$$(F(i, j), G(i, j)) = (i_0' + (((-r\,i_0' + i_1'')\,I + (-r\,i_0' + i_2'')\,2^{\alpha-\beta}\,J + 2^{\alpha+\rho-1}) >> (\alpha+\rho)),$$
$$j_0' + (((-r\,j_0' + j_1'')\,I + (-r\,j_0' + j_2'')\,2^{\alpha-\beta}\,J + 2^{\alpha+\rho-1}) >> (\alpha+\rho))),$$
$$(F_c(i_c, j_c), G_c(i_c, j_c)) = (((-r\,i_0' + i_1'')\,I_c + (-r\,i_0' + i_2'')\,2^{\alpha-\beta}\,J_c + 2W'r\,i_0' - 16W' + 2^{\alpha+\rho+1}) >> (\alpha+\rho+2),$$
$$((-r\,j_0' + j_1'')\,I_c + (-r\,j_0' + j_2'')\,2^{\alpha-\beta}\,J_c + 2W'r\,j_0' - 16W' + 2^{\alpha+\rho+1}) >> (\alpha+\rho+2)),$$

for 4:2:0 VOPs,

$$(F_c(i_c, j_c), G_c(i_c, j_c)) = (((-r\,i_0' + i_1'')\,I_c + (-r\,i_0' + i_2'')\,2^{\alpha-\beta}\,J_c + 2W'r\,i_0' - 16W' + 2^{\alpha+\rho+1}) >> (\alpha+\rho+2),$$
$$j_0' + (((-r\,j_0' + j_1'')\,I_c + (-r\,j_0' + j_2'')\,2^{\alpha-\beta}\,J_c + 2^{\alpha+\rho-1}) >> (\alpha+\rho))),$$

for 4:2:2 VOPs,

$$(F_c(i_c, j_c), G_c(i_c, j_c)) = (i_0' + (((-r\,i_0' + i_1'')\,I_c + (-r\,i_0' + i_2'')\,2^{\alpha-\beta}\,J_c + 2^{\alpha+\rho-1}) >> (\alpha+\rho)),$$
$$j_0' + (((-r\,j_0' + j_1'')\,I_c + (-r\,j_0' + j_2'')\,2^{\alpha-\beta}\,J_c + 2^{\alpha+\rho-1}) >> (\alpha+\rho))),$$

for 4:4:4 VOPs.

When no_of_sprite_warping_point == 4,

$$(F(i, j), G(i, j)) = ((a\,I + b\,J + c) /// (g\,I + h\,J + D\,W\,H),$$
$$(d\,I + e\,J + f) /// (g\,I + h\,J + D\,W\,H)),$$
$$(F_c(i_c, j_c), G_c(i_c, j_c)) = ((2\,a\,I_c + 2\,b\,J_c + 4\,c - (g\,I_c + h\,J_c + 2\,D\,W\,H)\,s) /// (4\,g\,I_c + 4\,h\,J_c + 8\,D\,W\,H),$$
$$(2\,d\,I_c + 2\,e\,J_c + 4\,f - (g\,I_c + h\,J_c + 2\,D\,W\,H)\,s) /// (4\,g\,I_c + 4\,h\,J_c + 8\,D\,W\,H)),$$

for 4:2:0 VOPs,

$$(F_c(i_c, j_c), G_c(i_c, j_c)) = ((2\,a\,I_c + 2\,b\,J_c + 4\,c - (g\,I_c + h\,J_c + 2\,D\,W\,H)\,s) /// (4\,g\,I_c + 4\,h\,J_c + 8\,D\,W\,H),$$
$$(d\,I_c + e\,J_c + f) /// (g\,I_c + h\,J_c + D\,W\,H)),$$

for 4:2:2 VOPs,

$$(F_c(i_c, j_c), G_c(i_c, j_c)) = ((a\,I_c + b\,J_c + c) /// (g\,I_c + h\,J_c + D\,W\,H),$$
$$(d\,I_c + e\,J_c + f) /// (g\,I_c + h\,J_c + D\,W\,H)),$$

for 4:4:4 VOPs.

where

$$g = ((i_0' - i_1' - i_2' + i_3')\,(j_2' - j_3') - (i_2' - i_3')\,(j_0' - j_1' - j_2' + j_3'))\,H,$$
$$h = ((i_1' - i_3')\,(j_0' - j_1' - j_2' + j_3') - (i_0' - i_1' - i_2' + i_3')\,(j_1' - j_3'))\,W,$$
$$D = (i_1' - i_3')\,(j_2' - j_3') - (i_2' - i_3')\,(j_1' - j_3'),$$
$$a = D\,(i_1' - i_0')\,H + g\,i_1',$$
$$b = D\,(i_2' - i_0')\,W + h\,i_2',$$
$$c = D\,i_0'\,W\,H,$$
$$d = D\,(j_1' - j_0')\,H + g\,j_1',$$
$$e = D\,(j_2' - j_0')\,W + h\,j_2',$$
$$f = D\,j_0'\,W\,H.$$

A set of parameters that causes the denominator of any of the above equations to be zero for any pixel in an opaque or boundary macroblock is disallowed. The implementor should be aware that a 32 bit register may not be sufficient for representing the denominator or the numerator in the above transform functions for affine and perspective transforms. The usage of a 64 bit floating point representation should be sufficient in such cases.

7.16.9.5 Sample reconstruction

The reconstructed value Y of the luminance sample (i, j) in the currently decoded VOP shall be defined as

$$Y = ((s - r_i)((s - r_i)\,Y_{00} + r_i\,Y_{01}) + r_i\,((s - r_i)\,Y_{10} + r_i\,Y_{11}))\,//\,s^2,$$

© ISO/IEC 2004 – All rights reserved

where Y_{00}, Y_{01}, Y_{10}, Y_{11} represent the sprite luminance sample at ($F(i, j)$////s, $G(i, j)$////s), ($F(i, j)$////s + 1,$G(i, j)$////s), ($F(i, j)$////s, $G(i, j)$////s + 1), and ($F(i, j)$////s + 1,$G(i, j)$////s + 1) respectively, and $r_i = F(i, j) - (F(i, j)$////$s)s$ and $r_j = G(i, j) - (G(i, j)$////$s)s$. Figure 7-121 illustrates this process.

In case any of Y_{00}, Y_{01}, Y_{10} and Y_{11} lies outside the sprite luminance binary mask, it shall be obtained by the padding process as defined in subclause 7.16.7.1.1.

When brightness_change_in_sprite == 1, the final reconstructed luminance sample (i, j) is further computed as Y = Y * (brightness_change_factor + 100)) // 100, clipped to the range of [0, 2^(bits_per_pixel-1)].

Similarly, the reconstructed value C of the chrominance sample (i_c, j_c) in the currently decoded VOP shall be defined as

$$C = ((s - r_j)((s - r_i)\, C_{00} + r_i\, C_{01}) + r_j\, ((s - r_i)\, C_{10} + r_i\, C_{11})) \,//\, s^2,$$

where C_{00}, C_{01}, C_{10}, C_{11} represent the sprite chrominance sample at ($F_c(i_c, j_c)$////s, $G_c(i_c, j_c)$////s), ($F_c(i_c, j_c)$////s + 1, $G_c(i_c, j_c)$////s), ($F_c(i_c, j_c)$////s, $G_c(i_c, j_c)$////s + 1), and ($F_c(i_c, j_c)$////s + 1, $G_c(i_c, j_c)$////s + 1) respectively, and $r_i = F_c(i_c, j_c) - (F_c(i_c, j_c)$////$s)s$ and $r_j = G_c(i_c, j_c) - (G_c(i_c, j_c)$////$s)s$. In case any of C_{00}, C_{01}, C_{10} and C_{11} lies outside the sprite chrominance binary mask, it shall be obtained by the padding process as defined in subclause 7.16.7.1.1.

The same method is used for the reconstruction of grayscale alpha and luminance samples. The reconstructed value A of the grayscale alpha sample (i, j) in the currently decoded VOP shall be defined as

$$A = ((s - r_j)((s - r_i)\, A_{00} + r_i\, A_{01}) + r_j\, ((s - r_i)\, A_{10} + r_i\, A_{11})) \,//\, s^2,$$

where A_{00}, A_{01}, A_{10}, A_{11} represent the sprite grayscale alpha sample at ($F(i, j)$////s, $G(i, j)$////s), ($F(i, j)$////s + 1,$G(i, j)$////s), ($F(i, j)$////s, $G(i, j)$////s + 1), and ($F(i, j)$////s + 1,$G(i, j)$////s + 1) respectively, and $r_i = F(i, j) - (F(i, j)$////$s)s$ and $r_j = G(i, j) - (G(i, j)$////$s)s$. In case any of A_{00}, A_{01}, A_{10} and A_{11} lies outside the sprite luminance binary mask, it shall be obtained by the padding process as defined in subclause 7.16.7.1.1.

The reconstructed values of luminance binary mask samples $BY(i,j)$ shall be computed following the identical process for the luminance samples. However, corresponding binary mask sample values shall be used in place of luminance samples Y_{00}, Y_{01}, Y_{10}, Y_{11}. Assume the binary mask sample opaque is equal to 255 and the binary mask sample transparent is equal to 0. If the computed value is bigger or equal to 128, $BY(i, j)$ is defined as opaque. Otherwise, BY (i, j) is defined as transparent. The chrominance binary mask samples shall be reconstructed by decimating the corresponding 2 x 2 adjacent luminance binary mask samples as specified in subclause 7.16.7.1.1.4.

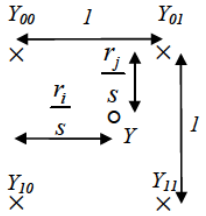


Figure 7-122 — Pixel value interpolation (it is assumed that sprite samples are located on an integer grid)

7.17 The FGS decoding process

This clause specifies the FGS decoding process that the decoder shall perform to recover visual data from the coded bitstream and reconstruct the enhancement VOP. In a typical application of FGS, the bitstream at the input of an FGS decoder is a truncated version of the bitstream at the output of an FGS encoder. It is likely that, at the end of each fgs vop bitstream before the next fgs_vop_start_code, only partial bits of an fgs block are at the input

© ISO/IEC 2004 – All rights reserved

of the decoder due to truncation of the fgs vop bitstream. These partial bits of an fgs block may be discarded. The description of the FGS decoding process in the following sub-sections is for an fgs block that contains all of its bits.

7.17.1 Bit-plane decoding of the absolute values of the DCT coefficients

The bit-plane decoding procedure can be described as follows:

- The absolute values of all the DCT coefficients are initialized to zeros.
- Variable length decoding of (RUN, EOP) symbols is performed. If frequency weighting is used, i.e., **fgs_frequency_weighting_enable** is '1', the choice of VLC tables for decoding depends on the first value of the frequency weighting matrix (shifted bits for the DC component). Table 7-44 indicates this dependency:

Table 7-44 — VLC Table Choices for Frequency Weighting

fw_matrix[0]	VLC tables
0	B.4.1 table_bcp
1	B.4.1 table_bcp
2	B.4.2 table_bcp4fw_m
3	B.4.2 table_bcp4fw_m
4	B.4.3 table_bcp4fw_h
5	B.4.3 table_bcp4fw_h
6	B.4.3 table_bcp4fw_h
7	B.4.3 table_bcp4fw_h

  where table_bpc, table_ bpc4fw_m, and table_bpc4fw_h are given in Annex B.4. Four 2-D VLC tables are used in each set. The first table in each set corresponds to the MSB block-bp. The second table in each set corresponds to the second MSB block-bp. The third table in each set corresponds to the third MSB block-bp. The fourth table in each set corresponds to the fourth MSB block-bp and all the lower block-bps. For the ESCAPE cases, RUN is represented with 6 bits to represent values from 0 to 63 and EOP is a 1-bit flag to indicate whether the end of the block-bp is reached (with value 1) or not (with value 0).

- The (RUN,EOP) symbols are translated into RUN number of consecutive 0's before a 1 and whether to fill 0's to the end of this block-bp. If the ALL-ZERO symbol is decoded, this block-bp contains all 0's. The ALL-ZERO symbol is not in the VLC tables for the MSB block-bp.
- Accumulate the bit to the partial result of each absolute value of the (bit shifted) DCT coefficients at the proper significant position.
- If frequency weighting or selective enhancement is used, the bit shifted DCT coefficient is right-shifted by S bits where S = fw_matrix[i]+ fgs_shifted_bits, and fw_matrix[i] is the i^{th} element of the frequency weighting matrix.
- After decoding all the bits for the fgs vop available to the FGS decoder, partial absolute values of all the DCT coefficients are available.

Note that the MSB block-bp in the above description is defined as the first block-bp of an 8x8 DCT block that contains at least one "1" bit value. The MSB block-bp of an 8x8 DCT block may or may not be in the first vop-bp.

© ISO/IEC 2004 – All rights reserved

7.17.2 Decoding of the signs of the DCT coefficients

A sign bit is decoded from the enhancement layer bitstream immediately after the (RUN, EOP) code corresponding to the MSB of the non-zero DCT coefficient.

7.17.3 Reconstruction of the enhancement DCT coefficients

Let DIFF and SIGN be the absolute value and sign of a DCT coefficient decoded from the enhancement bitstream, and ENH_COEFF be the reconstructed enhancement DCT coefficient.

ENH_COEFF = SIGN*DIFF

7.17.4 Reconstruction of the enhanced VOP

After enhanced DCT coefficients are obtained and inverse zigzag scan is used to convert the enhanced DCT coefficients into 8x8 blocks, an inverse DCT is applied to each 8x8 block. Inverse zigzag scan is performed according to Figure 7-4 (c) in subclause 7.4.2. If **fgs_vop_coding_type** is "I", the reconstructed residue VOP is added to the base-layer VOP pixel by pixel. As a reference to this operation, the base-layer VOP is the one after clipping. If **fgs_vop_coding_type** is "P" or "B", the reconstructed block after the inverse DCT followed by adding the motion compensated prediction block is the enhancement VOP. Motion vector decoding is performed according to subclause 7.6.3. Motion compensation is performed according to the applicable parts of subclause 7.9.1.1, subclause 7.9.1.3.2, and subclause 7.9.1.3.3. The reconstructed enhancement VOP pixels are limited into the value range between 0 and 255 by the clipping unit in the FGS enhancement layer to generate the final enhancement VOP.

8 Visual-Systems Composition Issues

8.1 Temporal Scalability Composition

Background composition is used in forming the background region for objects at the enhancement layer of temporal scalability when the value of both enhancement_type and background_composition is one. This process is useful when the enhancement VOP corresponds to the partial region of the VOP belonging to the reference layer. In this process, the background of a current enhancement VOP is composed using the previous and the next VOPs in display order belonging to the reference layer.

Figure 8-1 shows the background composition for the current frame at the enhancement layer. The dotted line represents the shape of the selected object at the previous VOP in the reference layer (called "forward shape"). As the object moves, its shape at the next VOP in the reference layer is represented by a broken line (called "backward shape").

For the region outside these shapes, the pixel value from the nearest VOP at the reference layer is used for the composed background. For the region occupied only by the forward shape, the pixel value from the next VOP at the reference layer is used for the composed frame. This area is shown as lightly shaded in Figure 8-1. On the other hand, for the region occupied only by the backward shape, pixel values from the previous VOP in the reference layer are used. This is the area shaded dark in Figure 8-1. For the region where the areas enclosed by these shapes overlap, the pixel value is given by padding from the surrounding area. The pixel value which is outside of the overlapped area should be filled before the padding operation.

© ISO/IEC 2004 – All rights reserved

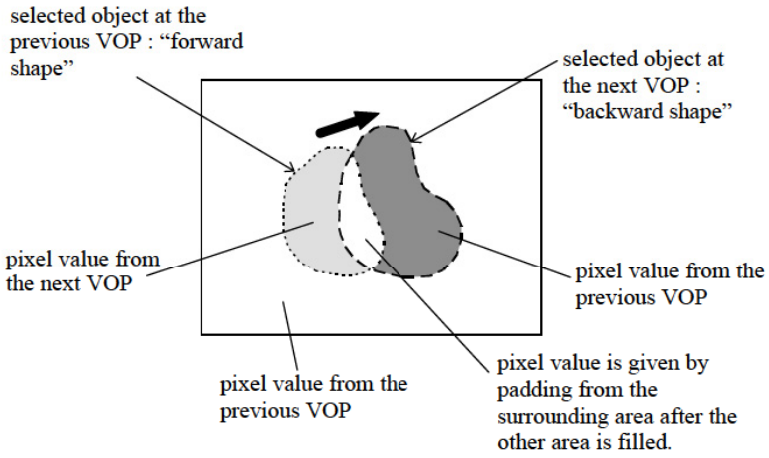


Figure 8-1 — Background composition

The following process is a mathematical description of the background composition method.

```
If s(x,y,ta)=0 and s(x,y,td)=0
    fc(x,y,t) = f(x,y,td)    (|t-ta|>|t-td|)
    fc(x,y,t) = f(x,y,ta)    (otherwise),
if s(x,y,ta)=1 and s(x,y,td)=0
    fc(x,y,t) = f(x,y,td)
if s(x,y,ta)=0 and s(x,y,td)=1
    fc(x,y,t) = f(x,y,ta)
if s(x,y,ta)=1 and s(x,y,td)=1
    The pixel value of fc(x,y,t) is given by repetitive padding from the surrounding area.
```

where

fc composed background
f decoded VOP at the reference layer
s shape information (alpha plane) , 0: transparent, 1: opaque
(x,y) the spatial coordinate
t time of the current VOP
ta time of the previous VOP
td time of the next VOP

Two types of shape information, $s(x, y, ta)$ and $s(x, y, td)$, are necessary for the background composition. $s(x, y, ta)$ is called a "forward shape" and $s(x, y, td)$ is called a "backward shape". If $f(x, y, ta)$ does not exist, the pixel value of $fc(x, y, t)$ for $s(x, y, td) = 0$ is given by $f(x, y, td)$ and the pixel value of $fc(x, y, t)$ for $s(x, y, td) = 1$ is given by repetitive padding from the boundary pixel value. If $f(x, y, td)$ does not exist, pixel value of $fc(x, y, t)$ for $s(x, y, ta) = 0$ is given by $f(x, y, ta)$ and the pixel value of $fc(x, y, t)$ for $s(x, y, ta) = 1$ is given by repetitive padding from the boundary pixel value.

8.2 Sprite Composition

The static sprite technology enables to encode very efficiently video objects which content is expected not to vary in time along a video sequence. For example, it is particularly well suited to represent backgrounds of scenes (decor, landscapes) or logos.

A static sprite (sometimes referred as mosaic in the literature) is a frame containing spatial information for a single object, obtained by gathering information for this object throughout the sequence in which it appears. A static sprite can be a very large frame: it can correspond for instance to a wide angle view of a panorama.

© ISO/IEC 2004 – All rights reserved

The ISO/IEC 14496-2 syntax defines a dedicated coding mode to obtain VOPs from static sprites: the so-called "S-VOPs". S-VOPs with sprite_enable == 'static' are extracted from a static sprite using a warping operation consisting in a global spatial transformation driven by few motion parameters (0,2,4, 6 or 8).

For composition with other VOPs, there are no special rules for S-VOPs with sprite_enable == 'static'. However, it is classical to use S-VOPs with sprite_enable == 'static' as background objects over which "classical" objects are superimposed.

8.3 Mesh Object Composition

A Mesh Object represents the geometry of a sequence of 2D triangular meshes. This data can be used along with separately coded image texture data to render texture-mapped images, e.g., by the composition process as defined in ISO/IEC 14496-1:2001. A Mesh Object stream may be contained in part of a BIFS animation stream, as defined in ISO/IEC 14496-1:2001. In terminals implementing mesh animation functionality using both ISO/IEC 14496-1:2001 and this part of ISO/IEC 14496, the decoded mesh data is used to update the appropriate fields of a BIFS IndexedFaceSet2D node, defined in ISO/IEC 14496-1:2001, for composition purposes. In this case, the appropriate fields of the IndexedFaceSet2D BIFS node are updated as described in the following.

a.) The coordinates of the mesh points (vertices) are obtained from the output of the Mesh Object decoder. The Mesh Object uses a pixel-based local coordinate system with *x*-axis pointing to the right and *y*-axis pointing down. However, ISO/IEC 14496-1:2001 specifies a coordinate system with *y*-axis pointing up. Therefore, a simple coordinate transform shall be applied to the *y*-coordinates of mesh points to ensure the proper orientation of the object after composition. The *y*-coordinate y_n of a decoded mesh node point *n* shall be transformed as follows:

$$Y_n = - y_n ,$$

where Y_n is the *y*-coordinate of this mesh node point in the coordinate system as specified in ISO/IEC 14496-1:2001. The origin of this object is at the top-left point. The same transform shall be applied to the coordinates of node points of each Mesh Object Plane (MOP).

b.) The coordinate indices are the indices of the mesh points forming faces (triangles) obtained from the output of the Mesh Object decoder. All decoded faces are triangles. The topology of a Mesh Object is constant starting from an intra-coded MOP, throughout a sequence of predictive-coded MOPs (until the next intra-coded MOP); therefore, the coordinate indices shall be updated only for intra-coded MOPs.

c.) Texture coordinates for mapping textures onto the mesh geometry are computed from the decoded node point locations of an intra-coded Mesh Object Plane and its bounding rectangle. Let x_{min}, y_{min} and x_{max}, y_{max} define the bounding rectangle of all node points of an intra-coded MOP. Then the width *w* and height *h* of the texture map shall be:

$$w = \text{ceil}(x_{max}) - \text{floor}(x_{min}) ,$$

$$h = \text{ceil}(y_{max}) - \text{floor}(y_{min}) .$$

A texture coordinate pair (s_n, t_n) is computed for each node point $p_n = (x_n, y_n)$ as follows:

$$s_n = (x_n - \text{floor}(x_{min}))/w ,$$

$$t_n = 1.0 - (y_n - \text{floor}(y_{min}))/h .$$

The topology of a Mesh Object is constant starting from an intra-coded MOP, throughout a sequence of predictive-coded MOPs (until the next intra-coded MOP); therefore, the texture coordinates shall be updated only for intra-coded MOPs.

d.) The texture coordinate indices are identical to the coordinate indices.

© ISO/IEC 2004 – All rights reserved

8.4 Spatial Scalability composition

Background composition is used in forming the background region for objects at the enhancement layer of spatial scalability when the value of both enhancement_type and background_composition is '1'. This process is useful when the enhancement VOP corresponds to the partial region of the VOP belonging to the reference layer. In this process, current enhancement VOP is composed with up-sampled VOP has same display time in the base layer as back ground image.

In object based spatial scalability, if enhancement_type is "1", use_ref_shape is "1" and background_composition is '1', reference layer should be up-sampled in the same process described in subclause 7.9.2.6 for background composition. If reference layer has binary shape information, it shall be up-sampled in the same process described in subclause 7.5.4.4. And for the region outside shape of the enhancement VOP, the both texture and shape pixels should be placed on the up-sampled VOP in the reference layer.

In object based spatial scalability, if enhancement_type is "1", use_ref_shape is "0" and background_composition is '1', reference layer shape and texture should be up-sampled in the same process described in subclause 7.5.4.4 and 7.9.2.6. And for the shape and texture outside the enhancement VOP, the shape and texture of the up-sampled VOP of the reference layer should be used. In this case, enhancement layer shape and texture are derived by composing the shapes and textures inside enhancement VOP and ouside enhancement VOP. The shape outside enhancement VOP is given by the collocated shape of the up-sampled VOP of the reference layer, and the texture outside enhancement VOP is given by the collocated texture of the up-sampled VOP of the reference layer.

9 Profiles and Levels

NOTE In this part of ISO/IEC 14496 the word "profile" is used as defined below. It should not be confused with other definitions of "profile" and in particular it does not have the meaning that is defined by ISO/IEC ISP.

Profiles and levels provide a means of defining subsets of the syntax and semantics of this part of ISO/IEC 14496 and thereby the decoder capabilities required to decode a particular bitstream. A profile is a defined sub-set of the entire bitstream syntax that is defined by this part of ISO/IEC 14496. A level is a defined set of constraints imposed on parameters in the bitstream. Conformance tests will be carried out against defined profiles at defined levels.

The purpose of defining conformance points in the form of profiles and levels is to facilitate bitstream interchange among different applications. Implementers of this part of ISO/IEC 14496 are encouraged to produce decoders and bitstreams which correspond to those defined conformance regions. The discretely defined profiles and levels are the means of bitstream interchange between applications of this part of ISO/IEC 14496.

In this clause the constrained parts of the defined profiles and levels are described. All syntactic elements and parameter values which are not explicitly constrained may take any of the possible values that are allowed by this part of ISO/IEC 14496. A decoder shall be deemed to be conformant to a given profile at a given level if it is able to properly decode all allowed values of all syntactic elements as specified by that profile at that level.

9.1 Visual Object Types

The following four tables list the tools included in each of the Object Types. Bitstreams that represent a particular object corresponding to an Object Type shall not use any of the tools for which the table does not have an 'X'.

© ISO/IEC 2004 – All rights reserved

Table 9-1 — Tools for Version 1 Visual Object Types

Visual Tools	Visual Object Types								
	Simple	Core	Main	Simple Scalable	N-bit	Animated 2D Mesh	Basic Animated Texture	Still Scalable Texture	Simple Face
Basic • I-VOP • P-VOP • AC/DC Prediction • 4-MV, Unrestricted MV	X	X	X	X	X	X			
Error resilience • GOB Resynchronization • Data Partitioning • Reversible VLC	X	X	X	X	X	X			
Short Header	X	X	X		X	X			
B-VOP		X	X	X	X	X			
P-VOP with OBMC (Texture)									
Method 1/Method 2 Quantisation		X	X		X	X			
P-VOP based temporal scalability • Rectangular • Arbitrary Shape		X	X		X	X			
Binary Shape		X	X		X	X	X		
Grey Shape			X						
Interlace			X						
Sprite			X						
Temporal Scalability (Rectangular)				X					
Spatial Scalability (Rectangular)				X					
N-Bit					X				
Scalable Still Texture						X	X	X	
2D Dynamic Mesh with uniform topology						X	X		
2D Dynamic Mesh with Delaunay topology						X			
Facial Animation Parameters									X

NOTE 1: ″Binary Shape Coding" includes constant alpha.

NOTE 2: The parameters are restricted as follows for the tool "P-VOP based temporal scalability":
- ref_select_code shall be either '00' or '01'.
- vop_coding_type of the enhancement layer VOP shall be either '00' or '01'.
- vop_coding_type of the reference layer VOP shall be either '00' or '01'.
- load_backward_shape shall be '0' and background composition is not performed.

NOTE 3: An 'X' for the "Method 1/Method 2 Quantisation" indicates that both quantisation methods are supported by the Visual Object Type. Where there is no 'X' the only quantisation method supported is the Second Inverse Quantisation Method (subclause 7.4.4.2).

© ISO/IEC 2004 – All rights reserved

Table 9-2 — Tools for Version 2 Visual Object Types

Visual Tools	Visual Object Types				
	Advanced Real Time Simple	Advanced Coding Efficiency	Advanced Scalable Texture	Core Scalable	Simple FBA
Basic • I-VOP • P-VOP • AC/DC Prediction • 4-MV, Unrestricted MV	X	X		X	
Error resilience • GOB Resynchronization • Data Partitioning • Reversible VLC	X	X		X	
Short Header	X	X		X	
B-VOP		X		X	
P-VOP with OBMC (Texture)					
Method 1/Method 2 Quantisation		X		X	
P-VOP based temporal scalability • Rectangular • Arbitrary Shape		X		X	
Binary Shape		X		X	
Grey Shape		X			
Interlace		X			
Sprite					
Temporal Scalability (Rectangular)				X	
Spatial Scalability (Rectangular)				X	
N-Bit					
Scalable Still Texture			X		
2D Dynamic Mesh with uniform topology					
2D Dynamic Mesh with Delaunay topology					
Facial Animation Parameters					X
Body_Animation_Parameters					X
Reduced Resolution VOP	X				
NEWPRED	X				
Global Motion Compensation		X			
Quarter-pel Motion Compensation		X			
SA-DCT		X			
Error Resilience for Visual Texture Coding			X		
Wavelet Tiling			X		

© ISO/IEC 2004 – All rights reserved

Scalable Shape Coding for Still Texture			X		
Object Based Temporal Scalability				X	
Object Based Spatial Scalability				X	

Table 9-3 — Tools for Studio Visual Object Types

	Visual Object Type	
Visual Tools	**Simple Studio**	**Core Studio**
Basic -Progressive/Interlaced -Frame/Field Structure -Slice Structure -Studio DPCM Block -Studio Binary Shape -Studio Grayscale Shape	X	X
I-VOP	X	X
P-VOP		X
Studio Sprite		X

NOTE 3: The allowed values of 'chroma_format' in StudioVideoObjectLayer() are defined in the level definition.

NOTE 4: The allowed values of 'bits_per_pixel' and 'alpha_bits_per_pixel' in StudioVideoObjectLayer() are defined in the level definition

Table 9-4 — Tools for FGS Visual Object Types

Visual Tools	**Visual Object Types**	
	Advanced Simple	**Fine Granularity Scalable**
I-VOP	X	X
P-VOP	X	X
B-VOP	X	X
DC Prediction	X	X
AC Prediction	X	X
4-MV, Unrestricted MV	X	X
GOB Resynchronization	X	X
Data Partitioning	X	X
Reversible VLC	X	X
Short Header	X	X
Method 1/Method 2 Quantisation	X	X
Interlace	X	X
Global Motion Compensation	X	
Quarter-pel Motion Compensation	X	
Fine Granularity Scalability		X
FGS Temporal Scalability		X

© ISO/IEC 2004 – All rights reserved

NOTE 1 — The interlace tools are not used for levels L0, L1, L2, and L3 of AS and FGS Profiles.

NOTE 2 — The Advanced Simple Object Type is the base-layer of the Fine Granularity Scalable Object Type. If FGS Temporal Scalability is used, B-VOP is not allowed in the base-layer. However, B-VOP is allowed in the base-layer when FGS Temporal Scalability is not used in the Fine Granularity Scalable Object Type.

NOTE 3 — The Advanced Simple Object Type is the base-layer of the Fine Granularity Scalable Object Type. If FGS or FGS Temporal Scalability is used, Short Header is not allowed in the base-layer. However, Short Header is allowed in the base-layer when FGS or FGS Temporal Scalability is not used in the Fine Granularity Scalable Object Type.

9.2 Visual Profiles

Decoders that conform to a Profile shall be able to decode all objects that comply to the Object Types for which the table lists an 'X'.

Table 9-5 — Version 1 Visual Profiles

	Object Types / Profiles	Simple	Core	Main	Simple Scalable	N-Bit	Animated 2D Mesh	Basic Animated Texture	Scalable Texture	Simple Face
1.	Simple	X								
2.	Simple Scaleable	X			X					
3.	Core	X	X							
4.	Main	X	X	X					X	
5.	N-Bit	X	X			X				
6.	Hybrid	X	X				X	X	X	X
7.	Basic Animated Texture							X	X	X
8.	Scaleable Texture								X	
9.	Simple FA									X

Table 9-6 — Version 2 Visual Profiles

	Object Types / Profiles	Simple	Simple Scalable	Core	Core Scalable	Advanced Real Time Simple	Advanced Coding Efficiency	Advanced Scable Texture	Simple FBA
V2-1.	Advanced Real Time Simple	X				X			
V2-2.	Core Scalable	X	X	X	X				
V2-3.	Advanced Coding Efficiency	X		X			X		
V2-4.	Advanced Core	X		X				X	
V2-5.	Advanced Scalable Texture							X	
V2-6.	Simple FBA								X

Note Object Types that are not listed in this table are not decodable by the decoders complying to the Profiles listed in this table.

© ISO/IEC 2004 – All rights reserved

Table 9-7 — Studio Visual Profiles

Object Types / Profiles	Simple Studio	Core Studio
AMD1-1 Simple Studio	X	
AMD1-2 Core Studio	X	X

Note that object types that are not listed in this table are not decordable by decoders complying to the Profiles listed in this table.

Note that the Profiles listed in this table can be grouped into Natural Visual.

Table 9-8 shows the definition of Advanced Simple Profile and Fine Granularity Scalable Profile.

Table 9-8 — Definition of Advanced Simple Profile and Fine Granularity Scalable Profile

ID	Object Types / Profiles	Simple	Advanced Simple	Fine Granularity Scalable
AS	Advanced Simple	X	X	
FGS	Fine Granularity Scalable	X	X	X

9.3 Visual Profiles@Levels

9.3.1 Natural Visual

The table that describes the natural visual profiles is given in Annex N.

9.3.2 Synthetic Visual

9.3.2.1 Scalable Texture Profile

Table 9-9 — Scalable texture profile levels

Profile	Level	Default wavelet filter	Max. down-load filter length	Max. no. of decom-position levels	Typical visual session size[1]	Max. Qp value	Max. no. of pixels/ session	VCV decoder rate (equi-valent MB/s)[2]	Max. no. of bit-planes for DC values	Max. VCV buffer size (equivalent MB)[2]
Scalable Texture	L3	Float, Integer	ON, 15	10	8192 x8192	12 bits	67108864	262144	18	262144
Scalable Texture	L2	Integer	ON, 15	8	2048 x2048	10 bits	4194304	16384	16	16384
Scalable Texture	L1	Integer	OFF	5	704 x576	8 bits	405504	1584	13	1584

Note: (1)This column is for informative use only. It provides an example configuration of the max. no. of pixels/session.
(2) This still texture VCV model is separate from the global video VCV model. An equivalent MB corresponds to 256 pixels.

© ISO/IEC 2004 – All rights reserved

9.3.2.2 Simple Face Animation Profile

All ISO/IEC 14496-2 facial animation decoders (for all object types) are required to generate at their output a facial model including all the feature points defined in this part of ISO/IEC 14496, even if some of the features points will not be affected by any information received from the encoder.

The Simple Face object is not required to implement the viseme_def/expression_def functionality.

Level 1:

- number of objects: 1,
- The total FAP decode frame-rate in the bitstream shall not exceed 72 Hz ,
- The decoder shall be capable of a face model rendering update of at least 15 Hz, and
- Maximum bitrate 16 kbit/s.

Level 2:

- maximum number of objects: 4,
- The FAP decode frame-rate in the bitstream shall not exceed 72 Hz (this means that the FAP decode framerate is to be shared among the objects),
- The decoder shall be capable of rendering the face models with the update rate of at least 60 Hz, sharable between faces, with the constraint that the update rate for each individual face is not required to exceed 30 Hz, and
- Maximum bitrate 32 kbit/s.

9.3.2.3 Simple FBA Profile

All ISO/IEC 14496-2 FBA decoders (for all object types) are required to generate at their output a humanoid model including all the feature points and joints defined in this part of ISO/IEC 14496, even if some of the features points and joints will not be affected by any information received from the encoder.

Level 1:

- number of objects: 1
- The total FBA decode frame-rate in the bitstream shall not exceed 72 Hz
- The decoder shall be capable of a humanoid model rendering update of at least 15 Hz
- Maximum bitrate 32 kbit/s
- The decoder is not required to animate Spine3, Spine4 and Spine5 BAP groups

Level 2:

- maximum number of objects: 4
- The FBA decode frame-rate in the bitstream shall not exceed 72 Hz (this means that the FBA decode framerate is to be shared among the objects)
- The decoder shall be capable of rendering the humanoid models with the update rate of at least 60 Hz, sharable between humanoids, with the constraint that the update rate for each individual humanoid is not required to exceed 30 Hz
- Maximum bitrate 64 kbit/s

© ISO/IEC 2004 – All rights reserved

9.3.2.4 Advanced Core Profile and Advanced Scalable Texture Profile

Table 9-10 — Advanced core profile levels

Visual Profile	Level	Default wavelet filter	Max. down-load filter, length	Max. no. of decom-position levels	Typical visual session size[1]	Max. Qp value (bits)	Max. no. of pixels / session[2]	VCV decoder rate (equi-valent MB/s)[3]	Max. no. of bit-plains for DC values	Max. VCV buffer size (equi-valent MB)	Max. STO packet length (bits)	Max. no. of pixels / tile	Max. no. of tile
Advanced Core	L2	Integer	ON, 15	8	8192 x8192	10	67108864	262144	16	262144	8192	262144	2048
Advanced Core	L1	Integer	OFF	5	2048 x2048	8	4194304	16384	13	16384	4096	65536	1024
Advanced Scalable Texture	L3	Float, Integer	ON, 15	10	8192 x8192	12	67108864	262144	18	262144	8192	67108864	4096
Advanced Scalable Texture	L2	Integer	ON, 15	8	2048 x2048	10	4194304	16384	16	16384	4096	4194304	2048
Advanced Scalable Texture	L1	Integer	OFF	5	704 x576	8	405504	1584	13	1584	2048	405504 (=4xCIF)	1024

Note: (1) This column is for informative use only. It provides an example configuration of the max. no. of pixels/session.
(2) When the number of pixels/session is larger than max. no. of pixels/tile, tiling_disable shall be 0.
(3) This still texture VCV model is separate from the global video VCV model. An equivalent MB corresponds to 256 pixels.

9.3.3 Synthetic/Natural Hybrid Visual

The *Levels* of the Profiles which support both Natural Visual Object Types and Synthetic Visual Object Types are specified by giving bounds for the natural objects and for the synthetic objects. Parameters like bitrate can be combined across natural and synthetic objects.

9.3.3.1 Basic Animated Texture Profile

Level 1 = Simple Facial Animation Profile @ Level 1 + Scalable Texture @ Level 1 + the following restrictions on Basic Animated Texture object types:

- Maximum number of Mesh objects (with uniform topology): 4,
- Maximum total number of nodes (vertices) in Mesh objects: 480,
  (= 4 x nr. Of nodes of a uniform mesh covering a QCIF image with 16x16 pixel elements),
- Maximum frame-rate of a Mesh object: 30 Hz, and
- Maximum bitrate of Mesh objects: 64 kbit/sec.

Level 2 = Simple Facial Animation Profile @ Level 2 + Scalable Texture @ Level 2 + the following restrictions on Basic Animated Texture object types:

- Maximum number of Mesh objects (with uniform topology): 8,
- Maximum total number of nodes (vertices) in Mesh objects: 1748,
  (= 4 x nr. of nodes of a uniform mesh covering a CIF image with 16x16 pixel elements),
- Maximum frame-rate of a Mesh object: 60 Hz, and
- Maximum bitrate of Mesh objects: 128 kbit/sec.

© ISO/IEC 2004 – All rights reserved

9.3.3.2 Hybrid Profile

Level 1 = Core Visual Profile @ Level 1 + Basic Animated Texture Profile @ Level 1 + the following restrictions on Animated 2D Mesh object types:

- Maximum number of Mesh objects (with uniform or Delaunay topology): 4
  (= maximum number of objects in visual session)
- Maximum total number of nodes (vertices) in Mesh objects: 480
  (= 4 x nr. of nodes of a uniform mesh covering a QCIF image with 16x16 pixel elements)
- Maximum frame-rate of a Mesh object: 30 Hz
  (= maximum frame-rate of video object)
- Maximum bitrate of Mesh objects: 64 kbit/sec.

Level 2 = Core Visual Profile @ Level 2 + Basic Animated Texture Profile @ Level 2 + the following restrictions on Animated 2D Mesh object types:

- Maximum number of Mesh objects(with uniform or Delaunay topology): 8
  (= maximum number of objects in visual session)
- Maximum total number of nodes (vertices) in Mesh objects: 1748
  (= 4 x nr. of nodes of a uniform mesh covering a CIF image with 16x16 pixel elements)
- Maximum frame-rate of a Mesh object: 60 Hz
  (= 2 x the maximum frame-rate of video object)
- Maximum bitrate of Mesh objects: 128 kbit/sec.

© ISO/IEC 2004 – All rights reserved

Annex A
(normative)

Coding transforms

A.1 Discrete cosine transform for video texture

The NxN two dimensional DCT is defined as:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

with u, v, x, y = 0, 1, 2, … N-1

where x, y are spatial coordinates in the sample domain

u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u,v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

If each pixel is represented by *n* bits per pixel, the input to the forward transform and output from the inverse transform is represented with (*n*+1) bits. The coefficients are represented in (*n*+4) bits. The dynamic range of the DCT coefficients is $[-2^{n+3}:+2^{n+3}-1]$.

The N by N inverse discrete transform shall conform to IEEE Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, Std 1180-1990, December 6, 1990, with the following modifications:

1) In item (1) of subclause 3.2 of the IEEE specification, the last sentence is replaced by: <<Data sets of 1 000 000 (one million) blocks each should be generated for (L=256, H=255), (L=H=5) and (L=384, H=383).>>

2) The text of subclause 3.3 of the IEEE specification is replaced by : <<For any pixel location, the peak error shall not exceed 2 in magnitude. There is no other accuracy requirement for this test.>>

3) Let F be the set of 4096 blocks Bi[y][x] (i=0..4095) defined as follows:

a) Bi[0][0] = i - 2048

b) Bi[7][7] = 1 if Bi[0][0] is even, Bi[7][7] = 0 if Bi[0][0] is odd

c) All other coefficients Bi[y][x] other than Bi[0][0] and Bi[7][7] are equal to 0

For each block Bi[y][x] that belongs to set F defined above, an IDCT that claims to be compliant shall output a block f[y][x] that as a peak error of 1 or less compared to the reference saturated mathematical integer-number IDCT fii(x,y). In other words, | f[y][x] - fii(x,y)| shall be <= 1 for all x and y.

© ISO/IEC 2004 – All rights reserved

NOTE 1 Clause 2.3 Std 1180-1990 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. Every macroblock is required to be refreshed before it is coded 132 times as predictive macroblocks. Macroblocks in B-pictures (and skipped macroblocks in P-pictures) are excluded from the counting because they do not lead to the accumulation of mismatch errors. This requirement is the same as indicated in 1180-1990 for visual telephony according to ITU-T Recommendation H.261.

NOTE 2 Whilst the IEEE IDCT standard mentioned above is a necessary condition for the satisfactory implementation of the IDCT function it should be understood that this is not sufficient. In particular attention is drawn to the following sentence from subclause 5.4: "Where arithmetic precision is not specified, such as the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values."

A.1.1 Discrete cosine transform for the Studio Profile

The NxN two dimensional DCT is defined as:

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

with u, v, x, y = 0, 1, 2, … N-1

where x, y are spatial coordinates in the sample domain

u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u,v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N}$$

If each pixel is represented by *n* bits per pixel, the input to the forward transform and output from the inverse transform is represented with (*n*+1) bits. The bit precision of the DCT coefficients changes in accordance with the value of mpeg2_stream defined in StudioVideoObjectLayer(). In case of mpeg2_stream = 0, The coefficients are represented in (n+7) bits including three fractional bits. The dynamic range of the coefficients is $[-2^{n+6}:+2^{n+6}-1]$. In case of mpeg2_stream = 1, The coefficients are represented in (n+4) bits. The dynamic range is $[-2^{n+3}:+2^{n+3}-1]$.

The N by N inverse discrete transform shall conform to IEEE Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, Std 1180-1990, December 6, 1990.

NOTE 1 Sublause 2.3 Std 1180-1990 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. Every macroblock is required to be refreshed before it is coded 132 times as predictive macroblocks. Macroblocks in B-pictures (and skipped macroblocks in P-pictures) are excluded from the counting because they do not lead to the accumulation of mismatch errors. This requirement is the same as indicated in 1180-1990 for visual telephony according to ITU-T Recommendation H.261.

NOTE 2 Whilst the IEEE IDCT standard mentioned above is a necessary condition for the satisfactory implementation of the IDCT function it should be understood that this is not sufficient. In particular, attention is drawn to the following sentence from subclause 5.4: "Where arithmetic precision is not specified, such as the calculation of the IDCT, the precision shall be sufficient so that significant errors do not occur in the final integer values."

© ISO/IEC 2004 – All rights reserved

A.2 Discrete wavelet transform for still texture

A.2.1 Adding the mean

Before applying the inverse wavelet transform, the mean of each color component ("mean_y", "mean_u", and "mean_v") is added to the all wavelet coefficients of DC subband.

A.2.2 Wavelet filter

A 2-D separable inverse wavelet transfrom is used to synthesize the still texture. The default wavelet composition is performed using Daubechies (9,3) tap biorthogonal filter bank. The inverse DWT is performed either in floating or integer operations depending on the field "wavelet_filter_type", defined in the syntax.

The floating filter coefficients are:

Lowpass	**g[ ] =**	
[0.35355339059327	0.70710678118655	0.35355339059327]
Highpass	**h[ ] =**	
[0.03314563036812	0.06629126073624	-0.17677669529665
-0.41984465132952	0.99436891104360	-0.41984465132952
-0.17677669529665	0.06629126073624	0.03314563036812]

The integer filter coefficients are:

Lowpass	**g[ ] =**	
32	64	32
Highpass	**h[ ] =**	
3	6	-16
-38	90	-38
-16	6	3

The synthesis filtering operation is defined as follows:

$$y[n] = \sum_{i=-1}^{1} L[n+i]*g[i+1] \quad + \sum_{i=-4}^{4} H[n+i]*h[i+4]$$

where

- n = 0, 1, ... N-1, and N is the number of output points;
- L[2*i] = xl[i] and L[2*i+1] = 0 for i=0,1,...,N/2-1, and {xl[i]} are the N/2 input wavelet coefficients in the low-pass band;
- H[2*i+1] = xh[i] and H[2*i] = 0 for i=0,1,...,N/2-1, and {xh[i]} are the N/2 input wavelet coefficients in the high-pass band.

NOTE 1 the index range for h[] is from 0 to 8;

NOTE 2 the index range for g[] is from 0 to 2;

NOTE 3 the index range for L[] is from -1 to N;

NOTE 4 the index range for H[] is from -4 to N+3; and

NOTE 5 the values of L[] and H[] for indexes less than 0 or greater than N-1 are obtained by symmetric extension described in the following subclause.

© ISO/IEC 2004 – All rights reserved

In the case of integer wavelet, the outputs at each composition level are scaled down with dividing by 8192 with rounding to the nearest integer.

A.2.3 Symmetric extension

A symmetric extension of the input wavelet coefficients is performed and the up-sampled and extended wavelet coefficients are generated. Note that the extension process shown below is an example when the extension is performed before up-sampling and that only the generated coefficients are specified. Two types of symmetric extensions are needed, both mirror the boundary pixels. Type A replicates the edge pixel and Type B does not replicate the edge pixel. This is illustrated in Figure A.1 and Figure A.2, where the edge pixel is indicated by z. The types of extension for the input data to the wavelet filters are shown in Table A.1.

Type A **…v w x y z** | **z** y x w v…

Type B **…...v w x y** | **z** y x w v…

Figure A.1 — Symmetrical extensions at leading boundary

Type A …v w x y **z** | **z y x w v…**

Type B ….v w x y **z** | **y x w v….**

Figure A.2 — Symmetrical extensions at the trailing boundary

Table A.1 — Extension method for the input data to the synthesis filters

	boundary	**Extension**
lowpass input xl[] to 3-tap filter g[]	leading trailing	TypeB TypeA
highpass input xh[] to 9-tap filter h[]	leading trailing	TypeA TypeB

The generated up-sampled and extended wavelet coefficients L[] and H[] are eventually specified as follows:

low-pass band :…0 L[2] 0 | **L[0] 0 L[2] 0 … L[N-4] 0 L[N-2] 0** | L[N-2] 0 L[N-4] 0 …

high-pass band:… H[3] 0 H[1] | **0 H[1] 0 H[3] … 0 H[N-3] 0 H[N-1]** | 0 H[N-1] 0 H[N-3]…

A.2.4 Decomposition level

The number of decomposition levels of the luminance component is defined in the input bitstream. The number of decompostion levels for the chrominance components is one level less than that of the luminance components. If texture_object_layer width or texture_object_layer height cannot be divisible by (2 ^ decomposition_levels), then shape adaptive wavelet is applied.

A.2.5 Shape adaptive wavelet filtering and symmetric extension

A.2.5.1 Shape adaptive wavelet

The 2-D inverse shape adaptive wavelet transform uses the same wavelet filter as specified in Table A.1. According to the shape information, segments of consecutive output points are reconstructed and put into the correct locations. The filtering operation of shape adaptive wavelet is a generalization of that for the regular wavelet. The generalization allows the number of output points to be an odd number as well as an even number. Relative to the bounding rectangle, the starting point of the output is also allowed to be an odd number as well as an even number according to the shape information. Within the generalized wavelet filtering, the regular wavelet filtering is a special case where the number of output points is an even number and the starting point is an even

© ISO/IEC 2004 – All rights reserved

number (0) too. Another special case is for reconstruction of rectangular textures with an arbitrary size where the number of output points may be even or odd and the starting point is always even (0).

The same synthesis filtering is applied for shape-adaptive wavelet composition, i.e:

$$y[n] = \sum_{i=-1}^{1} L[n+i]*g[i+1] + \sum_{i=-4}^{4} H[n+i]*h[i+4]$$

where

- n = 0, 1, ... N-1, and N is the number of output points;
- L[2*i+s] = xl[i] and L[2*i+1-s] = 0 for i=0,1,...,(N+1-s)/2-1, and {xl[i]} are the (N+1-s)/2 input wavelet coefficients in the low-pass band;
- H[2*i+1-s] = xh[i] and H[2*i+s] = 0 for i=0,1,...,(N+s)/2-1, and {xh[i]} are the (N+s)/2 input wavelet coefficients in the high-pass band.

The only difference from the regular synthesis filtering is to introduce a binary parameter s in up-sampling, where s = 0 if the starting point of the output is an even number and s = 1 if the starting point of the output is an odd number.

The symmetric extension for the generalized synthesis filtering is specified in Table A.2 if N is an even number and in Table A.3 if N is an odd number.

Table A.2 — Extension method for the data to the synthesis wavelet filters if N is even

	Boundary	extension (s=0)	extension(s=1)
lowpass input xl[] to 3-tap filter g[]	Leading Trailing	TypeB TypeA	TypeA TypeB
highpass input xh[] to 9-tap filter h[]	Leading Trailing	TypeA TypeB	TypeB TypeA

Table A.3 — Extension method for the data to the synthesis wavelet filters if N is odd

	Boundary	extension(s=0)	extension(s=1)
lowpass input xl[] to 3-tap filter g[]	Leading Trailing	TypeB TypeB	TypeA TypeA
highpass input xh[] to 9-tap filter h[]	Leading Trailing	TypeA TypeA	TypeB TypeB

A.3 Shape-Adaptive DCT (SA-DCT)

Similar to standard 8x8-DCT from subclause A.1, SA-DCT converts a 8x8-block with texture data f[y][x] into a block with corresponding SA-DCT coefficients and, vice versa, inverse SA-DCT converts inverse quantised coefficients F[v][u] into decoded texture data f[x][y]. The definition of forward SA-DCT is based on the four subsequent processing steps S1 to S4 which are described in more detail in subclause A.3.1 on the basis of pseudo-C code.

The SA-DCT process is controlled by a binary array 'f_shape[y][x]' which describes the shape of input data f[y][x] and which is collocated with the corresponding 8x8-block. The elements of 'f_shape[y][x]' have the value '255' for opaque and '0' for transparent pels. This binary array has to be derived from the decoded BAB. For luminance blocks the elements of 'f_shape[y][x]' are taken from the corresponding subblock of 16x16-BAB and for chrominance blocks BAB is subsampled as specified for shape decoding.

An auxiliary binary array 'shift_shape[y][x]' and two one-dimensional counter arrays 'pels_height[x]' and 'coeff_width[v]' are implicitly computed from 'f_shape[y][x]' during the internal shift operations of forward SA-DCT (see S1 and S3). These auxiliary shape parameters are also needed for inverse SA-DCT. At the decoder, these

© ISO/IEC 2004 – All rights reserved

required shape parameters are first derived from decoded BAB during a pre-processing step I-S1 followed by four subsequent processing steps for inverse SA-DCT. The pre-processing step I-S1 as well as the four processing steps I-S2 to I-S5 of inverse SA-DCT are described in more detail in subclause A.3.2 on the basis of pseudo-C code.

NOTES -

1 The auxiliary array 'shift_shape[y][x]' describes the binary shape of the texture pattern 'shift_texture[y][x]' resulting from vertical shift of SA-DCT (see S1). The elements of 'pels_height[x]' denote the number of opaque pels in each column of f[y][x]. The elements of 'coeff_width[v]' describe the number SA-DCT coefficients existing in each row of F[v][u]. The permitted values of elements in 'pels_height[x]' and 'coeff_width[v]' lie in a range from 0 to 7.

2 The kernels of the one-dimensional DCT transforms used separately in horizontal and vertical direction (see steps S2 and S4) as well as the kernels of the corresponding one-dimensional inverse DCT transforms (see steps I-S2 and I-S4) have to be determined adaptively in dependence on the above mentioned shape parameters 'pels_height[x]' and 'coeff_width[v]'.

3 Floating point arithmetic should be used during entire forward and inverse SA-DCT (i.e. from S1 to S4 and from I-S2 to I-S5). The input to forward SA-DCT is an integer with 9 bits representing the dynamic range of [-256;+255]. The output of inverse SA-DCT is rounded to corresponding integers. The coefficients at the input of inverse SA-DCT are integers of 12 bits with a dynamic range of [-2048;+2047].

4 The arithmetic precision for the internal calculations of inverse SA-DCT, especially the one used during one-dimensional inverse DCT transform in processing steps I-S2 and I-S4 as well as the one of the intermediate arrays 'shift_intermediate[v][x]' and 'F_intermediate[v][x]', is not specified. Taking into account all possible shapes for 8x8-blocks with 'opaque_pels < 64', the precision shall be sufficient so that significant errors do not occur in the final integer values derived from f[y][x]. Analogous to inverse standard 8x8-DCT, the maximal error should meet the requirements of 1189-1990 for visual telephony.

A.3.1 Definition of Forward SA-DCT

S1: vertical shift of all opaque pels in f[y][x] to the most upper position and construction of corresponding auxiliary arrays 'shift_texture[y][x]' and 'shift_shape[y][x]

```
for (x=0; x<8; x++) {
   for (y=0; y<8; y++)
      shift_shape[y][x]=0;
   pels_count=0;
   for (y=0; y<8; y++)
      if(f_shape[y][x]==255) {
         shift_texture[pels_count][x]= f[y][x];
         shift_shape[pels_count][x]=255;
         pels_count++;
      }
   pels_height[x]=pels_count;
}
```

S2: application of shape-adaptive one-dimensional DCT to the first 'pels_height[x]' pels in all columns of 'shift_texture[y][x] with 'pels_height[x] != 0'

```
for (x=0; x<8; x++) {
   if(pels_height[x]==0) continue;
   scaling = sqrt(2.0/pels_height[x]);
   for (v=0; v<pels_height[x]; v++) {
      c0=1.0;
      if(v==0) c0= sqrt(0.5);
      F_intermediate[v][x]=0.0;
      for (y=0; y<pels_height[x]; y++){
         DCT_N = c0 * cos(v*(y+0.5)*(PI/pels_height[x]));
         F_intermediate[v][x] += scaling * DCT_N * shift_texture[y][x];
      }
   }
}
```


© ISO/IEC 2004 – All rights reserved

S3: horizontal shift of all intermediate coefficients in 'F_intermediate[v][x]' to most left position and construction of corresponding auxiliary array 'shift_intermediate[v][x]'

```
for (v=0; v<8; v++) {
   coeff_count=0;
   for (x=0; x<8; x++)
      if(shift_shape[v][x]==255) {
         shift_intermediate[v][coeff_count]= F_intermediate[v][x];
         coeff_count++;
      }
   coeff_width[v]=coeff_count
}
```

S4: application of shape-adaptive one-dimensional DCT to the first 'coeff_width[v]' intermediate coefficients in all rows of 'shift_intermediate[v][x]' with 'coeff_width[v] != 0'

```
for (v=0; (v<8) && (coeff_width[v]!=0); v++) {
   scaling = sqrt(2.0/coeff_width[v]);
   for (u=0; u<coeff_width[v]; u++) {
      c0=1.0;
      if(u==0) c0= sqrt(0.5);
      F[v][u]=0.0;
      for (x=0; x<coeff_width[v]; x++) {
         DCT_N = c0 * cos(u*(x+0.5)*(PI/coeff_width[v]));
         F[v][u] += scaling * DCT_N *shift_intermediate[v][x];
      }
   }
}
```

A.3.2 Definition of Inverse SA-DCT

I-S1: computation of required shape parameters 'coeff_width[v]', 'shift_shape[y][x]' and 'pels_height[x]' from decoded binary shape array 'f_shape[y][x]' (only needed at decoder)

```
opaque_pels=0;
for (x=0; x<8; x++) {
   for (y=0; y<8; y++)
      shift_shape[y][x]=0;
   pels_count=0;
   for (y=0; y<8; y++)
      if(f_shape[y][x]==255) {
         shift_shape[pels_count][x]=255;
         pels_count++;
      }
   pels_height[x]=pels_count
opaque_pels+=pels_count
}
for (v=0; v<0; v++) {
   coeff_count=0;
   for (x=0; x<8; x++)
      if(shift_shape[v][x]==255)
         coeff_count++;
   coeff_width[v]=coeff_count
}
```

© ISO/IEC 2004 – All rights reserved

I-S2: application of shape-adaptive inverse one-dimensional DCT to the first 'coeff_width[v]' decoded SA-DCT coefficients in all rows of 'F[v][u]' with 'coeff_width[v] != 0':

```
for (v=0; (v<8) && (coeff_width[v]!=0); v++) {
   scaling = sqrt(2.0/coeff_width[v]);
   for (x=0; x<coeff_width[v]; x++) {
      shift_intermediate[v][x] = 0.0;
      for (u=0; u<coeff_width[v]; u++) {
      c0=1.0;
      if(u==0) c0= sqrt(0.5);
         DCT_N = c0 * cos(u*(x+0.5)*(PI/coeff_width[v]));
         shift_intermediate[v][x] += scaling * DCT_N * F[v][u];
      }
   }
}
```

I-S3: re-shift of all reconstructed intermediate coefficients in 'shift_intermediate[v][x]' from most left position to their original column position defined by 'shift_shape[y][x]'

```
for (v=0; v<8; v++) {
   coeff_count=0;
   for (x=0; x<8; x++)
   if(shift_shape[v][x]==255) {
      F_intermediate[v][x]=shift_intermediate[v][coff_count];
      coeff_count++;
   }
}
```

I-S4: application of shape-adaptive inverse one-dimensional DCT to the first 'pel_height[x]' intermediate coefficients in all columns of 'F_intermediate[v][x] with 'pels_height[x] != 0'

```
for (x=0; x<8; x++) {
   if(pels_height[x]==0) continue;
   scaling = sqrt(2.0/pels_height[x]);
   for (y=0; y<pels_height[x]; y++) {
      shift_texture[y][x]=0.0;
      for (v=0; v<pels_height[x]; v++) {
         c0=1.0;
         if(v==0) c0= sqrt(0.5);
         DCT_N = c0 * cos(v*(y+0.5)*(PI/pels_height[x]));
         shift_texture[y][x] += scaling * DCT_N * F_intermediate[v][x];
      }
   }
}
```

I-S5: re-shift of all reconstructed pels in 'shift_texture[y][x]' from most upper position to their original row position defined by 'f_shape[y][x]'

```
for (x=0; x<8; x++) {
   pels_count=0;
   for (y=0; y<8; y++)
   if(f_shape[y][x]==255) {
      f[y][x]=shift_texture[x];
      pels_count++;
   }
}
```

A.4 SA-DCT with DC Separation and ΔDC Correction (ΔDC-SA-DCT)

ΔDC-SA-DCT is an extension of SA-DCT towards additional pre- and post-processing steps, called DC separation which is applied at the encoder before forward SADCT and ΔDC correction which is used at the decoder after

© ISO/IEC 2004 – All rights reserved

inverse SA-DCT. The definition of forward ΔDC-SA-DCT is based on three processing steps ΔS1 to ΔS3 combined with the four steps S1 to S4 of forward SA-DCT from subclause A.3.1. Steps ΔS1 to ΔS3 are described in more detail in subclause A.4.1 on the basis of pseudo-C code. Inverse ΔDC-SA-DCT is based on corresponding reverse processing steps I-ΔS1, I-ΔS2 and I-ΔS4 combined with I-S1 to I-S5 of inverse SA-DCT from subclause A.3.2 plus an additional step I-ΔS3 which corrects an systematical error caused by the fact that an eventually non-zero output coefficient F[0][0] of SA-DCT is not transmitted in ΔDC-SA-DCT (see step ΔS3). Subclause A.4.2 presents a description of steps I-ΔS1 to I-ΔS4 on basis of pseudo-C codes.

NOTES -

1 Floating point arithmetic should be used during entire forward and inverse ΔDC-SA-DCT (i.e. from ΔS1 to ΔS3 including S1 to S4 of SA-DCT and from I-ΔS1 to I-ΔS4 including I-S2 to I-S5 of inverse SA-DCT). The input to forward ΔDC-SA-DCT is an integer with 9 bits representing the dynamic range from [-256;+255]. The output of inverse ΔDC-SA-DCT is rounded to corresponding integers. The coefficients at the input of inverse ΔDC-SA-DCT are integers of 12 bits with a dynamic range from [-2048;+2047].

2 The arithmetic precision for the internal calculations of inverse ΔDC-SA-DCT, especially the one used for mean value calculation in I-ΔS1, for inverse SA-DCT in I-ΔS2, for calculation of the ΔDC correction term in I-ΔS3 and for final additions in I-ΔS4 as well as the one of the intermediate variables 'mean_value', 'corr_term', 'f[y][x]' and 'F[v][u]', is not specified. Taking into account all possible shapes for 8x8-blocks with 'opaque_pels < 64', the precision shall be sufficient so that significant errors do not occur in the final integer values of f[y][x]. Analogous to inverse standard DCT, the maximal error should meet the requirements of 1189-1990 for visual telephony.

A.4.1 Definition of Forward ΔDC-SA-DCT

ΔS1: calculation of mean value and subtraction of mean value from f[y][x]

```
mean_value=0.0;
mean_count=0;
for (y=0; y<8; y++)
   for (x=0: x<8: x++) {
      bin_shape[y][x]=0;
      if(f_shape[y][x]==255)
          bin_shape[y][x]=1;
      mean_value+=bin_shape[y][x]*f[y][x];
      mean_count+=bin_shape[y][x];
   }
mean_value *= 1.0/mean_count;
for (y=0; y<8; y++)
   for (x=0; x<8; x++)
      if (f_shape[y][x]==255)
      f[y][x]-= mean_value;
```

ΔS2: application of SA-DCT including processing steps S1 to S4 from subclause A.3.1 to zero-mean output data f[y][x] from processing step ΔS1

```
f[y][x] => S1 => S2 => S3 => S4 => F[v][u]
```

ΔS3: transmission of scaled mean value as coefficient F[0][0] (an eventually non-zero output coefficient [0][0] of SA-DCT in ΔS2 is overwritten and not transmitted in ΔDC-SA-DCT because it is redundant and can be reconstructed from other data received at the decoder)

```
F[0][0] = 8.0*mean_value
```

A.4.2 Definition of Inverse ΔDC-SA-DCT

I-ΔS1: extraction of re-scaled mean value from decoded coefficients F[v][u] and zero-setting of F[0][0]

```
mean_value = F[0][0]/8.0
F[0][0] = 0.0
```

© ISO/IEC 2004 – All rights reserved

I-ΔS2: application of inverse SA-DCT including processing steps I-S1 to I-S5 from subclause A.3.2 to modified output coefficients F[v][u] from processing step I-ΔS1 (I-S1 is only needed at the decoder)

```
F[v][u] => (I-S1) => I-S2 => I-S3 => I-S4 => => I-S5 => f[y][x]
```

I-ΔS3: calculation of ΔDC correction term

```
check_sum=0.0;
sqrt_sum=0.0;
for (x=0;x<8; x++) {
if(pels_height[x]==0) continue;
   sqrt_sum += sqrt(pels_height(x));
   for (y=0; y<8; y++)
      check_sum += bin_shape[y][x]*f[y][x];
}
corr_term = check_sum/sqrt_sum
```

I-ΔS4: addition of re-scaled mean value from step I-ΔS1 and weighted ΔDC correction term from step I-ΔS3 to output of inverse SA-DCT from step I-ΔS2

```
for (x=0; x<8; x++)
   if(pels_height[x]==0) continue;
   for (y=0; y<0; y++)
      if(bin_shape[y][x])
         f[y][x] += mean_value-(corr_term/sqrt(pels_height[x]));
```


© ISO/IEC 2004 – All rights reserved

Annex B
(normative)

Variable length codes and arithmetic decoding

B.1 Variable length codes

B.1.1 Macroblock type

Table B.1 — Macroblock types and included data elements for I- and P-, and S-VOPs

VOP type	mb type	Name	not_coded	mcbpc	mcsel	cbpy	dquant	mvd	mvd_{2-4}
P	not coded	-	1						
P	0	inter	1	1		1		1	
P	1	inter+q	1	1		1	1	1	
P	2	inter4v	1	1		1		1	1
P	3	intra	1	1		1			
P	4	intra+q	1	1		1	1		
P	stuffing	-	1	1					
I	3	intra		1		1			
I	4	intra+q		1		1	1		
I	stuffing	-		1					
S (update)	not_coded	-	1						
S (update)	0	inter	1	1		1			
S (update)	1	inter+q	1	1		1	1		
S (update)	3	intra	1	1		1			
S (update)	4	intra+q	1	1		1	1		
S (update)	stuffing	-	1	1					
S (piece)	3	intra		1		1			
S (piece)	4	intra+q		1		1	1		
S (piece)	stuffing	-		1					
S (GMC)	not coded	-	1						
S (GMC)	0	inter	1	1	1	1		1	
S (GMC)	1	inter+q	1	1	1	1	1	1	
S(GMC)	2	inter4v	1	1		1		1	1
S (GMC)	3	intra	1	1		1			
S (GMC)	4	intra+q	1	1		1	1		
S (GMC)	stuffing	-	1	1					

NOTE "1" means that the item is present in the macroblock
S (piece) indicates S-VOPs with low_latency_sprite_enable == 1 and sprite_transmit_mode == "piece"
S (update) indicates S-VOPs with low_latency_sprite_enable == 1 and sprite_transmit_mode == "update"
S (GMC) indicates S-VOPs with sprite_enable == "GMC"

© ISO/IEC 2004 – All rights reserved

Table B.2 — Macroblock types and included data elements for a P-VOP (scalability && ref_select_code == '11')

VOP Type	mb_type	Name	not_coded	mcbpc	cbpy	dquant	MVD	MVD_{2-4}
P	not coded	-	1					
P	0	INTER	1	1	1			
P	1	INTER+Q	1	1	1	1		
P	3	INTRA	1	1	1			
P	4	INTRA+Q	1	1	1	1		
P	stuffing	-	1	1				
NOTE "1" means that the item is present in the macroblock								

Table B.3 — VLC table for modb

Code	cbpb	mb_type
1		
01		1
00	1	1

Table B.4 — mb_type and included data elements in coded macroblocks in B-VOPs (ref_select_code != '00'||scalability=='0')

Code	dbquant	mvd_f	mvd_b	mvdb	mb_type
1				1	direct
01	1	1	1		interpolate mc+q
001	1		1		backward mc+q
0001	1	1			forward mc+q

Table B.5 — mb_type and included data elements in coded macroblocks in B-VOPs (ref_select_code == '00'&&scalability!='0')

Code	dbquant	mvd_f	mvd_b	mb_type
01	1	1		interpolate mc+q
001	1			backward mc+q
1	1	1		forward mc+q

© ISO/IEC 2004 – All rights reserved

B.1.2 Macroblock pattern

Table B.6 — VLC table for mcbpc for I-VOPs and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode==”piece”

Code	mbtype	cbpc (56)
1	3	00
001	3	01
010	3	10
011	3	11
0001	4	00
0000 01	4	01
0000 10	4	10
0000 11	4	11
0000 0000 1	Stuffing	--

Table B.7 — VLC table for mcbpc for P-VOPs and S-VOPs with low_latence_sprite_enable==1 and sprite_transmit_mode==”update”, and S(GMC)-VOPs

Code	MB type	cbpc (56)
1	0	00
0011	0	01
0010	0	10
0001 01	0	11
011	1	00
0000 111	1	01
0000 110	1	10
0000 0010 1	1	11
010	2	00
0000 101	2	01
0000 100	2	10
0000 0101	2	11
0001 1	3	00
0000 0100	3	01
0000 0011	3	10
0000 011	3	11
0001 00	4	00
0000 0010 0	4	01
0000 0001 1	4	10
0000 0001 0	4	11
0000 0000 1	Stuffing	--

© ISO/IEC 2004 – All rights reserved

Table B.8 — VLC table for cbpy in the case of four non-transparent blocks

Code	cbpy(intra-MB) (12 34)	cbpy(inter-MB), (12 34)
0011	00 00	11 11
0010 1	00 01	11 10
0010 0	00 10	11 01
1001	00 11	11 00
0001 1	01 00	10 11
0111	01 01	10 10
0000 10	01 10	10 01
1011	01 11	10 00
0001 0	10 00	01 11
0000 11	10 01	01 10
0101	10 10	01 01
1010	10 11	01 00
0100	11 00	00 11
1000	11 01	00 10
0110	11 10	00 01
11	11 11	00 00

© ISO/IEC 2004 – All rights reserved

Table B.9 — VLC table for cbpy in the case of three non transparent blocks

Code	cbpy (intra-MB)	cbpy (inter-MB)
011	000	111
000001	001	110
00001	010	101
010	011	100
00010	100	011
00011	101	010
001	110	001
1	111	000

Table B.10 — VLC table for cbpy in the case of two non transparent blocks

Code	cbpy (intra-MB)	cbpy (inter-MB)
0001	00	11
001	01	10
01	10	01
1	11	00

Table B.11 — VLC table for cbpy in the case of one non transparent block

Code	cbpy (intra-MB)	cbpy (inter-MB)
01	0	1
1	1	0

B.1.3 Motion vector

Table B.12 — VLC table for MVD

Codes	Vector differences
0000 0000 0010 1	-16
0000 0000 0011 1	-15.5
0000 0000 0101	-15
0000 0000 0111	-14.5
0000 0000 1001	-14
0000 0000 1011	-13.5
0000 0000 1101	-13
0000 0000 1111	-12.5
0000 0001 001	-12
0000 0001 011	-11.5
0000 0001 101	-11
0000 0001 111	-10.5
0000 0010 001	-10

© ISO/IEC 2004 – All rights reserved

0000 0010 011	-9.5
0000 0010 101	-9
0000 0010 111	-8.5
0000 0011 001	-8
0000 0011 011	-7.5
0000 0011 101	-7
0000 0011 111	-6.5
0000 0100 001	-6
0000 0100 011	-5.5
0000 0100 11	-5
0000 0101 01	-4.5
0000 0101 11	-4
0000 0111	-3.5
0000 1001	-3
0000 1011	-2.5
0000 111	-2
0001 1	-1.5
0011	-1
011	-0.5
1	0
010	0.5
0010	1
0001 0	1.5
0000 110	2
0000 1010	2.5
0000 1000	3
0000 0110	3.5
0000 0101 10	4
0000 0101 00	4.5
0000 0100 10	5
0000 0100 010	5.5
0000 0100 000	6
0000 0011 110	6.5
0000 0011 100	7
0000 0011 010	7.5
0000 0011 000	8
0000 0010 110	8.5
0000 0010 100	9
0000 0010 010	9.5
0000 0010 000	10
0000 0001 110	10.5
0000 0001 100	11
0000 0001 010	11.5
0000 0001 000	12
0000 0000 1110	12.5
0000 0000 1100	13
0000 0000 1010	13.5
0000 0000 1000	14

© ISO/IEC 2004 – All rights reserved

0000 0000 0110	14.5
0000 0000 0100	15
0000 0000 0011 0	15.5
0000 0000 0010 0	16

NOTE The last code shall not be used when vop_fcode=1.

B.1.4 DCT coefficients

Table B.13 — Variable length codes for dct_dc_size_luminance

Variable length code	dct_dc_size_luminance
011	0
11	1
10	2
010	3
001	4
0001	5
0000 1	6
0000 01	7
0000 001	8
0000 0001	9
0000 0000 1	10
0000 0000 01	11
0000 0000 001	12

NOTE The variable length code for dct_dc_size_luminance of 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

Table B.14 — Variable length codes for dct_dc_size_chrominance

Variable length code	dct_dc_size_chrominance
11	0
10	1
01	2
001	3
0001	4
0000 1	5
0000 01	6
0000 001	7
0000 0001	8
0000 0000 1	9
0000 0000 01	10
0000 0000 001	11
0000 0000 0001	12

NOTE The variable length code for dct_dc_size_chrominance of 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

© ISO/IEC 2004 – All rights reserved

Table B.15 — Differential DC additional codes

Additional code	Differential DC	Size
000000000000 to 011111111111 *	-4095 to -2048	12
00000000000 to 01111111111 *	-2047 to -1024	11
0000000000 to 0111111111 *	-1023 to -512	10
000000000 to 011111111 *	-511 to -256	9
00000000 to 01111111	-255 to -128	8
0000000 to 0111111	-127 to -64	7
000000 to 011111	-63 to -32	6
00000 to 01111	-31 to -16	5
0000 to 0111	-15 to -8	4
000 to 011	-7 to -4	3
00 to 01	-3 to -2	2
0	-1	1
	0	0
1	1	1
10 to 11	2 to 3	2
100 to 111	4 to 7	3
1000 to 1111	8 to 15	4
10000 to 11111	16 to 31	5
100000 to 111111	32 to 63	6
1000000 to 1111111	64 to 127	7
10000000 to 11111111	128 to 255	8
100000000 to 111111111 *	256 to 511	9
1000000000 to 1111111111 *	512 to 1023	10
10000000000 to 11111111111 *	1024 to 2047	11
100000000000 to 111111111111 *	2048 to 4095	12

NOTE 1 The variable length code for "Size" of 10, 11 and 12 are not valid for any object types where the pixel depth is 8 bits. They shall not be present in a bitstream conforming to these object types.

NOTE 2 In cases where dct_dc_size is greater than 8, marked '*' in Table B.15, a marker bit is inserted after the dct_dc_additional_code to prevent start code emulations.

Table B.16 — VLC Table for Intra Luminance and Chrominance Tcoef

VLC CODE	LAST	RUN	LEVEL
10s	0	0	1
1111 s	0	0	3
0101 01s	0	0	6
0010 111s	0	0	9
0001 1111 s	0	0	10
0001 0010 1s	0	0	13
0001 0010 0s	0	0	14
0000 1000 01s	0	0	17
0000 1000 00s	0	0	18
0000 0000 111s	0	0	21
0000 0000 110s	0	0	22
0000 0100 000s	0	0	23
110s	0	0	2
0101 00s	0	1	2

VLC CODE	LAST	RUN	LEVEL
0111 s	1	0	1
0000 1100 1s	0	11	1
0000 0000 101s	1	0	6
0011 11s	1	1	1
0000 0000 100s	1	0	7
0011 10s	1	2	1
0011 01s	0	5	1
0011 00s	1	0	2
0010 011s	1	5	1
0010 010s	0	6	1
0010 001s	1	3	1
0010 000s	1	4	1
0001 1010 s	1	9	1
0001 1001 s	0	8	1

© ISO/IEC 2004 – All rights reserved

VLC CODE	LAST	RUN	LEVEL
0001 1110 s	0	0	11
0000 0011 11s	0	0	19
0000 0100 001s	0	0	24
0000 0101 0000s	0	0	25
1110 s	0	1	1
0001 1101 s	0	0	12
0000 0011 10s	0	0	20
0000 0101 0001s	0	0	26
0110 1s	0	0	4
0001 0001 1s	0	0	15
0000 0011 01s	0	1	7
0110 0s	0	0	5
0001 0001 0s	0	4	2
0000 0101 0010s	0	0	27
0101 1s	0	2	1
0000 0011 00s	0	2	4
0000 0101 0011s	0	1	9
0100 11s	0	0	7
0000 0010 11s	0	3	4
0000 0101 0100s	0	6	3
0100 10s	0	0	8
0000 0010 10s	0	4	3
0100 01s	0	3	1
0000 0010 01s	0	8	2
0100 00s	0	4	1
0000 0010 00s	0	5	3
0010 110s	0	1	3
0000 0101 0101s	0	1	10
0010 101s	0	2	2
0010 100s	0	7	1
0001 1100 s	0	1	4
0001 1011 s	0	3	2
0001 0000 1s	0	0	16
0001 0000 0s	0	1	5
0000 1111 1s	0	1	6
0000 1111 0s	0	2	3
0000 1110 1s	0	3	3
0000 1110 0s	0	5	2
0000 1101 1s	0	6	2
0000 1101 0s	0	7	2
0000 0100 010s	0	1	8
0000 0100 011s	0	9	2
0000 0101 0110s	0	2	5
0000 0101 0111s	0	7	3

VLC CODE	LAST	RUN	LEVEL
0001 1000 s	0	9	1
0001 0111 s	0	10	1
0001 0110 s	1	0	3
0001 0101 s	1	6	1
0001 0100 s	1	7	1
0001 0011 s	1	8	1
0000 1100 0s	0	12	1
0000 1011 1s	1	0	4
0000 1011 0s	1	1	2
0000 1010 1s	1	10	1
0000 1010 0s	1	11	1
0000 1001 1s	1	12	1
0000 1001 0s	1	13	1
0000 1000 1s	1	14	1
0000 0001 11s	0	13	1
0000 0001 10s	1	0	5
0000 0001 01s	1	1	3
0000 0001 00s	1	2	2
0000 0100 100s	1	3	2
0000 0100 101s	1	4	2
0000 0100 110s	1	15	1
0000 0100 111s	1	16	1
0000 0101 1000s	0	14	1
0000 0101 1001s	1	0	8
0000 0101 1010s	1	5	2
0000 0101 1011s	1	6	2
0000 0101 1100s	1	17	1
0000 0101 1101s	1	18	1
0000 0101 1110s	1	19	1
0000 0101 1111s	1	20	1
0000 011	escape		

© ISO/IEC 2004 – All rights reserved

Table B.17 — VLC table for Inter Lumimance and Chrominance Tcoef

VLC CODE	LAST	RUN	LEVEL
10s	0	0	1
1111 s	0	0	2
0101 01s	0	0	3
0010 111s	0	0	4
0001 1111 s	0	0	5
0001 0010 1s	0	0	6
0001 0010 0s	0	0	7
0000 1000 01s	0	0	8
0000 1000 00s	0	0	9
0000 0000 111s	0	0	10
0000 0000 110s	0	0	11
0000 0100 000s	0	0	12
110s	0	1	1
0101 00s	0	1	2
0001 1110 s	0	1	3
0000 0011 11s	0	1	4
0000 0100 001s	0	1	5
0000 0101 0000s	0	1	6
1110 s	0	2	1
0001 1101 s	0	2	2
0000 0011 10s	0	2	3
0000 0101 0001s	0	2	4
0110 1s	0	3	1
0001 0001 1s	0	3	2
0000 0011 01s	0	3	3
0110 0s	0	4	1
0001 0001 0s	0	4	2
0000 0101 0010s	0	4	3
0101 1s	0	5	1
0000 0011 00s	0	5	2
0000 0101 0011s	0	5	3
0100 11s	0	6	1
0000 0010 11s	0	6	2
0000 0101 0100s	0	6	3
0100 10s	0	7	1
0000 0010 10s	0	7	2
0100 01s	0	8	1
0000 0010 01s	0	8	2
0100 00s	0	9	1
0000 0010 00s	0	9	2
0010 110s	0	10	1
0000 0101 0101s	0	10	2
0010 101s	0	11	1
0010 100s	0	12	1
0001 1100 s	0	13	1
0001 1011 s	0	14	1

VLC CODE	LAST	RUN	LEVEL
0111 s	1	0	1
0000 1100 1s	1	0	2
0000 0000 101s	1	0	3
0011 11s	1	1	1
0000 0000 100s	1	1	2
0011 10s	1	2	1
0011 01s	1	3	1
0011 00s	1	4	1
0010 011s	1	5	1
0010 010s	1	6	1
0010 001s	1	7	1
0010 000s	1	8	1
0001 1010 s	1	9	1
0001 1001 s	1	10	1
0001 1000 s	1	11	1
0001 0111 s	1	12	1
0001 0110 s	1	13	1
0001 0101 s	1	14	1
0001 0100 s	1	15	1
0001 0011 s	1	16	1
0000 1100 0s	1	17	1
0000 1011 1s	1	18	1
0000 1011 0s	1	19	1
0000 1010 1s	1	20	1
0000 1010 0s	1	21	1
0000 1001 1s	1	22	1
0000 1001 0s	1	23	1
0000 1000 1s	1	24	1
0000 0001 11s	1	25	1
0000 0001 10s	1	26	1
0000 0001 01s	1	27	1
0000 0001 00s	1	28	1
0000 0100 100s	1	29	1
0000 0100 101s	1	30	1
0000 0100 110s	1	31	1
0000 0100 111s	1	32	1
0000 0101 1000s	1	33	1
0000 0101 1001s	1	34	1
0000 0101 1010s	1	35	1
0000 0101 1011s	1	36	1
0000 0101 1100s	1	37	1
0000 0101 1101s	1	38	1
0000 0101 1110s	1	39	1
0000 0101 1111s	1	40	1
0000 011	escape		

© ISO/IEC 2004 – All rights reserved

0001 0000 1s	0	15	1
0001 0000 0s	0	16	1
0000 1111 1s	0	17	1
0000 1111 0s	0	18	1
0000 1110 1s	0	19	1
0000 1110 0s	0	20	1
0000 1101 1s	0	21	1
0000 1101 0s	0	22	1
0000 0100 010s	0	23	1
0000 0100 011s	0	24	1
0000 0101 0110s	0	25	1
0000 0101 0111s	0	26	1

Table B.18 — FLC table for RUNS and LEVELS

Code	Run
000 000	0
000 001	1
	63

a) FLC code for RUN

Code	Level
forbidden	-2048
1000 0000 0001	-2047
.	.
1111 1111 1110	-2
1111 1111 1111	-1
forbidden	0
0000 0000 0001	1
0000 0000 0010	2
.	.
0111 1111 1111	2047

b) FLC code for LEVEL

Code	Level
forbidden	-128
1000 0001	-127
.	.
1111 1110	-2
1111 1111	-1
forbidden	0
0000 0001	1
0000 0010	2
.	.
0111 1111	127

c) FLC code for LEVEL when short_video_header is 1

Table B.19 — ESCL(a), LMAX values of intra macroblocks

LAST	RUN	LMAX
0	0	27
0	1	10
0	2	5
0	3	4
0	4-7	3
0	8-9	2
0	10-14	1
0	others	N/A

LAST	RUN	LMAX
1	0	8
1	1	3
1	2-6	2
1	7-20	1
1	others	N/A

© ISO/IEC 2004 – All rights reserved

Table B.20 — ESCL(b), LMAX values of inter macroblocks

LAST	RUN	LMAX
0	0	12
0	1	6
0	2	4
0	3-6	3
0	7-10	2
0	11-26	1
0	others	N/A

LAST	RUN	LMAX
1	0	3
1	1	2
1	2-40	1
1	others	N/A

Table B.21 — ESCR(a), RMAX values of intra macroblocks

LAST	LEVEL	RMAX
0	1	14
0	2	9
0	3	7
0	4	3
0	5	2
0	6-10	1
0	11-27	0
0	others	N/A

LAST	LEVEL	RMAX
1	1	20
1	2	6
1	3	1
1	4-8	0
1	others	N/A

Table B.22 — ESCR(b), RMAX values of inter macroblocks

LAST	LEVEL	RMAX
0	1	26
0	2	10
0	3	6
0	4	2
0	5-6	1
0	7-12	0
0	others	N/A

LAST	LEVEL	RMAX
1	1	40
1	2	1
1	3	0
1	others	N/A

Table B.23 — RVLC table for Tcoef

	intra			inter				
INDEX	LAST	RUN	LEVEL	LAST	RUN	LEVEL	BITS	VLC_CODE
0	0	0	1	0	0	1	4	110s
1	0	0	2	0	1	1	4	111s
2	0	1	1	0	0	2	5	0001s
3	0	0	3	0	2	1	5	1010s
4	1	0	1	1	0	1	5	1011s
5	0	2	1	0	0	3	6	00100s
6	0	3	1	0	3	1	6	00101s
7	0	1	2	0	4	1	6	01000s

© ISO/IEC 2004 – All rights reserved

8	0	0	4	0	5	1	6	01001s
9	1	1	1	1	1	1	6	10010s
10	1	2	1	1	2	1	6	10011s
11	0	4	1	0	1	2	7	001100s
12	0	5	1	0	6	1	7	001101s
13	0	0	5	0	7	1	7	010100s
14	0	0	6	0	8	1	7	010101s
15	1	3	1	1	3	1	7	011000s
16	1	4	1	1	4	1	7	011001s
17	1	5	1	1	5	1	7	100010s
18	1	6	1	1	6	1	7	100011s
19	0	6	1	0	0	4	8	0011100s
20	0	7	1	0	2	2	8	0011101s
21	0	2	2	0	9	1	8	0101100s
22	0	1	3	0	10	1	8	0101101s
23	0	0	7	0	11	1	8	0110100s
24	1	7	1	1	7	1	8	0110101s
25	1	8	1	1	8	1	8	0111000s
26	1	9	1	1	9	1	8	0111001s
27	1	10	1	1	10	1	8	1000010s
28	1	11	1	1	11	1	8	1000011s
29	0	8	1	0	0	5	9	00111100s
30	0	9	1	0	0	6	9	00111101s
31	0	3	2	0	1	3	9	01011100s
32	0	4	2	0	3	2	9	01011101s
33	0	1	4	0	4	2	9	01101100s
34	0	1	5	0	12	1	9	01101101s
35	0	0	8	0	13	1	9	01110100s
36	0	0	9	0	14	1	9	01110101s
37	1	0	2	1	0	2	9	01111000s
38	1	12	1	1	12	1	9	01111001s
39	1	13	1	1	13	1	9	10000010s
40	1	14	1	1	14	1	9	10000011s
41	0	10	1	0	0	7	10	001111100s
42	0	5	2	0	1	4	10	001111101s
43	0	2	3	0	2	3	10	010111100s
44	0	3	3	0	5	2	10	010111101s
45	0	1	6	0	15	1	10	011011100s
46	0	0	10	0	16	1	10	011011101s
47	0	0	11	0	17	1	10	011101100s
48	1	1	2	1	1	2	10	011101101s
49	1	15	1	1	15	1	10	011110100s
50	1	16	1	1	16	1	10	011110101s
51	1	17	1	1	17	1	10	011111000s
52	1	18	1	1	18	1	10	011111001s
53	1	19	1	1	19	1	10	100000010s
54	1	20	1	1	20	1	10	100000011s
55	0	11	1	0	0	8	11	0011111100s

© ISO/IEC 2004 – All rights reserved

56	0	12	1	0	0	9	11	0011111101s
57	0	6	2	0	1	5	11	0101111100s
58	0	7	2	0	3	3	11	0101111101s
59	0	8	2	0	6	2	11	0110111100s
60	0	4	3	0	7	2	11	0110111101s
61	0	2	4	0	8	2	11	0111011100s
62	0	1	7	0	9	2	11	0111011101s
63	0	0	12	0	18	1	11	0111101100s
64	0	0	13	0	19	1	11	0111101101s
65	0	0	14	0	20	1	11	0111110100s
66	1	21	1	1	21	1	11	0111110101s
67	1	22	1	1	22	1	11	0111111000s
68	1	23	1	1	23	1	11	0111111001s
69	1	24	1	1	24	1	11	1000000010s
70	1	25	1	1	25	1	11	1000000011s
71	0	13	1	0	0	10	12	00111111100s
72	0	9	2	0	0	11	12	00111111101s
73	0	5	3	0	1	6	12	01011111100s
74	0	6	3	0	2	4	12	01011111101s
75	0	7	3	0	4	3	12	01101111100s
76	0	3	4	0	5	3	12	01101111101s
77	0	2	5	0	10	2	12	01110111100s
78	0	2	6	0	21	1	12	01110111101s
79	0	1	8	0	22	1	12	01111011100s
80	0	1	9	0	23	1	12	01111011101s
81	0	0	15	0	24	1	12	01111101100s
82	0	0	16	0	25	1	12	01111101101s
83	0	0	17	0	26	1	12	01111110100s
84	1	0	3	1	0	3	12	01111110101s
85	1	2	2	1	2	2	12	01111111000s
86	1	26	1	1	26	1	12	01111111001s
87	1	27	1	1	27	1	12	10000000010s
88	1	28	1	1	28	1	12	10000000011s
89	0	10	2	0	0	12	13	001111111100s
90	0	4	4	0	1	7	13	001111111101s
91	0	5	4	0	2	5	13	010111111100s
92	0	6	4	0	3	4	13	010111111101s
93	0	3	5	0	6	3	13	011011111100s
94	0	4	5	0	7	3	13	011011111101s
95	0	1	10	0	11	2	13	011101111100s
96	0	0	18	0	27	1	13	011101111101s
97	0	0	19	0	28	1	13	011110111100s
98	0	0	22	0	29	1	13	011110111101s
99	1	1	3	1	1	3	13	011111011100s
100	1	3	2	1	3	2	13	011111011101s
101	1	4	2	1	4	2	13	011111101100s
102	1	29	1	1	29	1	13	011111101101s
103	1	30	1	1	30	1	13	011111110100s

© ISO/IEC 2004 – All rights reserved

104	1	31	1	1	31	1	13	011111110101s
105	1	32	1	1	32	1	13	011111111000s
106	1	33	1	1	33	1	13	011111111001s
107	1	34	1	1	34	1	13	100000000010s
108	1	35	1	1	35	1	13	100000000011s
109	0	14	1	0	0	13	14	0011111111100s
110	0	15	1	0	0	14	14	0011111111101s
111	0	11	2	0	0	15	14	0101111111100s
112	0	8	3	0	0	16	14	0101111111101s
113	0	9	3	0	1	8	14	0110111111100s
114	0	7	4	0	3	5	14	0110111111101s
115	0	3	6	0	4	4	14	0111011111100s
116	0	2	7	0	5	4	14	0111011111101s
117	0	2	8	0	8	3	14	0111101111100s
118	0	2	9	0	12	2	14	0111101111101s
119	0	1	11	0	30	1	14	0111110111100s
120	0	0	20	0	31	1	14	0111110111101s
121	0	0	21	0	32	1	14	0111111011100s
122	0	0	23	0	33	1	14	0111111011101s
123	1	0	4	1	0	4	14	0111111101100s
124	1	5	2	1	5	2	14	0111111101101s
125	1	6	2	1	6	2	14	0111111110100s
126	1	7	2	1	7	2	14	0111111110101s
127	1	8	2	1	8	2	14	0111111111000s
128	1	9	2	1	9	2	14	0111111111001s
129	1	36	1	1	36	1	14	1000000000010s
130	1	37	1	1	37	1	14	1000000000011s
131	0	16	1	0	0	17	15	00111111111100s
132	0	17	1	0	0	18	15	00111111111101s
133	0	18	1	0	1	9	15	01011111111100s
134	0	8	4	0	1	10	15	01011111111101s
135	0	5	5	0	2	6	15	01101111111100s
136	0	4	6	0	2	7	15	01101111111101s
137	0	5	6	0	3	6	15	01110111111100s
138	0	3	7	0	6	4	15	01110111111101s
139	0	3	8	0	9	3	15	01111011111100s
140	0	2	10	0	13	2	15	01111011111101s
141	0	2	11	0	14	2	15	01111101111100s
142	0	1	12	0	15	2	15	01111101111101s
143	0	1	13	0	16	2	15	01111110111100s
144	0	0	24	0	34	1	15	01111110111101s
145	0	0	25	0	35	1	15	01111111011100s
146	0	0	26	0	36	1	15	01111111011101s
147	1	0	5	1	0	5	15	01111111101100s
148	1	1	4	1	1	4	15	01111111101101s
149	1	10	2	1	10	2	15	01111111110100s
150	1	11	2	1	11	2	15	01111111110101s
151	1	12	2	1	12	2	15	01111111111000s

© ISO/IEC 2004 – All rights reserved

152	1	38	1	1	38	1	15	01111111111001s
153	1	39	1	1	39	1	15	10000000000010s
154	1	40	1	1	40	1	15	10000000000011s
155	0	0	27	0	0	19	16	001111111111100s
156	0	3	9	0	3	7	16	001111111111101s
157	0	6	5	0	4	5	16	010111111111100s
158	0	7	5	0	7	4	16	010111111111101s
159	0	9	4	0	17	2	16	011011111111100s
160	0	12	2	0	37	1	16	011011111111101s
161	0	19	1	0	38	1	16	011101111111100s
162	1	1	5	1	1	5	16	011101111111101s
163	1	2	3	1	2	3	16	011110111111100s
164	1	13	2	1	13	2	16	011110111111101s
165	1	41	1	1	41	1	16	011111011111100s
166	1	42	1	1	42	1	16	011111011111101s
167	1	43	1	1	43	1	16	011111101111100s
168	1	44	1	1	44	1	16	011111101111101s
169	ESCAPE						5	0000s

ESCAPE code is added at the beginning and the end of these fixed-length codes for realizing two-way decode as shown below. A marker bit is inserted before and after the 11-bit-LEVEL in order to avoid the resync_marker emulation.

ESCAPE	LAST	RUN	marker bit	LEVEL	marker bit	ESCAPE
00001	x	xxxxxx	1	xxxxxxxxxxx	1	0000s

NOTE There are two types for ESCAPE added at the end of these fixed-length codes, and codewords are "0000s". Also, S=0 : LEVEL is positive and S=1 : LEVEL is negative.

Table B.24 — FLC table for RUN

RUN	CODE
0	000000
1	000001
2	000010
:	:
63	111111

Table B.25 — FLC table for LEVEL

LEVEL	CODE
0	FORBIDDEN
1	00000000001
2	00000000010
:	:
2047	11111111111

© ISO/IEC 2004 – All rights reserved

B.1.5 Shape Coding

Table B.26 — Meaning of shape mode

Index	Shape mode
0	= “MVDs==0 && No Update”
1	= “MVDs!=0 && No Update”
2	transparent
3	opaque
4	“intraCAE”
5	“interCAE && MVDs==0”
6	“interCAE && MVDs!=0”

Table B.27 — bab_type for I-VOP

Index	(2)	(3)	(4)	Index	(2)	(3)	(4)
0	1	001	01	41	001	01	1
1	001	01	1	42	1	01	001
2	01	001	1	43	001	1	01
3	1	001	01	44	001	01	1
4	1	01	001	45	1	01	001
5	1	01	001	46	001	01	1
6	1	001	01	47	01	001	1
7	1	01	001	48	1	01	001
8	01	001	1	49	001	01	1
9	001	01	1	50	01	001	1
10	1	01	001	51	1	001	01
11	1	01	001	52	001	1	01
12	001	01	1	53	01	001	1
13	1	01	001	54	1	001	01
14	01	1	001	55	01	001	1
15	001	01	1	56	01	001	1
16	1	01	001	57	1	01	001
17	1	01	001	58	1	01	001
18	01	001	1	59	1	01	001
19	1	01	001	60	1	01	001
20	001	01	1	61	1	01	001
21	01	001	1	62	01	001	1
22	1	01	001	63	1	01	001
23	001	01	1	64	001	01	1
24	01	001	1	65	001	01	1
25	001	01	1	66	01	001	1
26	001	01	1	67	001	1	01
27	1	01	001	68	001	1	01
28	1	01	001	69	01	001	1
29	1	01	001	70	001	1	01
30	1	01	001	71	001	01	1
31	1	01	001	72	1	001	01

© ISO/IEC 2004 – All rights reserved

32	1	01	001	73	001	01	1
33	1	01	001	74	01	001	1
34	1	01	001	75	01	001	1
35	001	01	1	76	001	1	01
36	1	01	001	77	001	01	1
37	001	01	1	78	1	001	01
38	001	01	1	79	001	1	01
39	1	01	001	80	001	01	1
40	001	1	01				

Table B.28 — bab_type for P-VOP, B-VOP, and S(GMC)-VOP

		bab_type in current VOP						
		0	**1**	**2**	**3**	**4**	**5**	**6**
bab_type in reference VOP	**0**	1	01	00001	000001	0001	0010	0011
	1	01	1	00001	000001	001	0000001	0001
	2	0001	001	1	000001	01	0000001	00001
	3	1	0001	000001	001	01	0000001	00001
	4	011	001	0001	00001	1	000001	010
	5	01	0001	00001	000001	001	11	10
	6	001	0001	00001	000001	01	10	11

Table B.29 — VLC table for MVDs

MVDs	Codes
0	0
±1	10s
±2	110s
±3	1110s
±4	11110s
±5	111110s
±6	1111110s
±7	11111110s
±8	111111110s
±9	1111111110s
±10	11111111110s
±11	111111111110s
±12	1111111111110s
±13	11111111111110s
±14	111111111111110s
±15	1111111111111110s
±16	11111111111111110s

© ISO/IEC 2004 – All rights reserved

Table B.30 — VLC table for MVDs (Horizontal element is 0)

MVDs	Codes
±1	0s
±2	10s
±3	110s
±4	1110s
±5	11110s
±6	111110s
±7	1111110s
±8	11111110s
±9	111111110s
±10	1111111110s
±11	11111111110s
±12	111111111110s
±13	1111111111110s
±14	11111111111110s
±15	111111111111110s
±16	1111111111111110s
s: sign bit (if MVDs is positive s="1", otherwise s="0").	

Table B.31 — VLC for conv_ratio

conv_ratio	Code
1	0
2	10
4	11

These tables contain the probabilities for a binary alpha pixel being equal to 0 for intra and inter shape coding using CAE. All probabilities are normalised to the range [1,65535].

As an example, given an INTRA context number C, the probability that the pixel is zero is given by intra_prob[C].

Table B.32 — Probabilities for arithmetic decoding of shape

```
USInt intra_prob[1024] = {
65267,16468,65003,17912,64573,8556,64252,5653,40174,3932,29789,277,45152,1140,32768,2043,
4499,80,6554,1144,21065,465,32768,799,5482,183,7282,264,5336,99,6554,563,
54784,30201,58254,9879,54613,3069,32768,58495,32768,32768,32768,2849,58982,54613,32768,12892,
31006,1332,49152,3287,60075,350,32768,712,39322,760,32768,354,52659,432,61854,150,
64999,28362,65323,42521,63572,32768,63677,18319,4910,32768,64238,434,53248,32768,61865,13590,
16384,32768,13107,333,32768,32768,32768,32768,32768,32768,1074,780,25058,5461,6697,233,
62949,30247,63702,24638,59578,32768,32768,42257,32768,32768,49152,546,62557,32768,54613,19258,
62405,32569,64600,865,60495,10923,32768,898,34193,24576,64111,341,47492,5231,55474,591,
65114,60075,64080,5334,65448,61882,64543,13209,54906,16384,35289,4933,48645,9614,55351,7318,
49807,54613,32768,32768,50972,32768,32768,32768,15159,1928,2048,171,3093,8,6096,74,
32768,60855,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,55454,32768,57672,
32768,16384,32768,21845,32768,32768,32768,32768,32768,32768,32768,5041,28440,91,32768,45,
```

© ISO/IEC 2004 – All rights reserved

65124,10923,64874,5041,65429,57344,63435,48060,61440,32768,63488,24887,59688,3277,63918,14021,
32768,32768,32768,32768,32768,32768,32768,32768,690,32768,32768,1456,32768,32768,8192,728,
32768,32768,58982,17944,65237,54613,32768,2242,32768,32768,32768,42130,49152,57344,58254,16740,
32768,10923,54613,182,32768,32768,32768,7282,49152,32768,32768,5041,63295,1394,55188,77,
63672,6554,54613,49152,64558,32768,32768,5461,64142,32768,32768,32768,62415,32768,32768,16384,
1481,438,19661,840,33654,3121,64425,6554,4178,2048,32768,2260,5226,1680,32768,565,
60075,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,
16384,261,32768,412,16384,636,32768,4369,23406,4328,32768,524,15604,560,32768,676,
49152,32768,49152,32768,32768,32768,64572,32768,32768,32768,54613,32768,32768,32768,32768,32768,
4681,32768,5617,851,32768,32768,59578,32768,32768,32768,3121,3121,49152,32768,6554,10923,
32768,32768,54613,14043,32768,32768,32768,3449,32768,32768,32768,32768,32768,32768,32768,32768,
57344,32768,57344,3449,32768,32768,32768,3855,58982,10923,32768,239,62259,32768,49152,85,
58778,23831,62888,20922,64311,8192,60075,575,59714,32768,57344,40960,62107,4096,61943,3921,
39862,15338,32768,1524,45123,5958,32768,58982,6669,930,1170,1043,7385,44,8813,5011,
59578,29789,54613,32768,32768,32768,32768,32768,32768,32768,32768,32768,58254,56174,32768,32768,
64080,25891,49152,22528,32768,2731,32768,10923,10923,3283,32768,1748,17827,77,32768,108,
62805,32768,62013,42612,32768,32768,61681,16384,58982,60075,62313,58982,65279,58982,62694,62174,
32768,32768,10923,950,32768,32768,32768,32768,5958,32768,38551,1092,11012,39322,13705,2072,
54613,32768,32768,11398,32768,32768,32768,145,32768,32768,32768,29789,60855,32768,61681,54792,
32768,32768,32768,17348,32768,32768,32768,8192,57344,16384,32768,3582,52581,580,24030,303,
62673,37266,65374,6197,62017,32768,49152,299,54613,32768,32768,32768,35234,119,32768,3855,
31949,32768,32768,49152,16384,32768,32768,32768,24576,32768,49152,32768,17476,32768,32768,57445,
51200,50864,54613,27949,60075,20480,32768,57344,32768,32768,32768,32768,32768,45875,32768,32768,
11498,3244,24576,482,16384,1150,32768,16384,7992,215,32768,1150,23593,927,32768,993,
65353,32768,65465,46741,41870,32768,64596,59578,62087,32768,12619,23406,11833,32768,47720,17476,
32768,32768,2621,6554,32768,32768,32768,32768,32768,32768,5041,32768,16384,32768,4096,2731,
63212,43526,65442,47124,65410,35747,60304,55858,60855,58982,60075,19859,35747,63015,64470,25432,
58689,1118,64717,1339,24576,32768,32768,1257,53297,1928,32768,33,52067,3511,62861,453,
64613,32768,32768,32768,64558,32768,32768,2731,49152,32768,32768,32768,61534,32768,32768,35747,
32768,32768,32768,32768,13107,32768,32768,32768,32768,32768,32768,32768,20480,32768,32768,32768,
32768,32768,32768,54613,40960,5041,32768,32768,32768,32768,32768,3277,64263,57592,32768,3121,
32768,32768,32768,32768,32768,10923,32768,32768,32768,8192,32768,32768,5461,6899,32768,1725,
63351,3855,63608,29127,62415,7282,64626,60855,32768,32768,60075,5958,44961,32768,61866,53718,
32768,32768,32768,32768,32768,32768,6554,32768,32768,32768,32768,32768,2521,978,32768,1489,
58254,32768,58982,61745,21845,32768,54613,58655,60075,32768,49152,16274,50412,64344,61643,43987,
32768,32768,32768,1638,32768,32768,32768,24966,54613,32768,32768,2427,46951,32768,17970,654,
65385,27307,60075,26472,64479,32768,32768,4681,61895,32768,32768,16384,58254,32768,32768,6554,
37630,3277,54613,6554,4965,5958,4681,32768,42765,16384,32768,21845,22827,16384,32768,6554,
65297,64769,60855,12743,63195,16384,32768,37942,32768,32768,32768,32768,60075,32768,62087,54613,
41764,2161,21845,1836,17284,5424,10923,1680,11019,555,32768,431,39819,907,32768,171,
65480,32768,64435,33803,2595,32768,57041,32768,61167,32768,32768,32768,32768,32768,32768,1796,
60855,32768,17246,978,32768,32768,8192,32768,32768,32768,14043,2849,32768,2979,6554,6554,
65507,62415,65384,61891,65273,58982,65461,55097,32768,32768,32768,55606,32768,2979,3745,16913,
61885,13827,60893,12196,60855,53248,51493,11243,56656,783,55563,143,63432,7106,52429,445,
65485,1031,65020,1380,65180,57344,65162,36536,61154,6554,26569,2341,63593,3449,65102,533,
47827,2913,57344,3449,35688,1337,32768,22938,25012,910,7944,1008,29319,607,64466,4202,
64549,57301,49152,20025,63351,61167,32768,45542,58982,14564,32768,9362,61895,44840,32768,26385,
59664,17135,60855,13291,40050,12252,32768,7816,25798,1850,60495,2662,18707,122,52538,231,
65332,32768,65210,21693,65113,6554,65141,39667,62259,32768,22258,1337,63636,32768,64255,52429,
60362,32768,6780,819,16384,32768,16384,4681,49152,32768,8985,2521,24410,683,21535,16585,
65416,46091,65292,58328,64626,32768,65016,39897,62687,47332,62805,28948,64284,53620,52870,49567,

© ISO/IEC 2004 – All rights reserved

```
65032,31174,63022,28312,64299,46811,48009,31453,61207,7077,50299,1514,60047,2634,46488,235
};

USInt inter_prob[512] = {
65532,62970,65148,54613,62470,8192,62577,8937,65480,64335,65195,53248,65322,62518,62891,38312,
65075,53405,63980,58982,32768,32768,54613,32768,65238,60009,60075,32768,59294,19661,61203,13107,
63000,9830,62566,58982,11565,32768,25215,3277,53620,50972,63109,43691,54613,32768,39671,17129,
59788,6068,43336,27913,6554,32768,12178,1771,56174,49152,60075,43691,58254,16384,49152,9930,
23130,7282,40960,32768,10923,32768,32768,32768,27307,32768,32768,32768,32768,32768,32768,32768,
36285,12511,10923,32768,45875,16384,32768,32768,16384,23831,4369,32768,8192,10923,32768,32768,
10175,2979,18978,10923,54613,32768,6242,6554,1820,10923,32768,32768,32768,32768,32768,5461,
28459,593,11886,2030,3121,4681,1292,112,42130,23831,49152,29127,32768,6554,5461,2048,
65331,64600,63811,63314,42130,19661,49152,32768,65417,64609,62415,64617,64276,44256,61068,36713,
64887,57525,53620,61375,32768,8192,57344,6554,63608,49809,49152,62623,32768,15851,58982,34162,
55454,51739,64406,64047,32768,32768,7282,32768,49152,58756,62805,64990,32768,14895,16384,19418,
57929,24966,58689,31832,32768,16384,10923,6554,54613,42882,57344,64238,58982,10082,20165,20339,
62687,15061,32768,10923,32768,10923,32768,16384,59578,34427,32768,16384,32768,7825,32768,7282,
58052,23400,32768,5041,32768,2849,32768,32768,47663,15073,57344,4096,32768,1176,32768,1320,
24858,410,24576,923,32768,16384,16384,5461,16384,1365,32768,5461,32768,5699,8192,13107,
46884,2361,23559,424,19661,712,655,182,58637,2094,49152,9362,8192,85,32768,1228,
65486,49152,65186,49152,61320,32768,57088,25206,65352,63047,62623,49152,64641,62165,58986,18304,
64171,16384,60855,54613,42130,32768,61335,32768,58254,58982,49152,32768,60985,35289,64520,31554,
51067,32768,64074,32768,40330,32768,34526,4096,60855,32768,63109,58254,57672,16384,31009,2567,
23406,32768,44620,10923,32768,32768,32099,10923,49152,49152,54613,60075,63422,54613,46388,39719,
58982,32768,54613,32768,14247,32768,22938,5041,32768,49152,32768,32768,25321,6144,29127,10999,
41263,32768,46811,32768,267,4096,426,16384,32768,19275,49152,32768,1008,1437,5767,11275,
5595,5461,37493,6554,4681,32768,6147,1560,38229,10923,32768,40960,35747,2521,5999,312,
17052,2521,18808,3641,213,2427,574,32,51493,42130,42130,53053,11155,312,2069,106,
64406,45197,58982,32768,32768,16384,40960,36864,65336,64244,60075,61681,65269,50748,60340,20515,
58982,23406,57344,32768,6554,16384,19661,61564,60855,47480,32768,54613,46811,21701,54909,37826,
32768,58982,60855,60855,32768,32768,39322,49152,57344,45875,60855,55706,32768,24576,62313,25038,
54613,8192,49152,10923,32768,32768,32768,32768,32768,19661,16384,51493,32768,14043,40050,44651,
59578,5174,32768,6554,32768,5461,23593,5461,63608,51825,32768,23831,58887,24032,57170,3298,
39322,12971,16384,49152,1872,618,13107,2114,58982,25705,32768,60075,28913,949,18312,1815,
48188,114,51493,1542,5461,3855,11360,1163,58982,7215,54613,21487,49152,4590,48430,1421,
28944,1319,6868,324,1456,232,820,7,61681,1864,60855,9922,4369,315,6589,14
};
```

Table B.33 — Probability for arithmetic decoding of spatial scalable binary shape

```
USInt SI_bab_type_prob[1]={ 59808 }

USInt enh_intra_v_prob[128]={
65476, 64428, 62211, 63560, 52253, 58271, 38098, 31981, 50087, 41042,
54620, 31532, 8382, 10754, 3844, 6917, 63834, 50444, 50140, 63043,
58093, 45146, 36768, 13351, 17594, 28777, 39830, 38719, 9768, 21447,
12340,  9786, 60461, 41489, 27433, 53893, 47246, 11415, 13754, 24965,
51620, 28011, 11973, 29709, 13878, 22794, 24385, 1558, 57065, 41918,
25259, 55117, 48064, 12960, 19929, 5937, 25730, 22366, 5204, 32865,
3415, 14814, 6634, 1155, 64444, 62907, 56337, 63144, 38112, 56527,
40247, 37088, 60326, 45675, 51248, 15151, 18868, 43723, 14757, 11721,
```

© ISO/IEC 2004 – All rights reserved

```
62436, 50971, 51738, 59767, 49927, 50675, 38182, 24724, 48447, 47316,
56628, 36336, 12264, 25893, 24243, 5358, 58717, 56646, 48302, 60515,
36497, 26959, 43579, 40280, 54092, 20741, 10891, 7504, 8109, 30840,
6772, 4090, 59810, 61410, 53216, 64127, 32344, 12462, 23132, 19270,
32232, 24774, 9615, 17750, 1714, 6539, 3237, 152
}

USInt enh_intra_h_prob[128]={
65510, 63321, 63851, 62223, 64959, 62202, 63637, 48019, 57072, 33553,
37041, 9527, 53190, 50479, 54232, 12855, 62779, 63980, 49604, 31847,
57591, 64385, 40657, 8402, 33878, 54743, 17873, 8707, 34470, 54322,
16702, 2192, 58325, 48447, 7345, 31317, 45687, 44236, 16685, 24144,
34327, 18724, 10591, 24965, 9247, 7281, 3144, 5921, 59349, 33539,
11447, 5543, 58082, 48995, 35630, 10653, 7123, 15893, 23830, 800,
3491, 15792, 8930, 905, 65209, 63939, 52634, 62194, 64937, 53948,
60081, 46851, 56157, 50930, 35498, 24655, 56331, 59318, 32209, 6872,
59172, 64273, 46724, 41200, 53619, 59022, 37941, 20529, 55026, 52858,
26402, 45073, 57740, 55485, 20533,  6288, 64286, 55438, 16454, 55656,
61175, 45874, 28536, 53762, 58056, 21895, 5482, 39352, 32635, 21633,
2137, 4016, 58490, 14100, 18724, 10461, 53459, 15490, 57992, 15128,
12034, 4340, 6761, 1859, 5794, 6785, 2412, 35
}
```

B.1.6 Sprite Coding

Table B.34 — Code table for the first trajectory point

dmv value	SSS	VLC	dmv_code
-16383 … -8192, 8192 … 16383	14	111111111110	00000000000000...01111111111111, 10000000000000...11111111111111
-8191 … -4096, 4096 … 8191	13	11111111110	0000000000000...0111111111111, 1000000000000...1111111111111
-4095 … -2048, 2048 … 4095	12	1111111110	000000000000...011111111111, 100000000000...111111111111
-2047...-1024, 1024...2047	11	111111110	00000000000...01111111111, 10000000000...11111111111
-1023...-512, 512...1023	10	11111110	0000000000...0111111111, 1000000000...1111111111
-511...-256, 256...511	9	1111110	000000000...011111111, 100000000...111111111
-255...-128, 128...255	8	111110	00000000...01111111, 10000000...11111111
-127...-64, 64...127	7	11110	0000000...0111111, 1000000...1111111
-63...-32, 32...63	6	1110	000000...011111, 100000...111111
-31...-16, 16...31	5	110	00000...01111, 10000...1111
-15...-8, 8...15	4	101	0000...0111, 1000...1111
-7...-4, 4...7	3	100	000...011, 100...111
-3...-2, 2...3	2	011	00...01, 10...11
-1, 1	1	010	0, 1
0	0	00	-

© ISO/IEC 2004 – All rights reserved

Table B.35 — Code table for scaled brightness change factor

brightness_change_ factor value	brightness_ change_factor_ length value	brightness_ change_factor_ length VLC	brightness_change_factor
-16...-1, 1...16	1	0	00000...01111, 10000...11111
-48...-17, 17...48	2	10	000000...011111, 100000...111111
112...-49, 49...112	3	110	0000000...0111111, 1000000...1111111
113…624	4	1110	000000000...111111111
625...1648	4	1111	0000000000…1111111111

B.1.7 DCT based facial object decoding

Table B.36 — Viseme_select_table, 29 symbols

symbol	bits	code	symbol	bits	code	symbol	bits	code
0	6	001000	10	6	010001	20	6	010000
1	6	001001	11	6	011001	21	6	010010
2	6	001011	12	5	00001	22	6	011010
3	6	001101	13	6	011101	23	5	00010
4	6	001111	14	1	1	24	6	011110
5	6	010111	15	6	010101	25	6	010110
6	6	011111	16	6	010100	26	6	001110
7	5	00011	17	6	011100	27	6	001100
8	6	011011	18	5	00000	28	6	001010
9	6	010011	19	6	011000			

Table B.37 — Expression_select_table, 13 symbols

symbol	bits	code	symbol	bits	code	symbol	bits	code
0	5	01000	5	4	0011	10	5	01110
1	5	01001	6	1	1	11	5	01100
2	5	01011	7	4	0001	12	5	01010
3	5	01101	8	4	0000			
4	5	01111	9	4	0010			

Table B.38 — Viseme and Expression intensity_table, 127 symbols

symbol	bits	code	symbol	bits	code	symbol	bits	code
0	17	10010001101010010	43	16	1001000110100111	86	16	1001000110100110
1	17	10010001101010011	44	8	10011100	87	16	1001000110100100
2	17	10010001101010101	45	11	10010001111	88	16	1001000110100010
3	17	10010001101010111	46	9	100100010	89	16	1001000110100000
4	17	10010001101011001	47	10	1110001011	90	16	1001000110011110
5	17	10010001101011011	48	9	100011011	91	16	1001000110011100
6	17	10010001101011101	49	10	1110001001	92	16	1001000110011010
7	17	10010001101011111	50	9	100011010	93	16	1001000110011000

© ISO/IEC 2004 – All rights reserved

8	17	10010001101100001	51	9	100111010	94	16	1001000110010110
9	17	10010001101100011	52	10	1110001000	95	16	1001000110010100
10	17	10010001101100101	53	7	1000111	96	16	1001000110010010
11	17	10010001101100111	54	7	1000010	97	16	1001000110010000
12	17	10010001101101001	55	8	10010000	98	16	1001000110001110
13	17	10010001101101011	56	7	1001111	99	16	1001000110001100
14	17	10010001101101101	57	7	1110000	100	16	1001000110001010
15	17	10010001101101111	58	6	100000	101	16	1001000110001000
16	17	10010001101110001	59	6	100101	102	16	1001000110000110
17	17	10010001101110011	60	6	111010	103	16	1001000110000100
18	17	10010001101110111	61	5	11111	104	16	1001000110000010
19	17	10010001101111001	62	3	101	105	16	1001000110000000
20	17	10010001101111011	63	1	0	106	17	10010001101111110
21	17	10010001101111101	64	3	110	107	17	10010001101111100
22	17	10010001101111111	65	5	11110	108	17	10010001101111010
23	16	1001000110000001	66	6	111001	109	17	10010001101111000
24	16	1001000110000011	67	6	111011	110	17	10010001101110110
25	16	1001000110000101	68	6	100010	111	17	10010001101110010
26	16	1001000110000111	69	7	1001100	112	17	10010001101110000
27	16	1001000110001001	70	7	1001001	113	17	10010001101101110
28	16	1001000110001011	71	7	1001101	114	17	10010001101101100
29	16	1001000110001101	72	8	10001100	115	17	10010001101101010
30	16	1001000110001111	73	8	10000111	116	17	10010001101101000
31	16	1001000110010001	74	8	10000110	117	17	10010001101100110
32	16	1001000110010011	75	17	10010001101110100	118	17	10010001101100100
33	16	1001000110010101	76	9	111000110	119	17	10010001101100010
34	16	1001000110010111	77	11	11100010100	120	17	10010001101100000
35	16	1001000110011001	78	11	10011101111	121	17	10010001101011110
36	16	1001000110011011	79	17	10010001101110101	122	17	10010001101011100
37	16	1001000110011101	80	10	1001110110	123	17	10010001101011010
38	16	1001000110011111	81	16	1001000110101000	124	17	10010001101011000
39	16	1001000110100001	82	11	10010001110	125	17	10010001101010110
40	16	1001000110100011	83	10	1110001111	126	17	10010001101010100
41	11	11100010101	84	11	10011101110			
42	16	1001000110100101	85	10	1110001110			

Table B.39 — Runlength_table, 16 symbols

symbol	bits	code	symbol	bits	code	symbol	bits	code
0	1	1	6	9	000001011	12	8	00000000
1	2	01	7	9	000001101	13	8	00000010
2	3	001	8	9	000001111	14	9	000001110
3	4	0001	9	8	00000011	15	9	000001100
4	5	00001	10	8	00000001			
5	9	000001010	11	8	00000100			

© ISO/IEC 2004 – All rights reserved

Table B.40 — DC_table, 512 symbols

symbol	bits	code	symbol	bits	code	symbol	bits	code
0	17	11010111001101010	171	17	11010111001111001	342	17	11010111001111000
1	17	11010111001101011	172	17	11010111010000001	343	17	11010111001110000
2	17	11010111001101101	173	17	11010111010001001	344	17	11010111001110010
3	17	11010111001101111	174	17	11010111010010001	345	17	11010111001111010
4	17	11010111001110101	175	17	11010111010011001	346	17	11010111010000010
5	17	11010111001110111	176	17	11010111010101001	347	17	11010111010001010
6	17	11010111001111101	177	17	11010111010110001	348	17	11010111010010010
7	17	11010111001111111	178	17	11010111010111001	349	17	11010111010011010
8	17	11010111010000101	179	17	11010111011000001	350	17	11010111010101010
9	17	11010111010000111	180	17	11010111011001001	351	17	11010111010110010
10	17	11010111010001101	181	17	11010111011011001	352	17	11010111010111010
11	17	11010111010001111	182	17	11010111011111001	353	17	11010111011000010
12	17	11010111010010101	183	17	11010111100000001	354	17	11010111011001010
13	17	11010111010010111	184	17	11010111100001001	355	17	11010111011011010
14	17	11010111010011101	185	17	11010111100011001	356	17	11010111011111010
15	17	11010111010011111	186	17	11010111100100001	357	17	11010111100000010
16	17	11010111010101101	187	17	11010111100101001	358	17	11010111100001010
17	17	11010111010101111	188	17	11010111100111001	359	17	11010111100011010
18	17	11010111010110111	189	17	11010111101000001	360	17	11010111100100010
19	17	11010111010111101	190	17	11010111101001001	361	17	11010111100101010
20	17	11010111010111111	191	17	11010111101011001	362	17	11010111100111010
21	17	11010111011000111	192	17	11010111101111001	363	17	11010111101000010
22	17	11010111011001101	193	17	11010111110000001	364	17	11010111101001010
23	17	11010111011001111	194	17	11010111110001001	365	17	11010111101011010
24	17	11010111011011101	195	17	11010111110011001	366	17	11010111101111010
25	17	11010111011011111	196	17	11010111110111001	367	17	11010111110000010
26	17	11010111011111101	197	17	11010111111100001	368	17	11010111110001010
27	17	11010111011111111	198	17	11010111111101001	369	17	11010111110011010
28	17	11010111100000111	199	17	11010111111111001	370	17	11010111110111010
29	17	11010111100001101	200	16	1101011100000001	371	17	11010111111100010
30	17	11010111100001111	201	16	1101011100001001	372	17	11010111111101010
31	17	11010111100011101	202	16	1101011100011001	373	17	11010111111111010
32	17	11010111100011111	203	17	11010111111001001	374	16	1101011100000010
33	17	11010111100100101	204	17	11010111111010001	375	16	1101011100001010
34	17	11010111100100111	205	17	11010111111011001	376	16	1101011100011010
35	17	11010111100101101	206	16	1101011100101001	377	17	11010111111001010
36	17	11010111100101111	207	17	11010111110100001	378	17	11010111111010010
37	17	11010111100111101	208	17	11010111110101001	379	17	11010111111011010
38	17	11010111100111111	209	17	11010111101101001	380	16	1101011100101010
39	17	11010111101000101	210	17	11010111011100001	381	17	11010111110100010
40	17	11010111101000111	211	16	1101011100100000	382	17	11010111110101010
41	17	11010111101001101	212	16	1101011100100001	383	17	11010111101101010
42	17	11010111101001111	213	17	11010111111000001	384	17	11010111011100010
43	17	11010111101011101	214	16	1101011100010001	385	17	11010111011101010
44	17	11010111101011111	215	17	11010111111110001	386	17	11010111011101000

© ISO/IEC 2004 – All rights reserved

45	17	11010111101111101	216	17	11010111110110001	387	16	1101011100100010
46	17	11010111101111111	217	17	11010111110010001	388	17	11010111111000010
47	17	11010111110000101	218	11	11101100101	389	16	1101011100010010
48	17	11010111110000111	219	11	11011111011	390	17	11010111111110010
49	17	11010111110001101	220	11	11011110001	391	17	11010111110110010
50	17	11010111110001111	221	10	1101110011	392	17	11010111110010010
51	17	11010111110011101	222	17	11010111101110001	393	17	11010111101110010
52	17	11010111110011111	223	17	11010111010100000	394	17	11010111101010010
53	17	11010111110111101	224	17	11010111010100001	395	17	11010111101010000
54	17	11010111110111111	225	17	11010111011110100	396	17	11010111100010010
55	17	11010111111100101	226	17	11010111011110101	397	17	11010111100010000
56	17	11010111111100111	227	17	11010111011110001	398	17	11010111011010010
57	17	11010111111101101	228	17	11010111100010101	399	17	11010111011010000
58	17	11010111111101111	229	17	11010111100110000	400	16	1101011100110010
59	17	11010111111111101	230	17	11010111100110001	401	16	1101011100110000
60	17	11010111111111111	231	17	11010111101010101	402	17	11010111010100110
61	16	1101011100000101	232	11	11101100111	403	17	11010111010100100
62	16	1101011100000111	233	17	11010111101110101	404	17	11010111010100010
63	16	1101011100001101	234	11	11101100110	405	17	11010111011010110
64	16	1101011100001111	235	17	11010111110110101	406	17	11010111011010100
65	16	1101011100011101	236	17	11010111111000100	407	17	11010111011110110
66	16	1101011100011111	237	8	11010110	408	17	11010111011110010
67	17	11010111111001101	238	11	11011110010	409	17	11010111100010110
68	17	11010111111001111	239	9	110010100	410	17	11010111100110110
69	17	11010111111010101	240	10	1101110001	411	17	11010111100110100
70	17	11010111111010111	241	9	110001111	412	17	11010111100110010
71	17	11010111111011101	242	10	1101111100	413	17	11010111101010110
72	17	11010111111011111	243	9	110010101	414	17	11010111101110110
73	16	1101011100101101	244	9	110111111	415	17	11010111110010110
74	16	1101011100101111	245	10	1101110100	416	17	11010111110010100
75	17	11010111110100101	246	7	1100100	417	17	11010111110110110
76	17	11010111110100111	247	8	11101101	418	17	11010111111110110
77	17	11010111110101101	248	8	11001011	419	17	11010111111110100
78	17	11010111110101111	249	7	1101100	420	16	1101011100010110
79	17	11010111101101101	250	7	1101101	421	16	1101011100010100
80	17	11010111101101111	251	7	1110111	422	17	11010111111000110
81	17	11010111011100101	252	6	110100	423	16	1101011100100110
82	17	11010111011100111	253	6	111001	424	16	1101011100100100
83	17	11010111011101101	254	5	11111	425	17	11010111101100110
84	17	11010111011101111	255	3	100	426	17	11010111101100100
85	17	11010111101100001	256	1	0	427	17	11010111101100010
86	17	11010111101100011	257	3	101	428	17	11010111101100000
87	17	11010111101100101	258	5	11110	429	17	11010111011101110
88	17	11010111101100111	259	6	111000	430	17	11010111011101100
89	16	1101011100100101	260	6	111010	431	17	11010111011100110
90	16	1101011100100111	261	6	110000	432	17	11010111011100100
91	17	11010111111000111	262	7	1100111	433	17	11010111101101110
92	16	1101011100010101	263	7	1100110	434	17	11010111101101100

© ISO/IEC 2004 – All rights reserved

93	16	1101011100010111	264	7	1101010	435	17	11010111110101110
94	17	11010111111110101	265	8	11000101	436	17	11010111110101100
95	17	11010111111110111	266	8	11000110	437	17	11010111110100110
96	17	11010111110110111	267	8	11000100	438	17	11010111110100100
97	17	11010111110010101	268	17	11010111111000101	439	16	1101011100101110
98	17	11010111110010111	269	9	111011000	440	16	1101011100101100
99	17	11010111101110111	270	11	11011111010	441	17	11010111111011110
100	17	11010111101010111	271	11	11011110101	442	17	11010111111011100
101	17	11010111100110011	272	17	11010111100000101	443	17	11010111111010110
102	17	11010111100110101	273	10	1101111011	444	17	11010111111010100
103	17	11010111100110111	274	17	11010111011000101	445	17	11010111111001110
104	17	11010111100010111	275	11	11011110011	446	17	11010111111001100
105	17	11010111011110011	276	9	110001110	447	16	1101011100011110
106	17	11010111011110111	277	11	11011110000	448	16	1101011100011100
107	17	11010111011010101	278	10	1101110111	449	16	1101011100001110
108	17	11010111011010111	279	17	11010111010110101	450	16	1101011100001100
109	17	11010111010100011	280	16	1101011100110100	451	16	1101011100000110
110	17	11010111010100101	281	10	1101110010	452	16	1101011100000100
111	17	11010111010100111	282	10	1101110000	453	17	11010111111111110
112	16	1101011100110001	283	11	11011101010	454	17	11010111111111100
113	16	1101011100110011	284	17	11010111010110100	455	17	11010111111101110
114	17	11010111011010001	285	17	11010111011000100	456	17	11010111111101100
115	17	11010111011010011	286	17	11010111100000100	457	17	11010111111100110
116	17	11010111100010001	287	11	11011101100	458	17	11010111111100100
117	17	11010111100010011	288	17	11010111110110100	459	17	11010111110111110
118	17	11010111101010001	289	17	11010111101110100	460	17	11010111110111100
119	17	11010111101010011	290	17	11010111101010100	461	17	11010111110011110
120	17	11010111101110011	291	11	11101100100	462	17	11010111110011100
121	17	11010111110010011	292	17	11010111100010100	463	17	11010111110001110
122	17	11010111110110011	293	17	11010111011110000	464	17	11010111110001100
123	17	11010111111110011	294	11	11011110100	465	17	11010111110000110
124	16	1101011100010011	295	11	11011101011	466	17	11010111110000100
125	17	11010111111000011	296	17	11010111101110000	467	17	11010111101111110
126	16	1101011100100011	297	17	11010111110010000	468	17	11010111101111100
127	17	11010111011101001	298	17	11010111110110000	469	17	11010111101011110
128	17	11010111011101011	299	17	11010111111110000	470	17	11010111101011100
129	17	11010111011100011	300	16	1101011100010000	471	17	11010111101001110
130	17	11010111101101011	301	17	11010111111000000	472	17	11010111101001100
131	17	11010111110101011	302	11	11011101101	473	17	11010111101000110
132	17	11010111110100011	303	17	11010111011100000	474	17	11010111101000100
133	16	1101011100101011	304	17	11010111101101000	475	17	11010111100111110
134	17	11010111111011011	305	17	11010111110101000	476	17	11010111100111100
135	17	11010111111010011	306	17	11010111110100000	477	17	11010111100101110
136	17	11010111111001011	307	16	1101011100101000	478	17	11010111100101100
137	16	1101011100011011	308	17	11010111111011000	479	17	11010111100100110
138	16	1101011100001011	309	17	11010111111010000	480	17	11010111100100100
139	16	1101011100000011	310	17	11010111111001000	481	17	11010111100011110
140	17	11010111111111011	311	16	1101011100011000	482	17	11010111100011100

© ISO/IEC 2004 – All rights reserved

141	17	11010111111101011	312	16	1101011100001000	483	17	11010111100001110
142	17	11010111111100011	313	16	1101011100000000	484	17	11010111100001100
143	17	11010111110111011	314	17	11010111111111000	485	17	11010111100000110
144	17	11010111110011011	315	17	11010111111101000	486	17	11010111011111110
145	17	11010111110001011	316	17	11010111111100000	487	17	11010111011111100
146	17	11010111110000011	317	17	11010111110111000	488	17	11010111011011110
147	17	11010111101111011	318	17	11010111110011000	489	17	11010111011011100
148	17	11010111101011011	319	17	11010111110001000	490	17	11010111011001110
149	17	11010111101001011	320	17	11010111110000000	491	17	11010111011001100
150	17	11010111101000011	321	17	11010111101111000	492	17	11010111011000110
151	17	11010111100111011	322	17	11010111101011000	493	17	11010111010111110
152	17	11010111100101011	323	17	11010111101001000	494	17	11010111010111100
153	17	11010111100100011	324	17	11010111101000000	495	17	11010111010110110
154	17	11010111100011011	325	17	11010111100111000	496	17	11010111010101110
155	17	11010111100001011	326	17	11010111100101000	497	17	11010111010101100
156	17	11010111100000011	327	17	11010111100100000	498	17	11010111010011110
157	17	11010111011111011	328	17	11010111100011000	499	17	11010111010011100
158	17	11010111011011011	329	17	11010111100001000	500	17	11010111010010110
159	17	11010111011001011	330	17	11010111100000000	501	17	11010111010010100
160	17	11010111011000011	331	17	11010111011111000	502	17	11010111010001110
161	17	11010111010111011	332	17	11010111011011000	503	17	11010111010001100
162	17	11010111010110011	333	17	11010111011001000	504	17	11010111010000110
163	17	11010111010101011	334	17	11010111011000000	505	17	11010111010000100
164	17	11010111010011011	335	17	11010111010111000	506	17	11010111001111110
165	17	11010111010010011	336	17	11010111010110000	507	17	11010111001111100
166	17	11010111010001011	337	17	11010111010101000	508	17	11010111001110110
167	17	11010111010000011	338	17	11010111010011000	509	17	11010111001110100
168	17	11010111001111011	339	17	11010111010010000	510	17	11010111001101110
169	17	11010111001110011	340	17	11010111010001000	511	17	11010111001101100
170	17	11010111001110001	341	17	11010111010000000			

Table B.41 — AC_table, 512 symbols

symbol	no_ of_ bits	code	symbol	no_ of_ bits	code	symbol	no_ of_ bits	code
0	16	1000011100011000	171	16	1000011101100001	342	16	1000011101100000
1	16	1000011100011001	172	16	1000011110100001	343	15	100001110000000
2	16	1000011100011011	173	16	1000011111000001	344	16	1000011101101000
3	16	1000011100011101	174	16	1000011111100001	345	16	1000011110101000
4	16	1000011100011111	175	15	100001000100001	346	16	1000011111001000
5	16	1000011100100101	176	15	100001001100001	347	16	1000011111101000
6	16	1000011100100111	177	15	100001011000001	348	15	100001000101000
7	16	1000011100101101	178	15	100001011100001	349	15	100001001101000
8	16	1000011100101111	179	15	100001010100001	350	15	100001011001000
9	16	1000011100111101	180	15	100001010000001	351	15	100001011101000
10	16	1000011100111111	181	15	100001001000001	352	15	100001010101000
11	16	1000011101111101	182	15	100001000000001	353	15	100001010001000

© ISO/IEC 2004 – All rights reserved

12	16	1000011101111111	183	16	1000011110000001	354	15	100001001001000
13	16	1000011110111111	184	16	1000011101000001	355	15	100001000001000
14	16	1000011111011101	185	16	1000011101010001	356	16	1000011110001000
15	16	1000011111011111	186	16	1000011110010001	357	16	1000011101001000
16	16	1000011111111101	187	15	100001000010001	358	16	1000011101011000
17	16	1000011111111111	188	15	100001001010001	359	16	1000011110011000
18	15	100001000111101	189	15	100001010010001	360	15	100001000011000
19	15	100001000111111	190	15	100001010110001	361	15	100001001011000
20	15	100001001111101	191	15	100001011110001	362	15	100001010011000
21	15	100001001111111	192	15	100001011010001	363	15	100001010111000
22	15	100001011011101	193	15	100001001110001	364	15	100001011111000
23	15	100001011011111	194	15	100001000110001	365	15	100001011011000
24	15	100001011111101	195	16	1000011111110001	366	15	100001001111000
25	15	100001011111111	196	16	1000011111010001	367	15	100001000111000
26	15	100001010111111	197	16	1000011110110001	368	16	1000011111111000
27	15	100001010011101	198	16	1000011101110001	369	16	1000011111011000
28	15	100001010011111	199	16	1000011100110001	370	16	1000011110111000
29	15	100001001011111	200	15	100001110001001	371	16	1000011101111000
30	15	100001000011111	201	16	1000011100110101	372	16	1000011100111000
31	16	1000011110011111	202	16	1000011101110101	373	16	1000011100101000
32	16	1000011101011111	203	16	1000011110110101	374	16	1000011100100000
33	16	1000011101001111	204	16	1000011111010101	375	16	1000011100100010
34	16	1000011110001111	205	16	1000011111110101	376	16	1000011100101010
35	15	100001000001111	206	15	100001000110101	377	16	1000011100111010
36	15	100001001001111	207	15	100001001110101	378	16	1000011101111010
37	15	100001010001111	208	15	100001011010101	379	16	1000011110111010
38	15	100001010101111	209	15	100001011110101	380	16	1000011111011010
39	15	100001011101111	210	15	100001010110101	381	16	1000011111111010
40	15	100001011001111	211	15	100001010010101	382	15	100001000111010
41	15	100001001101111	212	15	100001001010101	383	15	100001001111010
42	15	100001000101111	213	15	100001000010101	384	15	100001011011010
43	16	1000011111101111	214	16	1000011110010101	385	15	100001011111010
44	16	1000011111001111	215	16	1000011101010101	386	15	100001010111010
45	16	1000011110101111	216	16	1000011101000101	387	15	100001010011010
46	16	1000011101101111	217	16	1000011110000101	388	15	100001001011010
47	15	100001110000111	218	15	100001000000101	389	15	100001000011010
48	16	1000011101100111	219	15	100001001000101	390	16	1000011110011010
49	16	1000011110100111	220	15	100001010000101	391	16	1000011101011010
50	16	1000011111000111	221	15	100001010100101	392	16	1000011101001010
51	16	1000011111100111	222	15	100001011100101	393	16	1000011110001010
52	15	100001000100111	223	15	100001011000101	394	15	100001000001010
53	15	100001001100111	224	15	100001001100101	395	15	100001001001010
54	15	100001011000111	225	15	100001000100101	396	15	100001010001010
55	15	100001011100111	226	16	1000011111100101	397	15	100001010101010
56	15	100001010100111	227	16	1000011111000101	398	15	100001011101010
57	15	100001010000111	228	16	1000011110100101	399	15	100001011001010
58	15	100001001000111	229	16	1000011101100101	400	15	100001001101010
59	15	100001000000111	230	15	100001110000101	401	15	100001000101010

© ISO/IEC 2004 – All rights reserved

60	16	1000011110000111	231	16	1000011101101101	402	16	1000011111101010
61	16	1000011101000111	232	16	1000011110101101	403	16	1000011111001010
62	16	1000011101010111	233	16	1000011111001101	404	16	1000011110101010
63	16	1000011110010111	234	16	1000011111101101	405	16	1000011101101010
64	15	100001000010111	235	15	100001000101101	406	15	100001110000010
65	15	100001001010111	236	15	100001001101101	407	16	1000011101100010
66	15	100001010010111	237	15	100001011001101	408	16	1000011110100010
67	15	100001010110111	238	15	100001011101101	409	16	1000011111000010
68	15	100001011110111	239	15	100001010101101	410	16	1000011111100010
69	15	100001011010111	240	15	100001010001101	411	15	100001000100010
70	15	100001001110111	241	15	100001001001101	412	15	100001001100010
71	15	100001000110111	242	15	100001000001101	413	15	100001011000010
72	16	1000011111110111	243	16	1000011110001101	414	15	100001011100010
73	16	1000011111010111	244	16	1000011101001101	415	15	100001010100010
74	16	1000011110110111	245	16	1000011101011101	416	15	100001010000010
75	16	1000011101110111	246	16	1000011110011101	417	15	100001001000010
76	16	1000011100110111	247	15	100001000011101	418	15	100001000000010
77	15	100001110001011	248	6	100000	419	16	1000011110000010
78	16	1000011100110011	249	15	100001001011101	420	16	1000011101000010
79	16	1000011101110011	250	15	100001010111101	421	16	1000011101010010
80	16	1000011110110011	251	7	1001110	422	16	1000011110010010
81	16	1000011111010011	252	6	100110	423	15	100001000010010
82	16	1000011111110011	253	5	10010	424	15	100001001010010
83	15	100001000110011	254	4	1010	425	15	100001010010010
84	15	100001001110011	255	2	11	426	15	100001010110010
85	15	100001011010011	256	16	1000011110111100	427	15	100001011110010
86	15	100001011110011	257	1	0	428	15	100001011010010
87	15	100001010110011	258	4	1011	429	15	100001001110010
88	15	100001010010011	259	6	100011	430	15	100001000110010
89	15	100001001010011	260	6	100010	431	16	1000011111110010
90	15	100001000010011	261	7	1001111	432	16	1000011111010010
91	16	1000011110010011	262	16	1000011110111101	433	16	1000011110110010
92	16	1000011101010011	263	8	10000110	434	16	1000011101110010
93	16	1000011101000011	264	15	100001010111100	435	16	1000011100110010
94	16	1000011110000011	265	15	100001001011100	436	15	100001110001010
95	15	100001000000011	266	15	100001000011100	437	16	1000011100110110
96	15	100001001000011	267	16	1000011110011100	438	16	1000011101110110
97	15	100001010000011	268	16	1000011101011100	439	16	1000011110110110
98	15	100001010100011	269	16	1000011101001100	440	16	1000011111010110
99	15	100001011100011	270	16	1000011110001100	441	16	1000011111110110
100	15	100001011000011	271	15	100001000001100	442	15	100001000110110
101	15	100001001100011	272	15	100001001001100	443	15	100001001110110
102	15	100001000100011	273	15	100001010001100	444	15	100001011010110
103	16	1000011111100011	274	15	100001010101100	445	15	100001011110110
104	16	1000011111000011	275	15	100001011101100	446	15	100001010110110
105	16	1000011110100011	276	15	100001011001100	447	15	100001010010110
106	16	1000011101100011	277	15	100001001101100	448	15	100001001010110
107	15	100001110000011	278	15	100001000101100	449	15	100001000010110

© ISO/IEC 2004 – All rights reserved

108	16	1000011101101011	279	16	1000011111101100	450	16	1000011110010110
109	16	1000011110101011	280	16	1000011111001100	451	16	1000011101010110
110	16	1000011111001011	281	16	1000011110101100	452	16	1000011101000110
111	16	1000011111101011	282	16	1000011101101100	453	16	1000011110000110
112	15	100001000101011	283	15	100001110000100	454	15	100001000000110
113	15	100001001101011	284	16	1000011101100100	455	15	100001001000110
114	15	100001011001011	285	16	1000011110100100	456	15	100001010000110
115	15	100001011101011	286	16	1000011111000100	457	15	100001010100110
116	15	100001010101011	287	16	1000011111100100	458	15	100001011100110
117	15	100001010001011	288	15	100001000100100	459	15	100001011000110
118	15	100001001001011	289	15	100001001100100	460	15	100001001100110
119	15	100001000001011	290	15	100001011000100	461	15	100001000100110
120	16	1000011110001011	291	15	100001011100100	462	16	1000011111100110
121	16	1000011101001011	292	15	100001010100100	463	16	1000011111000110
122	16	1000011101011011	293	15	100001010000100	464	16	1000011110100110
123	16	1000011110011011	294	15	100001001000100	465	16	1000011101100110
124	15	100001000011011	295	15	100001000000100	466	15	100001110000110
125	15	100001001011011	296	16	1000011110000100	467	16	1000011101101110
126	15	100001010011011	297	16	1000011101000100	468	16	1000011110101110
127	15	100001010111011	298	16	1000011101010100	469	16	1000011111001110
128	15	100001011111011	299	16	1000011110010100	470	16	1000011111101110
129	15	100001011011011	300	15	100001000010100	471	15	100001000101110
130	15	100001001111011	301	15	100001001010100	472	15	100001001101110
131	15	100001000111011	302	15	100001010010100	473	15	100001011001110
132	16	1000011111111011	303	15	100001010110100	474	15	100001011101110
133	16	1000011111011011	304	15	100001011110100	475	15	100001010101110
134	16	1000011110111011	305	15	100001011010100	476	15	100001010001110
135	16	1000011101111011	306	15	100001001110100	477	15	100001001001110
136	16	1000011100111011	307	15	100001000110100	478	15	100001000001110
137	16	1000011100101011	308	16	1000011111110100	479	16	1000011110001110
138	16	1000011100100011	309	16	1000011111010100	480	16	1000011101001110
139	16	1000011100100001	310	16	1000011110110100	481	16	1000011101011110
140	16	1000011100101001	311	16	1000011101110100	482	16	1000011110011110
141	16	1000011100111001	312	16	1000011100110100	483	15	100001000011110
142	16	1000011101111001	313	15	100001110001000	484	15	100001001011110
143	16	1000011110111001	314	16	1000011100110000	485	15	100001010011110
144	16	1000011111011001	315	16	1000011101110000	486	15	100001010011100
145	16	1000011111111001	316	16	1000011110110000	487	15	100001010111110
146	15	100001000111001	317	16	1000011111010000	488	15	100001011111110
147	15	100001001111001	318	16	1000011111110000	489	15	100001011111100
148	15	100001011011001	319	15	100001000110000	490	15	100001011011110
149	15	100001011111001	320	15	100001001110000	491	15	100001011011100
150	15	100001010111001	321	15	100001011010000	492	15	100001001111110
151	15	100001010011001	322	15	100001011110000	493	15	100001001111100
152	15	100001001011001	323	15	100001010110000	494	15	100001000111110
153	15	100001000011001	324	15	100001010010000	495	15	100001000111100
154	16	1000011110011001	325	15	100001001010000	496	16	1000011111111110
155	16	1000011101011001	326	15	100001000010000	497	16	1000011111111100

© ISO/IEC 2004 – All rights reserved

156	16	1000011101001001	327	16	1000011110010000	498	16	1000011111011110
157	16	1000011110001001	328	16	1000011101010000	499	16	1000011111011100
158	15	100001000001001	329	16	1000011101000000	500	16	1000011110111110
159	15	100001001001001	330	16	1000011110000000	501	16	1000011101111110
160	15	100001010001001	331	15	100001000000000	502	16	1000011101111100
161	15	100001010101001	332	15	100001001000000	503	16	1000011100111110
162	15	100001011101001	333	15	100001010000000	504	16	1000011100111100
163	15	100001011001001	334	15	100001010100000	505	16	1000011100101110
164	15	100001001101001	335	15	100001011100000	506	16	1000011100101100
165	15	100001000101001	336	15	100001011000000	507	16	1000011100100110
166	16	1000011111101001	337	15	100001001100000	508	16	1000011100100100
167	16	1000011111001001	338	15	100001000100000	509	16	1000011100011110
168	16	1000011110101001	339	16	1000011111100000	510	16	1000011100011100
169	16	1000011101101001	340	16	1000011111000000	511	16	1000011100011010
170	15	100001110000001	341	16	1000011110100000			

B.1.8 Shape decoding for still texture object

USInt sto_SI_bab_type_prob_even[1]={44651}

USInt sto_enh_odd_prob0[256]={

65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 25682, 29649, 83, 1096, 28876, 19253, 2320, 292, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 20597, 40071, 1, 1476, 6971, 9466, 4033, 314, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 34822, 58944, 1049, 888, 47296, 38677, 478, 14, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 21384, 35003, 26, 7, 9600, 13044, 44, 1, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 66, 13653, 1, 1, 25105, 19008, 285, 31, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1057, 27888, 1, 2521, 4252, 15367, 1, 751, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 290, 17476, 1, 1, 8625, 20056, 121, 3, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1960, 15984, 1, 1, 4747, 16480, 110, 9, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 87, 140, 1, 55, 1205, 1864, 1, 1, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 549, 2823, 1, 1, 1902, 782, 1285, 130, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 290, 6665, 1, 1, 15984, 8656, 1, 244, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 63, 2114, 1, 3, 2138, 1560, 69, 1, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1, 3178, 1, 218, 160, 66, 1, 2, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 35, 7047, 366, 180, 34, 113, 1, 9, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 837, 15124, 1, 1, 5571, 3236, 1, 2, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 14, 1538, 1, 14, 31, 115, 1, 1};

USInt sto_enh_even_prob0[256]={

65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536,25542, 38525, 48333, 53517, 26541, 23214, 61559, 59853,65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 7151, 31006, 33480, 29879, 2609, 4142, 16384, 11884, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 44817, 65535, 56097, 64984, 40735, 47710, 43386, 52046, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 17545, 29626, 37308, 42263, 4196, 13552, 7199, 7230, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1, 1, 1, 39322, 13578, 17416, 29218, 31831, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1311, 65535, 32768, 7282, 1, 3048, 25206, 19935, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1, 32768, 17873, 61861, 3417, 14895, 4541, 5293, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 4819, 32768, 39950, 43523, 1148, 4021, 12072, 5436, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1, 1, 134, 1, 1, 55, 5461, 2849, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 24, 5243, 590, 1079, 86, 95, 14564, 7159, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 328, 1, 2564, 14919, 21845, 1, 9362, 15880, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 43, 362, 150, 1179, 752, 529, 683, 331, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1, 1, 700, 862, 25, 24, 1317, 558, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 1, 1, 1, 172, 2, 4, 793, 342, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536,572, 1,

© ISO/IEC 2004 – All rights reserved

1928, 43080, 3337, 1680, 1401, 2131, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536,18, 1, 366, 7456, 8, 18, 40, 8};

USInt sto_enh_odd_prob1[256]={

65536, 65536, 54236, 62309, 65536, 65536, 39146, 32459, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 3293, 20806, 65536, 65536, 20132, 22080, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 62837, 40750, 65536, 65536, 36528, 20960, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 46875, 21025, 65536, 65536, 35747, 30778, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 63956, 52735, 65536, 65536, 32974, 6233, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 39449, 40885, 65536, 65536, 23406, 10898, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 62156, 12655, 65536, 65536, 23973, 3451, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 31892, 3756, 65536, 65536, 24045, 14281, 0, 0, 0, 0, 0, 0, 0, 0, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 51966, 33057, 1, 6782, 62238, 32046, 5919, 418, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 64435, 18941, 1224, 7203, 52134, 4674, 6753, 1113, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 62703, 34133, 16705, 28007, 31477, 15453, 21558, 6073, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 61951, 29954, 4826, 6481, 12288, 2410, 4466, 228, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 1789, 23069, 726, 7470, 30386, 26721, 9811, 1446, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 45333, 28672, 21363, 26870, 41125, 9455, 25752, 12372, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 46324, 37071, 4994, 5136, 18879, 28687, 9330, 366, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 31745, 26116, 346, 397, 2900, 13830, 361, 8};

USInt sto_enh_even_prob1[256]={

65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536,0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 0, 0, 0, 0, 0, 0, 0, 0, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 0, 0, 0, 0, 0, 0, 0, 0, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 54018, 8704, 1, 903, 61648, 31196, 327, 575, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 64400, 36956, 1673, 9758, 52289, 4361, 659, 1433, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 56482, 65535, 1, 1, 11905, 3034, 1, 1, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 61103, 56650, 925, 8814, 11845, 2075, 828, 223, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 4161, 1, 494, 5041, 52508, 32195, 11005, 2463, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 63286, 32768, 39133, 49486, 53351, 8541, 37603, 15011, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 18805, 13107, 1039, 1214, 5060, 21845, 3830, 387, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 65537, 31654, 32951, 490, 1496, 2535, 11699, 328, 13};

B.2 Arithmetic Decoding

B.2.1 Aritmetic decoding for still texture object

To fully initialize the decoder, the function ac_decoder_init is called followed by ac_model_init respectively:

```
void ac_decoder_init (ac_decoder *acd) {
    int i, t;
    acd->bits_to_go = 0;
    acd->total_bits = 0;
    acd->value = 0;
    for (i=1; i<=Code_value_bits; i++)  {
        acd->value = 2*acd->value + input_bit(acd);
    }
    acd->low = 0;
    acd->high = Top_value;
    return;
}
```

© ISO/IEC 2004 – All rights reserved

```
void ac_model_init (ac_model *acm, int nsym) {
    int i;

    acm->nsym = nsym;

    acm->freq = (unsigned short *) malloc (nsym*sizeof (unsigned short));
    check (!acm->freq, "arithmetic coder model allocation failure");
    acm->cfreq = (unsigned short *) calloc (nsym+1, sizeof (unsigned short));
    check (!acm->cfreq, "arithmetic coder model allocation failure");

    for (i=0; i<acm->nsym; i++) {
        acm->freq[i] = 1;
        acm->cfreq[i] = acm->nsym - i;
    }
    acm->cfreq[acm->nsym] = 0;

    return;
}
```

The acd is structures which contains the decoding variables and whose addresses act as handles for the decoded symbol/bitstreams. The fields bits_to_go, buffer, bitstream, and bitstream_len are used to manage the bits in memory. The low, high, and fbits fields describe the scaled range corresponding to the symbols which have been decoded. The value field contains the currently seen code value inside the range. The total_bits field contains the total number of bits encoded or used for decoding so far. The values Code_value_bits and Top_value describe the maximum number of bits and the maximum size of a coded value respectively. The ac_model structure contains the variables used for that particular probability model and it's address acts as a handle. The nsym field contains the number of symbols in the symbol set, the freq field contains the table of frequency counts for each of the nsym symbols, and the cfreq field contains the cumulative frequency count derived from freq.

The bits are read from the bitstream using the function:

```
static int input_bit (ac_decoder *acd) {
    int t;
    unsigned int tmp;

    if (acd->bits_to_go==0) {
        acd->buffer = ace->bitstream[ace->bitstream_len++];
        acd->bits_to_go = 8;
    }

    t = acd->buffer & 0x080;
    acd->buffer <<= 1;
    acd->buffer &= 0x0ff;
    acd->total_bits += 1;
    acd->bits_to_go -= 1;
    t = t >> 7;

    return t;
}
```

The decoding process has four main steps. The first step is to decode the symbol based on the current state of the probability model (frequency counts) and the current code value (value) which is used to represent (and is a member of) the current range. The second step is to get the new range. The third step is to rescale the range and

© ISO/IEC 2004 – All rights reserved

simultaneously load in new code value bits. The fourth step is to update the model. To decode symbols, the following function is called:

```
int ac_decode_symbol (ac_decoder *acd, ac_model *acm) {
    long range;
    int cum;
    int sym;

    range = (long)(acd->high-acd->low)+1;

    /*--- decode symbol ---*/
    cum = (((long)(acd->value-acd->low)+1)*(int)(acm->cfreq[0])-1)/range;
    for (sym = 0; (int)acm->cfreq[sym+1]>cum; sym++)
        /* do nothing */ ;

    check (sym<0||sym>=acm->nsym, "symbol out of range");

    /*--- Get new range ---*/
    acd->high = acd->low + (range*(int)(acm->cfreq[sym]))/(int)(acm->cfreq[0])-1;
    acd->low = acd->low +  (range*(int)(acm->cfreq[sym+1]))/(int)(acm->cfreq[0]);

    /*--- rescale and load new code value bits ---*/
    for (;;) {
        if (acd->high<Half) {
            /* do nothing */
        } else if (acd->low>=Half) {
            acd->value -= Half;
            acd->low -= Half;
            acd->high -= Half;
        } else if (acd->low>=First_qtr && acd->high<Third_qtr) {
            acd->value -= First_qtr;
            acd->low -= First_qtr;
            acd->high -= First_qtr;
        } else
            break;
        acd->low = 2*acd->low;
        acd->high = 2*acd->high+1;
        acd->value = 2*acd->value + input_bit(acd);
    }

    /*--- Update probability model ---*/
    update_model (acm, sym);

    return sym;
}
```

The update of the probability model used in the decoding of the symbols is shown in the following function:

```
static void update_model (ac_model *acm, int sym)
{
    int i, Max_frequency =127;
```

© ISO/IEC 2004 – All rights reserved

```
    if (acm->cfreq[0]==Max_frequency) {
        int cum = 0;
        acm->cfreq[acm->nsym] = 0;
        for (i = acm->nsym-1; i>=0; i--) {
            acm->freq[i] = ((int)acm->freq[i] + 1) / 2;
            cum += acm->freq[i];
            acm->cfreq[i] = cum;
        }
    }

    acm->freq[sym] += 1;
    for (i=sym; i>=0; i--)
        acm->cfreq[i] += 1;

    return;
}
```

This function updates the frequency counts based on the symbol just decoded. It also makes sure that the maximum frequency allowed is not exceeded. This is done by rescaling all frequency counts by 2.

B.2.2 Arithmetic decoding for shape decoding

B.2.2.1 Structures and Typedefs

```
typedef void Void;
typedef int Int;
typedef unsigned short int USInt;
#define  CODE_BIT 32
#define  HALF  ((unsigned) 1 << (CODE_BITS-1))
#define  QUARTER  (1 << (CODE_BITS-2))
struct arcodec {
  UInt L; /* lower bound */
  UInt R; /* code range */
  UInt V; /* current code value */
  UInt arpipe;
  Int bits_to_follow; /* follow bit count */
  Int first_bit;
  Int nzeros;
  Int nonzero;
  Int nzerosf;
  Int extrabits;
};
typedef struct arcodec ArCoder;
typedef struct arcodec ArDecoder;
#define MAXHEADING 3
#define MAXMIDDLE 10
#define MAXTRAILING 2
```

B.2.2.2 Decoder Source

```
Void StartArDecoder(ArDecoder *decoder, Bitstream *bitstream) {
  Int i,j;
  decoder->V = 0;
```


© ISO/IEC 2004 – All rights reserved

```
  decoder->nzerosf = MAXHEADING;
  decoder->extrabits = 0;
  for (i = 1; i<CODE_BITS; i++) {
    j=BitstreamLookBit(bitstream,i+decoder->extrabits);
    decoder->V += decoder->V + j;
    if (j == 0) {
      decoder->nzerosf--;
      if (decoder->nzerosf == 0) {
        decoder->extrabits++;
        decoder->nzerosf = MAXMIDDLE;
      }
    }
    else
      decoder->nzerosf = MAXMIDDLE;
  }
  decoder->L = 0;
  decoder->R = HALF - 1;
  decoder->bits_to_follow = 0;
  decoder->arpipe = decoder->V;
  decoder->nzeros = MAXHEADING;
  decoder->nonzero = 0;
}
Void StopArDecoder(ArDecoder *decoder, Bitstream *bitstream) {
  Int a = decoder->L >> (CODE_BITS-3);
  Int b = (decoder->R + decoder->L) >> (CODE_BITS-3);
  Int nbits,i;
  if (b -- 0)
    b = 8;
  if (b-a >= 4 || (b-a == 3 && a&1))
    nbits = 2;
  else
    nbits = 3;
  for (i = 1; i <= nbits-1; i++)
    AddNextInputBit(bitstream, decoder);
  if (decoder->nzeros < MAXMIDDLE-MAXTRAILING || decoder->nonzero == 0)
    BitstreamFlushBits(bitstream,1);
}
Void AddNextInputBit(Bitstream *bitstream, ArDecoder *decoder) {
  Int i;
  if (((decoder->arpipe >> (CODE_BITS-2))&1) == 0) {
    decoder->nzeros--;
    if (decoder->nzeros == 0) {
      BitstreamFlushBits(bitstream,1);
      decoder->extrabits--;
      decoder->nzeros = MAXMIDDLE;
      decoder->nonzero = 1;
    }
  }
  else {
    decoder->nzeros = MAXMIDDLE;
    decoder->nonzero = 1;
  }
  BitstreamFlushBits(bitstream,1);
  i = (Int)BitstreamLookBit(bitstream, CODE_BITS-1+decoder->extrabits);
  decoder->V += decoder->V + i;
```

© ISO/IEC 2004 – All rights reserved

```
    decoder->arpipe += decoder->arpipe + i;
    if (i == 0) {
      decoder->nzerosf--;
      if (decoder->nzerosf == 0) {
        decoder->nzerosf = MAXMIDDLE;
        decoder->extrabits++;
      }
    }
    else
      decoder->nzerosf = MAXMIDDLE;
}
Int ArDecodeSymbol(USInt c0, ArDecoder *decoder, Bitstream *bitstream ) {
    Int bit;
    Int c1 = (1<<16) - c0;
    Int LPS = c0 > c1;
    Int cLPS = LPS ? c1 : c0;
    unsigned long rLPS;
    rLPS = ((decoder->R) >> 16) * cLPS;
    if ((decoder->V - decoder->L) >= (decoder->R - rLPS)) {
      bit = LPS;
      decoder->L += decoder->R - rLPS;
      decoder->R = rLPS;
    }
    else {
      bit = (1-LPS);
      decoder->R -= rLPS;
    }
    DECODE_RENORMALISE(decoder,bitstream);
    return(bit);
}
Void DECODE_RENORMALISE(ArDecoder *decoder,  Bitstream *bitstream) {
    while (decoder->R < QUARTER) {
      if (decoder->L >= HALF) {
        decoder->V -= HALF;
        decoder->L -= HALF;
        decoder->bits_to_follow = 0;
      }
      else
        if (decoder->L + decoder->R <= HALF)
          decoder->bits_to_follow = 0;
        else{
          decoder->V -= QUARTER;
          decoder->L -= QUARTER;
          (decoder->bits_to_follow)++;
        }
      decoder->L += decoder->L;
      decoder->R += decoder->R;
      AddNextInputBit(bitstream, decoder);
    }
}
```

- BitstreamLookBit(bitstream,nbits) : Looks nbits ahead in the bitstream beginning from the current position in the bitstream and returns the bit.
- BitstreamFlushBits(bitstream,nbits) : Moves the current bitstream position forward by nbits.

© ISO/IEC 2004 – All rights reserved

- The parameter c0 (used in ArDecodeSymbol()) is taken directly from the probability tables of USint inter_prob or Usint intra_prob in Table B.32. That is, for the pixel to be coded/decoded, c0 is the probability than this pixel is equal to zero. The value of c0 depends on the context number of the given pixel to be decoded.

B.2.3 FBA Object Decoding

In FBA decoder, a symbol is decoded by using a specific model based on the syntax and by calling the following procedure which is specified in C.

```
static long low, high, code_value, bit, length, sacindex, cum, zerorun=0;

int aa_decode(int cumul_freq[ ])
{
  length = high - low + 1;
  cum = (-1 + (code_value - low + 1) * cumul_freq[0]) / length;
  for (sacindex = 1; cumul_freq[sacindex] > cum; sacindex++);
  high = low - 1 + (length * cumul_freq[sacindex-1]) / cumul_freq[0];
  low += (length * cumul_freq[sacindex]) / cumul_freq[0];

  for ( ; ; ) {
    if (high < q2) ;
    else if (low >= q2) {
      code_value -= q2;
      low -= q2;
      high -= q2;
    }
    else if (low >= q1 && high < q3) {
      code_value -= q1;
      low -= q1;
      high -= q1;
    }
    else {
      break;
    }
    low *= 2;
    high = 2*high + 1;
    bit_out_psc_layer();
    code_value = 2*code_value + bit;
    used_bits++;
  }
  return (sacindex-1);
}

void bit_out_psc_layer()
{
  bit = getbits(1);
}
```

© ISO/IEC 2004 – All rights reserved

Again the model is specified through `cumul_freq[ ]`. The decoded symbol is returned through its index in the model. The decoder is initialized to start decoding an arithmetic coded bitstream by calling the following procedure.

```
void decoder_reset( )
{
  int i;
  zerorun = 0;          /* clear consecutive zero's counter */
  code_value = 0;
  low = 0;
  high = top;
  for (i = 1;   i <= 16;   i++) {
    bit_out_psc_layer();
    code_value = 2 * code_value + bit;
  }
  used_bits = 0;
}
```

B.3 Variable length codes for the studio profile

B.3.1 Macroblock type

The properties of the macroblock are determined by the macroblock type VLC according to these tables.

Table B.42 — Variable length codes for macroblock_type in I-VOPs

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	Description
1	0	0	0	0	1	Intra
01	1	0	0	0	1	Intra, Quant

Table B.43 — Variable length codes for macroblock_type in P-VOPs

macroblock_type VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra	Description
1	0	1	0	1	0	MC, Coded
01	0	0	0	1	0	No MC, Coded
001	0	1	0	0	0	MC, Not Coded
0001 1	0	0	0	0	1	Intra
0001 0	1	1	0	1	0	MC, Coded, Quant
0000 1	1	0	0	1	0	No MC, Coded, Quant
0000 01	1	0	0	0	1	Intra, Quant

© ISO/IEC 2004 – All rights reserved

B.3.2 DCT coefficients

Table B.44 — Variable length codes for dct_dc_size_luminance

Variable length code	dct_dc_size_luminance
001110	0
00110	1
0000	2
0010	3
111	4
101	5
011	6
010	7
100	8
110	9
0001	10
0011110	11
00111110	12
001111110	13
0011111110	14
00111111110	15
001111111110	16
0011111111110	17
0011111111111	18

Table B.45 — Variable length codes for dct_dc_size_chrominance

Variable length code	dct_dc_size_chrominance
0000	0
0010	1
111	2
101	3
011	4
010	5
100	6
110	7
0001	8
00110	9
001110	10
0011110	11
00111110	12
001111110	13
0011111110	14
00111111110	15
001111111110	16
0011111111110	17
0011111111111	18

© ISO/IEC 2004 – All rights reserved

Table B.46 — Differential DC additional codes

Additional code	Differential DC	Size
000000000000000000 to 011111111111111111 *	-262143 to -131072	18
00000000000000000 to 01111111111111111 *	-131071 to -65536	17
0000000000000000 to 0111111111111111 *	-65535 to -32768	16
000000000000000 to 011111111111111 *	-32767 to -16384	15
00000000000000 to 01111111111111 *	-16383 to -8192	14
0000000000000 to 0111111111111 *	-8191 to -4096	13
000000000000 to 011111111111 *	-4095 to -2048	12
00000000000 to 01111111111 *	-2047 to -1024	11
0000000000 to 0111111111 *	-1023 to -512	10
000000000 to 011111111 *	-511 to -256	9
00000000 to 01111111	-255 to -128	8
0000000 to 0111111	-127 to -64	7
000000 to 011111	-63 to -32	6
00000 to 01111	-31 to -16	5
0000 to 0111	-15 to -8	4
000 to 011	-7 to -4	3
00 to 01	-3 to -2	2
0	-1	1
	0	0
1	1	1
10 to 11	2 to 3	2
100 to 111	4 to 7	3
1000 to 1111	8 to 15	4
10000 to 11111	16 to 31	5
100000 to 111111	32 to 63	6
1000000 to 1111111	64 to 127	7
10000000 to 11111111	128 to 255	8
100000000 to 111111111 *	256 to 511	9
1000000000 to 1111111111 *	512 to 1023	10
10000000000 to 11111111111 *	1024 to 2047	11
100000000000 to 111111111111 *	2048 to 4095	12
1000000000000 to 1111111111111 *	4096 to 8191	13
10000000000000 to 11111111111111 *	8192 to 16383	14
100000000000000 to 111111111111111 *	16384 to 32767	15
1000000000000000 to 1111111111111111 *	32768 to 65535	16
10000000000000000 to 11111111111111111 *	65536 to 131071	17
100000000000000000 to 111111111111111111 *	131072 to 262143	18

© ISO/IEC 2004 – All rights reserved

In cases where dct_dc_size is greater than 8, marked '*' in Table B46, a marker bit is inserted after the dct_dc_additional_code to prevent start code emulations.

Table B.47 — Additional codes for group No.1~6 (zero run length)

Size	Additional code	zero run length
0	---	1
1	0 to 1	2 to 3
2	00 to 11	4 to 7
3	000 to 111	8 to 15
4	0000 to 1111	16 to 31
5	00000 to 11111	32 to 63

Table B.48 — Additional codes for group No.7~12 (zero run length and +/-1 level)

Size	Additional code	zero run length	level(+1/-1)
6	000000 to 111110	32 to 63	-1
5	00000 to 11110	16 to 31	-1
4	0000 to 1110	8 to 15	-1
3	000 to 110	4 to 7	-1
2	00 to 10	2 to 3	-1
1	0	1	-1
0	---	---	---
1	1	1	+1
2	01 to 11	2 to 3	+1
3	001 to 111	4 to 7	+1
4	0001 to 1111	8 to 15	+1
5	00001 to 11111	16 to 31	+1
6	000001 to 111111	32 to 63	+1

Table B.49 — Additional codes for group No.13~20 (level value)

Size	Additional code	level value
8	00000000 to 01111111	-255 to -128
7	0000000 to 0111111	-127 to -64
6	000000 to 011111	-63 to -32
5	00000 to 01111	-31 to -16
4	0000 to 0111	-15 to -8
3	000 to 011	-7 to -4
2	00 to 01	-3 to -2
1	0	-1
0	---	---
1	1	1
2	10 to 11	2 to 3
3	100 to 111	4 to 7
4	1000 to 1111	8 to 15
5	10000 to 11111	16 to 31
6	100000 to 111111	32 to 63
7	1000000 to 1111111	64 to 127
8	10000000 to 11111111	128 to 255

© ISO/IEC 2004 – All rights reserved

Table B.50 — VLC Table T[0] for intra macroblock

symbol group	variable length code
0	0101
1	0100
2	0000101
3	000001001
4	00000100001
5	000001000001
6	0000010000001
7	0011
8	00011
9	000101
10	0000100
11	0000011
12	00000101
13	11
14	101
15	100
16	011
17	0010
18	000100
19	000011
20	0000010001
21	0000010000000

© ISO/IEC 2004 – All rights reserved

Table B.51 — VLC Table T[1] for intra macroblock

symbol group	variable length code
0	---
1	---
2	---
3	---
4	---
5	---
6	---
7	---
8	---
9	---
10	---
11	---
12	---
13	---
14	1
15	01
16	001
17	0001
18	00001
19	0000011
20	00000101
21	00000100

© ISO/IEC 2004 – All rights reserved

Table B.52 — VLC Table T[2] for intra macroblock

symbol group	variable length code
0	101
1	00011
2	00010
3	0000011
4	000001001
5	00000100000011
6	00000100000010
7	100
8	011
9	0011
10	0010
11	000011
12	0000010001
13	11
14	010
15	000010
16	00000101
17	00000100001
18	0000010000011
19	00000100000001
20	000001000000001
21	0000010000010

© ISO/IEC 2004 – All rights reserved

Table B.53 — VLC Table T[3] for intra macroblock

symbol group	variable length code
0	00101
1	0101
2	00100
3	000011
4	000001001
5	0000010000011
6	0000010000010
7	011
8	0100
9	0011
10	00011
11	00000101
12	0000010000001
13	11
14	10
15	00010
16	000010
17	0000011
18	0000010001
19	000001000011
20	0000010000000
21	000001000010

© ISO/IEC 2004 – All rights reserved

Table B.54 — VLC Table T[4] for intra macroblock

symbol group	variable length code
0	0000101
1	0011
2	00011
3	0000100
4	000001001
5	0000010000011
6	00000100000001
7	011
8	0010
9	000101
10	0000011
11	0000010001
12	000001000000001
13	11
14	10
15	010
16	000100
17	000011
18	00000101
19	00000100001
20	0000010000010
21	0000010000001

© ISO/IEC 2004 – All rights reserved

Table B.55 — VLC Table T[5] for intra macroblock

symbol group	variable length code
0	0000010011
1	00011
2	0000101
3	0000010010
4	000001000011
5	0000010000011
6	0000010000010
7	00010
8	0000100
9	00000101
10	00000100011
11	0000010000001
12	00000100000001
13	11
14	10
15	01
16	001
17	000011
18	0000011
19	00000100010
20	000001000000001
21	000001000010

© ISO/IEC 2004 – All rights reserved

Table B.56 — VLC Table T[6] for intra macroblock

symbol group	variable length code
0	00000100011
1	0001
2	00000111
3	0000010011
4	00000100010
5	00000100000011
6	00000100000010
7	000011
8	00000110
9	0000010010
10	000001000011
11	00000100000001
12	000001000000001
13	011
14	010
15	11
16	10
17	001
18	000010
19	00000101
20	000001000010
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.57 — VLC Table T[7] for intra macroblock

symbol group	variable length code
0	000001011
1	00011
2	00000111
3	0000111
4	0000110
5	00000100011
6	000001000001
7	0000101
8	00000110
9	000001010
10	0000010011
11	00000100010
12	000001000000
13	0011
14	0010
15	11
16	10
17	01
18	00010
19	0000100
20	0000010010
21	00000100001

© ISO/IEC 2004 – All rights reserved

Table B.58 — VLC Table T[8] for intra macroblock

symbol group	variable length code
0	0000010101
1	000011
2	0000010100
3	00000100011
4	00000111
5	000001000011
6	0000010000001
7	00000110
8	000001011
9	0000010011
10	0000010010
11	000001000010
12	0000010000000
13	0001
14	011
15	010
16	11
17	10
18	001
19	000010
20	00000100010
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.59 — VLC Table T[9] for intra macroblock

symbol group	variable length code
0	000001000011
1	000101
2	00000111
3	000100
4	000011
5	0000010011
6	000001000010
7	0000101
8	0000100
9	00000110
10	0000010010
11	000001000001
12	000001000000
13	00011
14	0011
15	011
16	010
17	11
18	10
19	0010
20	00000101
21	0000010001

© ISO/IEC 2004 – All rights reserved

Table B.60 — VLC Table T[10] for intra macroblock

symbol group	variable length code
0	0000010000011
1	0000101
2	00000111
3	0011
4	00000100001
5	0000010000010
6	0000010000001
7	0000100
8	00000110
9	000001011
10	000001010
11	0000010001
12	0000010000000
13	00011
14	00010
15	0010
16	011
17	010
18	11
19	10
20	000011
21	000001001

© ISO/IEC 2004 – All rights reserved

Table B.61 — VLC Table T[11] for intra macroblock

symbol group	variable length code
0	0000010011
1	00011
2	000011
3	000001101
4	000001100
5	00000100001
6	00000100000
7	00010
8	000010
9	00000111
10	000001011
11	0000010010
12	0000010001
13	101
14	100
15	0101
16	0100
17	0011
18	0010
19	011
20	11
21	000001010

© ISO/IEC 2004 – All rights reserved

Table B.62 — VLC Table T[0] for inter macroblock

symbol group	variable length code
0	---
1	00101
2	00100
3	0000011
4	000001001
5	00000100001
6	0000010000010
7	101
8	100
9	0101
10	0100
11	00011
12	00010
13	11
14	011
15	0011
16	000011
17	000010
18	00000101
19	0000010001
20	0000010000001
21	0000010000000

© ISO/IEC 2004 – All rights reserved

Table B.63 — VLC Table T[1] for inter macroblock

symbol group	variable length code
0	---
1	---
2	---
3	---
4	---
5	---
6	---
7	---
8	---
9	---
10	---
11	---
12	---
13	---
14	1
15	01
16	001
17	0001
18	00001
19	0000011
20	00000101
21	00000100

© ISO/IEC 2004 – All rights reserved

Table B.64 — VLC Table T[2] for inter macroblock

symbol group	variable length code
0	101
1	00101
2	00100
3	000011
4	0000011
5	0000010001
6	0000010000011
7	100
8	011
9	010
10	0011
11	00011
12	00000101
13	11
14	00010
15	000010
16	000001001
17	000001000011
18	0000010000010
19	0000010000001
20	0000010000000
21	000001000010

© ISO/IEC 2004 – All rights reserved

Table B.65 — VLC Table T[3] for inter macroblock

symbol group	variable length code
0	0101
1	0100
2	00011
3	000101
4	000100
5	0000010001
6	000001000001
7	101
8	100
9	0011
10	0010
11	000011
12	00000101
13	11
14	011
15	000010
16	0000011
17	000001001
18	00000100001
19	00000100000001
20	000001000000001
21	0000010000001

© ISO/IEC 2004 – All rights reserved

Table B.66 — VLC Table T[4] for inter macroblock

symbol group	variable length code
0	000101
1	0101
2	00101
3	000100
4	000011
5	00000100001
6	0000010000001
7	011
8	0100
9	0011
10	00100
11	0000011
12	000001001
13	11
14	10
15	00011
16	000010
17	00000101
18	0000010001
19	00000100000001
20	000001000000001
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.67 — VLC Table T[5] for inter macroblock

symbol group	variable length code
0	000101
1	011
2	00101
3	0000101
4	0000100
5	000001000011
6	0000010000001
7	0011
8	00100
9	000100
10	0000011
11	00000101
12	0000010001
13	11
14	10
15	010
16	00011
17	000011
18	000001001
19	000001000010
20	0000010000000
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.68 — VLC Table T[6] for inter macroblock

symbol group	variable length code
0	000101
1	0011
2	000100
3	0000101
4	00000101
5	000001000011
6	0000010000001
7	00101
8	00100
9	000011
10	0000100
11	000001001
12	0000010001
13	011
14	11
15	10
16	010
17	00011
18	0000011
19	000001000010
20	0000010000000
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.69 — VLC Table T[7] for inter macroblock

symbol group	variable length code
0	0000101
1	0011
2	000101
3	000001011
4	0000010011
5	00000100011
6	000001000001
7	00011
8	000100
9	0000100
10	0000011
11	000001010
12	00000100010
13	011
14	010
15	11
16	10
17	0010
18	000011
19	0000010010
20	000001000000
21	00000100001

© ISO/IEC 2004 – All rights reserved

Table B.70 — VLC Table T[8] for inter macroblock

symbol group	variable length code
0	000101
1	0011
2	0000111
3	0000010001
4	000001000011
5	0000010000001
6	0000010000000
7	000100
8	0000110
9	0000101
10	0000100
11	00000101
12	000001001
13	0010
14	011
15	010
16	11
17	10
18	00011
19	0000011
20	000001000010
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.71 — VLC Table T[9] for inter macroblock

symbol group	variable length code
0	000101
1	0011
2	0000101
3	0000100
4	00000100001
5	0000010000001
6	0000010000000
7	0010
8	000100
9	000011
10	0000011
11	0000010001
12	000001001
13	111
14	110
15	101
16	100
17	011
18	010
19	00011
20	00000101
21	000001000001

© ISO/IEC 2004 – All rights reserved

Table B.72 — VLC Table T[10] for inter macroblock

symbol group	variable length code
0	00000100011
1	0011
2	0000011
3	00000100010
4	0000010000011
5	0000010000010
6	0000010000001
7	00011
8	000011
9	00000101
10	000001001
11	000001000011
12	0000010000000
13	101
14	100
15	11
16	011
17	0010
18	010
19	00010
20	000010
21	000001000010

© ISO/IEC 2004 – All rights reserved

Table B.73 — VLC Table T[11] for inter macroblock

symbol group	variable length code
0	0000010101
1	0001
2	000011
3	0000010100
4	000001000011
5	0000010000001
6	00000100000001
7	000001011
8	0000010011
9	0000010010
10	00000100011
11	00000100010
12	000001000000001
13	001
14	11
15	10
16	01
17	00000111
18	00000110
19	000010
20	000001000010
21	000001000001

© ISO/IEC 2004 – All rights reserved

B.3.3 Shape Coding

Table B.74 — Meaning of shape mode

Index	Shape mode
0	"MVs==0 && No Update"
1	"MVs!=0 && No Update"
2	transparent
3	opaque
4	“HHC”

Table B.75 — bab_type for I-VOP

Index	(2)	(3)	(4)
0	1	001	01
1	001	01	1
2	01	001	1

Table B.76 — bab_type for P-VOP

		bab_type in current VOP (n)				
		0	**1**	**2**	**3**	**4**
bab_type in previous VOP(n-1)	**0**	1	01	00001	000001	0001
	1	01	1	00001	000001	001
	2	0001	001	1	000001	01
	3	1	0001	000001	001	01
	4	011	001	0001	00001	1

Table B.77 — VLC table for MVs

MVs	Codes
0	0
±1	10s
±2	110s
±3	1110s
±4	11110s
±5	111110s
±6	1111110s
±7	11111110s
±8	111111110s
±9	1111111110s
±10	11111111110s
±11	111111111110s
±12	1111111111110s
±13	11111111111110s
±14	111111111111110s
±15	1111111111111110s
±16	11111111111111110s

© ISO/IEC 2004 – All rights reserved

Table B.78 — VLC table for MVs (Horizontal element is 0)

MVDs	Codes
±1	0s
±2	10s
±3	110s
±4	1110s
±5	11110s
±6	111110s
±7	1111110s
±8	11111110s
±9	111111110s
±10	1111111110s
±11	11111111110s
±12	111111111110s
±13	1111111111110s
±14	11111111111110s
±15	111111111111110s
±16	1111111111111110s
s: sign bit (if MVs is positive s="1", otherwise s="0").	

Table B.79 — VLC table for cbbp

cbbp index	code	cbbp index	code
1	1000	9	0000001
2	1010	10	1011
3	111	11	0111
4	1001	12	110
5	001	13	0101
6	000001	14	0110
7	0100	15	00001
8	0001		

© ISO/IEC 2004 – All rights reserved

Table B.80 — VLC table for clp

clp index	code	clp index	code	clp index	code
0	00111	86	0010111101001000000001010	172	001011110100100000010011
1	01	87	10000111100101110	173	00101111010010000010010
2	100010	88	100000111110100	174	00101111010010000011010
3	1111	89	0010111101001000000111110	175	100000010001001
4	00101001	90	001011110100100000010101	176	00101111011111
5	100000011000	91	001011110100100000000001	177	00101000010100011
6	100000110	92	001011110100100001	178	0010111101001000000000111
7	1110	93	0010111101001000000110110	179	00110110000101111
8	10000010	94	10000111100101011	180	100000011011110001
9	001011100101	95	00110110111101	181	0010111101001000000111001
10	0010111001100	96	1000010101	182	001011110100100000011001
11	001101100010	97	001011110000000	183	00110110000101110
12	0011011100	98	00110110111100	184	00110110111011
13	001101101100	99	0010100001100	185	00110110000101101
14	0010111011	100	100001111000110	186	001101101101001010
15	1100	101	0010111101001000000001001	187	001101101101000111
16	100000000	102	00110110000100010	188	0011011000001101
17	0010100001011	103	00101110011110	189	001101101101000100
18	0010111001101	104	10000001101110	190	001101100000100
19	10000100101010	105	00110110000100110	191	100000011010
20	10000011110110	106	00110110000101011	192	1011
21	001101101101001100	107	10000111100101100	193	001010001
22	001101100000011	108	001101101101001011	194	0011011000011
23	1000011110000	109	001101101101000000	195	1000010001
24	1000000010	110	0010111101001000000011	196	1000001111010
25	00110110111010	111	10000011111000	197	0010111101001000000110111
26	1000011110011	112	1000000011	198	10000101100110
27	0010111101100	113	001011100111110	199	0011011010

© ISO/IEC 2004 – All rights reserved

28	1000000111	114	100001111000111	200	0011011011011
29	10000001101101	115	100000111110011	201	1000001111101111
30	0011011001	116	100000010001000	202	00110110000101100
31	10011	117	00101111010010000011111	203	001101100000010
32	001100	118	100000011011110011	204	00101111011110
33	001011110101	119	001011110100101	205	100000011011110000
34	100001001011001	120	0011011110	206	00110110111001
35	0010111101101	121	00101000011011	207	0010111010
36	0010111101000	122	1000001111101110	208	00101000001
37	00110110000100111	123	100001111000100	209	0011011000001011
38	00101111010010000011100	124	0010101000	210	001101101101001101
39	001010000111	125	100000011011111	211	0010100001010000
40	100001111000101	126	10000001001	212	10000111100101111
41	1000010010111011	127	10000110	213	00101111010010000001101
42	0010100001010001	128	1010	214	001011110100100000001011
43	10000111100101101	129	100001110	215	1000001111101101
44	1000001111101010	130	1000001110	216	10000100101111
45	00101111010010000011000	131	1000010100	217	00110110000101010
46	0010111101001001	132	0010111001000	218	00110110000101001
47	1000010110010	133	0011011000001100	219	1000001111101100
48	1000010111	134	10000101100111	220	100000111110010
49	100001001011010	135	1000010011	221	001101101101001001
50	00110110000000	136	10000100101011	222	001101101101001000
51	00110110110101	137	001011110000001	223	00101110001
52	0010111100010	138	00101111010010000000101	224	000
53	001101101101000101	139	100000010001011	225	001010101
54	1000011110010011	140	00101110011101	226	1000000100011
55	00110110111000	141	001011110100100000001000	227	0011011101
56	0010111111	142	1000011110010001	228	001010000100
57	00101000011010	143	0010111110	229	00101111010010000010000

© ISO/IEC 2004 – All rights reserved

58	001011100111111	144	100000111111	230	00101000010101
59	00110110000100101	145	00110110000101000	231	0010101001
60	10000111101	146	00101111010010000000110	232	100001011000
61	00101111010011	147	1000000110111101	233	001011110100100000001100
62	1000000101	148	00101111010010000111011	234	001101101101000011
63	100011	149	0010111101001000010001	235	1000001111101011
64	00101011	150	001101101101000010	236	10000011110111
65	0010111100011	151	1000011110010100	237	00110110000100100
66	100000010000	152	100000010001010	238	0011011000001010
67	0011011011111	153	0010100001010010	239	00101110000
68	100001001011000	154	00101111010010000111010	240	1101
69	0010111101001000010110	155	001101100001000011	241	1000011111
70	001101100000111110	156	100001001011100	242	1000010010100
71	0010111101110	157	001101101101001110	243	10000101101
72	10000001101100	158	100001001011011	244	001101100011
73	0010111101001000000100	159	10000100100	245	001101100001000000
74	00101111010010001	160	00101111001	246	00101111000001
75	001101101101001111	161	00101110011100	247	00101000000
76	001101100001000001	162	001101100000111111	248	10010
77	00101111010010000111101	163	10000111100101010	249	1000010000
78	001101101101000110	164	1000011110010010	250	100000111100
79	0010111001001	165	100000011011110010	251	100000011001
80	0010111100001	166	0010111101001000010111	252	00100
81	1000011110010000	167	0011011000001110	253	0011011111
82	1000010010111010	168	00101111010010000111100	254	0010110
83	001101101101000001	169	0010111101001000001111	255	0011010
84	0010111101001000001110	170	001011110100100000000		
85	0010111101001000000010	171	0010111101001000010100		

© ISO/IEC 2004 – All rights reserved

Table B.81 — VLC table for lbp

lbp index	code	lbp index	code	lbp index	code
1	0111110110	86	0111101001110110101	171	01111101110100110011
2	01110101111	87	011101010101011011100101	172	011101101100011
3	0111010011	88	01111010011110	173	01110101010101101101
4	01110101001	89	011110100110001100	174	0111011001111011
5	01111101000110111	90	011110100111011001 1	175	0111110000001101
6	0111110100100	91	01111010011101 1000	176	01110110010
7	011101101010	92	011101101100010	177	011100001011111
8	0111001010	93	011101010101011011 00100	178	0111001011011100
9	0111111011001010	94	01110110011111000	179	0111011001101100
10	0111011001110110	95	01110110011111010	180	011100001011110
11	01110110011111001	96	01110011	181	011111111101010011
12	01110101000	97	0111000011110	182	01111101110000011
13	01111000111101111	98	011101010101010	183	0111110000001100
14	011100001001	99	011111000000100	184	011110001110
15	011111111100	100	01110101110010	185	011100101100001
16	011110010	101	01111101110000000	186	011111101100101111
17	01111101000111	102	0111011001111000	187	0111011001111010
18	011101101100001	103	0111001011010101	188	011110001101
19	0111011001100001	104	01110010110110	189	01111101110000010
20	01110110110010	105	011110100110001101	190	0111010101100
21	011101010101011011 00110	106	01110010110111010	191	0111101001101
22	011101001001100101	107	011101010101011011 00010	192	100
23	0111010101010110111	108	011100101101011	193	011100100
24	0111101101	109	0111101001110110100	194	011100001100
25	011101001001101	110	011101010101011000	195	0111010110
26	01111111110101011	111	01110010110111100	196	011101101011
27	011111111101010000	112	01110001	197	011100101100000

© ISO/IEC 2004 – All rights reserved

28	01110100101	113	0111010101001	198	0111000010110
29	011111111101010100	114	011101010110110	199	011101001000
30	0111110100010	115	01111010011100	200	011111111111
31	011100001000	116	01110101110011	201	011111101100101110
32	011111100	117	0111011001110101	202	0111011001100000
33	01111010011001	118	011110001100000	203	011101100111111110
34	011111011101011	119	011111011100001	204	011101011101
35	011111011100000001	120	011110111	205	011101100111111101
36	01110100100111	121	01111111110111	206	01111000110010
37	011101010101011001	122	01111000111100	207	0111110100000
38	011101100111111011	123	01110110011100	208	0111101100
39	011110100110001011	124	011110000	209	01110010110100
40	0111010010010	125	01111101110001	210	0111101001100000
41	011101010101011010	126	0111100110	211	011110001111010
42	01111010011101101111	127	0111100111	212	011100101100001
43	011101001001100011	128	001	213	011101100111111100
44	0111011001101101	129	01111101111	214	011101010101001
45	01111010011101101110	130	01111111110110	215	011111000000111
46	011101001001100010	131	011100001110	216	011111111110
47	011100101101010001	132	0111100011111	217	01111101110100001
48	011111110	133	011100101101010000	218	011101010101000
49	0111001011001	134	011111000000101	219	01111101110100000
50	011110100111111	135	0111010101011	220	011100001101
51	011101010110101	136	011100001010	221	011110100111010111
52	011111011100010	137	011101100111111111	222	0111010101000
53	011110001111011110	138	011110100110001010	223	0111110111011
54	0111100011000011	139	01111101110100110010	224	000
55	011100101101111110	140	011101101100011	225	0111110101
56	011111001	141	01111010011101101101	226	011101010111

© ISO/IEC 2004 – All rights reserved

57	01111111110100	142	01110110011010	227	0111110001
58	011111011100110	143	0111010111000	228	011111010011
59	0111011001110100	144	01110110100	229	01110000111110
60	011100000	145	011101010110111	230	0111110000000
61	0111100011000010	146	0111011001110111	231	01110110111
62	011110100001	147	01111101110100111	232	01111100001
63	01111010001	148	011111010001100	233	011111111101011
64	01110111	149	011101001001100111	234	011110100111110
65	01111000110011	150	01110010110111011	235	01111000110001
66	0111110100011010	151	011111111101010010	236	01110010111
67	0111011001111001	152	011110100000	237	011101101100000
68	011111011100111	153	01111101000110110	238	0111110100001
69	01110101010101101100001	154	011111011101001101	239	011101101101
70	0111101001110110010	155	011110100110001111	240	010
71	0111111111010100001	156	01111110110011	241	0111111110
72	011111101100100	157	011101001001100100	242	011111101101
73	01111101110100110001	158	01110000111111	243	0111111010
74	01111101110100110000	159	01110000101110	244	0111100010
75	01110101010101101100111	160	01111110111	245	0111110100101
76	0111100011110110	161	011110100111010	246	011111000001
77	01111010011101101100	162	011101100110111	247	0111011000
78	011101010101011011000011	163	01111110111010010	248	1010
79	011101001001100110	164	011111011101010	249	0111101011
80	0111111011000	165	0111111011001011110	250	01111010010
81	011101100110001	166	011111111101010101	251	0111101010
82	01111010011000100	167	01110010110111101	252	1011
83	01110010110111111	168	01110110011001	253	011101000
84	0111010101010111	169	011101001001100001	254	0110
85	011101001001100000	170	011110100110001110	255	11

© ISO/IEC 2004 – All rights reserved

B.3.4 Macroblock pattern

Table B.82 — Variable length codes for coded_block_pattern

coded_block_pattern VLC code	cbp	coded_block_pattern VLC code	cbp
111	60	0001 1100	35
1101	4	0001 1011	13
1100	8	0001 1010	49
1011	16	0001 1001	21
1010	32	0001 1000	41
1001 1	12	0001 0111	14
1001 0	48	0001 0110	50
1000 1	20	0001 0101	22
1000 0	40	0001 0100	42
0111 1	28	0001 0011	15
0111 0	44	0001 0010	51
0110 1	52	0001 0001	23
0110 0	56	0001 0000	43
0101 1	1	0000 1111	25
0101 0	61	0000 1110	37
0100 1	2	0000 1101	26
0100 0	62	0000 1100	38
0011 11	24	0000 1011	29
0011 10	36	0000 1010	45
0011 01	3	0000 1001	53
0011 00	63	0000 1000	57
0010 111	5	0000 0111	30
0010 110	9	0000 0110	46
0010 101	17	0000 0101	54
0010 100	33	0000 0100	58
0010 011	6	0000 0011 1	31
0010 010	10	0000 0011 0	47
0010 001	18	0000 0010 1	55
0010 000	34	0000 0010 0	59
0001 1111	7	0000 0001 1	27
0001 1110	11	0000 0001 0	39
0001 1101	19	0000 0000 1	0 (NOTE)
NOTE — This entry shall not be used with 4:2:0 chrominance structure			

© ISO/IEC 2004 – All rights reserved

Table B.83 — VLC table for cbpa in the case of four non-transparent blocks

Code	cbpa
0011	00 00
0010 1	00 01
0010 0	00 10
1001	00 11
0001 1	01 00
0111	01 01
0000 10	01 10
1011	01 11
0001 0	10 00
0000 11	10 01
0101	10 10
1010	10 11
0100	11 00
1000	11 01
0110	11 10
11	11 11

© ISO/IEC 2004 – All rights reserved

Table B.84 — VLC table for cbpa in the case of three non transparent blocks

Code	cbpa
011	000
000001	001
00001	010
010	011
00010	100
00011	101
001	110
1	111

Table B.85 — VLC table for cbpa in the case of two non transparent blocks

Code	cbpa
0001	00
001	01
01	10
1	11

Table B.86 — VLC table for cbpa in the case of one non transparent block

Code	cbpa
01	0
1	1

© ISO/IEC 2004 – All rights reserved

B.3.5 Motion vector

Table B.87 — Variable length codes for motion_code

Variable length code	motion_code[r][s][t]
0000 0011 001	-16
0000 0011 011	-15
0000 0011 101	-14
0000 0011 111	-13
0000 0100 001	-12
0000 0100 011	-11
0000 0100 11	-10
0000 0101 01	-9
0000 0101 11	-8
0000 0111	-7
0000 1001	-6
0000 1011	-5
0000 111	-4
0001 1	-3
0011	-2
011	-1
1	0
010	1
0010	2
0001 0	3
0000 110	4
0000 1010	5
0000 1000	6
0000 0110	7
0000 0101 10	8
0000 0101 00	9
0000 0100 10	10
0000 0100 010	11
0000 0100 000	12
0000 0011 110	13
0000 0011 100	14
0000 0011 010	15
0000 0011 000	16

© ISO/IEC 2004 – All rights reserved

B.3.6 Sprite Coding

Table B.88 — Code table for the first trajectory point

dmv value	SSS	VLC	dmv_code
-2097151 … -1048576, 1048576 … 2097152	21	111111111111 1111110	00000000000000000000...01111111111111111111 , 10000000000000000000...11111111111111111111
-1048575 … -524288, 524288 … 1048575	20	111111111111 111110	00000000000000000000...01111111111111111111, 10000000000000000000...11111111111111111111
-524287 … -262144, 262144 … -524287	19	111111111111 11110	0000000000000000000...011111111111111111, 1000000000000000000...111111111111111111
-262143 … -131072, 131072 …262143	18	111111111111 1110	000000000000000000...011111111111111111, 100000000000000000...111111111111111111
-131071 … -65536, 65536 … 131071	17	111111111111 110	00000000000000000...01111111111111111, 10000000000000000...11111111111111111
-65535 … -32768, 32768 … 65535	16	111111111111 10	0000000000000000...0111111111111111, 1000000000000000...1111111111111111
-32767 … -16384, 16384 … 32767	15	111111111111 0	000000000000000...011111111111111, 100000000000000...111111111111111
-16383 … -8192, 8192 … 16383	14	111111111110	00000000000000...01111111111111, 10000000000000...11111111111111
-8191 … -4096, 4096 … 8191	13	11111111110	0000000000000...0111111111111, 1000000000000...1111111111111
-4095 … -2048, 2048 … 4095	12	1111111110	000000000000...011111111111, 100000000000...111111111111
-2047...-1024, 1024...2047	11	111111110	00000000000...01111111111, 10000000000...11111111111
-1023...-512, 512...1023	10	11111110	0000000000...0111111111, 1000000000...1111111111
-511...-256, 256...511	9	1111110	000000000...011111111, 100000000...111111111
-255...-128, 128...255	8	111110	00000000...01111111, 10000000...11111111
-127...-64, 64...127	7	11110	0000000...0111111, 1000000...1111111
-63...-32, 32...63	6	1110	000000...011111, 100000...111111
-31...-16, 16...31	5	110	00000...01111, 10000...1111
-15...-8, 8...15	4	101	0000...0111, 1000...1111
-7...-4, 4...7	3	100	000...011, 100...111
-3...-2, 2...3	2	011	00...01, 10...11
-1, 1	1	010	0, 1
0	0	00	-

Table B.89 — Code table for scaled brightness change factor

brightness_change_ factor value	brightness_ change_factor_ length value	brightness_ change_factor_ length VLC	brightness_change_factor
-16...-1, 1...16	1	0	00000...01111, 10000...11111
-48...-17, 17...48	2	10	000000...011111, 100000...111111
112...-49, 49...112	3	110	0000000...0111111, 1000000...1111111
113…624	4	1110	000000000...111111111
625...1648	4	1111	0000000000…1111111111

© ISO/IEC 2004 – All rights reserved

B.4 Variable length codes for FGS

VLC Tables for FGS is specified in this Annex.

B.4.1 table_bpc

Table B.90 — VLC table for MSB block-bp if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	3	101
3	2	0	4	1101
4	3	0	4	0001
5	4	0	5	11110
6	5	0	5	10011
7	6	0	6	110011
8	7	0	6	111010
9	8	0	6	100010
10	9	0	6	100101
11	10	0	7	1110011
12	11	0	7	0000001
13	12	0	8	11101101
14	13	0	8	10001111
15	14	0	9	110000101
16	15	0	9	111011100
17	16	0	9	000000001
18	17	0	9	000010001
19	18	0	9	000010000
20	19	0	10	1100000010
21	20	0	9	110000000
22	21	0	10	1000111010
23	22	0	11	11000011011
24	23	0	11	11101110110
25	24	0	11	10001110010
26	25	0	12	111011001100
27	26	0	12	000000000001
28	27	0	12	100011101100
29	28	0	13	1000111001110
30	29	0	13	1000111001111
31	30	0	14	00000000000001
32	31	0	13	1110010011000
33	32	0	13	1110010011010
34	33	0	13	1110010011011
35	34	0	13	1110010011001
36	35	0	13	0000000000001
37	0	1	3	001
38	1	1	5	11111
39	2	1	5	10000
40	3	1	6	111000
41	4	1	6	000011

© ISO/IEC 2004 – All rights reserved

42	5	1	6	000001
43	6	1	7	1100010
44	7	1	7	1100100
45	8	1	7	1100101
46	9	1	6	100100
47	10	1	7	1100011
48	11	1	7	1000110
49	12	1	7	0000101
50	13	1	8	11000001
51	14	1	8	11100101
52	15	1	8	00000001
53	16	1	9	000010011
54	17	1	9	110000111
55	18	1	9	111001000
56	19	1	9	110000100
57	20	1	8	11101111
58	21	1	9	111011000
59	22	1	9	000010010
60	23	1	10	1100001100
61	24	1	10	1100000011
62	25	1	10	1110111010
63	26	1	11	11000011010
64	27	1	11	11100100100
65	28	1	11	00000000010
66	29	1	11	00000000011
67	30	1	11	10001110111
68	31	1	11	11101100111
69	32	1	11	11100100111
70	33	1	11	11101110111
71	34	1	11	11100100101
72	35	1	10	1000111000
73	36	1	11	00000000001
74	37	1	12	111011001101
75	38	1	12	100011101101
76	39	1	12	100011100110
77	Escape	Code	10	1110110010

Table B.91 — VLC table for MSB-1 block-bp if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	2	10
3	2	0	3	111
4	3	0	4	0010
5	4	0	4	1100
6	5	0	5	00010
7	6	0	5	11010
8	7	0	6	001110
9	8	0	6	000001

© ISO/IEC 2004 – All rights reserved

10	9	0	7	0011000
11	10	0	7	0001111
12	11	0	7	1101100
13	12	0	8	00011101
14	13	0	8	00000011
15	14	0	9	000110000
16	15	0	10	0011001111
17	16	0	10	0000000011
18	17	0	10	0000001001
19	18	0	11	00110011001
20	19	0	11	00110011011
21	20	0	11	00000000001
22	21	0	12	001100110000
23	22	0	12	000000101001
24	23	0	12	110111110000
25	24	0	13	0011001100011
26	25	0	13	0000001011101
27	26	0	13	1101111100010
28	27	0	15	000000000000001
29	0	1	6	001101
30	1	1	6	000011
31	2	1	7	0011111
32	3	1	7	0001101
33	4	1	7	0000100
34	5	1	7	0000101
35	6	1	7	1101101
36	7	1	8	00110010
37	8	1	8	00111100
38	9	1	8	00111101
39	10	1	8	00011100
40	11	1	8	00011001
41	12	1	8	00000001
42	13	1	8	11011110
43	14	1	9	000110001
44	15	1	10	0011001110
45	16	1	10	0000000010
46	17	1	10	0000000001
47	18	1	10	0000001000
48	19	1	10	1101111101
49	20	1	10	1101111111
50	21	1	11	00110011010
51	22	1	11	00000010110
52	23	1	11	11011111100
53	24	1	11	00000010101
54	25	1	11	11011111001
55	26	1	12	000000000001
56	27	1	12	000000101111
57	28	1	13	0000000000001
58	29	1	13	0000001010001

© ISO/IEC 2004 – All rights reserved

59	30	1	13	0000001011100
60	31	1	13	0000001010000
61	32	1	13	1101111100011
62	33	1	14	00000000000001
63	34	1	14	00110011000100
64	35	1	14	00110011000101
65	All	Zero	7	1101110
66	Escape	Code	11	11011111101

Table B.92 — VLC table for MSB-2 block-bp if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	1	1
2	1	0	3	010
3	2	0	3	001
4	3	0	4	0111
5	4	0	5	01100
6	5	0	5	00001
7	6	0	6	011011
8	7	0	6	000101
9	8	0	7	0110100
10	9	0	7	0001110
11	10	0	8	00000101
12	11	0	8	00011001
13	12	0	9	000001000
14	13	0	10	0110101111
15	14	0	10	0000000001
16	15	0	11	00000100101
17	16	0	12	011010111011
18	17	0	12	000001001000
19	18	0	13	0110101110001
20	19	0	13	0110101101100
21	20	0	13	0110101101110
22	21	0	13	0001111000110
23	22	0	14	01101011100000
24	23	0	15	000000000000001
25	0	1	6	000100
26	1	1	7	0000001
27	2	1	7	0001101
28	3	1	8	01101010
29	4	1	8	00000110
30	5	1	8	00000001
31	6	1	8	00011111
32	7	1	8	00011000
33	8	1	9	000001111
34	9	1	9	000001110
35	10	1	9	000000001
36	11	1	9	000111101
37	12	1	10	0110101100

© ISO/IEC 2004 – All rights reserved

38	13	1	10	0000010011
39	14	1	10	0001111001
40	15	1	11	00000000001
41	16	1	12	011010111010
42	17	1	12	011010111001
43	18	1	12	000001001001
44	19	1	12	000000000001
45	20	1	12	000111100010
46	21	1	13	0110101101101
47	22	1	13	0110101101111
48	23	1	13	0000000000001
49	24	1	13	0001111000111
50	25	1	14	01101011100001
51	26	1	14	00000000000001
52	All	Zero	11	01101011010
53	Escape	Code	11	00011110000

Table B.93 — VLC table for MSB-3 block-bp and other block-bps if fw_matrix[0] = 0 or 1

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	1	1
2	1	0	2	01
3	2	0	3	001
4	3	0	5	00010
5	4	0	5	00001
6	5	0	6	000001
7	6	0	7	0001100
8	7	0	8	00011101
9	8	0	9	000111001
10	9	0	9	000000011
11	10	0	10	0000000010
12	11	0	11	00011010100
13	12	0	11	00000001010
14	13	0	12	000000000001
15	14	0	13	0000000011110
16	15	0	13	0000000101101
17	16	0	14	00000000000011
18	0	1	7	0001111
19	1	1	7	0000001
20	2	1	8	00011011
21	3	1	9	000111000
22	4	1	9	000110100
23	5	1	10	0001101011
24	6	1	10	0000000001
25	7	1	10	0000000100
26	8	1	11	00000000110
27	9	1	11	00000000001
28	10	1	12	000110101011
29	11	1	12	000000001110

© ISO/IEC 2004 – All rights reserved

30	12	1	12	000000010111
31	13	1	13	0000000011111
32	14	1	13	0000000101100
33	15	1	14	00000000000001
34	16	1	14	00000000000010
35	17	1	15	000000000000001
36	All	Zero	16	0000000000000001
37	Escape	Code	12	000110101010

The above VLC tables are implemented as the following arrays in C. The small array table_symbol[4][3] indicates the number of entries in the VLC tables for EOP=0, EOP=1, and ALL_ZERO. For example, the first row {36, 40, 0} indicates that the VLC table for the MSB block-bp has 36 RUN entries for EOP=0, 40 RUN entries for EOP=1, and no ALL_ZERO entry before the Escape Code. The second row {28, 36, 1} means that the VLC table for the MSB-1 block-bp has 28 RUN entries for EOP=0, 36 RUN entries for EOP=1, and 1 ALL_ZERO entry before the Escape Code. The large array table_bpc[4][154] contains the 4 VLC tables. The array contains all the VLC Codes and Code Lengths in the format of the decimal value of a VLC Code and its Code Length as a pair. For example, the first pair of numbers in table_bpc[4][154] is (1,2) that corresponds to the first VLC Code in the VLC Table for the MSB Block-bp, 01, with a Code Length 2. The second pair (5,3) corresponds to the second VLC Code, 101, in the same table with a Code Length 3.

```
int table_symbol[4][3] = {

{ 36,  40,  0},

{ 28,  36,  1},

{ 24,  27,  1},

{ 17,  18,  1},

};

int table_bpc[4][154] = {

{

    1, 2, 5, 3, 13, 4, 1, 4, 30, 5, 19, 5, 51, 6, 58, 6, 34, 6, 37, 6, 115, 7, 1, 7,  237, 8, 143, 8, 389, 9, 476, 9, 1, 9, 17, 9, 16, 9, 770,10, 384, 9, 570,10, 1563,11, 1910,11, 1138,11, 3788,12, 1,12, 2284,12, 4558,13, 4559,13, 1,14, 7320,13, 7322,13, 7323,13, 7321,13, 1,13, 1, 3, 31, 5, 16, 5, 56, 6, 3, 6, 1, 6, 98, 7, 100, 7, 101, 7, 36, 6, 99, 7, 70, 7, 5, 7, 193, 8, 229, 8, 1, 8, 19, 9, 391, 9, 456, 9, 388, 9, 239, 8, 472, 9, 18, 9, 780,10, 771,10, 954,10, 1562,11, 1828,11, 2,11, 3,11, 1143,11, 1895,11, 1831,11, 1911,11, 1829,11, 568,10, 1,11, 3789,12, 2285,12, 2278,12, 946,10},

{

    1, 2, 2, 2, 7, 3, 2, 4, 12, 4, 2, 5, 26, 5, 14, 6, 1, 6, 24, 7, 15, 7, 108, 7, 29, 8, 3, 8, 48, 9, 207,10, 3,10, 9,10, 409,11, 411,11, 1,11, 816,12, 41,12, 3568,12, 1635,13, 93,13, 7138,13, 1,15, 13, 6, 3, 6, 31, 7, 13, 7, 4, 7, 5, 7, 109, 7, 50, 8, 60, 8, 61, 8, 28, 8, 25, 8, 1, 8, 222, 8, 49, 9, 206,10, 2,10, 1,10, 8,10, 893,10, 895,10, 410,11, 22,11, 1788,11, 21,11, 1785,11, 1,12, 47,12, 1,13, 81,13, 92,13, 80,13, 7139,13, 1,14, 3268,14, 3269,14, 110, 7, 1789,11},

{

    1, 1, 2, 3, 1, 3, 7, 4, 12, 5, 1, 5, 27, 6, 5, 6, 52, 7, 14, 7, 5, 8, 25, 8, 8, 9, 431,10, 1,10, 37,11, 1723,12, 72,12, 3441,13, 3436,13, 3438,13, 966,13, 6880,14, 1,15, 4, 6, 1, 7, 13, 7, 106, 8, 6, 8, 1, 8, 31, 8, 24, 8, 15, 9, 14, 9, 1, 9, 61, 9, 428,10, 19,10, 121,10, 1,11, 1722,12, 1721,12, 73,12, 1,12, 482,12, 3437,13, 3439,13, 1,13, 967,13, 6881,14, 1,14, 858,11, 240,11},
```


© ISO/IEC 2004 – All rights reserved

{

1, 1, 1, 2, 1, 3, 2, 5, 1, 5, 1, 6, 12, 7, 29, 8, 57, 9, 3, 9, 2,10, 212,11, 10,11, 1,12, 30,13, 45,13, 3,14, 15, 7, 1, 7, 27, 8, 56, 9, 52, 9, 107,10, 1,10, 4,10, 6,11, 1,11, 427,12, 14,12, 23,12, 31,13, 44,13, 1,14, 2,14, 1,15, 1,16, 426,12}

};

B.4.2 table_bpc4fw_m

Table B.94 — VLC table for MSB block-bp if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	10
2	1	0	4	0110
3	2	0	5	01110
4	3	0	6	011110
5	4	0	6	010111
6	5	0	6	000011
7	6	0	6	000110
8	7	0	7	0101010
9	8	0	7	0101000
10	9	0	7	0000001
11	10	0	8	01111111
12	11	0	8	01011001
13	12	0	8	00010111
14	13	0	9	011111000
15	14	0	9	010101101
16	15	0	9	000000001
17	16	0	10	0111110011
18	17	0	10	0111110010
19	18	0	10	0101011110
20	19	0	10	0101101101
21	20	0	11	00000000001
22	0	1	2	11
23	1	1	3	001
24	2	1	4	0100
25	3	1	6	000100
26	4	1	7	0101001
27	5	1	7	0000010
28	6	1	7	0001010
29	7	1	7	0001110
30	8	1	7	0001111
31	9	1	8	01111101
32	10	1	8	01111110
33	11	1	8	01011000
34	12	1	8	00000001
35	13	1	8	00000110
36	14	1	8	00010110
37	15	1	9	010101100
38	16	1	9	010110111

© ISO/IEC 2004 – All rights reserved

39	17	1	9	010101110
40	18	1	9	010110101
41	19	1	9	010110100
42	20	1	9	000001110
43	21	1	10	0101011111
44	22	1	10	0000011111
45	23	1	10	0000000001
46	24	1	10	0000011110
47	25	1	10	0101101100
48	Escape	Code	6	000010

Table B.95 — VLC table for MSB-1 block-bp if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	3	001
3	2	0	3	100
4	3	0	4	1101
5	4	0	5	00001
6	5	0	5	11100
7	6	0	6	000110
8	7	0	6	000001
9	8	0	6	110001
10	9	0	7	0001111
11	10	0	7	1110111
12	11	0	7	1010010
13	12	0	8	11101001
14	13	0	8	11000010
15	14	0	9	000111000
16	15	0	9	110000111
17	16	0	9	101010100
18	17	0	10	0000000011
19	18	0	10	1110110111
20	19	0	10	1100001101
21	20	0	10	1010111010
22	21	0	11	00000000001
23	22	0	11	11000011001
24	23	0	11	10101010111
25	24	0	12	000111001010
26	0	1	4	1111
27	1	1	5	00010
28	2	1	5	11001
29	3	1	6	101000
30	4	1	7	0000001
31	5	1	7	1110101
32	6	1	7	1010100
33	7	1	7	1010110
34	8	1	7	1010011
35	9	1	8	00011101

© ISO/IEC 2004 – All rights reserved

36	10	1	8	00000001
37	11	1	8	11101000
38	12	1	8	11101100
39	13	1	8	11000000
40	14	1	8	10101111
41	15	1	9	111011010
42	16	1	9	101010111
43	17	1	9	101010110
44	18	1	9	101011100
45	19	1	10	0000000010
46	20	1	10	0000000001
47	21	1	10	1110110110
48	22	1	10	1010101010
49	23	1	10	1010111011
50	24	1	11	00011100100
51	25	1	11	00011100110
52	26	1	11	11000011000
53	27	1	11	10101010110
54	28	1	12	000111001011
55	29	1	12	000111001111
56	30	1	12	000111001110
57	31	1	12	000000000001
58	32	1	13	0000000000001
59	All	Zero	4	1011
60	Escape	Code	8	11000001

Table B.96 — VLC table for MSB-2 block-bp if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	2	10
3	2	0	3	110
4	3	0	4	0001
5	4	0	4	1110
6	5	0	5	00101
7	6	0	5	11110
8	7	0	6	000001
9	8	0	6	111111
10	9	0	7	0000001
11	10	0	7	0011010
12	11	0	8	00001100
13	12	0	8	00110010
14	13	0	9	000010010
15	14	0	9	001101101
16	15	0	10	0000100111
17	16	0	10	0000000001
18	17	0	11	00001011111
19	18	0	11	00000000100
20	19	0	11	00000000001

© ISO/IEC 2004 – All rights reserved

21	20	0	11	00110110000
22	21	0	12	000011111011
23	22	0	12	001001010010
24	23	0	13	0000101111010
25	24	0	13	0000100110010
26	25	0	14	00000000000001
27	0	1	5	00111
28	1	1	6	001000
29	2	1	6	111110
30	3	1	7	0010011
31	4	1	7	0011000
32	5	1	8	00001010
33	6	1	8	00001110
34	7	1	8	00001101
35	8	1	8	00000001
36	9	1	8	00100100
37	10	1	8	00110011
38	11	1	8	00110111
39	12	1	9	000010110
40	13	1	9	000011110
41	14	1	9	001001011
42	15	1	10	0000111111
43	16	1	10	0000000011
44	17	1	10	0010010101
45	18	1	10	0011011001
46	19	1	11	00001001101
47	20	1	11	00001111100
48	21	1	11	00000000101
49	22	1	11	00100101000
50	23	1	11	00110110001
51	24	1	12	000010111100
52	25	1	12	000010011000
53	26	1	12	000000000001
54	27	1	12	001001010011
55	28	1	13	0000101111011
56	29	1	13	0000100110011
57	30	1	13	0000111110101
58	31	1	13	0000111110100
59	32	1	13	0000000000001
60	All	Zero	8	00001000
61	Escape	Code	10	0000101110

© ISO/IEC 2004 – All rights reserved

Table B.97 — VLC table for MSB-3 block-bp and other block-bps if fw_matrix[0] = 2 or 3

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	1	1
2	1	0	3	001
3	2	0	3	010
4	3	0	4	0001
5	4	0	5	00001
6	5	0	5	01111
7	6	0	6	000001
8	7	0	6	011100
9	8	0	7	0110100
10	9	0	7	0111010
11	10	0	8	01101111
12	11	0	8	01110111
13	12	0	9	011011101
14	13	0	10	0000001001
15	14	0	10	0110110010
16	15	0	11	00000000001
17	16	0	11	01100110110
18	17	0	12	000000100001
19	18	0	12	011011001100
20	19	0	12	011001110010
21	20	0	13	0000001000100
22	21	0	13	0110110011110
23	22	0	14	00000010001010
24	23	0	14	00000000000001
25	0	1	6	011000
26	1	1	7	0110101
27	2	1	7	0110010
28	3	1	8	00000011
29	4	1	8	00000001
30	5	1	8	01101101
31	6	1	8	01110110
32	7	1	9	000000101
33	8	1	9	011011100
34	9	1	9	011011000
35	10	1	9	011001100
36	11	1	9	011001111
37	12	1	10	0000000001
38	13	1	10	0000000011
39	14	1	10	0110011010
40	15	1	11	00000000101
41	16	1	11	01100110111
42	17	1	11	01100111000
43	18	1	12	000000100011
44	19	1	12	000000000001
45	20	1	12	011011001110
46	21	1	12	011011001101
47	22	1	12	011001110011

© ISO/IEC 2004 – All rights reserved

48	23	1	13	0000001000001
49	24	1	13	0000000000001
50	25	1	13	0110110011111
51	26	1	14	00000010001011
52	27	1	14	00000010000001
53	28	1	14	00000010000000
54	All	Zero	11	00000000100
55	Escape	Code	10	0110011101

int table_symbol4fw_m[4][3] = {

{ 21, 26, 0},

{ 25, 33, 1},

{ 26, 33, 1},

{ 24, 29, 1},

};

int table_bpc4fw_m[4][122] = {

{

2, 2, 6, 4, 14, 5, 30, 6, 23, 6, 3, 6, 6, 6, 42, 7, 40, 7, 1, 7, 127, 8, 89, 8, 23, 8, 248, 9, 173, 9, 1, 9, 499,10, 498,10, 350,10, 365,10, 1,11, 3, 2, 1, 3, 4, 4, 4, 6, 41, 7, 2, 7, 10, 7, 14, 7, 15, 7, 125, 8, 126, 8, 88, 8, 1, 8, 6, 8, 22, 8, 172, 9, 183, 9, 174, 9, 181, 9, 180, 9, 14, 9, 351,10, 31,10, 1,10, 30,10, 364,10, 2, 6},

{

1, 2, 1, 3, 4, 3, 13, 4, 1, 5, 28, 5, 6, 6, 1, 6, 49, 6, 15, 7, 119, 7, 82, 7, 233, 8, 194, 8, 56, 9, 391, 9, 340, 9, 3,10, 951,10, 781,10, 698,10, 1,11, 1561,11, 1367,11, 458,12, 15, 4, 2, 5, 25, 5, 40, 6, 1, 7, 117, 7, 84, 7, 86, 7, 83, 7, 29, 8, 1, 8, 232, 8, 236, 8, 192, 8, 175, 8, 474, 9, 343, 9, 342, 9, 348, 9, 2,10, 1,10, 950,10, 682,10, 699,10, 228,11, 230,11, 1560,11, 1366,11, 459,12, 463,12, 462,12, 1,12, 1,13, 11, 4, 193, 8},

{

1, 2, 2, 2, 6, 3, 1, 4, 14, 4, 5, 5, 30, 5, 1, 6, 63, 6, 1, 7, 26, 7, 12, 8, 50, 8, 18, 9, 109, 9, 39,10, 1,10, 95,11, 4,11, 1,11, 432,11, 251,12, 594,12, 378,13, 306,13, 1,14, 7, 5, 8, 6, 62, 6, 19, 7, 24, 7, 10, 8, 14, 8, 13, 8, 1, 8, 36, 8, 51, 8, 55, 8, 22, 9, 30, 9, 75, 9, 63,10, 3,10, 149,10, 217,10, 77,11, 124,11, 5,11, 296,11, 433,11, 188,12, 152,12, 1,12, 595,12, 379,13, 307,13, 501,13, 500,13, 1,13, 8, 8, 46,10},

{

1, 1, 1, 3, 2, 3, 1, 4, 1, 5, 15, 5, 1, 6, 28, 6, 52, 7, 58, 7, 111, 8, 119, 8, 221, 9, 9,10, 434,10, 1,11, 822,11, 33,12, 1740,12, 1650,12, 68,13, 3486,13, 138,14, 1,14, 24, 6, 53, 7, 50, 7, 3, 8, 1, 8, 109, 8, 118, 8, 5, 9, 220, 9, 216, 9, 204, 9, 207, 9, 1,10, 3,10, 410,10, 5,11, 823,11, 824,11, 35,12, 1,12, 1742,12, 1741,12, 1651,12, 65,13, 1,13, 3487,13, 139,14, 129,14, 128,14, 4,11, 413,10}

};

© ISO/IEC 2004 – All rights reserved

B.4.3 table_bpc4fw_h

Table B.98 — VLC table for MSB block-bp if fw_matrix[0] = 4, 5, 6, or 7

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	5	00001
3	2	0	7	0000011
4	3	0	9	000000001
5	0	1	1	1
6	1	1	3	001
7	2	1	4	0001
8	3	1	7	0000010
9	4	1	8	00000011
10	5	1	8	00000001
11	Escape	Code	8	00000010

Table B.99 — VLC table for MSB-1 block-bp if fw_matrix[0] = 4, 5, 6, or 7

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	10
2	1	0	3	011
3	2	0	4	0001
4	3	0	6	110001
5	4	0	6	000010
6	5	0	8	00000010
7	6	0	9	000001000
8	7	0	9	000000011
9	8	0	9	000000010
10	9	0	10	0000000010
11	10	0	11	00000000001
12	0	1	3	111
13	1	1	3	001
14	2	1	4	1101
15	3	1	5	11001
16	4	1	6	110000
17	5	1	6	000011
18	6	1	8	00000101
19	7	1	9	000001001
20	8	1	9	000000111
21	9	1	9	000000110
22	10	1	10	0000000011
23	11	1	10	0000000001
24	All	Zero	3	010
25	Escape	Code	7	0000011

© ISO/IEC 2004 – All rights reserved

Table B.100 — VLC table for MSB-2 block-bp if fw_matrix[0] = 4, 5, 6, or 7

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	10
2	1	0	2	01
3	2	0	4	1100
4	3	0	5	11010
5	4	0	5	00001
6	5	0	6	000001
7	6	0	7	1111101
8	7	0	8	11110011
9	8	0	8	00011110
10	9	0	8	00000001
11	10	0	9	111110000
12	11	0	9	000000100
13	12	0	10	1111000000
14	13	0	10	1111100101
15	14	0	10	0000001010
16	15	0	11	11110000110
17	16	0	11	11111001000
18	0	1	3	001
19	1	1	4	1110
20	2	1	5	11011
21	3	1	5	00010
22	4	1	6	111111
23	5	1	6	000110
24	6	1	7	0001110
25	7	1	8	11110001
26	8	1	8	00011111
27	9	1	8	00000011
28	10	1	9	111110001
29	11	1	9	111110011
30	12	1	9	000000001
31	13	1	10	1111000001
32	14	1	10	1111000010
33	15	1	10	0000000001
34	16	1	11	11110000111
35	17	1	11	11111001001
36	18	1	11	00000010111
37	19	1	11	00000010110
38	20	1	11	00000000001
39	21	1	12	000000000001
40	All	Zero	6	111101
41	Escape	Code	8	11110010

© ISO/IEC 2004 – All rights reserved

Table B.101 — VLC table for MSB-3 block-bp and other block-bps if fw_matrix[0] = 4, 5, 6, or 7

Entry Number	RUN	EOP	Code Length	VLC Code
1	0	0	2	01
2	1	0	2	10
3	2	0	3	110
4	3	0	4	0001
5	4	0	5	00111
6	5	0	6	001100
7	6	0	6	001001
8	7	0	6	111011
9	8	0	7	0000011
10	9	0	8	00110111
11	10	0	8	00000011
12	11	0	8	11101011
13	12	0	9	000000001
14	13	0	9	111010010
15	14	0	10	0000000001
16	15	0	10	1110100010
17	16	0	11	00000000001
18	17	0	11	11101001111
19	18	0	12	000000011011
20	19	0	12	000000000001
21	20	0	12	000011000011
22	21	0	12	111010001101
23	22	0	13	0000000000001
24	0	1	4	1111
25	1	1	5	00101
26	2	1	5	11100
27	3	1	6	001000
28	4	1	6	000010
29	5	1	7	0011010
30	6	1	7	0000111
31	7	1	8	00110110
32	8	1	8	00000010
33	9	1	8	00001101
34	10	1	8	11101010
35	11	1	9	000000010
36	12	1	9	000001011
37	13	1	9	000011001
38	14	1	9	111010000
39	15	1	10	1110100110
40	16	1	11	00000001100
41	17	1	11	00000001111
42	18	1	11	00001100000
43	19	1	11	00001100010
44	20	1	11	11101001110
45	21	1	11	00001100011
46	22	1	11	11101000111
47	23	1	12	000000011101

© ISO/IEC 2004 – All rights reserved

48	24	1	12	000000011010
49	25	1	12	000000011100
50	26	1	12	000011000010
51	27	1	12	111010001100
52	28	1	14	00000000000001
53	All	Zero	8	00000100
54	Escape	Code	9	000001010

```
int table_symbol4fw_h[4][3] = {

{ 4,  6,  0},

{ 11,  12,  1},

{ 17,  22,  1},

{ 23,  29,  1},

};

int table_bpc4fw_h[4][108] = {

{

    1, 2, 1, 5, 3, 7, 1, 9, 1, 1, 1, 3, 1, 4, 2, 7, 3, 8, 1, 8, 2, 8},

{

    2, 2, 3, 3, 1, 4, 49, 6, 2, 6, 2, 8, 8, 9, 3, 9, 2, 9, 2,10, 1,11, 7, 3, 1, 3, 13, 4, 25, 5, 48, 6, 3, 6, 5, 8, 9, 9, 7, 9, 6,
9, 3,10, 1,10, 2, 3, 3, 7},

{

    2, 2, 1, 2, 12, 4, 26, 5, 1, 5, 1, 6, 125, 7, 243, 8, 30, 8, 1, 8, 496, 9, 4, 9, 960,10, 997,10, 10,10, 1926,11,
1992,11, 1, 3, 14, 4, 27, 5, 2, 5, 63, 6, 6, 6, 14, 7, 241, 8, 31, 8, 3, 8, 497, 9, 499, 9, 1, 9, 961,10, 962,10, 1,10,
1927,11, 1993,11, 23,11, 22,11, 1,11, 1,12, 61, 6, 242, 8},

{

    1, 2, 2, 2, 6, 3, 1, 4, 7, 5, 12, 6, 9, 6, 59, 6, 3, 7, 55, 8, 3, 8, 235, 8, 1, 9, 466, 9, 1,10, 930,10, 1,11, 1871,11,
27,12, 1,12, 195,12, 3725,12, 1,13, 15, 4, 5, 5, 28, 5, 8, 6, 2, 6, 26, 7, 7, 7, 54, 8, 2, 8, 13, 8, 234, 8, 2, 9, 11, 9, 25,
9, 464, 9, 934,10, 12,11, 15,11, 96,11, 98,11, 1870,11, 99,11, 1863,11, 29,12, 26,12, 28,12, 194,12, 3724,12, 1,14,
4, 8, 10, 9}

};
```


© ISO/IEC 2004 – All rights reserved

Annex C
(normative)

Face and body object decoding tables and definitions

FAPs names may contain letters with the following meaning: l = left, r = right, t = top, b = bottom, i = inner, o = outer, m = middle. The sum of two corresponding top and bottom eyelid FAPs must equal 1024 when the eyelids are closed. Inner lips are closed when the sum of two corresponding top and bottom lip FAPs equals zero. For example: (lower_t_midlip + raise_b_midlip) = 0 when the lips are closed. All directions are defined with respect to the face and not the image of the face.

Table C.1 — FAP definitions, group assignments and step sizes

#	FAP name	FAP description	units	Uni-orBi dir	Pos motion	Grp	FDP sub-grp num	Quan-tisa-tion step size	Min/Max I-Frame quantised values	Min/Max P-Frame quantised values
1	viseme	Set of values determining the mixture of two visemes for this frame (e.g. pbm, fv, th)	na	na	na	1	na	1	viseme_ blend: +63	viseme_ blend: +-63
2	expression	A set of values determining the mixture of two facial expression	na	na	na	1	na	1	expression _intensity1, expression _intensity2: +63	expression _intensity1, expression _intensity2: +-63
3	open_jaw	Vertical jaw displacement (does not affect mouth opening)	MNS	U	down	2	1	4	+1080	+-360
4	lower_t_midlip	Vertical top middle inner lip displacement	MNS	B	down	2	2	2	+-600	+-180
5	raise_b_midlip	Vertical bottom middle inner lip displacement	MNS	B	up	2	3	2	+-1860	+-600
6	stretch_l_cornerlip	Horizontal displacement of left inner lip corner	MW	B	left	2	4	2	+-600	+-180
7	stretch_r_cornerlip	Horizontal displacement of right inner lip corner	MW	B	right	2	5	2	+-600	+-180
8	lower_t_lip_lm	Vertical displacement of midpoint between left corner and middle of top inner lip	MNS	B	down	2	6	2	+-600	+-180
9	lower_t_lip_rm	Vertical displacement of midpoint between	MNS	B	down	2	7	2	+-600	+-180

© ISO/IEC 2004 – All rights reserved

		right corner and middle of top inner lip								
10	raise_b_lip_lm	Vertical displacement of midpoint between left corner and middle of bottom inner lip	MNS	B	up	2	8	2	+-1860	+-600
11	raise_b_lip_rm	Vertical displacement of midpoint between right corner and middle of bottom inner lip	MNS	B	up	2	9	2	+-1860	+-600
12	raise_l_cornerlip	Vertical displacement of left inner lip corner	MNS	B	up	2	4	2	+-600	+-180
13	raise_r_cornerlip	Vertical displacement of right inner lip corner	MNS	B	up	2	5	2	+-600	+-180
14	thrust_jaw	Depth displacement of jaw	MNS	U	forward	2	1	1	+600	+-180
15	shift_jaw	Side to side displacement of jaw	MW	B	right	2	1	1	+-1080	+-360
16	push_b_lip	Depth displacement of bottom middle lip	MNS	B	forward	2	3	1	+-1080	+-360
17	push_t_lip	Depth displacement of top middle lip	MNS	B	forward	2	2	1	+-1080	+-360
18	depress_chin	Upward and compressing movement of the chin (like in sadness)	MNS	B	up	2	10	1	+-420	+-180
19	close_t_l_eyelid	Vertical displacement of top left eyelid	IRISD	B	down	3	1	1	+-1080	+-600
20	close_t_r_eyelid	Vertical displacement of top right eyelid	IRISD	B	down	3	2	1	+-1080	+-600
21	close_b_l_eyelid	Vertical displacement of bottom left eyelid	IRISD	B	up	3	3	1	+-600	+-240
22	close_b_r_eyelid	Vertical displacement of bottom right eyelid	IRISD	B	up	3	4	1	+-600	+-240
23	yaw_l_eyeball	Horizontal orientation of left eyeball	AU	B	left	3	na	128	+-1200	+-420

© ISO/IEC 2004 – All rights reserved

24	yaw_r_eyeball	Horizontal orientation of right eyeball	AU	B	left	3	na	128	+-1200	+-420
25	pitch_l_eyeball	Vertical orientation of left eyeball	AU	B	down	3	na	128	+-900	+-300
26	pitch_r_eyeball	Vertical orientation of right eyeball	AU	B	down	3	na	128	+-900	+-300
27	thrust_l_eyeball	Depth displacement of left eyeball	ES	B	forward	3	na	1	+-600	+-180
28	thrust_r_eyeball	Depth displacement of right eyeball	ES	B	forward	3	na	1	+-600	+-180
29	dilate_l_pupil	Dilation of left pupil	IRISD	B	growing	3	5	1	+-420	+-120
30	dilate_r_pupil	Dilation of right pupil	IRISD	B	growing	3	6	1	+-420	+-120
31	raise_l_i_eyebrow	Vertical displacement of left inner eyebrow	ENS	B	up	4	1	2	+-900	+-360
32	raise_r_i_eyebrow	Vertical displacement of right inner eyebrow	ENS	B	up	4	2	2	+-900	+-360
33	raise_l_m_eyebrow	Vertical displacement of left middle eyebrow	ENS	B	up	4	3	2	+-900	+-360
34	raise_r_m_eyebrow	Vertical displacement of right middle eyebrow	ENS	B	up	4	4	2	+-900	+-360
35	raise_l_o_eyebrow	Vertical displacement of left outer eyebrow	ENS	B	up	4	5	2	+-900	+-360
36	raise_r_o_eyebrow	Vertical displacement of right outer eyebrow	ENS	B	up	4	6	2	+-900	+-360
37	squeeze_l_eyebrow	Horizontal displacement of left eyebrow	ES	B	right	4	1	1	+-900	+-300
38	squeeze_r_eyebrow	Horizontal displacement of right eyebrow	ES	B	left	4	2	1	+-900	+-300
39	puff_l_cheek	Horizontal displacement of left cheeck	ES	B	left	5	1	2	+-900	+-300
40	puff_r_cheek	Horizontal displacement of right cheeck	ES	B	right	5	2	2	+-900	+-300
41	lift_l_cheek	Vertical displacement of left cheek	ENS	U	up	5	3	2	+-600	+-180
42	lift_r_cheek	Vertical displacement of right cheek	ENS	U	up	5	4	2	+-600	+-180

© ISO/IEC 2004 – All rights reserved

43	shift_tongue_tip	Horizontal displacement of tongue tip	MW	B	right	6	1	1	+-1080	+-420
44	raise_tongue_tip	Vertical displacement of tongue tip	MNS	B	up	6	1	1	+-1080	+-420
45	thrust_tongue_tip	Depth displacement of tongue tip	MW	B	forward	6	1	1	+-1080	+-420
46	raise_tongue	Vertical displacement of tongue	MNS	B	up	6	2	1	+-1080	+-420
47	tongue_roll	Rolling of the tongue into U shape	AU	U	concave upward	6	3, 4	512	+300	+-60
48	head_pitch	Head pitch angle from top of spine	AU	B	down	7	na	170	+-1860	+-600
49	head_yaw	Head yaw angle from top of spine	AU	B	left	7	na	170	+-1860	+-600
50	head_roll	Head roll angle from top of spine	AU	B	right	7	na	170	+-1860	+-600
51	lower_t_midlip_o	Vertical top middle outer lip displacement	MNS	B	down	8	1	2	+-600	+-180
52	raise_b_midlip_o	Vertical bottom middle outer lip displacement	MNS	B	up	8	2	2	+-1860	+-600
53	stretch_l_cornerlip_o	Horizontal displacement of left outer lip corner	MW	B	left	8	3	2	+-600	+-180
54	stretch_r_cornerlip_o	Horizontal displacement of right outer lip corner	MW	B	right	8	4	2	+-600	+-180
55	lower_t_lip_lm_o	Vertical displacement of midpoint between left corner and middle of top outer lip	MNS	B	down	8	5	2	+-600	+-180
56	lower_t_lip_rm_o	Vertical displacement of midpoint between right corner and middle of top outer lip	MNS	B	down	8	6	2	+-600	+-180
57	raise_b_lip_lm_o	Vertical displacement of midpoint between left corner and middle of bottom outer lip	MNS	B	up	8	7	2	+-1860	+-600

© ISO/IEC 2004 – All rights reserved

58	raise_b_lip_rm_o	Vertical displacement of midpoint between right corner and middle of bottom outer lip	MNS	B	up	8	8	2	+-1860	+-600
59	raise_l_cornerlip_o	Vertical displacement of left outer lip corner	MNS	B	up	8	3	2	+-600	+-180
60	raise_r_cornerli _o	Vertical displacement of right outer lip corner	MNS	B	up	8	4	2	+-600	+-180
61	stretch_l_nose	Horizontal displacement of left side of nose	ENS	B	left	9	1	1	+-540	+-120
62	stretch_r_nose	Horizontal displacement of right side of nose	ENS	B	right	9	2	1	+-540	+-120
63	raise_nose	Vertical displacement of nose tip	ENS	B	up	9	3	1	+-680	+-180
64	bend_nose	Horizontal displacement of nose tip	ENS	B	right	9	3	1	+-900	+-180
65	raise_l_ear	Vertical displacement of left ear	ENS	B	up	10	1	1	+-900	+-240
66	raise_r_ear	Vertical displacement of right ear	ENS	B	up	10	2	1	+-900	+-240
67	pull_l_ear	Horizontal displacement of left ear	ENS	B	left	10	3	1	+-900	+-300
68	pull_r_ear	Horizontal displacement of right ear	ENS	B	right	10	4	1	+-900	+-300

Table C.2 — FAP grouping

Group	Number of FAPs
1: visemes and expressions	2
2: jaw, chin, inner lowerlip, cornerlips, midlip	16
3: eyeballs, pupils, eyelids	12
4: eyebrow	8
5: cheeks	4
6: tongue	5
7: head rotation	3
8: outer lip positions	10
9: nose	4
10: ears	4

© ISO/IEC 2004 – All rights reserved

In the following, each facial expression is defined by a textual description and a pictorial example. This reference was also used for the characteristics of the described expressions.

Table C.3 — Values for expression_select

expression_select	expression name	textual description
0	na	na
1	joy	The eyebrows are relaxed. The mouth is open and the mouth corners pulled back toward the ears.
2	sadness	The inner eyebrows are bent upward. The eyes are slightly closed. The mouth is relaxed.
3	anger	The inner eyebrows are pulled downward and together. The eyes are wide open. The lips are pressed against each other or opened to expose the teeth.
4	fear	The eyebrows are raised and pulled together. The inner eyebrows are bent upward. The eyes are tense and alert.
5	disgust	The eyebrows and eyelids are relaxed. The upper lip is raised and curled, often asymmetrically.
6	surprise	The eyebrows are raised. The upper eyelids are wide open, the lower relaxed. The jaw is opened.

© ISO/IEC 2004 – All rights reserved

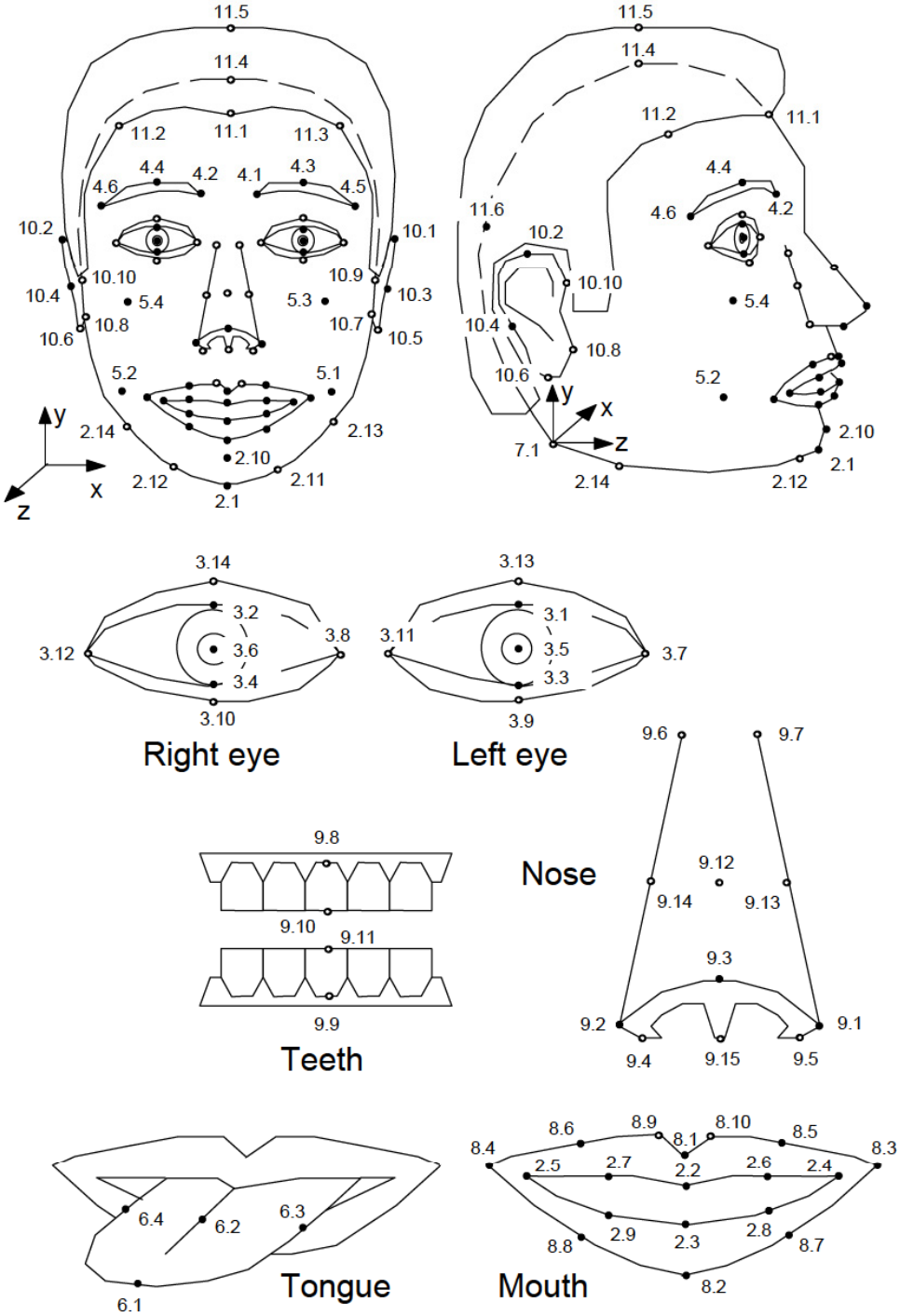


Figure C.1 — FDP feature point set

© ISO/IEC 2004 – All rights reserved

In the following, the notation 2.1.x indicates the x coordinate of feature point 2.1.

Feature points		Recommended location constraints		
#	Text description	x	y	z
2.1	Bottom of the chin	7.1.x		
2.2	Middle point of inner upper lip contour	7.1.x		
2.3	Middle point of inner lower lip contour	7.1.x		
2.4	Left corner of inner lip contour			
2.5	Right corner of inner lip contour			
2.6	Midpoint between f.p. 2.2 and 2.4 in the inner upper lip contour	(2.2.x+2.4.x)/2		
2.7	Midpoint between f.p. 2.2 and 2.5 in the inner upper lip contour	(2.2.x+2.5.x)/2		
2.8	Midpoint between f.p. 2.3 and 2.4 in the inner lower lip contour	(2.3.x+2.4.x)/2		
2.9	Midpoint between f.p. 2.3 and 2.5 in the inner lower lip contour	(2.3.x+2.5.x)/2		
2.10	Chin boss	7.1.x		
2.11	Chin left corner	> 8.7.x and < 8.3.x		
2.12	Chin right corner	> 8.4.x and < 8.8.x		
2.13	Left corner of jaw bone			
2.14	Right corner of jaw bone			
3.1	Center of upper inner left eyelid	(3.7.x+3.11.x)/2		
3.2	Center of upper inner right eyelid	(3.8.x+3.12.x)/2		
3.3	Center of lower inner left eyelid	(3.7.x+3.11.x)/2		
3.4	Center of lower inner right eyelid	(3.8.x+3.12.x)/2		
3.5	Center of the pupil of left eye			
3.6	Center of the pupil of right eye			
3.7	Left corner of left eye			
3.8	Left corner of right eye			
3.9	Center of lower outer left eyelid	(3.7.x+3.11.x)/2		
3.10	Center of lower outer right eyelid	(3.7.x+3.11.x)/2		
3.11	Right corner of left eye			
3.12	Right corner of right eye			
3.13	Center of upper outer left eyelid	(3.8.x+3.12.x)/2		
3.14	Center of upper outer right eyelid	(3.8.x+3.12.x)/2		
4.1	Right corner of left eyebrow			
4.2	Left corner of right eyebrow			
4.3	Uppermost point of the left eyebrow	(4.1.x+4.5.x)/2 or x coord of the uppermost point of the contour		
4.4	Uppermost point of the right eyebrow	(4.2.x+4.6.x)/2 or x coord of the uppermost point of the contour		
4.5	Left corner of left eyebrow			
4.6	Right corner of right eyebrow			

© ISO/IEC 2004 – All rights reserved

5.1	Center of the left cheek		8.3.y	
5.2	Center of the right cheek		8.4.y	
5.3	Left cheek bone	> 3.5.x and < 3.7.x	> 9.15.y and < 9.12.y	
5.4	Right cheek bone	> 3.6.x and < 3.12.x	> 9.15.y and < 9.12.y	
6.1	Tip of the tongue	7.1.x		
6.2	Center of the tongue body	7.1.x		
6.3	Left border of the tongue			6.2.z
6.4	Right border of the tongue			6.2.z
7.1	top of spine (center of head rotation)			
8.1	Middle point of outer upper lip contour	7.1.x		
8.2	Middle point of outer lower lip contour	7.1.x		
8.3	Left corner of outer lip contour			
8.4	Right corner of outer lip contour			
8.5	Midpoint between f.p. 8.3 and 8.1 in outer upper lip contour	(8.3.x+8.1.x)/2		
8.6	Midpoint between f.p. 8.4 and 8.1 in outer upper lip contour	(8.4.x+8.1.x)/2		
8.7	Midpoint between f.p. 8.3 and 8.2 in outer lower lip contour	(8.3.x+8.2.x)/2		
8.8	Midpoint between f.p. 8.4 and 8.2 in outer lower lip contour	(8.4.x+8.2.x)/2		
8.9	Right hiph point of Cupid's bow			
8.10	Left hiph point of Cupid's bow			
9.1	Left nostril border			
9.2	Right nostril border			
9.3	Nose tip	7.1.x		
9.4	Bottom right edge of nose			
9.5	Bottom left edge of nose			
9.6	Right upper edge of nose bone			
9.7	Left upper edge of nose bone			
9.8	Top of the upper teeth	7.1.x		
9.9	Bottom of the lower teeth	7.1.x		
9.10	Bottom of the upper teeth	7.1.x		
9.11	Top of the lower teeth	7.1.x		
9.12	Middle lower edge of nose bone (or nose bump)	7.1.x	(9.6.y + 9.3.y)/2 or nose bump	
9.13	Left lower edge of nose bone		(9.6.y +9.3.y)/2	
9.14	Right lower edge of nose bone		(9.6.y +9.3.y)/2	
9.15	Bottom middle edge of nose	7.1.x		
10.1	Top of left ear			
10.2	Top of right ear			
10.3	Back of left ear		(10.1.y+10.5.y)/2	
10.4	Back of right ear		(10.2.y+10.6.y)/2	
10.5	Bottom of left ear lobe			
10.6	Bottom of right ear lobe			

© ISO/IEC 2004 – All rights reserved

10.7	Lower contact point between left lobe and face			
10.8	Lower contact point between right lobe and face			
10.9	Upper contact point between left ear and face			
10.10	Upper contact point between right ear and face			
11.1	Middle border between hair and forehead	7.1.x		
11.2	Right border between hair and forehead	< 4.4.x		
11.3	Left border between hair and forehead	> 4.3.x		
11.4	Top of skull	7.1.x		> 10.4.z and < 10.2.z
11.5	Hair thickness over f.p. 11.4	11.4.x		11.4.z
11.6	Back of skull	7.1.x	3.5.y	

Table C.4 — FDP fields

FDP field	Description
featurePointsCoord	contains a **Coordinate** node. Specifies feature points for the calibration of the proprietary face. The coordinates are listed in the 'point' field in the **Coordinate** node in the prescribed order, that a feature point with a lower label is listed before a feature point with a higher label (e.g. feature point 3.14 before feature point 4.1).
textureCoords	contains a **Coordinate** node. Specifies texture coordinates for the feature points. The coordinates are listed in the **point** field in the **Coordinate** node in the prescribed order, that a feature point with a lower label is listed before a feature point with a higher label (e.g. feature point 3.14 before feature point 4.1).
textureType	contains a hint to the decoder on the type of texture image, in order to allow better interpolation of texture coordinates for the vertices that are not feature points. If **textureType** is 0, the decoder should assume that the texture image is obtained by cylindrical projection of the face. If **textureType** is 1, the decoder should assume that the texture image is obtained by orthographic projection of the face.
faceDefTables	contains **faceDefTables** nodes. The behavior of FAPs is defined in this field for the face in **faceSceneGraph**.
faceSceneGraph	contains a **Group** node. In case of option 1, this can be used to contain a texture image as explained above. In case of option 2, this is the grouping node for face model rendered in the compositor and has to contain the face model. In this case, the effect of Facial Animation Parameters is defined in the **faceDefTablesfield**.

© ISO/IEC 2004 – All rights reserved

Table C.5 — Values for viseme_select

viseme_select	phonemes	example
0	none	na
1	p, b, m	put, bed, mill
2	f, v	far, voice
3	T,D	think, that
4	t, d	tip, doll
5	k, g	call, gas
6	tS, dZ, S	chair, join, she
7	s, z	sir, zeal
8	n, l	lot, not
9	r	red
10	A:	car
11	e	bed
12	I	tip
13	Q	top
14	U	book

The symbolic constants and variables used in the syntax diagrams are listed in the table below. Every group and BAP is assumed to have a unique unsigned integer value assignment.

TABLE:

Name	Definition	Description
NBAP_GROUP	Array[1..BAP_NUM_GROUPS]	Number of baps in each group.
BAPS_IN_GROUP	Array[1..BAP_NUM_GROUPS] [1..MAX_BAPS]	List of BAPs belonging to each group

Name	Value	Definition
MAX_BAPS	22	Maximum number of BAPs that belong to any group
BAP_NUM_GROUPS	24	Total number of BAP groups
NUM_BAPS	296	Total number of BAPs

The detailed BAPs and the BAP groups are given below:

BAP ID	BAP NAME	DESCRIPTION	Quanti-sation step size	Min/Max I-Frame quantised values	Min/Max P-Frame quantised values
1	sacroiliac_tilt	Forward-backward motion of the pelvis in the sagittal plane	64	-960/ +960	-600/ +600
2	sacroiliac_torsion	Rotation of the pelvis along the body vertical axis (defined by skeleton root)	64	-960/ +960	-600/ +600

© ISO/IEC 2004 – All rights reserved

3	sacroiliac_roll	Side to side swinging of the pelvis in the coronal plane	64	-960/ +960	-600/ +600
4	l_hip_flexion	Forward-backward rotation in the sagittal plane	128	-1260/ +1260	-600/ +600
5	r_hip_flexion	Forward-backward rotation in the sagittal plane	128	-1260/ +1260	-600/ +600
6	l_hip_abduct	Sideward opening in the coronal plane	128	-960/ +480	-600/ +600
7	r_hip_abduct	Sideward opening in the coronal plane	128	-960/ +480	-600/ +600
8	l_hip_twisting	Rotation along the thigh axis	256	-960/ +960	-360/ +360
9	r_hip_twisting	Rotation along the thigh axis	256	-960/ +960	-360/ +360
10	l_knee_flexion	Flexion-extension of the leg in the sagittal plane	128	-1500/ +180	-600/ +600
11	r_knee_flexion	Flexion-extension of the leg in the sagittal plane	128	-1500/ +180	-600/ +600
12	l_knee_twisting	Rotation along the shank axis.	256	-960/ +960	-360/ +360
13	r_knee_twisting	Rotation along the shank axis.	256	-960/ +960	-360/ +360
14	l_ankle_flexion	Flexion-extension of the foot in the sagittal plane	128	-780/ +780	-600/ +600
15	r_ankle_flexion	Flexion-extension of the foot in the sagittal plane	128	-780/ +780	-600/ +600
16	l_ankle_twisting	Rotation along the knee axis	256	-960/ +960	-360/ +360
17	r_ankle_twisting	Rotation along the knee axis	256	-960/ +960	-360/ +360
18	l_subtalar_flexion	Sideward orientation of the foot	256	-780/ +780	-600/ +600
19	r_subtalar_flexion	Sideward orientation of the foot	256	-780/ +780	-600/ +600
20	l_midtarsal_twisting	Internal twisting of the foot (also called navicular joint in anatomy)	256	-180/ +180	-120/ +120
21	r_midtarsal_ twisting	Internal twisting of the foot (also called navicular joint in anatomy)	256	-180/ +180	-120/ +120
22	l_metatarsal_flexion	Up and down rotation of the toe in the sagittal plane	256	-780/ +780	-600/ +600
23	r_metatarsal_flexion	Up and down rotation of the toe in the sagittal plane	256	-780/ +780	-600/ +600
24	l_sternoclavicular_abduct	Up and down motion in the coronal plane	128	-60/ +240	-120/ +120
25	r_sternoclavicular_abduct	Up and down motion in the coronal plane	128	-240/ +60	-120/ +120
26	l_sternoclavicular_rotate	Rotation in the transverse plane	128	-120/ +120	-60/ +60

© ISO/IEC 2004 – All rights reserved

27	r_sternoclavicular_rotate	Rotation in the transverse plane	128	-120/ +120	-60/ +60
28	l_acromioclavicular_abduct	Up and down motion in the coronal plane	128	-60/ +360	-120/ +120
29	r_acromioclavicular_abduct	Up and down motion in the coronal plane	128	-360/ +60	-120/ +120
30	l_acromioclavicular_rotate	Rotation in the transverse plane	128	-360/ +360	-120/ +120
31	r_acromioclavicular_rotate	Rotation in the transverse plane	128	-360/ +360	-120/ +120
32	l_shoulder_flexion	Forward-backward motion in the sagittal plane	64	-1080/ +1860	-600/ +600
33	r_shoulder_flexion	Forward-backward motion in the sagittal plane	64	-1080/ +1860	-600/ +600
34	l_shoulder_abduct	Sideward motion in the coronal plane	64	-240/ +1860	-600/ +600
35	r_shoulder_abduct	Sideward motion in the coronal plane	64	-1860/ +240	-600/ +600
36	l_shoulder_twisting	Rotation along the scapular axis	256	-960/ +960	-360/ +360
37	r_shoulder_twisting	Rotation along the scapular axis	256	-960/ +960	-360/ +360
38	l_elbow_flexion	Flexion-extension of the arm in the sagittal plane	64	-60/ +1560	-600/ +600
39	r_elbow_flexion	Flexion-extension of the arm in the sagittal plane	64	-60/ +1560	-600/ +600
40	l_elbow_twisting	Rotation of the forearm along the upper arm axis.	256	-960/ +960	-360/ +360
41	r_elbow_twisting	Rotation of the forearm along the upper arm axis.	256	-960/ +960	-360/ +360
42	l_wrist_flexion	Rotation of the hand in the coronal plane	128	-960/ +960	-600/ +600
43	r_wrist_flexion	Rotation of the hand in the coronal plane	128	-960/ +960	-600/ +600
44	l_wrist_pivot	Rotation of the hand in the sagittal planes	128	-660/ +660	-360/ +360
45	r_wrist_pivot	Rotation of the hand in the sagittal planes	128	-660/ +660	-360/ +360
46	l_wrist_twisting	Rotation of the hand along the forearm axis	256	-660/ +660	-360/ +360
47	r_wrist_twisting	Rotation of the hand along the forearm axis	256	-660/ +660	-360/ +360
48	skullbase_roll	Sideward motion of the skull along the frontal axis	128	-1860/ +1860	-600/ +600
49	skullbase_torsion	Twisting of the skull along the vertical axis	128	-1860/ +1860	-600/ +600
50	skullbase_tilt	Forward-backward motion in the sagittal plane along a lateral axis	128	-1860/ +1860	-600/ +600
51	vc1_roll	Sideward motion of vertebra C1	256	-240/ +240	-120/ +120

© ISO/IEC 2004 – All rights reserved

52	vc1_torsion	Twisting of vertebra C1	256	-240/ +240	-120/ +120
53	vc1_tilt	Forward-backward motion of vertebra C1 in the sagittal plane	256	-240/ +240	-120/ +120
54	vc2_roll	Sideward motion of vertebra C2	128	-660/ +660	-360/ +360
55	vc2_torsion	Twisting of vertebra C2	128	-660/ +660	-360/ +360
56	vc2_tilt	Forward-backward motion of vertebra C2 in the sagittal plane	128	-660/ +660	-360/ +360
57	vc3_roll	Sideward motion of vertebra C3	256	-240/ +240	-120/ +120
58	vc3_torsion	Twisting of vertebra C3	256	-240/ +240	-120/ +120
59	vc3_tilt	Forward-backward motion of vertebra C3 in the sagittal plane	256	-240/ +240	-120/ +120
60	vc4_roll	Sideward motion of vertebra C4	128	-660/ +660	-360/ +360
61	vc4_torsion	Twisting of vertebra C4	128	-660/ +660	-360/ +360
62	vc4_tilt	Forward-backward motion of vertebra C4 in the sagittal plane	128	-660/ +660	-360/ +360
63	vc5_roll	Sideward motion of vertebra C5	256	-240/ +240	-120/ +120
64	vc5_torsion	Twisting of vertebra C5	256	-240/ +240	-120/ +120
65	vc5_tilt	Forward-backward motion of vertebra C5 in the sagittal plane	256	-240/ +240	-120/ +120
66	vc6_roll	Sideward motion of vertebra C6	256	-240/ +240	-120/ +120
67	vc6_torsion	Twisting of vertebra C6	256	-240/ +240	-120/ +120
68	vc6_tilt	Forward-backward motion of vertebra C6 in the sagittal plane	256	-240/ +240	-120/ +120
69	vc7_roll	Sideward motion of vertebra C7	256	-240/ +240	-120/ +120
70	vc7_torsion	Twisting of vertebra C7	256	-240/ +240	-120/ +120
71	vc7_tilt	Forward-backward motion of vertebra C7 in the sagittal plane	256	-240/ +240	-120/ +120
72	vt1_roll	Sideward motion of vertebra T1	128	-660/ +660	-360/ +360
73	vt1_torsion	Twisting of vertebra T1	128	-660/ +660	-360/ +360
74	vt1_tilt	Forward-backward motion of vertebra T1 in the sagittal plane	128	-660/ +660	-360/ +360
75	vt2_roll	Sideward motion of vertebra T2	256	-240/ +240	-120/ +120
76	vt2_torsion	Twisting of vertebra T2	256	-240/ +240	-120/ +120

© ISO/IEC 2004 – All rights reserved

77	vt2_tilt	Forward-backward motion of vertebra T2 in the sagittal plane	256	-240/ +240	-120/ +120
78	vt3_roll	Sideward motion of vertebra T3	256	-240/ +240	-120/ +120
79	vt3_torsion	Twisting of vertebra T3	256	-240/ +240	-120/ +120
80	vt3_tilt	Forward-backward motion of vertebra T3 in the sagittal plane	256	-240/ +240	-120/ +120
81	vt4_roll	Sideward motion of vertebra T4	256	-240/ +240	-120/ +120
82	vt4_torsion	Twisting of vertebra T4	256	-240/ +240	-120/ +120
83	vt4_tilt	Forward-backward motion of vertebra T4 in the sagittal plane	256	-240/ +240	-120/ +120
84	vt5_roll	Sideward motion of vertebra T5	256	-240/ +240	-120/ +120
85	vt5_torsion	Twisting of vertebra T5	256	-240/ +240	-120/ +120
86	vt5_tilt	Forward-backward motion of vertebra T5 in the sagittal plane	256	-240/ +240	-120/ +120
87	vt6_roll	Sideward motion of vertebra T6	128	-660/ +660	-360/ +360
88	vt6_torsion	Twisting of vertebra T6	128	-660/ +660	-360/ +360
89	vt6_tilt	Forward-backward motion of vertebra T6 in the sagittal plane	128	-660/ +660	-360/ +360
90	vt7_roll	Sideward motion of vertebra T7	256	-240/ +240	-120/ +120
91	vt7_torsion	Twisting of vertebra T7	256	-240/ +240	-120/ +120
92	vt7_tilt	Forward-backward motion of vertebra T7 in the sagittal plane	256	-240/ +240	-120/ +120
93	vt8_roll	Sideward motion of vertebra T8	256	-240/ +240	-120/ +120
94	vt8_torsion	Twisting of vertebra T8	256	-240/ +240	-120/ +120
95	vt8_tilt	Forward-backward motion of vertebra T8 in the sagittal plane	256	-240/ +240	-120/ +120
96	vt9_roll	Sideward motion of vertebra T9	256	-240/ +240	-120/ +120
97	vt9_torsion	Twisting of vertebra T9	256	-240/ +240	-120/ +120
98	vt9_tilt	Forward-backward motion of vertebra T9 in the sagittal plane	256	-240/ +240	-120/ +120
99	vt_10roll	Sideward motion of vertebra T10	128	-660/ +660	-360/ +360
100	vt10_torsion	Twisting of vertebra T10	128	-660/ +660	-360/ +360
101	vt10_tilt	Forward-backward motion of vertebra T10 in sagittal plane	128	-660/ +660	-360/ +360

© ISO/IEC 2004 – All rights reserved

102	vt11_roll	Sideward motion of vertebra T11	256	-240/ +240	-120/ +120
103	vt11_torsion	Twisting of vertebra T11	256	-240/ +240	-120/ +120
104	vt11_tilt	Forward-backward motion of vertebra T11 in sagittal plane	256	-240/ +240	-120/ +120
105	vt12_roll	Sideward motion of vertebra T12	256	-240/ +240	-120/ +120
106	vt12_torsion	Twisting of vertebra T12	256	-240/ +240	-120/ +120
107	vt12_tilt	Forward-backward motion of vertebra T12 in sagittal plane	256	-240/ +240	-120/ +120
108	vl1_roll	Sideward motion of vertebra L1	128	-660/ +660	-360/ +360
109	vl1_torsion	Twisting of vertebra L1	128	-660/ +660	-360/ +360
110	vl1_tilt	Forward-backward motion of vertebra L1 in sagittal plane	128	-660/ +660	-360/ +360
111	vl2_roll	Sideward motion of vertebra L2	256	-240/ +240	-120/ +120
112	vl2_torsion	Twisting of vertebra L2	256	-240/ +240	-120/ +120
113	vl2_tilt	Forward-backward motion of vertebra L2 in sagittal plane	256	-240/ +240	-120/ +120
114	vl3_roll	Sideward motion of vertebra L3	128	-660/ +660	-360/ +360
115	vl3_torsion	Twisting of vertebra L3	128	-660/ +660	-360/ +360
116	vl3_tilt	Forward-backward motion of vertebra L3 in sagittal plane	128	-660/ +660	-360/ +360
117	vl4_roll	Sideward motion of vertebra L4	256	-240/ +240	-120/ +120
118	vl4_torsion	Twisting of vertebra L4	256	-240/ +240	-120/ +120
119	vl4_tilt	Forward-backward motion of vertebra L4 in sagittal plane	256	-240/ +240	-120/ +120
120	vl5_roll	Sideward motion of vertebra L5	128	-660/ +660	-360/ +360
121	vl5_torsion	Twisting of vertebra L5	128	-660/ +660	-360/ +360
122	vl5_tilt	Forward-backward motion of vertebra L5 in sagittal plane	128	-660/ +660	-360/ +360
123	l_pinky0_flexion	Metacarpal flexing mobility of the pinky finger	512	-240/ +240	-120/ +120
124	r_pinky0_flexion	Metacarpal flexing mobility of the pinky finger	512	-240/ +240	-120/ +120
125	l_pinky1_flexion	First knukle of the pinky finger	128	-1140/ +240	-240/ +240
126	r_pinky1_flexion	First knukle of the pinky finger	128	-1140/ +240	-240/ +240

© ISO/IEC 2004 – All rights reserved

127	l_pinky1_pivot	Lateral mobility of the pinky finger	128	-420/ +120	-120/ +120
128	r_pinky1_pivot	Lateral mobility of the pinky finger	128	-120/ +420	-120/ +120
129	l_pinky1_twisting	Along the pinky finger axis	256	-180/ +360	-120/ +120
130	r_pinky1_twisting	Along the pinky finger axis	256	-360/ +180	-120/ +120
131	l_pinky2_flexion	Second knuckle of the pinky number	128	-1380/ +60	-240/ +240
132	r_pinky2_flexion	Second knuckle of the pinky number	128	-1380/ +60	-240/ +240
133	l_pinky3_flexion	Third knuckle of the pinky finger	128	-960/ +60	-240/ +240
134	r_pinky3_flexion	Third knuckle of the pinky finger	128	-960/ +60	-240/ +240
135	l_ring0_flexion	Metacarpal flexing mobility of the ring finger	256	-180/ +180	-120/ +120
136	r_ring0_flexion	Metacarpal flexing mobility of the ring finger	256	-180/ +180	-120/ +120
137	l_ring1_flexion	First knukle of the ring finger	128	-1140/ +360	-240/ +240
138	r_ring1_flexion	First knukle of the ring finger	128	-1140/ +360	-240/ +240
139	l_ring1_pivot	Lateral mobility of the ring finger	128	-240/ +120	-120/ +120
140	r_ring1_pivot	Lateral mobility of the ring finger	128	-120/ +240	-120/ +120
141	l_ring1_twisting	Along the ring finger axis	256	-240/ +240	-120/ +120
142	r_ring1_twisting	Along the ring finger axis	256	-240/ +240	-120/ +120
143	l_ring2_flexion	Second knuckle of the ring number	128	-1380/ +60	-240/ +240
144	r_ring2_flexion	Second knuckle of the ring number	128	-1380/ +60	-240/ +240
145	l_ring3_flexion	Third knuckle of the ring finger	128	-960/ +60	-240/ +240
146	r_ring3_flexion	Third knuckle of the ring finger	128	-960/ +60	-240/ +240
147	l_middle0_flexion	Metacarpal flexing mobility of the middle finger	256	-120/ +120	-120/ +120
148	r_middle0_flexion	Metacarpal flexing mobility of the middle finger	256	-120/ +120	-120/ +120
149	l_middle1_flexion	First knukle of the middle finger	128	-1140/ +360	-240/ +240
150	r_middle1_flexion	First knukle of the middle finger	128	-1140/ +360	-240/ +240
151	l_middle1_pivot	Lateral mobility of the middle finger	128	-180/ +180	-120/ +120

© ISO/IEC 2004 – All rights reserved

152	r_middle1_pivot	Lateral mobility of the middle finger	128	-180/ +180	-120/ +120
153	l_middle1_twisting	Along the middle finger axis	256	-180/ +180	-120/ +120
154	r_middle1_twisting	Along the middle finger axis	256	-180/ +180	-120/ +120
155	l_middle2_flexion	Second knuckle of the middle number	128	-1380/ +60	-240/ +240
156	r_middle2_flexion	Second knuckle of the middle number	128	-1380/ +60	-240/ +240
157	l_middle3_flexion	Third knuckle of the middle finger	128	-960/ +60	-240/ +240
158	r_middle3_flexion	Third knuckle of the middle finger	128	-960/ +60	-240/ +240
159	l_index0_flexion	Metacarpal flexing mobility of the index finger	256	-60/ +60	-60/ +60
160	r_index0_flexion	Metacarpal flexing mobility of the index finger	256	-60/ +60	-60/ +60
161	l_index1_flexion	First knukle of the index finger	128	-1140/ +360	-240/ +240
162	r_index1_flexion	First knukle of the index finger	128	-1140/ +360	-240/ +240
163	l_index1_pivot	Lateral mobility of the index finger	128	-120/ +240	-60/ +60
164	r_index1_pivot	Lateral mobility of the index finger	128	-240/ +120	-60/ +60
165	l_index1_twisting	Along the index finger axis	256	-240/ +180	-120/ +120
166	r_index1_twisting	Along the index finger axis	256	-180/ +240	-120/ +120
167	l_index2_flexion	Second knuckle of the index number	128	-1380/ +60	-240/ +240
168	r_index2_flexion	Second knuckle of the index number	128	-1380/ +60	-240/ +240
169	l_index3_flexion	Third knuckle of the index finger	128	-960/ +60	-240/ +240
170	r_index3_flexion	Third knuckle of the index finger	128	-960/ +60	-240/ +240
171	l_thumb1_flexion	First knukle of the thumb finger	128	-480/ +960	-240/ +240
172	r_thumb1_flexion	First knukle of the thumb finger	128	-960/ +480	-240/ +240
173	l_thumb1_pivot	Lateral mobility of the thumb finger	128	-120/ +1080	-240/ +240
174	r_thumb1_pivot	Lateral mobility of the thumb finger	128	-1080/ +120	-240/ +240
175	l_thumb1_twisting	Along the thumb finger axis	256	-720/ +120	-240/ +240
176	r_thumb1_twisting	Along the thumb finger axis	256	-720/ +120	-240/ +240

© ISO/IEC 2004 – All rights reserved

177	l_thumb2_flexion	Second knuckle of the thumb number	128	-780/ +60	-240/ +240
178	r_thumb2_flexion	Second knuckle of the thumb number	128	-780/ +60	-240/ +240
179	l_thumb3_flexion	Third knuckle of the thumb finger	128	-960/ +60	-240/ +240
180	r_thumb3_flexion	Third knuckle of the thumb finger	128	-960/ +60	-240/ +240
181	HumanoidRoot_tr_vertical	Body origin translation in vertical direction	64	-1860/ +1860	-600/ +600
182	HumanoidRoot_tr_lateral	Body origin translation in lateral direction	64	-1860/ +1860	-600/ +600
183	HumanoidRoot_tr_frontal	Body origin translation in frontal direction	64	-1860/ +1860	-600/ +600
184	HumanoidRoot_rt_body_turn	Rotation of the skeleton root along the body coordinate system's vertical axis	64	-1860/ +1860	-600/ +600
185	HumanoidRoot_rt_body_roll	Rotation of the skeleton root along the body coordinate system's frontal axis	64	-1860/ +1860	-600/ +600
186	HumanoidRoot_rt_body_tilt	Rotation of the skeleton root along the body coordinate system's lateral axis	64	-1860/ +1860	-600/ +600

BAPs 187 through 296 are extension BAPs for use with the BodyDefTable nodes., see ISO/IEC 14496-1:2001 for specification of BodyDefTable nodes. These BAPs are user-defined. The step sizes of these BAPs are 1, and I-Frame and P-Frame default minmax values are not defined.

GROUP ID	GROUP NAME	BAPS
1	Pelvis	sacroiliac_tilt, sacroiliac_torsion, sacroiliac_roll (1,2,3)
2	Left leg1	l_hip_flexion, l_hip_abduct, l_knee_flexion, l_ankle_flexion (4,6,10,14)
3	Right leg1	r_hip_flexion, r_hip_abduct, r_knee_flexion, r_ankle_flexion (5,7,11,15)
4	Left leg2	l_hip_twisting, l_knee_twisting, l_ankle_twisting, l_subtalar_flexion, l_midtarsal_flexion, l_metatarsal_flexion (8,12,16,18,20,22)
5	Right leg2	r_hip_twisting, r_knee_twisting, r_ankle_twisting, r_subtalar_flexion, r_midtarsal_flexion, r_metatarsal_flexion (9,13,17,19,21,23)
6	Left arm1	l_shoulder_flexion, l_shoulder_abduct, l_shoulder_twisting, l_elbow_flexion, l_wrist_flexion (32,34,36,38,42)
7	Right arm1	r_shoulder_flexion, r_shoulder_abduct, r_shoulder_twisting, r_elbow_flexion, r_wrist_flexion (33,35,37,39,43)
8	Left arm2	l_sternoclavicular_abduct, l_sternoclavicular_rotate, l_acromioclavicular_abduct, l_acromioclavicular_rotate, l_elbow_twisting, l_wrist_pivot, l_wrist_twisting (24,26,28,30,40,44,46)

© ISO/IEC 2004 – All rights reserved

9	Right arm2	r_sternoclavicular_abduct, r_sternoclavicular_rotate, r_acromioclavicular_abduct, r_acromioclavicular_rotate, r_elbow_twisting, r_wrist_pivot, r_wrist_twisting (25,27,29,31,41,45,47)
10	Spine1	skullbase_roll, skullbase_torsion, skullbase_tilt, vc4roll, vc4torsion, vc4tilt, vt6roll, vt6torsion, vt6tilt, vl3roll, vl3torsion, vl3tilt, (48,49,50,60,61,62,87,88,89,114,115,116)
11	Spine2	vc2roll, vc2torsion, vc2tilt, vt1roll, vt1torsion, vt1tilt, vt10roll, vt10torsion, vt10tilt, vl1roll, vl1torsion, vl1tilt, vl5roll, vl5torsion, vl5tilt (54,55,56,72,73,74,99,100,101,108, 109,110,120,121,122)
12	Spine3	vc3roll, vc3torsion, vc3tilt, vc6roll, vc6torsion, vc6tilt, vt4roll, vt4torsion, vt4tilt, vt8roll, vt8torsion, vt8tilt, vt12roll, vt12torsion, vt12tilt vl4roll, vl4torsion, vl4tilt, (57,58,59,66,67,68,81,82,83,93,94,95, 105,106,107,117,118,119)
13	Spine4	vc5roll, vc5torsion, vc5tilt, vc7roll, vc7torsion, vc7tilt vt2roll, vt2torsion, vt2tilt, vt7roll, vt7torsion, vt7tilt, vt11roll, vt11torsion, vt11tilt, vl2roll, vl2torsion, vl2tilt, (63,64,65,69,70,71,75,76,77,90,91, 92,102,103,104,111,112,113)
14	Spine5	vc1roll, vc1torsion, vc1tilt, vt3roll, vt3torsion, vt3tilt, vt5roll, vt5torsion, vt5tilt, vt9roll, vt9torsion, vt9tilt, (51,52,53,78,79,80,84,85,86,96,97,98)
15	Left hand1	l_pinky1_flexion, l_pinky2_flexion, l_pinky3_flexion, l_ring1_flexion, l_ring2_flexion, l_ring3_flexion, l_middle1_flexion, l_middle2_flexion,l_middle3_flexion, l_index1_flexion, l_index2_flexion, l_index3_flexion, l_thumb1_flexion, l_thumb1_pivot, l_thumb2_flexion, l_thumb3_flexion (125,131,133,137,143,145,149,155,157, 161,167,169,171,173,177,179)

© ISO/IEC 2004 – All rights reserved

16	Right hand1	r_pinky1_flexion, r_pinky2_flexion, r_pinky3_flexion, r_ring1_flexion, r_ring2_flexion, r_ring3_flexion, r_middle1_flexion, r_middle2_flexion,r_middle3_flexion, r_index1_flexion, r_index2_flexion, r_index3_flexion, r_thumb1_flexion, r_thumb1_pivot, r_thumb2_flexion, r_thumb3_flexion (126,132,134,138,144,146,150,156,158, 162,168,170,172,174,178,180)
17	Left hand2	l_pinky0_flexion, l_pinky1_ pivot, l_pinky1_twisting, l_ring0_flexion, l_ring1_ pivot, l_ring1_twisting, l_middle0_flexion, l_middle1_ pivot, l_middle1_twisting, l_index0_flexion, l_index1_ pivot, l_index1_twisting, l_thumb1_twisting (123,127,129,135,139,141,147, 151,153,159,163,165,175)
18	Right hand2	r_pinky0_flexion, r_pinky1_ pivot, r_pinky1_twisting, r_ring0_flexion, r_ring1_ pivot, r_ring1_twisting, r_middle0_flexion, r_middle1_ pivot, r_middle1_twisting, r_index0_flexion, r_index1_ pivot, r_index1_twisting, r_thumb1_twisting (124,128,130,136,140,142,148, 152,154,160,164,166,176)
19	Global positioning	HumanoidRoot_tr_vertical, HumanoidRoot_tr_lateral, HumanoidRoot_tr_frontal, HumanoidRoot_rt_body_turn, HumanoidRoot_rt_body_roll, HumanoidRoot_rt_body_tilt (181,182,183,184,185,186)

Additionally, groups 20 to 24 contain extension BAPs. Extension BAPs are user-defined BAPs for use with BodyDefTables. See Systems CD for specification of BodyDefTable nodes. Groups 20 to 24 each contain 22 BAPs. Thus, the extension BAP groups are as follows:

GROUP ID	GROUP NAME	BAPs
20	Extension1	187…208
21	Extension2	209…230
22	Extension3	231…252
23	Extension4	253…274
24	Extension5	275…296

Detailed Degrees of Freedom

The ***degrees of freedom (DOF)*** sufficient to locate and animate a human body are decomposed into six DOF for the global location and 180 DOF for the internal mobilities. The topology of the suggested joints is given below. The hands are optional with an additional set of joints.

© ISO/IEC 2004 – All rights reserved

Suggested default joint center values for broadcast applications

HumanoidRoot	0.0000	0.9723	-0.0728
sacroiliac	0.0000	0.9710	-0.0728
l_hip	0.0956	0.9364	0.0000
l_knee	0.0956	0.5095	-0.0036
l_ankle	0.0946	0.0762	-0.0261
l_subtalar	0.0956	0.0398	0.0069
l_midtarsal	0.1079	0.0317	0.0670
l_metatarsal	0.0942	0.0092	0.1239
r_hip	-0.0956	0.9364	0.0000
r_knee	-0.0956	0.5095	-0.0036
r_ankle	-0.0946	0.0762	-0.0261
r_subtalar	-0.0956	0.0398	0.0069
r_midtarsal	-0.1079	0.0317	0.0670
r_metatarsal	-0.0942	0.0092	0.1239
vl5	0.0000	1.0817	-0.0728
vl4	0.0000	1.1174	-0.0727
vl3	0.0000	1.1525	-0.0727
vl2	0.0000	1.1795	-0.0727
vl1	0.0000	1.2161	-0.0727
vt12	0.0000	1.2527	-0.0727
vt11	0.0000	1.2918	-0.0727
vt10	0.0000	1.3098	-0.0737
vt9	0.0000	1.3375	-0.0752
vt8	0.0000	1.3631	-0.0758
vt7	0.0000	1.3875	-0.0745
vt6	0.0000	1.4116	-0.0712
vt5	0.0000	1.4351	-0.0657
vt4	0.0000	1.4569	-0.0587
vt3	0.0000	1.4832	-0.0482
vt2	0.0000	1.5011	-0.0397
vt1	0.0000	1.5201	-0.0300
vc7	0.0000	1.5382	-0.0213
vc6	0.0000	1.5607	-0.0073
vc5	0.0000	1.5770	-0.0012
vc4	0.0000	1.5912	-0.0014
vc3	0.0000	1.6050	-0.0033
vc2	0.0000	1.6178	-0.0033
vc1	0.0000	1.6394	0.0036
skullbase	0.0000	1.6440	0.0036
l_sternoclavicular	0.0757	1.4844	-0.0251
l_acromioclavicular	0.0899	1.4525	-0.0322
l_shoulder	0.1968	1.4642	-0.0265
l_elbow	0.1982	1.1622	-0.0557
l_wrist	0.1972	0.8929	-0.0690
l_thumb1	0.1912	0.8734	-0.0657
l_thumb2	0.1912	0.8156	-0.0079
l_thumb3	0.1912	0.8007	0.0070
l_index0	0.1912	0.8259	-0.0460

© ISO/IEC 2004 – All rights reserved

l_index1	0.1912	0.8050	-0.0460
l_index2	0.1912	0.7595	-0.0460
l_index3	0.1912	0.7370	-0.0460
l_middle0	0.1912	0.8282	-0.0710
l_middle1	0.1912	0.8071	-0.0710
l_middle2	0.1912	0.7525	-0.0710
l_middle3	0.1912	0.7263	-0.0710
l_ring0	0.1912	0.8302	-0.0972
l_ring1	0.1912	0.8098	-0.0972
l_ring2	0.1912	0.7570	-0.0972
l_ring3	0.1912	0.7328	-0.0972
l_pinky0	0.1912	0.8373	-0.1211
l_pinky1	0.1912	0.8173	-0.1211
l_pinky2	0.1912	0.7759	-0.1211
l_pinky3	0.1912	0.7584	-0.1211
r_sternoclavicular	-0.0757	1.4844	-0.0251
r_acromioclavicular	-0.0899	1.4525	-0.0322
r_shoulder	-0.1968	1.4642	-0.0265
r_elbow	-0.1982	1.1622	-0.0557
r_wrist	-0.1972	0.8929	-0.0690
r_thumb1	-0.1912	0.8734	-0.0657
r_thumb2	-0.1912	0.8156	-0.0079
r_thumb3	-0.1912	0.8007	0.0070
r_index0	-0.1912	0.8259	-0.0460
r_index1	-0.1912	0.8050	-0.0460
r_index2	-0.1912	0.7595	-0.0460
r_index3	-0.1912	0.7370	-0.0460
r_middle0	-0.1912	0.8282	-0.0710
r_middle1	-0.1912	0.8071	-0.0710
r_middle2	-0.1912	0.7525	-0.0710
r_middle3	-0.1912	0.7263	-0.0710
r_ring0	-0.1912	0.8302	-0.0972
r_ring1	-0.1912	0.8098	-0.0972
r_ring2	-0.1912	0.7570	-0.0972
r_ring3	-0.1912	0.7328	-0.0972
r_pinky0	-0.1912	0.8373	-0.1211
r_pinky1	-0.1912	0.8173	-0.1211
r_pinky2	-0.1912	0.7759	-0.1211
r_pinky3	-0.1912	0.7584	-0.1211

© ISO/IEC 2004 – All rights reserved

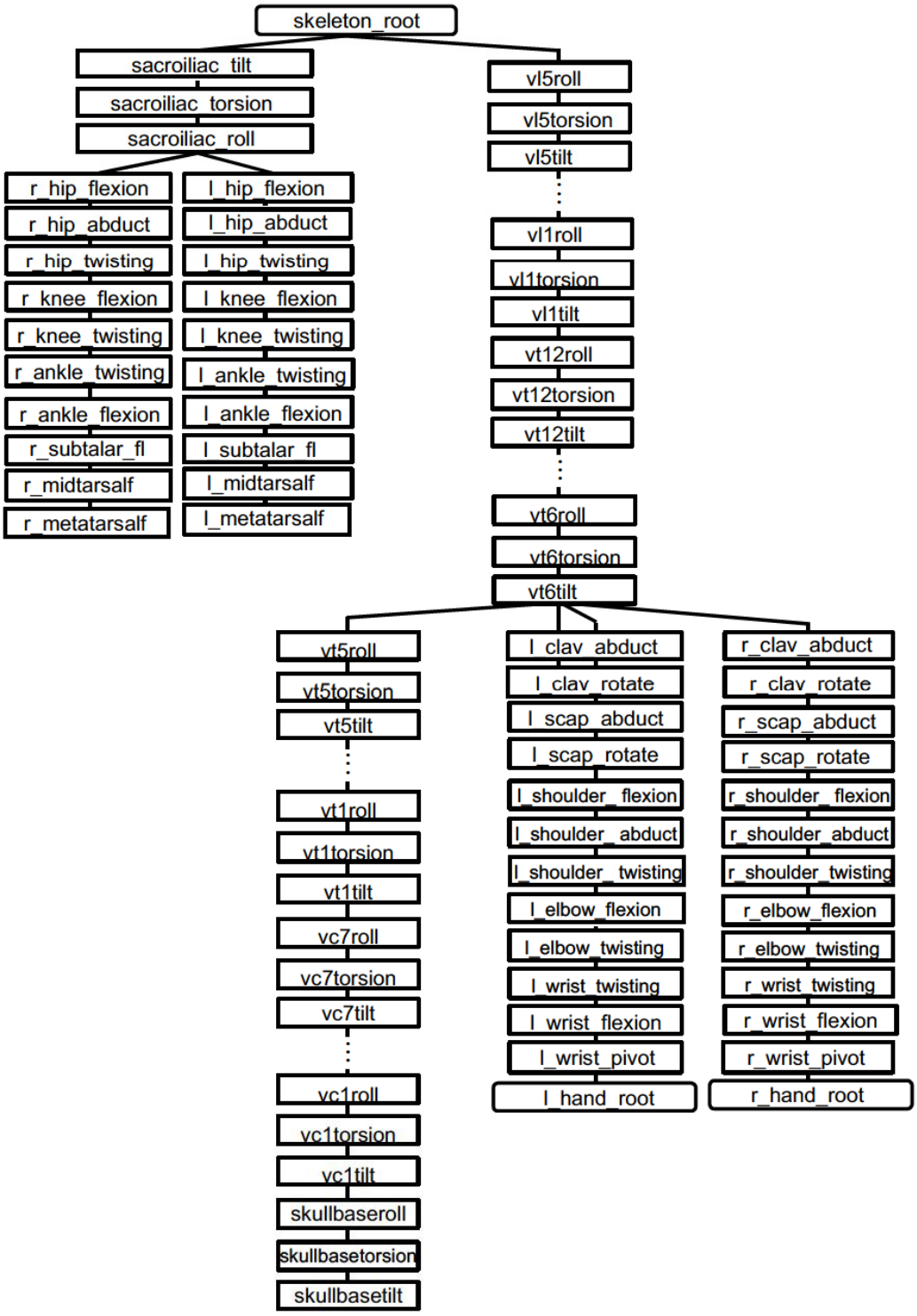


Figure C.2 — Body topology

© ISO/IEC 2004 – All rights reserved

Lower Body

From the skeleton_root node, three DOF allow flexible pelvic orientation followed by nine DOF per leg, from the hip joint to the toe joint.

First, the pelvic mobilities are very important to convey a gender personification to the motion. The naming convention of these mobilities is also used for the spine DOF. The degrees of freedom are:

sacroiliac_tilt : forward-backward motion in the sagital plane

sacroiliac_torsion : rotation along the body vertical axis (defined by skeleton root)

sacroiliac_roll : side to side swinging in the coronal plane

The leg mobilities follow in this orde :

At the hip :

hip_flexion : forward-backward rotation in the sagital plane

hip_abduct : sideward opening in the coronal plane

hip_twisting : rotation along the thigh axis.

At the knee:

knee_flexion : flexion-extension of the leg in the sagital plane

knee_twisting : rotation along the shank axis.

At the ankle:

ankle_twisting : rotation along the shank axis. This joint is redundant with the knee_twisting except that only the foot rotate and not the shank segment.

ankle_flexion : flexion-extension of the foot in the sagital plane

At the foot complex:

The foot complex region is completely described with three degrees of freedom with independent position and orientation : the subtalar joint, the *mid_foot* joint, between the subtalar joint and the toe joint to capture complex internal relative movement of the foot bones (also called the navicular joint in the literature).

subtalar_flexion : sideward orientation of the foot

midtarsal_flexion : internal twisting of the foot (also called navicular joint in anatomy)

metatorsal_flexion : up and down rotation of the toe in the sagital plane

© ISO/IEC 2004 – All rights reserved

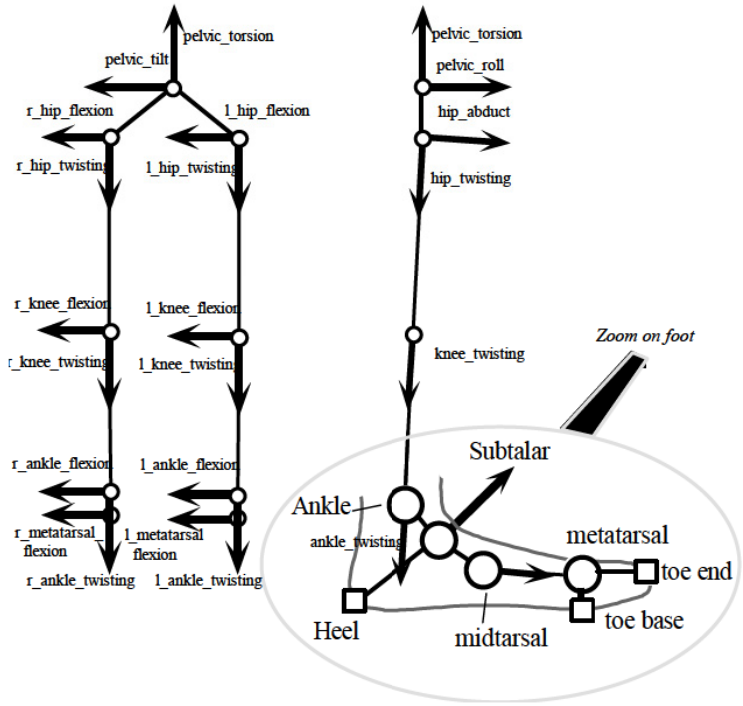


Figure C.3 — Front and side views of the mobilities of the leg

Upper Body

The suggested upper body degrees of freedom and their corresponding axes of rotation are shown in the following diagrams.

© ISO/IEC 2004 – All rights reserved

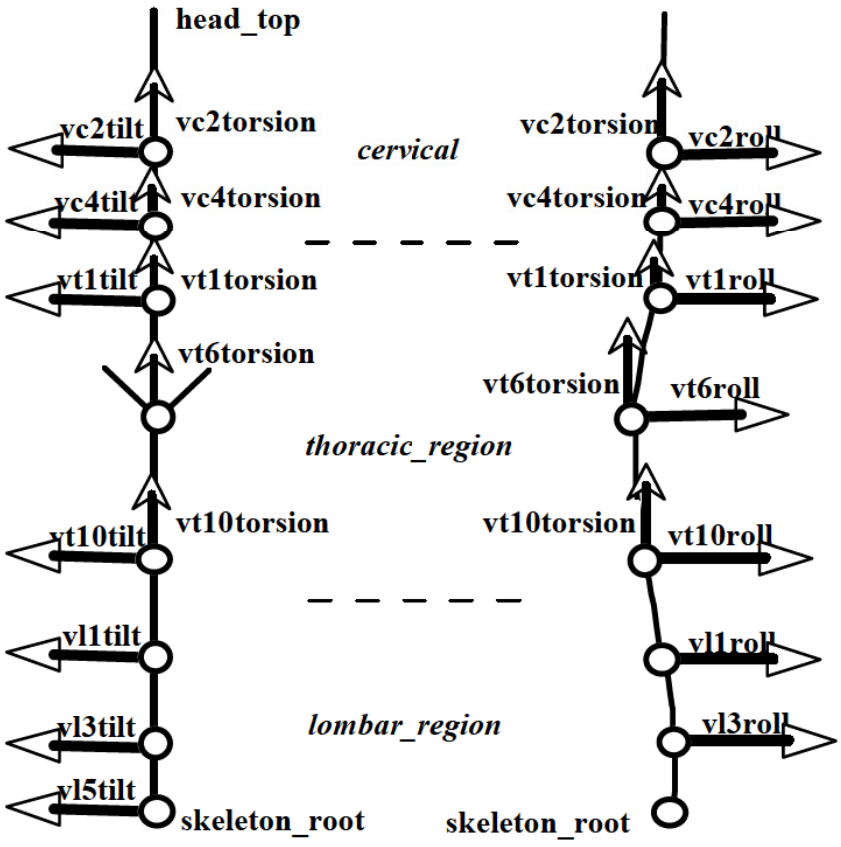


Figure C.4 — Front and side views of the mobilities of the simple spine

At the spine:

The vertebrae are dispatched into 5 *lumbar*, 12 *thoracic* and *cervical* groups.A total of 72 DOFs are defined for the spine, therefore complex applications with whole spine mobilities can be developed. However, typically, the application will use simpler spines that balance the computational speed, with the realism of animation. Therefore, 5 groups of spine are defined, from the simplest to most complex. It is suggested that the spine groups (Spine1, Spine2) are used for simple spine.

At the arms:

The arm structure is attached to the spine. The mobilities are similar to the ones defined for the leg when the arm is twisted such that the hand palm is facing backward. The name of the DOF respects this structure similarity.

Strictly speaking, the two clavicula DOF and the two scapula DOF are not part of the arm structure, they only initiate its articulated chain. The scapula joints improve the mechanical model of the shoulder complex. It should be noted that such a chain representation is only a step toward a mechanically correct representation of this region, mainly because the shoulder complex is in fact a closed loop mechanism. The scapula holds the same mobility as the clavicula but very close to the shoulder joint.

© ISO/IEC 2004 – All rights reserved

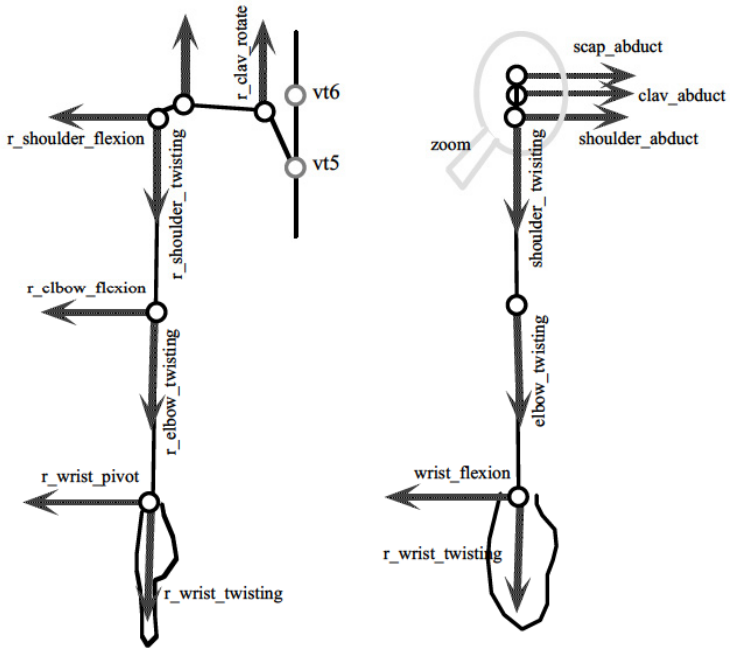


Figure C.5 — Front and side views of the mobilities of the arm in rest position

At the clavicula:

sternoclavicular_abduct : up and down motion in the coronal plane

sternoclavicular_rotate : rotation in the transverse plane

At the scapula:

acromioclavicular_abduct : up and down motion in the coronal plane

acromioclavicular_rotate : rotation in the transverse plane

At the shoulder:

shoulder_flexion : forward-backward motion in the sagital plane

shoulder_abduct : sideward motion in the coronal plane

shoulder_twisting : along the forearm axis

At the elbow:

elbow_flexion : flexion-extension of the arm in the sagital plane

elbow_twisting : along the arm axis.

© ISO/IEC 2004 – All rights reserved

At the wrist:

wrist_twisting : along the arm axis. This DOF is redundant with the elbow twisting except that only the hand rotate and not the forearm

wrist_flexion : rotation of the hand in the coronal plane

wrist_pivot : rotation of the hand in the sagital planes

Head rotations

Note that there are three BAPs defined for head rotation: skullbase_roll, skullbase_torsion, skullbase_tilt. There are also 3 FAPs for head rotation. If both FAPs and BAPs are used for head rotation, then these FAPs shall denote the head rotation with respect to the skullbase coordinate system.

Hands

The hand mobilities have a standard structure for the five fingers. This structure is organized as:

a **flexing** DOF for closing the first knuckle

a **pivoting** rotation for the lateral mobility of the finger

a **twisting** rotation for small adjustments of the finger grasping orientation

two other **flexing** DOF for closing the second and third knuckles

© ISO/IEC 2004 – All rights reserved

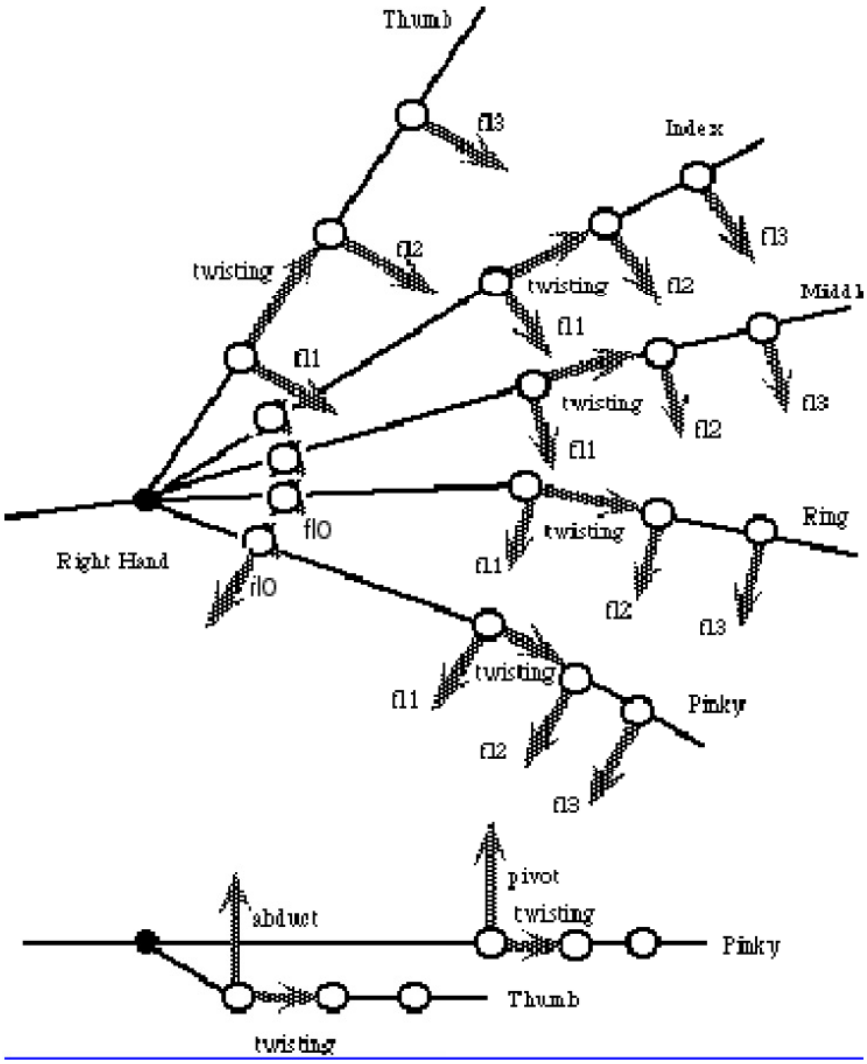


Figure C.6 — Mobilities of the right hand

© ISO/IEC 2004 – All rights reserved

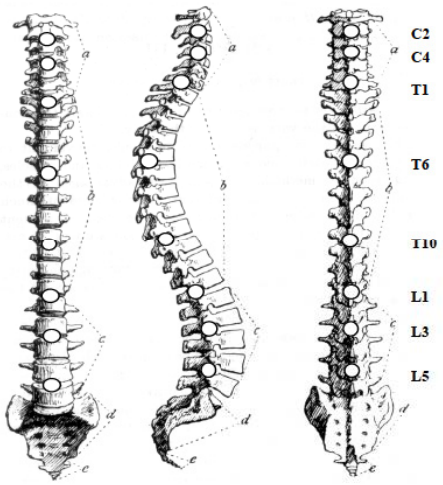


Figure C.7 — Example spine mobilities

Integration of Facial Animation with TTS

The following figure shows the complete block diagram describing the integration of a proprietary TTS Synthesizer into an ISO/IEC 14496 face animation system. The FAP bookmarks defined by the user in the input text of the TTS Stream are identified by the speech synthesizer and transmitted in ASCII format to the Phoneme to FAP converter using the TtsFAPInterface as defined in ISO/IEC 14496-3:2001.

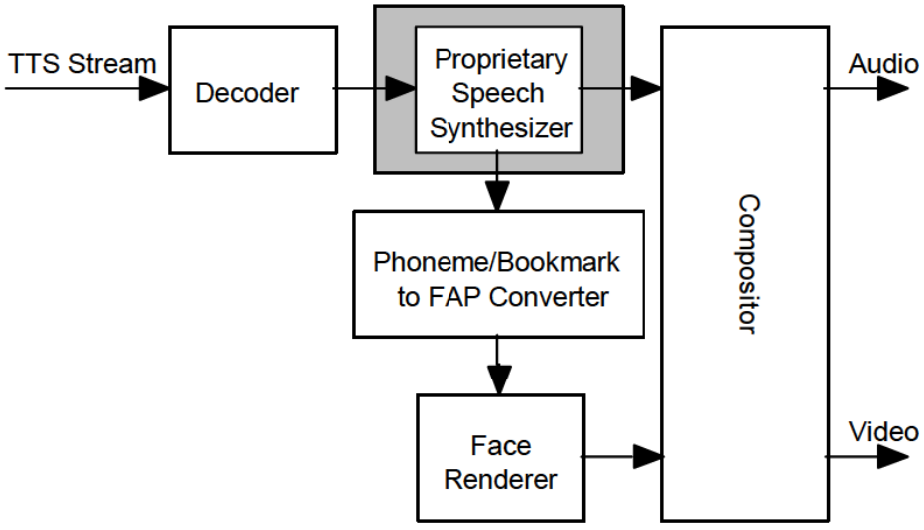


Figure C.8 — Blockdiagram showing the integration of a proprietary TTS into an ISO/IEC 14496 face animation system with Phoneme/Bookmark to FAP Converter.

© ISO/IEC 2004 – All rights reserved

The syntax of the bookmark sequences used to convey commands to the TTS system is the following:

<FAP *n* FAPfields *T C* >
n: FAP number defined according to annex C,Table C.1 with 2<=n<=68.
FAPfields := expression_select1* expression_intensity1 expression_select2* expression_intensity2, in case n == 2
FAPfields := *a***, in case 2 < n < =68
T: transition time defined in ms
C: time curve for computation of *a* during transition time*T*

where ":=" indicates a production rule

* defined according to Table C.3 (expression select)

** defined in units according to Table C.1

Phoneme/Bookmark to FAP Converter

In addition to its function to convert phonemes into visemes, the Phoneme/Bookmark to FAP Converter (Fig. 1) is responsible for translating the FAP bookmarks defined by the user and made available by the bookmark field of the TtsFAPInterface as defined in ISO/IEC 14496-1:2001 into a sequence of FAPs that can be interpreted by the Face Renderer, which then has to use them. An interpolation function is used for computing the FAP amplitudes. The converter merges the FAPs it derives from phonemes and bookmarks into one set of FAP parameters for each time instant using the following steps:

1. Set all low-level FAPs to interpolate.
2. Set specified FAPs to values computed using the interpolation function and phonemes.
3. init_face is set according to the following suggestion: In case of FAP 1 viseme_select1/2 != 0 and FAP 2 expression_select1/2 !=0, init_face=1; In case of FAP 2 expression_select1/2 !=0 and using FAP 3-68 for visemes, init_face=0.

Facial expressions and visemes are combined such that mouth closures are achieved for the relevant visemes while maintaining the overall facial expression. In case that the bookmarks contain visemes, they are ignored.

Transitions between facial expressions are achieved by computing the interpolation function for expression_intensity1 and expression_intensity2. Smooth transitions have to be achieved if expression_select1 and expression_select2 are the same for consecutive bookmarks.

Face Renderer:

In the case that the face model is also animated from an FAP parameter stream, the face renderer has to animate the face using the input from the TtsFAPInterface which is processed by the Phoneme/Bookmark to FAP converter and the input from the FAP decoder. In case that the renderer receives values for the same FAP from both sources, the values derived from the FAP stream take precedence.

Interpolation Functions

For simplicity, the following description on how to compute the amplitude of an FAP is explained using the amplitude *a*. The same method is applied to expression_intensity1 and expression_intensity2.

The FAP amplitude *a* defines the amplitude to be applied at the end of the transition time *T*. The amplitude a_s of the FAP at the beginning of the transition depends on previous bookmarks and can be equal to:

— 0 if the FAP bookmark is the first one with this FAP *n* made available through the TtsFAPInterface.

— *a* of the previous FAP bookmark with the same FAP *n* if a time longer than the previous transition time *T* has elapsed between these two FAP bookmarks.

— The actual reached amplitude due to the previous FAP definition if a time shorter than the previous transition time *T* has elapsed between the two FAP bookmarks.

© ISO/IEC 2004 – All rights reserved

At the end of the transition time *T*, *a* is maintained until another FAP bookmark gives a new value to reach. To reset an FAP, a bookmark for FAP *n* with *a*=0 is transmitted in the text.

To avoid too many parameters for defining the evolution of the amplitude during the transition time, the functions that compute for each frame the amplitude of the FAP to be sent to the face renderer are predefined. Assuming that the transition time *T* is always 1, the following 3 functions *f(t)* are selected according the value of *C*:

$$f(t) = a_s + (a - a_s)t \quad \text{(linear) (1)}$$

$$f(t) = a_s + \begin{cases} (a - a_s)2t & for\ t \le 0.5 \\ (a - a_s)2(1-t) & for\ 0.5 < t \le 1 \end{cases} \quad \text{(triangle) (2)}$$

$$f(t) = (2t^3 - 3t^2 + 1)a_s + (-2t^3 + 3t^2)a + (t^3 - 2t^2 + t)g_s \quad \text{(Hermite function) (3)}$$

with time $t \in [0,1]$, the amplitude a_s at the beginning of the FAP at t=0, and the gradient g_s of $f(0)$ which is the FAP amplitude over time at t=0. If the transition time $T \neq 1$, the time axis of the functions (1) to (4) has to be scaled. These functions depend on a_s, g_s, a and T, and thus they are completely determined as soon as the FAP bookmark is known.

The Hermite function of third order enables one to match the tangent at the beginning of a segment with the tangent at the end of the previous segment, so that a smooth curve can be guarantied. Usually, the computation of the Hermite function requires 4 parameters as input, which are a_s, g_s, a and the gradient of $f(t)$ at t=1. Here a horizontal gradient at the end of the transition time is assumed. The gradient $g_s(t)$ at time t is computed according to

$$g_s(t) = (6t^2 - 6t)(a_s - a) + (3t^2 - 4t + 1)g_s \quad (4)$$

with a_s, g_s ,and a defined by the amplitude prior to the current bookmark.

© ISO/IEC 2004 – All rights reserved

Annex D
(normative)

Video buffering verifier

D.1 Introduction

The video verifier comprises the following models, as shown in Figure D.1:

1. A video rate buffer model
2. A video complexity model, and
3. A video reference memory model

A video rate buffer model is required in order to bound the memory requirements for the bitstream buffer needed by a video decoder. With a rate buffer model, the video encoder can be constrained to make bitstreams which are decodable with a predetermined buffer memory size.

A video complexity model is required in order to bound the processing speed requirements needed by a compliant video decoder. With a video complexity model, the video encoder can be constrained to make bitstreams which are decodable with a predetermined decoder processor capability.

A video reference memory model is required in order to bound the macroblock (pixel) memory requirements needed by a video decoder. With a video reference memory model, a video encoder can be constrained to make bitstreams which are decodable with a predetermined reference memory size.

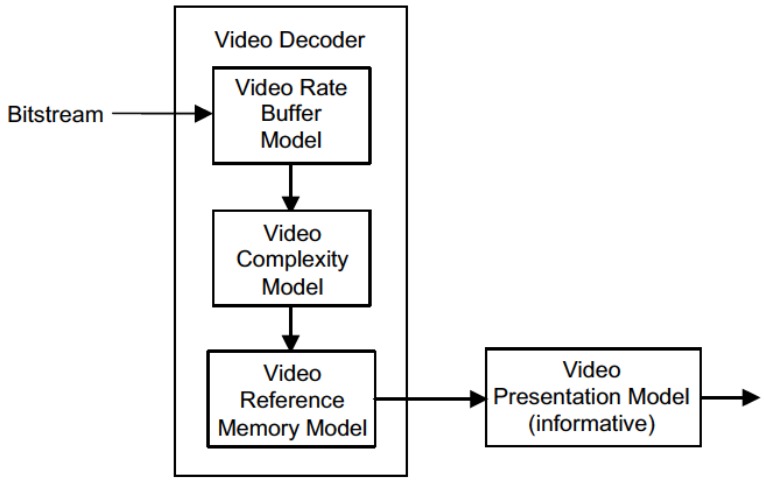


Figure D.1 — Video Verifier Model

D.2 Video Rate Buffer Model Definition

The ISO/IEC 14496-2 video buffering verifier (VBV) is an algorithm for checking a bitstream with its delivery rate function, R(t), to verify that the amount of rate buffer memory required in a decoder is less than the stated buffer size. If a visual bitstream is composed of multiple VOs each with one or more VOLs, the rate buffer model is applied independently to each VOL (using buffer size and rate functions particular to that VOL).

© ISO/IEC 2004 – All rights reserved

The VBV applies to natural video coded as a combination of I, P, B and S-VOPs, and still texture. Face animation and mesh objects are not constrained by this model.

The coded video bitstream shall be constrained to comply with the requirements of the VBV defined as follows:

1. When the vbv_buffer_size and vbv_occupancy parameters are specified by systems-level configuration information, the bitstream shall be constrained according to the specified values. When the vbv_buffer_size and vbv_occupancy parameters are not specified (except in the short video header case as described below), this indicates that the bitstream should be constrained according to the default values of vbv_buffer_size and vbv_occupancy. The default value of vbv_buffer_size is the maximum value of vbv_buffer_size allowed within the profile and level. The default value of vbv_occupancy is 170 × vbv_buffer_size, where vbv_occupancy is in 64-bit units and vbv_buffer_size is in 16384-bit units. This corresponds to an initial occupancy of approximately two-thirds of the full buffer size.

2. The VBV buffer size is specified by the vbv_buffer_size field in the VOL header in units of 16384 bits. A vbv_buffer_size of 0 is forbidden. Define B = 16384 × vbv_buffer_size to be the VBV buffer size in bits.

3. The instantaneous video object layer channel bit rate seen by the encoder is denoted by $R_{vol}(t)$ in bits per second. If the bit_rate field in the VOL header is present, it defines a peak rate (in units of 400 bits per second; a value of 0 is forbidden) such that $R_{vol}(t) <= 400 \times$ bit_rate. Note that $R_{vol}(t)$ counts only visual syntax for the current VOL (refer to the definition of d_i below). If the channel is a serial time mutiplex containing other VOLs or as defined by ISO/IEC 14496-1:2001 with a total instantaneous channel rate seen by the encoder of R(t), then

$$R_{vol}(t) = \begin{cases} R(t) & \text{if } t \in \{\text{channel bit duration of a bit from VOL } vol\} \\ 0 & \text{otherwise} \end{cases}$$

4. The VBV buffer is initially empty. The vbv_occupancy field specifies the initial occupancy of the VBV buffer in 64-bit units before decoding the initial VOP, subtracted by the length of any immediately preceding configuration information (see subclause 6.2.1). For basic sprite sequences, the vbv_occupancy field specifies the initial occupancy of the VBV buffer in 64-bit units before decoding the first S-VOP in the elementary stream (see subclause 6.2.1). The first bit in the VBV buffer is the first bit of the visual bitstream including the elementary stream and any immediately preceding configuration information for all types of sequences.

5. Define d_i to be size in bits of the i-th VOP plus any immediately preceding GOV header and configuration information and any immediately following stuffing_start_code and stuffing_byte(s), where i is the VOP index which increments by 1 in decoding order. A VOP includes any trailing stuffing code words before the next start code and the size of a coded VOP (d_i) is always a multiple of 8 bits due to start code alignment.

6. Let t_i be the decoding time associated with VOP i in decoding order. All bits (d_i) of VOP i are removed from the VBV buffer instantaneously at t_i. This instantaneous removal property distinguishes the VBV buffer model from a real rate buffer. The method of determining the value of t_i is defined in item 7 below.

7. τ_i is the composition time (or presentation time in a no-compositor decoder) of VOP *i*. For a video object plane, τ_i defined by vop_time_increment (in units of 1/vop_time_increment_resolution seconds) plus the cumulative number of whole seconds specified by modulo_time_base. In the case of interlaced video, a VOP consists of lines from two fields and τ_i is the composition time of the first field. The relationship between the composition time and the decoding time for a VOP is given by:

   $t_i = \tau_i$ if ((vop_coding_type of VOP *i* == B-VOP) || low_delay || scalability || sprite_enable == "static")

   $t_i = \tau_{p(i)}$ otherwise

   Low_delay, scalability and sprite_enable are defined in subclause 6.3.3, and *p(i)* is the index of the nearest temporally-previous non-B VOP relative to VOP *i*. If low_delay, scalability and sprite_enable are all '0', then the composition time of I, P, and S(GMC) VOP's is delayed until all immediately temporally-previous B-VOPs have been composed; the decoding time of VOP *i* is then equal to the composition time of the temporally-previous non-B VOP instead of the composition time of VOP *i* itself.

© ISO/IEC 2004 – All rights reserved

For the first VOP (*i*=0), if low_delay, scalability, and sprite_enable are all '0', it is necessary to define an initial decoding time t_0 (since the timing structure is locked to the B-VOP times and the first decoded VOP would not be a B-VOP). This defined decoding timing shall be that $t_0 = 2t_1 - t_2$ (i.e., assuming that $t_1 - t_0 = t_2 - t_1$), since $\tau_{p(0)}$ is not defined.

The following example demonstrates how t_i is determined for a sequence with variable numbers of consecutive B-VOPs:

Decoding order : $I_0 P_1 P_2 P_3 B_4 P_5 B_6 P_7 B_8 B_9 P_{10} B_{11} B_{12}$

Display order : $I_0 P_1 P_2 B_4 P_3 B_6 P_5 B_8 B_9 P_7 B_{11} B_{12} P_{10}$

Assume that vop_time_increment=1 and modulo_time_base=0 in this example. The sub-index *i* is in decoding order.

Table D.1 — An example that demonstrates how t_i is determined

i	τ_i	t_i	$p\ (i)$
0	0	-1	Not defined
1	1	0	0
2	2	1	1
3	4	2	2
4	3	3	Irrelevant
5	6	4	3
6	5	5	Irrelevant
7	9	6	5
8	7	7	Irrelevant
9	8	8	Irrelevant
10	12	9	7
11	10	10	Irrelevant
12	11	11	Irrelevant

8. Define b_i as the buffer occupancy in bits immediately following the removal of VOP i from the rate buffer. Using the above definitions, b_i can be iteratively defined

$$b_0 = 64 \times \text{vbv\_occupancy} + (\text{size of configuration information}) - d_0$$

$$b_{i+1} = b_i + \int_{t_i}^{t_{i+1}} R_{voi}(t)dt - d_{i+1} \text{ for } i \geq 0$$

Since the first S-VOP in the elementary stream is regarded as the initial VOP (VOP 0) for basic sprite sequences in the above equation, the equation does not define the buffer occupancy for the I-VOP of basic sprite sequences. For defining the buffer occupancy, the I-VOP of a basic sprite sequence is divided into (N_{sp} + 395) / 396 *sections*, where N_{sp} is the number of macroblocks in the I-VOP. The *N*-th ($N \geq 0$) macroblock of the I-VOP (stuffing macroblocks are not included) in transmission order is included in the (*N* / 396)-th section. All the bits of the *k*-th ($0 \leq k < (N_{sp} + 395) / 396$) section are removed from the VBV buffer instantaneously at the section decoding time ts_k, where ts_k is the time when the last bit of this section enters the buffer. Thus the buffer occupancy immediately before the removal of section *k* is equal to the size in bits of this section, and the buffer occupancy immediately after the removal is 0. The initial section of the I-VOP includes all the bits from the beginning of the elementary stream to the end of the section. The last section includes all the stuffing bits added at the end of the I-VOP information.

© ISO/IEC 2004 – All rights reserved

9. The rate buffer model requires that the VBV buffer never overflow or underflow, that is

$$0 <= b_i \quad and \quad b_i + d_i <= B \quad for\ all\ i$$

For the initial I-VOP of basic sprite sequences, the VBV buffer occupancy immediately before the removal of a section shall not exceed *B*.

Real-valued arithmetic is used to compute b_i so that errors are not accumulated.

A coded VOP size must always be less than the VBV buffer size, i.e., $d_i < B$ for all i.

10. If the short video header is in use (i.e., when short_video_header = 1), then the parameter vbv_buffer_size is not present and the following conditions are required for VBV operation. The buffer is initially empty at the start of encoder operation (i.e., *t*=0 being at the time of the generation of the first video plane with short header), and its fullness is subsequently checked after each time interval of 1001/30000 seconds (i.e., at *t*=1001/30000, 2002/30000, etc.). If at least one complete VOP is in the buffer at the checking time, the data for the earliest VOP in bitstream order is removed. The buffer fullness after the removal of a VOP, b_i, shall be greater than or equal to zero and less than (4 *R*max 1001) / 30000 bits, where *R*max is the maximum bit rate in bits per second allowed within the profile and level. The number of bits used for coding any single VOP, d_i, shall not exceed *k* 16384 bits, where *k* = 4 for QCIF and Sub-QCIF, *k* = 16 for CIF, *k* = 32 for 4CIF, and *k* = 64 for 16CIF, unless a larger value of *k* is specified in the profile and level definition. Furthermore, the total buffer fullness at any time shall not exceed a value of *B* = *k* 16384 + (4 *R*max 1001) / 30000.

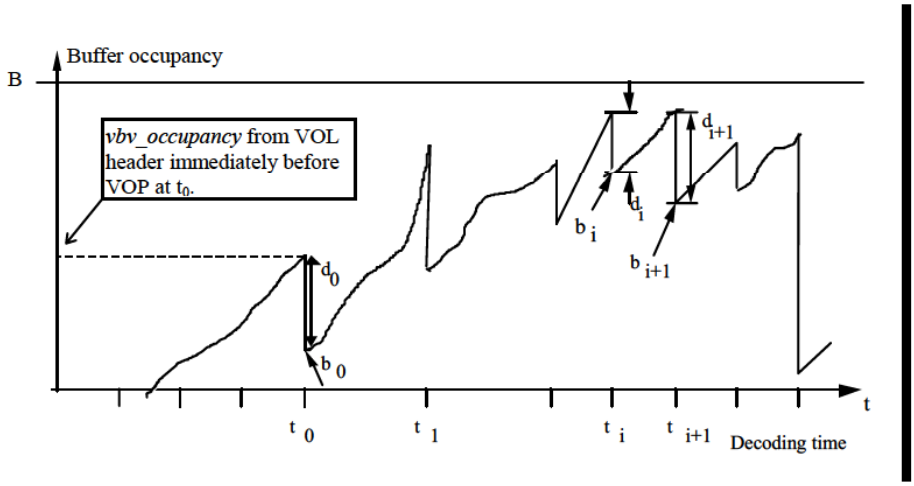


Figure D.2 — VBV Buffer occupancy

It is a requirement on the encoder to produce a bitstream which does not overflow or underflow the VBV buffer. This means the encoder must be designed to provide correct VBV operation for the range of values of $R_{vol,decoder}(t)$ over which the system will operate for delivery of the bitstream. A channel has constant delay if the encoder bitrate at time t when particular bit enters the channel, $R_{vol,encoder}(t)$ is equal to $R_{vol,decoder}(t + L)$, where the bit is received at (t + L) and L is constant. In the case of constant delay channels, the encoder can use its locally estimated $R_{vol,encoder}(t)$ to simulate the VBV occupancy and control the number of bits per VOP, d_i, in order to prevent overflow or underflow.

The VBV model assumes a constant delay channel. This allows the encoder to produce a bitstream which does not overflow or underflow the buffer using $R_{vol,encoder}(t)$ – note that $R_{vol}(t)$ is defined as $R_{vol,encoder}(t)$ in item 2 above.

© ISO/IEC 2004 – All rights reserved

D.3 Comparison between ISO/IEC 14496-2 VBV and the ISO/IEC 13818-2 VBV (Informative)

The ISO/IEC 13818-2 and ISO/IEC 14496-2 VBV models both specify that the rate buffer may not overflow or underflow and that coded pictures (VOPs) are removed from the buffer instantaneously. In both models a coded picture/VOP is defined to include all higher-level syntax immediately preceding the picture/VOP regardless of whether the separate configuration/elementary streams mode or the combined configuration/elementary streams mode is used for the carriage of ISO/IEC 14496-2 bitstreams (see subclause 6.2.1). It is assumed that both the configuration information and the elementary stream is input to a common rate buffer memory.

ISO/IEC 13818-2:2000 has a constant frame period (although the bitstream can contain both frame and field pictures and frame pictures can use explicit 2:3 pulldown via the repeat_first_field flag). In ISO/IEC 14496 terms, this frame rate would be the output of the compositor (the ISO/IEC 13818 terminology is the output of the display process that is not defined normatively by ISO/IEC 13818). This output frame rate together with the ISO/IEC 13818-2 picture_structure and repeat_first_field flag precisely defines the time intervals between consecutive decoded picture (either frames or fields) passed between the decoding process and the display process.

ISO/IEC 13818-2 bitstream may contains B pictures (we assume ISO/IEC 13818-2 low_delay = 0). This means the coding order and display order of pictures is different (since both reference pictures used by a B picture must precede the B picture in coding order). The ISO/IEC 13818-2 VBV (and ISO/IEC 13818-1 systems T-STD) specifies that a B picture is decoded and presented (instantaneously) at the same time and the reference pictures are re-ordered to make this possible. This is the same reordering model specified in item 7 above.

A model ISO/IEC 14496-2 decoder using its VBV buffer model emulates a model decoder using the ISO/IEC 13838-2 VBV buffer model if the VOP time stamps given by vop_time_increment and the cumulative modulo_time_base agree with the sequence of ISO/IEC 13818-2 picture presentation times. We assume here that both coded picture/VOPs use the common subset of both standards (frame structured pictures and no 3:2 pulldown on the decoder, i.e., repeat_first_field = 0). For example, if the ISO/IEC 14496-2 sequence is coded at 29.97Hz (the NTSC picture rate), vop_time_increment_resolution will be 30000 and the change in vop_time_increment between consecutive VOPs in display order will be 1001 because picture skipping is not permitted in ISO/IEC 13818-2:2000 (when low_delay = 0).

D.4 Video Complexity Model Definition

The ISO/IEC 14496-2 video complexity verifier (VCV) is an algorithm for checking a bitstream to verify that the amount of processing capability required in a decoder is less than the stated complexity measure in macroblocks per second. This model applies to all macroblocks in a MPEG-4 visual bitstream.

A separate VCV buffer accumulates the boundary macroblocks of all VOLs in a combined MPEG-4 visual bitstream. The boundary MB VCV has a separate decoding rate (specified in Annex N) but it is subject to the same maximum value used in the VCV accumulating the total number of macroblocks. Boundary macroblocks (containing coded shape information which is not totally transparent or totally opaque) are included in both the VCV model and in the boundary VCV model. For S-VOPs in low-latency sprite sequences, the number of boundary macroblocks used in the boundary VCV model is defined as the number of boundary macroblocks with coded shape information included in sprite_shape_texture() (see subclauses 6.2.5.4 and 7.8.3).

The VCV applies to natural video coded as a combination of I, P, B and S-VOPs, while a separate VCV applies to still texture. Face animation and mesh objects are not constrained by this model. For basic and low-latency sprite sequences, an equivalent number of macroblocks is defined for each S-VOP.

The coded video bitstream shall be constrained to comply with the requirements of the VCV defined as follows:

1. A vcv_buffer_size is defined as the maximum number of MBs which can be contained in the VCV-buffer and is specified in Annex N. These MBs are consumed by the decoder at vcv_decoder_rate or H (in MB/s), as specified in Annex N for each profile and level. A vcv_decoder_latency or L is defined as the time needed for the decoder to decode a full VCV-buffer (vcv_buffer_size MBs) with the vcv_decoder_rate (MB/s). Thus the relation vcv_buffer_size = H*L holds. The VCV-buffer is initially empty at the start of decoding.

© ISO/IEC 2004 – All rights reserved

2. Let M_i be number of macroblocks in each VOP.

   For S-VOPs in basic or low-latency sprite sequences, M_i is $MB_{S\text{-}VOP}$. The hypothetical number of macroblocks in an S-VOP, $MB_{S\text{-}VOP}$, for the usage in the VCV model is defined as follows:

   $$MB_{S\text{-}VOP} = (MB_{rcn} > MB_{ref} \; ? \; MB_{rcn} : MB_{ref}) + MB_{lls},$$

   where MB_{rcn} is the number of macroblocks (including opaque, boundary, and transparent macroblocks) fully or partially included in the bounding rectangle of the reconstructed VOP, MB_{ref} is he number macroblocks included in the support area in the sprite, and MB_{lls} is the number of updated macroblocks in the sprite. MB_{ref} is defined as follows:

   $MB_{ref} = MB_{rcn}$, when no_of_sprite_warping_points == 0 or 1,

   $= (MB_{rcn} \times ((i_1'-i_0')^2 + (j_1'-j_0')^2))/(W^2 s^2)$, when no_of_sprite_warping_points == 2,

   $= (MB_{rcn} \times |(i_1' - i_0')(j_2' - j_0') - (i_2' - i_0')(j_1' - j_0')|) / (W H s^2)$, when no_of_sprite_warping_points == 3, and

   $= (MB_{rcn} \times (|(i_1' - i_0')(j_2' - j_0') - (i_2' - i_0')(j_1' - j_0')| + |(i_1' - i_3')(j_2' - j_3') - (i_2' - i_3')(j_1' - j_3')|))/(2 W H s^2)$, when no_of_sprite_warping_points == 4,

   where the definition of W, H, s, i_0', j_0', i_1', j_1', i_2', j_2', i_3', and j_3' is described in subclause 7.8.4. MB_{lls} denotes the number of macroblocks with at least one coded DCT coefficient included in sprite_shape_texture() (see subclauses 6.2.5.4 and 7.8.3). The value of MB_{lls} is 0 when low_latency_sprite_enable == 0.

   For the VOP with vop_reduced_resolution flag equal to 1, M_i is defined as the number of macroblocks in the $VOP_i \times 4$.

3. At time t_i, M_i is added to the VCV-buffer occupancy, v(t), where t_i is the VOP decoding time calculated according to subclause D.2 for each VOL. For the I-VOP in a basic sprite sequence, the number of macroblocks included in section k (see item 8 of subclause D.2 for the definition of a section), MS_k, is added to the VCV buffer occupancy at ts_k, where ts_k is the section decoding time defined in subclause D.2. The occupancy of the VCV buffer decreases linearly at rate H (defined in Table N.1) until the occupancy is zero or until the time reaches t_{next} where t_{next} is the earliest VOP or section decoding time greater than t_i or ts_k for some VOP in the visual bitstream. If the VCV occupancy becomes zero (the VCV decoder is idle), then the VCV remains idle until t_{next}. See Figure D.3.

4. The interval of time where VOP i is being decoded extends from s_i to e_i which are defined as

   $$s_i = t_i + \text{Max}[v(t_i)/H, v_B(t_i)/H_B]$$
   $$e_i = t_i + \text{Max}[(v(t_i) + M_i)/H, (v_B(t_i) + M_{Bi})/H_B]$$

   $v(t_i)$ is the VCV occupancy before the macroblocks representing VOP i, M_i, are added to v(t),
   H is the VCV decoder rate (see Table N.1),
   $v_B(t_i)$ is the boundary MB VCV occupancy before the boundary macroblocks representing VOP i, M_{Bi}, are added to $v_B(t)$, and
   H_B is the VCV boundary MB decoder rate (see Table N.1).

   For the I-VOP in a basic sprite sequence, the interval of time where section k is being decoded extends from ss_k to es_k which are defined as

   $$ss_k = ts_k + \text{Max}[v(ts_k)/H, v_B(ts_k)/H_B]$$
   $$es_k = ts_k + \text{Max}[(v(ts_k) + MS_k)/H, (v_B(ts_k) + MS_{Bk})/H_B]$$

   $v(ts_k)$ is the VCV occupancy before the number of macroblocks in section k, MS_k, is added to the VCV occupancy, and
   $v_B(ts_k)$ is the boundary MB VCV occupancy before the boundary macroblocks representing section k, MS_{Bk}, are added to $v_B(ts_k)$.

© ISO/IEC 2004 – All rights reserved

5. To decode a VOL, the decoding of each VOP i must be complete by τ_i + L (composition time plus the latency of the VCV decoding process). The latency L of the VCV decoding process is constant for all VOPs. This constraint does not apply to the I-VOP in a basic sprite sequence.

6. The complexity model allows the VCV buffer to underflow, in which case the decoder remains idle, and the VCV buffer occupancy, v(t), remains during the idle period.

D.5 Video Reference Memory Model Definition

The ISO/IEC 14496-2 video memory verifier (VMV) is an algorithm for checking a bitstream to verify that the amount of pixel memory required in a decoder is less than the maximum VMV buffer size (in units of MBs) specified in Annex N for each profile and level. If a visual bitstream is composed of multiple VOs each with one or more VOLs, the VMV models the memory requirements of all VOLs (this model assumes a shared memory space shared by all VOLs of all VOs).

The VMV applies to natural video coded as a combination of I, P, B, S-VOPs, and still texture. Face animation, mesh objects and I-VOPs in basic sprite sequences are not constrained by this model.

The coded video bitstream shall be constrained to comply with the requirements of the VMV defined as follows:

1. The reference memory is initially empty. It is filled with the decoded data as each macroblock is decoded.

2. For I, P, B, and S(GMC)-VOPs, the amount of reference memory required for the decoding of the ith VOP is defined as the number of macroblocks in the VOP, M_i. For the VOP with vop_reduced_resolution flag equal to 1, M_i is defined as the number of macroblocks in the $VOP_i \times 4$. This memory is consumed at the same constant rate of

$$\frac{M_i}{e_i - s_i} \text{ (MB/s)}$$

   as the decoding process occurs.

   For S-VOPs in basic or low-latency sprite sequences, the amount of reference memory required for the decoding of the VOP is defined as the number of macroblocks in the reconstructed VOP, MB_{rcn} (see item 2 of subclause D.4 for the definition of MB_{rcn}). This memory is consumed at the constant rate of

$$\frac{MB_{rcn}}{e_i - s_i} \text{ (MB/s)}$$

   as the decoding process occurs. The memory used for storing the sprite is ignored by the VMV.

   The decoding duration of VOP i, T_i , is identical in the VCV and VMV models and starts at s_i and ends at e_i as defined in subclause D.4 above.

3. At the composition time (or presentation time in a no-compositor decoder) plus VCV latency, τ_i + L, of an I, P, or S(GMC)-VOP the total memory allocated to the previous I, P, or S(GMC)-VOP in the decoding order is released instantaneously.

4. At the composition time (or presentation time in a no-compositor decoder) plus VCV latency, τ_i + L, of a B-VOP, or an S-VOP in a basic or low latency sprite sequence, the total memory allocated to that B-VOP or S-VOP is released instantaneously.

5. In case of scalability coding, the release of VMV buffer is managed by each VOL independently. In addition to this constraint, the time of memory release for any VOP in enhancement layer follows definition 6 below.

© ISO/IEC 2004 – All rights reserved

6. At the composition time (or presentation time in a no-compositor decoder) plus VCV latency, τ_i +L, of a I-, P- or B-VOP of enhancement layer, the total memory allocated to the previous I-, P- or B-VOP in the decoding order is released instantaneously.

7. The reference memory model requires that the VMV buffer never exceeds the vmv_buffer_size obtained from the Table N.1. The buffer occupancy of the video memory verifier is shown in Figure D.3.

D.6 Interaction between VBV, VCV and VMV (informative)

The VBV model defines when the bitstream is available for the decoding and is removed from the buffer. The VCV model defines the speed at which the macroblocks are decoded. The reference memory model defines the amount of reference memory that is consumed and released. Obviously, it is advantageous for the video decoder to decode as far in advance as possible. However, this is constrained by the VBV and the VMV. The decoder can only start decoding if the bits are available for decoding. At the same time as the decoder decodes the bitstream, it generates macroblocks which consumes the reference memory. So if it decodes too fast it will overflow the reference memory.

On the other hand if the decoder start decoding too late, then it will not be able to complete the decoding in time and the bitstream will be removed from the VBV before it could be processed. Similarly the reference memory required for the prediction of the current VOP may also be removed from the VMV.

Therefore, the encoder will have to simulate the VBV, VCV and VMV in order to produce a bitstream compliant with the intended profile and level. If the simulated decoder VBV becomes too full, the encoder should produce more bits (decrease the coarseness of quantisation) or generate stuffing bits to prevent overflow. If the VBV occupancy becomes low, fewer bits are to produced. If the VCV approaches its limit, the encoder must allow more time for the decoder to produce its output. This can be done, for example, by increasing the interval of time between VOPs If the VMV becomes too full, decoder memory usage must be decreased. Using smaller VOPs (fewer macroblocks) and avoiding B-VOPs reduce the decoder's memory requirements.

D.7 Video Presentation Model Definition (informative)

The video presentation verifier (VPV) is not part of this specification. It is an algorithm for checking a bitstream to verify that the amount of presentation buffer required in a decoder is less than a given amount of memory in units of MB. It is also used to constraint the speed of the compositor in terms of maximum number of MB/sec.

The VPV operates in the same manner as the VCV:

1. At the composition time plus VCV decoder latency of the i^{th} VOP, τ_i, + L, the VOP is placed in the presentation buffer.

2. The data in the presentation buffer is consumed by the compositor at a rate equivalent to the maximum number of MB/sec.

3. At τ_i + L + compositor_latency the VOP should be composited.

4. The presentation memory model requires that the VPV buffer never overflows

© ISO/IEC 2004 – All rights reserved

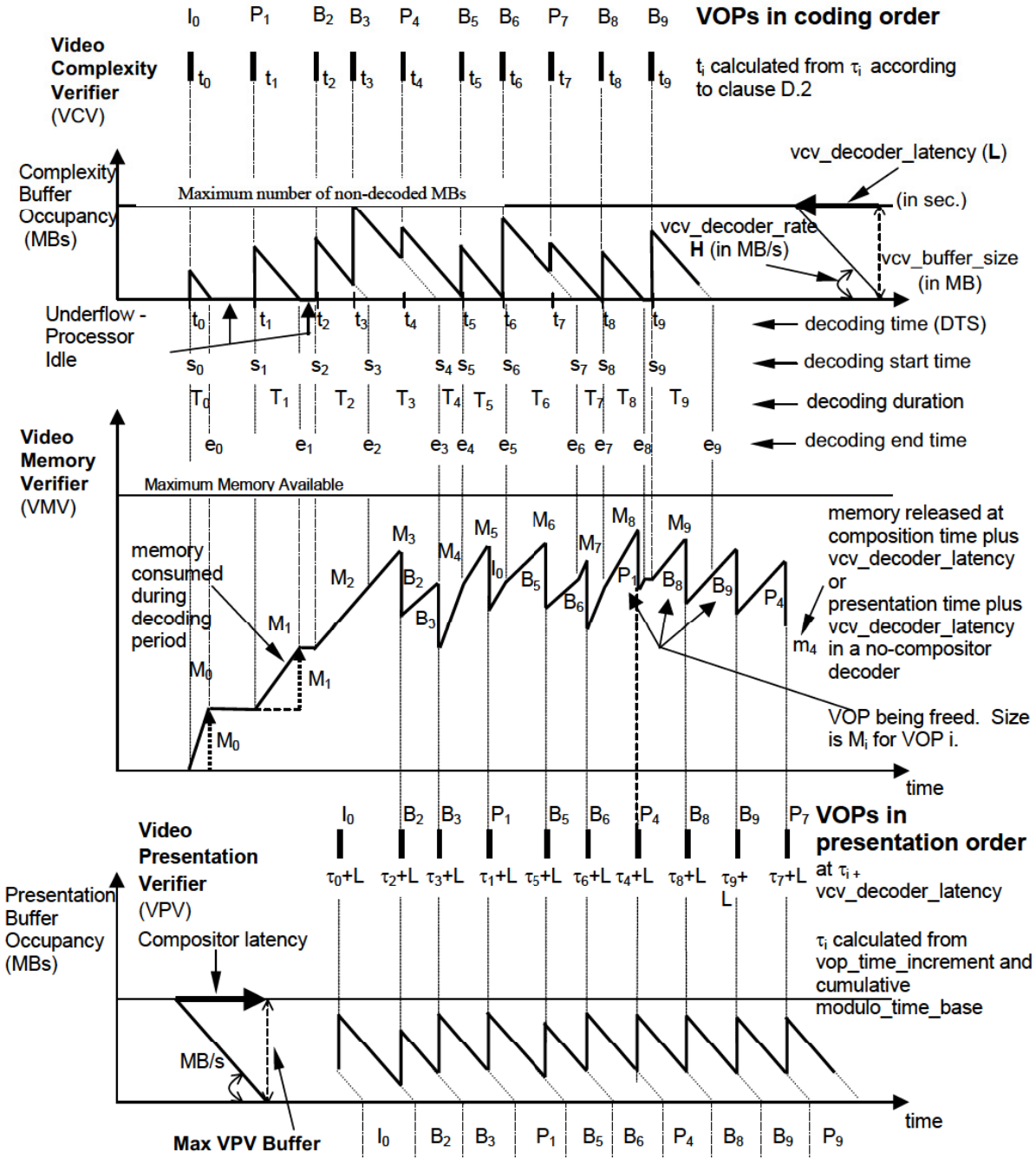


Figure D.3 — VCV, VMV and VPV buffer occupancy (video_object_layer_shape == "rectangular")

© ISO/IEC 2004 – All rights reserved

Annex E
(informative)

Features supported by the algorithm

E.1 Error resilience

E.1.1 Resynchronization

Resynchronization tools, as the name implies, attempt to enable resynchronization between the decoder and the bitstream after a residual error or errors have been detected. Generally, the data between the synchronization point prior to the error and the first point where synchronization is reestablished, is discarded. If the resynchronization approach is effective at localizing the amount of data discarded by the decoder, then the ability of other types of tools that recover data and/or conceal the effects of errors is greatly enhanced.

The resynchronization approach adopted by ISO/IEC 14496, referred to as a packet approach, is similar to the Group of Blocks (GOBs) structure utilized by the ITU-T Recommendations H.261 and H.263. In these standards a GOB is defined as one or more rows of macroblocks (MB). At the start of a new GOB, information called a GOB header is placed within the bitstream. This header information contains a GOB start code, which is different from a picture start code, and allows the decoder to locate this GOB. Furthermore, the GOB header contains information which allows the decoding process to be restarted (i.e., resynchronize the decoder to the bitstream and reset all coded data that is predicted).

The GOB approach to resynchronization is based on spatial resynchronization. That is, once a particular macroblock location is reached in the encoding process, a resynchronization marker is inserted into the bitstream. A potential problem with this approach is that since the encoding process is variable rate, these resynchronization markers will most likely be unevenly spaced throughout the bitstream. Therefore, certain portions of the scene, such as high motion areas, will be more susceptible to errors, which will also be more difficult to conceal.

The video packet approach adopted by ISO/IEC 14496, is based on providing periodic resynchronization markers throughout the bitstream. In other words, the length of the video packets are not based on the number of macroblocks, but instead on the number of bits contained in that packet. If the number of bits contained in the current video packet exceeds a predetermined threshold, then a new video packet is created at the start of the next macroblock.

Resync Marker	macroblock_number	quant_scale	HEC	Macroblock Data	Resync Marker

Figure E.1 — Error Resilient Video Packet

In Figure E.1, a typical video packet is described. A resynchronization marker is used to distinguish the start of a new video packet. This marker is distinguishable from all possible VLC code words as well as the VOP start code. Header information is also provided at the start of a video packet. Contained in this header is the information necessary to restart the decoding process and includes: the macroblock address (number) of the first macroblock contained in this packet and the quantiser scale (quant_scale) necessary to decode that first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantiser scale allows the differential decoding process to be resynchronized. Following the quant_scale is the Header Extension Code (HEC). As the name implies, HEC is a single bit used to indicate whether additional information will be available in this header. If the HEC is equal to one then the following additional information is available in this packet header: modulo_time_base, vop_time_increment, vop_coding_type, intra_dc_vlc_thr, vop_fcode_forward, vop_fcode_backward.

© ISO/IEC 2004 – All rights reserved

The Header Extension Code makes each video packet (VP) possible to be decoded independently, when its value is equal to 1. The necessary information to decode the VP is included in the header extension code field, if the HEC is equal to 1.

If the VOP header information is corrupted by the transmission error, they can be corrected by the HEC information. The decoder can detect the error in the VOP header, if the decoded information is inconsistent with its semantics. For example, because it is prohibited that the values of the vop_fcode_forward and vop_fcode_backward are set to "0", if they are 0, the decoder can detect the error in the fcode information. In such a case, the decoder can correct the value by using the HEC information of the next VP.

When utilizing the error resilience tools within ISO/IEC 14496, some of the compression efficiency tools are modified. For example, all predictively encoded information must be confined within a video packet so as to prevent the propagation of errors. In other words, when predicting (i.e., AC/DC prediction and motion vector prediction) a video packet boundary is treated like a VOP boundary.

In conjunction with the video packet approach to resynchronization, a second method called fixed interval synchronization has also been adopted by ISO/IEC 14496. This method requires that VOP start codes and resynchronization markers (i.e., the start of a video packet) appear only at legal fixed interval locations in the bitstream. This helps to avoid the problems associated with start codes emulations. That is, when errors are present in a bitstream it is possible for these errors to emulate a VOP start code. In this case, when fixed interval synchronization is utilized the decoder is only required to search for a VOP start code at the beginning of each fixed interval. The fixed interval synchronization method extends this approach to be any predetermined interval.

The fixed interval synchronization is achieved by first inserting a bit with the value 0 and then, if necessary, inserting bits with the value 1 before the start code and the resync marker. The video decoder can determine if errors are injured in a video packet by detecting the incorrect number of the stuffing bits. (e.g. eight or more 1's are followed after 0 at the last part of a video packet, or the remaining bit pattern is not "011...")

E.1.2 Data Partitioning

Error concealment is an extremely important component of any error robust video codec. Similar to the error resilience tools discussed above, the effectiveness of an error concealment strategy is highly dependent on the performance of the resynchronization scheme. Basically, if the resynchronization method can effectively localize the error then the error concealment problem becomes much more tractable. For low bitrate, low delay applications the current resynchronization scheme provides very acceptable results with a simple concealment strategy, such as copying blocks from the previous frame.

In recognizing the need to provide enhanced concealment capabilities, the Video Group has developed an additional error resilient mode that further improves the ability of the decoder to localize an error. Specifically, this approach utilizes data partitioning. This data partitioning is achieved by separating the motion and macroblock header information away from the texture information. This approach requires that a second resynchronization marker be inserted between motion and texture information. Data partitioning, like the use of RVLCs, is signaled to the decoder in the VOL. Figure E-2 illustrates the syntactic structure of the data partitioning mode. If the texture information is lost, this approach utilizes the motion information to conceal these errors. That is, due to the errors the texture information is discarded, while the motion is used to motion compensate the previously decoded VOP.

Resync Marker	macroblock_number	quant_scale	HEC	Motion &Header Information	Motion Marker	Texture Information	Resync Marker

Figure E.2 — Data Partitioning

E.1.3 Reversible VLC

Reversible Variable Length Codes (RVLC) are designed such that they can be instantaneously decoded both in forward and reverse directions. A part of a bitstream which cannot be decoded in the forward direction due to the presence of errors can often be decoded in the backward direction. This is illustrated in Figure E.3. Therefore number of discarded bits can be reduced. RVLC is applied only to Tcoef coding

© ISO/IEC 2004 – All rights reserved

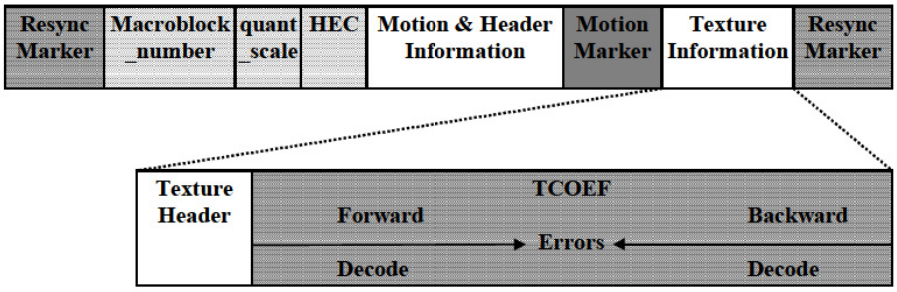


Figure E.3 — Reversible VLC

E.1.4 Decoder Operation

E.1.4.1 General Error Detection

An illegal VLC is received.

A semantic error is detected.

More than 64 DCT coefficients are decoded in a block.

Inconsistent resyncronization header information (i.e., QP out of range, MBA(k)<MBA(k-1),etc.)

E.1.4.2 Resynchronization

When an error is detected in the bitstream, the decoder should resynchronize at the next suitable resynchronization point(vop_start_code or resync_marker).

After that, it can be determined by detecting the incorrect number of the stuffing bits whether or not errors are injured in a video packet. If eight or more 1's are followed after 0 at the last part of a video packet, or the remaining bit pattern is not "011…", it means there is any error in this video packet.

If the VOP start code is corrupted by the transmission error and the frame synchronization is lost, the decoder may establish the resynchronization by using the HEC information. The decoder compares the vop_time_increment in the VOP header with one in the HEC field. If they are not same, the decoder may find that the current VOP start code is corrupted by the error. In this case, there must not be the error in the both vop_time_increments. The simple method is to check whether the vop_time_increment is mutilple of frame interval of the original source format (NTSC, PAL and so on). Therefore, it is expected that the number of the vop_time_increment is as many as possible. As this check method does not always detect the error, this is the auxiliary technique.

Missing blocks may be replaced with the same block from the previous frame.

E.1.4.3 Data Partitioning

E.1.4.4 Reversible VLC

This subclause describes a decoding methodology for Reversible Variable Length Codes (RVLC) when errors in the video bitstream are detected during the decoding process. This particular decoding methodology was developed during the RVLC core experiment process.

E.1.4.4.1 Process for detecting errors for both forward and backward decoding

Errors are present in the following cases:

(1) An illegal RVLC is found, where an illegal RVLC is defined as follows:

A codeword whose pattern is not listed in the RVLC table (e.g. 169 codeword patterns and escape codes).

Escape coding is used (i.e., a legal codeword is not available in the RVLC table) and the decoded value for LEVEL is zero.

© ISO/IEC 2004 – All rights reserved

The second escape code is incorrect (e.g. codeword is not “00000” or “00001” for forward decoding, and/or is not “00001” for backward decoding).

There is a decoded value of FLC part using escape codes (e.g. LAST, RUN, LEVEL) in the RVLC table.

An incorrect number of stuffing bits for byte alignment (e.g. eight or more 1s follow 0 at the last part of a Video packet (VP), or the remaining bit pattern is not “0111...” after decoding process is finished).

(2) More than 64 DCT coefficients are decoded in a block.

E.1.4.4.2 Decoding information

The bitstream is decoded in the forward direction first. If no errors are detected, the bitstream is assumed to be valid and the decoding process is finished for that video packet. If an error is detected however, two-way decoding is applied. The following strategies for determining which bits to discard are used. These strategies are described using the figures given below along with the following definitions:

L : Total number of bits for DCT coefficients part in a VP.
N : Total number of macroblocks (MBs) in a VP.
L1 : Number of bits which can be decoded in a forward decoding.
L2 : Number of bits which can be decoded in a backward decoding.
N1 : Number of MBs which can be completely decoded in a forward decoding.
(0 <= N1 <= (N-1))
N2 : Number of MBs which can be completely decoded in a backward decoding.
(0 <= N2 <= (N-1))
f_mb(S) : Number of decoded MBs when S bits can be decoded in a forward direction.
(Equal to or more than one bit can be decoded in a MB, f_mb(S) counter is up.)
b_mb(S) : Number of decoded MBs when S bits can be decoded in a backward direction.
T : Threshold (90 is used now).

E.1.4.4.2.1 Strategies for decoding RVLC

(1) Strategy 1 : *L1+L2 < L* and *N1+N2 < N*

MBs of *f_mb(L1-T)* from the beginning and MBs of *b_mb(L2-T)* from the end are used. In the figure below, the MBs of the dark part are discarded.

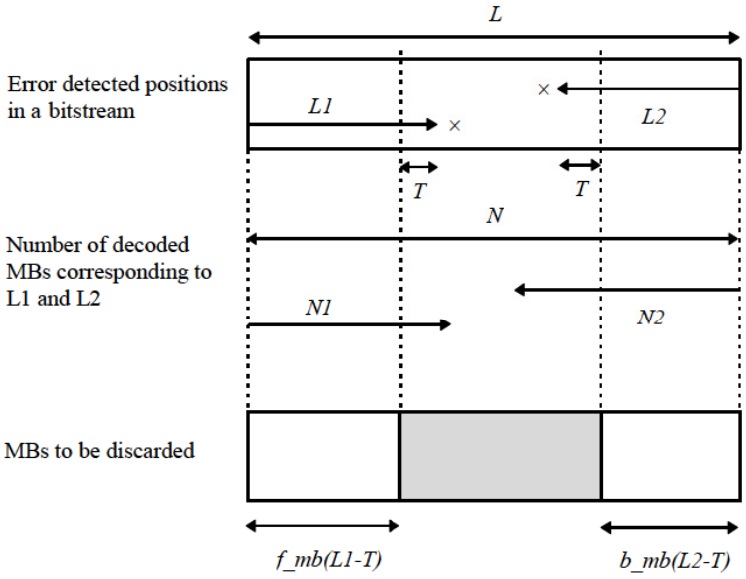


Figure E.4 — RVLC decoding; Strategy 1

© ISO/IEC 2004 – All rights reserved

(2) Strategy 2 : *L1+L2 < L* and *N1+N2 >= N*

MBs of *N-N2-1* from the beginning and MBs of *N-N1-1* from the end are used. MBs of the dark part are discarded.

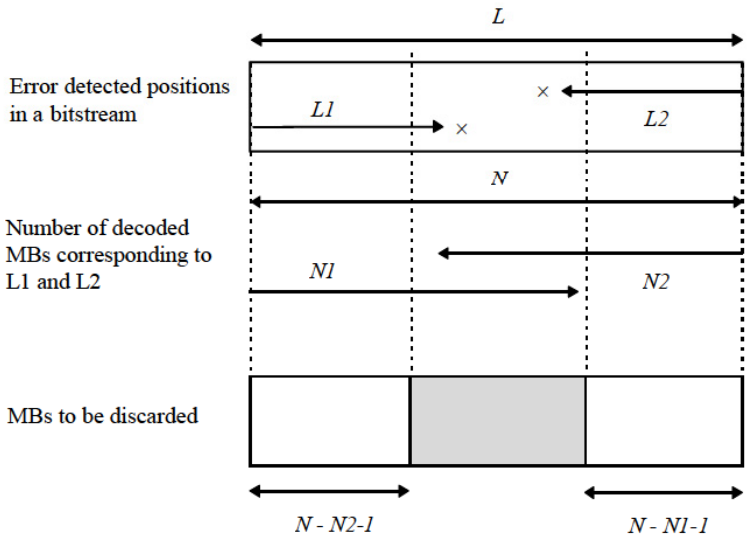


Figure E.5 — RVLC decoding; Strategy 2

(3) Strategy 3 : *L1+L2 >= L* and *N1+N2 < N*

MBs of *N-b_mb(L2)* from the beginning and MBs of *N-f_mb(L1)* from the end are used. MBs of the dark part are discarded.

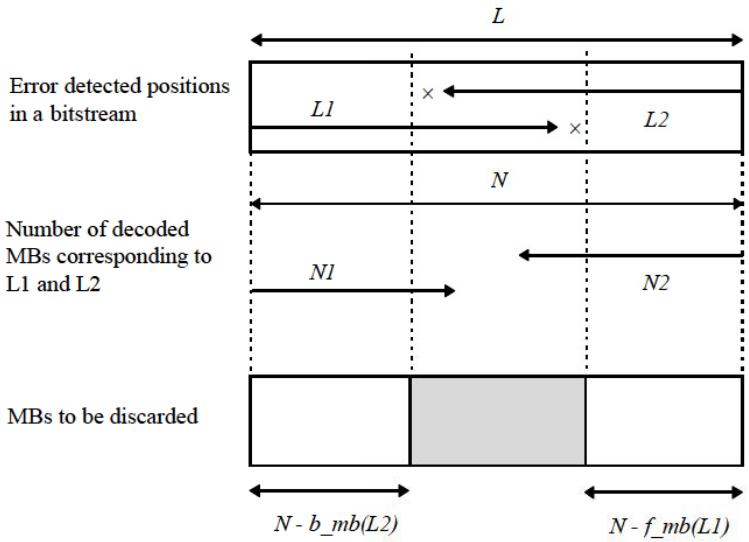


Figure E.6 — RVLC decoding; Strategy 3

© ISO/IEC 2004 – All rights reserved

(4) Strategy 4 : *L1+L2 >= L* and *N1+N2 >= N*

MBs of *min{N-b_mb(L2), N-N2-1}* from the beginning and MBs of *min{N-f_mb(L1), N-N1-1}* from the end are used. MBs of the dark part are discarded.

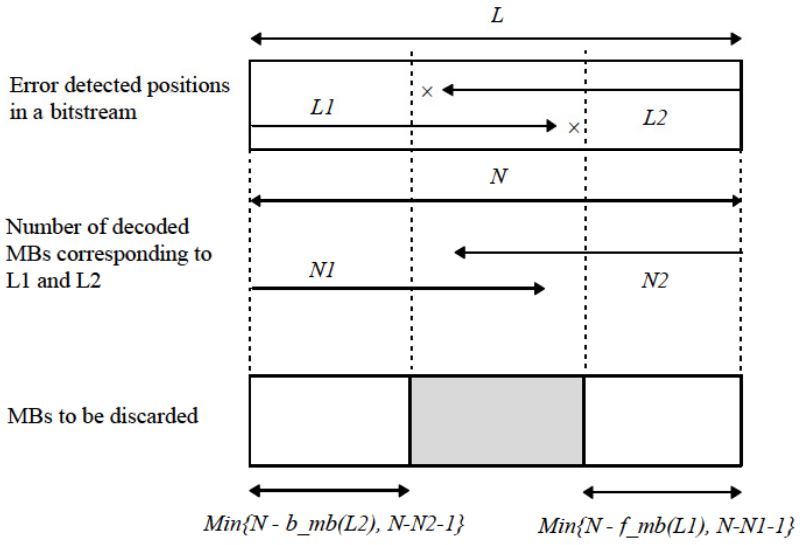


Figure E.7 — RVLC decoding; Strategy 4

E.1.4.4.2.2 INTRA MBs within a bitstream

In the above strategies (**Strategy 1** - **Strategy 4**), INTRA MBs are discarded even though they could have been decoded. An example of such a case is shown below.

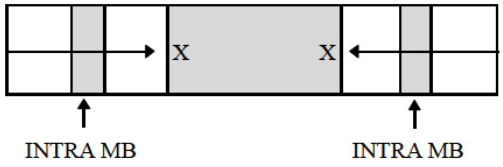


Figure E.8 — Intra MB discarding

Although these intra MBs are thought to be correct, the result of displaying an Intra MB that does contain an error can substantially degrade the quality of the video. Therefore, when a video packet is determined to contain errors, all Intra MBs are not displayed, but instead concealed.

E.1.5 Adaptive Intra Refresh (AIR) Method

The AIR is the technique of the intra refresh method for the error resilience. In the AIR, motion area is encoded frequently in Intra mode. Therefore, it is possible to recover the corrupted motion area quickly.

NOTE This informative technique is used for the rectangular VOP. It was designed in order to extract the motion area from the rectangular VOP at the encoder and encode it in intra mode frequently. Therefore, the quality of this area that is corrupted by the transmission error can be recovered quickly. As the objects are extracted in advance in the arbitrary shape coding mode, it is possible to obtain the same performance by using the conventional cyclic intra refresh within the arbitrary shape.

© ISO/IEC 2004 – All rights reserved

The method of the "AIR"

The number of Intra MBs in a VOP is fixed and pre-determined. It depends on bitrates and frame rate and so on.

The encoder estimates motion of each MB and the only motion area is encoded in Intra mode. The results of the estimation are recorded to the **Refresh Map** MB by MB. The encoder refers to the Refresh Map and decides to encode current MB in Intra mode or not. The estimation of motion is performed by the comparison between SAD and SAD_th. SAD is the Sum of the Absolute Differential value between the current MB and the MB in same location of the previous VOP. The SAD has been already calculated in the Motion Estimation part. Therefore, additional calculation for the AIR is not needed. SAD_th is the threshold value. If the SAD of the current MB is larger than the SAD_th, this MB is regarded as motion area. Once the MB is regarded as motion area, it is regarded as motion area until it is encoded in Intra mode predetermined times. The predetermined value is recorded to the Refresh Map. (*See* Figure E.9. *In this figure, predetermined value is "1" as an example*)

The holizontal scan is used to determine the MBs to be encoded in Intra mode within the moving area (*see* Figure E.10).

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0
0	0	0	1	1	1	1	0	1	0	0
0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	1	1	0
0	0	0	0	0	0	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0

Figure E.9 — Refresh Map for QCIF

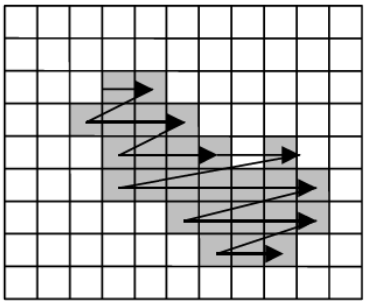


Figure E.10 — Scan order for the Adaptive Refresh

The processing of the "AIR"

The following is the explanation of the processing of AIR (see Figure E.11). The fixed number of the Intra MB in a VOP should be determined in advance. Here, it is set to "2" as an example.

[1] 1st VOP ([a]~[b] in Figure E.11)
The all MBs in the 1st VOP are encoded in Intra mode [a]. The Refresh Map is set to "0", because there is no previous VOP [b].

[2] 2nd VOP ([c] ~ [f])
The 2nd VOP is encoded as P-VOP. Intra refresh is not performed in this VOP, because all values in the Refresh Map is zero yet ([c] and [d]). The encoder estimates motion of each MB. If the SAD for current MB is larger than the SAD_th, it is regarded as motion area (hatched area in Figure E.11 [e]). And the Refresh Map is updated [f].

© ISO/IEC 2004 – All rights reserved

[3] 3rd VOP ([g] ~ [k])
When the 3rd VOP is encoded, the encoder refers to the Refresh Map [g]. If the current MB is the target of the Intra refresh, it is encoded in Intra mode [h]. The value of the MB in Refresh Map is decreased by 1 [i]. If the decreased value is 0, this MB is not regarded as motion area. After this, the processing is as same as the 2nd VOP [j]~[k].

[4] 4th VOP ([l]~[p])
It is as same as 3rd VOP

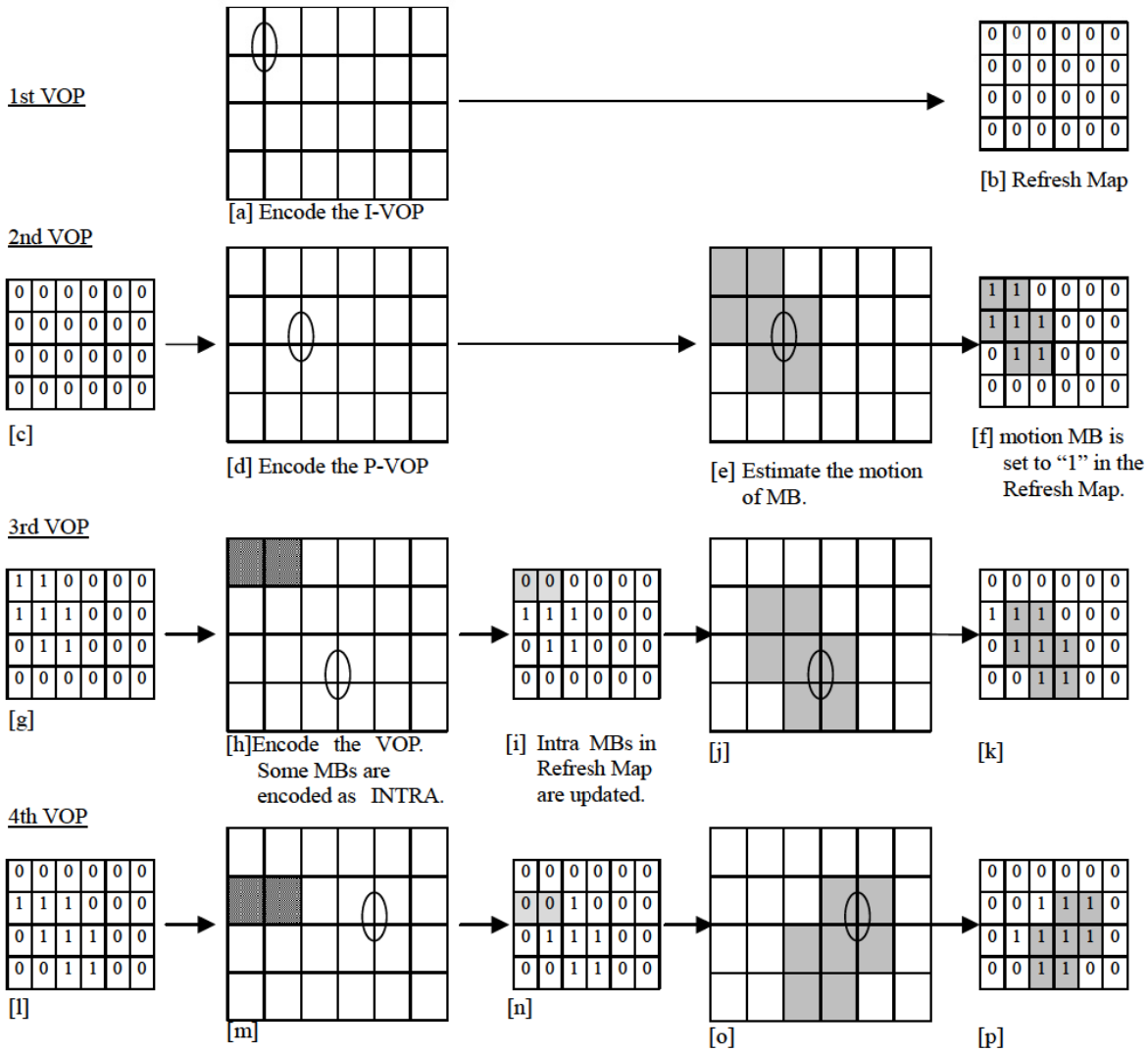


Figure E.11 — Explanation of AIR

E.1.6 NEWPRED

Recovery from temporal error propagation is an indispensable component of any error robust video communication system. Errors introduced during transmission will lead to frame-store mismatch between the encoder and the decoder, which will continue until some intra refresh occurs. When an upstream does not exist, cyclic intra refresh or adaptive intra refresh are available. In the case where the upstream exists from the decoder to the encoder, NEWPRED or demand intra refresh can be used. This subclause is presenting decoding process when newpred_enable = '1'.

© ISO/IEC 2004 – All rights reserved

E.1.6.1 Introduction to NEWPRED mode

NEWPRED mode is one of the error resilience tools. It uses upstream messages to indicate which NEWPRED segments (NP segment) are correctly decoded and which NP segments are erroneously decoded. The encoder, which receives an upstream message, uses only correctly decoded part for prediction in an inter-frame coding. This prevents a temporal error propagation without the insertion of Intra coded MBs and improves the picture quality in the erroneous environments.

E.1.6.2 Upstream channel

A special logical channel is opened for the upstream from the decoder to the encoder. The syntax of the upstream message is described in subclause 6.2.12.

It requires an external framing mechanism for synchronization of the messages within the upstream, as the back-channel syntax defined herein contains no synchronization flag words. This synchronization is provided by the lower layer such as ISO/IEC 14496-1:2001 (MPEG-4 Systems).

E.1.6.3 Definition of NEWPRED

NEWPRED is a technique in which the reference picture for inter-frame coding is replaced adaptively according to the upstream messaging from the decoder. One VOP or one Video Packet is treated as one NP segment. NP segment is indicated in newpred_segment_type in VOL header. The reference picture is replaced independently in each NP segment. When Video Packet is NP segment, any shape of NP segment shall be restricted to be rectangular.

The encoder has some kinds of mode for selection of the reference picture, and the strategy used by the encoder to select the reference picture is beyond the scope of this standard.

The decoder, however, takes the same action regardless of the modes. The decoder uses the NP segment which is indicated in the vop_id_for_prediction field as a reference picture if it exists in the picture buffer memory, and returns ether NP_ACK or NP_NACK message. This clause describes the processes of the decoder.

E.1.6.4 Decoder definition including upstream signaling

The decoder decodes a NP segment with reference to the previously decoded segment whose vop_id is vop_id_for_prediction as a referenced segment for inter-frame decoding. If vop_id_for_prediction_indication is '0', the decoder refers to the immediately previous decoded segment, or it refers to no segment. It shall not, however, refer to any segment which is decoded before the last I_VOP.

The decoder returns the upstream message to the encoder based on the upstream syntax defined in 7.14.2. The requested_upstream_type in VOL layer indicates which type of upstream message is needed by the encoder. The decoder returns only requested type of upstream message. The decoder, however, does not have to return any upstream message for B-VOP, because B-VOP is never referred by other VOP after that. After the received NP segment is correctly decoded, the ACK message of the NP segment is returned. When any error is detected during the decoding process or the error indication is delivered from the lower system layer, the decoder returns a NP_NACK message with the requested_vop_id_for_prediction which is the vop_id of the last correctly decoded NP segment of the same location of the current NP segment in the buffer memory. The error event includes the case that the NP segment whose vop_id is vop_id_for_prediction does not exist in the decoder. This occurs when the NP segment cannot be successfully decoded due to the previous errors. The vop_id field of the upstream message is copied from the corresponding field of the downstream message when the reliable vop_id is available. Otherwise, it may happen in the case of NP_NACK, vop_id is incremented by 1 from the reliable vop_id of the previously received NP segment of the same location of the current NP segment.

The decoder adds the correctly decoded NP segment into the buffer memory. If any errors are detected in the decoding process or the error indication is delivered from the lower system layer, the NP segment may not be added into the buffer memory. In the case of the buffer memory overflow, the oldest NP segment of the same location except for the segment which is referenced in the preceding encoding, may be removed.

© ISO/IEC 2004 – All rights reserved

E.1.6.5 Additional buffer memory for reference picture

When the NP_ACK message is requested by the encoder, the decoder may need several number of buffer memory for reference picture. The needed number of the buffer memories is according to the strategy of the reference picture selection by the encoder, and transmission delay between the encoder and the decoder. If the encoder selects the reference picture according NP_NACK message, the additional memory is not always needed. When some error concealment techniques are used in the decoder, it is desirable that the decoder has at least one additional memory.

E.1.6.6 Interaction between the video layer and the system layer

In the downstream, one NP segment is suggested to form a transmission unit of the system standard.

In the upstream, an upstream message is suggested to form a transmission unit of the system standard.

If the system layer detects the transmission errors in the downstream, it is desirable to deliver the error indication with video data in the decoder. In the upstream, the errored upstream message may be discarded in the encoder.

E.2 Complexity Estimation

The Complexity Estimation Tool enables the estimation of decoding complexity without the need of the actual decoding of the incoming VOPs. The tool is based on the trasmission of the statistic of the actual encoding algorithms, modes and parameters used to encode the incoming VOP. The 'cost' in complexity for the execution of each algorithm is measured or eastimated on each decoder platform. The actual statistic of the decoding algorithms is transmitted in the video bitstream and can be converted by means of the mentioned 'costs' into the VOP actual decoding cost for the specific decoder.

The tool is flexible since it enables, for each VO, the definition of the set of used statistics. Such definition is done in the VO header. The actual values of the defined statistics is then inserted into each VOP header according to the 'complexity estimation syntax'.

E.3 Resynchronization in Case of Unknown Video Header Format

Two video object layer starting indicators are supported:

1. video_object_layer_start_code, and
2. short_video_start_marker

The automatic detection of which of these byte aligned codes is present is unambiguous. The short_video_start_marker will never emulate a video_object_layer_start_code, since 23 byte-aligned zeros cannot occur in any video stream using the short_video_start_marker. The video_object_layer_start_code will also never emulate a short_video_start_marker, because its first non-zero bit is in a different location (provided byte alignment is not lost).

However, special attention needs to be paid if some application requires starting at any arbitrary point in a bitstream for which there is no prior knowledge of the format type. Although unlikely, a resync_marker can emulate a short_video_start_marker (for certain macroblock_number field lengths and macroblock_number values and vop_fcode_forward values).

Although the behavior of the decoder in these circumstances is not specified, it is suggested to perform validity checks on the first few bits beyond the short_video_start_marker if the video_object_layer_start_code is not the first starting indicator found. Numerous checks are possible, for example, checking the values of the bits 9, 10, 18-21 and 27 beyond the short_video_start_marker. The decoder may also choose to delay acquisition until an "I" vop-type is indicated in bit 17. Even simply discarding some data while searching for a video_object_layer_start_code prior to "timing out" with a decision to seek the short_video_start_marker may be acceptable for some applications.

© ISO/IEC 2004 – All rights reserved

Annex F
(informative)

Preprocessing and postprocessing

F.1 VOP Generation Tools: Automatic and Semi-automatic Segmentations

F.1.1 Automatic Segmentation

F.1.1.1 Introduction

The video coding scheme defined by this part of ISO/IEC 14496 offers several content-based functionalities, demanding the description of the scene in terms of so-called video-objects (VOs). The separate coding of the video objects may enrich the user interaction in several multimedia services due to flexible access to the bitstream and an easy manipulation of the video information. In this framework, the coder may perform a locally defined pre-processing, aimed at the automatic identification of the objects appearing in the sequence. Hence, segmentation is a key issue in efficiently applying the ISO/IEC 14496-2 coding scheme, although not affecting at all the bitstream syntax and thus not being a normative part of the standard.

Usually, the term segmentation denotes the operation aimed at partitioning an image or a video sequence into regions extracted according to a given criterion. In the case of video sequences, this partition should achieve the temporal coherence of the resulting sequence of object masks representing the video object. In the recent literature, different methods have been proposed for segmentation of video sequences, based on either a spatial homogeneity, a motion coherence criterion [13] or on joint processing of spatio-temporal [2][4][5][6][16]. These algorithms are expected to identify classes of moving objects, according to some luminance homogeneity and motion coherence criterion.

In this annex, a framework aiming at an appropriate combination of temporal and spatial segmentation strategies, developed throughout the standardisation phase of ISO/IEC 14496 Version 1, is described. The description is given only for informative purposes as the technique to extract objects from the scene is not standardised.

F.1.1.2 Description of a combined temporal and spatial segmentation framework

Throughout the work on automatic segmentation of moving objects, different proposals for temporal and spatial segmentation algorithms have been investigated. This resulted at the end in a combined temporal and spatial segmentation framework [11] which is shown in a high level block diagram in Figure F.1.

© ISO/IEC 2004 – All rights reserved

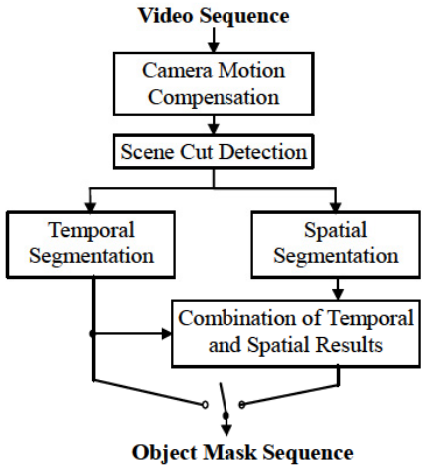


Figure F.1 — Block diagram of combined temporal and spatial segmentation framework

The combined scheme applies in a first step the general blocks of camera motion estimation and compensation [8][9][10] and scene cut detection [8][9][10] which can be seen as a kind of pre-processing in order to eliminate the influence of a moving camera. In a second step, either temporal or combined spatio-temporal segmentation of each image is carried out, depending on the requirements. The reason for this is, that in general only performing temporal segmentation requires less computational complexity. On the other hand, taking into account also spatial segmentation leads to more accurate segmentation results, but increases the computational complexity of the segmentation.

The evaluation of the segmentation results has been done subjectively by tape viewing for the MPEG-4 test sequences Mother Daughter, Akiyo, Hall Monitor, Container Ship, Coastguard, and Table Tennis, in QCIF and CIF formats with a frame rate of 10 Hz. In addition, the results have been objectively evaluated by the three criteria presented in [17], two for spatial accuracy and one for temporal coherence. The algorithms for camera motion estimation and compensation, for detection of scene-cuts, for temporal and spatial segmentation, and for combining temporal and spatial segmentation results will be described in the following.

F.1.1.2.1 Estimation and compensation of camera motion [8][9][10]

Given two successive images $I_{(k)}$ and $I_{(k+1)}$ of a video sequence captured by a static or moving camera, an apparent camera motion is estimated and compensated. For this purpose, a method is used which estimates the motion parameters of a rigid plane object within a 3D space. Thus, the background of the scene is assumed to be planar. The eight parameters of the motion model $a_1 - a_8$ can reflect any kind of motion, especially zoom and pan. For every pel (X_0, Y_0) in frame $I_{(k)}$ the corresponding pel (X_1, Y_1) in frame $I_{(k+1)}$ is given by:

$$X_1 = \frac{a_1 X_0 + a_2 Y_0 + a_3}{a_7 X_0 + a_8 Y_0 + 1} \qquad Y_1 = \frac{a_4 X_0 + a_5 Y_0 + a_6}{a_7 X_0 + a_8 Y_0 + 1}$$

The camera motion is estimated by regression considering only pels within background regions of the previous image. After the estimation of the motion parameters, the displacement vector for every pel is known. Then, a post-processing step counts for model failures in background regions due to the assumption of a rigid plane. By this post-processing step, the estimated displacement vector of every background pel is improved performing a full search within a squared area of limited size. For robustness reasons of the algorithm, the camera motion is only compensated if a moving camera has been detected.

F.1.1.2.2 Scene-cut detection [8][9][10]

In case of a scene cut between the two corresponding frames, the camera motion can not be compensated in a serious manner. Thus, scene cuts have to be detected for being able to perform further analysis steps on the video

© ISO/IEC 2004 – All rights reserved

sequence. The scene cut detector described here evaluates whether the difference between the current original image $I_{(k+1)}$ and the camera motion compensated previous image exceeds a given threshold. The evaluation is only performed within the background regions of the previous frame.

F.1.1.2.3 Temporal segmentation based on change detection [10]

This temporal segmentation algorithm, which is mainly based on a change detection, can be subdivided into several steps, assuming that a possible camera motion has already been compensated: by the first step, an initial change detection mask (CDMi) between two successive frames is generated. In this mask, every pel where the luminance of the image has changed due to a moving object is marked. After that, boundaries of changed image areas are smoothed by a relaxation technique [1][2], resulting in a mask CDMs. Thereby the algorithm adapts frame-wise automatically to camera noise. While in [1] the variance of camera noise and the variance within object regions are both measured based on the CDMi, here the previous mask of object regions (OM) is additionally used, where every pel is marked which belongs to a moving object. In order to finally get temporally stable object regions, a memory for change detection masks (CDM) is applied, denoted MEM. The temporal depth of MEM adapts automatically to the sequence. Now, the CDMs is simplified by usage of a morphological closing-operator and elimination of small regions, resulting in the final CDM.

After that, an initial object mask (OMi) is calculated by eliminating the uncovered background areas from the CDM [7]. Therefore displacement information for pels within the changed regions is used. The displacement is calculated by a hierarchical block-matcher (HBM) [3]. For a higher accuracy of the calculated displacement vector field (DVF) the estimated CDM is considered. At last, the boundaries of the OMi are adapted to grey level edges in the corresponding image in order to improve the accuracy of the object mask, resulting in the final object mask (OM).

F.1.1.2.3.1 Computation of the initial CDM

Given two successive frames $I_{(k)}$ and $I_{(k+1)}$ of an image sequence, first the initial CDM (CDMi) is computed by a threshold operation on the squared luminance difference image, as it is described in [1]. The threshold is calculated by performing a significance test. Therefore the luminance difference image D is modelled as Gaussian camera noise with a variance σ^2 equal to twice the variance of the camera noise, which means that a luminance difference d_i for pel i is assumed to correspond to camera noise (null hypothesis H_0) and not to moving objects (hypothesis H_1):

$$p(d_i \mid H_0) = \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma^2}} \cdot \exp\left\{-\frac{d_i^2}{2 \cdot \sigma^2}\right\} \tag{1}$$

Assuming that neighbouring pels are statistical independent, the sum of pels in the squared difference image within a window of size $(n \times n)$, normalised by twice the variance of camera noise, is χ^2 distributed with n^2 degrees of freedom (Eq. (2)).

$$T = \frac{1}{\sigma^2} \sum_{i=1}^{nn} d_i^2 \tag{2}$$

All pels for which the sum T exceeds a certain threshold t_α are marked as changed, the others are marked as unchanged. The threshold t_α is chosen in such a way, that assuming the pel has not changed caused by moving objects, the probability, that a calculated sum T is greater than this threshold, equals to a given value α.

$$\alpha = p(T > t_\alpha \mid H_0) \tag{3}$$

Here, the threshold t_α is equal to 220 in the case of QCIF sequences, while in the case of CIF sequences t_α is set to 165.

F.1.1.2.3.2 Relaxation of the initial CDM for spatial homogeneity

Because real objects have smooth contours, spatial homogeneity of the CDM is required. Spatial homogeneity is achieved by relaxation of the CDMi using a method [1]. There the CDMi is processed iteratively. In each iteration step a decision is made for every border pel in the CDMi, if it belongs to the changed or to the unchanged area.

© ISO/IEC 2004 – All rights reserved

Thus, for every border pel of changed image areas a local threshold is calculated taking into account the local neighbourhood of the pel. For the decision, if a pel has changed by moving objects or not a maximum a priori estimator (MAP estimator) is used:

$$\frac{P(Q_c \mid D)}{P(Q_u \mid D)} \underset{u}{\overset{c}{\gtrless}} 1 \Leftrightarrow \frac{p(D \mid Q_c)}{p(D \mid Q_u)} \underset{u}{\overset{c}{\gtrless}} \frac{P(Q_u)}{P(Q_c)} \tag{4}$$

Here, Q_c means that a pel is marked changed in the CDM, while Q_u means that this pel is marked unchanged in the CDM. *D* denotes the given difference image between frame $I_{(k)}$ and $I_{(k+1)}$. The probabilities in Eq. (4) are modelled as following:

The probability that a luminance difference d_i is caused by camera noise $p(D \mid Q_u)$ is described by the same Gaussian distribution as given in Eq. (1). The corresponding probability, that a luminance difference d_i is caused by moving objects $p(D \mid Q_c)$ is modelled as a Gaussian distribution, too, but the variance of this distribution is not twice the variance of the camera noise, but the variance σ_c^2 of the difference image D measured in foreground regions.

$$p(D \mid Q_u) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{d_i^2}{2\sigma^2}\right\} \quad (5), \qquad p(D \mid Q_c) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{d_i^2}{2\sigma_c^2}\right\} \quad (6)$$

The two a priori probabilities in Eq. (4) are modelled by a Markov random field considering a *(3x3)* - neighbourhood of a pel as shown in Figure F.2:

$$P(Q_u) = \frac{1}{Z}\exp\{-E(Q_u)\} \quad (7), \qquad P(Q_c) = \frac{1}{Z}\exp\{-E(Q_c)\} \quad (8)$$

The constant *Z* is for normalisation, while *E* denotes an energy term defined as:

$$E(Q_{u/c}) = v_B(u/c)B + v_C(u/c)C \tag{9}$$

The terms $v_B(q_k)$ and $v_C(q_k)$, $q_k \in \{u,c\}$, are a measure for the inhomogeneity of the neighbourhood of pel *k*, which is separated into horizontal or vertical neighbours (potential *B*) and diagonal neighbours (potential *C*), as shown in Figure F.2. So, the term $v_B(q_k)$ denotes the number of horizontal and vertical neighbours of pel *k* with the opposite label to q_k. In the same way the term $v_C(q_k)$ denotes the number of diagonal neighbours of pel *k* with the opposite label to q_k.

C	**B**	**C**
B	**X**	**B**
C	**B**	**C**

Figure F.2 — Neighbourhood of pel *X* and potentials *B* and *C* of the neighbours

From Eq. (4) to Eq. (9) the following decision rule can be derived, where d_i^2 denotes the squared luminance difference of pel *i*:

$$d_i^2 \underset{u}{\overset{c}{\gtrless}} 2 \cdot \frac{\sigma_c^2 \cdot \sigma^2}{\sigma_c^2 - \sigma^2} \cdot \left(\ln\frac{\sigma_c}{\sigma} + \left(v_B(c) - v_B(u)\right) \cdot B + \left(v_C(c) - v_C(u)\right) \cdot C \right) \tag{10}$$

© ISO/IEC 2004 – All rights reserved

The label of every border pel in the CDMi is decided by using this decision rule. The rule should read as follows: if d_i^2 exceeds the threshold term on the right hand side of Eq. (10), the pel is set to changed $(\mathrm{CDMs}_{(k+1)}(i) := 1)$, otherwise it is set to unchanged $(\mathrm{CDMs}_{(k+1)}(\mathrm{i}) := 0)$. The relaxation is processed iteratively, until nothing has changed by relaxation or the maximal number of iteration steps *N* is reached. The so obtained CDM is denoted CDMs.

For calculating the CDM the variances of the static background σ^2 and within object regions ${\sigma_c}^2$ are needed. While in [1] always the CDMi is used for calculating the variances ${\sigma_c}^2$ and σ^2, it is more exact to calculate the variance σ^2 within the unchanged regions of the CDM and the variance ${\sigma_c}^2$ within the object regions of the OM. Thus, the variance ${\sigma_c}^2$ does not include uncovered background. Since the positions of static background, uncovered background and moving objects are still unknown for the current frame, in this approach the OM and CDM from the previous frame are used. In order to increase the robustness in case of noise the measured variances are averaged with those of the last three times.

In the energy term (Eq. (9)) the potential B is set to 5 and the potential *C* is set to 2.5. Further, the maximal number of relaxation steps *N* is set to 20.

F.1.1.2.3.3 Temporal homogeneity of the OM

In order to finally get temporal stable object regions, the CDMs is updated with the previous OM, i.e. in the CDMs additionally all pels are set to changed which belong to the OM of the previous frame. This is based on the assumption that all pels which belonged to the previous OM should belong to the current change detection mask. However, in order to avoid infinite error propagation, a pel from the previous OM is only labelled as changed in the CDMs, if it was also labelled as changed in the CDMs of one of the last L frames, too. For this purpose, a storage for every pel is applied, building a memory (MEM). The value of this storage indicates, if the respective pel was set to changed in one of the *L* previous CDMs; the value L denotes the depth of the memory. Considering that CDMs, OM and MEM are two-dimensional fields and that MEM is the zero-matrix for the first frame, the update of MEM can be formulated as:

$$MEM_{(k+1)}(x,y) = \begin{cases} L & \text{if } \mathrm{CDMs}_{(k+1)}(x,y) = 1 \\ \max(0, MEM_{(k)}(x,y) - 1) & \text{if } \mathrm{CDMs}_{(k+1)}(x,y) = 0 \end{cases} \tag{11}$$

The current CDMs is then updated by logical OR operation between CDMs and the previous OM, taking into account the memory MEM:

$$\mathrm{CDMu}_{(k+1)}(x,y) = \mathrm{CDMs}_{(k+1)}(x,y) \vee \begin{cases} \mathrm{OM}_{(k)}(x,y) & \text{if } MEM_{(k+1)}(x,y) > 0 \\ 0 & \text{if } MEM_{(k+1)}(x,y) = 0 \end{cases} \tag{12}$$

It is reasonable to set the depth of the memory L to a large value, if the motion of the moving objects in the scene is small, and to set L to a small value, if the motion of the objects is large. Thus, L shall be decreased with the average amplitude D of the displacement vectors measured within object regions. Further, it is reasonable to increase the depth of the memory L with the size S of the moving objects. Hence, it is necessary for automatic segmentation to adapt the value of L with the sequence. Therefore, the value of L is heuristically calculated:

$$L = \max\left(1, L_{\max} - (D^4 / c_{L_1}) - c_{L_2} \cdot \exp\left(-10^{-3} \cdot (S / c_{L_3})\right)\right) \tag{13}$$

As can be seen in Eq. (13) the values of *L* are within the interval $[1; L_{\max}]$. L is decreased by the two terms D^4 / c_{L_1} and $c_{L_2} \exp\left(-10^{-3}\, s / c_{L_3}\right)$. The first term increases with the average amplitude of the displacement vectors within object regions powered by 4. The second term decreases exponentially with the size of the object regions. While the constants $\mathrm{c}_{\mathrm{L}_1}$ and $\mathrm{c}_{\mathrm{L}_3}$ are for normalisation to different image formats, $\mathrm{c}_{\mathrm{L}_2}$ is a weighting factor.

The maximum depth of MEM $L_{\max}$ is set to 34. While in case of QCIF sequences $\mathrm{c}_{\mathrm{L}_1} = 1$ and $\mathrm{c}_{\mathrm{L}_3} = 1$, in case of CIF sequences $\mathrm{c}_{\mathrm{L}_1} = 15$ and $\mathrm{c}_{\mathrm{L}_3} = 4$. The weighting factor $\mathrm{c}_{\mathrm{L}_2}$ is set to 80.

© ISO/IEC 2004 – All rights reserved

F.1.1.2.3.4 Simplification of the CDM

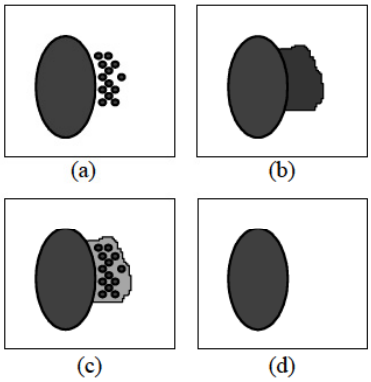


**Figure F.3 — Example for elimination of small regions using a ternary mask:
(a) CMD with a large region neighboured by smaller regions caused by noise;
(b) CMD after elimination of small regions without ternary mask;
(c) Ternary mask;
(d) CMD after elimination of small regions using a ternary mask**

In order to simplify the regions in the CDM, the morphological closing-operator is performed. Now, for elimination of small regions a ternary mask is generated, in which pels are labelled as changed, if they are set in the CDM before and after the closing operation. Pels, which are not set in the CDM before and after the closing operation, are set to unchanged in the ternary mask, while those pels, which are of different labels before and after the closing operation, are set to a third label. Within this ternary mask small changed and unchanged regions with a size below a certain threshold are eliminated. In order to prevent that very small regions, which are mostly due to noise spots, grow together with larger regions when the closing is performed, the third label is treated in a special way. An example for this method is given in Figure F.3a where a CDM is shown containing a large region, which is neighboured by many small regions caused by noise. Figure F.3b presents the CDM after closing is performed. It can be seen, that the noisy areas are grown together with the large region. In order to avoid this, the ternary mask is generated as described above and as it is shown for the example in Figure F.3c. The CDM after elimination of small regions by using a ternary mask is presented in Figure F.3d. At this time the final CDM is processed.

F.1.1.2.3.5 Detection and elimination of uncovered background

Based on the generated CDM the OMi is calculated by detecting and eliminating uncovered background, taking into account the displacement for pels within the calculated CDM. Therefore a DFV for the changed image areas is generated by hierarchical block-matching [3]. Hierarchical block-matching is used since the real motion is needed. In order to improve the DVF at object boundaries, the matching criterion is only evaluated for changed regions.

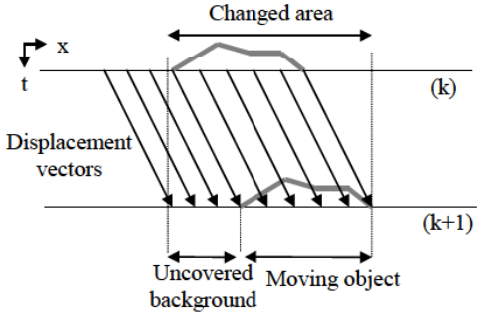


Figure F.4 — Separation of changed areas into foreground and uncovered background

© ISO/IEC 2004 – All rights reserved

Now, pels are set to foreground, if foot- and top-point of the corresponding displacement vector are both inside the changed area in the current CDM [7]. If not, these pels are set to background. An example for that is given in Figure F.4, where an object moves from the left to the right. Further, to prevent moving shadows being misclassified as moving objects or parts of moving objects, moving shadows can be explicitly detected.

F.1.1.2.3.6 Adaptation of the OM to grey level edges

At last, the OMi is adapted to grey level edges in the current frame, resulting in the final OM. For that an array for correction is calculated: for every region in the OMi the outside limit of the area for correction is given by the contour of that region considering the OMi after the morphological blow operator has been performed two times. The inside limit of the area for correction is given by considering a width of correction of 6 pel. Within this area of correction the luminance edges of the current image are calculated using a Sobel operator.

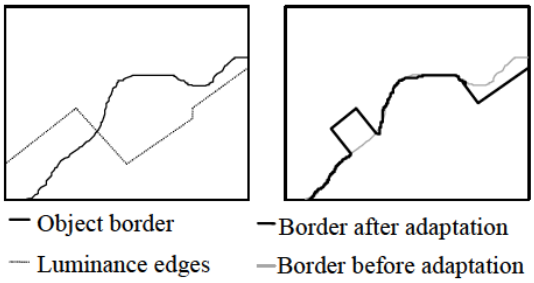


Figure F.5 — Exemplary result of the edge adaptation algorithm

By this edge adaptation technique every border pel in the OM is warped to the nearest grey level edge of frame $I_{(k+1)}$, if there exists such an edge in the area for correction within a circle of radius R = 12 around that pel.

In Figure F.5, an example for edge adaptation is given: On the left side a part of the border of an moving object in an OM is shown before edge adaptation is processed. Additionally the found luminance edges of the current image are drawn in. On the right side the corresponding part of the border after edge adaptation is presented.

F.1.1.2.4 Spatial segmentation based on watershed algorithm [5]

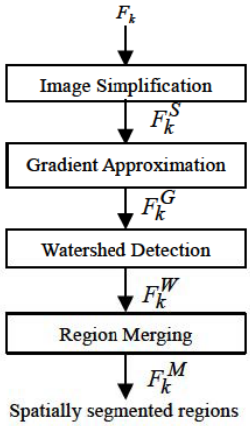


Figure F.6 — Spatial segmentation process

© ISO/IEC 2004 – All rights reserved

Spatial segmentation splits the entire image into homogeneous regions in terms of intensity. The different homogeneous regions are distinguished by their encompassing boundaries that can be obtained from the spatial segmentation. A method for spatial segmentation is here introduced which is based on a morphological segmentation that utilizes morphological filters and a watershed algorithm [15]. However the watershed usually introduces over-segmentation problems which cause many partitioned regions even in a single object region. A region merging process is devised to avoid the over segmentation after watershed in the spatial segmentation. Figure F.6 depicts the spatial segmentation process. The segmentation algorithm consists of the following four steps: Simplification, Gradient approximation, Watershed detection and Region merging. The details of each step are explained in the following subclauses.

F.1.1.2.4.1 Image Simplification

For spatial segmentation, images are first simplified for the ease of spatial segmentation. Many morphological filters are often designed for smoothing noisy gray level images by a determined composition of opening and closing with a given structuring element [14]. Here erosion and dilation with a flat structuring element are used in constructing a morphological filter for image simplification.

$$\varepsilon_n(F_k(x,y)) = \min(F_k(x+l, y+m)), \quad \forall (l,m) \in R_n \tag{14}$$

$$\delta_n(F_k(x,y)) = \max(F_k(x-l, y-m)), \quad \forall (l,m) \in R_n \tag{15}$$

Morphological opening and closing operations are given by

$$\gamma_n(F_k) = \delta_n(\varepsilon_n(F_k)) \tag{16}$$

$$\varphi_n(F_k) = \varepsilon_n(\delta_n(F_k)) \tag{17}$$

It can be noticed that the morphological opening operation $\gamma_n(F_k)$ simplifies the image by removing the bright components that do not fit within the structuring elements while $\varphi_n(F_k)$ eliminates the dark components compared to the structuring components. For the case of the images containing fluctuating scale values, the bright and dark components should be considered for the simplification. With the boundary preservation, morphological opening/closing by reconstruction filters are used for the purpose of image simplification [14]. That is, the usage of these filter pairs remove regions that are smaller than a given size but preserve the contours of the remaining objects in the image. The opening/closing by reconstruction filters is given by partial reconstruction:

morphological opening by reconstruction of opening: $\gamma^{(rec)}(\gamma_n(F_k), F_k)$ (18)

morphological closing by reconstruction of closing: $\varphi^{(rec)}(\varphi_n(F_k), F_k)$. (19)

From (18) and (19), the image F_k is simplified by $\gamma_n(F_k)$ which operation results in the maximum value among the inspection region compared to the structural element values while $\varphi_n(F_k)$ results in the minimum value at the locations (x, y).

By removing unwanted details in texture of the regions, the image is homogenized in terms of region textures otherwise small connections among the different parts of the image lead to undesirable structures. So this morphological operation helps spatial segmentation obtain more semantic region partitioning in the later stages. The image simplification is not necessary applied only to the gray level image. For the regions that show low contrasts, the simplification may results in merging of the two regions that should be separated. The size of the regions eliminated is dependent on applications. A simplified image is shown in Figure F.8b.

F.1.1.2.4.2 Gradient Approximation [5]

The spatial gradient of the simplified image is approximated by the use of morphological gradient operator [14]. Through the image simplification, the inside of each homogeneous region is small gradient but large gradients are expected along the region boundaries among different homogeneous regions in the image. So the spatial gradient

© ISO/IEC 2004 – All rights reserved

can be used as an input of watershed algorithm to partition an image into homogeneous intensity regions. The morphological gradient operator used is expressed such as:

$$G(F_k) = \delta_1(F_k) - \varepsilon_1(F_k) \tag{20}$$

Note that the morphological gradient is positive. The positive gradients usually indicate borders between neighbor regions and allow the watershed to split the image into regions which are divided by the borders.

In the regions where intensity variations are small, adjacent regions are ambiguous so that watershed algorithm may yields false contours due to the resulting small gradients. To overcome this problem, we incorporate color information into gradient computation in which the largest values among the gradients obtained in αG_Y, βG_{Cr} and γG_{Cb} are chosen. α, β and γ are weighting factors applied to the gradients of αY, βC_r and γC_b, respectively.

$$\max\{\alpha G_Y, \beta G_{Cr}, \gamma G_{Cb}\} \tag{21}$$

In many practical cases, the resulting gradient images exhibits many noisy gradients which can cause the watershed algorithm to oversegment the image. In order to prevent this, the gradient images are thresholded by a value. That is, small gradient values less than the threshold value are set to zero, otherwise they remain same. This gradient image is used as input image for the watershed detection. A gradient image by the morphological gradient operator is shown in Figure F.8c.

F.1.1.2.4.3 Watershed detection [5][14][15]

The image is often interpreted as geographical surface in mathematical morphology and its grey level is treated as altitude. As for geographical surfaces, image structure exhibits inclines, hills and plateaus over the image plane. Usually a semantic region is distinguished by its inclines from the surrounding contiguous regions in the image. The gradient exhibits large values in the inclines. So a watershed algorithm is applied to the gradient image to partition the image into homogeneous intensity regions. A number of watershed algorithms have been proposed in the literatures but here we use an immersion based watershed algorithm that is simple and computationally efficient. The watershed algorithm is a region growing algorithm and it assigns pixels in the uncertainty area to the most similar region with some segmentation criterion such as difference of intensity values. The final result of the algorithm is a tessellation of the input image into its different catchment basins.

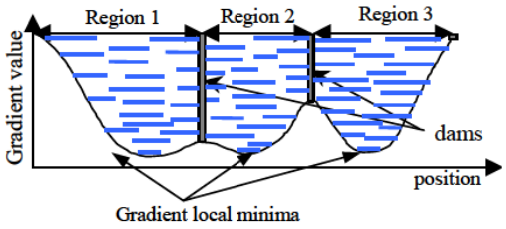


Figure F.7 — Watershed flooding

Figure F.7 depicts a 1-D graphical illustration of the watershed algorithm in an intuitive way. The local gradient minima are detected and used as seeds for region growing. In the image, these local minima are derived from homogeneous intensity regions. As the tessellation of the gradient image is immersed, each catchment basin is filled up from its lowest altitude which was selected as a seed. When the surface of the water in each catchment basin reach top of the crest, a dam is made for the water to overflow into the nearby catchment basins. Finally the water surface in each catchment basin reaches the same level but is confined within the region which is encompassed by the crest lines or dams built on the crest in the gradient image. Then each region is assigned a unique label.

The watershed algorithm is highly sensitive to gradient noise, which yields many catchment basins, so resulting in over-segmentation. To solve this problem, the region merging algorithm based on intensity homogeneity is incorporated. The image after the watershed detection is shown in Figure F.8d.

© ISO/IEC 2004 – All rights reserved

F.1.1.2.4.4 Region merging [5]

In the region merging step of the algorithm, small regions are merged in order to yield larger and meaningful regions which are homogeneous and different from its neighbors. This scheme represents both regions and their boundaries explicitly, and makes the storing and indexing of their semantic properties possible. For region merging, it must be determined which regions are merged to their neighbors according to certain predefined criteria. In this experiment, we used as a similarity measure (T) the ASAD (Average Sum of Absolute Difference denoted by R_t) of a region under consideration between two successive frames and the average intensity (R_s) of the region in the current frame.

$$T = \beta R_s + (1-\beta) R_t \tag{22}$$

Each region that consists of less than N pixels should be merged to its most similar region. That is, we compare the similarity measure (T) of a region under consideration with its neighbors. Then the region under consideration is merged to a region where the difference between two similarity measures is smallest. For QCIF and CIF sequences we used β = 0.2 in our experiments. The value N controls the amount of meaningful regions in a segmented image. We start the merging process with N = 10, and repeat merging iteratively with increasing N by 10 until the value N reaches 50. Therefore the merging process is done four times. A merged image is shown in Figure F.8e.



a) original image b) simplified image c) gradient image d) watershed image e) merged image

Figure F.8 — Interim results in Spatial Segmentation process

F.1.1.2.5 Combination of temporal and spatial results (method 1) [5]

In the last step, the decision of foreground(FG)/background(BG) is made on the spatially segmented regions with object masks obtained form temporal segmentation in the image. A spatial region ratio and temporal region ratio are computed to make judgement on FG/BG decision.

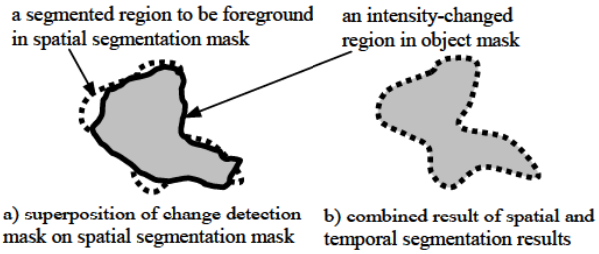


Figure F.9 — Combination of temporal and spatial segmentation results

First, we superimpose the change detection mask on top of the spatial segmentation mask. For the spatial region ratio, when a majority part of a segmented region belongs to changed region in the change detection mask, the whole area of the segmented region is declared as a foreground, otherwise a background. Figure F.9 illustrates a combination scheme of temporal and spatial segmentation results. The parts exceeding region boundaries in the spatially segmented region can be excluded from the foreground region in Figure F.9b when the portion of the exceeding pixels compared to the neighbour regions is small, while uncovered parts of the spatially segmented region by the object mask can be all foreground when their portion of areas are small compared to the part of the region covered by the foreground part of the object mask.

© ISO/IEC 2004 – All rights reserved

Now we consider a historical fact defined by the temporal region ratio. We overlay the spatially segmented regions on the previous labelled image. If the majority part of a region under consideration belongs to a foreground of the previous labelled image, the region is also considered a foreground. This step exploits historical attributes to correct the wrong results especially when a moving object stops.

F.1.1.2.6 Combination of temporal and spatial results (method 2)

This combing technique evaluates the results from the temporal and spatial segmentation. For time reasons, the described technique is considered as a preliminary version which will be further improved. The technique works as follows: for each region of the spatial segmentation a foreground / background decision is made. If at least 80% of the pels within a region were detected as foreground by the temporal segmentation, all pels of the region already detected as foreground and all pels within a correction range of 2 pel with respect to the boundary of the temporal segmentation are set to foreground, the other to background. If less than 80% of the pels within a region were detected as foreground by the temporal segmentation, all pels of the region already detected as background and all pels within a correction range of 2 pel with respect to the boundary of the temporal segmentation are set to background, all other to foreground. In future work, it is planned to additionally consider the size of the spatial regions and the quality of the temporal and spatial segmentation results in the combination criterion.

F.1.2 Semi-automatic Segmentation

In order to support the philosophy of the MPEG-4 Video standard, each image of video sequences should be represented in terms of VOPs to be encoded, that is, the objects to be encoded in image sequences should be prepared before the encoding process starts. The MPEG-4 Video has committed to providing an import tool to support for the object-based handling of video data during the phase 1 of MPEG-4 standardization.

However, the current technologies for automatic segmentation are somehow premature to obtain desirable segmentation results because automatic segmentation itself is an open-issue and ill-posed problem. Nevertheless, there is a more interesting and attractive approach to image segmentation which is semi-automatic. The approaches to semi-automatic segmentation are much more practical and give better segmentation results than the automatic segmentation.

In this informative annex, two different approaches are presented. The first method [18][19][20] consists of two processing steps: intra-frame segmentation and inter-frame segmentation. The intra-frame allows users to initially define any regions of interest via a graphical user interface and finds out precise boundaries of the region defined. The inter-frame segmentation automatically separates the region defined by object tracking techniques. The second method allows a user to define regions of interest by a rectangular box and to control user-supply parameters of the algorithm.

F.1.2.1 User-assisted segmentation of moving objects (method 1) [18][19][20]

As shown in Figure F.10, the semi-automatic segmentation can be thought of as a user-assisted segmentation technique. It consists of two steps: intra-frame segmentation and inter-frame segmentation. First, the intra-frame segmentation is applied in the first frame of the image sequence or the frames which contain newly appeared video objects or get bad segmentation results. The intra-frame segmentation can be processed by a user manually or semi-automatically. Then the inter-frame segmentation is applied in the next consecutive frames.

© ISO/IEC 2004 – All rights reserved

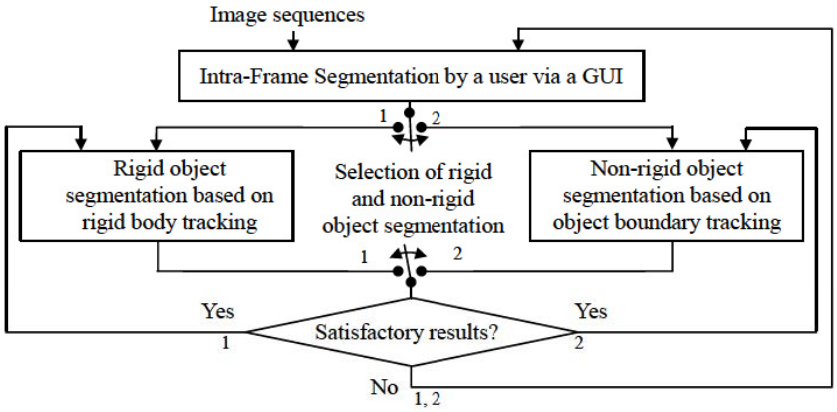


Figure F.10 — The block diagram of a user-assisted video object segmentation method

F.1.2.1.1 Intra-Frame Segmentation

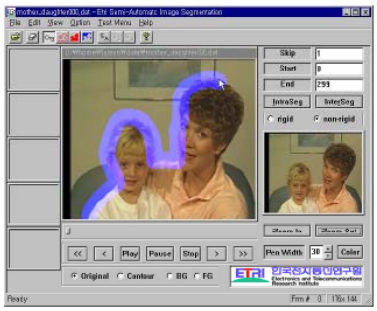


Figure F.11 — A graphical user interface for supporting intra-frame segmentation

The objective of intra-frame segmentation is to find precise object boundaries with cognitive information of Human at the beginning by user intervention. Since automatic segmentation separates moving objects as video objects without any cognitive information *a priori*, it is very difficult to obtain satisfactory segmentation results in general in generic video. Therefore much technological progress has to be made in order for automatic segmentation to be used in practice in real world problems. On one hand, since definition of a video object is somewhat subjective, user intervention is inevitable in segmentation process. That is, the human cognitive information can be provided into segmentation through a graphical user interface (GUI). Figure F.11 depicts an intra-frame segmentation.

Supposed the user is interested in extracting objects with precise boundaries of the objects in MOTHER_DAUGHTOR image sequences in the main window (a large window at the GUI center) of GUI in Figure F.11.

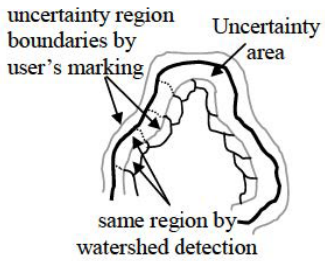


Figure F.12 — Findings of object boundaries with in the uncertainty area

© ISO/IEC 2004 – All rights reserved

A user can mark the entire objects all the way around the objects using a mouse with an 8-pixel wide width. A marked swath (light blue-colored region) made by the mouse constitutes an uncertainty area in which the real object boundaries are assumed to exists. A user can select a thickness of the mouse point in order to mark rough boundary regions around the true object boundaries of interest in an initial image. Figure F.13 illustrates the intra-frame segmentation process.

From the initial image with the marked region, globally labeled image is obtained in which the uncertainty region (UR) is labeled with '0', object regions of interest with '1' and the other regions with '2'. A watershed algorithm is then applied to the initial image in order to perform spatial segmentation.

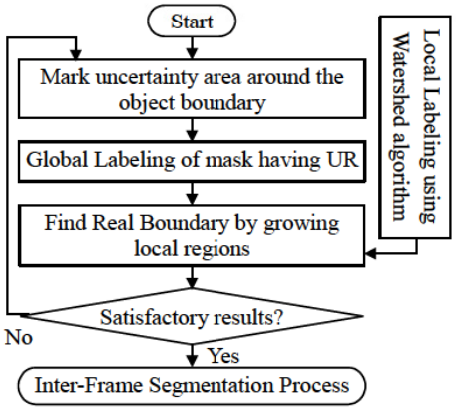


Figure F.13 — Intra-Frame Segmentation Process

The spatially segmented regions are labeled with integers, which forms a locally labeled image. In Figure F.13, locally labeled regions extending from the interior of the uncertainly region to the exterior are merged into a globally labeled region. That is, the regions that span between the uncertainty area and a globally labeled region are assigned into the globally labeled region in near vicinity. Finally the real boundaries of objects of interest can be obtained.

After finding the real boundary of the region of interest, the resulting resulting segmentation mask is taken as input into the inter-frame segmentation process if the intra-frame segmentaiton result is satisfactory. In the cases of unsatisfactory results for the real boundary finding, the intra-frame segmentation process can restart until the segmentation result is satisfactory. Note that any apriori initial segmentation mask can be used as input for the inter-frame segmentation process. This will be the case that the initial masks equivalent to the results of the intra-frame segmentaiton process can also be obtained from other intra-frame segmentation operations.

F.1.2.1.2 Inter-Frame Segmentation

In the inter-frame segmentation, the moving objects are segmented with precise object boundaries in automatic manners based on the object boundary tracking for non-rigid objects or the motion tracking for rigid objects. This automatic segmentation of the inter-frame segmentation alleviates user's effort of intervention during the segmentation process. Since the inter-frame segmentation starts with the initially obtained object mask having precise object boundary, it yields precise segmentation masks for the moving objects. In the following subclauses, the core parts of the inter-frame segmentation are described in details for non-rigid and rigid object segmentation.

F.1.2.1.2.1 Non-rigid object segmentation by object boundary tracking

The non-rigid object segmentation process is shown in Figure F.14. At first, the previous object boundary is extracted from the mask information obtained by intra-frame segmentation process. To obtain the initial object boundary in the current frame, the motion estimation and compensation is then applied to every coordinate composing the previous object boundary. Depending on the characteristics of the background, block or polygon matching criteria can be used to minimize the erroneous result due to noise.

© ISO/IEC 2004 – All rights reserved

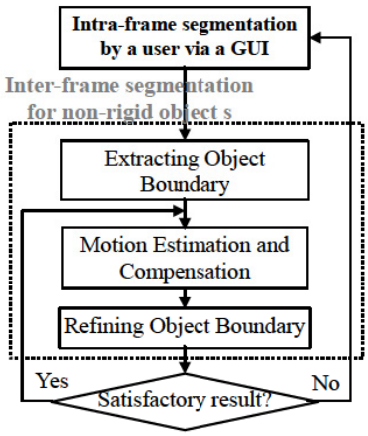


Figure F.14 — Inter-frame segmentation process for non-rigid object

$$MAD(u,v) = \sum_{i=-N/2}^{N/2} \sum_{j=-N/2}^{N/2} \left| S(i+u, j+v) - R(i,j) \right| \times \alpha(i,j)$$

$$MV = (u,v)\Big|_{\min MAD(u,v)}, \quad -p \le u, v \le p$$

where N is a reference block size and p is a search range. $S(i+u, j+v)$ and $R(i,j)$ are the search area data in the current frame and the reference block data in the previous frame, respectively. For the polygon based motion estimation $\alpha(i,j)$ is an alpha value of the corresponding pixel $R(i,j)$ while $\alpha(i,j)$ is always 1 for the block based motion estimation. Then the initial boundary is refined pixel by pixel by using both spatial and temporal information to reduce error propagation and the final object boundary is obtained. At this refinement step, the edge strength in the current frame is used as the spatial information and the motion compensated error is used as the temporal information.

$$MV_{refined} = (u+u', v+v')\Big|_{\max \frac{EdgeStrength(u+u', v+v')}{MAD(u+u', v+v')}}, \quad -p' \le u', v' \le p'$$

After refining the object boundary, a median filter is applied all the way around the object boundary in order to smooth out the rough boundary intervals.

F.1.2.1.2.2 Rigid object segmentation by motion tracking

In this subclause, the segmentation algorithm for rigid objects is described. It is based on parametric motion estimation and motion compensation which establish the mapping from one *rigid* object contour to the other. The segmentation algorithm consists of the following 4 steps: a) 2 parameter motion estimation; b) object mask refinement; c) 6 parameter motion estimation; d) motion compensation for the object contour. The block diagram is illustrated in Figure F.15.

© ISO/IEC 2004 – All rights reserved

F.1.2.1.2.2.1 Object Motion Models

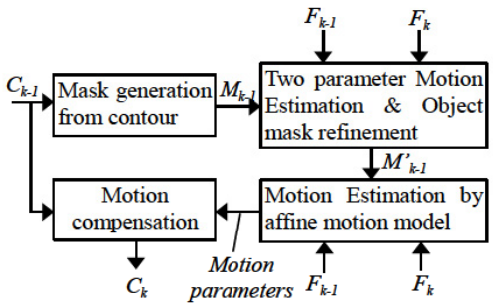


Figure F.15 — Block diagram for the object motion model

We use the affine 6-parameter motion model for the object motion estimation.

$$p(x, y, t) = a + bx + cy$$

$$q(x, y, t) = d + ex + fy$$

where $(p(x, y, t), q(x, y, t))$ is the displacement induced on pixel (x, y) by the motion of the object between frames t and $t+1$. Assuming the intensity is invariant, $I(x+p, y+q, t+1)$ is equal to $I(x, y, t)$. By use of Taylor expansion, following equation is obtained.

$$pI_x + qI_y + I_t = 0$$

where I_x, I_y, I_t are partial derivatives of the intensity $I(x, y, t)$ with respect to x, y, t.

$$E(p, q) = \sum_{(x,y)\in M} (pI_x + qI_y + I_t)^2$$

where M is the object mask. Deriving $E(p, q)$ with respect to the motion parameters and setting to zero yields six linear equations in the six unknowns: $\mathbf{m} = (a, b, c, d, e, f)^T$ as follows.

$$\begin{pmatrix} \sum I_x^2 & \sum xI_x^2 & \sum yI_x^2 & \sum I_xI_y & \sum xI_xI_y & \sum yI_xI_y \\ \sum xI_x^2 & \sum x^2I_x^2 & \sum xyI_x^2 & \sum xI_xI_y & \sum x^2I_xI_y & \sum xyI_xI_y \\ \sum yI_x^2 & \sum xyI_x^2 & \sum y^2I_x^2 & \sum yI_xI_y & \sum xyI_xI_y & \sum y^2I_xI_y \\ \sum I_xI_y & \sum xI_xI_y & \sum yI_xI_y & \sum I_y^2 & \sum xI_y^2 & \sum yI_y^2 \\ \sum xI_xI_y & \sum x^2I_xI_y & \sum xyI_xI_y & \sum xI_y^2 & \sum x^2I_y^2 & \sum xyI_y^2 \\ \sum yI_xI_y & \sum xyI_xI_y & \sum y^2I_xI_y & \sum yI_y^2 & \sum xyI_y^2 & \sum y^2I_y^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = -\begin{pmatrix} \sum I_xI_t \\ \sum xI_xI_t \\ \sum yI_xI_t \\ \sum I_yI_t \\ \sum xI_yI_t \\ \sum yI_yI_t \end{pmatrix}$$

Let the 8×8 matrix $\mathbf{A}$ and right side column vector $\mathbf{b}$.

© ISO/IEC 2004 – All rights reserved

F.1.2.1.2.2.2 Motion Estimation & Compensation

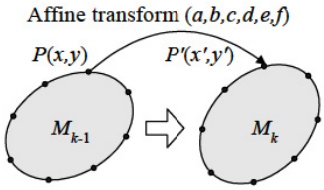


Figure F.16 — The object mask obtained by affine motion compensation

First, we generate the object mask M_{k-1} from previous contour C_{k-1}. Then we perform 2-parameter motion estimation between two successive frames F_{k-1}, F_k for the foreground object region. The object mask is refined by removing outlier points whose motion compensation error is larger than the given threshold. We calculate the affine motion parameters with the refined object mask. The motion parameter $\mathbf{m}$ is given by $\mathbf{A}^{-1}\mathbf{b}$. Since the displacement between two successive frames may not sufficiently small for the above equation, iterative estimation is needed to obtain correct motion parameter. Iterative estimation is as follows; 1) $\mathbf{m}^{(0)} \leftarrow \mathbf{0}^T$, $i \leftarrow 0$, 2) F_k is warped toward F_{k-1} using motion parameter $\mathbf{m}^{(i)}$, 3) calculate $\mathbf{A}, \mathbf{b}, \Delta\mathbf{m} = \mathbf{A}^{-1}\mathbf{b}$ between F_{k-1} and F_k^{warped}, 4) $\mathbf{m}^{(i+1)} = \mathbf{m}^{(i)} + \Delta\mathbf{m}$, $i \leftarrow i+1$, 5) if $\|\Delta\mathbf{m}\| < Th$ then stop iteration else go to step 2). All points in C_{k-1} are affine transformed and construct the new object contour C_k as shown in Figure F.16.

F.1.2.1.3 Results of the user-assisted segmentation



Figure F.17 — Results of the user-assisted segmentation

For the intra-frame segmentation, the top most window in the left column windows of the GUI displays a labelled image, called *mask image*, of the initially marked image by the user. The uncertainty region is remarked by the white swath and the globally labelled regions are indicated by red and blue colors. After finding the real object boundary, the resulting boundary image is shown in the second top most window. For the inter-frame segmentation, the non-rigid objects (foreground) are separated automatically and shown in the main window. These foreground and background images are also shown in the third and fourth windows in the left column windows. The foreground and background of the entire image can be used as video object planes for MPEG-4. Note that, in the user-assisted segmentation, the inter-frame segmentation process can be stopped at any frame by any metrics of measuring segmentation performance or user's judgement if the segmentation results are unsatisfactory.

F.1.2.2 Semi-automatic approach to segmentation (method 2)

Semi-automatic approach to segmentation could be a straight way to increase segmentation performance for non real time applications, enabling the user to introduce in the segmentation process high-level information.

© ISO/IEC 2004 – All rights reserved

A lot of interaction possibilities between the user and the segmentation process can be envisaged. For example, a user can initially mark objects of interest around the object boundaries; then the selected objects could be tracked during their temporal evolution in the video sequence until the next user interaction occurs. In this case, the user should be able to stop at any frame in image sequences and to revisit/remark the objects of interest with more exact boundaries whenever the segmentation performance is not satisfactory or a new object of interest has to be extracted.

Simple interactions could be useful to adapt segmentation procedures temporally (frame by frame), spatially (inside each frame) or sequence by sequence, by changing some significant parameter on the basis of the viewed results. The interactive adjustment of the parameters could be useful to discriminate more efficiently interesting objects from randomly varying background (see, grass, etc.).

A different approach enables the user to select, inside a frame, the area (Region Of Interest) in which the object is located. This allows to optimize the extraction process on the ROI only. Segmentation techniques based on motion analysis modules could take advantage from this approach as they usually detect several moving regions, but only one represents the object of interest for the user.

The algorithm for the semi-automatic segmentation separates, in each frame of an image sequence, regions corresponding to moving objects from regions belonging to a static background [12][21].

The algorithm is based on a change detection analysis performed along several frames in order to detect, for each frame, pixels that belong to moving objects. The motion detection is performed by tracking each pixel of the current frame, along a group of preceding frames in the space of the thresholded fourth order moment of the differences.

The algorithm [12][21], produces the segmentation map of each frame f_k of the sequence by processing a group of frames $\{f_{k-i}, i=0,..n\}$. The number of frames n varies on the basis of the estimated object velocity [12]. Any global motion component is removed by a preprocessing stage, aligning f_{k-i} to f_k, $j=1,..n$. For each frame f_k, the algorithm splits in three steps (see Figure F.18).

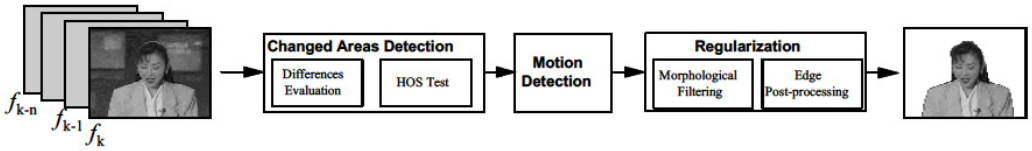


Figure F.18 — Segmentation Using Higher Order moments and motion tracking: Block Diagram

Step One: Changed areas detection. The differences $\{ d_{k-j}(x,y)=f_{k-j}(x,y)- f_{k-n}(x,y), j=0,..n-1 \}$ of each frame of the group with respect to the first frame f_{k-n} are evaluated in order to detect the changed areas, due to object motion, uncovered background and noise. In order to reject the luminance variations due to noise, an Higher Order Statistic test is performed. Namely, for each pixel (x,y) the fourth-order moment of each inter-frame difference $d(x,y)$ is estimated on a 3x3 window (see (23) and (24)), it is compared with a threshold adaptively set on the basis of the estimated background activity [12], and set to zero if it is below the threshold. On the sequence of the thresholded fourth-order moment maps $\tilde{m}^{(4)}_{d_{k-j}}(x,y)$, a motion detection procedure is performed.

$$\hat{m}_d(x,y)=\frac{1}{9}\sum_{(s,t)\in\eta(x,y)} d(s,t) \quad (23), \qquad \hat{m}^{(4)}_d(x,y)=\frac{1}{9}\sum_{(s,t)\in\eta(x,y)} (d(s,t)-\hat{m}_d)^4 \quad (24)$$

Step Two: Motion detection. This step aims at distinguish changed areas representing uncovered background (which stands still in the HOS maps) and moving objects (moving in the HOS maps). At the j-th iteration, the pair of thresholded HOS maps $\tilde{m}^{(4)}_{d_{k-j}}(x,y), \tilde{m}^{(4)}_{d_{k-j-1}}(x,y)$ is examined. For each pixel (x,y) the displacement is evaluated on a 3x3 window, adopting a SAD criterion (Sum of Absolute Differences), and if the displacement is not null the pixel is classified as moving. Then, the lag j is increased (i.e. the pair of maps slides) and the motion analysis is repeated, until $j=n-2$. Pixels presenting null displacements on all the observed pairs are classified as still. Note that, from a computational point of view, at each iteration, only pixels that were not already classified as moving are

© ISO/IEC 2004 – All rights reserved