

(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 parameters is -4095 to $+4095$. The warping parameters are defined using the resampling vectors by the relations:

$$\begin{aligned} w_x^0 &= 2r_x^0 & w_y^0 &= 2r_y^0 \\ w_x^x &= 2(r_x^x - (H_R - H)) & w_y^x &= 2r_y^x \\ w_x^y &= 2r_x^y & w_y^y &= 2(r_y^y - (V_R - V)) \\ w_x^{xy} &= 2r_x^{xy} & w_y^{xy} &= 2r_y^{xy} \end{aligned}$$

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	<i>color</i>
0 1	<i>black</i>
1 0	<i>gray</i>
1 1	<i>clip</i>

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:

$$\begin{aligned}
 u_x^{00} &= 16w_x^0 & u_y^{00} &= 16w_y^0 \\
 u_x^{H0} &= 16(w_x^0 + w_x^x + 2(H_R - H)) & u_y^{H0} &= 16(w_y^0 + w_y^x) \\
 u_x^{0V} &= 16(w_x^0 + w_x^y) & u_y^{0V} &= 16(w_y^0 + w_y^y + 2(V_R - V)) \\
 u_x^{HV} &= 16(w_x^0 + w_x^x + w_x^y + w_x^{xy} + 2(H_R - H)) & u_y^{HV} &= 16(w_y^0 + w_y^x + w_y^y + w_y^{xy} + 2(V_R - V))
 \end{aligned}$$

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:

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

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:

$$\begin{aligned}
 u_x^{LT} &= u_x^{00} & u_y^{LT} &= u_y^{00} \\
 u_x^{RT} &= ((H - H')u_x^{00} + H'u_x^{H0}) // H & u_y^{RT} &= ((H - H')u_y^{00} + H'u_y^{H0}) // H \\
 u_x^{LB} &= ((V - V')u_x^{00} + V'u_x^{0V}) // V & u_y^{LB} &= ((V - V')u_y^{00} + V'u_y^{0V}) // V \\
 u_x^{RB} &= ((V - V')((H - H')u_x^{00} + H'u_x^{H0}) + V'((H - H')u_x^{0V} + H'u_x^{HV})) // (HV) \\
 u_y^{RB} &= ((V - V')((H - H')u_y^{00} + H'u_y^{H0}) + V'((H - H')u_y^{0V} + H'u_y^{HV})) // (HV)
 \end{aligned}$$

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:

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

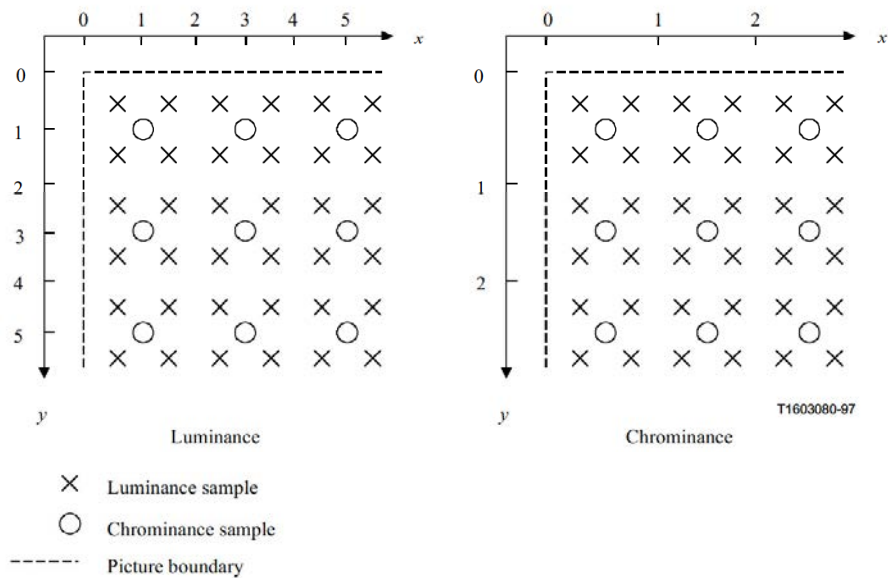


Figure P.3/H.263 – Coordinate systems for luminance and chrominance fields

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:

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

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

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:

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

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

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

$$\varnothing_x = I_R(i, j) - (I_R(i, j) /// P)P \quad \varnothing_y = J_R(i, j) - (J_R(i, j) /// P)P$$

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:

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

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

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 than 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:

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

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 = (V_Q - V_Q') * (T_T /// A) + V_Q' * (T_B /// A) + ((V_Q - V_Q') * (T_T % A) + V_Q' * (T_B % A)) /// A;
r = ((V_Q - V_Q') * (T_T % A) + V_Q' * (T_B % A)) % A;
if (q < 0)
    u_x^{RB} = q + (r + (A-1) / 2) / A;
else
    u_x^{RB} = 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(x_min, x, x_max) {
    if (x < x_min) {
        return x_min;
    } else if (x > x_max) {
        return x_max;
    }
```

```

        } 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, $\mathcal{O}_x$ and $\mathcal{O}_y$, are quantized in the range 0, ..., *P*-1 (inclusive).

```

filter(x0, y0,  $\mathcal{O}_x$ ,  $\mathcal{O}_y$ ) {
    return [(P- $\mathcal{O}_y$ )*((P- $\mathcal{O}_x$ )*prior_sample(x0, y0)+ $\mathcal{O}_x$ *prior_sample(x0+1, y0))+
             $\mathcal{O}_y$ *((P- $\mathcal{O}_x$ )*prior_sample(x0, y0+1)+ $\mathcal{O}_x$ *prior_sample(x0+1, y0+1))+
            P2 / 2-1+RCRPR] / P2;
}

```

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 $H' = 2^n$, 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:

```

axi = D * P + 2 * (uxR - uxL);
ayi = 2 * (uyR - uyL);
ax = uxL * S * 2m + (uxR - uxL); + D / 2;
ay = j * D * P + uyL * S * 2m + (uyR - uyL); + D / 2;
for (i = 0; i < S * H / 2; i++) {
    IR = ax /// D;
    JR = ay /// D;
    iR = IR /// P;
}

```

```

jR = JR /// P;
Ox = IR - (iR * P);
Oy = JR - (jR * P);
new_ref[i, j] = filter(iR, jR, Ox, Oy);
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;
Ox = IR - (iR * P);
Oy = 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;
Ox = IR & 15;
Oy = 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.

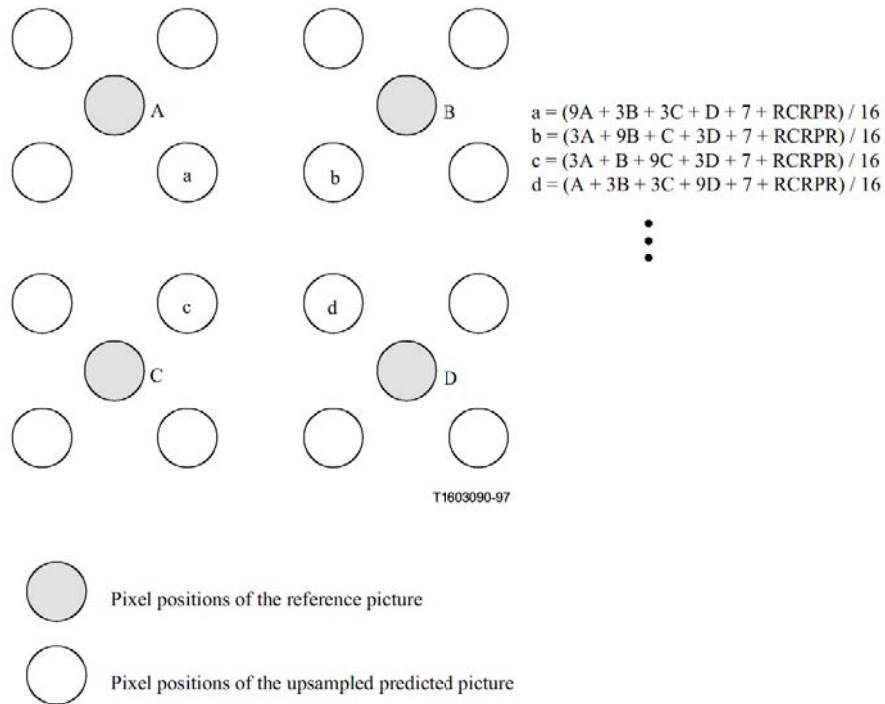


Figure P.4/H.263 – Factor-of-4 upsampling for pixels inside the picture

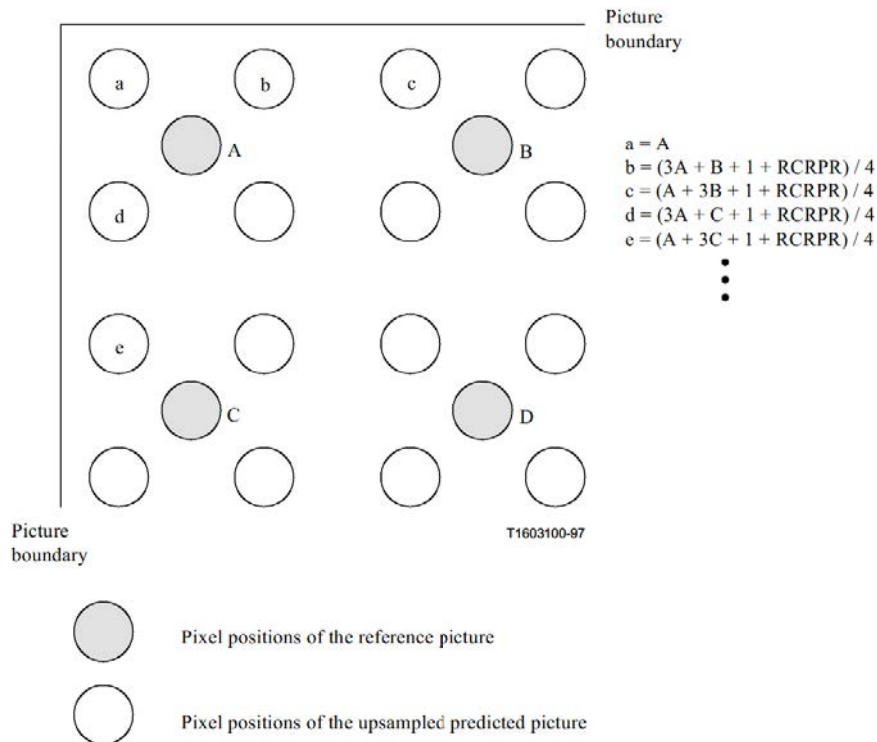


Figure P.5/H.263 – Factor-of-4 upsampling for pixels at the picture boundary (fill mode = *clip*)

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.

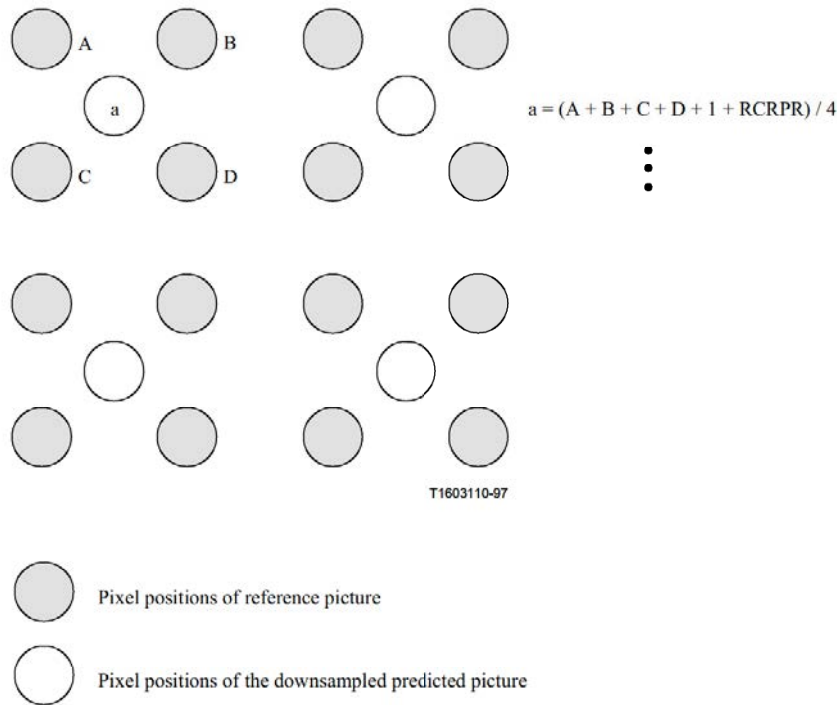


Figure P.6/H.263 – Factor-of-4 downsampling

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.

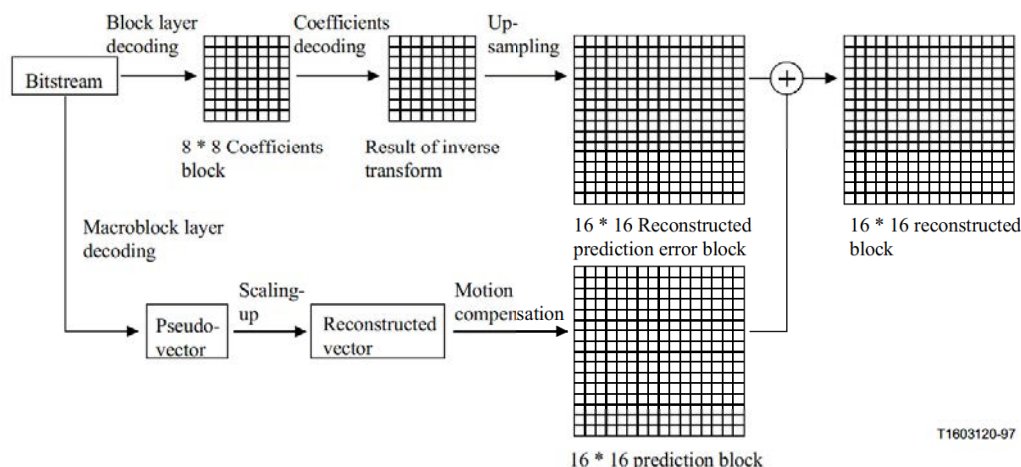


Figure Q.1/H.263 – Block diagram of block decoding in Reduced-Resolution Update mode

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_{2-4}). 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.

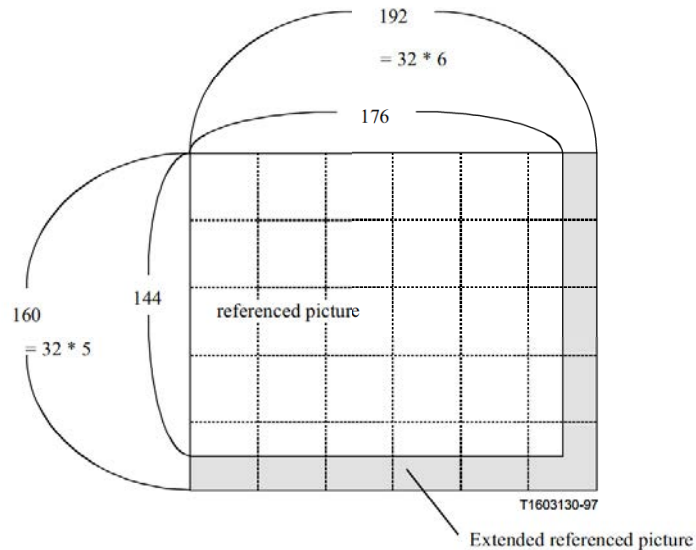


Figure Q.2/H.263 – Extension of referenced picture for QCIF picture size

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 $[-\text{limit}, \text{limit}]$ as defined in D.2 applies to the pseudo-motion vectors and translates into $[-(2 * \text{limit} - 0.5), 2 * \text{limit} - 1.5]$ 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.

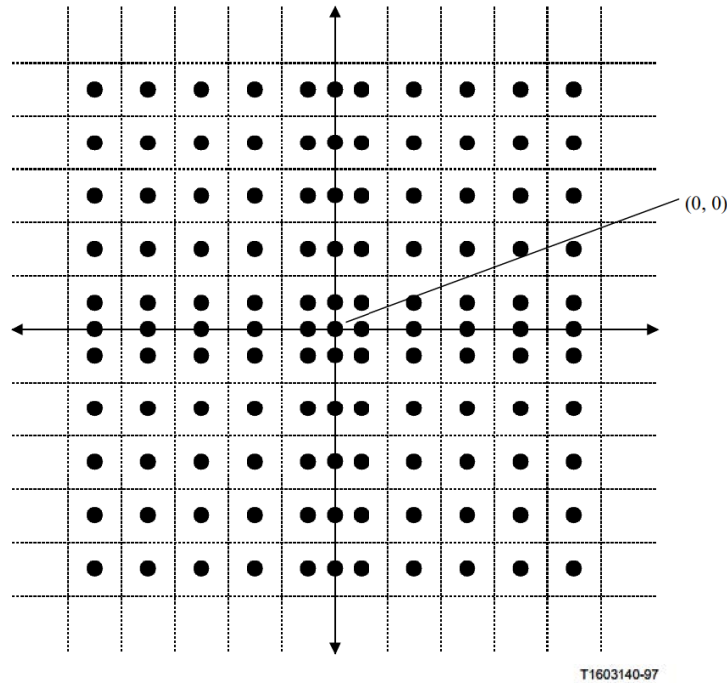


Figure Q.3/H.263 – Reconstruction of motion vector

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_{2-4} as follows:

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

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

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

"/" indicates floating-point division (without loss of accuracy). The prediction vector component P_C is defined as the median value of the vector components MV_1 , MV_2 and MV_3 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_{2-4}) 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_{2-4}) 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 MV_C is obtained from pseudo- MV_C in the following formula:
- $$MV_C = 0 \quad \text{if pseudo-}MV_C = 0$$
- $$MV_C = \text{sign}(\text{pseudo-}MV_C) * (2.0 * |\text{pseudo-}MV_C| - 0.5) \quad \text{if pseudo-}MV_C \neq 0$$
- 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 MV_F and/or 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_C , 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 MV_{DB} 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	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.

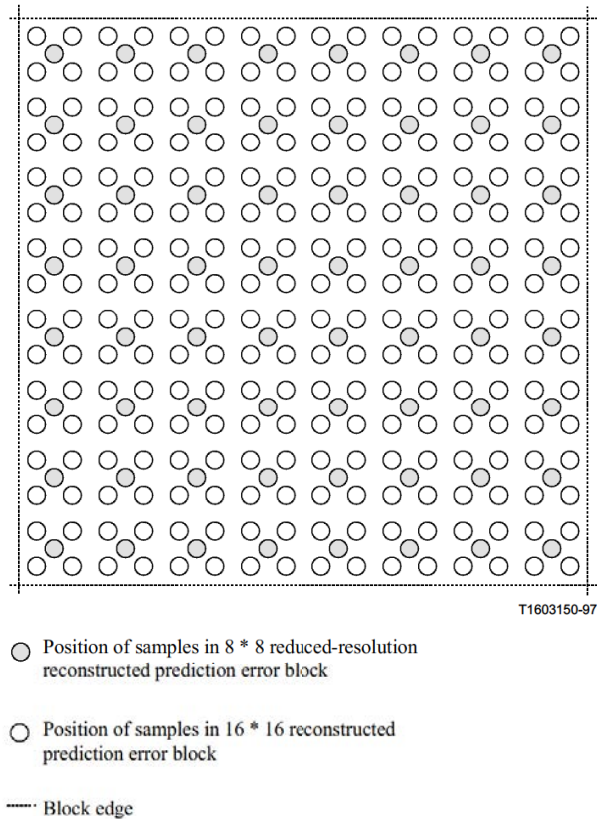


Figure Q.7/H.263 – Positioning of samples in 8×8 reduced-resolution reconstructed prediction error block and 16×16 reconstructed prediction error block

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.

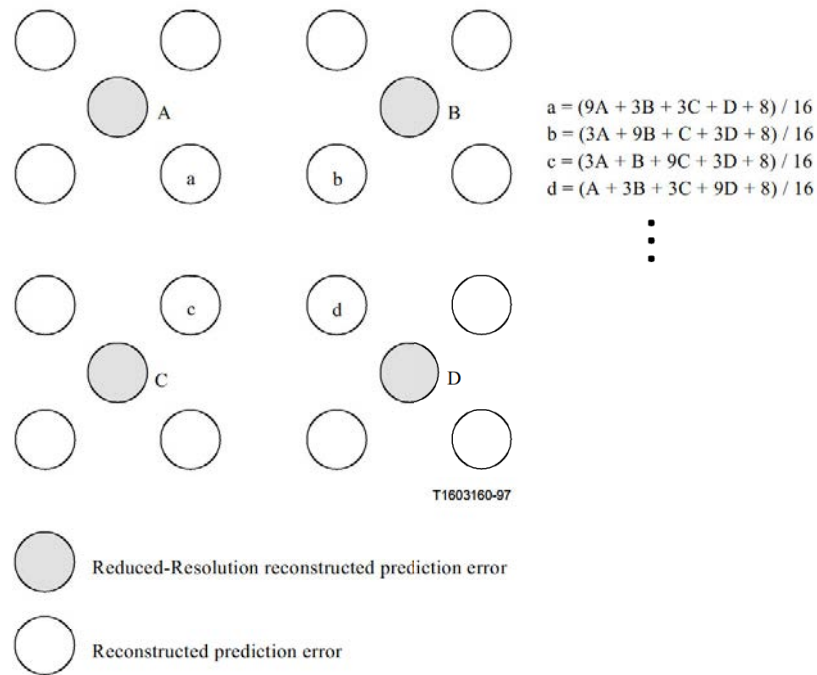
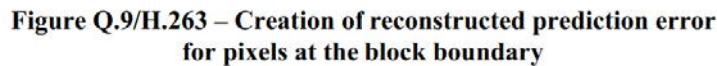


Figure Q.8/H.263 – Creation of reconstructed prediction error for pixels inside block

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



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.

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

141

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.

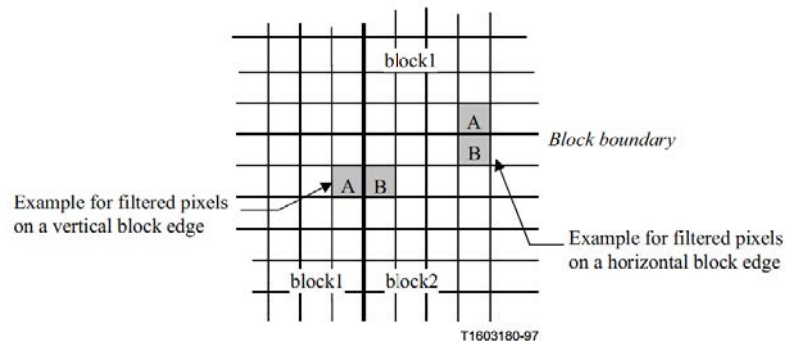


Figure Q.10/H.263 – Default block boundary filter

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 ($COD=0 \parallel MB\text{-type} = INTRA$); or
- block2 belongs to a coded macroblock ($COD=0 \parallel MB\text{-type} = INTRA$).

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:

$$B1 = \text{clip}(B + d1)$$

$$C1 = \text{clip}(C - d1)$$

$$A1 = A - d2$$

$$D1 = D + d2$$

$$d1 = (A - 4B + 4C - D) / 8$$

$$d2 = \text{clipd1}((A - D) / 4, d1/2)$$

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 $\frac{1}{4}$ -size or $\frac{1}{2}$ -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

Prior QUANT	Change of 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.

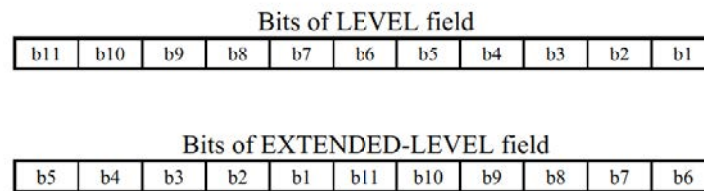


Figure T.1/H.263 – Cyclic rotation of coefficient representation

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 $|\text{REC}|$ 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 $mb = 1 \dots N$ 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 $n_{\text{next}} > n_{\text{err}}$. Then, the estimated error $E(\text{mb}, n_{\text{err}})$ in macroblock mb and frame n_{err} is initialized as:

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

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

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

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

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

In practice, error tracking information will only be stored for the latest M frames. Then, if $n_{\text{err}} < n_{\text{next}} - M$, 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-processing 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 operations 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 UI = "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 $15 \times (1000/1001)$ 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 $30 \times (1000/1001)$ 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

APPENDIX TAMU01



Libraries Catalog

All Fields

Find

Advanced

[Video coding for low bit rate...](#)

* Cite this

Text this

Email this

Print

Export Record

Permanent link



Video coding for low bit rate communication.

Corporate Author:

[International Telecommunication Union. Telecommunication Standardization Sector](#)

Format:

Book

Language:

English

Published:

Geneva, Switzerland : ITU, [1998]

Subjects:

[Digital video > Standards.](#)
[Video compression > Standards.](#)
[Coding theory.](#)

Holdings

Description

Similar Items

Staff View

MARC

LEADER	00000cam a2200000la 4500		
001	in00002182238		
005	20151104063015.0		
008	000511s1998 sz a 000 0 eng d		
035			a (OCoLC)ocm44027412
040			a IUP c IUP d TXA d UtOrBLW
049			a TXAM
090			a TK6680.5 b .I584 1998
110	2		a International Telecommunication Union. b Telecommunication Standardization Sector.
245	1	0	a Video coding for low bit rate communication.
246	1		i At head of title: a ITU-T recommendation H.263
246	1	7	a Recommendation H.263
264		1	a Geneva, Switzerland : b ITU, c [1998]
264		4	c ©1998
300			a x, 154 pages : b illustrations : c 30 cm.
336			a text b txt 2 rdacontent
337			a unmediated b n 2 rdamedia
338			a volume b nc 2 rdacarrier
500			a *Series H: Audiovisual and multimedia

			systems, infrastructure of audiovisual services - coding of moving video"--Cover.
500			ja "(02/98)"--Cover.
650		0	ja Digital video ix Standards.
650		0	ja Video compression ix Standards.
650		0	ja Coding theory.
994			ja C0 jb TXA
948			ja cataloged jb h jc 2006/8/10 jd c je biosas jf 9:57:36 am
999			ja MARS
999	f	f	js d2844665-7e19-3368-9f51-1ac52090f389 ji e0e211bd-11e1-3edc-bbaf-491a8ec98346 jt 0
952	f	f	jp normal ja Texas A&M University jb College Station jc Sterling C. Evans Library jd Evans: Library Stacks jt 0 je TK6680.5 .I584 1998 jh Library of Congress classification ji unmediated -- volume jm A14834696717
998	f	f	ja TK6680.5 .I584 1998 jt 0 jl Evans: Library Stacks