^0 + w_x^y\right) \qquad u_y^{0V} = 16\left(w_y^0 + w_y^y + 2(V_R - V)\right)$$

$$u_x^{HV} = 16\left(w_x^0 + w_x^x + w_x^y + w_x^{xy} + 2(H_R - H)\right) \qquad u_y^{HV} = 16\left(w_y^0 + w_y^x + w_y^y + w_y^{xy} + 2(V_R - V)\right)$$

Next, H' and V', which denote the horizontal and vertical size of the *virtual frame*, are defined as the smallest integers that satisfy the following condition:

$$H' \geq H, V' \geq V,\ H' = 2^m,\ V' = 2^n,\ \textit{m} \text{ and } \textit{n} \text{ are positive integers}$$

By applying bilinear extrapolation to the corner vectors of the luminance field, the integer parameters u_x^{LT}, u_y^{LT}, u_x^{RT}, u_y^{RT}, u_x^{LB}, u_y^{LB}, u_x^{RB}, and u_y^{RB} which denote the x- and y-displacements of the luminance field at the *virtual points* $(x, y) = (0, 0)$, $(H', 0)$, $(0, V')$, and (H', V') in $\frac{1}{32}$-pixel accuracy (the actual displacements are obtained by dividing these values by 32) are defined as:

$$u_x^{LT} = u_x^{00} \qquad u_y^{LT} = u_y^{00}$$

$$u_x^{RT} = \left((H - H')u_x^{00} + H' u_x^{H0}\right) // H \qquad u_y^{RT} = \left((H - H')u_y^{00} + H' u_y^{H0}\right) // H$$

$$u_x^{LB} = \left((V - V')u_x^{00} + V' u_x^{0V}\right) // V \qquad u_y^{LB} = \left((V - V')u_y^{00} + V' u_y^{0V}\right) // V$$

$$u_x^{RB} = \left((V - V')\left((H - H')u_x^{00} + H' u_x^{H0}\right) + V'\left((H - H')u_x^{0V} + H' u_x^{HV}\right)\right) // (HV)$$

$$u_y^{RB} = \left((V - V')\left((H - H')u_y^{00} + H' u_y^{H0}\right) + V'\left((H - H')u_y^{0V} + H' u_y^{HV}\right)\right) // (HV)$$

where "//" denotes integer division that rounds the quotient to the nearest integer, and rounds half integer values away from 0.

In the remainder of this annex, it is assumed that the centers of the pixels lie at the points $(x,y)=\left(i+\frac{1}{2}, j+\frac{1}{2}\right)$ for both the luminance and chrominance fields. Integer parameters i, j are defined as:

- $i = 0, \ldots, H-1$ and $j = 0, \ldots, V-1$ for luminance, and
- $i = 0, \ldots, \frac{H}{2}-1$ and $j = 0, \ldots, \frac{V}{2}-1$ for chrominance.

This implies that different coordinate systems are used for luminance and chrominance as shown in Figure P.3. Using the coordinate system for chrominance, the integer parameters u_x^{LT}, u_y^{LT}, u_x^{RT}, u_y^{RT}, u_x^{LB}, u_y^{LB}, u_x^{RB}, and u_y^{RB} defined above can also be regarded as the x- and y-displacements of the chrominance field at virtual points $(x, y) = (0, 0)$, $(H'/2, 0)$, $(0, V'/2)$, and $(H'/2, V'/2)$ in $\frac{1}{64}$-pixel accuracy (the actual displacements are obtained by dividing these values by 64). Using these parameters and an additional parameter S, which is defined as 2 for luminance and as 1 for chrominance, the resampling algorithm for the luminance and chrominance fields are defined using common equations.

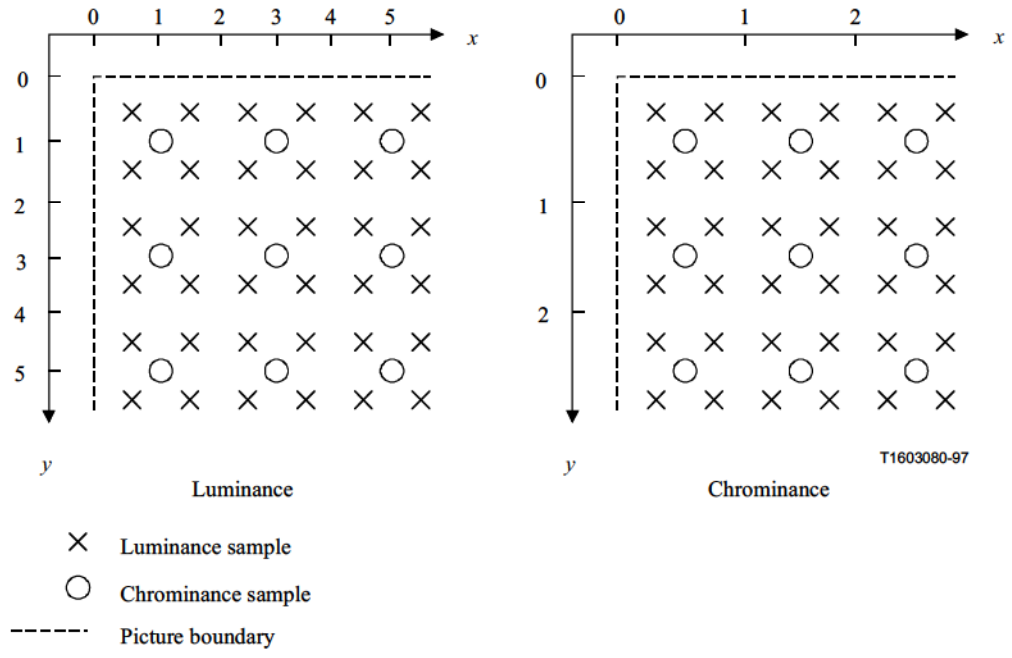


Figure P.3/H.263 – Coordinate systems for luminance and chrominance fields

The integer parameters $u_x^L(j)$, $u_y^L(j)$, $u_x^R(j)$, and $u_y^R(j)$ which denote the x- and y-displacements of the picture field at $(x, y) = (0, j+\frac{1}{2})$ and $(SH'/2, j+\frac{1}{2})$ in $\frac{s}{64}$-pixel accuracy (the actual displacements are obtained by dividing these values by $64/S$) are defined using one dimensional linear interpolation as:

$$u_x^L(j)=\left((SV'-2j-1)u_x^{LT}+(2j+1)u_x^{LB}\right)//(SV') \quad u_y^L(j)=\left((SV'-2j-1)u_y^{LT}+(2j+1)u_y^{LB}\right)//(SV')$$

$$u_x^R(j)=\left((SV'-2j-1)u_x^{RT}+(2j+1)u_x^{RB}\right)//(SV') \quad u_y^R(j)=\left((SV'-2j-1)u_y^{RT}+(2j+1)u_y^{RB}\right)//(SV')$$

where "//" denotes integer division that rounds the quotient to the nearest integer, and rounds half integer values away from 0.

Finally, the parameters that specify the transformed position in the reference picture become:

$$I_R(i,j)=Pi+\left((SH'-2i-1)u_x^L(j)+(2i+1)u_x^R(j)+32H'/P\right)///(64H'/P)$$

$$J_R(i,j)=Pj+\left((SH'-2i-1)u_y^L(j)+(2i+1)u_y^R(j)+32H'/P\right)///(64H'/P)$$

$$i_R(i,j)=I_R(i,j)///P \qquad j_R(i,j)=J_R(i,j)///P$$

$$\varnothing_x=I_R(i,j)-\left(I_R(i,j)///P\right)P \qquad \varnothing_y=J_R(i,j)-\left(J_R(i,j)///P\right)P$$

where

"///" integer division with truncation towards the negative infinity;

"/" integer division (in this case resulting in no loss of accuracy);

P accuracy of x- and y-displacements (P = 2 when WDA = "10" and P = 16 when WDA = "11" or is absent, see P.2.1 for the definition of WDA);

$\left(\frac{I_R(i,j)}{P}+\frac{1}{2},\frac{J_R(i,j)}{P}+\frac{1}{2}\right)$ (x, y) location of the transformed position (both $I_R(i,j)$ and $J_R(i,j)$ are integers);

$\left(i_R(i,j)+\frac{1}{2},j_R(i,j)+\frac{1}{2}\right)$ (x, y) location of the sampling point near the transformed position (both $i_R(i,j)$ and $j_R(i,j)$ are integers);

$(\varnothing_x,\varnothing_y)$ bilinear interpolation coefficients of the transformed position (both $\varnothing_x$ and $\varnothing_y$ are integers).

The computation of this equation can be simplified by replacing the divisions by shift operations, since $64H'/P = 2^{m+2}$ when P = 16 and $64H'/P = 2^{m+5}$ when P = 2.

Using these parameters, the sample value, $E_P(i,j)$, of the pixel located at $(x,y)=\left(i+\frac{1}{2},j+\frac{1}{2}\right)$ in the resampled picture is obtained using bilinear interpolation by:

$$E_P(i,j)=\Big((P-\varnothing_y)\big((P-\varnothing_x)E_R(i_R,j_R)+\varnothing_x E_R(i_R+1,j_R)\big)$$

$$+\varnothing_y\big((P-\varnothing_x)E_R(i_R,j_R+1)+\varnothing_x E_R(i_R+1,j_R+1)\big)+P^2/2-1+RCRPR\Big)/P^2$$

where "/" denotes division by truncation. i_R and j_R are simplified notations for $i_R(i,j)$ and $j_R(i,j)$, and $E_R(i_R,j_R)$ denotes the sample value of the pixel located at $(x,y)=\left(i_R+\frac{1}{2},j_R+\frac{1}{2}\right)$ in the reference picture after extrapolation using the proper fill mode if necessary. The value of parameter RCRPR is defined as follows:

- For a B-picture (or the B-part of an Improved PB-frame) which has a P-picture as its temporally subsequent anchor picture, RCRPR is equal to the rounding type (RTYPE) bit in

MPPTYPE (see 5.1.4.3) of this temporally subsequent P-picture. This implies that for Improved PB-frames, RCRPR has the same value for the P-part and the B-part.

- For other types of pictures, RCRPR is equal to the RTYPE bit of the current picture.

P.4 Example of implementation

In this subclause, an implementation example of the algorithm described in the previous subclause is provided as pseudo-code.

P.4.1 Displacements of virtual points

When a large picture is coded, a straightforward implementation of the equation for obtaining parameters u_x^{RB} and u_y^{RB} shown in P.3 may necessitate the usage of variables requiring more that 32 bits for their binary representation. For systems which cannot easily use 64-bit integer or floating point registers, an example of an algorithm which does not require variables with more than 32 bits for the calculation of u_x^{RB} and u_y^{RB} is shown below.

Since H, V, H', and V' are divisible by 4, the definition of u_x^{RB} can be rewritten as:

$$u_x^{RB} = \Big((V_Q - V_Q')\big((H_Q - H_Q')u_x^{00} + H_Q' u_x^{H0}\big) + V_Q'\big((H_Q - H_Q')u_x^{0V} + H_Q' u_x^{HV}\big)\Big) // A$$

where H_Q=H/4, V_Q=V/4, H_Q'=H'/4, V_Q'=V'/4, $A = H_Q V_Q$, and "//" denotes integer division that rounds the quotient to the nearest integer, and rounds half integer values away from 0. Next, parameters T_T and T_B are defined as:

$$T_T = (H_Q - H_Q')u_x^{00} + H_Q' u_x^{H0}$$
$$T_B = (H_Q - H_Q')u_x^{0V} + H_Q' u_x^{HV}$$

for simplicity of description. Using operator "///" which denotes integer division with truncation towards the negative infinity, and operator "%" which is defined as a % b = a – (a /// b) b, the value of u_x^{RB} can be obtained via the following pseudo-code:

```
q = (VQ - VQ')*(TT /// A) + VQ'*(TB /// A) + ((VQ - VQ')*(TT % A) + VQ'
    *(TB % A)) /// A;
r = ((VQ - VQ')*(TT % A) + VQ' *(TB % A)) % A;
if (q < 0)
  uxRB = q+(r+(A-1)/2)/A;
else
  uxRB = q+(r+A/2)/A;
```

The value of u_y^{RB} can also be calculated using this algorithm.

P.4.2 Resampling algorithm

To simplify the description of the algorithm, a function `prior_sample` is defined. Its purpose is to generate a pixel value for any integer location (i_p, j_p) relative to the prior reference picture sampling grid:

```
clip(xmin, x, xmax) {
  if (x < xmin) {
    return xmin;
  } else if (x > xmax) {
    return xmax;
```

```
        } else {
          return x;
        }
      }
prior_sample (ip, jp) {
    if (FILL_MODE = clip) {
      ic = clip (0, ip, S*HR /2-1);
      jc = clip (0, jp, S*VR /2-1);
      return prior_ref[ic, jc];
    } else {
      if ((ip < 0) OR (ip > S*HR /2-1) or (jp < 0) OR (jp > S*VR /2-1) {
        return fill_value;
      } else {
        return prior_ref[ip,jp];
      }
    }
}
```

In the pseudo-code, `prior_ref[i, j]` indicates the sample in column *i* and row *j* in the temporally previous reference picture.

Next, a filter function that implements the bilinear interpolation described in P.3 is defined. It is assumed that all arguments of the following function are integers and that the bilinear interpolation coefficients, $Ø_x$ and $Ø_y$, are quantized in the range 0, …, *P*–1 (inclusive).

```
filter(x0, y0, Øx, Øy) {
  return /((P-Øy)·((P-Øx)·prior_sample(x0, y0)+Øx·prior_sample(x0+1, y0))+
            Øy·((P-Øx)·prior_sample(x0, y0+1)+Øx·prior_sample(x0+1, y0+1))+
          P^2/2-1+RCRPR]/P^2;
}
```

Finally, the method for warping the reference picture to generate a prediction for the current picture can be specified in terms of these functions. The pixels of the prediction picture can be generated in a raster scan order. It is assumed that the values $u_x^L(j)$, $u_y^L(j)$, $u_x^R(j)$, and $u_y^R(j)$ are already calculated and loaded into variables u_x^L, u_y^L, u_x^R, and u_y^R. Defining parameter *D* as $D = 64H'/P$ and noting that $H' = 2^m$, the sample values of the pixels in the *j* th line of the resampled field (the topmost line is defined as the 0th line) is obtained by the following pseudo-code:

```
a_x^i = D * P + 2 * (u_x^R - u_x^L);
a_y^i = 2 * (u_y^R - u_y^L);
a_x = u_x^L * S * 2^m + (u_x^R - u_x^L); + D / 2;
a_y = j * D * P + u_y^L * S * 2^m + (u_y^R - u_y^L); + D / 2;
for (i = 0; i < S * H / 2; i++) {
      I_R = a_x /// D;
      J_R = a_y /// D;
      i_R = I_R /// P;
```

```
        jR = JR /// P;
        Øx = IR − (iR * P);
        Øy = JR − (jR * P);
        new_ref[i, j] = filter(iR, jR, Øx, Øy);
        ax += axi;
        ay += ayi;
}
```

where all the variables used in this code are integer variables and `new_ref[i, j]` indicates the sample generated for column i and row j in the resampled reference picture. According to the definition of the parameters, all the divisions in this code can be replaced by binary shift operations. For example, when $P = 16$:

```
IR = ax /// D;
JR = ay /// D;
iR = IR /// P;
jR = JR /// P;
Øx = IR − (iR  * P);
Øy = JR − (jR  * P);
```

can be rewritten, assuming that a_x, a_y, I_R, and J_R are binary-coded integer variables in two's complement representation, as:

```
IR = ax >> (m+2);
JR = ay >> (m+2);
iR = IR >> 4;
jR = JR >> 4;
Øx = IR & 15;
Øy = JR & 15;
```

where "$>> N_{shift}$" denotes a right arithmetic binary shift by N_{shift} bits (N_{shift} is a positive integer), and "&" denotes a bit-wise AND operation.

P.5 Factor-of-4 resampling

Factor-of-4 resampling, which converts both the horizontal and vertical size of the picture by a factor of 2 or $\frac{1}{2}$, is a special case of the resampling algorithm described in P.3. The simplified description of the resampling algorithm for this special case is provided in this subclause.

The value of the parameter RCRPR used in Figures P.4 to P.6 is determined by the Rounding Type (RTYPE) bit in MPPTYPE (see 5.1.4.3) as described in P.3. Additionally, "/" in the figures indicates division by truncation.

P.5.1 Factor-of-4 upsampling

The pixel value interpolation method used in factor-of-4 upsampling for internal pixels are shown in Figure P.4. By assuming the existence of pixels outside the picture according to the selected fill mode (see P.2.3 and P.2.4), the same interpolation method is applied for boundary pixels. The interpolation method for boundary pixels when *clip* is selected as the fill mode is shown in Figure P.5. Since precise factor-of-4 upsampling requires x- and y-displacements with at least

$\frac{1}{4}$-pixel accuracy, the Warping Displacement Accuracy (WDA) field specified in P.2.1 must be set to "11" or the resampling must be implicitly invoked in order to use this upsampling method.

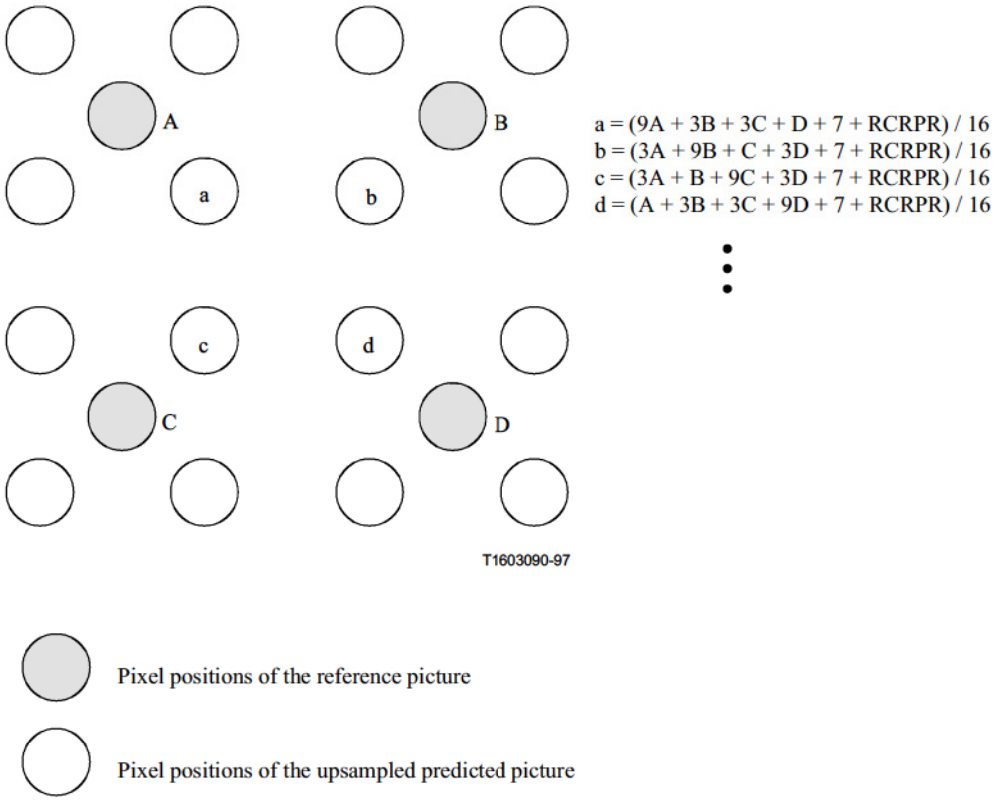


Figure P.4/H.263 – Factor-of-4 upsampling for pixels inside the picture

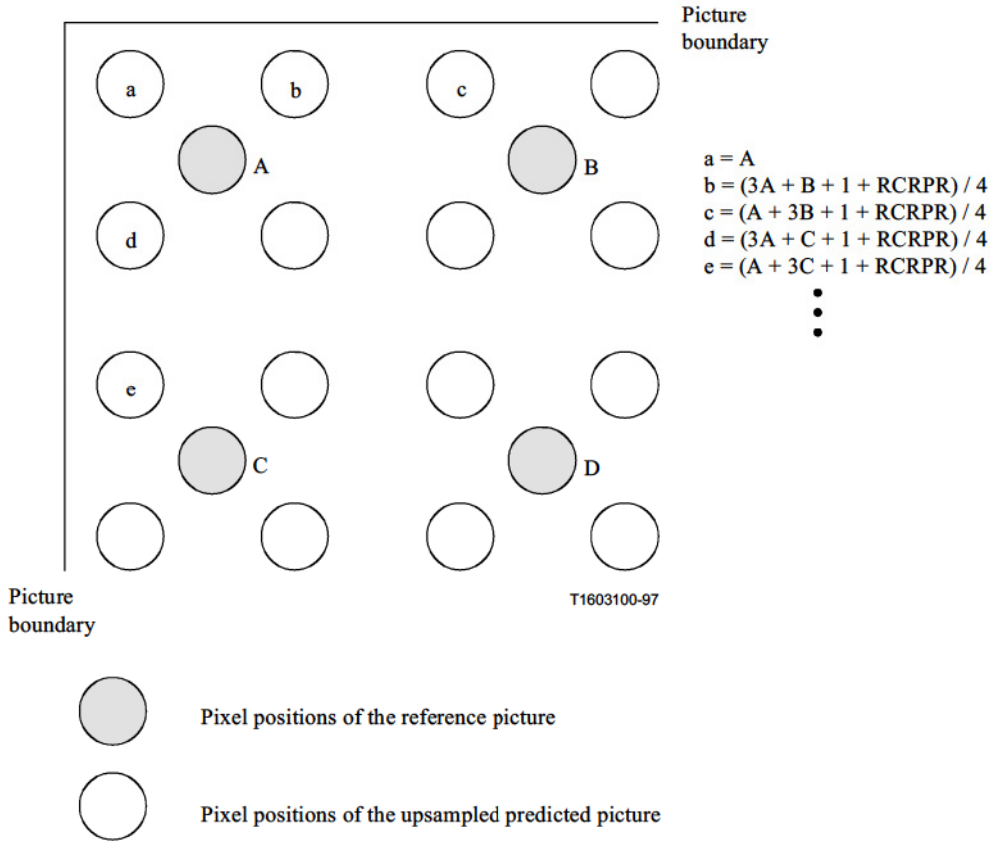


Figure P.5/H.263 – Factor-of-4 upsampling for pixels at the picture boundary (fill mode = *clip*)

P.5.2 Factor-of-4 downsampling

The pixel value interpolation method for factor-of-4 downsampling is shown in Figure P.6. Since x- and y-displacements with $\frac{1}{2}$-pixel accuracy is sufficient for precise factor-of-4 downsampling, both "10" and "11" are allowed as the value of the displacement accuracy (WDA) field (if present) specified in P.2.1.

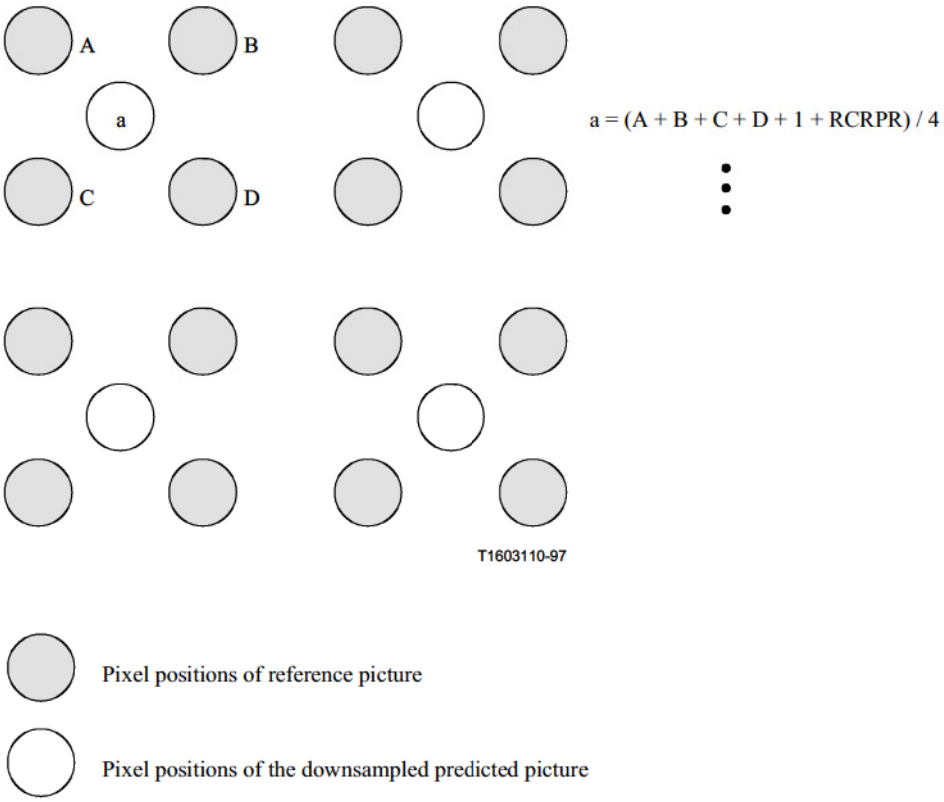


Figure P.6/H.263 – Factor-of-4 downsampling

ANNEX Q

Reduced-Resolution Update mode

Q.1 Introduction

This annex describes an optional Reduced-Resolution Update mode of this Recommendation. The capability of this mode is signaled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

The Reduced-Resolution Update mode is expected to be used when encoding a highly active scene, and provides the opportunity to increase the coding picture rate while maintaining sufficient subjective quality. This mode allows the encoder to send update information for a picture that is encoded at a reduced resolution, while preserving the detail in a higher resolution reference image to create a final image at the higher resolution.

The syntax of the bitstream in this mode is identical to the syntax for coding without the mode, but the semantics, or interpretation of the bitstream is somewhat different. In this mode, the portion of the picture covered by a macroblock is twice as wide and twice as high. Thus, there is approximately one-quarter the number of macroblocks as there would be without this mode. Motion vector data also refers to blocks of twice the normal height and width, or 32 × 32 and 16 × 16 instead of the normal 16 × 16 and 8 × 8. On the other hand, the DCT or texture data should be thought of as describing 8 × 8 blocks on a reduced-resolution version of the picture. To produce the final picture, the texture data is decoded at a reduced resolution and then up-sampled to the full resolution of the picture.

After upsampling, the full resolution texture image is added to the (already full resolution) motion compensated image to create the image for display and further reference.

In this mode, a picture which has the horizontal size H and vertical size V as indicated in the picture header is created as a final image for display.

In this mode, a referenced picture used for prediction and created for further decoding has a horizontal size H_R and a vertical size V_R which are the same as in the default mode as defined in 4.1. That is, the H_R and V_R are:

$$H_R = ((H + 15) / 16) * 16$$
$$V_R = ((V + 15) / 16) * 16$$

where H and V are the horizontal size and the vertical size as indicated in the picture header and "/" is defined as division by truncation.

Then in this annex, the texture is coded at a reduced resolution with height and width of H_C and V_C where:

$$H_C = ((H_R + 31) / 32) * 32$$
$$V_C = ((V_R + 31) / 32) * 32$$

and "/" is defined as division by truncation.

If H_C and H_R, or V_C and V_R are not identical to each other, such as in the QCIF format, an extension of the referenced picture is performed , and the picture is decoded in the same manner as if the width and height are H_C and V_C. Then the resulting picture which is tiled by 32 * 32 macroblocks is cropped at the right and the bottom to the width H_R and height V_R, and this cropped picture is stored as a reference picture for further decoding.

If both H and V are the same as those of a resulting picture having the width H_C and the height V_C, this resulting picture is used for display. Otherwise, this resulting picture is further cropped to the size H * V, and the cropped picture here is used for display purpose only.

If the Temporal, SNR, and Spatial Scalability mode (Annex O) or Reference Picture Resampling mode (Annex P) is also used with this option, the source format of the current picture might be different from that of the reference picture. In this case, the resampling of the reference picture shall be performed before decoding.

NOTE – This mode can be used with the Reference Picture Selection mode (see Annex N) without modification, since the reference picture (after possible resampling by the Reference Picture Resampling mode) has the same size as indicated in the current picture header when this option is used.

Q.2 Decoding procedure

Figure Q.1 shows the block diagram of the block decoding in the Reduced-Resolution Update mode.

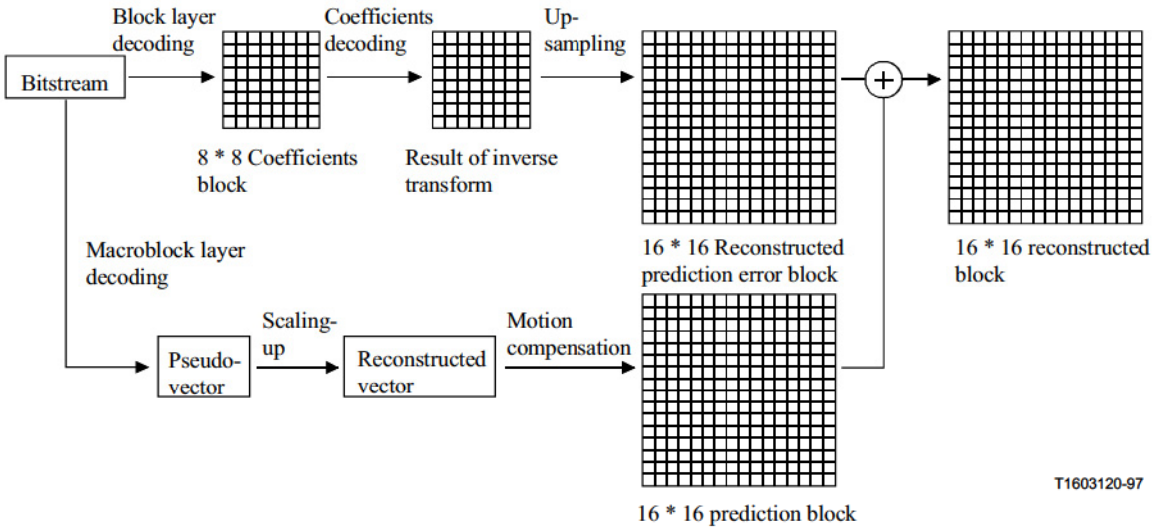


Figure Q.1/H.263 – Block diagram of block decoding in Reduced-Resolution Update mode

The decoding procedure is given in the following subclauses:

Q.2.1 Reference preparation

In some cases, the available reference picture has different size from H_C and V_C. Then, the reference picture shall be converted before decoding procedure according to Q.2.1.1 or Q.2.1.2.

Q.2.1.1 Reference Picture Resampling

If the Temporal, SNR, and Spatial Scalability mode (Annex O) or Reference Picture Resampling mode (Annex P) is also used with this option, the source format of the current picture might be different from that of the reference picture. In this case, the resampling of the reference picture according to each annex shall be performed first.

Q.2.1.2 Extension of Reference Picture

If H_R or V_R is not divisible by 32, such as with the QCIF format, the reference picture is extended. The detailed procedure for this extension is defined in Q.3.

Q.2.2 Macroblock layer decoding

Decoding can be thought of as operating on "enlarged" blocks of size 32 × 32 in luminance and 16 × 16 in chrominance. The texture and motion data for each enlarged block is decoded to create a 32 × 32 motion block and a 32 × 32 texture block, as described in Q.2.2.1 and Q.2.2.2 respectively. These motion and texture blocks are then added as described in Q.2.2.3.

Q.2.2.1 Motion compensation

First, each component of the macroblock motion vector (or four macroblock motion vectors) is formed from MVD (and possibly $MVD_{2\text{-}4}$). If in Improved PB-Frames mode, MV_F and MV_B for B-Picture is also formed from MVDB. The detailed procedure for this motion vector formation is defined in Q.4. If the current picture mode is a B-picture or EP-picture, the motion vector for forward and backward are also obtained according to Q.4.

The motion vector for the two chrominance blocks of the macroblock is obtained from the macroblock motion vector according to 6.1.1. If either the Advanced Prediction mode or the Deblocking Filter mode is in use and thus four motion vectors are defined for the macroblock, the motion vector for both chrominance blocks is obtained from the four motion vectors according to F.2. If in the Improved PB-frames mode, the creation of the chrominance vector is specified in Annex M. If a B-picture or EP-picture is used, the creation of the chrominance vector is specified in Annex O.

Then, a prediction is formed from the motion vector for an INTER macroblock. Four 16 × 16 luminance prediction blocks are obtained from the macroblock motion vector, and two 16 × 16 chrominance prediction blocks are obtained from the chrominance motion vector. For interpolation for subpixel prediction, refer to 6.1.2. If Advanced Prediction mode is also used, an enlarged overlapped motion compensation is performed to obtain four 16 × 16 luminance prediction blocks using enlarged weighting matrices, for which a detailed procedure is defined in Q.5. If the current picture mode is an Improved PB-frame, B-picture, or EP-picture, the prediction is obtained according to the other relevant annexes, except that the size of the predicted blocks is 16 × 16 instead of 8 × 8.

Q.2.2.2 Texture decoding

First, the bitstream of the block layer is decoded according to 5.4. Then, coefficients are decoded and the 8 × 8 reduced-resolution reconstructed prediction error blocks are obtained as the result of inverse transform according to 6.2.

Then the 16 × 16 reconstructed prediction error blocks are obtained by upsampling the 8 × 8 reduced-resolution reconstructed prediction error blocks. For the creation of the edge pixels in each 16 × 16 reconstructed prediction error block, only the pixels which belong to the corresponding block are used. The detailed procedure is defined in Q.6.

Q.2.2.3 Reconstruction of block

For each luminance and chrominance block, a summation of the prediction and prediction error is performed. The procedure is identical to 6.3.1, except that the size of blocks is 16 × 16 instead of 8 × 8. Then the clipping is performed according to the 6.3.2.

Then, a block boundary filter is applied to the boundary pixels of the 16 × 16 reconstructed blocks. The detailed procedure is described in Q.7.

Q.2.3 Picture store

If both H_R and V_R are divisible by 32, such as with the CIF format, the resulting picture reconstructed as described in Q.2.2 is stored as a reference picture as it is for further decoding. Otherwise, such as the QCIF format, the reconstructed picture which is just covered with 32 × 32 macroblocks is cropped at the right and the bottom to the width H_R and height V_R, and this cropped picture is stored as a reference picture for further decoding.

Q.2.4 Display

If both H and V are the same as H_C and V_C, the resulting picture in Q.2.2 is used for display purposes as it is. Otherwise, this resulting picture is further cropped to the size $H \times V$, and the cropped picture is used for display purpose only.

Q.3 Extension of referenced picture

If H_R or V_R is not divisible by 32, such as with the QCIF format, the extension of the referenced picture is performed before decoding macroblock/block layer. The width and the height of the extended reference picture for luminance are the next larger size that would be divisible by 32, and those for chrominance are the next larger size that would be divisible by 16.

NOTE – The width and height of the referenced picture in the default mode are always extended to be divisible by 16 even if the picture format has a width or height that is not divisible by 16, because the picture shall be decoded as if the width or height had the next larger size that would be divisible by 16. See 4.1.

If neither Unrestricted Motion Vector mode, Advanced Prediction mode nor Deblocking Filter mode is used with this option, the extended pixels can be arbitrary values, because the extended pixels will be never used as a reference pixels of the decoded picture to be reconstructed and displayed.

If either Unrestricted Motion Vector mode, Advanced Prediction mode or Deblocking Filter mode is also used with this option, the extension of the referenced picture is performed by duplicating the edge pixel of the referenced picture, in order to ensure the decoding when motion vectors point outside the right and bottom edge of the picture.

For example, if the Reduced-Resolution Update mode is used for a QCIF, the width of the referenced picture is 176 and the height is 144, which are not divisible by 32. In order to cover a QCIF picture with 32 × 32-sized macroblocks, the number of macroblock row should be 6, and the number of macroblock column should be 5. Therefore, the width of the extended referenced picture is 192 and the height is 160.

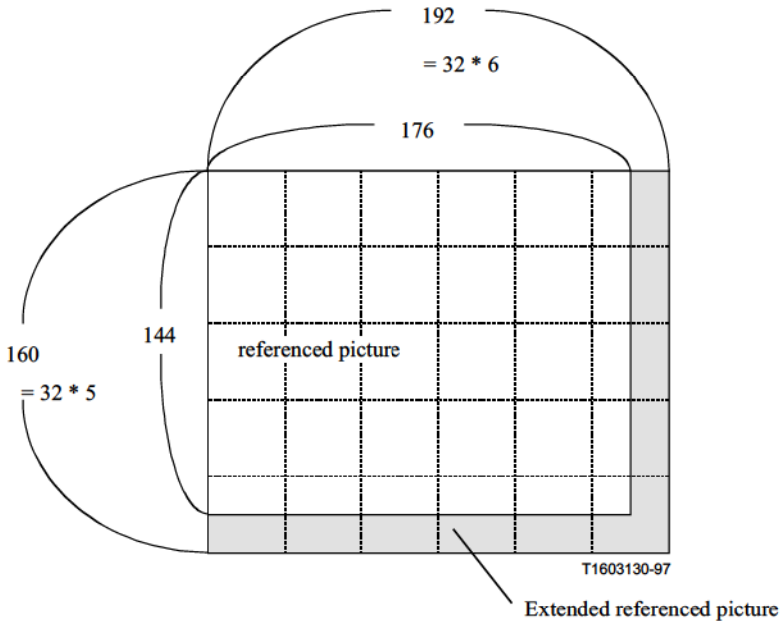


Figure Q.2/H.263 – Extension of referenced picture for QCIF picture size

The extension of the referenced picture in QCIF is illustrated in Figure Q.2. The extended referenced picture for luminance is given by the following formula:

$$R_{RRU}(x, y) = R(x', y')$$

where:

x, y = spatial coordinates of the extended referenced picture in the pixel domain,

x', y = spatial coordinates of the referenced picture in the pixel domain,

$R_{RRU}(x, y)$ = pixel value of the extended referenced picture at (x, y),

R(x′, y′) = pixel value of the referenced picture at (x′, y′),

$$x' = \begin{cases} 175 & \text{if } x > 175 \text{ and } x < 192 \\ x & \text{otherwise,} \end{cases}$$

$$y' = \begin{cases} 143 & \text{if } y > 143 \text{ and } y < 160 \\ y & \text{otherwise.} \end{cases}$$

The referenced pictures for chrominance is also extended in the same manner.

Q.4 Reconstruction of motion vectors

In Reduced-Resolution Update mode, the motion vector range is enlarged to approximately double size in both the horizontal and vertical directions. In order to realize an enlarged range using the VLC for MVD defined in Table 14, each vector component is restricted to be only half-pel or zero value. Therefore the range of each motion vector component is [–31.5, 30.5] in the default Reduced-Resolution Update mode. If the Unrestricted Motion Vector mode is used, the vector range [–limit, limit] as defined in D.2 applies to the pseudo-motion vectors and translates into [–(2 * limit–0.5), 2 * limit–1.5] for the motion vectors. For CIF this means that the pseudo-motion vector range is [–32, 31.5] and the motion vector range is [–63.5, 62.5]. If the UUI field is set to "01", the motion vectors are unlimited. However, the motion vectors (not just the pseudo-motion vectors) are always limited to point not more than 15 pixels outside the coded area as described in D.1.1. Figure Q.3 illustrates the possible positions of macroblock motion vector or four motion vector predictions around the (0, 0) vector value. The dashed lines indicate the integer coordinates.

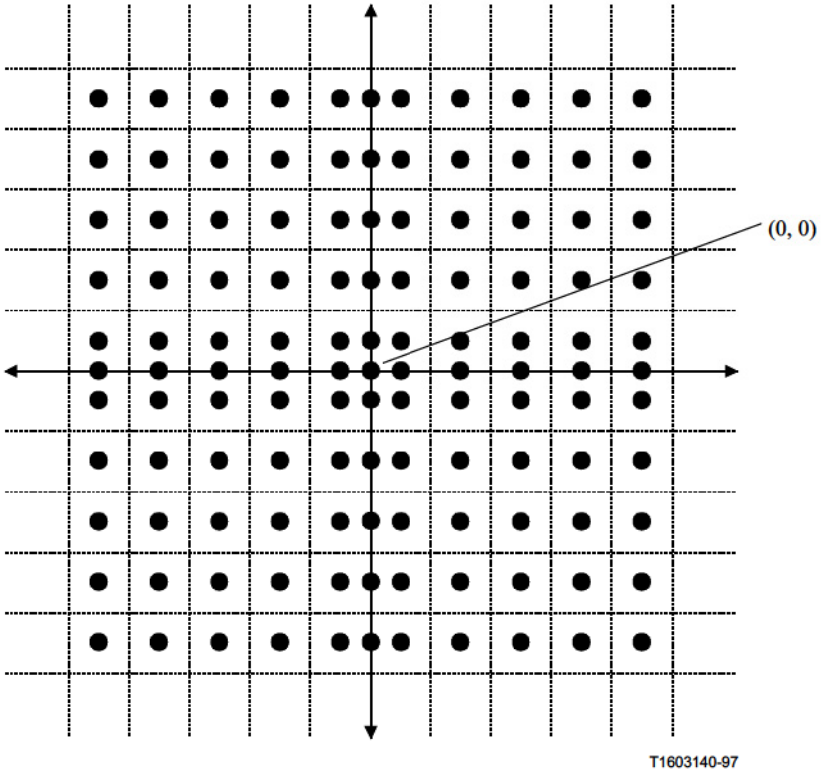


Figure Q.3/H.263 – Reconstruction of motion vector

For the macroblock using differential motion vectors in a B-picture, the motion vector for forward and backward prediction are obtained independently. In the Reduced-Resolution Update mode, the motion vector component MV_c for a luminance block is reconstructed from MVD and $MVD_{2\text{-}4}$ as follows:

1) Pseudo-prediction vector component **pseudo-P_C** is created from the prediction vector component **P_C**.

$$\text{pseudo-}P_C = 0 \qquad \text{if } P_C = 0$$

$$\text{pseudo-}P_C = \text{sign}(P_C) * (|P_C| + 0.5) / 2.0 \qquad \text{if } P_C \neq 0$$

"/" indicates floating-point division (without loss of accuracy). The prediction vector component P_C is defined as the median value of the vector components MV1, MV2 and MV3 as defined in 6.1.1 and F.2.

2) Pseudo-macroblock vector component **pseudo-MV_C** is obtained by adding the motion vector differences MVD (and $MVD_{2\text{-}4}$) from Table 14 to the pseudo-P_C.

In the default Reduced-Resolution Update mode, the value of **pseudo-MV_C** is restricted to the range [–16, 15.5]. Only one of the pair will yield a pseudo-MV_C falling within the permitted range. The procedure is performed in the similar way as defined in 6.1.1.

If the Unrestricted Motion Vector mode is also used with the Reduced-Resolution Update mode, **pseudo-MV_C** is obtained by adding the motion vector differences MVD (and $MVD_{2\text{-}4}$) from Table D.3.

If four motion vectors are present, the procedure is performed in the similar way as defined in F.2.

3) Motion vector component $\mathbf{MV_C}$ is obtained from pseudo-MV_C in the following formula:

$MV_C = 0$ if pseudo-$MV_C = 0$

$MV_C = \text{sign}(\text{pseudo-}MV_C) * (2.0 * |\text{pseudo-}MV_C| - 0.5)$ if pseudo-$MV_C \neq 0$

As a result, each vector component is restricted to have a half-integer or zero value, and the range of each motion vector component is enlarged to approximately twice the pseudo-motion vector range.

4) If the current picture mode is an Improved PB-frame, or when the MBTYPE indicates the direct mode in a B-picture, the motion vector components $\mathbf{MV_F}$ and/or $\mathbf{MV_B}$ for forward and backward prediction are created.

First, pseudo-motion vector components **pseudo-MV_F** and/or **pseudo-MV_B** are calculated based on the rules for the prediction modes defined in Annexes O or M.

In the case of the Bidirectional prediction in the Improved PB-frames mode (see M.2.1) or when the MBTYPE indicates the direct mode in a B-picture (see O.5.2), pseudo-MV_F and pseudo-MV_B are calculated from **pseudo-MV_D** and **pseudo-MV_C** assuming that pseudo-MV_D is zero and pseudo-MV_C is MV, as defined in Annexes G and M.

In the case of forward prediction in the Improved PB-frames mode (see M.2.2), **pseudo-MV_{DB}** is obtained by decoding the variable length code MVDB according to Table 13. Then, pseudo-MV_F is obtained by adding the pseudo-MV_{DB} to the **pseudo-Predictor**. In order to form the pseudo-Predictor, the predictor obtained according to the procedure defined in M.2.2 is converted to the pseudo-Predictor vector according to the formula defined in item 1) of this subclause.

In the case of backward prediction in the Improved PB-frames mode (see M.2.3), pseudo-MV_B is set to zero.

Then, the motion vectors MV_F and/or MV_B for forward and backward prediction are obtained from pseudo-MV_F and/or pseudo-MV_B according to the formula defined in item 3) of this subclause.

Q.5 Enlarged overlapped motion compensation for luminance

If Advanced Prediction mode is also used with Reduced-Resolution Update mode, enlarged matrices of weighting values are used to perform the overlapped motion compensation. Except that the size of each block and weighting matrices is 16 × 16, the procedure of the creation of each prediction block is identical to the description in F.3.

The enlarged matrices of weighting values for the 16 × 16 luminance prediction are given in Figures Q.4, Q.5, and Q.6.

4	4	5	5	5	5	5	5	5	5	5	5	5	5	4	4
4	4	5	5	5	5	5	5	5	5	5	5	5	5	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	6	6	6	6	6	6	6	6	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	5	5	5	5	5	5	5	5	5	5	5	5	4	4
4	4	5	5	5	5	5	5	5	5	5	5	5	5	4	4

Figure Q.4/H.263 – Weighting values, H_0, for prediction with motion vector of current 16 × 16 luminance block

2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
1	1	1	1	2	2	2	2	2	2	2	2	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Figure Q.5/H.263 – Weighting values, H_1, for prediction with motion vector of current 16 × 16 luminance blocks on top or bottom of current 16 × 16 luminance block

2	2	1	1	1	1	1	1	1	1	1	1	1	1	2	2
2	2	1	1	1	1	1	1	1	1	1	1	1	1	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	2	2	1	1	1	1	1	1	1	1	2	2	2	2
2	2	1	1	1	1	1	1	1	1	1	1	1	1	2	2
2	2	1	1	1	1	1	1	1	1	1	1	1	1	2	2

Figure Q.6/H.263 – Weighting values, H_2, for prediction with motion vector of 16 × 16 luminance blocks to the left or right of current 16 × 16 luminance block

Q.6 Upsampling of the reduced-resolution reconstructed prediction error

The 16 × 16 reconstructed prediction error block is obtained by upsampling the 8 × 8 reduced-resolution reconstructed prediction error block. In order to realize a simple implementation, filtering is closed within a block which enables to perform an individual upsampling on block basis. Figure Q.7 shows the positioning of samples. The upsampling procedure for the luminance and chrominance pixels which are inside the 16 × 16 reconstructed prediction error blocks is defined in Q.6.1. For the creation of the luminance and chrominance pixels which are at the boundary of 16 × 16 reconstructed prediction error block, the procedure is defined in Q.6.2. Chrominance blocks as well as luminance blocks are up-sampled. The symbol "/" in Figures Q.8 and Q.9 indicates division by truncation.

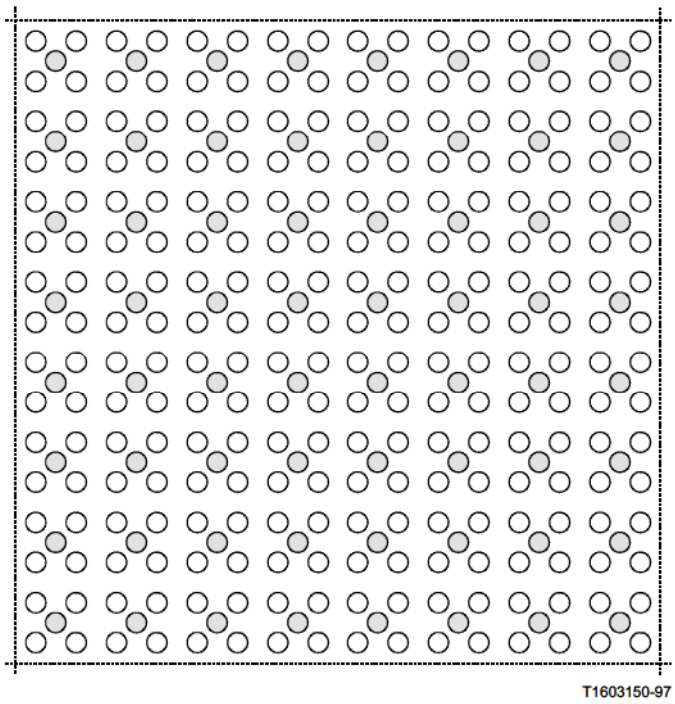


Figure Q.7/H.263 – Positioning of samples in 8 × 8 reduced-resolution reconstructed prediction error block and 16 × 16 reconstructed prediction error block

Q.6.1 Upsampling procedure for the pixels inside a 16 × 16 reconstructed prediction error block

The creation of reconstructed prediction error for pixels inside block is described in Figure Q.8. "/" indicates division by truncation.

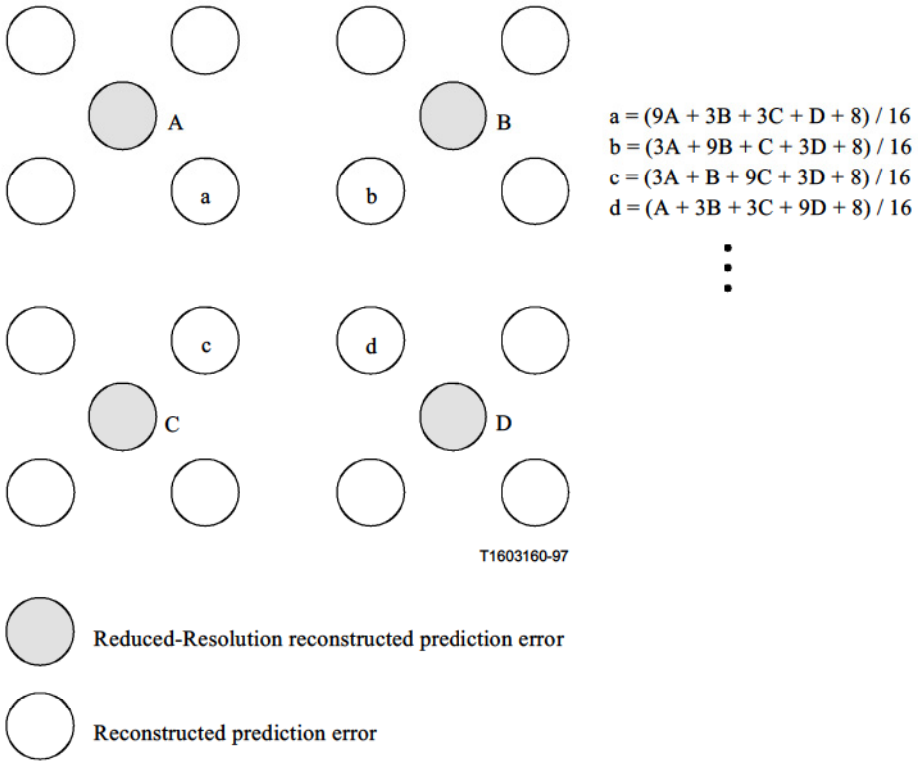


Figure Q.8/H.263 – Creation of reconstructed prediction error for pixels inside block

Q.6.2 Upsampling procdedure for the pixels at the boundary of 16 × 16 reconstructed prediction error block

The creation of reconstructed prediction error for pixels of a 16 × 16 block is shown in Figure Q.9.

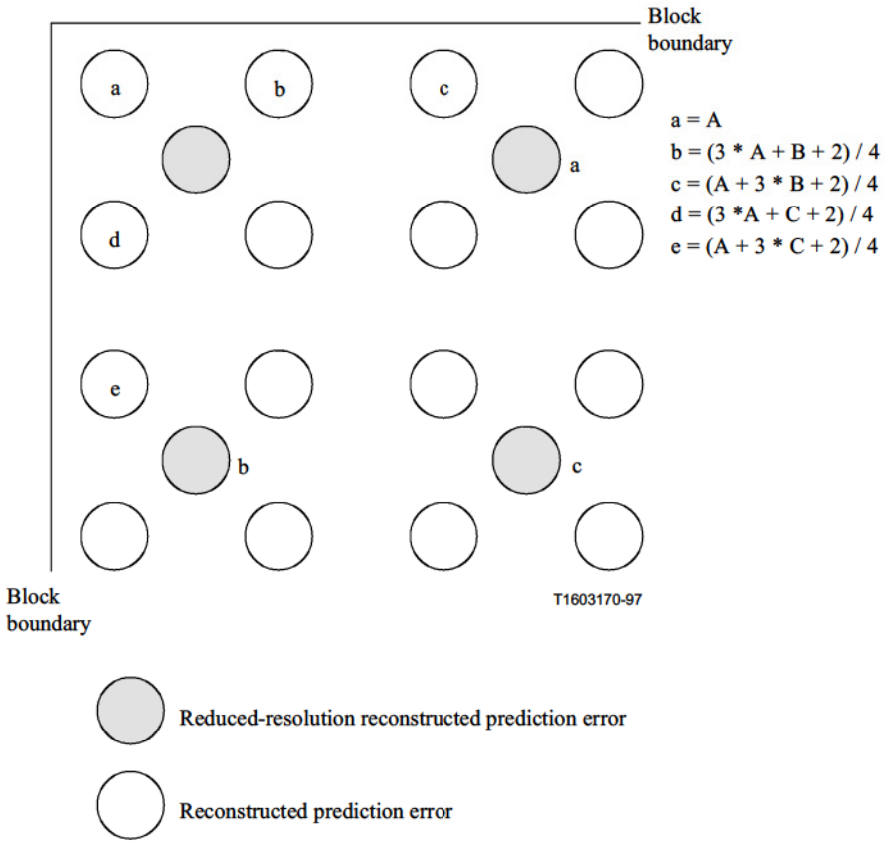


Figure Q.9/H.263 – Creation of reconstructed prediction error for pixels at the block boundary

Q.7 Block boundary filter

The filter operations are performed along the edges of the 16 × 16 reconstructed blocks at the encoder as well as on the decoder side. There are two alternative of filtering, depending on whether Deblocking Filter mode is used or not.

The default filtering in Reduced-Resolution Update mode is performed according to Q.7.1.

If Deblocking Filter mode is also used with Reduced-Resolution Update mode, the filtering is performed according to Q.7.2.

In both cases, filtering is performed on the complete reconstructed image data before storing the data in the picture store for future prediction. No filtering is performed across picture edges, slice edges in Slice Structured mode (see Annex K), or GOB boundaries having GOB headers present in Independent Segment Decoding mode (see Annex R). Chrominance as well as luminance data is filtered.

Q.7.1 Definition of the default block boundary filter

In the Reduced-Resolution Update mode, the default filtering is performed according to this subclause.

If A and B are two pixel values on a line – horizontal or vertical – of the reconstructed picture, and A belongs to one 16 × 16 block called block1 whereas B belongs to a neighboring 16 × 16 block called block2 which is to the right or below of block1. Figure Q.10 shows examples for the position of these pixels.

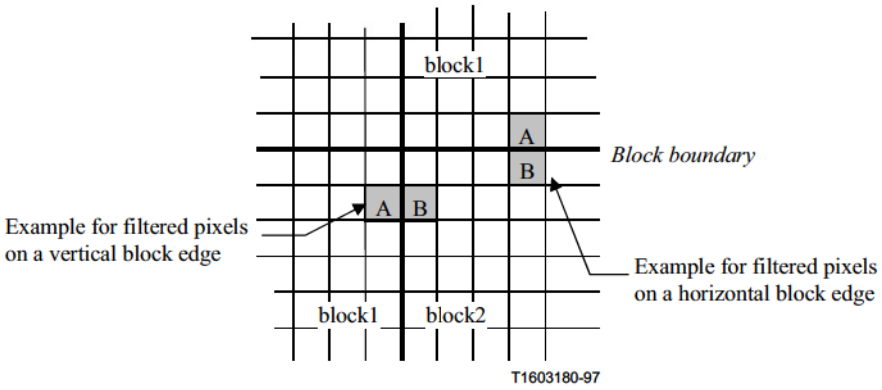


Figure Q.10/H.263 – Default block boundary filter

One of the following conditions must be fulfilled in order to turn the filter on for a particular edge:

- block1 belongs to a coded macroblock (COD==0 || MB-type == INTRA); **or**
- block2 belongs to a coded macroblock (COD==0 || MB-type == INTRA).

A shall be replaced by A1 and B shall be replaced by B1. "/" indicates division by truncation.

$$A1 = (3 * A + B + 2) / 4$$

$$B1 = (A + 3 * B + 2) / 4$$

The order of edges where filtering is performed is identical to the description provided in J.3.

Q.7.2 Definition of the block boundary filter when Deblocking Filter mode is used

If the Deblocking Filter mode (see Annex J) is used with the Reduced-Resolution Update mode, the filtering which is defined in Annex J with one modification is performed on the boundary pixels of 16 × 16 luminance and chrominance blocks, in place of the filtering described in Q.7.1. The one modification of the filtering in Annex J is that the parameter STRENGTH is given the value of positive infinity. This implies that the function UpDownRamp(x, STRENGTH) defined in J.3 becomes a linear function of x.

As a result, the procedure of the deblocking filter described in J.3 is redefined in the following manner:

B1 = clip(B + d1)

C1 = clip(C – d1)

A1 = A – d2

D1 = D + d2

d1 = (A–4B+4C–D) / 8

d2 = clipd1((A – D) / 4, d1/2)

ANNEX R

Independent Segment Decoding mode

R.1 Introduction

This annex describes the optional Independent Segment Decoding mode of this Recommendation, which allows a picture to be decoded without the presence of any data dependencies across slice boundaries or across GOB boundaries having non-empty GOB headers. The use of this mode is indicated in the PLUSPTYPE field of the picture header. The capability to use this optional mode is negotiated by external means (for example, Recommendation H.245).

When the use of this mode is indicated, the video picture segment boundaries (as defined by the boundaries of the slices or the upper boundaries of the GOBs for which GOB headers are sent, or the boundaries of the picture, whichever bounds a region in the smallest way) are treated as picture boundaries when decoding, including the treatment of motion vectors which cross those boundaries (which result in boundary extrapolation when the Unrestricted Motion Vector mode, the Advanced Prediction mode, the Deblocking Filter mode, or the Temporal, SNR, and Spatial Scalability mode are in use, and which are prohibited when none of those optional modes are in use).

R.2 Mode operation

A video picture segment is defined by the following.

If the Slice Structured mode (see Annex K) is not in use, then one GOB or a plural number of consecutive GOBs forms one video picture segment. The location of the top of each video picture segment is indicated by the presence of a non-empty GOB header for which the border of the video picture segment lies just above the macroblocks in the GOB for which a header is present, or the top of the picture, whichever is lower. The location of the bottom of each video picture segment is defined by the top of the next video picture segment, or the bottom of the picture, whichever is uppermost.

If the Slice Structured mode (see Annex K) is in use, then each slice forms one video picture segment.

In the Independent Segment Decoding mode, each video picture segment is decoded with complete independence from all other video picture segments, and is also independent of all data outside the same video picture segment location in the reference picture(s). This includes:

1) no use of motion vectors outside of the current video picture segment for motion vector prediction (as in 6.1.1);
2) no use of motion vectors outside of the current video picture segment as remote motion vectors for overlapped block motion compensation when the Advanced Prediction mode is in use (see F.3);
3) no deblocking filter operation across video picture segment boundaries (see J.3);
4) no use of motion vectors which reference data outside the current video picture segment unless the Unrestricted Motion Vector mode (see Annex D), the Advanced Prediction mode (see Annex F), the Deblocking Filter mode (see Annex J) or the Temporal, SNR, and Spatial Scalability mode (see Annex O) are in use; in which case the borders of the current video picture segment in the prior picture are extrapolated as described in Annex D to form predictions of the pixels which reference the out-of-bounds region;

5) no bilinear interpolation across the boundaries of the ¼-size or ½-size region corresponding to the current video picture segment for upward prediction in spatial scalability EI- and EP-pictures (as defined in Annex O);

6) when the Reduced-Resolution Update mode (see Annex Q) is in use, no block boundary filter operation across video picture segment boundaries;

7) no use of the Reference Picture Resampling mode with the Independent Segment Decoding mode.

R.3 Constraints on usage

Certain restrictions are placed on the use of other aspects of the video coding syntax when the Independent Segment Decoding mode is in use. These restrictions are made to prevent two pathological cases which otherwise would make operation of the Independent Segment Decoding mode difficult.

R.3.1 Constraint on segment shapes

In the use of the Slice Structured mode (Annex K) without the use of the Rectangular Slice submode (see K.1), there can arise cases in which the shape of a video picture segment may be non-convex (having "inside corners", or even comprising two distinct and separated regions of the picture).

Therefore, the Independent Segment Decoding mode shall not be used with the Slice Structured mode without the simultaneous use of the Rectangular Slice submode of the Slice Structured mode (see Annex K). This constraint is mandated to prevent the need for difficult special-case treatment in order to determine how and when to perform extrapolation of each video picture segment.

R.3.2 Constraint on changes of segment shapes

If the shape of the video picture segments were allowed to change in any way from picture to picture in the bitstream, there could arise cases in which the bitstream would be difficult to decode. This is because in such cases the bitstream content is not sufficient to determine the shape of each video picture segment prior to the possible appearance of motion vectors in the bitstream which require knowledge of the video picture segment shape for proper interpretation.

Therefore, when the Independent Segment Decoding mode is in use, the video picture segmentation for all pictures and frames allowing temporal prediction (i.e. all P-, B-, and EP-pictures and all Improved PB-frames) shall be the same as that used in its temporal reference picture. Also, when the Independent Segment Decoding mode is in use, the video picture segmentation for all EI-pictures shall either be the same or shall differ only by sub-dividing the video picture segmentation used in its reference picture. Also, the Independent Segment Decoding mode shall not be used in any picture or frame which uses reference pictures (all picture types except INTRA) unless the Independent Segment Decoding mode is also used in all of the reference picture(s) for the current picture. As a result of this constraint, the shape of the video picture segments in the Independent Segment Decoding mode shall never change from picture to picture except as changed in I- and EI-pictures (and the manner in which EI-pictures can change segmentation is itself also somewhat constrained).

ANNEX S

Alternative INTER VLC mode

S.1 Introduction

This annex describes an optional Alternative INTER VLC mode of this Recommendation, which improves the efficiency of inter-picture coding when significant changes are evident in the picture. This efficiency improvement is obtained by allowing some VLC codes originally designed for INTRA pictures to be used for some INTER picture coefficients and CBPY data as well. The use of this mode is indicated in the PLUSPTYPE field of the picture header. The capability to use this optional mode is negotiated by external means (for example, Recommendation H.245). The mode contains two syntax alterations, one for the encoding of INTER coefficients and another for the encoding of the INTER CBPY values.

S.2 Alternative INTER VLC for coefficients

The concept behind the design of the INTRA VLC table of Annex I is to use the same codewords as in the original INTER VLC but with a different interpretation of LEVEL and RUN. The INTRA VLC is better suited in cases where there are many and/or large-valued coefficients.

The INTRA VLC is constructed so that codewords have the same value for LAST (0 or 1) in both the INTER and INTRA tables. The INTRA table is therefore produced by "reshuffling" the meaning of the codewords with the same value of LAST. Furthermore, for events with large |LEVEL| the INTRA table uses a codeword which in the INTER table has a large RUN. In INTER blocks having a large number of large-magnitude coefficients, it can sometimes be more efficient to use the INTRA table than the INTER table, and in some such cases the choice of the VLC table can be apparent to the decoder since decoding using the INTER table would result in RUN values so large as to indicate the presence of more than 64 coefficients for a block. Under these circumstances, the INTRA table can be used to improve the efficiency of INTER coding.

S.2.1 Encoder action

The encoder may use the INTRA VLC table for coding an INTER block whenever the decoder can detect its use – in other words, whenever decoding using the INTER VLC table would cause coefficients outside the 64 coefficients of a block to be addressed.

The encoder would normally choose to use the INTRA VLC table for coding an INTER block only when the above condition is satisfied and also when the INTRA VLC usage results in fewer bits than the INTER VLC for the same coefficient values. This will often be the case when there are many large coefficients, due to the way the INTRA VLC was produced (since the ordinary INTER VLC table contains long run-lengths for the same codewords in which the INTRA VLC table contains large coefficient amplitudes).

S.2.2 Decoder action

The decoding process is as follows:

1) The decoder first receives all coefficient codes of a block.
2) The codewords are then interpreted assuming that INTER VLC is used. If the addressing of coefficients stays inside the 64 coefficients of a block, the VLC decoding is finished.
3) If coefficients outside the block are addressed, the codewords shall be interpreted according to the INTRA VLC.

S.3 Alternative INTER VLC for CBPY

The INTER CBPY codewords (Table 13) are designed with the assumption that there are more Y blocks with all zero coefficients than there are with at least one non-zero coefficient. When both C_B and C_R blocks have at least one non-zero coefficient, i.e. $CBPC_5 = CBPC_6 = 1$, this assumption no longer holds. For this reason, when the Alternative INTER VLC mode is in use, the CBPY codewords as defined in Table 13 for INTRA macroblocks shall also be used for INTER macroblocks whenever $CBPC_5 = CBPC_6 = 1$.

ANNEX T

Modified Quantization mode

T.1 Introduction

This annex describes an optional Modified Quantization mode of this Recommendation, which modifies quantizer operation. The use of this mode is indicated in the PLUSPTYPE field of the picture header. The capability to use this optional mode is negotiated by external means (for example, Recommendation H.245).

This mode includes four key features:

1) The bit-rate control ability for encoding is improved by altering the syntax for the DQUANT field.
2) Chrominance fidelity is improved by specifying a smaller step size for chrominance than that for luminance data.
3) The range of representable coefficient values is extended to allow the representation of any possible true coefficient value to within the accuracy allowed by the quantization step size.
4) The range of quantized coefficient levels is restricted to those which can reasonably occur, to improve the detectability of errors and minimize decoding complexity.

T.2 Modified DQUANT Update

This mode modifies the semantics of the DQUANT field. With this mode, it is possible to use DQUANT either to modify QUANT by plus or minus a small amount, or to signal any specific new value for QUANT. The size of the small amount of modification depends on the current value of QUANT. By use of this mode, more flexible control of the quantizer step size can be specified in the DQUANT field.

The codeword for DQUANT in this mode is no longer a two-bit fixed length field. It is a variable length field which can either be two bits or six bits in length. Whether it is two or six bits depends on the first bit of the code. The description below is therefore split into two sections, depending on the first bit.

T.2.1 Small-step QUANT alteration

When the first bit of the DQUANT field is 1, only one additional bit is sent in DQUANT. The single additional bit is used to modify QUANT by a differential value. The change in the value of QUANT is dependent on the second bit of DQUANT and on the prior value of QUANT, as shown in Table T.1.

Example: If the previous value of QUANT is 29 and DQUANT is signalled with the codeword "11", then the differential value is +2, and thus the resulting new QUANT value is 31.

Table T.1/H.263 – Semantics of small-step QUANT alteration

	Change of QUANT	
Prior QUANT	**DQUANT = 10**	**DQUANT = 11**
1	+2	+1
2-10	–1	+1
11-20	–2	+2
21-28	–3	+3
29	–3	+2
30	–3	+1
31	–3	–5

T.2.2 Arbitrary QUANT selection

When the first bit of the DQUANT field is 0, five additional bits are sent in DQUANT. The following five bits represent a new QUANT as defined in 5.1.19.

Example: Regardless of the current value of QUANT, if DQUANT is signalled with the code word '001111', then the new value of QUANT is 15.

T.3 Altered quantization step size for chrominance coefficients

When the modified quantization mode is in use, the quantization parameter of the chrominance coefficients is different from the quantization parameter of the luminance. The luminance quantization parameter is signaled in the bitstream. It is called QUANT. When this mode is in use, a different quantization parameter termed QUANT_C is used for the inverse quantization of chrominance coefficients. The relation between QUANT and QUANT_C is given in Table T.2. If the Deblocking Filter mode (see Annex J) is in use, QUANT_C shall also be used for the application of the deblocking filter to the chrominance data. Whenever QUANT is discussed herein in any other context, it shall mean the quantization step size of the luminance.

Table T.2/H.263 – Relationship between QUANT and QUANT_C

Range of QUANT	**Value of QUANT_C**
1-6	QUANT_C = QUANT
7-9	QUANT_C = QUANT – 1
10-11	9
12-13	10
14-15	11
16-18	12
19-21	13
22-26	14
27-31	15

T.4 Modified coefficient range

When the Modified Quantization mode is in use, quantized DCT coefficients having quantization level magnitudes greater than 127 can be represented in the bitstream. This has two advantages:

1) Encoder performance is improved by allowing the true full range of possible coefficient values to be represented.

2) Encoder complexity is reduced by eliminating the need to increase the quantization step size upon encountering certain large coefficient values which would otherwise be unrepresentable.

It is possible that the correct value of a DCT coefficient prior to quantization in the encoder will have a magnitude as high as 2040. Thus, a range of −127 to +127 for LEVEL is insufficient to cover the entire range of possible coefficient values whenever the quantization parameter QUANT or QUANT_C is less than 8. The expanded coefficient range broadens the range of LEVEL to allow any true coefficient value to be more properly encoded.

When the Modified Quantization mode is in use, the meaning of the LEVEL field following an ESCAPE code (0000 011, as per 5.4.2) is altered. In this mode, rather than being forbidden, the bit sequence 1000 0000 is used to represent an EXTENDED-ESCAPE code. An AC coefficient of magnitude greater than 127 is represented by sending an EXTENDED-ESCAPE code, immediately followed by a fixed-length EXTENDED-LEVEL field of 11 bits. An extended coefficient value is encoded into the EXTENDED-LEVEL field by taking the least significant 11 bits of the two's-complement binary representation of LEVEL and cyclically rotating them to the right by 5-bit positions. This rotation is necessary to prevent start code emulation. The cyclic rotation is illustrated in Figure T.1.

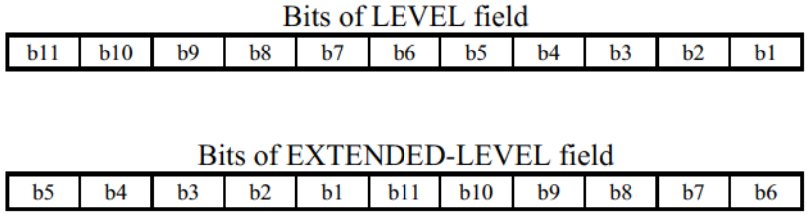


Figure T.1/H.263 – Cyclic rotation of coefficient representation

T.5 Usage restrictions

When the Modified Quantization mode is in use, certain restrictions are placed on the encoded coefficient values. This has several benefits:

1) The detectability of bit errors is improved by prohibiting certain unreasonable coefficient values, thus allowing these values to be recognized as bit errors by the decoder; and

2) Decoder complexity is reduced by reducing the wordlength necessary for inverse quantization prior to clipping.

3) Start-code emulation is prevented for coefficients encoded using the EXTENDED-ESCAPE mechanism described in T.4.

These restrictions are as follows. When the Modified Quantization mode is in use:

1) For any coefficient, the reconstruction level magnitude |REC| produced by the inverse quantization process described in 6.2.1 using the current value of QUANT or QUANT_C as appropriate, and the encoded value of LEVEL, shall be less than 4096. This additional restriction applies to all coefficients, regardless of whether the coefficient is sent using the EXTENDED-ESCAPE mechanism or not.

2) The bitstream shall not use either the normal ESCAPE code or the EXTENDED-ESCAPE code to encode a combination of LAST, RUN, and LEVEL for which there exists a codeword entry in the applicable VLC table, which is either Table 16 (see 5.4.2) or Table I.2 (see I.3).

3) The EXTENDED-ESCAPE code shall be used only when the quantization parameter for the coefficient (QUANT or QUANT_C) is less than eight (8).

4) The EXTENDED-ESCAPE code shall be used only when it is followed by an EXTENDED-LEVEL field representing a value of LEVEL which is outside of the range −127 to +127.

APPENDIX I

Error tracking

I.1 Introduction

This appendix describes a method to recover efficiently after transmission errors if erroneous MBs are reported via a feedback channel to the encoder. The capability of sending and processing feedback information is signalled via external means (for example, by Recommendation H.245). Furthermore, format and content of the feedback message are defined externally (for example, by Recommendation H.245).

I.2 Error tracking

Because INTRA coding stops temporal error propagation, it should be used for macroblocks which are severely affected by transmission errors. This requires that the location and extent of image artefacts can be made available to the encoder. The following algorithm provides an estimated error distribution based on feedback information received by the encoder. It considers spatial error propagation caused by motion-compensated prediction as well as the delay until the reception of the feedback message. The algorithm illustrates one possible approach to evaluate feedback messages for spatio-temporal error tracking. Other algorithms are possible.

Assume N macroblocks within each frame enumerated mb = 1...N from top-left to bottom-right. Let $\{n_{err}, mb_{first}, mb_{last}\}$ be the feedback message to the encoder, where $mb_{first} \leq mb \leq mb_{last}$ indicates a set of erroneous macroblocks in frame n_{err}.

To evaluate the feedback message, the encoder must continuously record information during the encoding of each frame. First, the initial error $E_0(mb, n)$ that would be introduced by the loss of macroblock mb in frame n needs to be stored. Assuming a simple error concealment where erroneous macroblocks are treated as not coded, $E_0(mb, n)$ is computed as the Summed Absolute Difference (SAD) of macroblock mb in frame n and n − 1. Second, the number of pixels transferred from macroblock mb_{source} in frame n − 1 to macroblock mb_{dest} in frame n is stored in dependencies $d(mb_{source}, mb_{dest}, n)$. These dependencies are derived from the motion vectors.

Assume that a feedback message arrives before frame n_{next} is encoded, such that $n_{next} > n_{err}$. Then, the estimated error E(mb, n_{err}) in macroblock mb and frame n_{err} is initialized as:

$$E(mb, n_{err}) = \begin{cases} E_0\,(mb, n_{err}) & \text{for } mb_{first} \le mb \le mb_{last} \\ 0 & \text{else} \end{cases}$$

For subsequent frames n, with $n_{err} < n < n_{next}$, the error may be estimated as:

$$E(mb, n) = \sum_{i=1}^{N} E(i,\ n-1)\ \frac{d(i,\ mb,\ n)}{256}$$

where a uniformly distributed error in each macroblock is assumed after each iteration.

The estimated error E(mb, $n_{next} - 1$) is incorporated into the mode decision of the next frame. For example, macroblock mb is coded in INTRA mode, if E(mb, $n_{next} - 1$) exceeds a threshold.

In practice, error tracking information will only be stored for the latest M frames. Then, if $n_{err} < n_{next} - M$, no error-tracking information is available and the encoder must take special action. For example, the next frame may be coded in INTRA mode. However, other procedures are possible and may be more effective.

APPENDIX II

Recommended Optional Enhancement

II.1 Introduction

With the variety of optional modes available in this Recommendation (H.263 Version 2), it is crucial that several preferred mode combinations for operation be defined, so that option-enhanced terminals will have a high probability of connecting to each other using some syntax better than the "baseline." This appendix contains a list of preferred mode combinations, structured into three "levels" of support. Each level includes support for the levels below it, creating an "onion peel" structure of capabilities. The primary objective of this appendix is to describe the order in which modes should be supported in decoders, rather than to enforce a particular small set of mode combinations upon encoders.

In determining which modes would be placed into this level structure, the primary criteria used were performance-related: improvement in subjective quality, impact on delay, and impact on complexity (this includes computational burden, data dependencies, and ease of implementation). However, because this appendix seeks to address a wide variety of applications and transport layers, features addressing error resilience and ease of packetization were also incorporated into the same level structure. In keeping with these objectives, the levels of support described in this appendix are intended to be *universal*, so that the means of transport will not be an issue and the need for application-specific "profiles" will be minimized. This universality is seen as a key to promoting good interoperability across different types of terminals and networks. To maintain universality, several optional features have not been included in the level structure described herein. The absence of these features is not meant to be a condemnation of their usage; instead, their absence reflects only a collective opinion about the likelihood of widespread adoption of these features across a full spectrum of terminals, networks, and applications.

II.2 Levels of preferred mode support

Each optional mode described in this appendix has been categorized into one of three levels of support. While the ratio of performance improvement to computational complexity was a major criterion used in this categorization, utility for packetization and error resilience were also considered. Decoder support for a given level implies support for all lower levels. Furthermore, decoder support for a given level implies support for all subset combinations of the constituent modes of that level and all lower levels. This last requirement exists so that the limitations placed upon an encoder's choice of mode combinations is minimized. This is in keeping with the primary objective of this appendix, which is to describe the order in which modes should be supported at the decoder, rather than to enforce a particular small set of mode combinations upon the encoder.

Backward compatibility is also an important issue to consider. For this reason, a preferred mode combination including only the Advanced Prediction mode (Annex F) is also recommended (although this Recommendation falls outside the level structure defined below), since Advanced Prediction is regarded as the most beneficial of the optional modes in the original edition (Version 1) of this Recommendation.

II.2.1 Level 1 preferred modes

The first level of support is composed of the following modes:

1) **Advanced INTRA Coding (Annex I)** – Use of this mode improves the coding efficiency for INTRA macroblocks (whether within INTRA pictures or predictively-coded pictures). The additional computational requirements of this mode are minimal at both the encoder and decoder (as low as a maximum of 8 additions/subtractions per 8 × 8 block in the decoding process plus the use of a different but very similar VLC table in order to obtain a significant improvement in coding efficiency). For these reasons, Advanced INTRA Coding is included at this lowest level of support.

2) **Deblocking Filter (Annex J)** – Because of the significant subjective quality improvement that may be realized with a deblocking filter, these filters are already widely in use as a method of post-processing in video communication terminals. Annex J represents the preferred mode of operation for a deblocking filter because it places the filter within the coding loop. This placement eases the implementation of the filter (by reducing the memory requirement) and somewhat improves the coding performance over a post-procesing implementation. As with the Advanced Prediction mode, this mode also includes the four-motion-vector-per-macroblock feature and picture boundary extrapolation for motion compensation, both of which can further improve coding efficiency. The computational requirements of the deblocking filter are several hundred operatons per coded macroblock, but memory accesses and computational dependencies are uncomplicated. This last point is what makes the Deblocking Filter preferable to Advanced Prediction. Also, the benefits of Advanced Prediction are not as substantial when the Deblocking Filter is used as well. Thus, the Deblocking Filter is included at this lowest level of support while Advanced Prediction is deferred to Level 3.

3) **Supplemental Enhancement Information (Full-Frame Freeze Only) (Annex L, see L.4)** – The full-frame freeze is very simple to implement, requiring only that the decoder be able to stop the transfer of data from its output buffer to the video display. This sub-mode is useful for preventing the display of low-fidelity pictures while the encoder is building up a higher fidelity picture.

4) **Modified Quantization (Annex T)** – This mode includes an extended DCT coefficient range, modified DQUANT syntax, and a modified step size for chrominance. The first two features allow for more flexibility at the encoder and may actually decrease the encoder's computational load (by eliminating the need re-encode macroblocks when coefficient level saturation occurs). The third feature noticeably improves chrominance fidelity, typically with little added bit-rate cost and with virtually no increase in computation. At the decoder, the only significant computational burden is the ability to parse several new bitstream symbols.

II.2.2 Level 2 preferred modes

The second level of support is composed of the following modes:

1) **Unrestricted Motion Vectors (With UUI = "1" Sufficient) (Annex D)** – Annex D has two primary features:

   a) picture boundary extrapolation; and

   b) longer motion vector support.

   The first is already supported by the inclusion of Annex J in the first level of support. The longer motion vector support can provide a significant improvement in coding efficiency, especially for large picture sizes, rapid motion, camera movement, and low picture rates. When used with PLUSPTYPE present, this mode also allows for longer motion vector differences, which can significantly simplify encoder operation. The longer motion vectors do present a potential problem for the decoder in terms of memory access, but frame-size-dependent limits on the maximum motion vector size prevent this problem from becoming an appreciable obstacle to implementation.

2) **Slice Structured Mode (Annex K)** – For compatibility with packet-based transport layers, the Slice Structured mode is included here. All submodes of Annex K are to be supported, including the Rectangular Slice submode and the Arbitrary Slice Ordering submode. The additional computational burden imposed by the Slice Structured mode is minimal, limited primarily to bitstream generation and parsing.

3) **Reference Picture Resampling (Implicit Factor-of-4 Mode Only) (Annex P)** – The implicit factor-of-4 mode of Reference Picture Resampling allows for automatic reference picture resampling only when the size of the new frame is changed, as indicated in the picture header. No bitstream overhead is required for this mode of operation. Predictive dynamic resolution changes allow an encoder to make intelligent trade-offs between temporal and spatial resolution. Furthermore, this simplest mode of operation for Annex P (factor of 4 upsampling or downsampling only) adds only a modest amount of computational complexity to both the encoder or decoder, since the factor of 4 case uses a simple fixed FIR filter (requiring roughly 4 operations per pixel, at most).

II.2.3 Level 3 preferred modes

The third and final level of support is composed of the following modes:

1) **Advanced Prediction (Annex F)** – From a coding efficiency standpoint, this mode is the most important of the modes available in the prior version (Version 1) of this Recommendation. It includes overlapped block motion compensation, the four-motion-vector-per-macroblock feature, and it allows for motion vectors to point outside of the picture boundaries. The use of Advanced Prediction results in significant improvements in both subjective and objective performance. It does, however, require an appreciable increase in computation. More importantly, the data dependencies introduced by Advanced Prediction result in a complicated order of processing at the decoder. Coupled with the fact

that the quality improvements are not as obvious when the Deblocking Filter is also in use, these complexity considerations are the reason this mode is part of the third support level, rather than the first. However, since implementations of Recommendation H.263 that were designed prior to the adoption of the other modes in this list might have implemented Advanced Prediction by itself, Advanced Prediction-only operation is also recommended for maximal backward compatibility.

2) **Improved PB-frames (Annex M)** – The Improved PB-frames mode represents a substantial improvement over the basic PB-frames mode (Annex G) and has been demonstrated to significantly improve the coding efficiency for video sequences with a low degree of motion. While the basic version of PB-frames had problems in handling scene cuts and irregular motion, the Improved PB-frames mode may be used effectively at all times. In terms of computational burden, this mode is roughly neutral (especially for macroblock-pipelined architectures). However, because the Improved PB-frames mode can have a negative impact on delay and because it requires an additional frame store to implement (at both the encoder and decoder), it is placed in a higher layer.

3) **Independent Segment Decoding (Annex R)** – While the Independent Segment Decoding mode does introduce a significant amount of complexity (in terms of ease of implementation), it also significantly improves the error robustness of a video decoder, particularly in the presence of packet losses. For this reason, it is likely that many applications in which data can be lost in transmission (whether through packet loss or channel interference) will make use of the Independent Segment Decoding mode, especially in combination with the Slice Structured mode.

4) **Alternate INTER VLC (Annex S)** – This mode has been demonstrated to improve coding efficiency especially for pictures with heavy motion which are coded with high fidelity. This mode uses the same VLC table as the Advanced INTRA Coding mode and is also simple from a computational standpoint (for each block, it requires at most one additional pass over the parsed symbols to decode them into DCT coefficients). However, it is possible that this mode could lead to some difficulty in implementation (especially on highly pipelined architectures), which is why it is not included at the lower levels of support.

II.3 Picture formats and picture clock frequencies

To ensure a high quality level of interoperability, encoders and decoders supporting a large standard picture format (QCIF, CIF, 4CIF, 16CIF) should support all smaller standard picture formats. (As specified elsewhere in this Recommendation, decoders shall support sub-QCIF and QCIF, and encoders shall support sub-QCIF or QCIF.) Decoders should be capable of operation with a smaller picture format at maximum frame rates no lower than the maximum frame rate for which it is capable of operation with a larger standard picture format. For example, a decoder capable of decoding 4CIF pictures at 15 pictures per second [actually 15*(1000/1001) pictures per second] should also be able to decode CIF, QCIF and SQCIF pictures at least at 15 pictures per second.

Encoders and decoders supporting custom picture formats and/or custom picture clock frequencies are recommended to follow the guidelines below:

1) An encoder or decoder which supports a custom picture format should support all standard picture formats equal or smaller in both height and width than those of any supported custom picture format. For example, an encoder supporting a custom picture format of 256 × 256 should support QCIF and SQCIF picture encoding.

2) An encoder or decoder which supports a custom picture clock frequency for a standard picture format should support a higher or equal frame rate (in pictures per second) at the standard picture clock frequency of 30 Hz [actually 30*(1000/1001) Hz] for the same picture format. For example, if a decoder supports QCIF pictures at 12.5 pictures per second with a 25 Hz picture clock, it should at least be able to decode QCIF pictures at 15 pictures per second at 30 Hz. If a custom picture format is used, all supported smaller or equal standard picture formats should be capable of operating at frame rates for the standard picture clock frequency no lower than the maximum frame rate of the custom picture format.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages