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.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 synthess filtering operation is defined as follows:

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

where

- n = 0, 1, ... N-1, and N is the number of output points
- L[2*i] = x[i] and L[2*i+1] = 0 for i=0,1,...,N/2-1, and {x[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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/ROOPA V. ADAMANE
ANSI Store order #X37445' Downloaded 7/18/2007 2:58:21 PM ET
Single user license only. Copying and networking prohibited.

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.

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]...

Licensed to UNIVERSITY OF TEXAS AT [illegible]
Single user license only. Copying and networking prohibited.

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 he 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 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:

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

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-samping, 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	TypeB	TypeA
	Trailing	TypeB	TypeA
highpass input xh[] to 9-tap filter h[]	Leading	TypeA	TypeB
	Trailing	TypeA	TypeB

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

Similar to standard 8x8-DCT from subclause A.1, SA-DCT converts a 8x8-block with texure data f[y][x] into a block with corresponding SA-DCT coefficients and, vice versa, inverse SA-DCT converts dequantized 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 descibes 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 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 1180-1990 for visual telephony.

A.3.1 Definiton 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];
      }
   }
}
```

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 Definiton of Inverse SA-DCT

I-S1: computation of required shape parameters 'coeff_width[v]', 'shift_shape[y][x]' and 'pels_height[x]' from decodec 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<8; v++) {
   coeff_count=0;
   for (x=0; x<8; x++)
      if(shift_shape[v][x]==255)
         coeff_count++;
   coeff_width[v]=coeff_count
}
```

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][coeff_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 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 0 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 1180-1990 for visual telephony.

A.4.1 Definiton 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
```

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;
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

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

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"

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..
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

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	--

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

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
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
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 coces 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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

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	VLC CODE	LAST	RUN	LEVEL
10s	0	0	1	0111 s	1	0	1
1111 s	0	0	3	0000 1100 1s	0	11	1
0101 01s	0	0	6	0000 0000 101s	1	0	6
0010 111s	0	0	9	0011 11s	1	1	1
0001 1111 s	0	0	10	0000 0000 100s	1	0	7
0001 0010 1s	0	0	13	0011 10s	1	2	1
0001 0010 0s	0	0	14	0011 01s	0	5	1
0000 1000 01s	0	0	17	0011 00s	1	0	2
0000 1000 00s	0	0	18	0010 011s	1	5	1
0000 0000 111s	0	0	21	0010 010s	0	6	1
0000 0000 110s	0	0	22	0010 001s	1	3	1
0000 0100 000s	0	0	23	0010 000s	1	4	1
110s	0	0	2	0001 1010 s	1	9	1
0101 00s	0	1	2	0001 1001 s	0	8	1

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

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

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

VLC CODE	LAST	RUN	LEVEL
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
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		

Table B-17 – VLC table for Inter Luminance and Chrominance TCOEF

VLC CODE	LAST	RUN	LEVEL
10s	0	0	1
1111 s	0	0	2
0101 01s	0	0	3
0010 11s	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
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		

0001 1011 s	0	14	1
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	LAST	RUN	LMAX
0	0	27	1	0	8
0	1	10	1	1	3
0	2	5	1	2-6	2
0	3	4	1	7-20	1
0	4-7	3	1	others	N/A
0	8-9	2			
0	10-14	1			
0	others	N/A			

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

LAST	RUN	LMAX	LAST	RUN	LMAX
0	0	12	1	0	3
0	1	6	1	1	2
0	2	4	1	2-40	1
0	3-6	3	1	others	N/A
0	7-10	2			
0	11-26	1			
0	others	N/A			

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

LAST	LEVEL	RMAX	LAST	LEVEL	RMAX
0	1	14	1	1	20
0	2	9	1	2	6
0	3	7	1	3	1
0	4	3	1	4-8	0
0	5	2	1	others	N/A
0	6-10	1			
0	11-27	0			
0	others	N/A			

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

LAST	LEVEL	RMAX	LAST	LEVEL	RMAX
0	1	26	1	1	40
0	2	10	1	2	1
0	3	6	1	3	0
0	4	2	1	others	N/A
0	5-6	1			
0	7-12	0			
0	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

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	6	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

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	5	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

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

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	8	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

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

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DIVYA K ADAPANE
ANSI Store order BX27445! Downloaded 7/15/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

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
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

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,
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAVANE
Single user license only. Copying and networking prohibited.

```
32768,16384,32768,21845,32768,32768,32768,32768,32768,32768,32768,5041,2440,91,32768,45,
65124,10923,64874,5041,65429,57344,63435,48060,51440,32768,63488,24887,59686,3277,63918,14021,
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,
1481,438,19661,840,33654,3121,64425,6554,4178,2048,32768,2260,5226,1060,32768,565,
60075,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,32768,
16384,261,32768,412,16384,636,32768,4360,23406,4328,32768,524,15604,560,32768,676,
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,
39862,15338,32768,1524,45123,5958,32768,58982,6869,930,1170,1043,7385,44,6813,5011,
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,
62805,32768,62013,42612,32768,32768,61681,16384,58982,60075,62313,58982,65279,58982,62894,62174,
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,
62873,37268,65374,6197,62017,32768,49152,299,54613,32768,32768,32768,35234,119,32768,3855,
31849,32768,32768,49152,16384,32768,32768,32768,24576,32768,49152,32768,17476,32768,32768,57445,
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,
63351,3855,63608,29127,62415,7282,64625,60855,32768,32768,60075,5958,44961,32768,61866,53718,
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,
41764,2161,21845,1836,17284,5424,10923,1680,11019,555,32768,431,39819,807,32768,171,
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,
61885,13827,60893,12196,60855,53248,51493,11243,56656,763,55563,143,63432,7106,52429,445,
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,
64540,57301,49152,20025,63351,61167,32768,45542,58982,14564,32768,9362,61895,44840,32768,26385,
59664,17135,60855,13291,40050,12252,32768,7818,25798,1850,60495,2662,18707,122,52538,231,
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,
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
Single user license only. Copying and networking prohibited.

```
65416,48091,65292,58328,64626,32768,65016,39897,62687,47132,62805,28946,64384,53620,52870,49587,
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,
63000,9830,32560,58982,11565,32768,25215,3277,53620,50972,63109,43691,54613,32768,39671,17129,
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,
58052,23400,32768,5041,32768,2849,32768,32768,47653,15073,57344,4096,32768,1176,32768,1320,
24858,410,24576,923,32768,16384,16384,5461,16384,1365,32768,5461,32768,5699,8192,13107,
46884,2381,23559,424,19661,712,655,182,58637,2094,49152,9362,8192,85,32768,1228,
65486,49152,65186,49152,61320,32768,57088,25206,65352,63047,62623,49152,64641,62165,58986,16384,
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 V2 - 42 – 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, 50441, 50140, 63343,
 58093, 45146, 35768, 13351, 17594, 28777, 39830, 38719,  9768, 21447,
 12340,  9783, 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,
 62436, 56971, 51738, 59757, 49927, 53675, 38182, 24724, 48447, 47316,
 55025, 36330, 12204, 25093, 24243, 5358, 56717, 55046, 46302, 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, 58055, 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-33 – 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...11111
-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	-

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
ANSI Store order X[illegible] Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

Table B-34 -- Code table for scaled brightness change factor

brightness_change_ factor value	brightness_ change_factor_ length value	brightness_ change_factor_ length VLC	brighiness_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-35 -- Viseme_seect_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-36 --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-37 -- 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
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			

Table B-39 -- 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

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

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/GOOJA Y. ADAMANE
ANSI Store order #X27445; Downloaded 7/18/2007 2:58:21 PM ET
Single user license only. Copying and networking prohibited.

93	16	1101011100010111	264	7	1101110	435	17	11010111110101110
94	17	11010111111110101	265	8	11001101	436	17	11010111110101100
95	17	11010111111110111	266	8	11001110	437	17	11010111110100110
96	17	11010111110110111	267	8	11001100	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	110011110	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

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-40 – 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

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

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

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	9	100000110	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

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

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_S_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, 24822, 58944, 1040, 888, 47206, 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, 19038, 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, 20096, 121, 3, 65536, 65536, 65536, 65536, 65536, 65536, 65536,
65536, 1950, 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, 6645, 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, 213, 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, 19835, 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,
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGARWAL
Single user license only. Copying and networking prohibited.

```
65536, 65536, 1, 1, 1, 172, 2, 4, 793, 342, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536,572, 1,
1928, 43080, 3337, 1680, 1401, 2131, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536,18, 1, 386,
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, 61951, 29954, 4826, 6481, 12288, 2410, 4466, 228, 65537, 65537, 65537,
65537, 65537, 65537, 65537, 65537, 1789, 23089, 726, 7470, 30386, 28721, 9811, 1446, 65537, 65537, 65537,
65537, 65537, 65537, 65537, 65537, 45333, 28672, 21363, 26870, 41125, 9455, 25752, 12372, 65537, 65537,
65537, 65537, 65537, 65537, 65537, 65537, 48324, 37071, 4994, 5138, 18879, 28687, 9330, 366, 65537, 65537,
65537, 65537, 65537, 65537, 65537, 65537, 31745, 26116, 346, 397, 2900, 13830, 381, 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, 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, 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, 58482, 65535, 1, 1, 11805, 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, 18805, 13107, 1039, 1214, 5060, 21845, 3830, 387, 65537, 65537, 65537, 65537, 65537, 65537,
65537, 65537, 31654, 32951, 490, 1498, 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;
}
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON [illegible]
Single user licence only. Copying and networking prohibited.

```
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 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;

    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 simply 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 3
```

B.2.2.2 Decoder Source

```
Void StartArDecoder(ArDecoder *decoder, Bitstream *bitstream) {
  Int i,j;
  decoder->V = 0;
  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;
    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(UInt 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 positon in the bitstream and returns the bit.
- BitstreamFlushBits(bitstream,nbits) : Moves the current bitstream position forward by nbits.

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 zeo. 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);
}
```

Again the model is specified through `cumul_freq[ ]`. The decoded symbol is returned through its index in the model. The tecoder is intialized 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;
}
```

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-orBidir	Pos motion	Grp	FDP subgrp num	Quant step size	Min/Max I-Frame quantized values	Min/Max P-Frame quantized 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	MNS	B	down	2	6	2	+-600	+-180

		lip								
9	lower_t_lip_rm	Vertical displacement of midpoint between right corner and middle of top inner lip	MNS	B	down	2	7	2	+-600	+-180
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

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
43	shift_tongue_tip	Horizontal displacement of	MW	B	right	6	1	1	+-1080	+-420

		tongue tip								
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
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_cornerlip_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

In the following, each facial expression is defined by a textual description and a pictorial example. (reference [10], page 114.) This reference was also used for the characteristics of the described expressons.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/JOHN ADAMS
ANSI Store order X027445 Downloaded 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

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.

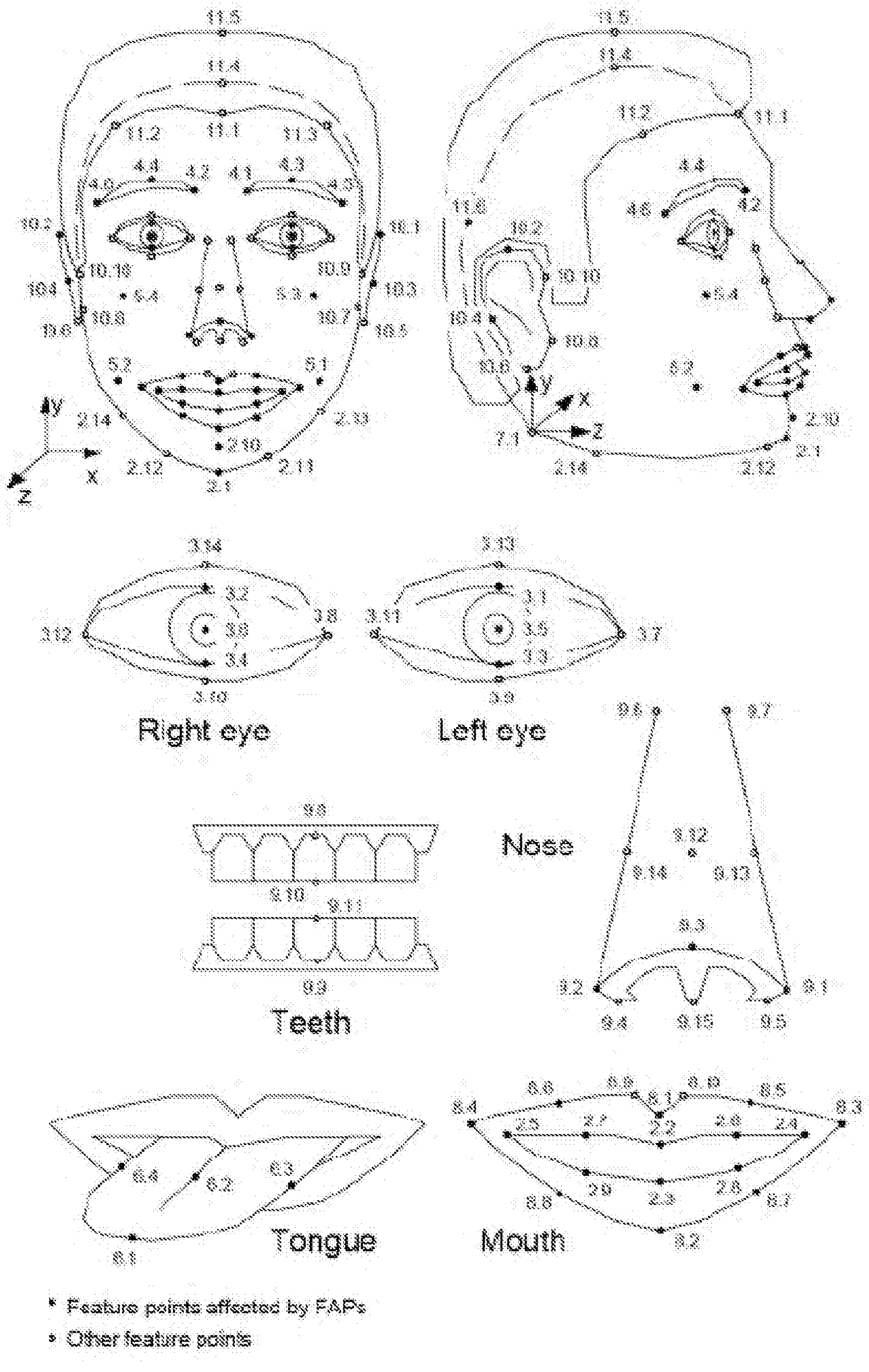


Figure C-1 – FDP feature point set

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			

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA L ADAMANE
ANSI Store order [illegible] Downloaded: 7/16/2007 [illegible] PM ET
Single user license only. Copying and networking prohibited.

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	> 3.4.x and < 3.7.x	> 9.15.y and < 9.12.y	
5.4	Right cheek bone	> 3.4.x and < 3.12x	> 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 high point of Cupid's bow			
8.10	Left high 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			

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 he **point** field in the **Coordinate** node in the prescribed order, tha 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 **texureType** 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 hat the texture image is obtaned by orthographic projection of the face.
faceDefTables	contains **faceDefTables** nodes. The behavio 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**.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAMANE
ANSI Store order X027445; Downloaded 7/16/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

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**	**Quant step size**	**Min/Max I-Frame quantized values**	**Min/Max P-Frame quantized 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
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/ +1200	-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 pane	256	-780/ +780	-600/ +600
23	r_metatarsal_flexion	Up and dows rotation of the toe in the sagittal pane	256	-780/ +780	-600/ +600
24	l_sternoclavicular_abduct	Up and dows motion in the coronal plane	128	-60/ +240	-120/ +120
25	r_sternoclavicular_abduct	Up and dows motion in the coronal plane	128	-240/ +60	-120/ +120
26	l_sternoclavicular_rotate	Rotation in the transverse plane	128	-120/ +120	-60/ +60
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_vrist_flexion	Rotation of the hand in the coronal plane	128	-960/ +960	-600/ +600
43	r_vrist_flexion	Rotation of the hand in the coronal plane	128	-960/ +960	-600/ +600
44	l_vrist_pivot	Rotation of the hand in the sagittal planes	128	-660/ +660	-360/ +360
45	r_vrist_pivot	Rotation of the hand in the sagittal planes	128	-660/ +660	-360/ +360
46	l_vrist_twisting	Rotation of the hand along the forearm axis	256	-660/ +660	-360/ +360
47	r_vrist_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	vcl_roll	Sideward motion of vertebra C1	256	-240/ +240	-120/ +120
52	vcl_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
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 insagittal plane	128	-660/ +660	-360/ +360
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
127	l_pinky1_pivot	Lateral mobility of the pinky finger	128	-420/ +120	-120/ +120

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAWANE
Single user license only. Copying and networking prohibited.

128	r_pinky1_pivot	Lateral mobilityof the pinky finger	128	-120/ +420	-120/ +120
129	l_pinky1_twisting	Along the pinky finger axis	256	-180/ +300	-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 mobilityof the middle finger	128	-180/ +180	-120/ +120
152	r_middle1_pivot	Lateral mobilityof 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 fexing mobility of the index finger	256	-60/ +60	-60/ +60
160	r_index0_flexion	Metacarpal fexing 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
177	l_thumb2_flexion	Second knuckle of the thumb number	128	-780/ +60	-240/ +240

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/FODJA I. ADAMANE
ANSI Store order X…; Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

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 14496-1:1999/Amd.1:2000 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 (32,34,36,3842)
7	Right arm1	r_shoulder_flexion, r_shoulder_abduct, r_shoulder_twisting, r_elbow_flexion, r_wrist_flexion (33,35,37,3943)
8	Left arm2	l_sternoclavicular_abduct, l_sternoclavicular_rotate, l_acromioclavicular_abduct, l_acromioclavicular_rotate, l_elbow_twisting, l_wrist_pivot, l_wrist_twisting (24,26,28,3040,44,46)

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, 105,106,137,117,118,119)
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)

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, 162,166,170 172 174,178,180)
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 (123,127,128,135,139,141,147, 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)

Additionaly, 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.

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	-0.0231
l_subtalar	0.0956	0.0398	0.0069
l_midtarsal	0.1079	0.0317	0.0670
l_metatarsal	0.0942	0.0092	0.1229
r_hip	-0.0956	0.9364	0.0000
r_knee	-0.0956	0.5095	-0.0036
r_ankle	-0.0946	0.0762	-0.0231
r_subtalar	-0.0956	0.0398	0.0069
r_midtarsal	-0.1079	0.0317	0.0670
r_metatarsal	-0.0942	0.0092	0.1229
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	-0.0732
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	1.4599	-0.0587
vt3	0.0000	1.4832	-0.0492
vt2	0.0000	1.5011	-0.0397
vt1	0.0000	1.5201	-0.0330
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	0.0030
l_sternoclavicular	0.0757	1.4844	-0.0251
l_acromioclavicular	0.0899	1.4525	-0.0322
l_shoulder	0.1968	1.4642	-0.0235
l_elbow	0.1982	1.1622	-0.0557
l_wrist	0.1972	0.8929	-0.0630
l_thumb1	0.1912	0.8734	-0.0657
l_thumb2	0.1912	0.8156	-0.0079
l_thumb3	0.1912	0.8097	0.0070

l_index0	0.1912	0.8259	-0.0460
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	0.8096	-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	-0.1983	1.1622	-0.0657
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	0.8096	-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

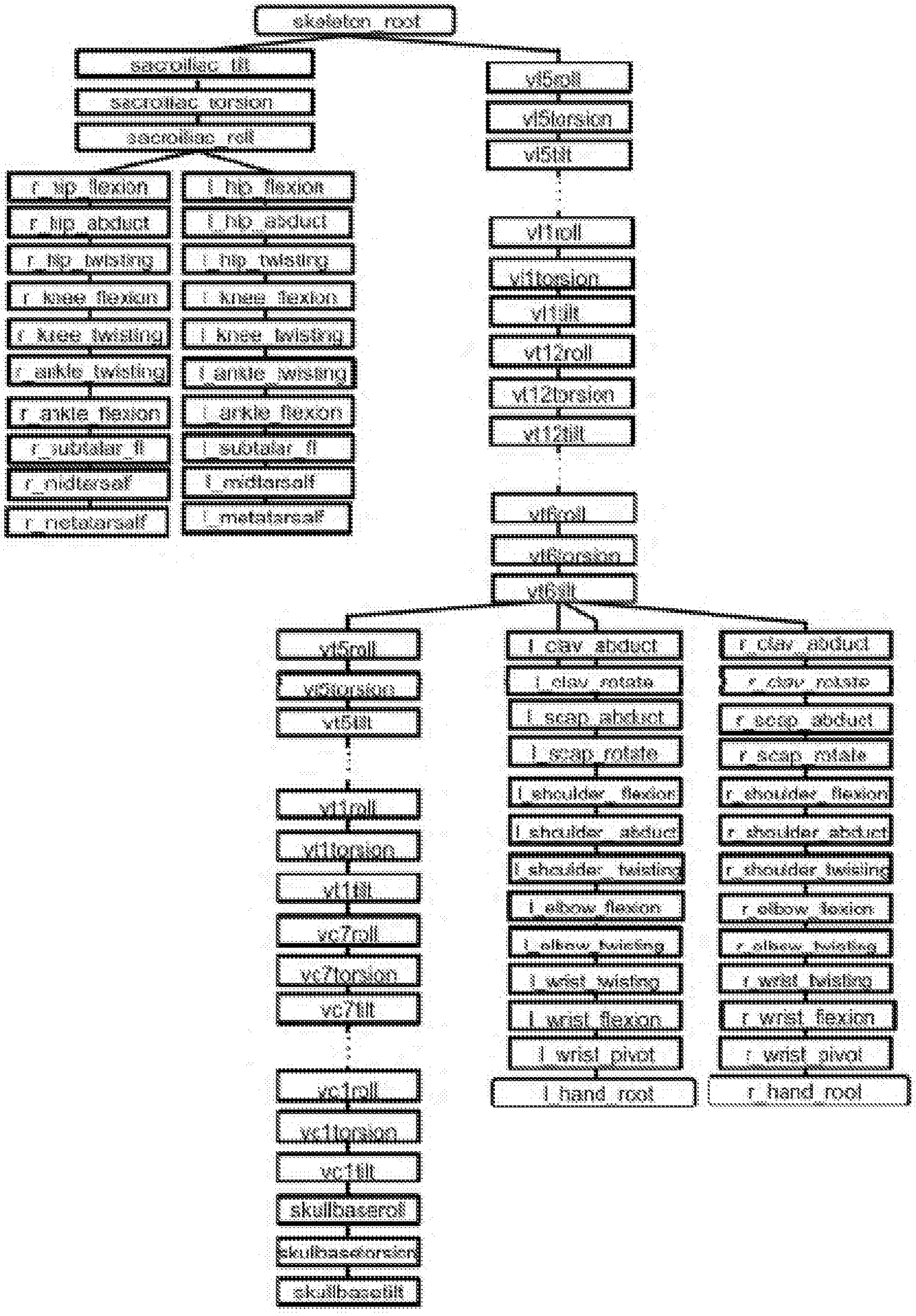


Figure V2 -58 – Body topology

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 order :

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 literatue).

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

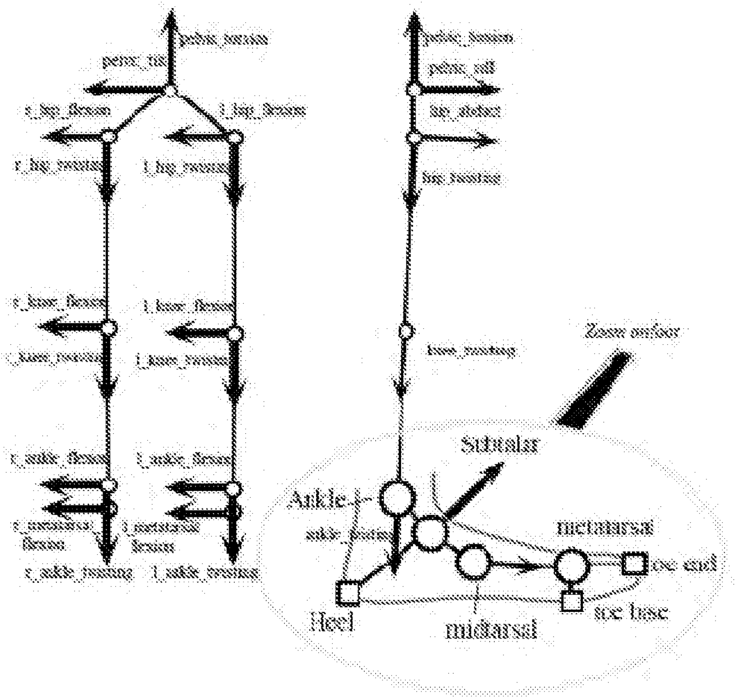


Figure V2 -40 -- Front and side views of the mobilities of the leg

Upper Body

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

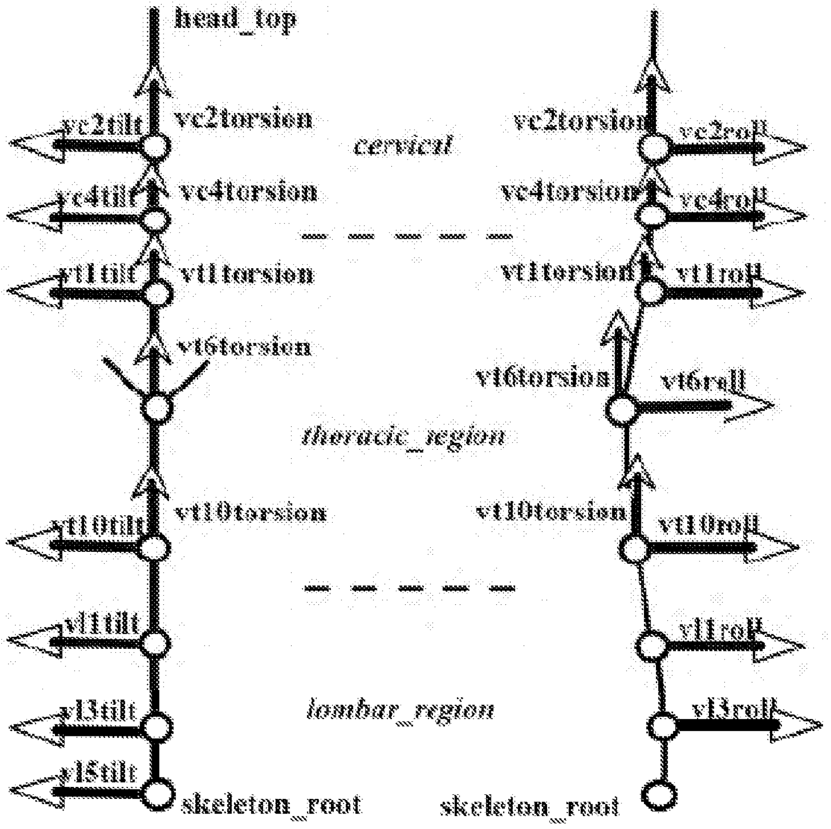


Figure V2 -80 -- Front and side views of the mobilities of the simple spine

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.

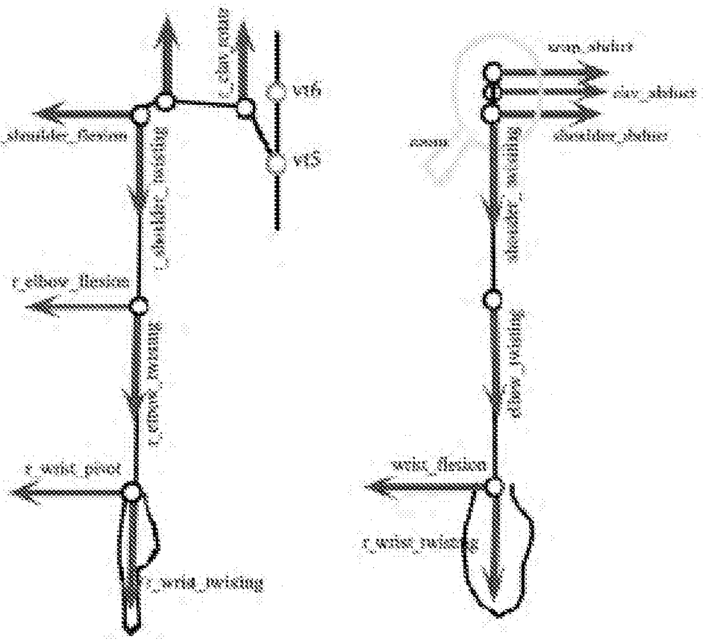


Figure V2 -61 -- Front and side views of the mobilities of the arm in rest position

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

Licensed to UNIVERSITY OF TEXAS AT [illegible]
Single user license only. Copying and networking prohibited.

At the elbow :

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

elbow_twisting : along the arm axis.

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

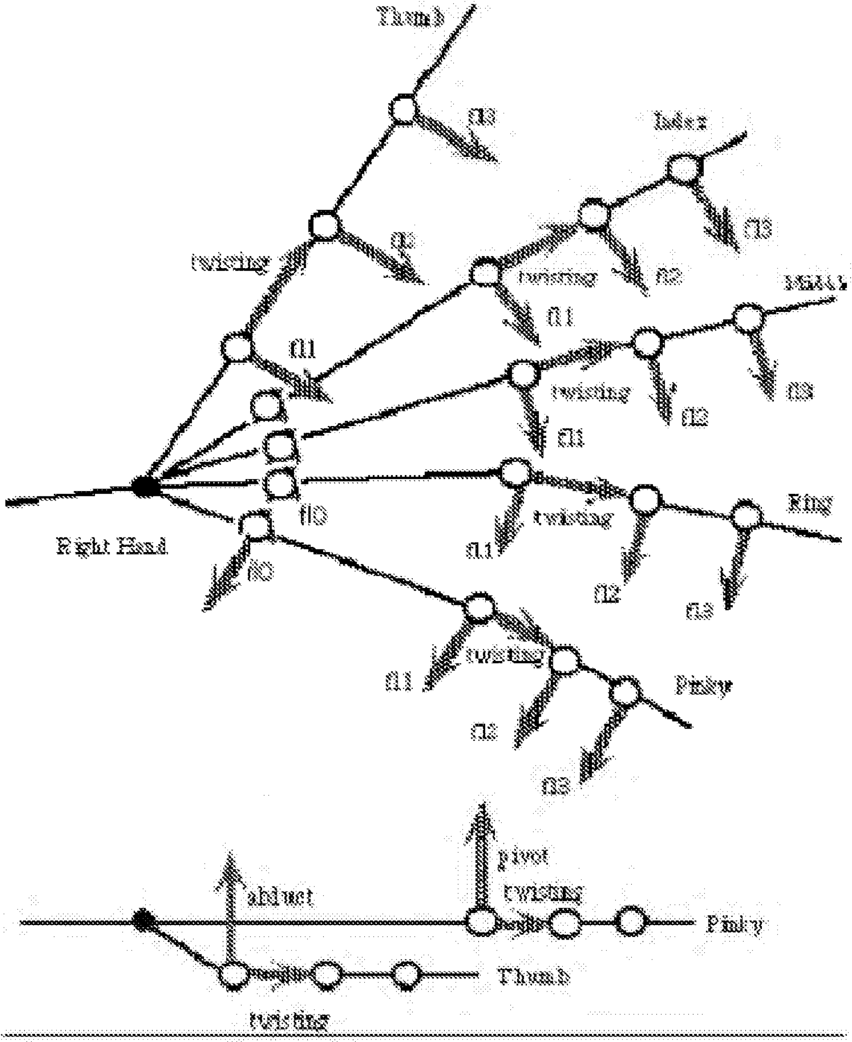


Figure V2 -62 -- Mobilities of the right hand

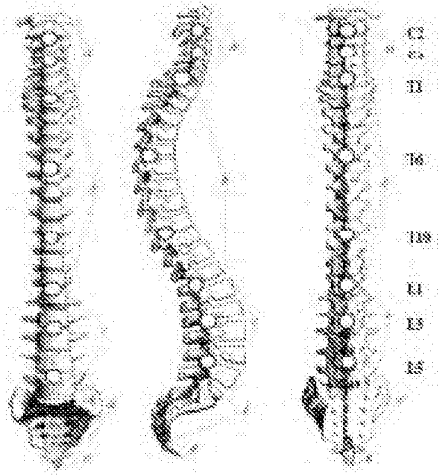


Figure V2 -63 -- Example spine mobilities

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 14496-3:1999.

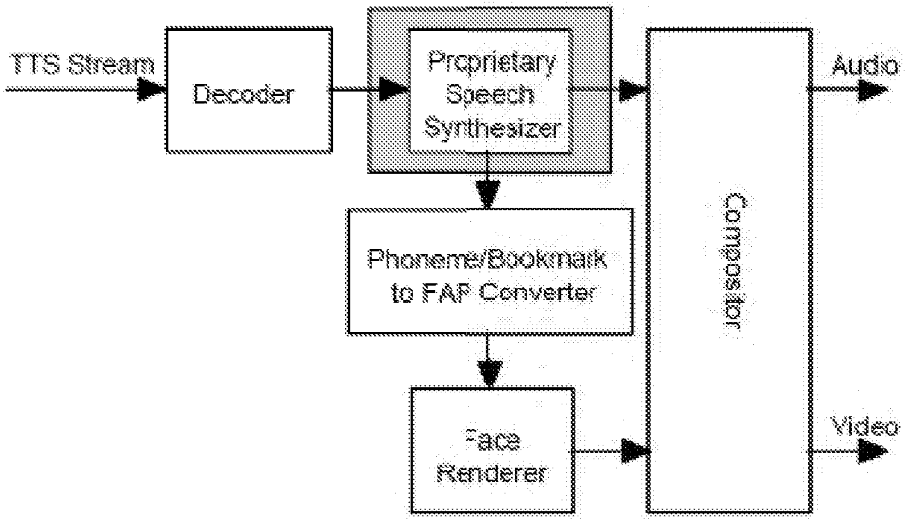

TTS Stream
Decoder
Proprietary Speech Synthesizer
Phoneme/Bookmark to FAP Converter
Face Renderer
Compositor
Audio
Video

Figure C.2: Blockdiagram showing the integration of a proprietary TTS into as ISO/IEC 14496 face animation system with Phoneme/Bookmark to FAP Converter.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
ANSI Store order X[illegible] Downloaded 7/18/2007 [illegible] PM ET
Single user license only. Copying and networking prohibited.

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 14496-1:1999 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.

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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODILA Y. ADAMANE
ANSI Store order X127445 Downloaded: 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

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:

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

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

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

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 guaranteed. 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

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

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

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 decodables with a predetermined buffer memory size.

A video complexity model is required in order to bound he 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.

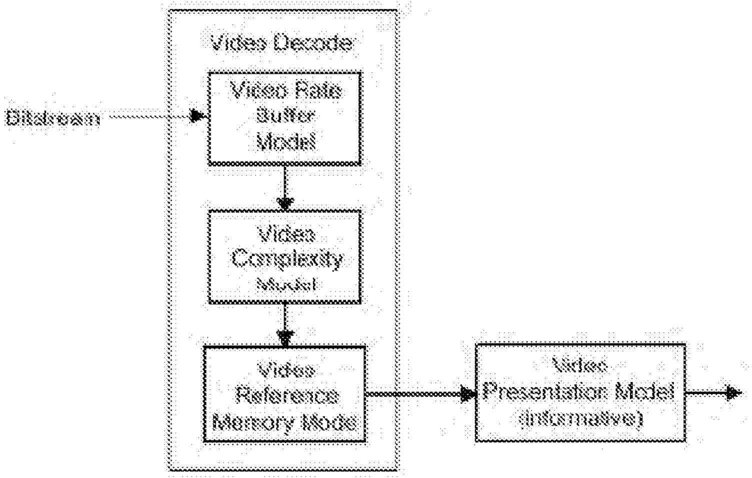


Figure D-1 – Video Verifier Model

D.2 VideoRate 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).

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 × 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 14496-1:1999 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. Generally, 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 module_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.

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

Presentation 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

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

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

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 ($0 \leq k < (N_{sp} + 396) / 396$) 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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
ANSI Store order X307445 Downloaded: 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

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

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

For the inital 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 Rmax 1001) / 30000 bits, where Rmax 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 Rmax 1001) / 30000.

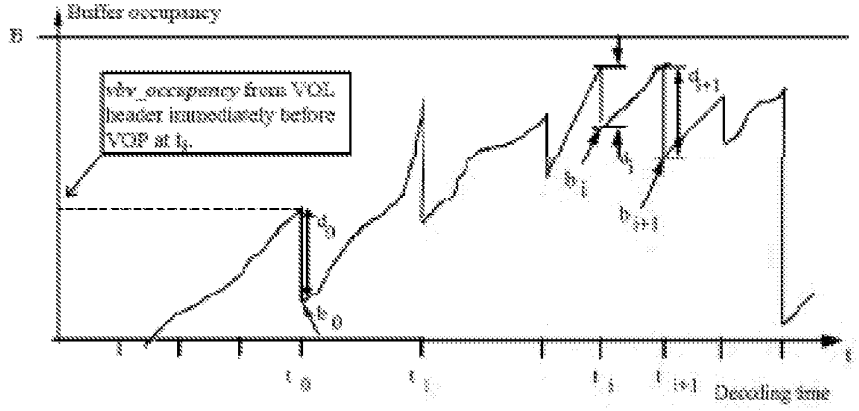


Figure D-2 – VBV Buffer occupancy

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 $R_{vol,decoder}(t)$ 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 $R_{vol,decoder}(t)$ in item 2 above.

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 13818-2:1996 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.

In general, the ISO/IEC 13818-2 bitstream 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 anchor 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 presentation order will be 1001 because picture skipping is not permitted in ISO/IEC 13818-2:1996 (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 compy 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.

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:

   $$MB_{S\text{-}VOP} = (MB_{wrp} > MB_{ref}\ ?\ MB_{wrp} : MB_{ref}) + MB_{res}$$

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODDA,Y. ADAMANE
ANSI Store order X637445. Downloaded 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

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

$MB_{sw} = MB_{vop}$, when no_of_sprite_warping_points == 0 or 1,

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

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

$= (MB_{vop} \times (|(i_1' - i_0')(j_2' - j_0') - (i_2' - i_0')(j_1' - j_0')| + |(i_1' - i_3')(j_2' - j_3') - (i_2' - i_3')(j_1' - j_3')|))/(2 W H s^2)$,
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_{su} 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_{su} 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 ide 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

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

   $v(t_i)$ is the VCV occupancy before the macroblocks representing VOP i, M_i, are added to $v(t_i)$,
   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 $v_B(t_i)$, 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

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

   $v(ts_k)$ is the VCV occupancy before the number of macroblocks in section *k*, MS_k is added to the 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 $v_B(ts_k)$.

5. To decode a VOL, the decoding of each VOP i must be complete by c_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 simply 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_{SR} (see item 2 of subclause D.4 for the definition of MB_{SR}). This memory is consumed at the constant rate of

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

   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, $\tau_i + L$, 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, $\tau_i + L$, 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.

6. At the composition time (or presentation time in a no-compositor decoder) plus VCV latency, $\tau_i + L$, 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, $\tau_i + L$, 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 $\tau_i + L$ + compositor_latency the VOP should be composited.

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

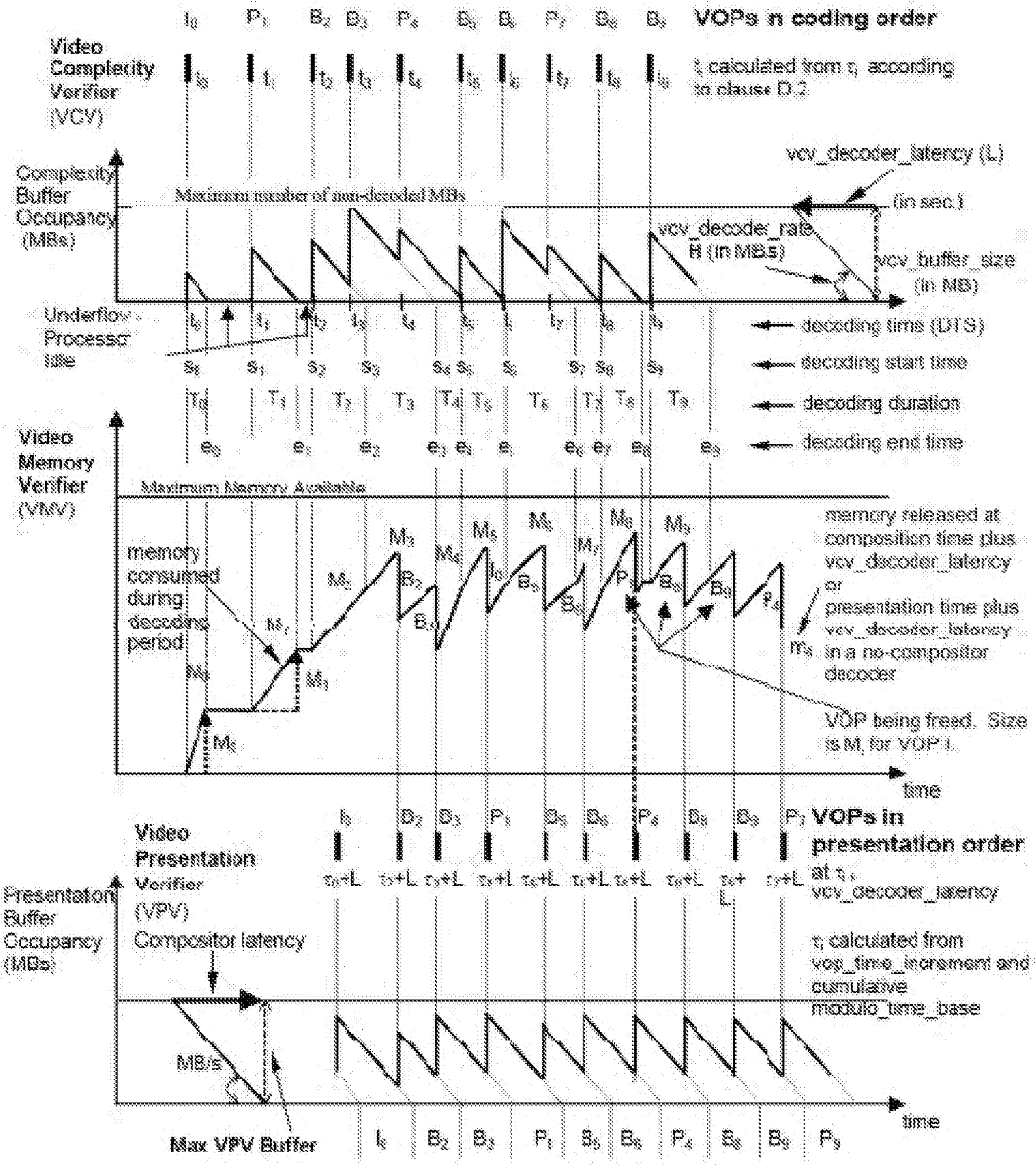

VOPs in coding order
I0 P1 B2 B3 P4 B5 B6 P7 B8 B9
Video Complexity Verifier (VCV)
ti calculated from τi according to clause D.2
vcv_decoder_latency (L)
(in sec.)
Complexity Buffer Occupancy (MBs)
Maximum number of non-decoded MBs
vcv_decoder_rate R (in MB/s)
vcv_buffer_size (in MB)
Underflow - Processor idle
decoding times (DTS)
decoding start time
decoding duration
decoding end time
Video Memory Verifier (VMV)
Maximum Memory Available
memory consumed during decoding period
memory released at composition time plus vcv_decoder_latency or presentation time plus vcv_decoder_latency in a no-compositor decoder
VOP being freed. Size is Mi for VOP i.
time
Video Presentation Verifier (VPV)
Compositor latency
Presentation Buffer Occupancy (MBs)
VOPs in presentation order
at τi + vcv_decoder_latency
τi calculated from vop_time_increment and cumulative modulo_time_base
MB/s
Max VPV Buffer
I0 B2 B3 P1 B5 B6 P4 B8 B9 P7
time

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

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/ROOJA Y. ADAMANE
ANSI Store order #X274451 Downloaded 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

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 he ability of other types of tools that recover data and/or conceal the effects of errors s 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 informaion contains a GOB start coce, 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 he 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 quantisation parameter (quant_scale) necessary to decode that first macroblock. The macroblock number provides the necessary spatial resynchronization while the quantisation parameter 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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. [illegible]
Single user license only. Copying and networking prohibited.

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 informaton 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 Maker	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

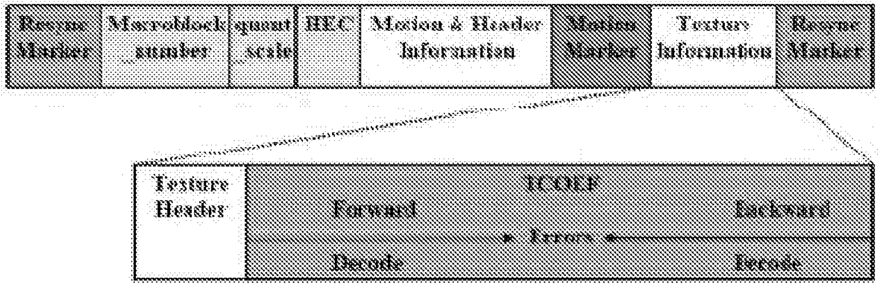


Figure E-3 – Reversible VLC

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 he 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 thedecoded value for LEVEL is zero

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT [illegible]
Single user license only. Copying and networking prohibited.

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 (eg. 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 backwad 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.

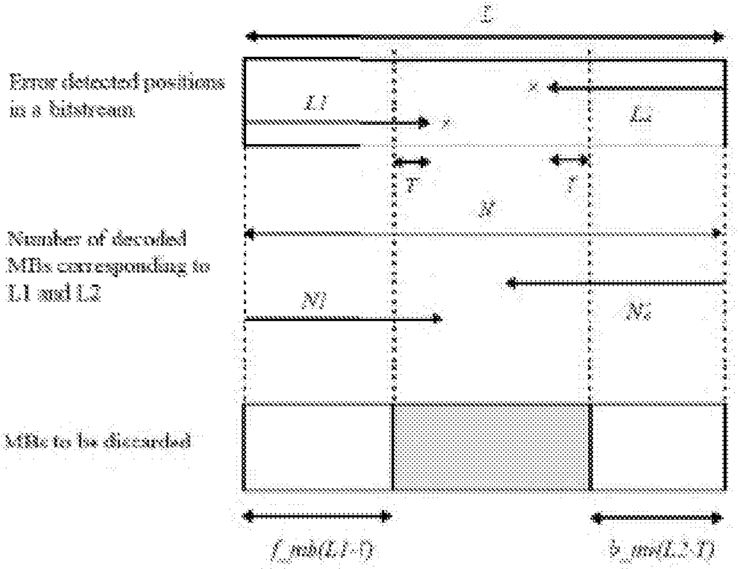

L
Error detected positions in a bitstream
L1
x
x
L2
T
T
N
Number of decoded MBs corresponding to L1 and L2
N1
N2
MBs to be discarded
f_mb(L1-T)
b_mb(L2-T)

Figure E-4 -- RVLC decoding; Strategy 1

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA Y. ADAVANE
ANSI Store order X[illegible] Downloaded 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

(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.

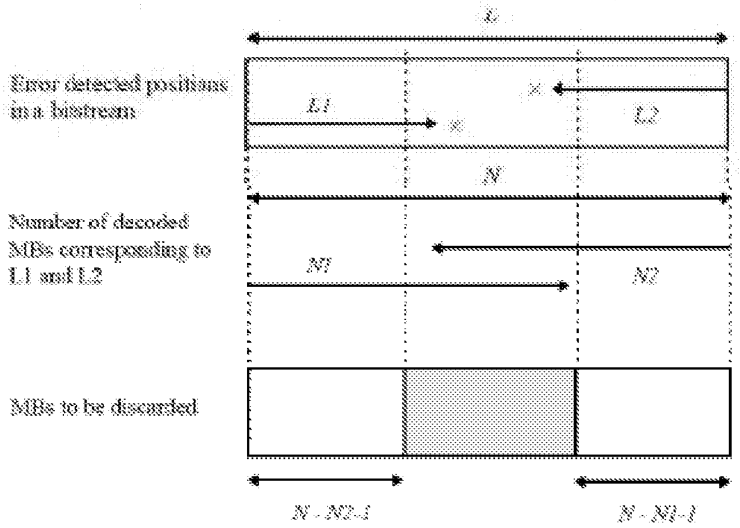

L
Error detected positions in a bitstream
L1
L2
N
Number of decoded MBs corresponding to L1 and L2
N1
N2
MBs to be discarded
N - N2-1
N - N1-1

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.

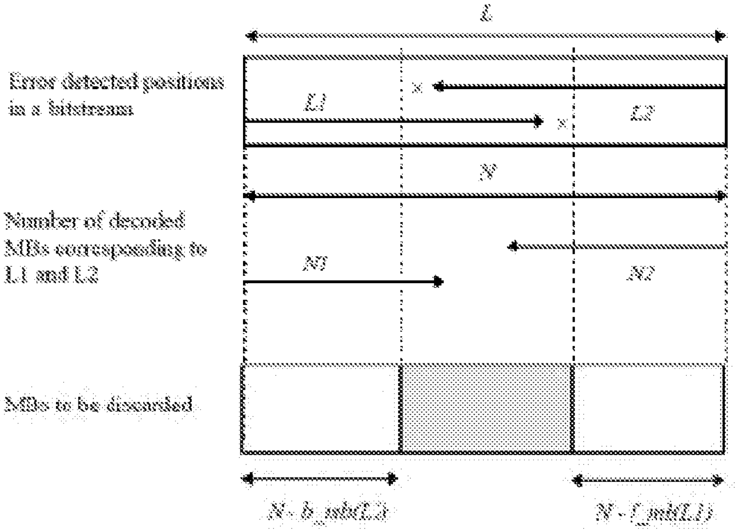

L
Error detected positions in a bitstream
L1
L2
N
Number of decoded MBs corresponding to L1 and L2
N1
N2
MBs to be discarded
N - b_mb(L2)
N - f_mb(L1)

Figure E-6 – RVLC decoding; Strategy 3

(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.

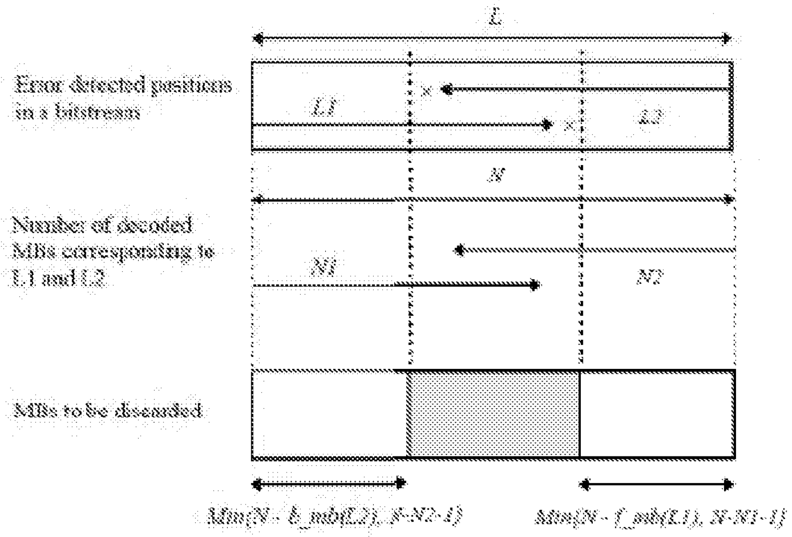

L
Error detected positions in a bitstream
L1
L2
N
Number of decoded MBs corresponding to L1 and L2
N1
N2
MBs to be discarded
Min(N - b_mb(L2), N-N2-1)
Min(N - f_mb(L1), N-N1-1)

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.

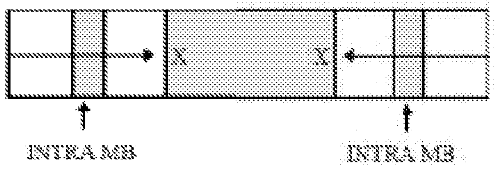


Figure E-8 -- Intra MB discarding

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.

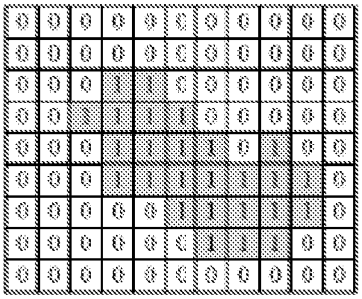
NOTE This informative technique is used for the rectangular VOP. It was designed in order to extract the motion area from the rectangular VOP at the encoder and encode it in intra mode frequently. Therefore, the quality of this area that is corrupted by the transmission error can be recovered quickly. 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.

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

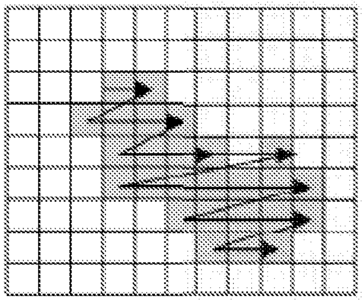


Figure E-10 -- Scan order for the Adaptive Refresh

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].

[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

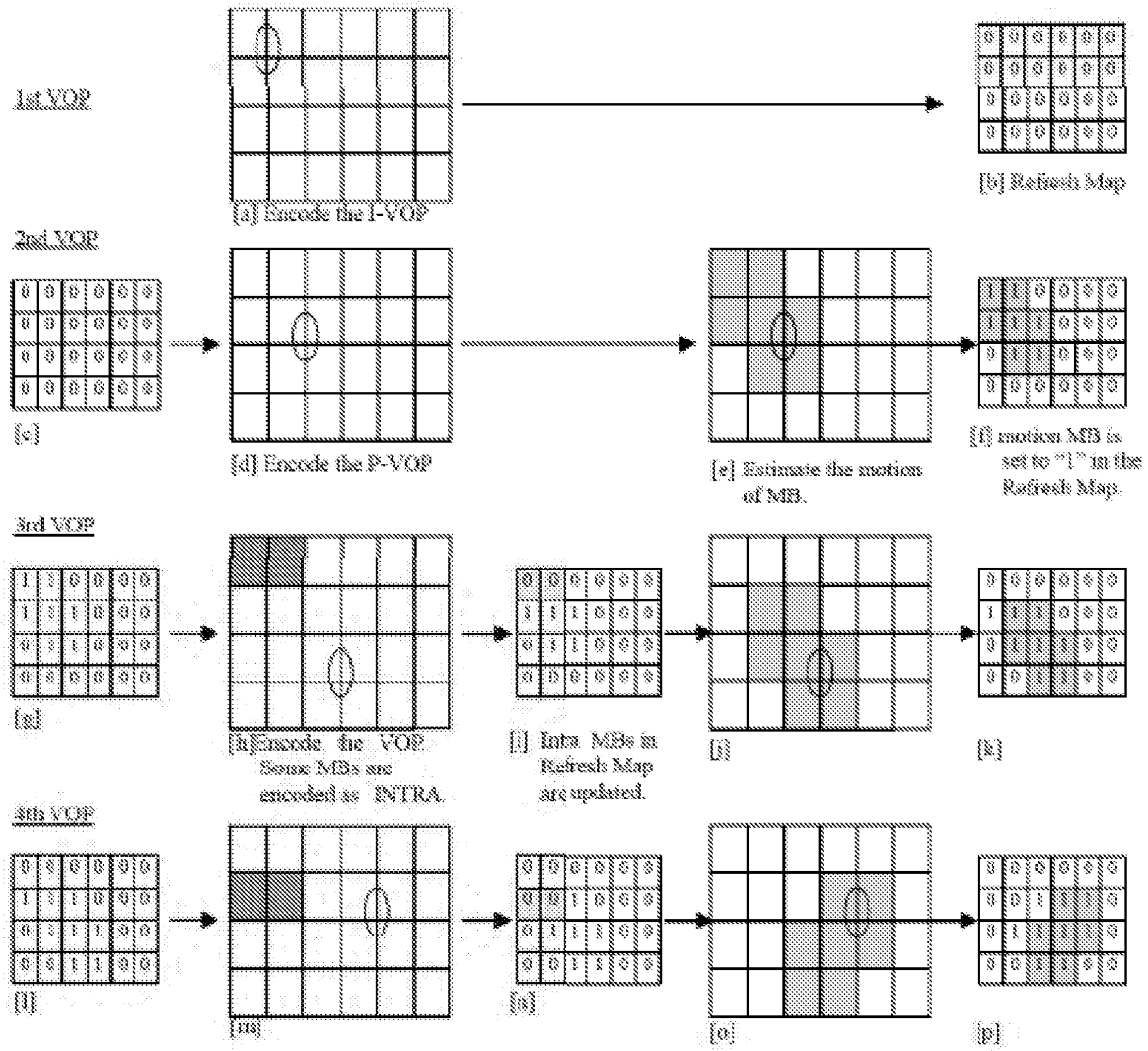

1st VOP
[a] Encode the I-VOP
[b] Refresh Map
2nd VOP
[c]
[d] Encode the P-VOP
[e] Estimate the motion of MB.
[f] motion MB is set to "1" in the Refresh Map.
3rd VOP
[g]
[h]Encode the VOP. Some MBs are encoded as INTRA.
[i] Intra MBs in Refresh Map are updated.
[j]
[k]
4th VOP
[l]
[m]
[n]
[o]
[p]

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'.

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 14496-1:1999 (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 either 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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
ANSI Store order X037445 Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

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 he 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:

video_object_layer_start_code, and

short_video_start_marker

The automatic detection of which of the 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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
Single user license only. Copying and networking prohibited.

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 sequene. Hence, segmentation is a key issue in efficiently applying the ISO/IEC 14496-2 coding scheme, although not afecting 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 shoud 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 same 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.

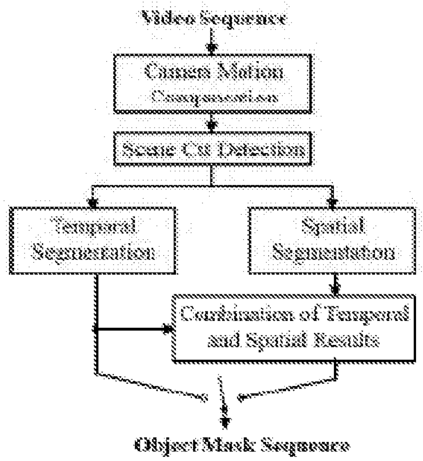


Figure F-1 -- Block diagram of combined temporal and spatial segmentation framework

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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

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 sequence. The scene cut detector described here evaluates whether the difference between the current original image $I_{(k,n)}$ and the camera motion compensated previous image exceeds a given threshold. The evaluation is only perfomed 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 CDN (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 hat 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 distibuted with n^2 degrees of freedom (Eq. (2)).

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

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. 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:

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

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 modeled 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.

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

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 V2 -64:

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

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 V2 -64. 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 V2 -64 – Neighbourhood of pel *X* and potentials *B* and *C* of the neighbours

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/KOO,JA Y. ADAMANE
ANSI Store order X027445' Downloaded 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

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:

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

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 $(\mathrm{CDMs}_{(k+1)}(i)=0)$. 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 σ_c^2 are needed. While in [1] always the CDMi is used for calculating the variances σ_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 σ_c^2 within the object regions of the OM. Thus, the variance σ_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:

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

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 objec regions powered by 4. The second term decreases exponentially with he size of the object regions. While the constants c_{L_1} and c_{L_3} are for normalisation to different image formats, c_{L_2} is a weighting factor.

The maximum depth of MEN L_{max} is set to 34. While in case of QCIF sequences $c_{L_1}=1$ and $c_{L_3}=1$, in case of CIF sequenies $c_{L_1}=15$ and $c_{L_3}=4$. The weighting factor c_{L_2} is set to 80.

F.1.1.2.3.4 Simplification of the CDM

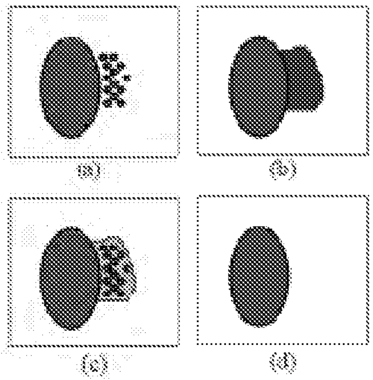


Figure V2 -65 – Example for elimination of small regions using a ternary mask:
(a) CMD with a large region neighboured by smaller regions caused by noise;
(b) CMD after elimination of small regions without ternary mask;
(c) Ternary mask;
(d) CMD after elimination of small regions using a ternary mask

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 is 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 V2 -65a where a CDM is shown containing a large region, which is neighboured by many small regions caused by noise. Figure V2 -65b 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 V2 -65c. The CDM after elimination of small regions by using a ternary mask is presented in Figure V2 -65d. 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 CMI is calculated by detecting and eliminating uncovered background, taking into account the displacement for pels within the calculaed CDM. Therefore a DPV 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.

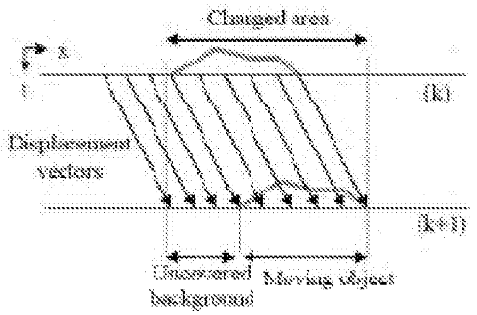


Figure V2 -66 -- Separation of changed areas into foreground and uncovered background

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 V2 -66, 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.

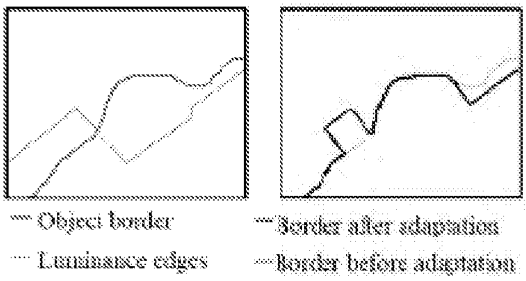


Figure V2 -67 -- Exemplary result of the edge adaptation algorithm

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 $R = 2$ around that pel.

In Figure V2 -67, 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 watershel algorithm [5]

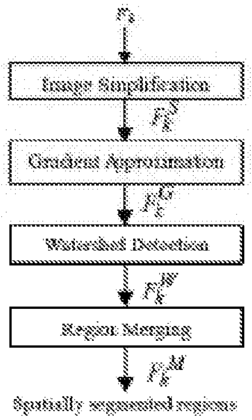


Figure V2 -68 -- Spatial segmentation process

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 V2 -68 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.

$$\varepsilon_n(F_k(x,y)) = \min(F_k(x+l,y+m)), \quad \forall(l,m)\in B_n \qquad (14)$$

$$\delta_n(F_k(x,y)) = \max(F_k(x-l,y-m)), \quad \forall(l,m)\in B_n \qquad (15)$$

Morphological opening and closing operations are given by

$$\gamma_n(F_k) = \delta_n(\varepsilon_n(F_k)) \qquad (16)$$

$$\varphi_n(F_k) = \varepsilon_n(\delta_n(F_k)) \qquad (17)$$

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 elemens while $\varphi_n(F_k)$ eliminates the dark components compared to the structuring components. For the case of the images containing fluctuating scae 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 simplificaton [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 reconstrucion 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 V2 -70b.

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 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 V2 -70c.

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 tesselliation of the input image into its different catchment basins.

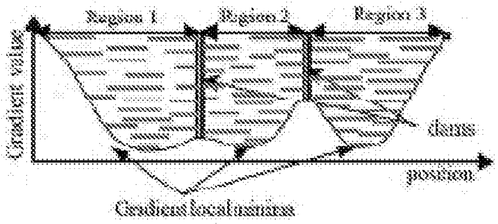


Figure V2 -69 – Watershed flooding

Figure V2 -69 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 he 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.

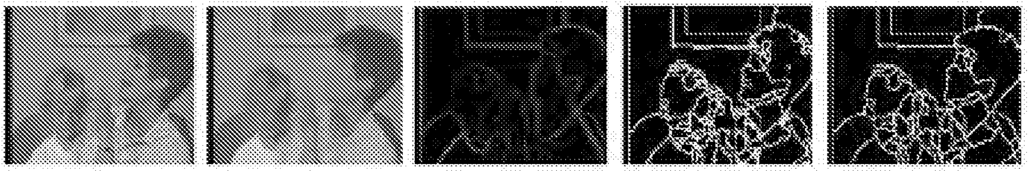
The watershed algorithm is highly sensitive to gradient noise, which yields many catchment basins, so resulting in over-segmentation. To solve this problem, the region merging algorithm based on intensity homogeneity is incorporated. The image after the watershed detection is shown in Figure V2 -70d.

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 ther 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_d) of a region under consideration between two successive frames and the average intensity (R_I) of the region in the current frame.

$$T = \beta R_d + (1 - \beta) R_I \qquad (22)$$

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 $\beta = 0.2$ 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 V2 -70e.

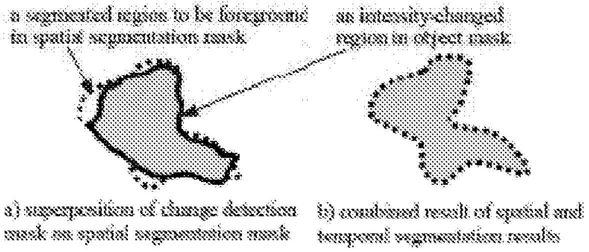


a) original image b) simplified image c) gradient image d) watershed image e) merged image

Figure V2 -70 – 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 spatialy segmented regions with object masks obtained form temporal segmentation in the image. A spatial region ratic and temporal region ratio are computed o make judgement on FG/BG decision.


a segmented region to be foreground in spatial segmentation mask
an intensity-changed region in object mask

a) superposition of change detection mask on spatial segmentation mask b) combined result of spatial and temporal segmentation results

Figure V2 -71 – 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 V2 -71 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 V2 -71b 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.

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.

Licensed to UNIVERSITY OF TEXAS AT [illegible]
Single user license only. Copying and networking prohibited.

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 V2 -72, 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.

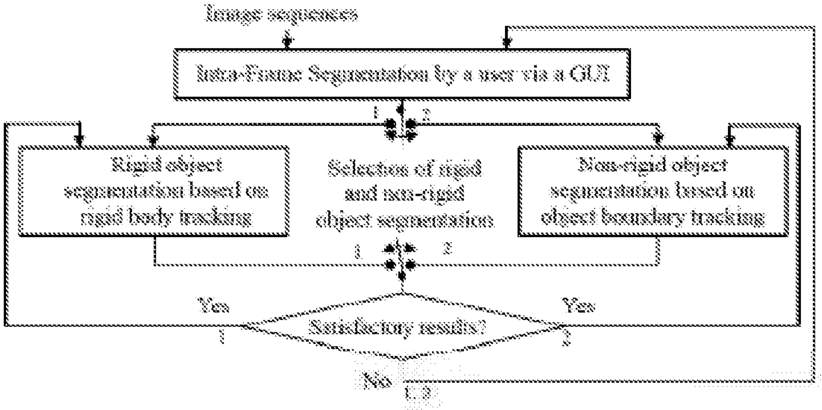


Image sequences
Intra-Frame Segmentation by a user via a GUI
1
2
Rigid object segmentation based on rigid body tracking
Selection of rigid and non-rigid object segmentation
Non-rigid object segmentation based on object boundary tracking
1
2
Yes
Yes
Satisfactory results?
1
2
No
1, 2

Figure V2 -72 -- The block diagram of a user-assisted video object segmentation method

F.1.2.1.1 Intra-Frame Segmentation

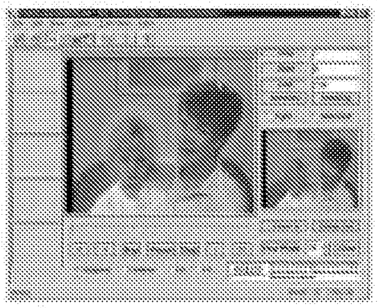


Figure V2 -73 -- A graphical user interface for supporting intra-frame segmentation

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 V2 -73 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 V2 -73.

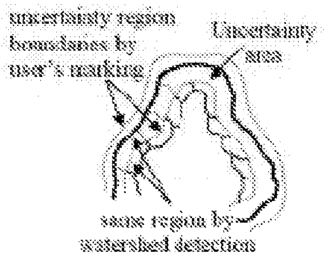

uncertainty region boundaries by user's marking
Uncertainty area
mask region by watershed detection

Figure V2 -74 -- Findings of object boundaries with in the uncertainty area

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 V2 -75 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.

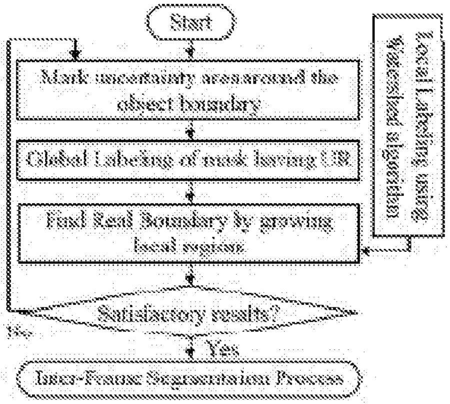

Start
Mark uncertainty area around the object boundary
Global Labeling of mask having UR
Find Real Boundary by growing local regions
Local Labeling using Watershed algorithm
Satisfactory results?
No
Yes
Inter-Frame Segmentation Process

Figure V2 -75 -- Intra-Frame Segmentation Process

The spatially segmented regions are labeled with integers, which forms a locally labeled image. In Figure V2 -75, 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.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
Single user license only. Copying and networking prohibited.

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 V2 -76. 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.

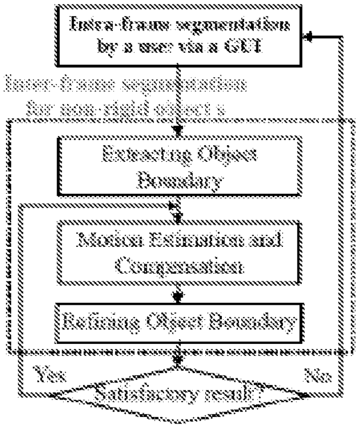


Figure V2 -76 – Inter-frame segmentation process for non-rigid object

$$MAD(u,v) = \sum_{i=-N/2}^{N/2} \sum_{j=-N/2}^{N/2} \left| S(i+u, j+v) - R(i,j) \right| \times \alpha(i,j)$$

$$MV = (u,v)\Big|_{\min MAD(u,v)}, \quad -p \le u, v \le p$$

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 bounday 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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA Y. ADMANE
Single user license only. Copying and networking prohibited.

F.1.2.1.2.2 Rigid object segmentation by motion tracting

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 V2 -77.

F.1.2.1.2.2.1 Object Motion Models

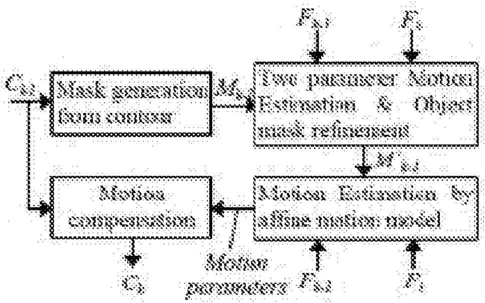

F_{k-1}
F_k
C_{k-1}
Mask generation from contour
M_{k-1}
Two parameter Motion Estimation & Object mask refinement
M'_{k-1}
Motion compensation
Motion Estimation by affine motion model
C_k
Motion parameters
F_{k-1}
F_k

Figure V2 -77 – Block diagramfor the object motion mode

We use the affine 6-parameter motion model for the objectmotion 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: $m = (a,b,c,d,e,f)^T$ 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 6×6 matrix $\mathbf{A}$ and right side column vector $\mathbf{b}$.

F.1.2.1.2.2.2 Motion Estimation & Compensation

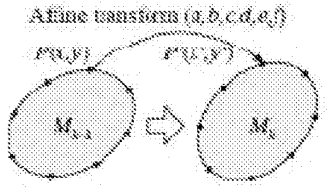


Figure V2 -78 – The object mask obtained by affine motion compensation

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) $\mathbf{m}^{(0)} \leftarrow 0^T$, $i \leftarrow 0$, 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 V2 -78.

F.1.2.1.3 Results of the user-assisted segmentation

Figure V2 -79 – Results of the user-assisted segmentaton

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.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
ANSI Store order X037445 Downloaded: 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

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 $\{f_{k-i}, i=0,\dots,n\}$. 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-j} to f_k, $j=1,\dots,n$. For each frame f_k, the algorithm splits in three steps (see Figure V2 -80).

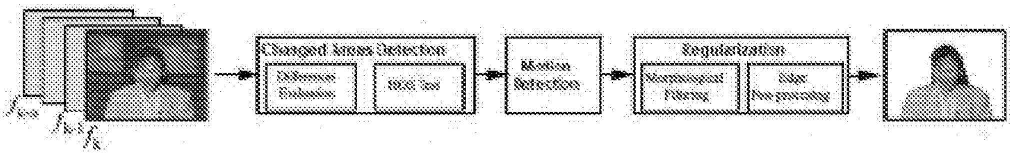


Figure V2 -80 -- Segmentation Using Higher Order moments and motion tracking: Block Diagram

Step One: Changed areas detection. The differences $\{ d_{k-j}(x,y)=f_{k-j}(x,y)-f_{k-n}(x,y), j=0,\dots,n-1 \}$ 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 $\hat{m}^{(4)}_{d_{k-j}}(x,y)$, a motion detection procedure is performed.

$$\hat{m}_d(x,y)=\frac{1}{9}\sum_{(s,t)\in\eta(x,y)} d(s,t) \quad (23), \qquad \hat{m}^{(4)}_d(x,y)=\frac{1}{9}\sum_{(s,t)\in\eta(x,y)} \left(d(s,t)-\hat{m}_d\right)^4 \; (24)$$

Step Two: Motion detection. This step aims at distingush 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 $\hat{m}^{(4)}_{d_{k-j}}(x,y), \hat{m}^{(4)}_{d_{k-j-1}}(x,y)$ is examined. For each pixel (x,y) the displacement is evaluated on a 3x3 window, adopting a SAD criterion (Sum of Absoute 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

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

examined. Moreover, the matching is not necessary on the pixels which are zero and are included in a 3x3 zero window, and they are assumed to be still. Thus, at the expense of some comparisons, the matching is performed on few pixels, and at the expense of examining more than a pair of difference, the search window is small.

Step Three: Regularization Still regions, internal to moving regions, are re-assigned to foreground. The regularization algorithm refines the imposing a priori local connectivity constraints, on both the background and the foreground.

Namely, topological constraints on the size of objects irregularities such as holes, isthmi, gulfs and isles are imposed by means of morphological filtering. Five morphological operations, with circular structuring element, are applied (Table V2 - 43). From a topological point of view, the regularization supports multiple moving regions whose size exceeds the size of the structuring element, corresponding to different moving objects. Finally, a post-processing operation refines the results on the basis of spatial edges.

Introducing user interaction in the segmentation algorithm allows to enhance its capability to extract the shapes of moving objects with accuracy. In fact, to minimize false alarm errors, the objects of interest may be partially missed and in particular the extracted object borders usually do not match the real object borders. To increase the segmentation performance the following user-algorithm interactions are provided:

Table V2 - 43 -- Morphological operations

	Radius	**Radius**	**Iteration**	**Iteration**
Closing by Rec.	2	3	5	8
Opening by Rec.	7	12	20	35
Opening	4	8		
Closing	3	6		
Closing by Rec.	10	15	200	250

The user driven threshold setting in order to tune, sequence by sequence and frame by frame, the algorithm capability to select the objects of interest.

The user marked area in which the object of interest is located.

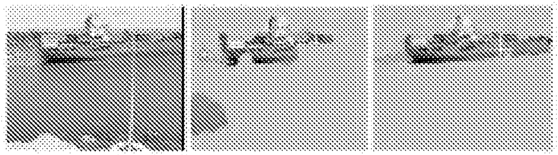


Figure V2 -81 -- Segmentation results comparison: original, automatic, semiautomatic

In this case the application of the automatic segmentation technique can be restriced and optimized within that ROI only, and in most the cases this gives a performance enhancement because parameters critical for revealing the object are estimated on the region where the object is included and not on the entire image where regions with different characteristics are usually present.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON
ANSI Store order Downloaded: PM ET
Single user license only. Copying and networking prohibited.

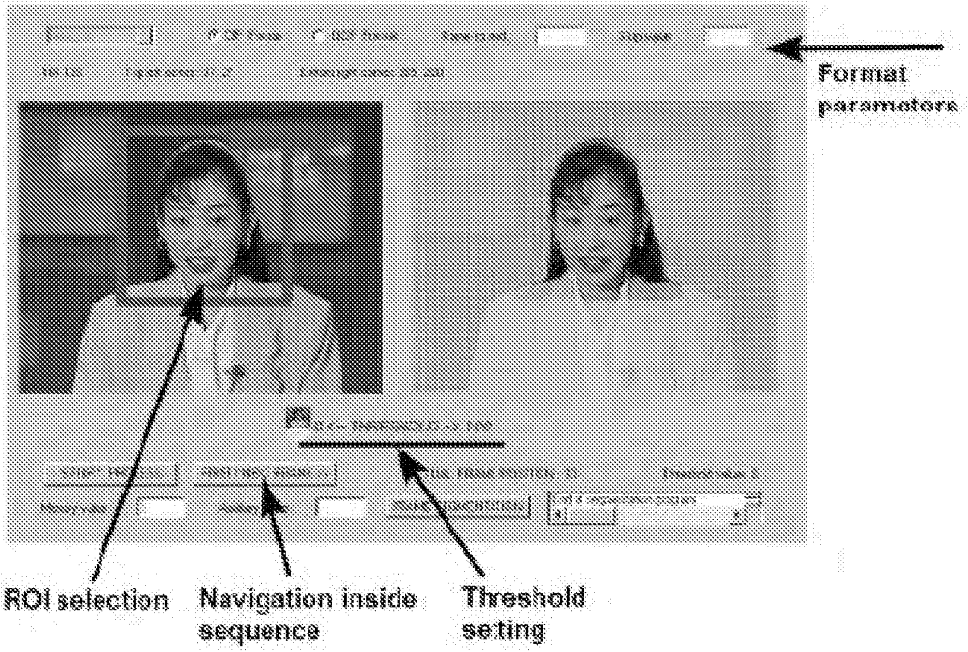


Figure V2 -82 – Java based Graphical User Interface for semi-automatic segmentation

An example of the obtained results is reported in Figure V2 -81, in comparison with the result obtained by a completely automatic technique. The enhanced algorithm capability to estimate critical parameters allows to improve its performance in revealing the object shape with accuracy.

A graphical user interface (GUI) allowing easy user interactions with the segmentation process is shown in Figure V2 -82. Basically, the result of the segmentation could be presented to the user frame by frame, so that he can decide to continue the process with the same parameters or to stop the process, revise the information he previously gave (such as parameters values or ROI selections), and restart the process.

F.1.3 References

[1] T. Aach, A. Kaup, R. Mester, "Statistical model-based change detection in moving video," *Signal Processing*, Vol. 31, No. 2, pp. 165-180, March 1993.

[2] T. Aach, A. Kaup, R. Mester, "Change detection in image sequences using Gibbs random fields: a Bayesian approach," *Proceedings Int. Workshop on Intelligent Signal Processing and Communication Systems*, Sendai, Japan, pp. 56-61, October 1993.

[3] J. Benois, L. Wu, "Joint Contour-based and Motion-based Image Sequences Segmentation for TV Image Coding at Low Bit-rate," *Visual Communications and Image Processing*, Chicago, Illinois, September 1994.

[4] M. Bierling, "Displacement estimation by hierarchical blockmatching," *3rd SPIE Symposium on Visual Communications and Image Processing*, Cambridge USA, pp. 942-951, November 1988.

[5] P. Bouthemy, E. François, "Motion segmentation and qualitative dynamic scene analysis from an image sequence," *International Journal of Computer Vision* vol.10, no.2, pp157-182, 1993.

[6] J. Dugelay, H. Sanson, "Differential methods for the identification of 2D and 3D motion models in image sequences," *Signal Processing*, vol.7, pp. 105-127, Sept. 1995.

[7] C. Gu, T. Ebrahimi, M. Kunt, "Morphological Spatio-temporal Segmentation for Content-based Video Coding," *International Workshop on Coding Techniques for Very Low Bit-rate Video*, Tokyo, Japan, November 1995.

[8] M. Hötter, R. Thoma, "Image Segmentation based on object oriented mapping parameter estimation," *Signal Processing*, vol. 15, No. 3, pp. 315-334, October 1988.

[9] R. Mech, M. Wollborn, "A Noise Robust Method for Segmentation of Moving Objects in Video Sequences," *International Conference on Acoustic, Speech and Signal*, Munich, Germany, April 1997.

[10] R. Mech, M. Wollborn, "A Noise Robust Method for 2D Shape Estimation of Moving Objects in Video Sequences Considering a Moving Camera," *Workshop on Image Analysis for Multimedia Interactive Services*, Louvain-la-Neuve, Belgium, June 1997.

[11] R. Mech, M. Wollborn, "A Noise Robust Method for 2D Shape Estimation of Moving Objects in Video Sequences Considering a Moving Camera," *Signal Processing: Special Issue on Video Sequence Segmentation for Content-based Processing and Manipulation*, Vol. 66, No. 2, pp. 203-217, April 1998.

[12] A. Neri, S. Colonnese, G. Russo, "Video Sequence Segmentation for Object-based Coders using Higher Order Statistics," *ISCAS '97*, Hongkong, June 1997.

[13] A. Neri, S. Colonnese, G. Russo, "Automatic Moving Objects and Background Segmentation by means of Higher Order Statistics," *IS&T Electronic Imaging '97 Conference: Visual Communication and Image Processing*, San Jose, 8-14 February 1997, SPIE Vol. 3024.

[14] F. Pedersini, A. Sarti, S. Tubaro, "Combined Motion and Edge Analysis for a Layer-based Representation of Image Sequences," *IEEE International Conference on Image Processing*, Lausanne, Switzerland, September 1996.

[15] P. Salembier, M. Pardàs, "Hierarchical Morphological Segmentation for Image Sequence Coding," *IEEE Transactions on Image Processing*, Vol.3, No.5, pp. 639-651, September 1994.

[16] Luc Vincent, Pierre Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on PAMI*, Vol.13, No. 6, pp. 583-598, June 1991.

[17] N. T. Watsuji, H. Katata, T. Aono, "Morphological Segmentation with Motion Based Feature Extraction," *International Workshop on Coding Techniques for Very Low Bit-rate Video*, Tokyo, Japan, November 1995.

[18] *Core Experiments on Multifunctional and Advanced Layered Coding Aspects of MPEG-4 Video*, ISO/IEC JTC 1/SC 29/WG 11 N 2176, May 1998.

[19] Jae Gark Choi, Munchurl Kim, Myoung Ho Lee and Chieteuk Ahn, "A user-assisted segmentation method for video object plane generation," *1998 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'98)*, Sokcho, Korea, pp. 7 – 10, July 1998.

[20] Jinsul Kwak, Jun Geun Jeon, Munchurl Kim, Myoung Ho Lee and Chieteuk Ahn, "A semi-automatic video segmentation method for object oriented multimedia applications," *1999 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC'99)*, Niigata, Japan, pp. 114 – 117, July 13-15, 1999.

[21] Juergen Stauder, Roland Mech, and Joern Ostermann, "Detection of moving Cast Shadows for Object Segmentation," to be published in *IEEE Transactions on Multimedia*, vol. 1, no. 1, 1999.

F.2 Bounding Rectangle of VOP Formation

This clause describes the bounding rectangle of VOP formation process. The formation of the bounding rectangle of VOP is based on the segmented shape information. The following explains the process to achieve the bounding rectangle of a VOP in such a way that the minimum number of macroblocks containing the object will be attained in order to achieve a higher coding efficiency.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA Y. ADAMANE
Single user license only. Copying and networking prohibited.

1. Generate the tightest rectangle whose vertical and horizontal positions of the top-left point are even numbered. In case of interlaced mode, the vertical position of the top-left point of the rectangle is a multiple of 4.
2. If the top left position of this rectangle is not the same as the origin of the image, the following steps have to be performed. Otherwise no further processing is necessary.
3. Form a control macroblock at the top left corner of the tightest rectangle as shown in Figure F-2.
4. Count the number of macroblocks that completely contain the VOP for all the points of which vertical and horizonal positions are even numbered (in case of interlaced mode, all the points of which horizontal positions are even numbered and vertical positions are multiple of 4) of the control macroblock, using the following procedure:
   - Generate a bounding rectangle from the control point to the right bottom side of the VOP which consists of multiples of 16x16 blocks.
   - Count the number of macroblocks in this bounding rectangle, which contain at least one object pel. To do so, it would suffice to take into account the boundary pels of a macroblock only.
5. Select the control point that results in the smallest number of non transparent macroblocks for the given object.
6. Extend the top left coordinate of the tightest rectangle generated in Figure F-2. to the selected control coordinate. This will create a rectangle that completely contains the object but with the minimum number of non transparent macroblocks in it. The VOP horizontal and vertical spatial references are taken directly from the modified top-left coordinate.

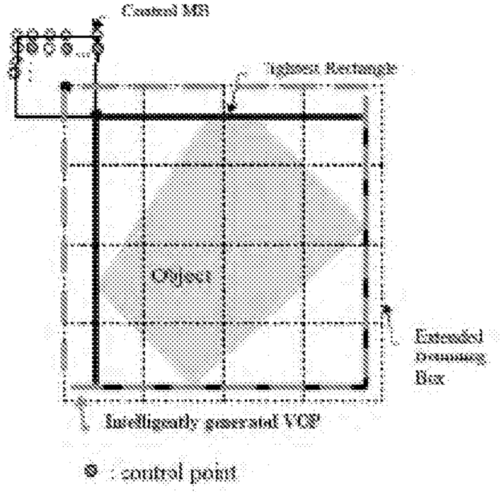


Figure F-2 -- Intelligent VOP Formation

F.3 Postprocessing for Coding Noise Reduction

The post-filter consists of deblocking filter and deringing filter. Either one or both of them can be turned on as needed.

F.3.1 Deblocking filter

The filter operations are performed along the 8x8 block edges at the decoder as a post-processing operation. Luminance as well as chrominace data is filtered. Figure F-3 shows the block boundaries.

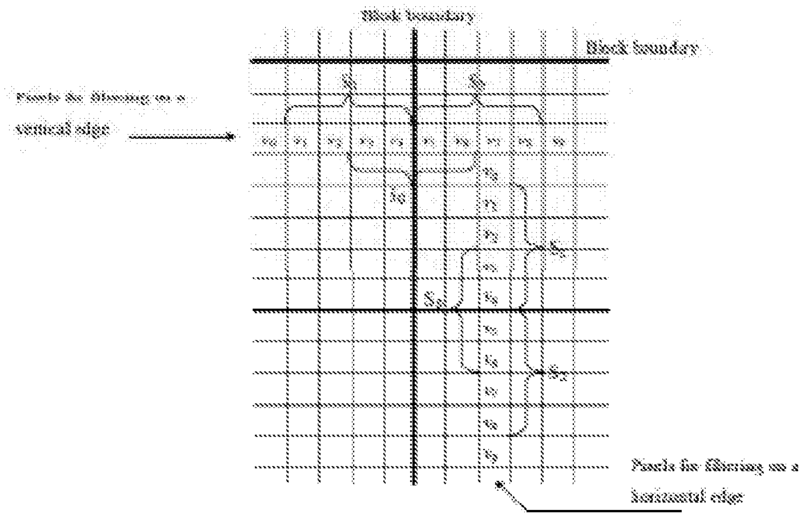

Block boundary
Block boundary
Pixels for filtering on a vertical edge
v_0 v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9
S_0
S_1
S_2
Pixels for filtering on a horizontal edge

Figure F-3 -- Boundary area around block of interest

In the filter operations, two modes are used separately depending on the pixel conditions around a boundary. The following procedure is used to find a very smooth region with blocking artifacts due to small dc offset and to assign it a DC offset mode. In the other case, default mode operations are applied.

```
eq_cnt = $\phi(v0-v1) + \phi(v1-v2) + \phi(v2-v3) + \phi(v3-v4) + \phi(v4-v5) + \phi(v5-v6) + \phi(v6-v7)$
              $+ \phi(v7-v8) + \phi(v8-v9)$,
where      $\phi(\gamma) = 1$ if $|\gamma| \le$ THR1 and 0 otherwise.
If (eq_cnt $\ge$ THR2)
        DC offset mode is applied,
else
        Default mode is applied.
```

For the simulation, threshold values of THR1 = 2 and THR2 = 6 are used.

In the default mode, a signal adaptive smoothing scheme is applied by differentiating image details at the block discontinuities using the frequency information of neighbor pixel arrays, S_0, S_1, and S_2. The filtering scheme in default mode is executed by replacing the boundary pixel values v_4 and v_5 with v_4' and v_5' as follows:

$$v_4' = v_4 - d,$$

$$v_5' = v_5 + d,$$

and $d = CLIP(5 \cdot (a_{3,0}' - a_{3,0})//8,\ 0,\ (v_4 - v_5)/2) \cdot \delta(|a_{3,0}| < QP)$

where $a_{3,0}' = SIGN(a_{3,0}) \cdot MIN(|a_{3,0}|, |a_{3,1}|, |a_{3,2}|)$.

Frequency components $a_{3,0}$, $a_{3,1}$, and $a_{3,2}$ can be evaluated from the simple inner product of the approximated DCT kernel [2 -5 5 -2] with the pixel vectors, i.e.,

$$a3,0 = ([2\ -5\ 5\ -2] \bullet [v3\ v4\ v5\ v6]^T)\ //\ 8,$$
$$a3,1 = ([2\ -5\ 5\ -2] \bullet [v1\ v2\ v3\ v4]^T)\ //\ 8,$$
$$a3,2 = ([2\ -5\ 5\ -2] \bullet [v5\ v6\ v7\ v8]^T)\ //\ 8.$$

Here CLIP(*x,p,q*) clips *x* to a value between *p* and *q*; and CP denotes the quantisation parameter of the macroblock where pixel v_5 belongs. δ(*condition*)=1 if the "*condition*" is true and 0 otherwise.

In very smooth region, the filtering in the default mode is not good enough to reduce the blocking artifact due to dc offset. So we treat this case in the DC offset mode and apply a stronger smoothing filter as follows :

```
max = MAX (v1, v2, v3, v4, v5, v6, v7, v8),
    min = MIN (v1, v2, v3, v4, v5, v6, v7, v8),
if (|max−min| < 2·QP ) {
    $v'_n = \sum_{k=-4}^{4} b_k \cdot p_{n+k}, 1 \le n \le 8$
    $p_m = \begin{cases} (|v_1 - v_0| < QP)\,?\,v_0 : v_1, & if\ m < 1 \\ v_m, & if\ 1 \le m \le 8 \\ (|v_8 - v_9| < QP)\,?\,v_9 : v_8, & if\ m > 8 \end{cases}$
    $\{b_k : -4 \le k \le 4\} = \{1,1,2,2,4,2,2,1,1\} // 16$
}
else
    No change will be done.
```

The above filter operations are applied for all the block boundaries first along the horizontal edges followed by the vertical edges. If a pixel value is changed by the previous filtering operation, the updated pixel value is used for the next filtering.

F.3.2 Deringing filter

This filter comprises three subprocesses; threshold determination, index acquisition and adaptive smoothing. This filter is applied to the pixels on 8x8 block basis. More specifically 8x8 pixels are processed by referencing 10x10 pixels at each block. The following notation is used to specify the six blocks in a macroblock. For instance, block[5] corresponds to the Cb block whereas block[*k*] is used as a general representation in the following subclauses.

F.3.2.1 Threshold determination

Firstly, calculate maximum and minimum gray value within a block in the decoded image. Secondary, the threshold denoted by *thr[k]* and the dynamic range of gray scale denoted by *range[k]* are set:

$$thr[k] = (\max imum[k] + \min imum[k] + 1) / 2$$

$$range[k] = \max imum[k] - \min imum[k]$$

An additional process is done only for the luminance blocks. Let *max_range* be the maximum value of the dynamic range among four luminance blocks.

$$\max\_range = range[k_{max}]$$

Then apply the rearrangement as follows.

```
for( k=1 ; k<5 ; k++ ){
    if( range[k] < 32 && max_range > =64 )
        thr[k] = thr[kmax];
    if( max_range<16 )
        thr[k] = 0;
}
```

F.3.2.2 Index acquisition

Once the threshold value is determined, the remaining operations are purely 8x8 block basis. Let *rec(h,v)* and *bin(h,v)* be the gray value at coordinates *(h,v)* where *h,v=0,1,2,...,7*, and the corresponding binary index, respectively. Then *bin(h,v)* can be obtained by:

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA K. ADAPANE
ANSI Store order X037445. Downloaded 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

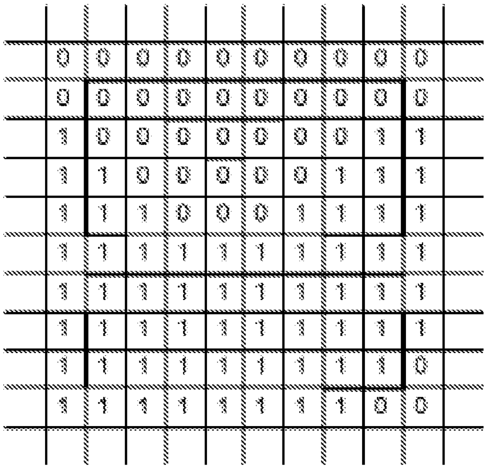
$$bin(h,v)=\begin{cases}1 & if\ rec(h,v)\geq thr\\ 0 & otherwise\end{cases}$$

Note that *(h,v)* is use to address a pixel in a block, while *(i,j)* is for accessing a pixel in a 3x3 window.

F.3.2.3 Adaptive smoothing

F.3.2.3.1 Adaptive filtering

The figure below is the binary indices in 8x8 block level, whereas practically 10x10 binary indices are calculated to process one 8x8 block.



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

Figure F-4 -- Example of adaptive filtering and binary index

The filter is applied only if the binary indices in a 3x3 window are all the same, i.e., all "0" indices or all "1" indices. Note 10x10 binary indices are obtained with a single threshold which corresponds to the 8x8 block shown in the above figure, where the shaded region represents the pixels to be filtered.

The filter coefficients used for both intra and non-intra blocks denoted by *coef(i,j)*, where *i,j*=-1,0,1, are:

1	2	1
2	4	2
1	2	1

Figure F-5 -- Filter mask for adaptive smoothing

Here the coefficient at the center pixel, i.e., *coef(0,0)*, corresponds to the pixel to be filtered. The filter output *flt'(i,j)* is obtained by:

$$flt'(h,v)=\left\{8+\sum_{i=-1}^{1}\sum_{j=-1}^{1}coef(i,j)\cdot rec(h+i,v+j)\right\}//16$$

F.3.2.3.2 Clipping

The maximum gray level change between the reconstructed pixel and the filtered one is limited according to the quantisation parameter, i.e., QP. Let *flt(h,v)* and *flt'(h,v)* be the filtered pixel value and the pixel value before limitation, respectively.

```
if( flt'(h,v) - rec(h,v) > max_diff )
     flt(h,v) = rec(h,v) + max_diff
else if( flt'(h,v) - rec(h,v) < -max_diff )
     flt(h,v) = rec(h,v) - max_diff
else
     flt(h,v) = flt'(h,v)
where max_diff=QP/2 for both intra and instr macroblocks.
```

F.3.3 Further issues

In order to reduce the number of computations in post-filtering, two kinds of semaphores can be defined: the blocking semaphores and the ringing semaphore. Depending on the blocking semaphores, the horizontal and the vertical deblocking filtering is applied strongly and weakly on the block boundary. If the ringing semaphore (RS) of a current block is "1", deringing filtering is applied. For extracting the semaphores in intra-frame, when only a DC component in the 8x8 inverse quantized coefficients (IQC), the DCT coefficients after inverse quantisation, has a non-zero value, both the horizontal blocking semaphore (HBS) and the vertical blocking semaphore (VBS) of the block are set to "1". When only the coefficients in the top row of the IQC have non-zero values, the VBS is set to "1". When only the coefficients in the far left column of the IQC have non-zero values, the HBS is set to "1". The RS is set to "1" if any non-zero coefficient exists in positions other than a DC component, the first horizontal AC component, and the first vertical AC comonent. Also the semaphores of the inter-frame are calculated from both the residual signal and the semaphores of the reference frame by using the motion vector.

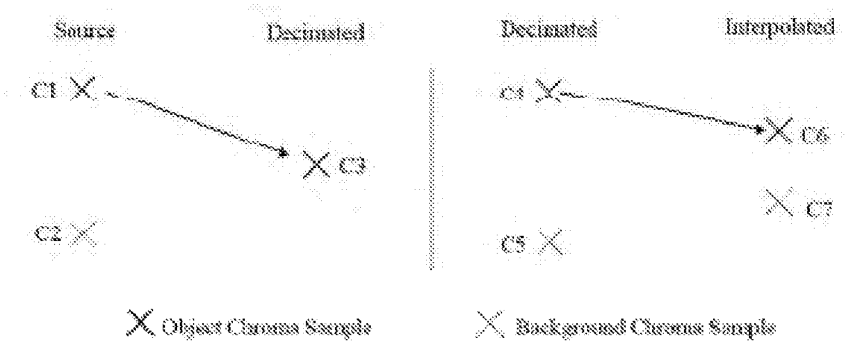
[1] M2723 Y.L. Lee et al.

F.4 Chrominance Decimation and Interpolation Filterng for Interlaced Object Coding

The chrominance decimation filtering and the chrominance interpolation filtering are peformed on the basis of the same field in case of interlaced object-based coding during the image format conversion process from 4:2:2 to 4:2:0 and from 4:2:0 to 4:2:2, respectively. In case that filter taps contain only object samples, conventional filtering methods are applied. When filter taps contain both object and background samples, the following decimation and interpolation filering methods are applied:

Decimation filtering: The source object and background chroma samples are identified from the luminance shape information. Among the decimated chroma samples, the object chroma samples are identified by the field-based chroma subsampling method described in subclause 7.6.1.5. When filter taps contain both of object and background samples at the object boundary, only chroma sample values of object are used for obtaining its decimated chroma sample value. For example, in case of a two-tap filter, one tap is on an object sample and the other is on a background sample, the resulting chroma sample is obtained using only the object chroma sample, not counting on the background sample. In Figure F-6 (a), the source chroma sample C1 is object and C2 is background. The value of the resulting decimated chroma sample C3 is the value of C1.

Interpolaton filtering: In the interpolation filtering process which is the inverse of the decimation filtering, object chroma samples are identified among the interpolated chroma samples from the luminance shape information and the values of the interpolated object chroma samples are obtained applying an interpolaton filter on input decimated chroma samples. When filter taps include both of decimated object and background chroma samples, only the decimated object chroma samples are counted. For example, in case of a two-tap filter in Figure F-6 (b), two decimated chroma samples, C4 and C5, are used for obtaining the interpolated chroma samples C6 and C7. Suppose that C6 is identified as object and C7 is identified as background from the luminance shape information and the chroma sample C4 is object and C5 is background, then the interpolated chroma sample C6 is obtained as the value of the chroma sample C4. The interpolation for C7 is not necessary because it is background.



(a) Decimation Filtering

(b) Interpolation Filtering

Figure F-6 – Chrominance Decimation and Interpolation Filtering at the Object Boundaries

Annex G
(normative)

Profile and level indication and restrictions

This annex specifies the syntax element restrictions and permissible layer combinations.

Table G-1 – FLC table for profile_and_level_indication

Profile/Level	Code
Reserved	00000000
Simple Profile/Level 1	00000001
Simple Profile/Level 2	00000010
Simple Profile/Level 3	00000011
Reserved	00000100 – 00010000
Simple Scalable Profile/Level 1	00010001
Simple Scalable Profile/Level 2	00010010
Reserved	00010011 – 00100000
Core Profile/Level 1	00100001
Core Profile/Level 2	00100010
Reserved	00100011 – 00110001
Main Profile/Level 2	00110010
Main Profile/Level 3	00110011
Main Profile/Level 4	00110100
Reserved	00110101 – 01000001
N-bit Profile/Level 2	01000010
Reserved	01000011 – 01010000
Scalable Texture Profile/Level 1	01010001
Reserved	01010010 – 01100000
Simple Face Animation Profile/Level 1	01100001
Simple Face Animation Profile/Level 2	01100010
Simple FBA Profile/Level 1	01100011
Simple FBA Profile/Level 2	01100100
Reserved	01100101 – 01110000
Basic Animated Texture Profile/Level 1	01110001
Basic Animated Texure Profile/Level 2	01110010
Reserved	01110011 – 10000000
Hybrid Profile/Level 1	10000001
Hybrid Profile/Level 2	10000010
Reserved	10000011 – 10010000
Advanced Real Time Simple Profile/Level 1	10010001
Advanced Real Time Simple Profile/Level 2	10010010
Advanced Real Time Simple Profile/Level 3	10010011
Advanced Real Time Simple Profile/Level 4	10010100
Reserved	10010101 – 10100000

Core Scalable Profile/Level1	10100001
Core Scalable Profile/Level2	10100010
Core Scalable Profile/Level3	10100011
Reserved	10100100 – 10110000
Advanced Coding Efficiency Profile/Level 1	10110001
Advanced Coding Efficiency Profile/Level 2	10110010
Advanced Coding Efficiency Profile/Level 3	10110011
Advanced Coding Efficiency Profile/Level 4	10110100
Reserved	10110101 – 11000000
Advanced Core Profile/Level 1	11000001
Advanced Core Profile/Level 2	11000010
Reserved	11000011 – 11010000
Advanced Scalable Texture/Level1	11010001
Advanced Scalable Texture/Level2	11010010
Advanced Scalable Texture/Level3	11010011
Reserved	11010100 – 11111111

Table G-2 -- possible combination of two tools

Tools		BA	B-VOP	OBMC	Q	ER	SH	BS	GS	IN	NB	TS(B)	TS(E)	SS(B)	SS(E)	SP	ST	DRC	NP	GMC	QMC	SADCT	ERST	WT	SSST	OSS(B)	OSS(E)	MAC
Basic • I-VOP • P-VOP • AC/DC Prediction • 4-MV, Unrestricted MV	BA		✓	✓	✓	✓	✓ a)	✓	✓	✓	✓	✓	✓	✓	✓	✓ b)	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
B-VOP	BV			✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓
P-VOP with OBMC (Texture)	OBMC				✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	D	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓
Method 1/Method 2 Quantisation	Q					✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓
Error resilience	ER						✗	✓	✓ d)	✓ e)	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓ d)
Short Header	SH							✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Binary Shape (progressive)	BS								✗	✓	✓	✓	✓	D	D	✓	H	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓
Greyscale Shape	GS									✓	D	D	D	D	D	✓ f)	H	✗	✗	✓	✓	✓	✗	✗	✗	D	D	✓ g)
Interlace	IN										✓	D	D	D	D	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	D	D	✓
N-Bit	NB											✓ c)	✓ c)	D	D	✗	✗	✗	✗	D	D	D	✗	✗	✗	D	D	D
Temporal Scalability (Base)	TS(B)															D	✗	✗	✗	✓	✓	✓	✗	✗	✗	D	D	D

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ABAWANE
ANSI Store order X037445; Downloaded 7/18/2007 3:59:21 PM ET
Single user license only. Copying and networking prohibited.

Temporal Scalability (Enhancement)	TS(E)															✗	✗	✗	✗	✗	D	D	✗	✗	✗	D	D	D
Spatial Scalability (Base)	SS(B)															D	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	D
Spatial Scalability (Enhancement)	SS(E)															✗	✗	✗	✗	✗	D	D	✗	✗	✗	✓	✓	D
Sprite	SP																✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Still Texture	ST																	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
Dynamic Resolution Conversion	DRC																		✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
NEWPRED	NP																			✗	✗	✗	✗	✗	✗	✗	✗	✗
Global Motion Compensation	GMC																				✓	✓	✗	✗	✗	✗	✗	D
Quarter-pel Motion Compensation	QMC																					✓	✗	✗	✗	✗	✗	D
SA-DCT	SADCT																						✗	✗	✗	✗	✗	D
Error Resilience for Still Texture Coding	ERST																							✓	✓	✗	✗	✗
Wavelet Tiling	WT																								✓	✗	✗	✗
Scalable Arbitrary Shape for Still Texture Coding	SSST																									✗	✗	✗
Object Based Spatial Scalability (Base)	OSS(B)																											D
Object Based Spatial Scalability (Enhancement)	OSS(E)																											D
Multiple Auxiliary Component	MAC																											

✓ – This combination is supported by the syntax. (Subject to further restrictions imposed by the definition of visual object types).

✗ – This combination is not supported by the syntax.

D – This combination is supported by the syntax but not alowed within one Visual Object(). This combination may be supported in the future, if requirements arise for the corresponding combination.

H – supported at a higher syntactic level.

a) Short Header does not support DC/AC Prediction, 4MV nor Unrestricted MV.
b) Sprite does not support 4MV nor Unrestricted MV.
c) P-VOP Temporal Scalability only.
d) Grey Scale does not support Data Partitioning nor RVLC.
e) Interlace does not support Data Partitioning nor RVLC.
f) Grey Scale does not support Low Latency Sprites.
g) MAC is extension of grayscale shape, no real combinaion.

Annex H
(infomative)

Patent statements

The user's attention is called to the possibility that, for some of the processes specified in this part of ISO/IEC 14496, conformance with this specification may require use of an invention covered by patent rights.

By publication of this part of ISO/IEC 14496, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. Information regarding such patents can be obtained from the following organisations.

H.1 Patent statements for ISO/IEC 14496 Version 1

The table summarises the formal patent statements received and indicates the parts of ISO/IEC 14496 Version 1 to which the statement applies (S: ISO/IEC 14496-1:1999, V: tools listed in Table 9-1, A: ISO/IEC 14496-3:1999, R: ISO/IEC 14496-5:2000, D: ISO/IEC 14496-6:1999). The list includes all organisations that have submitted informal patent statements. However, if no "X" is present, no formal patent statement has yet been received from that organisation.

		S	V	A	R	D
1.	Alcatel	x	x	x	x	x
2.	AT&T					
3.	BBC	x	x	x	x	x
4.	Bosch		x	x	x	
5.	British Telecommunications	x	x	x	x	x
6.	Canon	x	x	x	x	x
7.	CCETT	x	x	x	x	x
8.	Columbia University	x	x	x	x	x
9.	Creative	x		x	x	
10.	CSELT			x		
11.	DEmoGraFX		x		x	
12.	DirecTV	x	x	x		
13.	Dolby	x	x	x	x	x
14.	EPFL	x	x	x		
15.	ETRI	x	x	x	x	x
16.	FhG	x	x	x	x	x
17.	France Telecom	x	x	x	x	x
18.	Fujitsu Limited	x	x	x	x	x
19.	GC Technology Corporation	x	x	x		
20.	General Instrument		x		x	
21.	Hitachi	x	x	x	x	x
22.	Hyundai	x	x	x	x	x
23.	IBM	x	x		x	
24.	Institut für Rundfunktechnik	x	x	x		x
25.	InterTrust					
26.	JVC	x	x	x	x	x

		S	V	A	R	D
27.	KDD Corporation	x	x			
28.	KPN	x	x	x	x	x
29.	LG Semicon					
30.	Lucent					
31.	Matsushita	x	x	x	x	x
32.	Microsoft	x	x	x	x	x
33.	MIT					
34.	Mitsubishi	x	x	x	x	
35.	Motorola		x		x	
36.	NEC Corporation	x	x	x	x	x
37.	NHK	x	x	x	x	x
38.	Nokia		x	x	x	
39.	NTT	x	x	x	x	x
40.	OKI	x	x	x	x	x
41.	Philips	x	x	x	x	x
42.	PictureTel Corporation		x		x	
43.	Rockwell	x	x	x	x	x
44.	Samsung	x	x	x		
45.	Sarnoff	x	x	x	x	x
46.	Scientific Atlanta	x	x	x	x	x
47.	Sharp	x	x	x	x	x
48.	Siemens	x	x	x		
49.	Sony	x	x	x	x	x
50.	Telenor	x	x	x	x	x
51.	Teltec DCU		x		x	
52.	Texas Instruments					
53.	Thomson	x	x	x		
54.	Toshiba		x			
55.	Unisearch Ltd.		x		x	
56.	Vector Vision		x			

H.2 Patent statements for the extensions provided in ISO/IEC 14496 Version 2

The table summarises the formal patent statements received and indicates the parts of the extensions provided in ISO/IEC 14496 Version 2 to which the statement applies (S: ISO/IEC 14496-1:1999/Amd. 1:2000, V: tools listed in Figure V2 -39 and not listed in Table 9-1, A: ISO/IEC 14496-3:1999/Amd. 1:2000, R: ISO/IEC 14496-5:2000/Amd. 1:2000, D: ISO/IEC 14496-6:1999/Amd. 1:2000). The list includes all organisations that have submitted informal patent statements. However, if no "X" is present, no formal patent statement has yet been received from that organisation.

		S	V	A	R	D
1.	Apple	x			x	
2.	AT&T	x	x	x		
3.	British Telecommunications					
4.	Bosch	x	x	x	x	x
5.	CCETT	x	x	x	x	x
6.	Columbia Innovation Enterprise	x	x	x	x	x

		S	V	A	R	D
7.	DemoGraFX	x	x	x	x	x
8.	DirecTV	x	x	x		
9.	Dolby	x	x	x	x	x
10.	EPFL	x	x		x	
11.	France Telecom	x	x	x	x	x
12.	Fraunhofer			x	x	
13.	Fujitsu		x		x	
14.	Hitachi	x	x	x	x	x
15.	Hyundai	x	x	x	x	x
16.	IBM	x	x	x	x	x
17.	Intertrust					
18.	JVC	x	x	x	x	x
19.	KPN		x			
20.	Lucent					
21.	Matsushita Electric Industrial Co., Ltd.	x	x	x	x	x
22.	Microsoft	x	x	x	x	x
23.	Mitsubishi	x	x	x	x	x
24.	NEC	x	x	x	x	x
25.	NHK	x	x	x	x	x
26.	Nokia	x		x	x	
27.	NTT		x			
28.	NTT Mobile Communication Networks			x		
29.	OKI	x	x	x	x	x
30.	Optibase	x			x	
31.	Philips					
32.	Samsung	x	x	x	x	
33.	Sarnoff	x	x	x	x	x
34.	Sharp	x	x	x	x	x
35.	Siemens	x	x	x	x	x
36.	Sony	x	x	x	x	x
37.	Sun	x				
38.	Thomson	x	x	x	x	x
39.	Toshiba		x			

Annex I
(informative)

Encoder Complexity Reduction Based on Intelligent Pre-Quantisation

I.1 Introduction

This subclause describes an encoder complexity reduction scheme. The proposed algorithm is aimed at reducing computations by a significant amount at the encoder. It is based on extracting a low cost feature (SAE) from the motion compensated residual blocks, and using this feature and an intelligently derived threshold value to detect "zero blocks" (blocks have all zero quantized DCT coefficients). Once a "zero block" is detected, we can skip DCT, quantisation, and corresponding IDCT and inverse quantisation for this block to reduce the encoder computation.

Most of the existing video coding standards (MPEG-1, MPEG-2, MPEG-4, H.261 and H.263) are based on motion estimation, DCT, quantisation, and entropy coding. The core of these algorithms is a differential loop where a displaced frame difference is calculated based on the motion of the current frame from the immediate past frame. This displaced frame difference is then quantized and entropy coded via a variable length code. In the end these variable length codes, along with the codes which contain the header information conforming to the syntax of a particular algorithm, are transmitted over the channel. Motion estimation is the process by which we calculate the displacement of one frame in a sequence with respect to the previous frame. By calculating the displacement on a block basis, we can compute a displaced frame difference that is easier to code, thereby reducing the temporal redundancy. In the next step, DCT of individual blocks (8x8 pixels) helps in removing the spatial redundancy by bringing down the most relevant information into the lower most coefficients in the DCT domain. After having obtained a block of DCT coefficients that contain the energy of the displaced blocks, quantisation of these blocks is done. Quantisation is a uniform quantisation with step size varying from 2 to 62. It is implemented as a division, or as a table look up operation for a fixed-point implementation. Lastly, having obtained the quantized coefficients, entropy coding is done to further reduce the overall coding rate. Traditionally most video coding standards use Huffman coding for entropy coding

In terms of computations motion estimation is the costliest module. Typically, it accounts for more than half of the MIPs at the encoder. For this reason, most of the video encoding solutions prefers to do motion estimation either by some dedicated hardware, or by some fast sub-optimal software scheme. DCT and IDCT, which is needed in the encoder for reconstructing the quantized frame for motion estimation, comprise the next most costly part of the encoder. Then follows quantisation and Huffman encoding. In very low bit rate applications, after the quantisation of the motion compensated DCT blocks, many of the blocks have all of their coefficients as zero. It is observed that a majority of the blocks are zero after quantisation. This means that if we can detect a priori which blocks are being zeroed out after quantisation, we can forego DCT, quantsation, dequantisation and IDCT for those blocks. This would translate to a great amount of savings in computations at the encoder. In the following, we outline a method to achieve this goal with no loss in the perceived quality of he coded sequence.

I.2 Feature Selection and Pre-quantisation

We begin with identifying some key features of the motion compensated blocks that will help in classifying them into zero and non-zero blocks. A good feature should allow us to classify the blocks very accurately and at the same time it should be relatively easy to compute too. Using a complicated classification scheme defeats our main goal of reducing the overall computations. We consider several features and rank them in the order of their performanse.

Most of the features we select are easy to compute and are based on first or second ordered moments of the motion compensated blocks. We consider the following features:

SAE: meanabsolute error

$$SAE = \sum_{i=1}^{8} \sum_{j=1}^{8} |f(i, j)| \text{, where } f(i, j) \text{ is the pel value at } (i, j)\text{ - th location}$$

MSE: meansquare error

$$MSE = \sum_{i=1}^{8} \sum_{j=1}^{8} (f(i, j))^2 \text{, where } f(i, j) \text{ is the pel value at } (i, j)\text{ - th location}$$

MAD: mean absolute difference

$$MAD = \sum_{i=1}^{8} \sum_{j=1}^{8} |f(i, j) - \mu| \text{ where } f(i, j) \text{ is the pel value at } (i, j)\text{ - th location } \& \mu \text{ is the mean of the block}$$

VAR: block variance

$$VAR = \sum_{i=1}^{8} \sum_{j=1}^{8} (f(i, j) - \mu)^2 \text{, where } f(i, j) \text{ is the pel value at } (i, j)\text{ - th location } \& \mu \text{ is the mean of the block}$$

MAX: maximum value

$$MAX = \max(f(i, j)) \quad i, j \in (1,8)$$

We use each of the above features in classifying the motion compensated blocks as zero or one. A zero block is defined as a block with all zero quantized DCT coefficients and a one block as a block with at least one non-zero quantized DCT coefficient. For doing the classification, we select one of the above features and use a corresponding (experimentally determined) threshold to decide whether or not this block should be classified as zero. Each time we classify a block as zero, we would be zeroing out the block to begin with and would not be doing DCT and quantisation, and consequently IDCT and inverse quantisation. This translates to a lot of savings in computations. Next we define some terminology which would help us in ranking the performance of different features and in quantifying the degradation in performance while using any such feature based classification scheme.

As is the case with any multiple hypotheses problem, there are two types of errors associated with this feature based classification scheme. Each time we use a feature to do the above classification, there are instances where we classify a block as zero, when it should have been coded instead (one). We call this error as false alarm (FA). The second type of error we encounter in such a classifcation scheme is when we classify a block to be one when in reality it should have been zero. We call this as misdetection (MD). We formally define a quantitative measure of false alarm and misdetection as follows:

$$FA_{rate} = \frac{FA}{N1}, \quad MD_{rate} = \frac{MD}{N0},$$

where FA is the number of blocks classified as zero when they actualy should have been one, MD is the number of blocks classified as one when they actually should have been zero, N1 is the number of true zero blocks, and N0 is the number of true one blocks. We compare the effectiveness of different features by plotting false alarm (FA) vs. misdetection (MD) curves resulting from using the above features in coding a video sequence. Ideally we would want a feature to give us a low false alarm and a low misdetection.

From our experiments we conclude that SAE and MSE have the best performance lowest false alaram rate and lowest misdetection rate), MAX has the worst performance (highest curve), and MAD, VAR are in between. Since SAE is computationally less expensive to calculate than MSE, we will use SAE as our major feature in classifying the blocks. Also, since MAX is very easy to compute, and it also promises to catch some edge information in a block, we will also investigate using it further.

I.3 Model Verification and Threshold Setting

We apply the classification schemes based on above features on a wide range of video sequences. Since motion estimation acts like a normalization process, the statistics of residue blocks is expected o be similar. We verify this fact by encoding several typical sequences. We observe that the curves for misdetection rate and false alarm rate are similar for all sequences. In other words, the feature classification scheme is quite robust to the variations in the input sequence.

Based on experiments, we find SAE gives best result in terms of false alarm and miscetection rate. Here we will find an optimal threshold that results in negligible number of false alarm blocks and at the same time small number of misdetection blocks. Negligible false alarm rate means only very few true one blocks will be misclassified as zero blocks, therefore, the quality degradation will be totally insignificant. Small misdetection rate means most true zero blocks will be classified as zero blocks, and these blocks will skip DCT and quantisation, therefore, computational complexity will be reduced. Here we first calculate the upper bound of SAE that will guarantee zero false alarm rate.

The forward DCT is:

$$F(u,v) = \frac{1}{4} C(u) C(v) \sum_{i=0}^{7} \sum_{j=0}^{7} \cos((2i+1)u\frac{\pi}{16}) \cos((2j+1)v\frac{\pi}{16}) f(i,j)$$

$$\Rightarrow |F(u,v)| \le \frac{1}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} |f(i,j)| = {}^{1}\!/_{4}\,\text{SAE, for } u,v \ne 0$$

$$= {}^{1}\!/_{8}\,\text{SAE, for } u,v = 0$$

I.3.1 H.263 Quantiser

When using the H.263 quantiser, the quantisation for INTER mode blocks AC/DC coefficients can be expressed as:

$$\text{coeff}(u,v) = \text{sign}(F(u,v)) * \frac{|F(u,v)| - QP/2}{2*QP}$$

In this case, to guarantee all coeff(u, v)=0, we have

$$|F(u,v)| - QP/2 < 2*QP$$
$$|F(u,v)| < 5/2*QP$$

From above equations we can guarantee that the DC componet of the block would be zero if the following condition is satisfied:

$$\frac{1}{8}\text{SAE} < 5/2*QP$$
$$\text{SAE} < 20*QP$$

I.3.2 MPEG-4 Quantiser

When using the MPEG-4 quantiser with the MPEG-4 non-intra quantiser matrix, the quantisation for INTER mode blocks AC/DC coefficients can be expressed as:

$$\text{coeff}(u,v) = \text{sign}(F(u,v)) * \frac{(16*|F(u,v)|)/w[u][v]}{2*QP},$$

where w is thenon-intra quantiser matrix. In this case, to guarantee all coeff(u, v)=0, we have

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON [illegible]
Single user licence only. Copying and networking prohibited.

$$|F(u, v)| < 2 * QP$$

From aboveequations we can guarantee that the DC componet of the block would bezero if the following condition is satisfied:

$$\frac{1}{8}\mathrm{SAE} < 2 * QP$$
$$\mathrm{SAE} < 16 * QP$$

Since the DC component is the most important and more likely to have non-zero value if a block has any non-zero components, this threshold value guarantees no quality degradation when applying the pre-quantisation scheme and receiving coding complexity reduction.

To conclude, the threshold is set to 20*QP when H263 quantiser is used, and 16*QP when MPEG-4 quantiser with default quantisation matrix is used. These thresholds wil guarantee accurate prediction of zero coefficients both for AC as well as the DC coefficients.

Annex J
(normative)

View dependent object scalability

J.1 Introduction

Coding of View-Dependent Scalability (VDS) parameters for texture can provide for efficient incremental decoding of 3D images (e.g. 2D texture mapped onto a gridded 3D mesh such as terrain). Corresponding tools from ISO/IEC 14496-1:1999 and this part of ISO/IEC 14496 are used in conjunction with downstream and upstream channels of a decoding terminal. The combined capabilities provide the means for an encoder to react to a stream of viewpoint information received from a terminal. The encoder transmits a series of coded textures optimized for the viewing conditions which can be applied in the rendering of textured 3D meshes by the receiving terminal. Each encoded view-dependent texture (initial texture and incremental updates) typically corresponds to a specific 3D view in the user's viewpoint that is first transmitted from the receiving terminal.

A tool defined in ISO/IEC 14496-1 transmits 3D viewpoint parameters in the upstream channel back to the encoder. The encoder's response is a frequency-selective, view-dependent update of DCT coefficients for the 2D texture (based upon view-dependent projection of the 2D texture in 3D) back to the receiving terminal, along the downstream channel, for decoding by a Visual DCT tool at the receiving terminal. This bilateral communication supports interactive server-based refinement of texture for low-bandwidth transmissions to a decoding terminal that renders the texture in 3D for a user controlling the viewpoint movement. A gain in texture transmission efficiency is traded for longer closed-loop latency in the rendering of the textures in 3D. The terminal coordinates inbound texture updates with local 3D renderings, accounting for network delays so that texture cached in the terminal matches each rendered 3D view.

A method to obtain an optimal coding of 3D data is to take into account the viewing position in order to transmit only the most visible information. This approach reduces greatly the transmission delay, in comparison to transmitting all scene texture that might be viewable in 3D from the encoding database server to the decoder. At a given time, only the most important information is sent, depending on object geometry and viewpoint displacement. This technique allows the data to be streamed across a network, given that a upstream channel is available for sending the new viewing conditions to the remote database. This principle is applied to the texture data to be mapped on a 3D grid mesh. The mesh is first downloaded into the memory of the decoder using the appropriate BIFS node, and then the DCT coefficients of the texture image are updated by taking into account the viewing parameters, i.e. the field of view, the distance and the direction to the viewpoint.

J.2 Decoding Process of a View-Dependent Object

This clause explains the process for decoding the texture data using the VDS parameters. In order to determine which of the DCT coefficients are to be updated, a "mask", which is a simple binary image, shall be computed. The first step is to determine the viewing parameters obtained from the texture-mesh composition procedure that drives 3D rendering in the user's decoding terminal. These parameters are used to construct the DCT mask corresponding to the first viewpoint of the session (VD mask). This mask is then updated with differential masks, built with the new viewing parameters that allow the texture image to be streamed. The bitstream syntax for view parameters and incremental transmission of DCT coefficients is given elsewhere in ISO/IEC 14496-1:1999 and this part of ISO/IEC 14496.

J.2.1 General Decoding Scheme

The following subclauses outline the oveall process for the decoder and encode to accomplish the VDS functionalities.

J.2.1.1 View-dependent parameters computation

The VDS parameters (α and β angles, distance d for each cell) shall be computed using the geometrical parameters (Mesh, Viewpoint, Aimpoint, Rendering window). These parameters shal be computed for each cell of the grid mesh.

J.2.1.2 VD mask computation

For each 8x8 block of texture elements within a 3D mesh cell, the locations of the visible DCT coefficients inside the DCT block shall be computed using α and β angles, and the distance d defined for each cell relative to the viewpoint. The result shall be out in a binary mask image.

J.2.1.3 Differential mask computation

With the knowledge of which DCT coefficients have already been received (Binary mask buffered image) and which DCT coefficients are necessary for the current viewing conditions (Binary VD mask image), the new DCT coefficients shall be determined (Binary Differential masl image) as described in subcause J.2.4.

J.2.1.4 DCT coefficients decoding

The Video Intra bitstream, in the downstream channel, shall be decoded by the receiver terminal to obtain the DCT coefficients (DCT image). The decoding procedure is described in subclause J.2.5.

J.2.1.5 Texture update

The current DCT buffer in the receiver terminal shall be updated according to the Differential mask, using the received DCT image. The new received DCT coefficients shall be added to the buffered DCT image.

J.2.1.6 IDCT

The inverse DCT of the updated DCT image shall computed, as specified in subclause J.2.7, to obtain the final texture.

J.2.1.7 Rendering

The texture is mapped onto the 3D mesh and the rendering of the scene is done, taking into account the mesh and the viewing conditions. This part of the procedure is outside the scope of this part of ISO/IEC 14496.

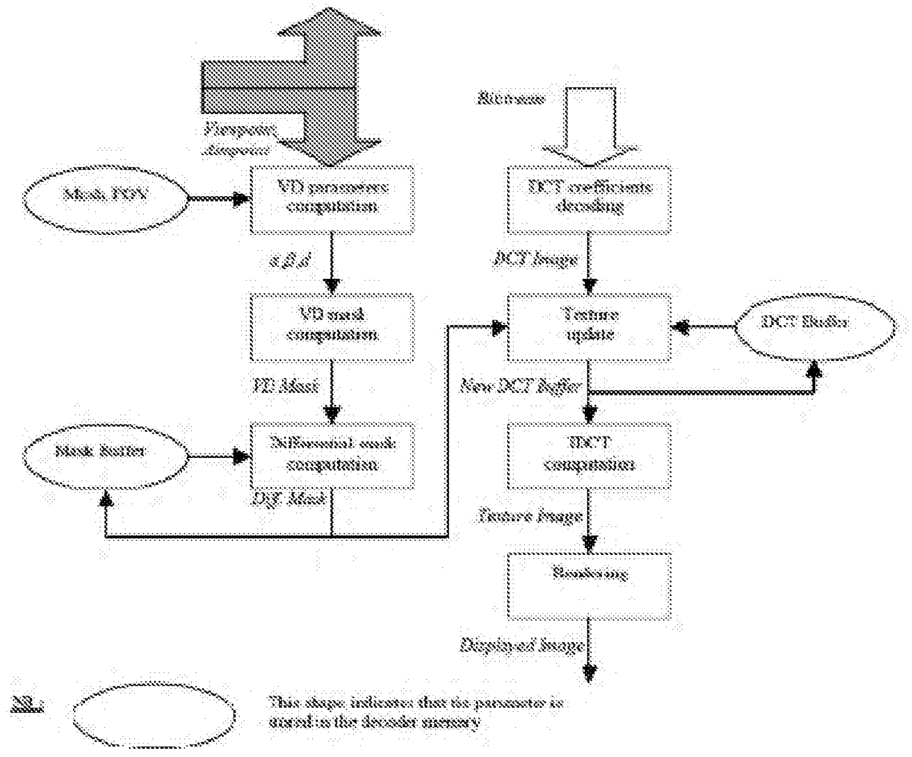


Figure J-1 – General Decoding Scheme of a View-Dependent Object

J.2.2 Computation of the View-Dependent Scalability parameters

The VDS parameters shall be computed for each cell of the grid mesh. The mesh may either be a quadrilateral or a triangular mesh. The number of cells in each dimension shall be equal to the texture size divided by 8.

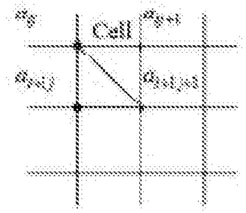


Figure J-2 –Mesh cell

J.2.2.1 Distance criterion:

$$Rd = \frac{1}{u}$$

u is the distance between viewpoint and Cell center: $u = \|\vec{v} - \vec{c}\|$ with $\vec{c} = \frac{1}{4}(\vec{a}_{ij} + \vec{a}_{i+1j} + \vec{a}_{ij+1} + \vec{a}_{i+1j+1})$ and $\vec{v}$ is the viewpoint vector.

J.2.2.2 Rendering criterion:

$$R_t = \frac{p}{q}$$

p is the distance between viewpoint and projection plane normalized to window width. p may be computed using:

$$p = \frac{1}{2\tan(FOV/2)}$$

$$p = \frac{1}{2\tan(FOV/2)}, \text{FOV is the Field Of View}$$

where FOV is the Field of View specified in radians, and q = <TextureWidth>/<WindowWidth> where the texture width is the width of the full texture (1024 for instance) and the WindowWidth is the width of the rendering window.

J.2.2.3 Orientation criteria:

$$Ra = \cos(\alpha)$$
$$Rb = \cos(\beta)$$

The angle between the aiming direction and the normal of the current cell center shall be projected into two planes. These two planes are spans of normal vector $\vec{n}$ of the cell and the cell edges in x and y directions, respectively. Then the angles (α, β) between projected vectors and the normal $\vec{n}$ shall be calculated, respectively.

The angle β is specified as the projection of the angle between $\vec{n}$, the normal of the quad cell, and $\vec{u}$, the aiming direction, onto the plane Π_x that passes through $\vec{g}_x$ and is parallel to $\vec{n}$. Similarly, the angle α is specified as the projection of the same angle onto the plane Π_y that passes through $\vec{g}_y$ and its parallel to $\vec{n}$.

This is illustrated in Figure J-3

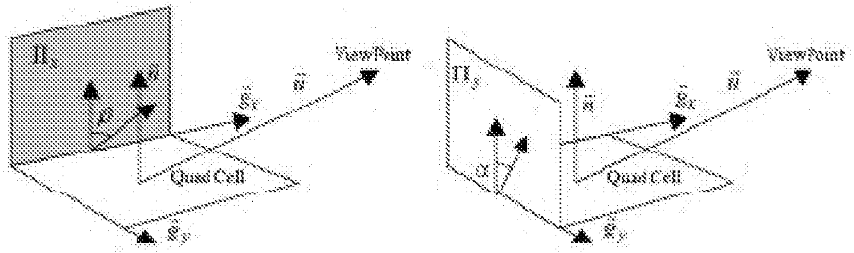


Figure J-3 -- Definition of α and β angles

J.2.2.4 Cropping criterion:

Cells that are out of the field of view shall not be transmitted/received: that is, at least one of the 4 vertices which define the cell should all be inside the horizontal and vertical Field Of View (FOV).

The horizonal FOV shall be deduced from the vertical FOV using the screen geomery. The vertical FOV is equal to the FOV. Then the following shall be calculated

$HFOV = A\tan(\frac{w}{h}\tan(FOV/2))$ where w and h are the width and height, respectively, of the rendered image.

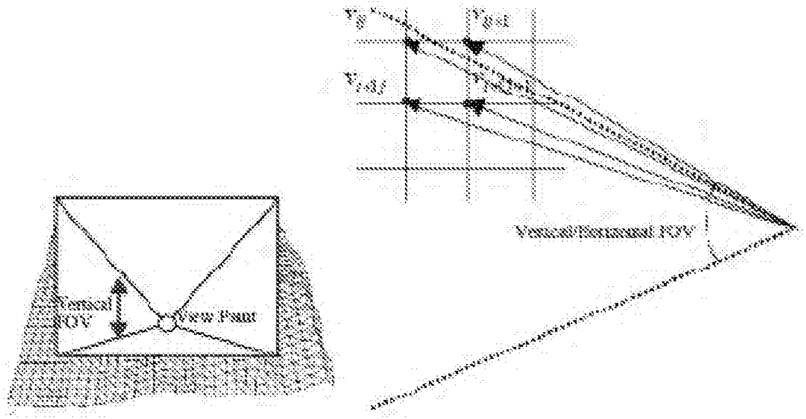


Figure J-4 -- Definition of Out of Field of View cells

J.2.3 VD mask computation

The VD mask s a binary image of the same size as the texture image. Each value in the mask shall indicate if the corresponding DCT coefficient is needed (1) or not (0), given the VDS parameters.

For each cell, the following rules shall be applied to fill the corresponding 8x8 block of the VD mask:

- **Use of cropping criterion**: If all the vertices of the cell are out of the field of view, the corresponding 8x8 block of the mask image shall be set to 0.
- **Use of rendering, distance, tilting and rotation criteria**: For each 8x8 block of the mask (corresponding to a quad cell), the 4 criteria mentioned above shall be computed. Two values of the rotation and tilting criteria shall be obtained for a quad cell, but only the higher value of each criterion shall be kept.

Two thresholds, T_X and T_Y, shall be calculated as the product of the three VDS parameters *Rr*, *Rd*, *Rb*, and *Rr*, *Rd*, *Ra*, respectively, and the value 8. The results shall be bounded to 8. This procedure may be indicated symbolically as follows

$$Tx = Min\left(8, 8 \cdot R_r \cdot R_d \cdot R_b\right)$$
$$Ty = Min\left(8, 8 \cdot R_r \cdot R_d \cdot R_a\right)$$

The flag (i,j) of the 8x8 block corresponding to the current cell shall be set to 1 if $i < T_X$ and $j < T_Y$. The flag shall be set to 0 in all other cases, as illustrated in the figure below.

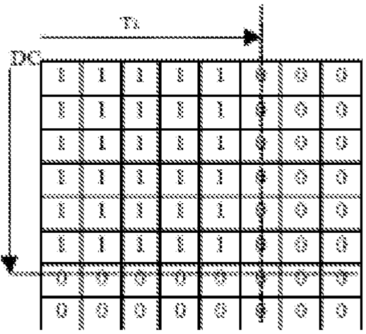


Figure J-5 -- VD mask of an 8x8 block using VD parameters

J.2.4 Differential mask computation

Once the first image has been received using the previously described filter, less data is necessary to update the texture data for the following frames. (assuming there is a correlation between viewpoint positions). Since this computation is exactly the same for each flag of each cell, it shall be performed directly on the full mask images and not on a cell by cell basis.

If the coefficient has not been already transmitted (buffer mask set to 0) and is needed according to VDS visibility criteria (VD mask set to 1), then the corresponding pixel of the differential mask shall be set to 1. This implies that the texture shall be updated, according to the procedure described in subclause J.2.6.

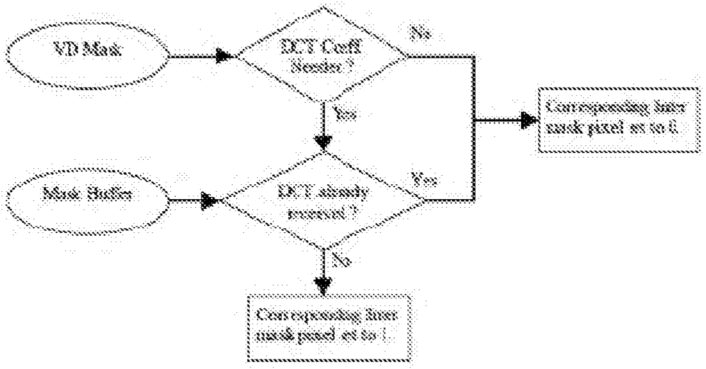

VD Mask
DCT Coeff Needed ?
No
Yes
Corresponding filter mask pixel set to 0.
Mask Buffer
DCT already received ?
Yes
No
Corresponding filter mask pixel set to 1.

Figure J-6 -- Differential mask computation scheme

J.2.5 DCT coefficients decoding

The DCT coefficients shall be decoded using the Video Intra mode, Separated Texture/Motion mode as described in the subclause 7.4.

J.2.6 Texture update

The Differential mask image shall be used to select which DCT coefficients of the buffered texture should be updated using the decoded DCT coefficients.

- Y component

If the Differential mask is set to 0, the corresponding DCT value of the buffer shall be left unchanged, otherwise the value shall be updated with the previously decoded DCT coefficient.

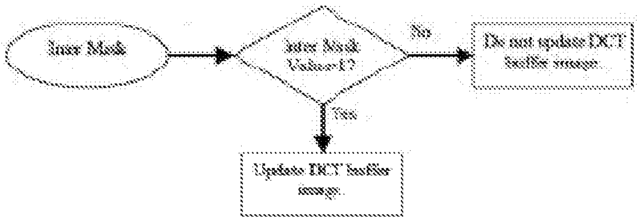


Figure J-7 – Texture update scheme

- U and V component

The texture is coded in 4:2:0 format, as specified in subclause 6.1.3.6, which shall imply that for each chrominance DCT coefficient, 4 Differential mask flags shall be available. The chrominance coefficients shall be received/transmitted if at least 1 of these 4 flags is set to 1.

J.2.7 IDCT

The IDCT and de-quantisation shall be performed using the same process as in the Video Intra mode as described in subclause 7.4.

Annex K
(normative)

Decoder Configuration Information

K.1 Introduction

This annex identifies the syntax elements defined in the main body of this part of ISO/IEC 14496 that describe the configuration of the visual decoder. This Annex amplifies the information provided in subclause 6.2.1. These elements shall be processed differently if this part of the specification is used jointly with the Systems part, ISO/IEC 14496-1:1999. Instead of conveying the configuration at the beginning of a visual elementary bitstream, it shall be conveyed as part of an Elementary Stream Descriptor that is itself included in an Object Descriptor describing the visual object. This descriptor framework is specified in ISO/IEC 14496-1:1999.

K.2 Description of the set up of a visual decoder (informative)

The process of accessing ISO/IEC 14496 content is specified in ISO/IEC 14496-1:1999. It is summarized here and visualized in Figure L.1 to outline the processing of decoder configuration information in an ISO/IEC 14496 terminal. This description assumes that all elementary streams are accessible and solely the problem of identifying them is to be solved.

The content access procedure starts from an Initial Object Descriptor that may be made available through means that are not defined in this part of ISO/IEC 14496, like a http or ftp link on an HTML page, a descriptor in an ISO/IEC 13818-1 Transport Stream, or some H.245 signaling, etc. Note that this may need standardisation by the responsible group.

This Initial Object Descriptor contains pointers at least to a scene description stream and an object descriptor stream. The scene description stream conveys the time variant spatiotemporal layout of the scene. For each streaming media object incorporated in the scene, an Object Descriptor exists that describes the set of streams associated to this media object. The set of Object Descriptors is conveyed in a separate stream, in order to distinguish scene description from the description of the streaming resources.

Both the scene description stream and the object desciptor stream allow for time stamped updates of the scene description and the object descriptors, respectively. Due to the time stamps it is always known at which point in time, or from which point in time onwards a data item, called Access Unit, is valid.

The Object Descriptor associated to a given visual object is identified by its ObjectDescriptor_ID. Each visual object may require more than one elementary stream (ES) that convey its coded representation, especially if any form of scaleability is used. Each of these streams is described by an ES_Descriptor. This description contains a unique label for the stream, the ES_Id, and, among others, the DecoderSpecificInfo structure that is of concern for the purpose of this Annex.

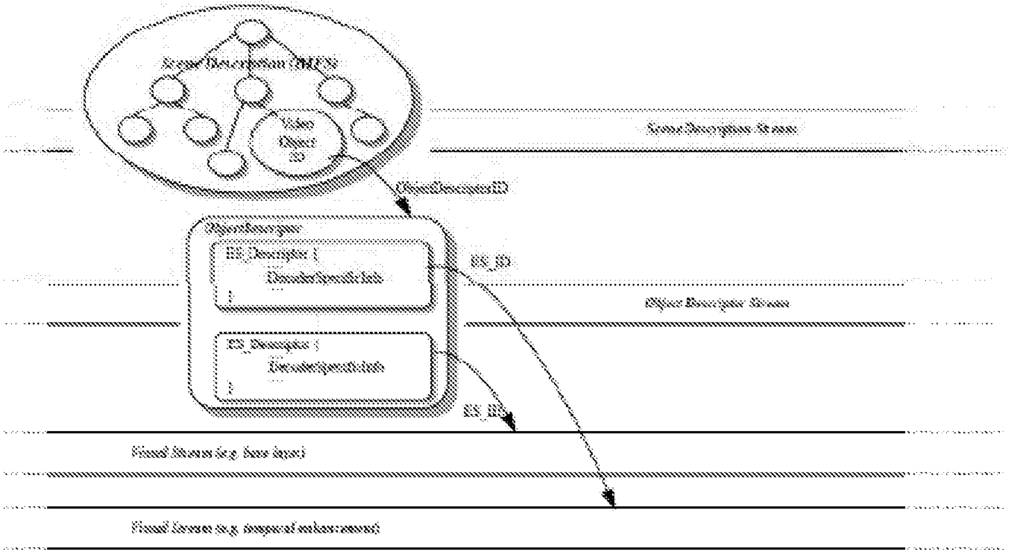


Figure K-1 -- Visual decoder setup

K.2.1 Processing of decoder configuration information

After the retrieval of the Object Descriptor for a media object a decoder for the visual stream(s) is instantiated, connected to the stream(s) and initialised with the data found in ES_Descriptor.DecConfigDescr.DecSpecificInfo for each stream. Subsequently, in a random access scenario, the decoder is expected to search forward to the next random access point, while in a client-server or local storage scenario, data in the visual stream may already be aligned in a way that the first data arriving in the visual stream corresponds to a random access point.

The difference between a visual-only scenario, as specified in the main body of this part of ISO/IEC 14496, and an integrated application using both ISO/IEC 14496-1:1999 and this part of ISO/IEC 14496 is visualized in a figure. Figure K-2 shows the Visual-only approach with configuration information at the beginning of a bitstream and optionally repeated thereafter to enable random access. Figure K-3 with the integrated Systems and Visual approach shows the plain bitstreams with the configuration information extracted into the object descriptors. In this case the object descriptors will be repeated if random access is desired. The Access Units shown in the figure correspond to VOPs in the case of visual media streams.

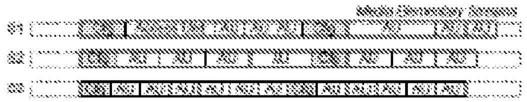


Figure K-2 -- Visual-only scenario

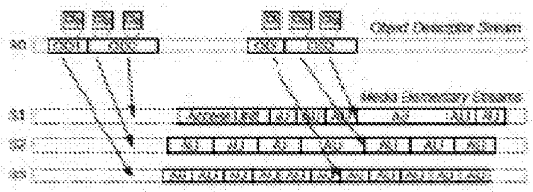


Figure K-3 -- Integrated Systems and Visual approach

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/PODJA,Y. ADAMANE
ANSI Store order X027445. Downloaded 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

K.3 Specification of decoder configuration information

The decoder configuration information for a visual elementary stream is given by a concatenation of those syntax elements that precede the actual encoded data, i. e. that form the 'stream header' according to the syntax specification in subclause 6.2. Those syntax elements are identified separately for each type of visual object in the subsequent subclauses. The syntax elements that are conveyed as decoder configuation information shall not be present in the visual elementary stream itself. Furthermore, the generic syntax definition in subclause 6.2 is constrained to clarify that an elementary stream may not contain a concatenation of, for example, multiple VisualObject() structures or multiple VideoObjectLayer() structures.

The decoder configuration information shall be conveyed in the DecConfigDescr.DecSpecificInfo in field of the respective ES_Descriptor and passed to the decoder before any data of the visual elementary stream itself.

VisualObjectSequence() is transmitted explicitly in each DecConfigDescr.DecSpecificInfo of an Object Descriptor.

Identification and priority of objects is signaled genericaly for all kinds of audiovisual objects in the ES_Descriptor. This signalling overrides the value of VisualObject.is_visual_object_identifier and VideoObjectLayer.is_object_layer_identifier.

VisualObject() carries information describing all the elementary streams associated with an object and is carried in a container associated with all the layers of this object.

VideoObjectLayer() carries information describing a single elementary stream, i.e. a single-layer object or one layer of a multi-layer object. This information is carried in a container associated with the corresponding elementary stream.

K.3.1 VideoObject

The decoder configuration information for a visual object of type VideoObject consists of the following elements:

- All syntax elements of VisualObjectSequence()

including all syntax elements of one VisualObject()

  - including all syntax elements of one VideoObject()
  - including all syntax elements of one VideoObjectLayer() excluding the trailing Group_of_VideoObjectPlane() and VideoObjectPlane() that convey the coded data.

The elementary stream consists of the Group_of_VideoObjectPlane(), VideoObjectPlane(), video_plane_with_short_header and all lower layers.

K.3.2 StillTextureObject

The decoder configuration information for a visual object of type StillTextureObject consists of the following elements:

- All syntax elements of VisualObjectSequence()
  - including all syntax elements of one VisualObject()
  - including all syntax elements of one StillTextureObject() up to but not including the first call to wavelet_dc_decode() in case of visual_object_verid == "0001".
  - including all syntax elements of one StillTextureObject() up to but not including the first call to shape_object_decoding() in case of visual_object_verid != "0001" and tiling_disable=="1" and texture_object_layer_shape == "01"

- Including all syntax elements of one StillTextureObject() up to but not including the first texture_tile_start_code in case of visual_object_verid != "0001" and tiling_disable=="0".

K.3.3 MeshObject

The decoder configuration information for a visual object of type MeshObject consists of the following elements

- All syntax elements of VisualObjectSequence()

including all syntax elements of one VisualObject()

K.3.4 FaceObject

The decoder configuration information for a visual object of type FaceObject consists of he following elements

- All syntax elements of VisualObjectSequence()

including all syntax elements of one VisualObject()

K.3.5 3DMeshObject

The decoder configuration information for a visual object of type 3DMeshObject consists of the following elements

- All syntax elements of VisualObjectSequence()

including all syntax elements of one VisualObject()

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/ROYAL Y ADAMSKE
ANSI Store order #X274451 Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

Annex L
(informative)

Rate control

L.1 Frame Rate Control

L.1.1 Introduction

Rate control and buffer regulation is an important issue for both VBR and CBR applicatons. In the case of VBR encoding, the rate controller attempts to achieve optimum quality for a given target rate. In the case of CBR encoding and real-time application, the rate control scheme has to satisfy the low-latency and VBV buffer constraints. In addition, the rate control scheme has to be applicable to a wide variety of sequences and bit rates.

The scalable rate control (SRC) scheme is designed to meet both VBR without delay constraints and CBR with low-latency and buffer constraints. The SRC scheme is scalable for various bit rates (e.g. 10kbps to 1Mbps), various spatial resolutions (e.g. qcif to cif) and various temporal resolutions (e.g. 7.5fps to 30fps) and various coders (e.g. DCT and wavelet). The technique can handle I, P, and B pictures. The SRC is for single VO, and extensions to M-VOs and other improvements are being addressed in clause L.2 The current descripton in clause L.1 only handles I and P pictures.

L.1.2 Description

The SRC scheme assumes that the encoder rate distortion function can be modeled as

$$R=X1*S*Q^{-1}+X2*S*Q^{-2}$$

The encoding bit count is denoted as R. The encoding complexity which is mean absolute difference (MAD) is denoted as S. The quantisation parameter is denoted as Q. The modeling parameters are denoted as X1 and X2. Because of the generality of the assumption, the scalable rate control scheme is applicable to a variety of bit rates, spatial resolutions, temporal resolutions, buffer constraints and types of coders.

There are four steps in the SRC scheme:

1). Initialization
    - X1 and X2 are the first and second order coefficients.
2). Computation of the target bit rate before encoding.
    - The target bit rate is computed based on the bits available and the last encoded frame bits. If the last frame is complex and uses excessive bits, more bits should be assigned to this frame. However, there are fewer bits left for encoding. Thus, fewer bits can be assigned to this frame. A weighted average reflects a compromise of these two factors.
    - A lower bound of target rate (R/30) is used so that minimal quality is guaranteed.
    - The target rate is adjusted according to the buffer status to prevent both overflow and underflow.
3). Computation of the quantisation parameters (QP) before encoding.
    - QP is solved based on the model parameters X1 and X2.
    - QP is clipped between 1 and 31.
    - QP is limited to vary within 25 percent of the previous QP to maintain a VBR quality.
4). After encoding model parameters is updated based on the encoding results of the current frame.

- The rate distorton model is updated based on the encoding results of he current frame. The bits used for the header and the motion vectors are deducted since they are not related to QP.
- The data points are selected using a window whose size depends on the change in complexity. If the complexity changes significantly, a smaller window with more recent data points is used.
- The model is calibrated again by rejecting he outlier data points. The rejection criteria are that data point is discarded when the prediction error is more than one standard deviation.
- The next frame is skipped if the current buffer status is above 80 percents.

The algorithm s described as follows:

Step 1: Initialization after the first frame is coded

Rs: bitrate for the sequence (or segment). /* e.g., 24000 bits/sec */
Rf: bits used for the first frame. /* e.g., 10000 bits */
Rc: bits used for the current frame. It is the bit count obtained after encoding.
Rp: bits to be removed from the buffer per picture.
Ts: number of seconds for the sequence (or segment). /* e.g., 10 sec */
Ec: mean absolute difference for the current frame after motion compensation.
If he macroblock is intra coded, the original spatial pixel values are summed.
Qc: quantisation level used for the current frame.
Nr: number of P frames remaining for encoding.
Ns: distance between encoded frames. /* e.g., 4 for 7.5 fps */
Rr: number of bits remaining for encoding this sequence (or segment).
T: target bit to be used for the current frame.
S: number of bits used for encoding the previous frame.
Hc: header and motion vector bits used in the current frame. It includes al the information except to the residual information.
Hp: header and motion vector bits used in the previous frame. It includes all the information except to the residual information.
Ql: quantisation level used in the previous frame
Bs: buffer size e.g., R/2
B: current buffer level e.g., R/4 - start from the middle of the buffer

```
Rr = Ts*Rs-Rf;          /* total number of bits available for this segment */
Rp = Rr/Nr;             /* the average bits to be removed from the buffer */
if(first frame){
    Qc=15;
    goto step 4
}
```

Step 2: Target Bit Calculation (Before Encoding the Current Frame)

```
T=Max(Rs/30,Rr/Nr*0.95+S*0.05);/*each frame is assigned a minimum of R/30 */
T=T*(B+2*(Bs-B))/(2*B+(Bs-B)); /* increase if less han half */
                    /* decrease if more than half, don't change if half */
if (B+T > 0.9*Bs)
    T = Max(Rs/30, 0.9*Bs-B);   /* to avoid overflow*/
else if (B-Rp+T < 0.1*Bs)
    T = Rp-B+0.1*Bs;            /* to avoid underflow*/
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
Single user licence only. Copying and networking prohibited.

Step 3: Quantisation Level Calculation (Before Encoding the Current Frame)

```
T= Max(Rp/3+Hp, T);

if ((X2==0) || (((X1*Ec)**2+4*X2*Ec*(T-Hp))<0))
     Qc = X11*Ec/(T-Hp);          /* fall back 1st order mode */
else Qc= (2*X2*Ec)/(sqrt((X1*Ec)**2+4*X2*Ec*(T-Hp))-X1*Ec)
                         /* 2nd order mode */

Qc = Min (ceil(Ql*1.25), Qc, 31); /* clipping*/
Qc = Max (ceil(Ql*0.75), Qc, 1);  /* clipping */
Quantisation();
Encoding();
```

Step 4: After encoding the current frame

```
B += Rc - Rp;           /* update buffer fullness */
Rr -= Rc;               /* update the remaining bit counts */
S = Rc;              /* update the previous bit counts */
Hp = Hc;                /* update the previous header and motion bit counts */
Ql = Qc;                /* update the previous quantisation level */
Nr--;                   /* update the frame counter*/
UpdateRDModel(Qc,Rc,Ec,Hc,X1,X2);
                     /* estimation of a new model */
while (B > 0.8 * Bs) {
     skip_next_frame();
     Nr--;
     B -= Rp;
}
```

If the buffer has reached 80% of the buffer size, the encoder will skip the upcoming frame to be encoded. Thus the buffer is reduced to prevent from the buffer overflow.

```
UpdateRDModel (Qc, Rc, Ec, Hc, X1, X2) {
    Qall[n]: quantisation levels for the past frames
    Rall[n]: scaled encoding complexity used for the past frames
    n: number of encoded past frames;
    x: matrix contains Q;
    y: matrix contains Q*(R-H)/E;
    Ep: mean absolute difference for the previous frame. This is computed after motion compensation for
    the Y component only.

    Rall[n] = (Rc-Hc)/Ec;
    Qall[n] = Qc;

    w = Min(total_data_number, 20) /* Maximum data number set to 20 */
    if(Ep>Ec)
      w = ceil(Ec/Ep*w);            /* sliding window for scene change */
    else
      w = ceil(Ep/Ec*w);            /* sliding window for scene change */
    Ep = Ec;                    /* update mad */

    for(i=n; i>n-w; i++){
        Qp[i]=Qall[i]
        Rp[i]=Rall[i]
    }
    Estimator(Qp,Rp,w,b);
    RemoveOutlier(X1,X2,Qp,Rp,Ec,w);
    Estimator(Qp,Rp,w,b);
}

Estimator(Qp, Rp,w,b) {
    X11: First order coefficient.
    x = [1, Qp[i]**(-1)( i=1,2,...w)]            /* Dimenson wx2 */
    y = [Qp[i]*Rp[i] (i=1,2,.....w)]       /* Dimension wx1 */

    for (i=1; i<=w; i++)   X11 += y[i]/w;

    if (all Qp[i] are not all the same) {
        b = (x_Transpose*x)**(-1)*x_Transpose*y;
        /* Dimension of the matrix */
        /* 2x1 = (2xw * wx2)**(-1)*(2xw)*(w*1) */
        X1 = b(1,1);
        X2 = b(2,1);
    }
}
```

```
RemoveOutlier(X1,X2,Qp,Rp,Ec,w) {
    error[w];      Estimation error
    new_w;

    for (i=1; i<=w; i++) {
        std += ((X1*Ec*Qp[i]^(-1)+X2*Ec*Qp[i]^(-2)-Rp[i]*Ec))**2;
        error[i] = X1*Ec*Qp[i]^(-1)+X2*Ec*Qp[i]^(-2)-Rp[i]*Ec;
    }

    /* When w is 2 the threshold should be 0, some implementation may resolve a small number due to
    precision problem. */
    threshold = sqrt(std/w);              /* Setup rejection threshold */

    new_w = 0
    for (i=1; i<=w; i++){
        if(abs(error[i]) <= threshold){
            new_w++;
            Qp[new_w] = Qp[i];
            Rp[new_w] = Rp[i];
        }
    }
    w = new_w;
}
```

L.1.3 Summary

The scalable rate control scheme achieves frame leve rate control for both VBR and CBR cases. It assumes a simple quadratic rate distortion function of the video encoder. In the case of CBR encoding, a variable frame rate approach is used to achieve the target rate. If a tighter rate control is desired, the same technique is applicable at either slice layer or macroblock layer. Because of the generality of this scheme, extension of such a scheme to shape coding and Multiple-VO coding can be easily made on the basis of the existing rate control framework.

L.2 Multiple Video Object Rate Control

This clause describes an algorithm which achieves a constant bit rate when coding multiple video objects. The scheme assumes the following relationship between the bits and quantisation parameter for the i-th object:

R[i]=X1[i]*S[i]*Q[i]**(-1)+X2[i]*S[i]*Q[i]**(-2)

The above is a simple extension of the relationship used for S-VOP rate control when sprite_enable == 'static'.

As shown in Figure L-1, the algorithm for performing the joint rate control can be decomposed into a number of steps, which may be grouped into a pre-encoding and post-encoding stage. Each block is briefly described below.

L.2.1 Initialization

The initialization process is very similar to the S-VOP process when sprite_enable == 'static'. Since a single buffer is used, the buffer drain rate is the same. The major difference is that most parameters are extended to vector quantities.

L.2.2 Quantisation Level Calculation for I-frame and first P-frame

To calculate the quantisation parameters which are used for the I-frame and first P-frame, a bit allocation table based on Human Visual Sensitivity (HVS) of color tolerance is used. This scheme assigns a target bit rate which is predicted by histogram of variance and HVS classification.

Color/Variance classification: A macro block is classified into one of 32 macro block casses based on HVS and one of 16 classes based on the variance. A reference quantisation step size is assigned to the macro block according to its macro block class. A macro block is classifed into a sensitive category if the number of pixels in the macro block which satisfy LUT (Look-Up Table) is larger than a threshold value. Otherwise the macro block is classified into nsensitive category.

$$texture\_class = H_v + V_v$$

$$H_v = \sum_{y=1}^{16}\sum_{x=1}^{16}\left(\left|P_{x,y} - P_{x-1,y}\right| > th\right)$$

$$V_v = \sum_{y=1}^{16}\sum_{x=1}^{16}\left(\left|P_{x,y} - P_{x,y-1}\right| > th\right)$$

$$th = f\left(\sum_{y=1}^{16}\sum_{x=1}^{16} P_{x,y}\right)$$

$$color\_class = \min(color\_tolerance)$$

$$color\_tolerance = LUT(color_{mean})$$

$$color_{mean} = \frac{1}{64}\sum_{y=1}^{16}\sum_{x=1}^{16}\left(P_{x,y}\right)$$

The block variance classification allows us to estimate the number of coded bits in a block. Generally, the number of coded bits generated in a block is proportional to the block variance when the quantisation step size is assumed to be constant. We divide the magnitude of block variance into 16 intervals. Threshold values in each interval can be found in Table L-1. Table L-2 illustrates the reference quantisation step size based on HVS for each macro block class and picture type.

Table L-1 -- Block variance classes

Bv VOP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I	16	32	64	128	256	384	512	768	1024	1280	1792	2048	3072	4096	6154	¥
P	8	16	32	48	64	96	128	160	192	256	320	448	640	896	1536	¥
Bv : Block Variance Class Number, I : INTRA FRAME TYPE, P : INTER FRAME TYPE																

Table L-2 -- Reference quantisation step size

Mc	0	1	2	3	4	5	6	7
Qss	-4	-2	-1	0	1	2	3	4

Initial QP setting method

```
SetInitialQP()
{
    PrriDiff = 1000000;
    Classification();
    if(flat image)        TargetBit = AverageBitPerFrame * 9;
    else                TargetBit = AverageBitPerFrame * 4;
    /* Find Optimal Qp value */
    for(i=1; i<31; i++){
     SetMB_QP(i); /* Allocate HVS optimal QP to each MB */
     EstimatedBit = Bitestimation(); /* Pre-bit-estimation */
     Diff = abs(TargetBit - EstimatedBit);
     if(Diff<PreDiff){
        PreDiff = Diff;
        MinErrorQ = i;
     }
    } /*  MinErrorQ is the final average Qp value*/
    SetMB_QP(MinErrorQ); /* Allocate HVS Optimal Qp to each MB */
}
```

Frame target bit allocation

$$\text{FrameTargetBit} = \begin{cases} \frac{Bitrate}{fr} * (4 \sim 9) \text{------Intra frame} \\ \frac{Bitrate}{fr} \text{--------First Inter Fr.} \end{cases}$$

Frame coded-bit estimation: There are 512 classes which are combinations of 32 macro block color classes(Mc) and 16 block variance classes(Bv). We can estimate the total number of coded bits in a frame with the histogram and the bit model table. **FrameEstimatedBit** is the estimated number of coded bit in the current frame

$$\text{FrameEstimateBit} = \sum_{Mc=0}^{31} \sum_{Bv=0}^{15} \{if\,(Coded)\,B[R[Mc]][Bv] * H[Mc][Bv]\} + \text{ShapeBit}$$

In the above equation, B[Mc][Bv] is the number of bits in the bit table and H[Mc][Bv] is the computation result of the histogram when a block is classified into Bv. The histogram can be computed with the information of macro block classes and block classes of the previous frame because the current frame to be coded and the previous frame are very highly correlated. Therefore the real-time coding process is possible.

Bit Table

The number of macro block classes and block classes in a frame depends on the statistical characteristics of the frame. However, the target number of bits which is assigned to the frame by bit allocation is not related with statistical characteristics of the frame. Hence the target number of bits for frame has much difference from the total number of coded bits in the frame when all macro blocks in the frame are quantized with their reference quantisation step sizes. So reference quantisation step sizes should be updated in order that the total number of coded bit in the frame becomes almost the same as the number of target bits for the frame. To determine the ratio of change of reference quantisation step size, we have to estimate the total number of coded bits in a frame when all macro blocks in the frame are quantized with their reference quantisation step sizes. The bit table shows the average value of the estimated number of bits in a block according to macro block classification and block classification in Intra/Inter frame. It is possible to estimate the total number of coded bits in a frame with the bit table.

```
(INTRA FRAME Bit Table)
int BitTable[32][16] =
{
65,145,174,208,242,278,298,317,341,352,368,386,400,418,435,438,
24, 79,102,127,153,178,194,209,232,243,255,268,272,279,288,292,
16, 52 70, 93,114,136,148,161,180,190,201,214,220,226,231,234,
12, 38 53, 73, 91,110,122,132,150,159,166,179,185,188,193,195,
11, 29 42, 59, 75, 92,104,112,131,138,144,154,153,160,163,168,
10, 24 35, 49, 64, 79, 91, 98,116,122,127,135,137,142,148,159,
 9, 19, 29, 42, 55, 69, 80, 87,103,109,116,122,126,130,137,147,
 9, 17, 25, 37, 48, 61, 72, 79, 95,100,105,111,114,119,125,135,
 9, 15, 22, 33, 43, 55, 65, 71, 86, 91, 97,103,106,110,115,123,
 8, 13, 19, 29, 39, 50, 58, 65, 79, 84, 89, 95, 98,100,103,114,
 8, 12, 18, 26, 35, 45, 54, 60, 73, 78, 84, 88, 89, 95, 99,104,
 8, 11, 16, 24, 32, 42, 51, 55, 68, 74, 79, 83, 85, 91, 94, 98,
 8, 11, 15, 22, 30, 38, 47, 51, 63, 69, 74, 78, 81, 86, 88, 90,
 8, 10, 14, 21, 28, 35, 44, 48, 59, 64, 69, 75, 77, 79, 80, 83,
 8, 10, 13, 19, 26, 33, 41, 45, 55, 61, 65, 71, 71, 74, 76, 78,
 8,  9, 12, 18, 24, 31, 39, 42, 52, 58, 61, 66, 66, 70, 72, 75,
 8,  9, 12, 17, 23, 29, 37, 40, 48, 55, 57, 62, 63, 65, 69, 73,
 8,  9, 11, 16, 21, 27, 35, 37, 46, 51, 55, 60, 60, 64, 66, 69,
 8,  9, 11, 14, 20, 26, 33, 36, 43, 48, 52, 56, 59, 63, 65, 66,
 8,  9, 10, 14, 19, 25, 31, 34, 41, 45, 50, 55, 57, 61, 64, 66,
 8,  9, 10, 13, 18, 23, 30, 33, 40, 44, 48, 53, 54, 59, 62, 63,
 8,  8, 10, 12, 17, 22, 28, 31, 38, 42, 46, 50, 52, 57, 60, 62,
 8,  8,  9, 12, 16, 21, 27, 30, 36, 40, 44, 49, 51, 56, 57, 60,
 8,  8,  9, 11, 16, 21, 26, 29, 35, 39, 43, 46, 48, 54, 55, 58,
 8,  8,  9, 11, 15, 20, 25, 28, 33, 38, 41, 44, 45, 51, 54, 57,
 8,  8,  9, 11, 14, 19, 24, 27, 32, 36, 40, 43, 44, 48, 50, 55,
 8,  8,  9, 10, 13, 18, 24, 26, 31, 35, 38, 42, 43, 46, 48, 55,
 8,  8,  8, 10, 13, 17, 23, 25, 29, 33, 37, 41, 41, 43, 46, 55,
 8,  8,  8, 10, 13, 17, 22, 24, 28, 32, 36, 39, 40, 42, 41, 53,
 8,  8,  8,  9, 12, 16, 22, 23, 28, 31, 35, 38, 39, 40, 43, 52,
 8,  8,  8,  9, 12, 16, 21, 22, 27, 30, 34, 36, 37, 40, 43, 52,
};
```

L.2.3 Update Rate-Distortion Model

After the encoding stage, the parameters for the RD model are updated. Given the number of texture bits, R[i], the quantisation level, Q[i], and the mean absolute difference, S[i], for the current and a specified number of past frames, the model parameters, X1[i] and X2[i] can be calculated using a least-squares estimation.

L.2.4 Post-Frameskip Control

The post-frameskip control determines the value of *N_skip_post*. It is invoked after the buffer has been updated, i.e., the sum of bits used to code every VO is added to the current buffer level and the number of bits to be removed per picture is subtracted. Essentially, if the "virtual buffer" has exceeded a specified level *(e.g., 80% of the* buffer size), the encoder will skip the upcoming frames so as to reduce the current bufer level. The virtual buffer level (VB) is given by:

$$VB = B + Bp - (N\_skip\_post+1)*Rp$$

where the new variable, Bp, denotes the number of bits used in the previous frame. As before, B is the current buffer level and Rp is the number of bits removed from the buffer per picture. After the value of *N_skip_post* has been found, the value of *N_skip_pre* is added to it, where *N_skip_pre* is calculated in the pre-frameskip control. The time instant is updated based on the total frames to be skipped, *N_skip* = *N_skip_pre* + *N_skip_post*.

Mode of Operation: Since coding decisions may change according to the environment, it is desirable to have a mechanism to detect and keep an update of such changes.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/RAO,K.R. ADMIN/[illegible]
Single user license only. Copying and networking prohibited.

```
if(N_skip > SkipTH)
    Operate in LowMode
else
    Operate in HighMode
```

The mode of operation can be used to impose constraints on the coding parameters, assist in setting the weights for target distribution, and influence the shape rate control mechanism.

Target Estimation: The initial target bit rate can be calculated based on the number of available bits for the segment and the number of bits used in the previous corresponding VO. A similar lower bound to the S-VOP simulation can be used to target minimum quality.

Joint Buffer Control: The joint buffer control symbolizes the same operation as buffer control in S-VOP with sprite_enable == 'static', however, the target is a sum of individual targets, i.e., T = sum of T[i].

Pre-FrameSkip Control: Often, in low bit rate coding conditions, the target which emerges from the joint buffer control may not be enough to even code the motion, shape, and header information, let alone the texture. In that case, there should be a mechanism to alert other parts of the system (e.g., target distribution, QP calculation, time instant update) that there is a deficiency in the number of allocated bits. Let s = T - Hp be the difference between the target and the amount of bits used in the previous frame for shape, motion, and header, and Beta be a bit threshold greater than zero. N_skip_pre is determined by:

```
while(s < Beta)
    N_skip_pre = N_skip_pre + 1
    s = s + Rd
```

Target Distribution: The joint buffer control provides a new total target to be distributed among each VO. The distribution can be based on the size, the motion, and the variance (MAD^2) of each object:

T[i] = (MOTION[i]*wm + SIZE[i]*ws + MAD2[i]*wv)*T

This type of distribution is only executed when the target is valid. The target is valid when it is greater than the sum of header bits used for coding the previous frame. If it is not valid, all the targets are made negative. The negative targets can serve as a flag for clipping the quantisation parameters to lie in a specified range [Oc 31].

Shape Rate Control: The value of AlphaTH has considerable effect over the number of bits which will be spent on the shape information. Using the current mode of operation, an appropriate value can be determined.

- If the mode of operation is LowMode, then increment the current AlphaTH by AlphaINC;

- If the mode of operation is HighMode, then decrement the current AlphaTH by AlphaDEC.

The minimum value of AlphaTH is 0. This implies lossless shape encoding. The largest value that AlphaTH may take is 255, however, it is recommended that an AlphaMAX in the range [32, 64] be specified.

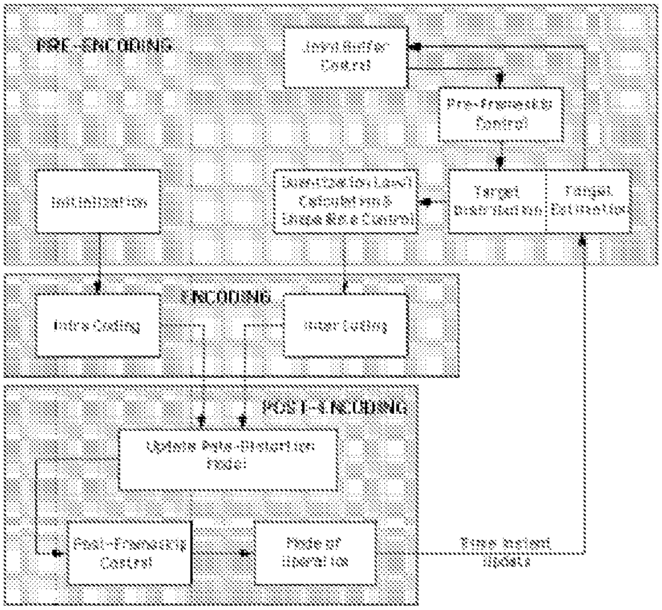


Figure L-1 -- Block diagram for multi-VO rate control

L.3 Macroblock Rate Control

The rate control method described in clause L.1 used a fixed quantisation parameter QP for all the macroblocks in a VOP, where the value of QP was chosen to encode the VOP with a target number of bits. In this clause we describe a method for adapting the value of QP at each macroblock within the VOP. This technique achieves the VOP target more accurately, which is useful for low-delay video applications with small buffer constraints, and allows adapting QP with the macroblock energy and other perceptually relevant measures. The method attempts to combine the strengths of the following contributions:

- A model of the human visual system as described in clause L.2 can be used to compute W_i.
- The model with two parameters maintains a near-constant value of QP for all the macroblocks in the VOP. This approach is more appropriate at lower bit rates, since it requires a small number of bits for encoding the QP's (using DQUANT).
- The model with single parameter adapts QP with the energy of the block by using finer quantisation (smaller QP) for macroblocks of flatter image regions. It requires a larger QP bit overhead and hence is more appropriate at higher bit rates.

L.3.1 Rate-Distortion Model

We model the number of bits B_i produced for the i-th macroblock as follows:

$$B_i = \begin{cases} \dfrac{A_1}{QP_i^2}\mathrm{MAD}_i^2; & \mathrm{Bitrate} > R \\ \dfrac{A_2}{QP_i^2}\mathrm{MAD}_i + \dfrac{A_3}{QP_i}\mathrm{MAD}_i; & \mathrm{Bitrate} \leq R \end{cases} \qquad (1)$$

- A1, A2 and A3 are the model parameters.
- Bitrate is the rate available (in bits per pixel) for the texture of the VOP.
- R is a threshold. By default, we set R=0.085.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/OUPOOJA Y. ADAPANE
ANSI Store order X027445 Downloaded: 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

L.3.2 Target Number of Bits for Each Macroblock

Let N be the number of macroblocks in a VOP. The target number of bits, T_i, for the i-th macroblock is:

$$T_i = \frac{W_i\,\mathrm{MAD}_i}{\sum_{j=1}^{N} W_j\,\mathrm{MAD}_j} T_{\mathrm{TEX}}, \qquad (2)$$

where:

- W_i is a weight that indicates the perceptual importance of the i-th macroblock. By default, we set $W_1 = W_2 = \ldots = W_N = 1$, but they could be obtained from similar techniques as described in clause L.2.
- T_{TEX} is the target number of bits for the texture or DCT coefficients (luminance and chrominance components) of the VOP, which is the target frame bits T from frame-layer rate control [see clause L.1] minus the bits for motion, shape, and syntax overhead. If the latter number of bits is not available for the current VOP, one can subtract those from the previous VOP. Bitrate in (1) is T_{TEX} divided by the number of pixels in the VOP.

Observe that if we combine (1) and (2), we obtain the following expression:

$$QP_i^2 = \begin{cases} \mathrm{MAD}_i\,C_1; & \mathrm{Bitrate} > R \\ C_2; & \mathrm{Bitrate} \le R \end{cases} \qquad (3)$$

where the values of C_1 and C_2 depend on { A_1, MAD_1, ..., MAD_N, T } and {A_2, A_3, MAD_1, ..., MAD_N, T}, respectively. The model parameters A_1, A_2, and A_3 are estimated while encoding and in practice they are nearly constant within a VOP, and hence C_1 and C_2 are also approximately constant. As a result, observe that for large Bitrate, the quantisation parameter QP will increase with the MAD energy of the macroblock, and for low Bitrate, QP will be nearly constant for all the macroblocks in the VOP.

L.3.3 Macoblock Rate Control

Step 1 Initialization

N - number of macroblocks in the VOP.
T_{TEX} - number of bits available for encoding the VOP texture

If first frame, let A1= A1_prev= 100, A2= A2_prev= 400, A3= A3_prev= 0
Otherwise, let A1= A1_prev , A2= A2_prev, and A3= A3_prev.

If Bitrate > R = 0.085, let K=1, otherwise let k=2.
Set macroblock index i=1 and skip= 0.

Finally, let $S_1 = \sum_{j=1}^{N} W_j\,\mathrm{MAD}_j$. (By default, set W1= W2= ...=WN = 1.)

Step 2 Compute QP for i_th macroblock

Let $T_i = \frac{W_i\,\mathrm{MAD}_i}{S_i} T_{\mathrm{TEX}}$, the target number of bits per macroblock and:

$$QP = \begin{cases} \mathrm{MAD}_i\sqrt{\frac{A_1}{T_i}}; & K = 1 \\ Q: \frac{A_2}{Q^2}\mathrm{MAD}_i + \frac{A_3}{Q}\mathrm{MAD}_i = T_i; & K = 2 \end{cases}$$

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/[illegible]
ANSI Store order X[illegible] Downloaded: [illegible] PM ET
Single user licence only. Copying and networking prohibited.

If $(A_2MAD_i)^2 + 4\,T_iA_3MAD_i$ is negative, there is no value of Q that solves the second-order equation for K=2. In that case, the term with A_2 is dropped and the following is used instead:

$$QP = \frac{MAD_i A_{3\_X11}}{T_i},$$

where A_{3_X11} is similar to the X11 parameter in the frame-based rate control in clause L.1.2.

Round QP to value in 1, ..., 31, and indicate the difference between the current value of QP and that of the previous macroblock in DQUANT. (Recall that DQUANT can only take values within {-2, -1, 0, 1, 2 }, so clipping may be needed.)

Step 3 Encode Macroblock with QP

Step 4 Update Counters

Let B_i' be the actual number of texture (DCT) bits produced when encoding the i_th macroblock. Compute:

$$T_{TEX} = T_{TEX} - B_i', \quad \text{and} \quad S_{i+1} = S_i - W_i\,MAD_i$$

Step 5 Update A_1, or A_2, A_3, A_{3_X11}

If B_i' is zero (the DCT produced no bits), then skip= skip +1, and go to Step 6.

Otherwise, do the updating:

$$\text{If } K = 1,\ \hat{A}_1 = \frac{B_i'}{MAD_i^2}QP^2.$$

$$\text{Let } A_1' = \hat{A}_1\frac{1}{i-skip} + A_1'\frac{i-skip-1}{i}, \quad A_1 = A_1'\frac{i}{N} + A_{1\_prev}\frac{N-i}{N}.$$

If K =2, use linear regression to estimate A2, A3, and A3_X11, similarly as in subclause L.1.2:
We order the values of B'i, QP, and MADi, for the last W encoded macroblocks (whose B'i > 0) into three sets: {b1, b2, b3, ..., bW}, {q1, q2, ..., qW}, and {m1, m2, ..., mW}, where the bj's, qj's and mj's correspond to the values of B'i, QP, and MADi, respectively, for the j-th macroblock in the set.

Let NC be the number of macroblocks (whose B'i > 0) and set W= min{20, NC }. Define:

P1= [1, 1, ..., 1]T,
P2= [1/q1, 1/q2, ..., 1/qW]T,
P3= [b1q1/m1, b2q2/m2, ..., bWqW/mW]T,
P4= [P1 P2],

where the superscript "T" denotes the transpose of a vector or matrix. P1, P2 and P3 are vectors of dimension W x 1, and P4 is a matrix of W x 2 elements.

Compute:

$$\begin{bmatrix} A_2 \\ A_3 \end{bmatrix} = (P4^T \cdot P4)^{-1} P4^T \cdot P3,$$

and $$A_{3\_X11} = \frac{P3^T \cdot P2}{W},$$

where the superscript "-1" denotes the inverse of the matrix.

Step 6. If i =N, Stop (all macroblocks are encoded), let $A_{f\_prev} = A_f$ for f =1,2,3.

Otherwise, i = i+1, and go to Step 2.

Annex M
(informative)

Binary shape coding

M.1 Introduction

This part of ISO/IEC 14496 will be the first international standard allowing the transmission of arbitrarily shaped video objects (VO). A time instant of a VO is called Video Object Plane (VOP). A VOP is a rectangular video frame or a part thereof. Following an object-based approach the ISO/IEC 14496-2 video encoder applies the motion, texture and shape coding tools to the VOP using I, P, and B modes similar to the modes of MPEG-2. The encoder transmits texture, motion, and shape information of one VO within one bit stream. The bit streams of several VOs and accompanying composition information can be multiplexed such that the decoder receives all the information to decode the VOs and arrange them into a video scene (Figure M-1).

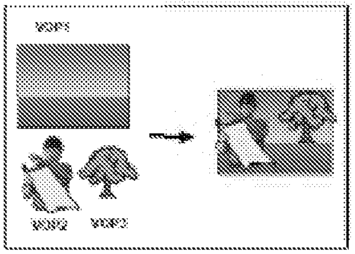


Figure M-1 -- Object-based coding requires the decoder to compose different Video Object Planes (VOP) into a scene.

An arbitrarily shaped VO can have binary shape information only or binary shape and texture information. Several shape coding algorithms were evaluated [1][2]. Binary shape information is transmitted as a bitmap using a context-based arithmetic coder described in clause M.2. Coding of texture for arbitrarily shaped VOs is described in clause M.3. This part of ISO/IEC 14496 leaves it up mainly to the encoder to achieve efficient coding of texture at object boundaries by employing an appropriate texture extrapolation algorithm. The efficient coding of a VO with lossily encoded shapes requires an encoder architecture as described in clause M.4. In clause M.5, guidelines for coding arbitrarily shaped VOs will be presented. Different shape encoding modes are investigated in order to achieve a bit efficient representation of a VO or in order to allow for fast encoding that is important for real-time applications.

M.2 Context-Based Arithmetic Shape Coding

In order to enable content based access to video objects, this part of ISO/IEC 14496 codes the shape of video objects. This part of ISO/IEC 14496 chose to code the shape as a bitmap [2]. For binary shape coding, a rectangular bounding rectangle enclosing the arbitrarily shaped VOP is formed such that its horizontal and vertical dimensions are multiples of 16 pels (macroblock size).

Each block of size 16x16 pels within this bounding rectangle is called binary alpha block (bab). Each bab is associated with the co-located macroblock. Three types of babs are distinguished and signaled to the decoder: Transparent blocks do not contain information about the object, opaque blocks are located entirely inside the object and boundary blocks cover part of the object as well as part of the background. For boundary blocks a context-based shape coder was developed. This coder exploits the spatial redundancy of the binary shape information to be coded. Pels are coded in scan-line order and row by row. In the following paragraphs, shape encoding in intra mode is described first. Then, this technique is extended to include an inter mode.

M.2.1 Intra Mode

In intra mode, three different types of macroblocks are distinguished: Transparent and opaque blocks are signaled as macroblock type. The macroblocks on the object boundary containing transparent as well as opaque pels belong to the third type. For these boundary macroblocks, a template of 10 pels is used to define the causal context for predicting the shape value of the current pel (Figure M-2a). For encoding the state transition, a context-based arithmetic encoder corresponding to the arithmetic decoder as defined in subclause 7.5.3 is used. The probability table of the arithmetic encoder for the 1024 contexts was derived from sequences that are outside of the test set used for comparing different shape coders. With two bytes allocated to describe the symbol probability for each context, the table size is 2048 bytes. In order to avoid emulation of start codes like VOP start code, the arithmetic coder stuffs one '1' into the bitstream whenever a long sequence of '0' is sent.

The template extends up to 2 pels to the left, to the right and to the top of the pel to be coded (Figure M-2a). Hence, for encoding the pels in the 2 top and left rows of a bab, parts of the template are defined by the shape information of he already transmitted babs on the top and on the left side of the current bab. For the 2 right-most columns, each undefined pel of the context is set to the value of its closest neighbor inside the bab. In case the top and left babs are transparent, the top and left 2 rows of the current bab are duplicated in order to define the context for the template for intra mode.

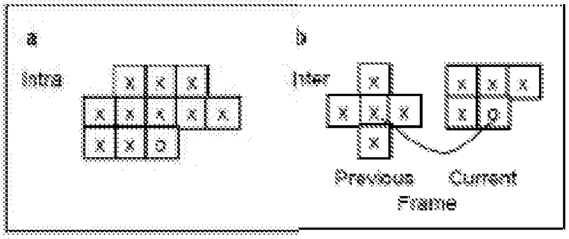


Figure M-2 – Templates for defining the context of the pel to be coded (o), a) defines the intra mode context, b) the context when coding in inter mode. The alignment is done after motion compensating the previous VOP [1].

In order to increase coding efficiency as well as to allow lossy shape coding, a bab can be subsampled by a factor of 2 or 4 resulting in a sub-block of size 8*8 pels or 4*4 pels, respectively. The sub-block is encoded using the encoder as described above. The encoder transmits to the decoder the subsampling factor such that the decoder decodes the shape data and then upsamples the decoded sub-block to macroblock size. Obviously, encoding the shape using a high subsampling factor is more efficient, but the decoded shape after upsampling may or may not be the same as the original shape. Hence, this subsampling is mostly used for lossy shape coding. In order to achieve smooth object boundaries after upsampling, an adaptive non-linear upsampling filter is used. The context of this upsampling filter is shown in Figure M-3.

The efficiency of the shape coder differs depending on the orientation of the shape data. Therefore the encoder can choose to code the bab using horizontal scanning as described above or transpose the macroblock using vertical scanning prior to arithmetic coding.

M.2.2 Inter Mode

In order to exploit temporal redundancy in the shape information, the coder described above is extended by an inter mode requiring motion compensation and a different template for defining the context.

For motion compensation, a 2D integer pel motion vector is estimated using full search for each bab in order to minimize the prediction error between the previous coded VOP shape M'_{k-1} and the current shape M_k. The shape motion vectors are predictively encoded with respect to the shape motion vectors of neighboring macroblocks. If no shape motion vector is available, texture motion vectors are used as predictors. The shape motion vector of the current block is used to align a new template designed for coding shape in inter mode (Figure M-2b). The template defines a context of 9 pels resulting in 512 contexts. The probability for one symbol is described by 2 bytes giving a probability table size of 1024 bytes. Four pels of the context are neighbors of the pel to be coded, 5 pels are located at the motion compensated location in the previous VOP. Assuming that the motion vector (dx,dy)T points from the current vopk to the previous coded vop'k-1, the part of the template located in the previously coded shape is centered at m'(x-dx,y-dy) with (x,y)T being the location of the current pel to be coded.

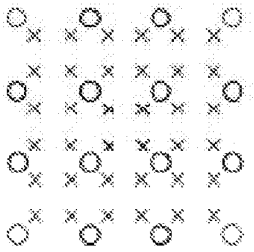


Figure M-3 – For shape upsampling, the upsampled pels (x) lie between the location of the subsampled pels (o). Neighboring pels (bold o) defining the values (transparent or opaque) of the pels to be upsampled (bold x).

In inter mode, the same options as in intra mode like subsampling and transposing are available. For lossy shape coding, the encoder may also decide that the shape representation achieved by just carrying out motion compensation is sufficient thus saving bits by avoiding the coding of the prediction error. The encoder can select one of 7 modes for the shape information of each bab: transparent, opaque, intra, inter with and without shape motion vectors, and inter with/without shape motion vectors and prediction error coding. These different options with optional subsampling and transposition allow for encoder implementations of different coding efficiency and implementation complexity.

M.3 Texture Coding of Boundary Blocks

For motion compensated prediction of the texture of the current VOP, the reference VOP is motion compensated using block motion compensation. In order to guarantee that every pel of the current VOP has a value to be predicted from, all of the boundary blocks and some of the transparent blocks of the reference VOP have to be padded using motion compensation (MC) padding. Boundary blocks are padded using repetitive padding: First boundary pels are replicated in horizontal direction, then in vertical direction making sure that if a value can be assigned to a pel by both padding directions an average value is assigned to the pel. Since this repetitive padding puts a significant computational burden on the decoder, a simpler mean padding is used in a second step. Transparent macroblocks bordering boundary blocks are assigned to an average value determined by the pels of its neighboring padded blocks.

In order to encode the texture of a boundary block, this part of ISO/IEC 14496 treats the macroblock as a regular macroblock and encodes each block using an 8*8 DCT. The decoder decodes the texture and discards all information that falls outside of the decoded shape. In order to increase coding efficiency, the encoder can choose the texture of pels outside of the object such that the bitrate is minimized. This non-normative process is called texture padding. For intra mode, a lowpass extrapolation filter increasing the PSNR of boundary blocks by an average of 2dB was developed [4]. For inter mode setting these pels to 0 gives good efficiency.

VOP texture may be encoded using B-frames. The related shape information of the B-frame is coded using the inter mode as described in below using the closest P or I frame as a reference VOP.

M.4 Encoder Architecture

Figure M-4a shows the block diagram of this object-based video coder. This diagram focuses on the object-based mode in order to allow a better understanding of how shape coding influences the encoder and decoder. Image analysis creates the bounding rectangle for the current VOP S_k and estimates texture and shape motion of the current VOP S_k with respect to the reference VOP S'_{k-1}. Parameter coding encodes the parameters predictively. The parameters get transmitted, decoded and the new reference VOP is stored in the VOP memory and also handed to the compositor of the decoder for display. The increased complexity due to the coding of arbitrarily shaped video objects becomes evident in Figure M-4b that shows a detailed view of the parameter coding.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
Single user license only. Copying and networking prohibited.

Since this part of ISO/IEC 14496 defines motion vectors for motion compensation of texture and for motion compensation of shape, we refer to them as texture motion vectors and shape motion vectors, respectively. The parameter coder encodes first the shape of each boundary block using shape motion vectors and texture motion vectors for prediction. Lossy shape coding may change a boundary block to a transparent block. If the boundary block is not transparent, its shape motion vector is coded. The shape motion coder knows which motion vectors to code by analyzing the possibly lossily encoded shape parameters.

For texture prediction, the reference VOP is padded as described above. The prediction error is then padded using the original shape parameters to determine the area to be padded. Using the origina shape as a reference for padding is again an encoder choice. Using the original or the coded shape for defining the texture of a macroblock makes obviously only a difference if the shape is coded lossily. Assuming that the texture between object and background in the original VOP differs significantly, using the original shape to define the area to be padded in the macroblock prevents the background texture from influencing the padding process. Texture padding will use the object texture for padding resulting in an efficiently to code block. If on the other hand the coded shape is used to define the area to be padded, background texture may affect the padding result potentially increasing the bitrate for texture coding drastically. Finally, the texture of each macroblock is encoded using DCT.

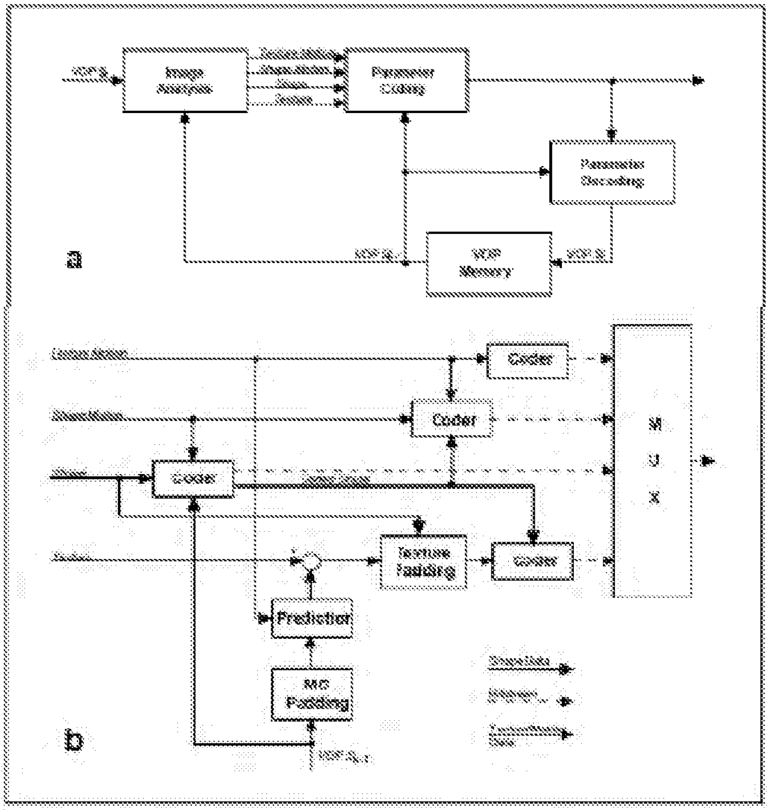


Figure M-4 – Block diagram of the video encoder (a) and the parameter coder (b) for coding of arbitrarily shaped video objects.

M.5 Encoding Guidelines

When encoding arbitrarily shaped VOs several new encoding options become available when compared to coding of rectangular video sequences. Here we provide some hints on lossy shape coding, shape coding mode selection.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON
ANSI Store order
Single user licence only. Copying and networking prohibited.

M.5.1 Lossy Shape Coding

The decoder has no possibility to find out whether the encoder uses lossless or lossy shape coding, what shape coding strategy the encoder uses or texture padding algorithm is used. Therefore, post-processing filter for the binary shape information may not be used. In its reference implementation of a video coder, this part of ISO/IEC 14496 chose to control lossy shape coding by using an alpha-threshold. The alpha threshold defines the maximum number of incorrectly coded pels within a boundary block. A fixed or adaptive alpha-threshold can be used for each bab. Using an alpha threshold may change the topology of the shape. Often, isolated pels at the object boundary are coded as part of the object. Using morphological filters at the encoder to smooth object boundaries or to filter the prediction error with respect to the original shape provides a much more predictable quality of a lossily encoded shape. Furthermore, the encoder can choose to neglect the prediction error signal at the object boundary if it does not change the topology of the object and to update the prediction error in those parts of the block where the prediction error leads to a change of the object topology.

M.5.2 Coding Mode Selection

As far as shape coding is concerned, this part of ISO/IEC 14496 allows for 18 coding modes of each BAB: (Intra/Inter/Inter with MC)*(horizontal/vertical scanning)*(Subsampling factor 0/1/2). The influence of different shape coding modes on the performance of the coder in terms of coding efficiency but also in terms of computational complexity is of interest, since the shape coder including padding can account for up to 50% of the encoder operations [3]. When encoding a binary shape, checking every mode achieves the highest compression. However, the gain achieved by this checking should be evaluated in relation to the overall bitrate required for coding a VOP. Table M-1 shows the main usage for each coding mode.

Table M-1 – Shape coding modes and their main usage

	Mode	**Main Usage**
1	intra	I frames, arbitrarily shaped still texture object, error resilience
2	inter, inter MC	P frames
3	horizontal/vertical scanning	Low-bitrate shape coding
4	subsampling to a block size 8x8 or 4x4	Low-bitrate lossy shape coding

The intra mode is used for I frames and for coding the shape of the arbitrarily shaped still texture object. Similarly, the use of the intra mode as well as the inter mode with and without motion compensation should be investigated for coding of P frames. For low-bitrate applications, the shape usually requires a significant percentage of the overall bitrate. Then it makes sense for the encoder to investigate the gain of adapting the scanning direction of each BAB reducing the shape bit rate by up to 8%. In case of lossy shape coding at low bit rates, the subsampling filter reduces the bit rate for shape coding by up to 20% for an alpha threshold of 16. The use of the subsampling filter is most effective for I babs. In case of lossless shape coding, the coder should signal at the beginning of each VOP that the subsampling feature would not be used. This avoids the overhead of signaling the subsampling for each BAB.

M.6 Conclusions

This part of ISO/IEC 14496 standardizes an object-based decoder. Several choices regarding texture extrapolation at object boundaries and shape encoding are available to the encoder. It was found that for texture extrapolation a simple filter setting the texture outside of the *original* object shape to gray is sufficient. In the case of lossy shape coding, not using the *original* object shape for defining the support region of the texture extrapolation may increase the overall bitrate by more than 100%. Therefore, lossy shape coding makes only sense if the encoder architecture presented in this paper is used.

Fixing the bab scanning to horizontal instead of adaptively switching between horizontal and vertical scanning increases the overall bitrate by just 1% but saves significant compute time since the compute intensive arithmetic coding with context determination has to be used only half as often.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
Single user license only. Copying and networking prohibited.

The downsampling of the shape information increases the computational complexity of he encoder. In the case of lossy shape coding in intra mode, the subsampling increases the subjective quality of the coded object shape due to a non-linear shape upsampling filter. For low bitrate video coding using lossily codec shapes, this filter reduces the overall bitrate.

As the experiments show, the ISO/IEC 14496-2 binary shape coder can operate in a very bit efficient manner. In order to save computational complexity for high bit rate applications, the encoder may choose to not explore all of the encoding options and choose a less efficient shape encoding. The increase of the overall bitrate may be less than 1% at high bit rates but the computational complexity of the encoder decreases by 15% to 50%.

M.7 References

[1] N. Brady, F. Bossen, N. Murphy, "Context-based arithmetic encoding of 2D shape sequences", Special session on shape coding, ICIP 97, Santa Barbara, 1997.

[2] A. Katsaggelos, L. Kondi, F. Meier, J. Ostermann, G. Schuster, "MPEG-4 and rate-distortion based shape coding techniques," Proceedings of the IEEE, pp. 1126-1154, June 1998.

[3] J. Ostermann, "Coding of arbitrarily shaped video objects with binary and greyscale alpha maps: What can MPEG-4 do for you?" ISCAS 98, Monterey, CA, TPA 2-1, June 1998.

[4] A. Kaup, "Adaptive low-pass extrapolation for object-based texture coding of moving video", Proc. Visual Communications and Image Processing, SPIE vol. 3024, pp. 731-741, Feb. 1997.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA C ADHIKARI
ANSI Store order #X274451 Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

Annex N
(normative)

Visual profiles@levels

The table that describes the visual profiles and levels is given below, with the following notes:

1 Enhancement layers are not counted as separate objects.

2 The maximum VMV buffer size is the bound on the memory (in macroblock units) which can be used by the VMV algorithm. This algorithm (see subclause D.5) models the pixel memory needed by the entire visual decoding process. This includes memory needed for reference VOPs in the prediction of P, B, and S(GMC)-VOPs and the storage of the reconstructed VOPs until such time that they are released by the decoder, plus the memory required to queue B-VOPs until composition occurs. In profiles that contain more than one layer, the memory requirements include all base and enhancement layers. When belonging to different, overlapping objects, some of these macroblocks may overlay on the display, however separate memory is required (prior to composition) in the VMV.

3 The conformance point for the base layer of Simple Scalable Visual Profile is the Simple Profile @L1 when Simple Scalable Profile @L1 is used and the Simple Profile @ L2 when Simple Scalable Profile @ L2 is used.

4 The VCV decoder rate is the vcv_decoder_rate (H) referred to in subclause D.4; it is the number of macroblocks/second based on the typical spatial and temporal resolutions, as follows:

- 1485 MBs/s corresponds to QCIF at 15Hz.
- 5940 MBs/s corresponds to CIF at 15 Hz and also twice QCIF at 30 Hz.
- 11880 MB/s corresponds to CIF at 30 Hz.
- 7425 MB/s corresponds to 1.25 times CIF at 15 Hz.
- 23760 MB/s corresponds to twice CIF at 30 Hz.
- 97200 MB/s corresponds to twice CCIR 601 at 30 Hz.
- 489600 MB/s corresponds to twice 1920x1088 at 30 Hz.

5 The total (aggregated) vbv_buffer_size is the sum of the individual VBV buffer occupancies at any given time (in units of 16384 bits) for all VOLs of all VOs. This total VBV size is limited according to the profile and level.

6 The maximum video packet length is defined as the maximum number of bits of data_partitioned_motion_shape_texture() in one video packet. The constraint applies only when the data-partitioning tool is enabled in the bitstream. When data partitioning is disabled, there is no limit on the size of video packet length.

7 N. A. means Not Applicable.

8 The maximum VCV buffer size (cumulative over all layers of all VOs) is twice the maximum number of macroblocks per VOP in the profile and level combination except for the Simple Visual Profile, Simple Scalable profile (Level 1) and Advanced Real Time Simple Profile. For the Simple Visual Profile and the Advanced Real Time Simple Profile, this value is maximum number of macroblocks per VOP. For the Simple Scalable profile (Level 1), it is 1.25 times of the maximum number of macroblocks per VOP. The limit applies to both VCV buffer and the boundary MB VCV buffer.

9 The VCV boundary MB decoder rate column bounds the number of macroblocks containing non trivial shape information. The VCV boundary MB decoder rate constrains the total number of boundary MBs in all VOLs concurrently. Note that the boundary macroblocks are added to both the VCV and boundary VCV buffers.

Table N-1 -- Definition of Version 1 Natural Visual Profiles@Levels

Visual Profile	Level	Typical Visual Session Size	Max objects [1]	Maximum number per type	Max unique Quant Tables	Max. VMV buffer size (MB units)[2]	Max VCV buffer size (MB)[3]	VCV decoder rate (MB/s)[4]	VCV Boundary MB decoder rate (MB/s)[5]	Max total VBV buffer size (units of 16384 bits)[6]	Max vbv buffer size (units of 16384 bits)	Max. video packet length (bits)[7]	Max sprite size (MB units)	Wavelet restrictions	Max bitrate (kbit/s)	Max. enhancement layers per object
N-Bit	L2	CIF	16	16 x Core or Simple or N-Bit	4	2376	792	23760	11880	80	80	8192	N. A.	N. A.	2000	1
Main	L4	1920 x 1088	32	32 x Main or Core or Simple	4	65344	16320	489600	244800	760	760	16384	65280	Scalable Texture Profile@L2	38400	1
Main	L3	CCIR 601	32	32 x Main or Core or Simple	4	11304	3240	97200	48600	320	320	16384	6480	Scalable Texture Profile@L1	15000	1
Main	L2	CIF	16	16 x Main or Core or Simple	4	3960	1188	23760	11880	80	80	8192	1584	Scalable Texture Profile@L1	2000	1
Core	L2	CIF	16	16 x Core or Simple	4	2376	792	23760	11880	80	80	8192	N. A.	N. A.	2000	1
Core	L1	QCIF	4	4 x Core or Simple	4	594	198	5940	2970	16	16	4096	N. A.	N. A.	384	1
Simple Scalable[8]	L2	CIF	4	4 x Simple or Simple Scalable	1	3168	792	23760	N.A.	40	40	4096	N. A.	N. A.	256	1 spatial or temporal enhancement layer
Simple Scalable	L1	CIF	4	4 x Simple or Simple Scalable	1	1782	495	7425	N.A.	40	40	2048	N. A.	N. A.	128	1 spatial or temporal enhancement layer
Simple	L3	CIF	4	4 x Simple	1	792	396	11880	N.A.	40	40	8192	N. A.	N. A.	384	N. A.
Simple	L2	CIF	4	4 x Simple	1	792	396	5940	N.A.	40	40	4096	N. A.	N. A.	128	N. A.
Simple	L1	QCIF	4	4 x Simple	1	198	99	1485	N.A.	10	10	2048	N. A.	N. A.	64	N. A.

Table V2 - 44 -- Definition of Version 2 Natural Visual Profiles@Levels

Visual Profile	Level	Typical Visual Session Size	Max objects[1]	Maximum number per type	Max unique Quant Tables	Max. VMV buffer size (MB units)[2]	Max VCV buffer size (MB)[3]	VCV decoder rate (MB/s)[4]	VCV Bound-ary MB decoder rate (MB/s)[5]	Max total VBV buffer size (units of 16384 bits)[6]	Max vbv buffer size (units of 16384 bits)	Max. video packet length (bits)[7]	Max sprite size (MB units)	Wavelet restrictions	Max bitrate (kbit/s)	Max. enhance-ment layers per object
Advanced Coding Efficiency	L4	1920 x 1088	32	32 x Adv. Coding Efficiency or Core or Simple	4	48960	16320	489600	244800	760	760	16384	N.A.	N. A.	38400	1
Advanced Coding Efficiency	L3	CCIR 601	32	32 x Adv. Coding Efficiency or Core or Simple	4	9720	3240	97200	48600	320	320	16384	N.A.	N. A.	15000	1
Advanced Coding Efficiency	L2	CIF	16	16 x Adv. Coding Efficiency or Core or Simple	4	2376	1188	23760	11880	80	80	8192	N.A.	N. A.	2000	1
Advanced Coding Efficiency	L1	CIF	4	4 x Adv. Coding Efficiency or Core or Simple	4	1188	792	11880	5940	40	40	8192	N.A.	N. A.	384	1
Core Scalable	L3	CCIR 601	16	16 x Core or Simple or Core scalable or Simple Scalable	4	12906	4032	120960	60480	80	80	16384	N.A.	N. A	4000	2
Core Scalable	L2	CIF	8	8 x Core or Simple or Core scalable or Simple	4	2870	990	29700	14850	80	80	4096	N.A.	N.A.	1500	1
Core Scalable	L1	CIF	4	4 x Core or Simple or Core scalable or Simple Scalable	4	2376	792	14850	7425	64	64	4096	N.A.	N.A.	768	1
Advanced Core	L2	CIF	16	16 x Core or Simple or Adv scalable Texture	4	2376	792	23760	11880	80	40	8192	N.A.	see Table V2 - 41	2000	1
Advanced Core	L1	QCIF	4	4 x Core or Simple or Adv Scalable Texture	4	594	198	5940	2970	16	8	4096	N.A.	see Table V2 - 41	384	1

Visual Profile	Level	Typical Visual Session Size	Max ob-jects [1]	Maximum number per type	Max unique Quant Tables	Max. VMV buffer size (MB units) [2]	Max VCV buffer size (MB) [3]	VCV decoder rate (MB/s) [4]	VCV Bound-ary MB decoder rate (MB/s) [5]	Max total VBV buffer size (units of 16384 bits) [6]	Max vbv buffer size (units of 16384 bits)	Max. video packet length (bits) [7]	Max sprite size (MB units)	Wavelet restrictions	Max bitrate (kbit/s)	Max. enhance-ment layers per object
Advanced Real Time Simple	L4	CIF	16	16 x Simple or Adv. Real Time Simple	1	792	396	11880	N.A.	80	80	16384	N. A.	N. A.	2000	N. A.
Advanced Real Time Simple	L3	CIF	4	4 x Simple or Adv. Real Time Simple	1	792	396	11880	N.A.	40	40	16384	N. A.	N. A.	384	N. A.
Advanced Real Time Simple	L2	CIF	4	4 x Simple or Adv. Real Time Simple	1	792	396	5940	N.A.	40	40	16384	N. A.	N. A.	128	N. A.
Advanced Real Time Simple	L1	QCIF	4	4 x Simple or Adv. Real Time Simple	1	198	99	1485	N.A.	10	10	8192	N. A.	N. A.	64	N. A.

Annex O
(informative)

3D Mesh Coding

O.1 Introduction

Representation of a virtual world (including 3-D models) is not something new in this part of ISO/IEC 14496. For example, ISO/IEC 14772-1:1997 describes an efficient way to represent virtual world. It, however, remains challenging if entities of the virtual world such as 3-D models are to be transmitted or stored due to the large size of data. 3D Mesh Coding in this part of ISO/IEC 14496 is a way to efficiently compress a 3-D model with enhanced network capabilities: computational graceful degradation control, incremental representation, support for non-manifold model, error resilience, and quality scalability via hierarchical transmission of levels of details.

The Topological Surgery (TS) representation is the key element in coding of a 3-D mesh. A 3-D mesh can be thought of as IndexedFaceSet in ISO/IEC 14772-1:1997. And it consists of connectivity, geometry, and properties (e.g. colour, normal, texture coordinate) information. The connectivity information is encoded losslessly, whereas the other information could be quantized before compression. In order to maintain the congruence of the system, geometry and properties information are encoded in a similar fashion. The output of the decoder can be directly used as an IndexedFaceSet node represented in ISO/IEC 14496-1:1999.

O.2 Topological Surgery Representation

The connectivity of a 3D mesh is the relationship among the vertices, edges, and faces of the 3D mesh. In the Topological Surgery representation (TS) the connectivity of a connected component of a manifold 3D mesh is decomposed into a *Simple Polygon* and a *Vertex Graph*.

A non-manifold or non-orientable 3D mesh is first converted to an oriented manifold 3D mesh by cutting through some vertices and edges. One such method is described in reference [10]. The information necessary to reconstruct the original connectivity from the resulting orientable manifold connectivity can be encoded using *Stitches*.

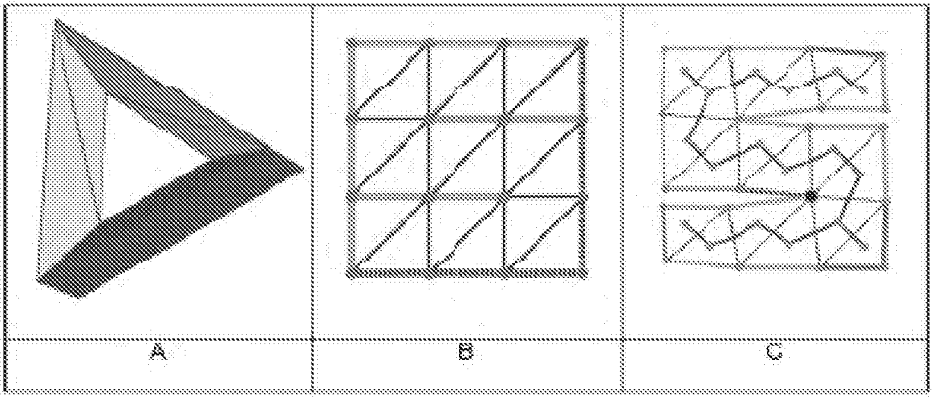


Figure V2 - 83 – Example of Topological Surgery on a torus.
The torus (A) can be obtained from the quadrilateral (B) by identifying boundary vertices. The simple polygon is obtained by constructing a spanning tree in the dual graph of the mesh (C). The boundary of the simple polygon is the polygon loop. The vertex spanning tree is constructed as a spanning tree in the vertex spanning graph (B).

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAVANE
ANSI Store order X027445 Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

The first step of the encoding process, which generates this representation, is to construct a *face forest* spanning the dual graph of the 3D mesh. Two sets of the edges of the 3D mesh can be determined by the face forest: the *face forest edges* and the *vertex graph edges*. The vertex graph edges define the vertex graph spanning the graph of the 3D mesh. All the faces of each connected component of the 3D mesh are connected by face forest edges, resulting in one-to-one correspondence between connected components and trees of the face forest. The simple polygons are defined as the connected components of the 3D mesh resulting from cutting the original 3D mesh through the vertex graph edges. The decoder uses the vertex gr aph to identify how to stitch the 3D mesh back together. Figure V2 - 83 shows an example of this decomposition.

The simple polygons are divided into triangle tree and triangle data pairs. This enables the decoder to incrementally render the model as data for each triangle is received from triangle data bitstream.

A non-manifold 3D mesh is represented as a manifold 3D mesh and a vertex clustering array. The vertex clustering array is represented as a series of stitches operations, with one command per vertex. The encoding process encodes the stitches after the vertex graph is encoded using the vertex clustering array, and optionally using a v_father array. The v_father array provides for each vertex in the vertex graph the index of its predecessor (father) in the directed vertex graph without jumps (potentially the vertex itself if it is a root of the graph without jumps). A pair (vertex, predecessor) is an edge of the 3D mesh.

In the actual representation to be described below, we force the face tree to be binary. We do this, by triangulating polygonal faces, effectively, adding additional edges to the mesh. Since this operation adds extra bits, we avoid unnecessary overheads in the vertex graph, by computing t previously to the triangulation.

O.2.1 Simple Polygon Representation

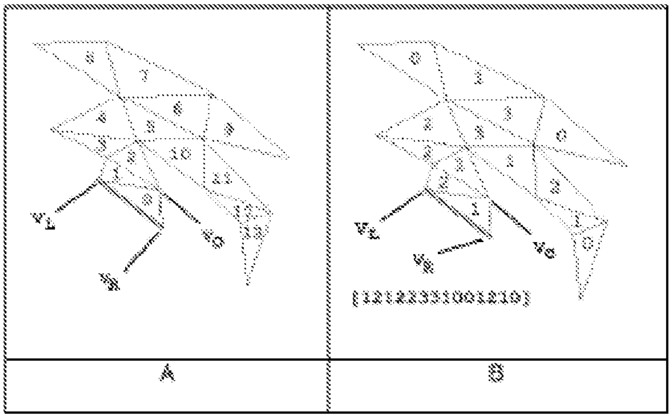


Figure V2 - 84 – Constant-length encoding of a simple polygon. A: Triangles labels according to their order of traversal. B: Triangles labels according to their two bit code. The encoding of the polygon is the sequence between the brackets.

The simple polygon can be represented as a string of two-bit codes by recursively traversing the triangle tree as illustrated in Figure V2 - 84. The traversal starts by entering a first triangle called *root triangle* through a polygon loop edge called the *root edge* (green edge in Figure V2 - 84). One of the two endpoints of the root edge is called the *root vertex* of the simple polygon. The traversal algorithm uses three vertex pointers, the *left vertex* vL, the *right vertex* vR, and the *opposite vertex* vO. The right vertex is initialized to the root vertex, the left vertex to the other endpoint of the root edge, and the opposite vertex to the third vertex of the root triangle. The edge eL=(vL,vO) is called *left edge*, and the edge eR=(vR,vO) is called *right edge*. Each of the two edges is either a boundary edge of the polygon, or internal to the polygon. One bit of the encoding is used to indicate which case, with the bit associated with the left edge first followed by the bit associated with the right edge, and with values 1 for internal, and 0 for boundary. If only the left edge is internal, the traversal algorithm sets vR=vO and continues with the other triangle incident to eL. If only the right edge is internal, the traversal algorithm sets vL=vO and continues with the other triangle incident to eR. If both edges are internal, the traversal algorithm pushes vO and vR onto a traversal stack, it sets vR=vO and continues with the other triange incident to eL. If both edges are boundary, and the

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAVANE
Single user license only. Copying and networking prohibited.

traversal stack is not empty, the traversal algorithm pops vR and vL from the traversalstack, and continues with the other triangle incident to the edge (vL,vR). If both edges are boundary, and the traversal stack is empty, all the triangles of he simple polygon have been visited. For example, in Figure V2 - 84-A, the triangles are labeled with their order of traversal, and in Figure V2 - 84-B, with their corresponding two-bit cose, as a number in the range 0,...,3. The encoding of this polygon is [1212233100121].

The simple polygon can also be represented as a *table of triangle runs* in the same way as the vertex tree, except that each triangle run record is composed of only two fields *(tt_run_length, tt_leaf)*. A *Tlast* field is not necessary because the triangle tree is a binary tree, and each triangle run which ends in a branching node must be followed by exactly two runs. The structure of the triangle tree does not completely describe he triangulaton of the simple polygon, though. To complete the description, an extra bit per triangle associated with a regular node of the triangle tree must be included. This *sequence of marching patterns* determines how to triangulate the triangle runs by advancing either on the left or on the right on the boundary of the simple polygon.

Note that there is a very simple transformation from the previous representation of the simple polygon to this one. The 3s and 0s in the sequence in the previous representation mark the end of the triangle runs. In the example of Figure V2 - 84, the triangle-runs are defined by the subsequences [121223], [3], [10], [0], and [1210], which are in the same order defined by the table of triangle runs. The tt_run_length field value is the number of two-bit codes in the corresponding subsequence (6,1,2,1, and 4 in this case). The tt_leaf field value is determined by the last code in the subsequence (3 → 0, 0 → 1), and the marching patterns by the other codes in the sequence, skipping the 3s and 0s (1 → 0, 2 → 1). In this example the marching edge is the following sequence of bits [010110010]. Clearly, the opposite transformation is equally simple.

O.2.2 Vertex Graph representation

For a connected manifold without boundary, each edge of the vertex graph corresponds to two polygon loop edges (i.e., boundary edges of the simple polygon), but for manifolds with boundary, some edges of the vertex graph (those which correspond to boundary edges of the mesh) correspond to only one polygon loop edge. In the case of a *simple mesh* (manifold without boundary with Euler number 2, i.e., topologically equivalent to a sphere), the vertex graph is a tree. When the manifold mesh is not simple (i.e, when the mesh has boundary edges, or when the Euler number is different from 2), the vertex graph contains loops. If the vertex tree is constructed as a spanning tree by depth first traversing the vertex graph, then one back-edge, or *jump* edge, connects a leaf node of the vertex tree with a previously visited node, forming a loop.

The vertex graph is represented as data structure VG composed of sequence of *runs*. Even though each jump corresponds to a single edge in the vertex graph, we represent a jump by two runs in the VG, one of them having zero length. The first, and true, run starts at the first node the edge gets visited. The second run is generated when we reach the previously visited node, which, then, needs to be considered as a branching node, and the run generated has zero length. Each run record of VG is composed of a variable number of fields. A vertex run connects a leaf or branching node to another leaf or branching node through a path of zero or more regular nodes. The order of traversal of the tree defines a sequential order for the vertex runs, and a first and last node for each run. Each vertex run is represented as a vertex run record composed of the following fields. The vg_last field is a bit that determines if the run shares the first node with the next run or not. The vg_run_length field is an integer with a value equal to the number of edges in the run, not counting the jump edges. The vg_leaf field is a bit which determines if the run ends in a leaf or branching node. Each extended vertex run ending in a leaf node (vg_leaf=1) has an additional vg_loop field, indicating if the last node of the vertex run is connected to another node through a jump edge. Note that each extended vertex runs of length zero, which is always associated with a jump edge, should be considered as ending in a leaf. For these extended vertex runs, first and last node are the same. If vg_loop=1, the record also has a vg_forward_run field indicating whether this run corresponds to the start or end of the run. Each jump edge connects the last node of a run to a previously visited node. The node visited first is the *start* of the jump (vg_forward_run =1), and the *other one* is the *end* of the jump (vg_forward_run =0). If vg_forward_run =0, the record also has a vg_loop_index field. Every time the start of a jump is encountered, the compression and decompression algorithms push the reference to the jump edge onto an auxiliary *jump stack*. The value of vg_loop_index is the depth of the corresponding jump edge in the jump stack at the time the compression or decompression programs encounter the end of the jump. The reference is immediately removed from the stack, moving subsequent references up, and reducing the total depth of the stack by one (and so reducing the number of bits required to represent the depth of subsequent jump edges). Figure V2-85 illustrates some of these concepts.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
Single user license only. Copying and networking prohibited.

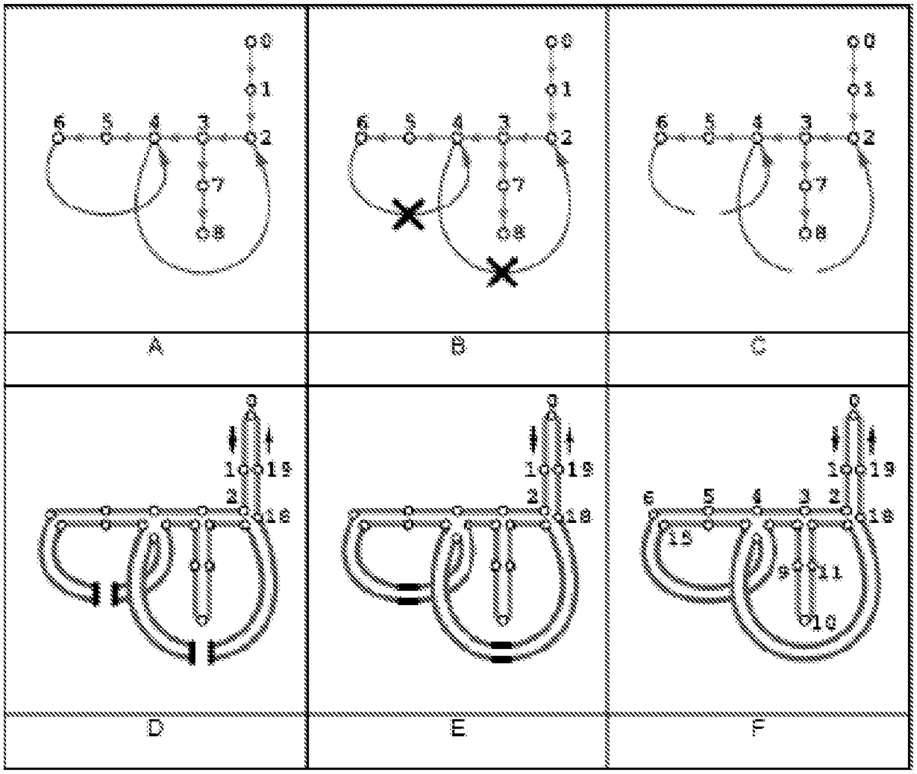


Figure V2 - 85 -- Steps to build the bounding loop from the vertex graph.
A : vertex graph decomposed into vertex tree (green) and jump edges (red).
B : extended vertex tree is created by cutting jump edges in half.
C : the extended vertex tree has two leaves for each jump edge.
D : build the extended vertex tree loop.
E : connect the start and end of each jump edge.
F : resulting boundary loop.

O.3 Encoding guidelines for 3D Mesh Coding

O.3.1 Topological Surgery Encoding

A manifold 3D mesh is represented in compressed form as the concatenation of its connected components represented in compressed form. The algorithm to compress a connected component of the connectivity of a manifold 3D mesh can be decomposed into the following steps:

1. Construct the face tree as a spanning tree in the dual graph of the 3D mesh.

2. Define the vertex graph as the graph resulting from deleting the face tree edges from the graph of the mesh.

3. Construct the vertex tree as a spanning tree in the vertex graph.

4. Label each edge of the 3D mesh as a polygon tree edge, a vertex tree edge, or a jump edge.

5. Triangulate the polygonal faces by adding internal non-polygonal edges to the faces.

6. The simple polygon is the mesh resulting from cutting the triangular mesh through the edges of the vertex graph. The triangle tree is the dual graph of the simple polygon.

7. Label each triangle tree edge as a polygon tree edge or non-polygon tree edge.

8. Choose an edge of the vertex tree and call it root edge of the vertex tree.

9. Choose an endpoint of the root edge of the vertex tree and call it root vertex of the vertex tree.

10. Of the faces incident to the root edge of the vertex tree, choose an incident face and call it root face of the face tree.

11. Call the edge of the root face of the face tree corresponding to the root edge of the vertex tree, root edge of the face tree.

12. Call the vertex of the root face of the face tree corresponding to the root vertex of the vertex tree, root vertex of the face tree.

13. Encode the simple polygon by depth first traversal starting at the root edge of the simple polygon, with the root vertex of the simple polygon as right vertex, and the other endpoint of the root edge as left vertex. Upon entering a branching triangle, traverse the left branch before the right branch.

14. Encode the vertex graph as a table of extended vertex runs by depth first traversing the extended vertex tree according to the orientation of the simple polygon induced by the order of traversal of the simple polygon.

This encoding procedure is described in further detail in [17].

O.3.2 Support for non-manifolds and Non-orientable manifolds

A non-manifold 3D mesh is represented as an orientable manifold mesh and a vertex clustering array. A non-orientable manifold 3D mesh is considered in the same way as a non-manifold with the inconsistent edges as singular. A first step in the encoding process consists of converting a non-manifold mesh to a manifold mesh and a vertex clustering array. There are several possible methods for performing this operation, one of which being described by Gueziec *et al.* in reference [10].

The Stitches Encoder operates in a manner symmetric to the Stitches Decoder. Its operation is summarized in the pseudo-code of Figure V2 - 86 below.

```
encode(anchor_stack) {
 if (has_stitches == true) {
  encode has_stitches;
  for (i = nV0; i < nV1; i++) {
  // nV0 is the first vertex of the current component, and nV1 -1 is the last vertex
   encode stitch_cmd;
    if (stitch_cmd) {
     encode stitch_pop_or_get;
     if (stitch_pop_or_get) {
       encode stitch_pop;
```

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAWANE
ANSI Store order X037445 Downloaded: 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

```
        encode stitch_stack_index;
        retrieve stitching_anchor from anchor_stack;
        if (stitch_pop) {
          remove stitching_anchor from anchor_stack;
        } // end if
        encode stitch_incr_length;
        if (stitch_incr_length != 0) {
          encode stitch_incr_length_sign;
        } // end if
        decode stitch_push;
        if (stitch_push)
          push i to the back of anchor_stack
        retrieve stitch_length at stitching_anchor;
        total_length = stitch_length + stitch_incr_length;
        if (total_length < 0)
          encode stitch_reverse;
        save total_length at stitching_anchor;
      } // end if (stitch_pop_or_get)
      encode stitch_length;
      push i to the back of anchor_stack;
    } // end if (stitch_cmd)
  } // end for
}
```

Figure V2 - 80 – Pseudo-code for the Stitches encoder

If the boolean **has_stitches** flag in the current component is true, then for each vertex of the Vertex Graph in decoder order, the Stitches encoder encodes a stitching command from the bitsteam. The boolean value **stitch_cmd** is encoded. If the boolean value **stitch_cmd** is true, then the boolean value **stitch_pop_or_get** is encoded; if the boolean value **stitch_pop_or_get** is false, an unsigned integer **stitch_length** is encoded. The current vertex is then pushed to the back of the anchor_stack.

If **stitch_pop_or_get** is true, then the boolean value **stitch_pop** is encoded, followed with the unsigned integer value **stitch_stack_index**. Using **stitch_stack_index**, a stitching_anchor is retrieved from the anchor_stack. This is the anchor that the current vertex i will be stitched to. If the **stitch_pop** boolean variable is true, then the stitching_anchor is removed from the anchor_stack. Then an integer **stitch_incr_length** is encoded as an unsigned integer. If it is different from zero, its sign (boolean **stitch_incr_length_sign**) is encoded. A **stitch_push** boolean value is encoded. If **stitch_push** is true, the current vertex i is pushed to the back of the anchor_stack. An integer **stitch_length** associated with the stitching_anchor is retrieved. A total_length is computed by adding **stitch_length** and **stitch_incr_length**; if total_length is greater than zero, then a **stitch_reverse** boolean value is encoded. Finally, the value total length is saved at (associated with) the stitching_anchor.

O.3.3 Support for Error Resilience

To support error resilience, data partitioning is very important. Component based data partitioning (CODAP) provides a new functionality to the TS method. CODAP provides a way to implicitly divide a large 3D mesh into multiple partitions. A partition may include part of a connected component or multiple connected components. Since vertex graph is relatively small compared to simple polygon, the division of connected component is only applied to simple polygon. However, the correspondence between vertex graph (vg) and simple polygon is kept through the whole coding process. There are three different partition types, which reflect different combinations of vertex graphs and simple polygons. In front of each partition, a start code is provided for resynchronization.

In order to make a partition into a given size, traversal order determination and partitioning methods are used. The traversal order information is coded with particular branching triangles. Virtual leaf triangle information is inserted at the end of each divided triangle tree to make a virtually complete triangle tree.

If channel errors hit the bit stream, some partitions will be corrupted or lost. However, other partitions will be decoded with its simple polygon and corresponding vg. Forward error correction may be used to ensure the vg is protected.

O.3.3.1 Traversal Order Determination

In TS, the reconstruction procedure is as follows: 1) building the bounding loop table, 2) computing the boundary offset for each branch, and 3) reconstructing triangles and polygons.

In basic TS coding, the traversal direction is always clockwise or left most search. In the case that the left branch is much larger than the right branch of the triangle tree, the decoder has to wait until it decodes the vertices all the way down the left branch before decoding and rendering the smaller right branch. See Figure V2 - 87. However in the CODAP technique, in order to avoid unnecessary delay, the encoder will choose the shorter (left or right) branch. A one-bit flag will be sufficient to inform the decoder of the traversal order of the triangle tree.

In Figure V2 - 87, the information of the traversal order is given at a branch on a main stem of the partitioned tree. The green branching triangle has the right first traversal order and the red branching triangle has the left first traversal order. Within a branch, the traversal order is always clockwise, as shown with the purple line.

me_1	me_2	…	$td\_orientation_1$	…	me_i	$td\_orientation_N$	…	me_N

The triangle data contains triangle marching edge (me) information where there is a triangle run. However, the triangle data contains the **td_orientation** fields where there is a branch in the main stem of the binary triangle tree.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/SOOJA Y. ADAMANE
ANSI Store order X#27445, Downloaded 7/18/2007 3:56:21 PM ET
Single user license only. Copying and networking prohibited.

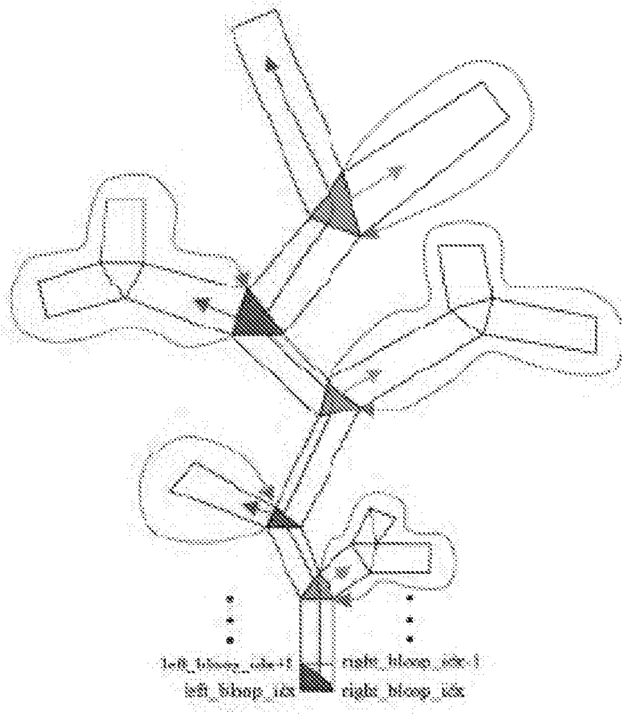


Figure V2 - 87 -- Determination of traversal order

O.3.3.2 Partitioning

One way to handle the Y-vertex while maintaining independence between partitioned simple polygon data is to make the partitioned data as a virtual connected component by inserting virtual bits, as shown in Figure V2 - 88. There are two main cases for defining the virtual bits. When the partition ends with a run or a leaf triangle, a pair of virtual bits (1,1) is added to the (tt_run_length, tt_leaf) par. When the partition ends on a branching triangle, two pairs of virtual bits (1,1) are added to the (tt_run_length, tt_leaf) pair. Also, if the partition ends on a branching triangle, the length of the branch (codap_branch_len) is coded so that Y-vertex index can be calculated.

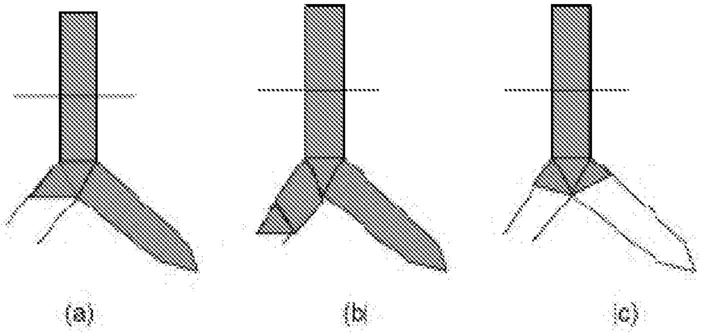


Figure V2 - 88 -- Examples of forming virtual connected component by inserting virtual bits to make virtual triangles (marked in red). The partitioning was done at : (a) a leaf (b) a run (c) a branch.

When the simple polygon data is partitioned and formed as virtual connected components, the decoder can find out if the virtual bits exist by checking the following condition:

codap_right_bloop_idx – codap_left_bloop_idx – 1 > total number of triangles decoded in tt.

If this condition is true, then virtual bits exist. To find out if there are one or two pairs of virtual triangles, the third triangle from the end is checked to see whether it is a branch or not. If it is a branch, then the partition ended on a branch and two virtual triangles are added. Otherwise, the partition ended on a leaf or a run and only one virtual triangle is added at the end. However, if the leaf is the only triangle in the subtree in Figure V2 - 88 (a), encoders shall not make a partition at this point. This is to avoid a conflict with Figure V2 - 88 (c).

Also the extended boundary prediction mode shall be used for the vertices in the partition of the subtree unless every vertex in the partition of the subtree is not used in the partitions after the subtree.

In topological surgery, to code the mesh information containing polygons, the polygons are first divided into triangles. Figure V2 - 89 shows an example for this case.

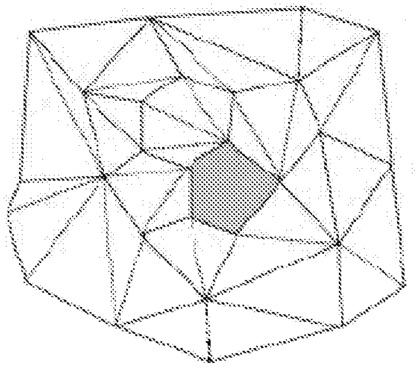


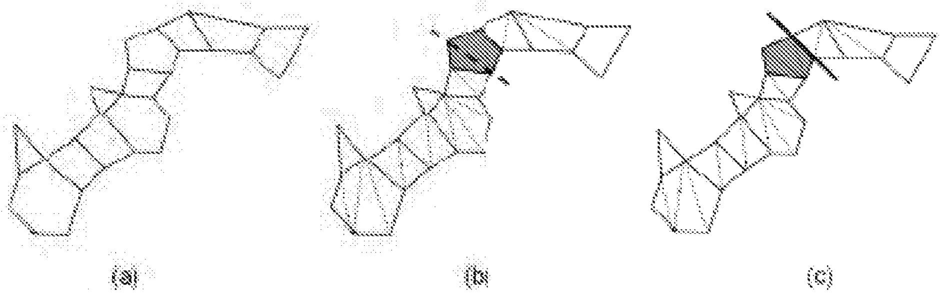
Figure V2 - 89 -- Example of dividing 3D mesh into triangle mesh : the solid lines are the original 3D mesh edge, and the dotted lines are the imaginary edges added for the division of polygon into triangles.

When polygons are divided into triangles, imaginary edges are added. Therefore, this information must be sent to the decoder to reconstruct the polygons. This information is called polygon edge. So when a polygon edge information is sent for each triangle, value 1 is sent for a real edge, and 0 is sent for an imaginary edge. However, if 3D mesh does not exist (only triangle meshes : triangulated==1), then polygon edge information is not needed.

In the topological surgery method, every polygon in the mesh is divided into triangles and polygon edge information is produced by the number of triangles. All of this information is coded, transported and stored. However, since the first triangle must start with a real edge, the corresponding polygon information will always be 1, and so it is not coded. In this case, the decoder assumes that the polygon edge information for the first edge is 1.

When data partitioning is performed, the partitioning may be done at the imaginary edge inside the polygon, as in Figure V2 - 90 (b). In this case, it cannot be assumed that the polygon edge of a data partition starts with 1. Therefore the polygon edge must be coded by the number of triangles, except for the simple polygon data in partition type 0, which is described in the next section.

Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. ADAVANE
Single user license only. Copying and networking prohibited.



tt_1	tt_2	…	tt_n	me_1	me_2	…	me_n

(d)

tt_1	tt_2	…	tt_n	me_1	me_2	…	me_n	pe_2	pe_3	…	pe_n	…

(e)

tt_1	tt_2	…	tt_n	me_1	me_2	…	me_n	pe_1	pe_2	pe_3	…	pe_n	…

(f)

Figure V2 - 90 – Example of partitioning on 3D mesh and the related syntax : tt is the triangle run information, me is the marching edge information, pe is the polygon edge information, annex at the bottom shows the order of tm and pe corresponding to tt, n is the number of triangles. (a) the original 3D mesh, (b) partitioning inside a polygon, (c) partitioning at the boundary edge(real edge) of two polygons, (d) when triangulated == 1(no 3D mesh) (e) when 3D mesh exists and partitioning is only allowed at the real edge, (f) when 3D mesh exists and partitioning is allowed at the imaginary edge.

On the other hand, if there exists a polygon with many sides (edges) but it is partitioned only at a real edge, the size of a partition may not come into the range of target size. So when the partition inside the polygon is allowed, it is possible to achieve the size that within a small range of target size. Also, allowing the partition inside the polygon makes the method more generalized.

O.3.3.3 Boundary Prediction

Once a connected component is partitioned into pieces, prediction of geometry coding (as well as other attributes that use predictive coding) should allow independent decoding between partitions.

There are two modes of coding vertices on the boundary between partitions: restricted boundary prediction mode and extended boundary prediction mode.

Confer the details of both modes in Clause 7.13.5.3

When partitioning is done at a branching triangle, it is recommended to use extended boundary prediction mode for the first branch from the branching triangle.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/DOD,R.T. ADAMSINE
ANSI Store order X037445 Downloaded: 7/18/2007 3:38:21 PM ET
Single user license only. Copying and networking prohibited.

O.4 Encoder considerations for efficient compresson of Vertex Properties

Given a geometry sample (x, y, z), the quantized geomery sampe (qx,qy,qz) is obtaired as:

xq = floor((x– **coord_xmin**)/**coord_size***((1<<**coord_quant**)-1)+0.5)

yq = floor((y– **coord_ymin**)/**coord_size***((1<<**coord_quant**)-1)+0.5)

zq = floor((z– **coord_zmin**)/**coord_size***((1<<**coord_quant**-1)+0.5)

The procedure for colors and texture coordinates is the same.

A higher-order geometrical prediction scheme is employed in coding of vertex positions, color properties, texture positions, and normal vectors. The encoder transmits **pred_type**, the information of prediction type, to the decoder, as shown in Table V2 - 45.

Table V2 - 45 -- Meaning of pred_type

pred_type	Prediction type
00	No_prediction
01	Tree_prediction
10	Parallelogram_prediction
11	reserved

For example, if the **pred_type** indicates the prediction type is tree-prediction or parallelogram, the predicted value($\tilde{p}$) can be expressed as shown in Eq. O.x.1.

$$\tilde{p} = \sum_{i=1}^{N} \lambda_i p_i \qquad \text{(Eq. O.x.1)}$$

The p_i's in Eq. O.x.1 represent the vertex properties of the adjacent vertices. The encoder transmits the each λ_i and N to the decoder.

However, the vertex property values such as colors, positions, normal vectors, and texture positions along the vertex traversal sequences has low correlation between successive vertices, it degraces the prediction accuracy. In the case of low correlation sequences, the prediction scheme can be limited to the simple 1st-order prediction. The simple 1st-order prediction scheme can be easily implemented to the encoder and decoder by fixing the λ_i's, as shown in Eq. O.x.2.

$$\lambda_i = \begin{cases} 1 & \text{, where } \ i = 1 \\ 0 & \text{, otherwise} \end{cases} \qquad \text{(Eq. O.x.2)}$$

Tabe V2 - 46 shows the weighting factors of the prediction schemes. The higher-order prediction mode is the general parallelogram predictor, while the simple 1st-order prediction mode is the modified encoding scheme. The simple 1st-order prediction mode shows better performarce for color property coding.

Tabe V2 - 46 -- weighting factors of predictor

Predicton scheme	□	$\square_i$	Coding efficiency
paralleiogram	3	1, 1, -1	-
Simple 1st-order prediction	L (> 0)	1, 0, ... , 0	better for color encoding

O.5 Progressive Forest Split Representation

In the progressive forest split (PFS) compression scheme, a 3D model is represented as a low resolution model followed by a sequence of forest split operations. A forest split operation is represented by a forest in the graph of vertices and edges of a mesh, a sequence of simple polygons, and a sequence of vertex and property displacements. The mesh is refined by cutting the mesh through the forest, splitting the resulting boundaries apart, filling each of the resulting tree boundary loops with one of the simple polygons, and finally, displacing the new vertices and properties. The representation, encoding scheme, simplification algorithm, encoding and decoding methods are described in [6].

The general structure of a simplification scheme to generate the progressive representation, and the associated level of detail hierarchy, from a single resolution 3D mesh, is described as well. This method may require additional compression parameters, which may be tuned interactively for each transition from one level of detail to the next one.

O.5.1 Encoding the Forest

A simple encoding of the forest requires one bit per edge, for example a value 1 for the edges belonging to the forest and 0 for the rest. These bits are placed in the compressed data stream in the edge index order defined above. However, since any subset of edges of a forest form a forest, the edges with bit value 1 read up to a certain point may determine that certain edges with higher edge index should have bit value 0, otherwise they would create loops in the forest defined so far by the bits with bit value 1. These edges can be skipped in the compressed data stream. Arithmetic encoding of the bit-stream results in fewer bits. In the experiments that we have run so far, where the number of triangles increase by 50-70% with each forest split operation, the skipping of predictable bits described above trims the length of the forest bit-stream by about 1-8%, and the simple encoding of the resulting bit-stream is usually shorter than the run-length encoded stream. A pseudo-code implementation of the method to determine which edges to encode is as follows:

```
/ forest encoder

// marked edges form a forest in the graph of the mesh



int nV = number of vertices of mesh

Partition p(nV)

// p defines a partition of a set of nV elements

// initially each part contains a single element

// the method find(i) retrieves the number of the part containing element i

// the method join(i,j) joins the parts containing elements i and j, respectively



for ( every edge e=(i,j) of mesh ) {

  bool skip_edge =false

  bool edge_bit  = e.isInForest()?true:false;

  if(p.find (i)!=p.find(j)) {
    if(edge_bit==true)
      p.join(i,j);
  } else {
    // forest => edge_bit==false
    skip_edge = true;
  }
  if(skip_edge==false)
    encode_bit(edge_bit);
}
```

Figure V2 - 91 -- forest encoder pseudo-code

O.5.2 Support for meshes with polygonal faces

The decoder process supports polygonal faces for the base mesh, as well for the simple polygons used to fill the tree loops.

O.5.3 Method for generating of a PFS Representation of a Triangular 3D Mesh

The method developed in this section takes as input a 3D mesh, where all polygons are triangles and generates a series of levels of detail that can be encoded as a series of PFS operations. The implementation included in the reference software takes as an input a restricted VRML single-resolution file format (described in the Core Experiments document) and returns a Clustered multi-resolution file format (extension ".cmr" as described in the Core Experiments document). Although a Clustered multi-resolution file format can represent any hierarchy of levels of detail, the output of our method is a particular hierarchy corresponding to a base mesh and a succession of PFS refinements. The method is based on applying a succession of edge collapses to the original mesh. The edge collapse operation is described in the Core Experiments document, in the section dedicated to the Progressive Vertex Split method: it consists of merging the two endpoints of an edge, and thereby eliminating two triangles from the mesh (one triangle if the edge belongs to the mesh boundary). The method of the reference implementation consists of entering all the edges of the mesh in a priority queue that is keyed with the length of the edge. Then, in succession the edges are removed from the queue, and collapsed provided that they pass a series of tests. The tests are of two types: "topological" tests and "geometric" tests. Topological tests are performed first, as they are expected to be less costly than the geometric ones. If the tests are all passed, the edge is removed from the queue, and all neighboring edges are removed and re-entered in the queue that should reflect their new lengths. Before the geometric tests are performed a decision is made on how to position the vertex resulting from the edge collapse. Depending on the user's choice, this position can be either edge endpoint or the midpoint of the edge. In the sequel, the vertex resulting from the edge collapse is referred to as "simplified vertex". Figure V2 - 92 is a diagram illustrating the algorithm.

```
while queue not empty
        take shortest edge
        if edge can be collapsed, e.g.,
                (valence does not exceed maximum &&
                mesh is still manifold after operation &&
                all collapses correspond to pfs &&
                triangle rotation is acceptable &&
                triangle compactness is within bounds)
            remove all edges in the star from the queue
            insert new edges
        else /* collapse of edge is not possible */
            simply remove edge from queue
```

Figure V2 - 92 -- Algorithm for generating PFS representation by collapsing edges of a 3D mesh

O.5.4 Topolcgical Tests

The purpose of the topological tests is to ensure that the nesh is still a manifold after collapsing a given edge and that the collecton of triangles that were removed are such that the dual graph of this colection is a forest. The first test verifies that the valence of the vertex resulting from tle edge collapse, or simplified vertex, is at east 3 if it is not a boundary vertex. The second test ensures that all tle vertices incident to the sinplified vertex are different (otherwise we would have a non-manifold mesh). The third test verifies that the collection of triangles that were removed by colapsing edges form a forest as well as a manifold mesh. Both properties are checked by mears of a comprehensive test. To keep track of the triangles that are eliminated, we introduce a color coding of the triangles as follows: blues triangles are the ones eliminated during edge collapses; red triangles are the ones remaining; purple triangles are the triangles that would be eliminated (would become blue) if the edge collapse under current testing would be performed. After several collapses are performed, an edge in the simplified surface can correspond to a pair of non-adjacent edges in the original surface (this is quite easy to see, for instance when collapsing in sequence two adjacent edges). It follows that the triangles that are removed during the collapse (purple triangles) are not necessary adjacent. Two cases are distinguished, depending on whether the purple triangles are adjacent or not. In the first case, the two purple triangles, which we refer to as T0 and T1, are adjacent. We use the notations of Figure V2 - 93. We first test whether T0 and T1 are adjacent to 3 or more blue triangles, in which case the test fails. Otherwise, if T0 and T1 are adjacent to 0 or 1 blue triangle, we consider the vertices among V0, V1, V2 and V3 (shown in Figure V2 - 93) that are not incident to a blue triangle that is adjacent to either T0 or T1. If there are incident to any blue triangle the test is failed (Figure V2 - 93B), and otherwise the test is passed. Otherwise, if T0 and T1 are adjacent to exactly two blue triangles, we first check whether the two adjacent blue triangles are incident to a common vertex, if which case the test is failed. Otherwise, the two adjacent blue triangles must belong to a different connected component of blue triangles in order to pass the test (Figure V2 - 93A). The connected components of triangles are maintained using a partition of the triangles.

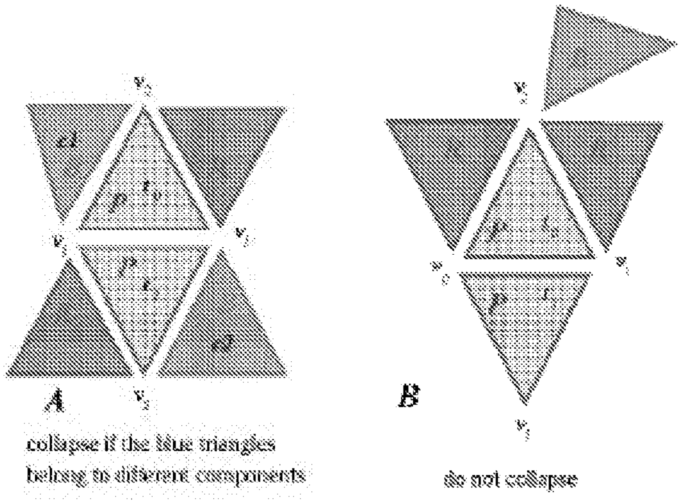


Figure V2 - 93 – Additional edge collapsibility tests necessary to obtain a PFS representation. The cases illustrated here apply when the two triangles of the original surface that are removed by the edge collapse are adjacent

The second case applies when the two purple triangles T0 and T1 are not adjacent. This case is illustrated in Figure V2 - 94. We first test whether Vertex V2 is a boundary vertex. If yes, T0 must be a boundary triangle otherwise the test fails (Case A in). If T0 is a boundary triangle, then the test is the same as when V2 is not a boundary vertex (Cases B and C in Figure V2 - 94): all triangles incident on V2 must be red. The same sequence of tests is applied to Vertex V4: if Vertex V4 is a boundary vertex, T1 must be a boundary triangle otherwise the test fails (Case A in). If T1 is a boundary triangle, then the test is the same as when V4 is an interior vertex (Cases B and C in): All triangles incident on V4 must be red.

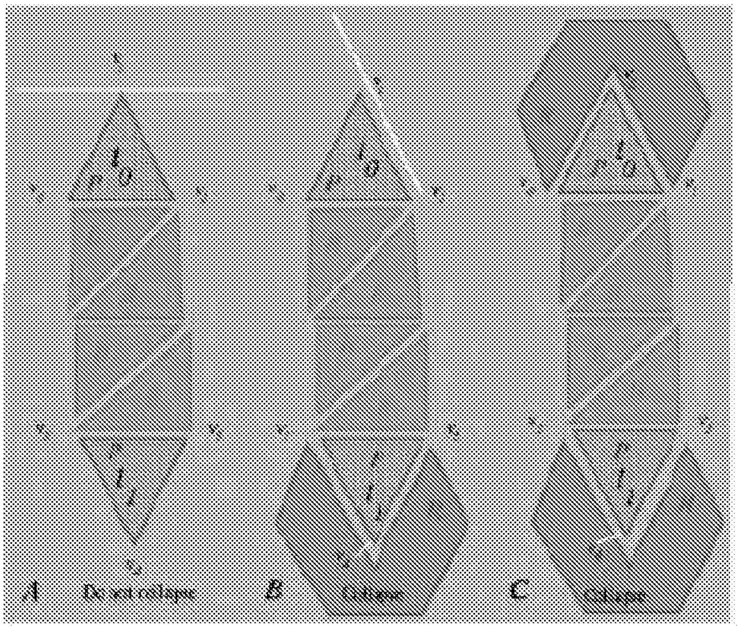


Figure V2 - 14 – Additional edge collapsibility tests necessary to obtain a PFS representation. The cases illustrated here apply when the two triangles of the orginal surface that are removed (the purple triangles T0 and T1) are not adjacent. Boldface letters inside triangles indicate colors.
A: do not collapse. B, C: collapse.

O.5.5 Geometric Tests

The geometric tests described in this section are not compulsory in order to represent the level of detail hierarchy in PFS. However they were experimentally useful to produce levels of detail that are visually satisfactory. This part of the algorithm is most open for changes and improvements by parties implementing an encoder. The first geometric test of the sample implementation verifies that the normals of the triangles incident to any vertex of the collapsed edge do not rotate by more that a user-specified angle during the collapse. The user can specify this maximum angle. The second geometric test verifies that the minimum compactness values of the triangles incident to any endpoint of the collapsed edge, before the edge collapse (C0) and after the edge collapse (C1) satisfy the following relation: C1 > r C0, where r is a user-specified ratio. The compactness of a triangle is defined using the above formula, $c = \frac{4\sqrt{3}a}{l_0^2 + l_1^2 + l_2^2}$, where a is the area of the triangle and the l_i are the three side lengths.

O.6 Complexity estimation for Computational Graceful Degradation

Computational Graceful Degradation (CGD) is the process of gracefully degrading the quality of the rendering of the 3D scene in case the 3D rendering engine has not enough processing power for performing the rendering in real-time. In this case, some lower-quality rendering processes can be performed instead, as for example flat shading instead of smooth shading, mesh-rendering with a smaller number of triangles, ... The implementation of specific CGD algorithms which provide the highest complexity reduction for the lowest image degradation is an implementation dependent issue which has no normative implications.

For being able to apply Computational Graceful Degradation, each specific 3D rendering engine should be able to estimate its required rendering processing power, before the 3D rendering is actually started. The encoder must therefore transmit, as soon as possible, some complexity estimation (ce) characteristics of the 3D objects, which are relevant to 3D rendering process. The definition of these complexity parameters is normative. These parameters are defined as follows:

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT ARLINGTON/POOJA V. AGAWANE
ANSI Store order X337445? Downloaded: 7/18/2007 3:58:21 PM ET
Single user license only. Copying and networking prohibited.

1. Number of vertices
2. Number of triangles
3. Number of edges
4. Projected surface of the object in any viewpoint

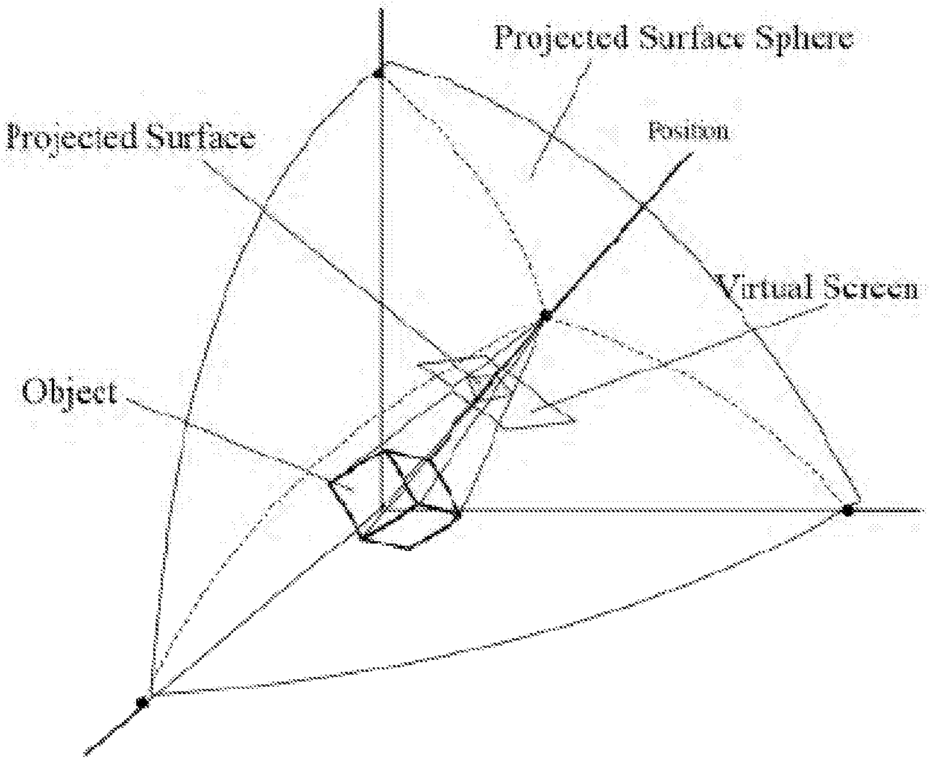


Figure V2 - 95 -- Projected Surface Sphere constructed by projecting the pixels of the object onto a virtual screen for different viewing angles

The projected surface is the number of pixels that are actually projected onto the screen (called the virtual screen) during the rendering of the object in that viewing position (see Figure V2 - 95). This projected surface largely varies with orders of magnitude, depending on the viewing angle and viewing distance to the object, which can be modified by the user, without any encoder interaction. In principle, the projected surface should be transmitted:

for any position around the object, with constant distance to a reference point of the object (e.g. gravity point), creating the so-called Projected Surface Sphere (see Figure V2 - 95).

for one particular viewing angle, in a variety of distances to the reference point.

However, simulations have shown (see [9]) that the projected surface should only be measured in a limited number of positions around the object, from which the projected surface for any other position can be recovered - within an acceptable error bound - by linear interpolation or scaling. Linear interpolation is used within a Projected Surface Sphere with constant radius, while scaling is required for extrapolating (interpoating) information from one (different) Projected Surface Sphere(s) to another Projected Surface Sphere. For better accuracy, we advise to apply the following scaling formula, using one particular Projected Surface Sphere:

Licensed to UNIVERSITY OF TEXAS AT [illegible]
Single user license only. Copying and networking prohibited.

$$p_i = \frac{R_{ref}}{R_i}\left(p_{ref} - \min_{ref}\right) + \min_i$$

in which:

p_i is the estimated projected surface in a specific direction for radius R_i.

p_{ref} is the measured projected surface in that specific direction for radius R_{ref}.

R_{ref} is the radius of the reference Projected Surface Sphere.

R_i is the radius of the estimated Projected Surface Sphere.

min_{ref} is the minimum projected surface for the reference Projected Surface Sphere.

min_i is the minimum projected surface for the estimated Projected Surface Sphere.

The accuracy can even be increased by allowing some additional measured Projected Surface Spheres (thus at different radii). Typically, only a very limited points will be added for each additional Projected Surface Sphere.

A 1-bit switch should indicate in the 3D_Mesh_Object_Header whether Computational Graceful Degradation is enabled or not. This bit is called the ce_SNHC. If the ce_SNHC is set to off, no additional information with respect to complexity estimation is sent through the bitstream. Otherwise, the Projected Surface Sphere(s) is (are) sent, from which the required processing power for 3D rendering of the 3D object can be estimated. Since the projected surface does not drastically change with the 3D mesh resolution, it is specified in the 3D_Mesh_Object_Header for all layers, instead of being specified in 3D_Mesh_Object_Layer.

For each 3D_Mesh_Object_Layer, information regarding the number of vertices, triangles and edges should be transmitted if the ce_SNHC is enabled in 3D_Mesh_Object_Header. Transmitting the number of vertices, triangles and edges at the level of the 3D_Mesh_Object_Layer has the advantage that the 3D rendering estimation can be performed at any 3D mesh resolution.

O.7 QoS for SNHC through upstream

Defining profiles and levels in SNHC has proved to be considerably difficult, since processing power required for decoding and rendering the 3D content varies with orders of magnitudes in time, while additionally the performance of decoders is highly dependent on the target platforms (software versus hardware implementations, very powerful graphics cards, …).

Figure V2 - 96 shows how this can be resolved by using Computational Graceful Degradation techniques in the decoder.

However, this does not resolve the problem of profile and level definitions, in which basically the data that is transmitted should comply to some predefined thresholds (equivalent to "compliant bitstreams"). Therefore, we provide an upstream, which syntax conveys some information that reveals to the encoder (server) the performances of the decoder (see Figure V2 - 97). This basically means that the decoder determines its capabilities in advance (one-shot calibration process), transmits this information to the encoder, which will "restrict" its data to the capabilities of the decoder (e.g. by mesh simplification). The data to be transmitted through the upstream is defined in subclause 6.3.12.3 (upstream_SNHC_QoS).

The decoder performances are measured by decoding and rendering different SNHC objects and measuring the frame rate in different configurations, i.e. a varying number of triangles, a varying number of pixels projected on the screen and different rendering modes. The performance information is transmitted using different performance curves. Each performance curve corresponds to the frame rate achieved for one particular setting of the number of triangles and rendering mode, but for a varying number of screen coverages.

© ISO/IEC 2001 – All rights reserved
Licensed to UNIVERSITY OF TEXAS AT [illegible]
Single user license only. Copying and networking prohibited.

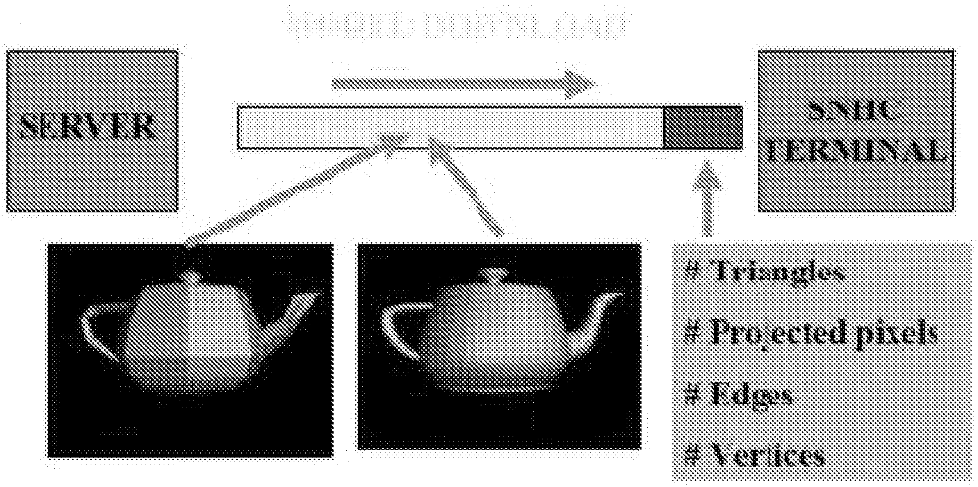


Figure V2 - 96 – Downloading complexity estimation (Computational Graceful Degradation) parameters to the decoder

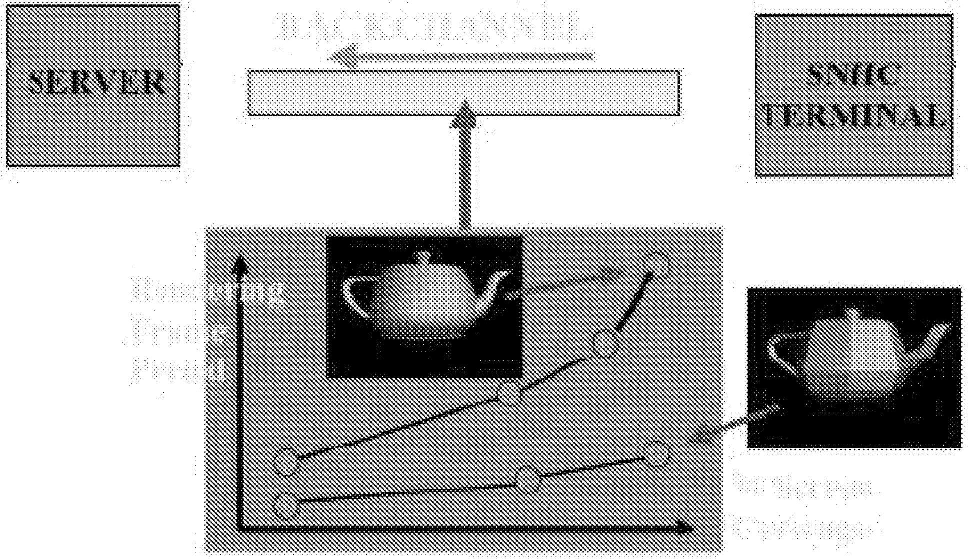


Figure V2 -97 – Uploading terminal (renderer) performance figures to the server, which will in turn transmit the data model at the correct Level of Detail for enabling real-time rendering at the decoder

O.8 References

[1] W.B. Pennebaker and J. Mitchell, "*JPEG Still Image Data Compression Standard*" New York: Van Nostrand Reinhold, 1993.

[2] Jiankun Li, Jin Li and C.-C. Jay Kuo, "An Embedded DCT Approach to Progressive Image Compression, ICIP'96," *IEEE International Conference on Image Processing*, Vol. 1.

[3] David Taubman and Aviceh Zakhor, "Multirate 3D Sub-band Coding of Video," *IEEE Transactions on Image Processing*, Vol 3.,No. 5, September 1994.

[4] G. Taubin, "A Signal Processing Approach To Fair Surface Design," *Siggraph'95 Conference Proceedings*, pp 351-358, Los Angeles, August 1995.

[5] G. Taubin and J. Rossignac, "Geometry Compression through Topological Surgery," *ACM Transactions on Graphics*, pp 84-115, April 1998.

[6] G. Taubin, W.P. Horn, F. Lazarus, and J. Rossignac, "Geometric Coding and VRML," *Proceedings of the IEEE, Special Issue on Multimedia Signal Processing*, July 1998.

[7] G. Taubin, W.P. Horn, and F. Lazarus, "*The VRML Compressed Binary Format*," 1997, http://www.research.ibm.com/vrml/binary.

[8] G. Taubin, A.Gueziec, W.P. Horn, F. Lazarus, "Progressive Forest Split Compression," *Siggraph'98 Conference Proceedings*, pp. 123-132, Orlando, July 1998.

[9] Gauthier Lafruit, Lode Nachtergaele, Kristof Denolf, Jan Bormans, "3D Computational Graceful Degradation," *The 2000 IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, May 2000.

[10] A. Gueziec, G. Taubin, F. Lazarus and W.P. Horn, "Converting Sets of Polygons to Manifold Surfaces by Cutting and Stitching," in *Visualization'98 Conference Proceedings*, Research Triangle Park, NC, October 1998.

[11] A. Gueziec, G. Taubin, F. Lazarus, and B. Horn, "A Framework for Streaming Geometry in VRML," *IEEE Computer Graphics & Applications*, special issue on VRML, March-April 1999.

[12] A. Gueziec, F. Bossen, G. Taubin, and C. Silva, "Efficient Compression of Non-Manifold Polygonal Meshes," IEEE Visualization'99, 1999.

[13] A. Gueziec, and G. Taubin, "A Framework for Memory-Efficient Levels of Detail," *Tenth Canadian Conference on Computational Geometry*, Montreal, Canada, August 1998.

[14] G. Taubin, and J. Rossignac, "Course on 3D Geometry Compression," *Siggraph'98*, Orlando, FL, July 1998.

[15] G. Taubin, and J. Rossignac, "Course on 3D Geometry Compression," *Siggraph'99*, Los Angeles, CA, 1999.

[16] G. Taubn, W. Horn, and P. Borrel, "Compression and Transmission of Multi-Resolution Clustered Meshes," *IBM Technical Report* RC-21398, February 1999.

[17] F. Bossen, "*On the Art of Compressing Three-dimensional Polygonal Meshes and their Associated Properties*," Thesis no. 2012, Ecole Polytechnique Federale de Lausanne, September 1999.

Bibliography

1 Arun N. Netravali and Barry G. Haskell "*Digital Pictures: representation and compression*" Plenum Press, 1988.

2 Barry G. Haskell, Atul Puri, Arun N. Netravali, "*Digital Video: An Introduction to MPEG-2,*" Chapman & Hall, ISBN 0-412-08411-2, 1997.

3 F. I. Parke, K. Waters, *Computer Facial Animation*, A K Peters, Wellesley, MA, USA., 1996.

ICS 35.040

Price based on 517 pages

XP-002179318

P.D. 00-09-1997
P. 1-42 (42)

OpenDML AVI File Format Extensions

Version 1.02

OpenDML AVI M-JPEG File Format Subcommittee

Last revision: February 28, 1996
Reformatting: September 1997

Please note
The information provided by this document is provided on an "as is" basis. No responsibility is assumed by Matrox Electronic Systems Ltd. for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Matrox Electronic Systems Ltd.

Table of Contents

1.0 **Introduction** 1

Scope 1

OpenDML Sub-Committee on File Format: High Level Goals 1

Version History 1

1.0 **Overview of Profession Video Requirements** 2

Goal 2

Issues 2

AVI: Support Current AVI Format 2

60 Fields Per Second vs. 30 Frames Per Second; or 24 Frames Per Second For Film 2

Improved Tombstone and Header Information 2

2.0 **Increased AVI File Size** 3

Introduction: RIFF Chunk Format 3

Increased File Size Limits (>> 1 GB) 3

Extension for File Size > 1 GB 3

3.0 **Frame and Field Indexing** 5

Introduction: 'idx1' Index Chunk 5

Issues 5

Allow Field Indexing 5

Allow Incremental Growth of Files 5

Minimize Disk Seeks 5

Proposed Index List 6

Overview 6

Base Index Form 'indx' 6

AVI Standard Index Chunk 7

AVI Field Index Chunk 8

AVI Super Index Chunk 9

Index Locations in RIFF File 10

4.0 **Interleaving Audio with Video** 13

Issues 13

Interleaving Rate 13

Single or Multiple Files for Video and Audio 13
Multiple Tracks of Audio Storage 13
Mono and Stereo Audio 13
Audio / Video Time Stamping (Sync Point List) 13
Notes on Interleaving Audio and Video 13
5.0 Source and Header Information Storage 15
Goal 15
Required Information 15
Main AVI Header (avih) 15
AVI Stream Header (strh) 15
AVI Stream Format (strf) 16
Extended AVI Header (dmlh) 16
Video Properties Header (vprp) 16
Source and Timecode Information 20
Film Transfer Log Information 22
Microsoft-Defined Tombstone Data 23
6.0 Other File Issues 28
Issues 28
Continuation Over Multiple Files 28
Minimized Time to Open Files 28
Simultaneous Reading and Writing / Multiple Access 28
7.0 MJPG Motion JPEG DIB Extensions 30
Issues 30
ISO vs. Non ISO Formats 30
8-bit vs. 10-bit YUV 30
Video vs. Bitmap 30
Proposed Data Chunk Format 30
From the DIB spec: 30
And a new proposal: 32
MJPEG Baseline DCT - YCbCr DIB Map 33
Sector Alignment and Padding 34
8.0 OpenDML File Format Certification Procedures 36
Concurrently Develop a Test Suite to measure compatibility 36

Table of Contents

1.0 Introduction

The OpenDML File Format Subcommittee is defining an *AVI-compatible* file format that addresses the particular needs of professional video. As such, this document relates specific proposals to these needs.

Further work by OpenDML will elaborate on extensions to the Codec and Vidcap specifications of Video for Windows to meet the needs of professional video.

Scope

This document describes the proposed format of the OpenDML-compliant AVI extensions, specifically for the Motion-JPEG DIB AVI file. This format is an extension to the AVI M-JPEG DIB as defined in the *JPEG DIB FORMAT* technical note from Microsoft. The format is based on the ISO 10918 defined JPEG format.

It is assumed that the reader is familiar with JPEG as defined in the ISO 10918 document. For additional information on JPEG, see the *ISO 10918 Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines.*

For additional information about the JPEG DIB format, see Microsoft Multimedia Technical Note *JPEG DIB FORMAT.*

For additional information about RIFF files see the Microsoft Windows Software Development Kit *Multimedia Programmer's Guide* and *Multimedia Programmer's Reference.*

For additional information on installable compressors and decompressors, see the *Video Compression/Decompression Drivers* technical note from Microsoft.

Portions of the above Microsoft documents have been reprinted with permission of Microsoft Corp.

OpenDML Sub-Committee on File Format: High Level Goals

- Interoperability and AVI Compatibility
- Large File Support
- Frame and Field Indexing
- Interleaving Audio with Video
- Source and Header Information Storage
- OpenDML File Format Certification Code
- Microsoft Certification and Communication

Version History

Version 1.0	November 13, 1995	First Release
Version 1.1	December 18, 1995	Includes small changes from Microsoft

1.0 Overview of Profession Video Requirements

Goal

Provide interoperability between different hardware and software vendors' Motion JPEG codecs. Support Video for Windows AVI file format for MJPG video, while improving the performance of these systems.

Issues

AVI: Support Current AVI Format

The goal is to support interoperability by standardizing on the AVI file format for Motion-JPEG video. This implies a minimum compatibility to support the standard AVI file access interface, as well as the vidcap and codecs that are part of Video for Windows. This means support of the MJPG DIB specification, supporting the RIFF file format as well as the standard AVI stream headers, allowing for optional audio interleaving, and supporting one storage of one frame of video per ##dc data chunk in the LIST 'movi' chunk of the AVI file.

60 Fields Per Second vs. 30 Frames Per Second; or 24 Frames Per Second For Film

Professional Video Applications require the ability to sequence individual fields of the video data for certain playback rates and video effects. For example, in order to do a slow motion effect, it is not sufficient to just repeat the frames; the fields must be individually repeated. For example, as sequence of fields in a file numbered 123456... should be played at half speed as 11223344... rather than 12123434.... As such, the individual fields should be accessible. AVI requires that frames are stored per data chunk. Storing individual fields in each chunk would break AVI applications.

The extended AVI format will allow for access to individual fields. In addition, the format will allow for storage of 24 frame based files, such that file readers can properly convert between 24 frame-based data storage and 25 or 30 frame PAL and NTSC playback.

Improved Tombstone and Header Information

Professional video applications need extended information such as starting timecode and reel ID to be contained inside the AVI file. More advanced information such as the location of timecode discontinuities, etc. are also desirable.

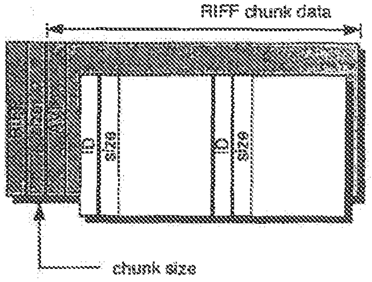
2.0 Increased AVI File Size

Introduction: RIFF Chunk Format

The AVI file begins as a standard RIFF chunk (note that this is a non-standard representation that shows the 4-byte size):

```
RIFF (size) 'AVI ' ...
```

Standard RIFF AVI form



There is only one RIFF chunk per file. RIFF sub-chunks may be either LIST chunks or regular sub-chunks. The LIST chunk obeys the same structure and may have regular or LIST sub-chunks; all other RIFF sub-chunks have just an ID identifier and a size. These regular sub-chunks may not have sub-chunks.

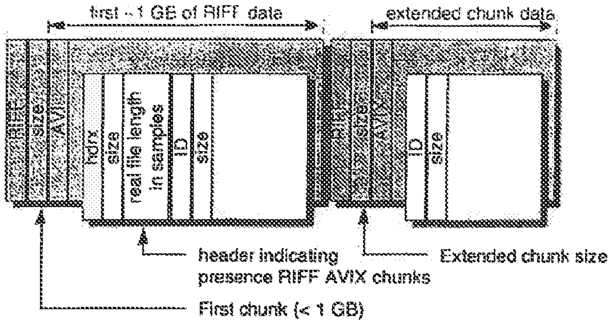
Increased File Size Limits (>> 1 GB)

The current RIFF file format implies a maximum chunk size of 4 GB because the size is stored as a 32-bit value. However, limitations to the RIFF parser code and MCIAVI limits the file to only 1 GB. At data rates of 10 MB/s, a 1 GB file will last less than 2 minutes.

Extension for File Size > 1 GB

An AVI file can be extended beyond 1 GB by placing more than one RIFF chunk in the same file (which is legal). Each RIFF chunk can have a maximum size of 1 GB. Standard AVI applications will see the first RIFF chunk (RIFF 'AVI") as a standard AVI file, and this chunk should be complete. Information that the file is extended over 1 GB should be placed in the Extended AVI header (see below). Subsequent RIFF chunks will be identified by the AVIX (for AVI extended) chunk id. These RIFF 'AVIX' chunks do not need to contain further data. It is expected that the AVIX chunks only contain LIST 'movi' data. All other AVI information should be stored in the first (RIFF 'AVI') chunk so that it is accessible by all applications.

RIFF AVI form for files > 1 GB

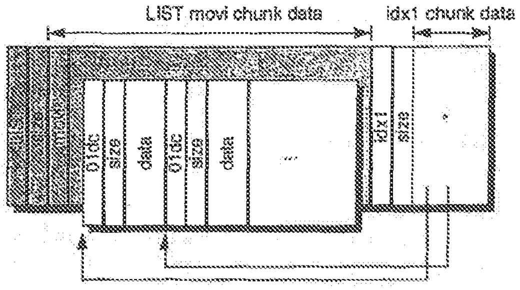


3.0 Frame and Field Indexing

Introduction: 'idx1' Index Chunk

The AVI file format specifies an optional 'idx1' chunk, which contains a list of the offsets and chunk types of every chunk inside the LIST 'movi' chunk. This list is used to make AVI playback and seeks more efficient because the location of each frame of video can be found without scanning through each sub-chunk of the LIST 'movi' data.

Standard idx1 form



* data of idx1 chunk is an index to each movi sub-chunk, includes offset and length of each chunk

Issues

Allow Field Indexing

In order to support field accurate indexing while still retaining AVI compatibility, the video data must be stored one frame per ##dc 'movi' chunk, while an index must be present that gives the locations of both fields (if they exist) within the frame. Applications could then use field-based codecs to perform field-based effects (e.g., slow motion, etc.).

Allow Incremental Growth of Files

The 'idx1' chunk follows the LIST 'movi' chunk in the file. As such, it must be moved in order to insert new data into the file. An interleaved frame index would be more efficient for the purposes of file growth.

Minimize Disk Seeks

In playback, there are several causes of inefficiency. One is that the index is at the end of the file, requiring a large seek when opening the file. The other is that the index itself contains all the chunks inside the 'movi' data, not just those for one AVI stream. As such, the index must be preprocessed so that the location of each frame in the index can be found.

Proposed Index List

Overview

Instead of using an index chunk like 'idx1', a new index structure is proposed. The index has the following main characteristics:

- The can either be a single index chunk, or a two tiered index, with a super index pointing to interleaved index segments in the 'movi' data.
- The index can include locations of the fields within a frame.
- The index has no restrictions to 4 GB file size.
- There is an index per data stream (if necessary).

The advantages of the index are as follows:

- The index can contain the locations of the individual fields in each chunk as well as the chunk information as a whole. The 'idx1' chunk only contains indices of the frames.
- The index segments are interleaved within the 'movi' data, meaning that the file can be grown. The 'idx1' chunk is present at the end of the file. It must be moved for the file to grow.
- The index segment is smaller than an 'idx1' chunk, so the amount of data needed to be read in order to access a particular frame is less. Also, if the index is interleaved before its data (i.e., less efficient in write), then playback will be more efficient than the 'idx1' chunk.
- The index lists are present per stream number (##). As such, the list can be accesses like an array, unlike the 'idx1' chunk, which contains entries for all 'movi' chunks from all streams.

The index is implemented as an index of indexes (optional) and an index of chunks. Both are based on a base index form.

Base Index Form 'indx'

Thus the actual implementation is based on a base index form 'indx':

```
struct _aviindex_chunk {
   FOURCC fcc;
   DWORD  cb;
   WORD   wLongsPerEntry;   // size of each entry in aIndex array
   BYTE   bIndexSubType;    // future use.  must be 0
   BYTE   bIndexType;       // one of AVI_INDEX_* codes
   DWORD  nEntriesInUse;    // index of first unused member in aIndex array
   DWORD  dwChunkId;        // fcc of what is indexed
   DWORD  dwReserved[3];    // meaning differs for each index
                            // type/subtype.   0 if unused
   struct _aviindex_entry {
      DWORD adw[wLongsPerEntry];
   } aIndex[ ];
};
```

The actual size of the entries of the `aIndex` array is entered into `wLongsPerEntry`. Between the `cb` field and `wLongsPerEntry` the actual size of the array is known. The field `nEntriesInUse` allows a chunk to be allocated longer than the actual number of used elements in the array.

The types defined for the `bIndexType` and `bIndexSubtype` field are the following:

```
// bIndexType codes
//
#define AVI_INDEX_OF_INDEXES 0x00    // when each entry in aIndex
                                             // array points to an index chunk
#define AVI_INDEX_OF_CHUNKS  0x01    // when each entry in aIndex
                                             // array   points to a chunk in the
file
#define AVI_INDEX_IS_DATA        0x80        // when each entry is aIndex is
                                             // really the data
// bIndexSubtype codes for INDEX_OF_CHUNKS
//
#define AVI_INDEX_2FIELD      0x01    // when fields within frames
                                             // are also indexed
```

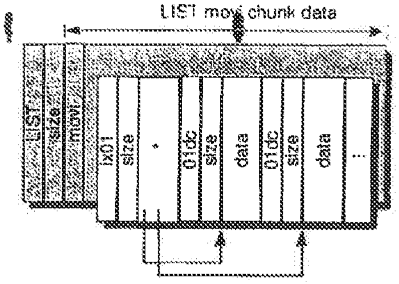
AVI Standard Index Chunk

The AVI Standard Index chunk contains information that indexes AVI frames.

```
typedef struct _avistdindex_chunk {
   FOURCC   fcc;              // 'ix##'
   DWORD    cb;
   WORD     wLongsPerEntry;  // must be sizeof(aIndex[0])/sizeof(DWORD)
   BYTE     bIndexSubType;   // must be 0
   BYTE     bIndexType;      // must be AVI_INDEX_OF_CHUNKS
   DWORD    nEntriesInUse;   //
   DWORD    dwChunkId;       // '##dc' or '##db' or '##wb' etc..
   QUADWORD qwBaseOffset;    // all dwOffsets in aIndex array are
                             // relative to this
   DWORD    dwReserved3;     // must be 0
   struct _avistdindex_entry {
      DWORD    dwOffset;     // qwBaseOffset + this is absolute file offset
      DWORD    dwSize;       // bit 31 is set if this is NOT a keyframe
   } aIndex[ ];
} AVISTDINDEX, * PAVISTDINDEX;
```

A single standard index chunk can only index data within a 4 GB region. The `dwOffset` field points to the start of the data itself, and not to the start of the RIFF chunk for that field.

Standard Index Chunk



* data of Standard AVI index chunk is an index to each movi chunk data, includes offset and length of chunk.

The index chunk above is shown in the 'movi' data, but may be found in the stream header (see below).

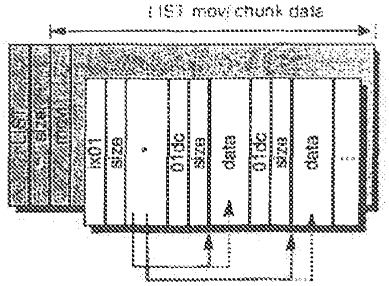
AVI Field Index Chunk

The AVI Field Index Chunk is the same as the Standard Index Chunk except that it contains the locations of each field in the frame.

```
typedef struct _avifieldindex_chunk {
   FOURCC    fcc;               // 'ix##'
   DWORD     cb;
   WORD      wLongsPerEntry;    // must be 3 (size of each entry in
                                // aIndex array)
   BYTE      bIndexSubType;     // AVI_INDEX_2FIELD
   BYTE      bIndexType;        // AVI_INDEX_OF_CHUNKS
   DWORD     nEntriesInUse;     //
   DWORD     dwChunkId;         // '##dc' or '##db'
   QUADWORD qwBaseOffset;       // offsets in aIndex array are relative to this
   DWORD     dwReserved3;       // must be 0
   struct _avifieldindex_entry {
      DWORD    dwOffset;
      DWORD    dwSize;          // size of all fields
                                  // (bit 31 set for NON-keyframes)
      DWORD    dwOffsetField2; // offset to second field
   } aIndex[  ];
} AVIFIELDINDEX, * PAVIFIELDINDEX;
```

The field `wLongsPerEntry` is set to 3 because of the addition of the `dwOffsetField2` field in the array. This is indicated by setting the sub type to `AVI_INDEX_2FIELD`.

Field Index Chunk



* data of Field Index Chunk is an index to each movi sub-chunk data, includes offset and length of chunk, as well as offset to the second field.

The index chunk above is shown in the movi data, but may be found in the stream header (see below).

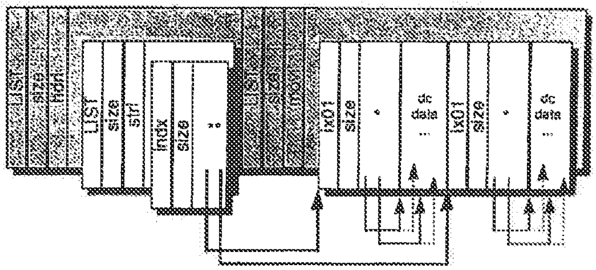
AVI Super Index Chunk

The Super Index Chunk is an index of indexes and is always found in the 'indx' chunk of an AVI file. It is defined as follows:

```
typedef struct _avisuperindex_chunk {
   FOURCC  fcc;             // 'ix##'
   DWORD   cb;              // size of this structure
   WORD    wLongsPerEntry;  // must be 4 (size of each entry in aIndex array)
   BYTE    bIndexSubType;   // must be 0 or AVI_INDEX_2FIELD
   BYTE    bIndexType;      // must be AVI_INDEX_OF_INDEXES
   DWORD   nEntriesInUse;   // number of entries in aIndex array that
                            // are used
   DWORD   dwChunkId;       // '##dc' or '##db' or '##wb', etc
   DWORD   dwReserved[3];   // must be 0
   struct _avisuperindex_entry {
      QUADWORD qwOffset;    // absolute file offset, offset 0 is
                            // unused entry??
      DWORD    dwSize;      // size of index chunk at this offset
      DWORD    dwDuration;  // time span in stream ticks
   } aIndex[ ];
} AVISUPERINDEX, * PAVISUPERINDEX;
```

The `bIndexSubType` is set to the type of index that the Super Index points to. If the index chunks are Standard Index Chunks, then the value should be 0. If the index chunks are AVI Field Index Chunks, then the value should be `AVI_INDEX_2FIELD`. This implies that a stream cannot mix Field and Standard Index Chunks.

Super Index Chunk



* data of ##ix chunk is an index to each movi sub-chunk data, includes offset and length of chunk, and possibly the offset to the second field.
** data of Super Index indx chunk is an index to each ##ix movi sub-chunk.

If we use the ##pc chunks, then they must be indexed by their own Index List: Palette changes should not appear in the same Index List as ##dc chunks, since they eliminate the ability to access an element (frame) directly without having to scan the list.

Index Locations in RIFF File

Unlike the 'idx1' chunk, a single index is stored per *stream* in the AVI file. An 'indx' chunk follows the 'strf' chunk in the LIST 'strl' chunk of an AVI header. This 'indx' chunk may either be an index of indexes (super index), or may be an index to the chunks directly. In the case of video, this means that the chunk is either a `AVISUPERINDEX` or an `AVIFIELDINDEX/AVISTDINDEX`.

If the 'indx' chunk is a standard or field index chunk (i.e., not an index of indexes) then the stream has only one index chunk and there is none in the 'movi' data.

If the 'indx' chunk is a Super Index, then the corresponding index chunks are marked with 'ix##' in the 'movi' data. The ## is the stream number, the same as for the ##dc or ##wb. The index chunks can be either standard or field index chunks.

A file can be easily grown if it has a standard index in the 'indx' chunk position. The chunk can be moved to a new 'ix##' chunk, and a new super index can be inserted into the stream header ('indx' position). New 'ix##' chunks can be added to grow the file.

Note the reversal of the '##ix' to 'ix##'. This is for AVI backward compatibility. Note that INDEX_IS_DATA streams remain as '##ix'.

Here is a sample RIFF file in the format shown by the RIFFWALK utility:

```
RIFF (7F038718) 'AVI '
     LIST (0001084C) 'hdrl'
         avih (00000038)
```

```
        LIST (000044E0) 'strl'
            strh (00000038)
            strf (00000428)
            indx (00003FF8)    <- video superindex
            vprp (00000064)    <- video property header
        LIST (0000405E) 'strl'
            strh (00000038)
            strf (00000012)
            indx (00003FF8)    <- audio superindex
        LIST (0000404C) 'strl'
            strh (00000038)    <- timecode stream...
            strf (00000000)
            indx (00003FF8)    <- timecode superindex
        LIST (00000060) 'odml'
            dmlh (00000054)    <- size still temp.
    LIST (7F000000) 'movi'
        01wb (00002dF0)
        00db (00030004)
        01wb (00002dF0)
        00db (00030004)
        ... time passes
        02ix (00003FF8)     <- data chunk for timecode stream
        ... time passes
        ix00 (00003FF8)     <- partial video index
        ix01 (00003FF8)     <- partial audio index
        .. etc ..
    idx1 (00030000)         <- original index (optional)
RIFF (7F038718) 'AVIX'
```

4.0 Interleaving Audio with Video

Issues

Interleaving Rate
Files for Editing
Files for Playback

Single or Multiple Files for Video and Audio

Multiple Tracks of Audio Storage

Mono and Stereo Audio

Support of multiple streams of data exists in AVI. A proposal to recommend that audio be stored as two mono streams should be examined. Not all applications would be able to play this back in stereo, however.

Audio / Video Time Stamping (Sync Point List)

It is necessary to add time stamping information to account (and correct) for drift between audio and video when using hardware that is not able to maintain a perfect synchronization between audio and video (via a hardware connection) and when there exists a drift in the crystal frequency on audio hardware.

This can be accomplished by adding a sync point list into the file. A sync point list identifies two samples in different streams that are to be played at the same time. Thus, drift can be corrected by identifying sync points and if necessary, correcting for the drift during playback.

Notes on Interleaving Audio and Video

With the exception of AVI files playing off CD-ROM drives, it is not necessary to interleave the audio and video into REC chunks. In fact, with higher video and audio data rates, it is probably more efficient to store the audio in large consecutive blocks. As such, the audio can be interleaved in large blocks (not necessarily done on a frame basis).

Audio data chunks may or may not be in integral frames, or multiple of frames. The other option would be a constant number of bytes. In addition, audio may be interleaved before or after the corresponding video. This is just a tradeoff between record and playback performance. Typically, saving audio chunks after video in record will give better record performance, which is important because hard drives tend to be less performant during write operations. However, during playback, multiple streams and cuts may be played, reducing the drive performance.

5.0 Source and Header Information Storage

Goal

The goal is to implement a standard way to store certain header and tombstone information in an AVI file so that other applications can interpret and use this data. In many cases, the header data is not uniformly set between different AVI files.

Required Information

Main AVI Header (avih)

Total Frames

The `dwTotalFrames` field indicates the size (number of frames) within the first RIFF 'AVI' chunk.

AVI Stream Header (strh)

Quality Information

AVI defines the quality of a clip by a DWORD value from 0 to 10000. This value seems arbitrary. For MJPG files, a standard quality should be defined. The quality can be defined by either of these three modes:

- constant average data rate
- constant Q-table
- lossless

Lossless is not part of the Baseline process. As such it requires the definition of a new Process. Under baseline, lossless can be achieved by setting the Q-tables to unity[1].

Under constant data rate, the desired data rate for the file should be specified (in kb/s). This, plus the number of fields per frame, should be able to determine the Q-tables required to compress the image.

---

[1] While a lossless (or pseudo lossless) process can be defined by setting the Q-tables to unity, the ISO specification of lossless JPEG compression is a different process. Lossless encoding as defined by ISO does not use DCT transformed data. For a sample to be encoded, a predictor is formed from the reconstructed values of up to three neighborhood samples. This gives a prediction of the of the sample to be encoded. The prediction is subtracted from the actual value of the sample and the difference is losslessly entropy coded using either Huffman or arithmetic coding. As far as forming a type of pseudo lossless compression from a baseline JPEG process with unity Q tables, there is the potential of some slight loss of data during the DCT process due to arithmetic precision. Practically speaking, there is no real image quality loss, but the word *lossless* implies a guarantee of reconstructing the source image on a bit by bit basis.

Under constant Q-table, there are two possibilities. Either the absolute Q-tables are given (8x8 matrices) per component (in our case, 2), or a factor which is used to generate any Q-table of a standard one. The constant Q-table file would of course have the same tables repeated per field, as per the MJPG DIB spec. The advantage of a Q-factor is the reduction of header information. However, tweaked Q-tables would no longer be possible.

Scale / Rate

The scale and rate parameters will define the correct ratio for different video standards. Known ratio values are (rate/scale):

NTSC	30000/1001
PAL	25/1

rcFrame

The on-disk RECT coordinates are 16-bit values. The RECT structure in NT is a 32-bit (LONG) for each coordinate, but the on-disk ones remain 16-bit.

AVI Stream Format (strf)

Height

The MJPG DIB spec lists that heights less than 288 are for single frames (one chunk) and greater than 288 are for interleaved fields. However, there is no real connection between frame height and interleaving. The **biHeight** parameter refers to the raw height of the frame (interleaved or not). OpenDML codecs that operate in field by field modes will be passed this value divided by 2 to get the field height.

Extended AVI Header (dmlh)

```
typedef struct {
   DWORD dwTotalFrames;
} ODMLExtendedAVIHeader;
```

Total Frames

The `dwTotalFrames` field indicates the **real** size of the AVI file. Since the same field in the Main AVI Header 'avih' indicates the size within the first RIFF 'AVI' chunk.

Video Properties Header (vprp)

The video properties header identifies video signal properties associated with a digital video stream in an AVI file. This header attempts to address two main video properties:

- The type of video signal (PAL, NTSC, etc., as well as the resolution of the video signal).
- The framing of the compression within a video signal.

The parameters can be used to uniquely describe a video signal.

```
typedef struct {
   DWORD           VideoFormatToken;
   DWORD           VideoStandard;
   DWORD           dwVerticalRefreshRate;
   DWORD           dwHTotalInT;
   DWORD           dwVTotalInLines;
```

```
    DWORD               dwFrameAspectRatio;
    DWORD               dwFrameWidthInPixels;
    DWORD               dwFrameHeightInLines;
    DWORD               nbFieldPerFrame;
    VIDEO_FIELD_DESC    FieldInfo[nbFieldPerFrame];
} VideoPropHeader;
```

Video Format Token

```
enum {FORMAT_UNKNOWN, FORMAT_PAL_SQUARE, FORMAT_PAL_CCIR_601,
   FORMAT_NTSC_SQUARE, FORMAT_NTSC_CCIR_601,...} VIDEO_FORMAT;
```

The format token indicates that a known standard is defined for the following data fields. Those fields must be filled, but their value can be expected to be the defined standard. If the format is defined as `FORMAT_UNKNOWN` then the fields may contain special values. Known tokens are defined in the table below:

Token	*Standard*	*Refresh Rate*	*H-Total*	*V-Total*	*Frame Aspect Ratio*	*Frame Width*	*Frame Height*	*Pixel Aspect Ratio (derived)*
NTSC CCIR 601	NTSC	60	858	525	0x00040003 (4:3)	720	485	2160:1940
NTSC SQUARE	NTSC	60	780	525	0x00040003 (4:3)	640	480	1:1
PAL CCIR 601	PAL	50	864	625	0x00040003 (4:3)	720	576	2160:2304
PAL SQUARE	PAL	50	944	625	0x00040003 (4:3)	768	576	1:1

Video Standard

```
enum {STANDARD_UNKNOWN, STANDARD_PAL, STANDARD_NTSC, STANDARD_SECAM}
   VIDEO_STANDARD;
```

Defines standards such as NTSC, PAL etc. Implicitly defines vertical refresh rate.

Vertical Refresh Rate

Used when an unknown standard is specified. Normally, 60 for NTSC, and 50 for PAL.

H-Total in T

Defines the horizontal total, in T (one luminance sample: pixel)

V-Total in Lines

Defines the vertical total, in lines.

Active Frame Aspect Ratio

The aspect ratio is stored as a DWORD value with a word each storing the x:y ratio. For example, 1 to 1 is 0x00010001. Standard values for television is 4:3 or 16:9. This value can be used with the frame width and height to calculate the pixel aspect ratio.

Active Frame Width in Pixels

Defines the active frame width in pixels. The bitmap might digitize a region that is smaller or bigger than the active video width.

Active Frame Height in Lines

Defines the frame height in lines. The bitmap might digitize a region that is smaller or bigger than the active video height.

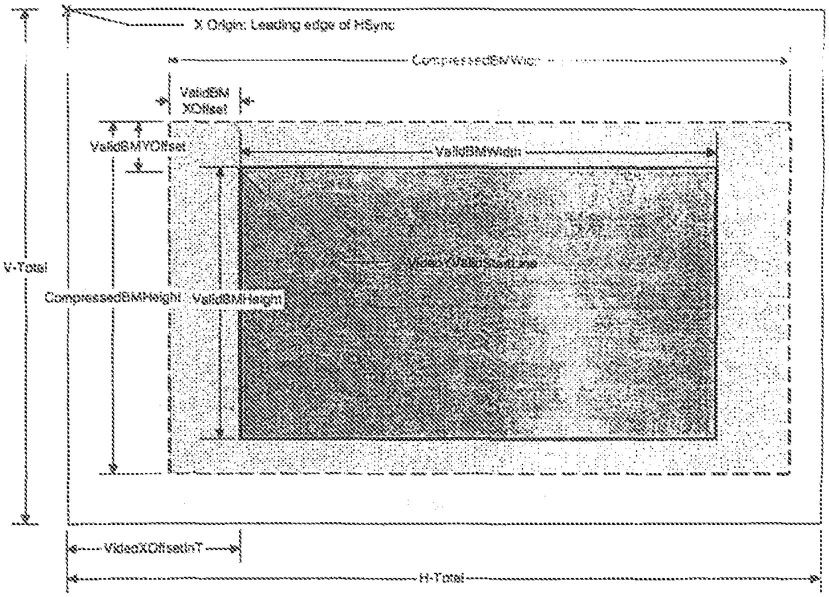
Number of Fields Per Frame

One or two, depending on whether the video is interlaced or progressive.

Field Framing Information

The field framing information defines where the compressed image is with respect to the video signal. The data is present for each field (and may be different).

```
Typedef struct {
   DWORD   CompressedBMHeight;
   DWORD   CompressedBMWidth;
   DWORD   ValidBMHeight;
   DWORD   ValidBMWidth;
   DWORD   ValidBMXOffset;
   DWORD   ValidBMYOffset;
   DWORD   VideoXOffsetInT;
   DWORD   VideoYValidStartLine;
} VIDEO_FIELD_DESC;
```



Compressed Bitmap Height and Width

The compressed bitmap height and width represent the size of the compressed image. For JPEG, these values are multiples of 8.

Valid Bitmap Height and Width, X and Y Offset

The valid bitmap height, width and x and y offsets represent the size of the valid data within the compressed bitmap. Because padding may be required when compressing, it is not guaranteed that all the data within the compressed image is valid. Note that compressing blanking is still valid. In the case where all the compressed bitmap comes from the video signal, then the valid height and width are equal to the compressed height and width, and the offsets are 0.

Valid X-Offset In T

The `VideoXOffsetInT` is used to locate the x position of the start of the valid bitmap with reference to the video signal. The value is a measurement in units of T, which is one luminance-sampling clock, from the leading edge of the horizontal sync pulse (CCIR 624-3).

Valid Y Start Line

The `VideoYValidStartLine` field is used to locate the line that the valid bitmap starts on. This value will be different for each field. (CCIR 624-3).

A typical value for the Framing Information for NTSC CCIR 601 would be:

Vide	*Co*	*Co*	*Vali*	*Vali*	*Vali*	*Va*	*Vide*	*Video*

o Type	*mpressed BM Height*	*mpressed BM Width*	*d BM Height*	*d BM Width*	*d BM X Offset*	*lid BM Y Offset*	*o X Offset In T*	*Y Valid Start Line*
Field 1	248	720	248	720	0	0	122	20
Field 1	248	720	248	720	0	0	122	283

Source and Timecode Information

The following information is used to describe timecode and source information inside an AVI file.

Base Timecode Structure

All SMPTE timecode information is stored in the following format:

```
typedef union _timecode {
   struct {
      WORD       wFrameRate;  // 0 is 30 drop.  do we need other drop
                              // frame rates?
      WORD       wFrameFract; // fractional frame. full scale is always 0x10000
      LONG       lFrame;
   };
   __int64  qwAll;
} TIMECODE, *PTIMECODE;
```

This format allows for a frame, and a frame fractional value to be specified, with the frame rate stored in the low word of the value. As the values are stored, two timecodes can be compared and subtracted. For drop frame code, no `lFrame` values are skipped, so a drop frame timecode of 1:00:00;00 would be a `lFrame` value of 107892.

Timecode Discontinuity Table (Stream) 'tcdl'

The following structure defines a timecode discontinuity table. This would be stored as a stream itself. The table would either be a table alone in the 'indx' chunk of that stream, or if the table is large, an index to the table could be stored with the table itself stored as a ##ix in the 'movi' data.

```
#define FILM_SEQUENCE_NONE       0
#define FILM_SEQUENCE_AABBBCCDDD 1
#define FILM_SEQUENCE_AAABBCCCDD 2
// structure of a timecode discontinuity table. note that this fits
// within the definition of the structure of an index chunk
//
typedef struct _timecode_dl_chunk {
   FOURCC    fcc;               // '##ix'
   DWORD     cb;                // sizeof this structure less 8
   WORD      wLongsPerEntry;    // must be 4 or more
   BYTE      bIndexSubType;     // must be 0
   BYTE      bIndexType;        // AVI_INDEX_IS_DATA
   DWORD     nEntriesInUse;     // index of first unused entry in aIndex array
   DWORD     dwChunkId;         // 'tcdl' (timecode discontinuity list)
   DWORD     dwReserved[3];     // future, must be 0
   struct _timecode_dl_entry {
      QUADWORD  qwTick;         // time in terms of this streams's tick rate
      TIMECODE  timecode;       // timecode
      DWORD     dwUser;         // timecode user data
      struct {
         DWORD  ColorFrame : 4;        // Which frame in color sequence
         DWORD  ColorSequence : 4;     // Duration in frames of complete sequence
         DWORD  FilmFrame : 5;         // Offset into pull-down sequence
         DWORD  FilmSequenceType : 3;  // One of FILM_SEQUENCE_XXX defines
         DWORD  Reserved : 16;         // Future use - set to 0
      } Flags;
      TCHAR     szReelId[32]; // source id
   } aIndex[ ];
};
```

The Flags structure can optionally specify color framing, and film pull-down information. If all fields are 0 then no such optional information is present. Notice that there is no explicit flag for drop frame NTSC since that information in already stored in the TIMECODE structure.

If `ColorSequence` is non-zero value, then `ColorFrame` contains a frame number from 0 to `ColorSequence-1`, which specifies the color framing of the frame specified in the timecode structure.

If `FilmSequenceType` is not `FILM_SEQUENCE_NONE`, then `FilmFrame` contains the offset into the pull-down sequence corresponding to the frame specified in the timecode structure. For example, if `FilmFrame` is 1 and `FilmSequenceType` is `FILM_SEQUENCE_AABBBCCDDD` then the referenced frame is a BB frame. Again, if `FilmFrame` is 1 and `FilmSequenceType` is `FILM_SEQUENCE_AAABBCCCDD` then the referenced frame is an AB frame.

Timecode Stream 'time'

This structure defines a timecode value for every tick of the timecode stream

```
// structure of a timecode stream data chunk. for storing every
// event in a timecode stream rather than just the discontinuities
//
typedef struct _timecode_stream_chunk {
   FOURCC    fcc;              // '##ix'
   DWORD     cb;               // sizeof this structure less 8
   WORD      wLongsPerEntry;   // must be 3
```

```
    BYTE      bIndexSubType;   // must be 0
    BYTE      bIndexType;      // AVI_INDEX_IS_DATA
    DWORD     nEntriesInUse;   // index of first unused entry in aIndex   array
    DWORD     dwChunkId;       // 'time' (timecode stream)
    DWORD     dwReserved[3];   // future, must be 0
    struct _timecode_stream_entry {
       TIMECODE  timecode;     // timecode
       DWORD     dwUser;       // timecode user data
       struct {
          DWORD  ColorFrame : 4;        // Which frame in color sequence
          DWORD  ColorSequence : 4;     // Duration in frames of complete sequence
          DWORD  FilmFrame : 5;         // Offset into sequence
          DWORD  FilmSequenceType : 3;  // One of FILM_SEQUENCE_XXX defines
          DWORD  Reserved : 16;         // Future use - set to 0
       } Flags;
    } aIndex[ ];
};
```

Film Transfer Log Information

```
#define FILM_LOG_EVERTZ          1
#define FILM_LOG_FLEX            2
// structure of a film transfer log.  Note that this fits
// within the definition of the structure of an index chunk
//
typedef struct _film_transfer_chunk {
    FOURCC    fcc;              // '##ix'
    DWORD     cb;               // sizeof this structure less 8
    WORD      wLongsPerEntry;   // must be 4 or more
    BYTE      bIndexSubType;    // must be 0
    BYTE      bIndexType;       // AVI_INDEX_IS_DATA
    DWORD     nEntriesInUse;    // index of first unused entry in aIndex array
    DWORD     dwChunkId;        // 'film' (film transfer log)
    DWORD     dwReserved[3];    // future, must be 0
    struct _film_transfer_header {
       DWORD     dwLogType;     // Which type of FILM_LOG_XXX file format
       DWORD     dwHeaderSize;  // Size of this _film_transfer_header structure
       DWORD     dwReserved[4];// Future use - set to 0
       // DWORD  dwUserDefined[];  // Optional user data
    } FilmHeader;
    // BYTE  bTransferLog[];      // The actual transfer log begins here
};
```

The size of the `FilmHeader` is given by the `dwHeaderSize`. This allows for user defined additions to the header, which appear at the end of the `FilmHeader`. The ASCII film transfer log begins immediately after the `FilmHeader` structure. The size of the ASCII data can be determined from the `dwHeaderSize` of the header and the `cb` size of the entire structure. The ASCII data should be padded with NULLs to the nearest DWORD structure size.

The contents of the `bTransferLog` array is byte for byte the contents of the ASCII transfer log file of the type given in the `dwLogType` field. For now, two types are defined:

FILM_LOG_EVERTZ: the detailed .FTL file format can be obtained from

Evertz Microsystems, Ltd.

3465 Mainway

Burlington, Ontario L7M 1A9, Canada

905-335-3700

FILM_LOG_FLEX: the detailed FLEX file format can be obtained from

Time Logic, Inc.

11992 Challenger Ct.

Moorpark, CA 93021

805-529-1155

Microsoft-Defined Tombstone Data

The following tombstone data has already been defined by Microsoft:

Additional AVI information following the 'strl' chunk in the AVI header LIST 'hdrl' chunk. (from *AVI RIFF form, Video for Windows SDK*):

Additional Header Data

After the 'strl' chunk, some AVI files might have additional header data. Additional information can include timecode chunks that apply to the whole file. The following timecode four-character codes are useful in AVI files:[2]

Four-character code	Description
ISMP	Indicates the chunk contains the SMPTE timecode of the digitization starting point. The time is expressed as a zero-terminated text string of the form HH:MM:SS.FF. If performing MCI capture in AVICAP, this chunk is automatically set from the MCI start time.
IDIT	Indicates the chunk specifies the time and date digitizing commenced. This time is contained in an ASCII string of

---

[2] Note that these chunks are Ixxx chunks, and as such, should be found in an INFO chunk (see comment below)?

exactly 26 characters and has the format "Wed Jan 02 02:03:55 1990\n\0". The ctime and asctime functions can be used to create strings in this format.

The 'ISMP' timecode chunk usually corresponds to the starting timecode copied from the first sample of a captured sequence. Use this chunk to index the captured sample to the original sample. For example, if you need to recapture a video sequence, you can use the timecode to reposition the original sequence and restart capture from that point.

The 'IDIT' timecode chunk usually corresponds to the time the sample was captured. This timecode provides a reference to the age of a sample and creates a history if a series of samples is captured.

The weakness with the above section is that the ISMP information does not include any reel ID, so the effective timecode is meaningless.

Additional information, global to RIFF files (from *Multimedia File Formats, Windows 3.1 SDK*):

The INFO List Chunk

The "INFO" list is a registered global form type that can store information that helps identify the contents of the chunk. This information is useful but does not affect the way a program interprets the file; examples are copyright information and comments. An "INFO" list is a "LIST" chunk with list type "INFO." The following example shows a sample "INFO" list chunk:

```
LIST('INFO'   INAM("Two Trees"Z)
          ICMT("A picture for the opening screen"Z) )
```

An "INFO" list should contain only the following chunks. New chunks may be defined, but an application should ignore any chunk it doesn't understand. The chunks listed below may only appear in an "INFO" list. Each chunk contains a ZSTR, or null-terminated text string.

Chunk ID	Description
IARL	*Archival Location*. Indicates where the subject of the file is archived.
IART	*Artist*. Lists the artist of the original subject of the file; for example, "Michaelangelo."
ICMS	*Commissioned*. Lists the name of the person or organization that commissioned the subject of the file; for example, "Pope Julian II."
ICMT	*Comments*. Provides general comments about the file or the subject of the file. If the comment is several sentences long, end each sentence with

	a period. Do not include new-line characters.
ICOP	*Copyright*. Records the copyright information for the file; for example, "Copyright Encyclopedia International 1991." If there are multiple copyrights, separate them by a semicolon followed by a space.
ICRD	*Creation date*. Specifies the date the subject of the file was created. List dates in year-month-day format, padding one-digit months and days with a zero on the left; for example, "1553-05-03" for May 3, 1553.
ICRP	*Cropped*. Describes whether an image has been cropped and, if so, how it was cropped; for example, "lower-right corner."
IDIM	*Dimensions*. Specifies the size of the original subject of the file; for example, "8.5 in h, 11 in w."
IDPI	*Dots Per Inch*. Stores dots per inch setting of the digitizer used to produce the file, such as "300."
IENG	*Engineer*. Stores the name of the engineer who worked on the file. If there are multiple engineers, separate the names by a semicolon and a blank; for example, "Smith, John; Adams, Joe."
IGNR	*Genre*. Describes the original work, such as "landscape," "portrait," "still life," etc.
IKEY	*Keywords*. Provides a list of keywords that refer to the file or subject of the file. Separate multiple keywords with a semicolon and a blank; for example, "Seattle; aerial view; scenery."
ILGT	*Lightness*. Describes the changes in lightness settings on the digitizer required to produce the file. Note that the format of this information depends on hardware used.
IMED	*Medium*. Describes the original subject of the file, such as "computer image," "drawing," "lithograph," and so on.
INAM	*Name*. Stores the title of the subject of the file, such as "Seattle From Above."
IPLT	*Palette Setting*. Specifies the number of colors requested when digitizing an image, such as "256."
IPRD	*Product*. Specifies the name of the title the file was originally intended for, such as "Encyclopedia of Pacific Northwest Geography."
ISBJ	*Subject*. Describes the contents of the file, such as "Aerial view of Seattle."
ISFT	*Software*. Identifies the name of the software package used to create the file, such as "Microsoft WaveEdit."

ISHP	*Sharpness*. Identifies the changes in sharpness for the digitizer required to produce the file (the format depends on the hardware used).
ISRC	*Source*. Identifies the name of the person or organization who supplied the original subject of the file; for example, "Trey Research."
ISRF	*Source Form*. Identifies the original form of the material that was digitized, such as "slide," "paper," "map," and so on. This is not necessarily the same as IMED.
ITCH	*Technician*. Identifies the technician who digitized the subject file; for example, "Smith, John."

Microsoft-Defined Tombstone Data

6.0 Other File Issues

Issues

Continuation Over Multiple Files

With restrictions to the maximum file format and limitations on the maximum size of disk partitions, it may be desirable to implement a mechanism that specifies a linkage across multiple files. It will certainly be desired that the data of one stream of video can be captured across multiple files. For the moment, this will be external to the AVI file.

Minimized Time to Open Files

tbd

Simultaneous Reading and Writing / Multiple Access

tbd

Issues

7.0 MJPG Motion JPEG DIB Extensions

Issues

ISO vs. Non ISO Formats

For interoperability between various hardware and software codecs, it is not sufficient that the file format be standardized; the contents of the video data chunks must be standardized. In order to comply with industry standards, the AVI MJPG specification and OMF 2.0 file format specifications, the standard ISO Motion JPEG specification is proposed, with certain additions to support efficient identification of the interleaved fields in the frame.

Non-ISO formats could be used, provided that they provide the same outward appearance as the ISO data chunk. This would require different codecs to decompress and compress the data.

8-bit vs. 10-bit YUV

This document currently addresses 8-bit 4:2:2 YUV as per the Microsoft specification.

Video vs. Bitmap

Issues between the fact that a bitmap is a pure rectangle, while video starts with half lines; Motion JPEG compresses on multiple of 8 lines, so what is the frame of video that is compressed vs. the real frame.

See the section of the Video Properties Header in Section 5.0.

Proposed Data Chunk Format

The data chunk format is defined as per the Microsoft *JPEG DIB FORMAT* Technical Note, Type 2: Motion JPEG DIB.

From the DIB spec:

Type 2: Motion JPEG

Motion JPEG DIBs shall accommodate interchange formats which satisfy the "General sequential and progressive syntax" (ISO 10918 Part 1, Annex B, Para. B.2). A set of images of this type with compatible parameters can be placed in an AVI file to describe a motion sequence. Frame headers for these DIBs shall be limited to those specified in Para B.2.2 of the cited Annex B. These types are SOF0, SOF1, SOF2, SOF3, SOF9, SOF10 and SOF11. Of the types accommodated, this specification provides implementation only for the Baseline Sequential DCT (SOF0).

This DIB type contains incomplete JPEG data (Abbreviated Format per ISO 10918) and is not intended for stand-alone single image frame disk files. It may be used within RIFF files and other contexts where it is appropriate to:

a. Decode an image without supplying the associated JPEG Huffman tables. This presumes the codec has been properly pre-initialized prior to image decode.

b. Request encoder output of compressed image data absent embedded Huffman Tables.

All motion JPEG data will use YCbCr encoding.

In an AVI sequence all JPEG frames will be key frames as this ensures that within the AVI and Video for Windows architecture all frames will be directly and independently addressable.

For optimal size and speed during playback of an AVI file the Huffman data used by motion JPEG will be fixed and defined by this document. This will make the individual frames of every motion sequence smaller and more efficient to play back. Also as all sequences of motion images use the same Huffman data and color space it is much more likely that motion data can be directly exchanged without re-compression. A definition of the Huffman data will be provided in MMREG.H (which is listed at the end of this document) as a byte string which can be concatenated onto the start of a motion JPEG image to form a valid still JPEG image-

```
MJPGDHTSeg = { X'FF', DHT, length, JPEG Huffman table
parameters }
```

Q-table data is present and may vary in every frame of a motion sequence to permit control over the bandwidth of sequences that contain bursts of frames of varying levels of complexity. The restart interval used during the compression process may also vary for every frame.

Only the interleaved form of YCrCb images is supported for motion JPEG data. This implies that only one SOS segment will be present in any particular motion JPEG image.

The following applies (again from MS DIB spec):

As in the JPEG DIB format the JPEG stream syntax is used for the image data with the following constraints. The following JPEG marker codes SOI, DRI, DQT, SOF0, SOS and EOI are expected (mandatory) in the image data chunk, and the constrained values shown in the example below are mandatory for the image data within the AVI stream.

Any parameters in the SOF0 (frame) and SOS (start of scan) headers that are duplicated in the `BITMAPINFOHEADER` for JPEG must be the same. This would include Sample Precision, subsampling, number of components (as implied by `JPEGColorSpaceID`), etc. The number of lines and samples per lines in the SOF0 segment and the width and height defined in the format chunk must match the main AVI header width and height values. All of these values are expected to remain the same for every image data chunk in the AVI sequence.

Within the image data chunk two JPEG segments beginning with the SOI marker and ending with the EOI marker are allowed to accommodate field-interleaved streams. There is an APP0 marker immediately following the SOI marker that contains information about the video image. Specifically, this allows the identification of the two fields that comprise an interleaved frame. This APP0 marker is expected to have the first 4 bytes following the length bytes set to the characters 'A', 'V', 'I', '1'. The next byte indicates which field the JPEG data was compressed from and has an expected value of 0x01 for the FIRST JPEG data segment and 0x02 for the SECOND segment, indicating the FIRST and SECOND fields respectively. If the stream is not field interleaved then this value will be 0x00 and there will only be one JPEG segment. The remaining seven bytes are expected to be set to 0 and will be ignored by the codec.

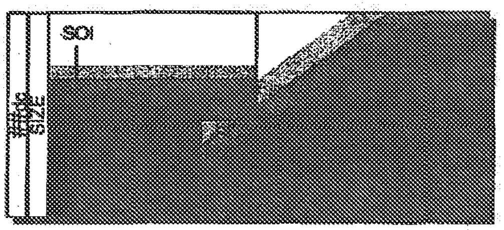
If a codec cannot handle the interleaved fields, the codec will use only the FIRST field and will replicate the lines as necessary to provide an image that conforms to the image size defined in the main AVI header. Conversely if a capture system only accesses a single field of each source frame only a single field image may be present in a JPEG stream. This implies that the single field data should be used as the source of both fields by a decompressor that wishes to process full-interlaced data.

And a new proposal:

The APP0 marker will have an added use beside the field polarity and will also specify the size of the current field (entirely, from SOI to EOI), as well as any padding required. This is to allow applications and codecs to avoid being forced to scan for the second SOI-EOI pair. The size information in the APP0 marker does not include any information about a second field in the ##dc chunk. This way, fields can be copied as integral units without having to have the contained data modified for the presence and size of neighbouring fields.

The byte following the polarity will be reserved for future expansion. The remaining 8 bytes of this marker will be used to store two sizes values. The first value specifies the length in bytes of the field (entirely, from SOI to EOI) including any padding after the EOI (which should be small) and the second value specifies the size of the same field excluding any padding after the EOI. Both size values will be stored in the usual ISO JPEG fashion where bytes are stored in decreasing order with the most significant byte being first. The length of this marker was previously 14 and is now set to 16.

MJPG DIB RIFF chunk (##dc)



An example MJPG frame with two interleaved fields

▨ First field (SOI EOI pair)
▨ Second field (SOI EOI pair)
▩ Padding

MJPEG Baseline DCT - YCbCr DIB Map

A Baseline DCT - YCbCr will now therefore have the following look :

```
SOI  (xFFD8)
APP0 (xFFE0),
     Length (16),
     "AVI1", Polarity, 0, FieldSize,
FieldSizeLessPadding.
DRI  (xFFDD),
     Length (4),
     Restart interval.
DQT  (xFFDB),
     Length (132),
     Precision (0), Table ID (0),
     DQT data [64] for table 0,
     Precision (0), Table ID (1),
     DQT data [64] for table 1.
SOF0 (xFFC0),
     Length (17),
     Sample Precision (8),
     Number of lines (biHeight : multiple of 8),
     Sample per line (biWidth : multiple of 16),
     Number of components (3)
     ID of 1st component (1),
     Sampling ratio of 1st component (H,V),
     Q-table ID of 1st component (0),
     ID of 2nd component (2),
     Sampling ratio of 2nd component (H,V),
     Q table ID of 2nd component (1),
     ID of 3rd component (3),
     Sampling ratio of 3rd component (H,V),
     Q table ID of 3rd component (1).
SOS  (xFFDA),
     Length (12),
     Number of components (3)
     ID of 1st component (1),
     DC and AC Huffman ID for 1st component (0,0),
     ID of 2nd component (2),
       DC and AC Huffman ID for 2nd component (1,1),
       ID of 3rd component (3),
       DC and AC Huffman ID for 3rd component (1,1),
       Start of spectral (0),
       End of spectral (63),
       Successive approximation bit position, high and
  low (0,0).
  *** IMAGE DATA ***
  There could be some Restart markers.
  Values are from xFFD0 to xFFD7.
  EOI (xFFD9).
```

Be sure to have a single DQT as the previous ones will be overridden.

The order of the markers may affect some codecs.

IDs for the 3 component as recommended by the Microsoft DIB specification are 1,2 and 3.

Sector Alignment and Padding

The ##dc chunks may be sector aligned if necessary. As such, a padding chunk would be inserted following the ##dc to preserve the RIFF format. Padding between fields should be discouraged; however the ISO specification allows padding if the data contains 0xFF for each padded byte.

Proposed Data Chunk Format

8.0 OpenDML File Format Certification Procedures

Concurrently Develop a Test Suite to measure compatibility

Test older Codecs with new AVI files.

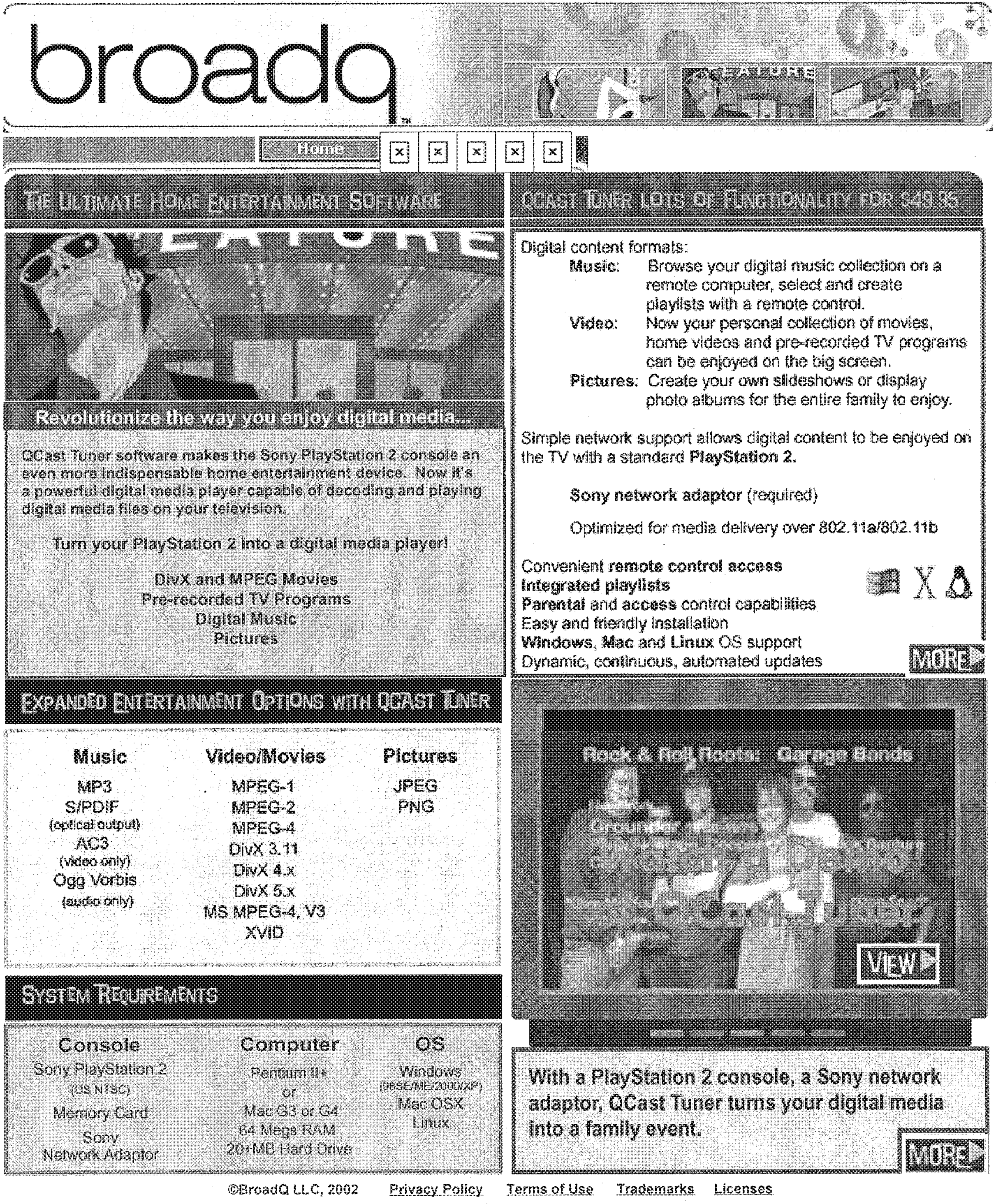
Test old AVI files with new Codecs.

Circulate all files through industry (i.e., everybody tests everybody else's files).

File checker routine to verify AVI format. Program may grow to include test decompressions using software Codec.

Test using MCIAVI with standard codecs.

Go to MS porting labs for specific video file format / Codec test week.



THE ULTIMATE HOME ENTERTAINMENT SOFTWARE

Revolutionize the way you enjoy digital media...

QCast Tuner software makes the Sony PlayStation 2 console an even more indispensable home entertainment device. Now it's a powerful digital media player capable of decoding and playing digital media files on your television.

Turn your PlayStation 2 into a digital media player!

DivX and MPEG Movies
Pre-recorded TV Programs
Digital Music
Pictures

QCAST TUNER LOTS OF FUNCTIONALITY FOR $49.95

Digital content formats:

Music:	Browse your digital music collection on a remote computer, select and create playlists with a remote control.
Video:	Now your personal collection of movies, home videos and pre-recorded TV programs can be enjoyed on the big screen.
Pictures:	Create your own slideshows or display photo albums for the entire family to enjoy.

Simple network support allows digital content to be enjoyed on the TV with a standard **PlayStation 2.**

Sony network adaptor (required)

Optimized for media delivery over 802.11a/802.11b

Convenient **remote control access**
Integrated playlists
Parental and **access** control capabilities
Easy and friendly installation
Windows, Mac and **Linux** OS support
Dynamic, continuous, automated updates

EXPANDED ENTERTAINMENT OPTIONS WITH QCAST TUNER

Music	Video/Movies	Pictures
MP3	MPEG-1	JPEG
S/PDIF (optical output)	MPEG-2	PNG
AC3 (video only)	MPEG-4	
Ogg Vorbis (audio only)	DivX 3.11	
	DivX 4.x	
	DivX 5.x	
	MS MPEG-4, V3	
	XVID	

With a PlayStation 2 console, a Sony network adaptor, QCast Tuner turns your digital media into a family event.

SYSTEM REQUIREMENTS

Console	Computer	OS
Sony PlayStation 2 (US NTSC)	Pentium II+	Windows (98SE/ME/2000/XP)
Memory Card	or	Mac OSX
Sony Network Adaptor	Mac G3 or G4	Linux
	64 Megs RAM	
	20+MB Hard Drive	

©BroadQ LLC, 2002 Privacy Policy Terms of Use Trademarks Licenses

PROTECTING DIGITAL CONTENT USING CLOAKWARE CODE TRANSFORMATION TECHNOLOGY

VERSION 1.2
MAY 2002

SUMMARY

Broadband networks and media, coupled with open computing platforms, offer tremendous potential for the distribution and delivery of digital content. They offer content providers cost-effective and far reaching distribution channels, and offer users convenient access to content.

One of the challenges that must be overcome to enable this market is to ensure that digital content is used only by authorized users, in ways that the content provider intended. Digital content protection systems are intended to confine the use of content to authorized users and uses. These systems rely on software resident on the user's PC or open device to enforce the interests of content owners.

Software deployed on open platforms is susceptible to tampering and reverse engineering attacks that can by easily bypass a content protection system. Cloakware has developed a new security technology based on code transformations that converts ordinary software into a tamper-resistant form. The technology offers this tamper resistance with the convenience and cost-effectiveness of being entirely software based. It is a critical enabling technology for the distribution and delivery of digital content to open platforms via the Internet or other broadband media.

© Cloakware Corporation, 2002

cloakware

INTRODUCTION

Broadband networks and media, coupled with open computing platforms, offer tremendous potential for the distribution and delivery of digital content. They offer content providers cost-effective and far reaching distribution channels, and offer users convenient access to content.

The success of the Internet as a distribution channel for digital content will depend not only on providing users with convenient and cost effective access to digital content, but also on providing assurances to content providers that their copyrights will be enforced. Ensuring that digital content is used only by authorized users, in ways that the content provider intended, is one of the biggest challenges facing this market.

Digital content protection (DCP) systems provide a technological answer to enforcing the copyrights of content owners. They are intended to confine the use of content to authorized users and uses. These systems must rely on software agents that can be downloaded to user's PC or open device such as a content player, and be trusted to enforce the interests of content owners. Software deployed on open platforms is susceptible to tampering and reverse engineering attacks – attacks that can by easily bypass a content protection system. Since these software agents will be deployed in hostile environments, they must be made tamper-resistant.

Cloakware has developed a revolutionary new security technology that converts ordinary software into a tamper-resistant form. Based on code transformations, the technology provides tamper resistance with the convenience and cost-effectiveness of being entirely software based. Cloakware is working with vendors to embed this technology into their digital content protection systems. It is a critical enabling technology for the distribution and delivery of digital content to open platforms via the Internet or other broadband media.

DIGITAL CONTENT PROTECTION

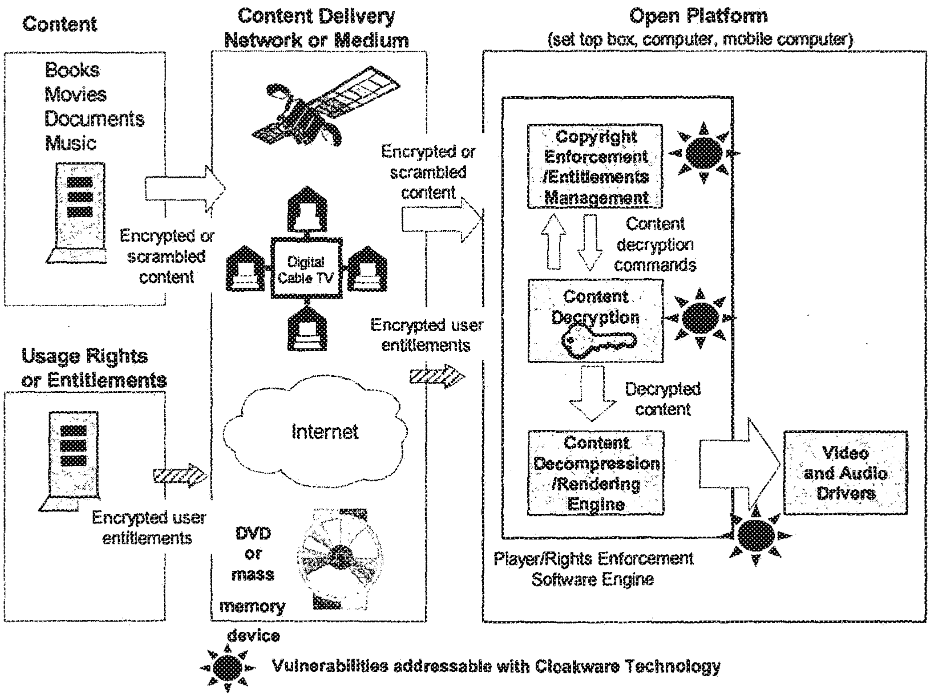
The Internet offers corporations and content companies an efficient distribution channel for digital content, however, copyright enforcement and the protection of corporate information remains a fundamental challenge. Unprotected digital content can be easily copied and broadcast to hundreds of millions of Internet users, as Napster demonstrated.

Another market trend that makes content more vulnerable is the move to open platforms for content playback. This includes content playback on PCs, mobile phones, PDAs and devices such as set-top boxes, which are becoming more general computing platforms.

With over $200 billion in annual content sales at stake, the opportunity is enormous for technologies and systems that provide content owners with a mechanism to enforce usage rights for digital content.

Figure 1 provides a general illustration of how digital content is distributed to open platforms and the key functions required to protect that content against unauthorized use. This architecture is generic to all digital content protection schemes, although not all functions have been implemented in existing systems (for example, the current DVD standard does not include usage rights or entitlements).

Figure 1: Digital Content Protection – General System Architecture


Content
Books
Movies
Documents
Music
Encrypted or scrambled content
Usage Rights or Entitlements
Encrypted user entitlements
Content Delivery Network or Medium
Digital Cable TV
Internet
DVD or mass memory device
Encrypted or scrambled content
Encrypted user entitlements
Open Platform
(set top box, computer, mobile computer)
Copyright Enforcement /Entitlements Management
Content decryption commands
Content Decryption
Decrypted content
Content Decompression /Rendering Engine
Video and Audio Drivers
Player/Rights Enforcement Software Engine
Vulnerabilities addressable with Cloakware Technology

The protected digital content may be consumer oriented (such as books, music and movies) or enterprise oriented (such as documents). Content may be distributed via satellite network (digital TV), cable networks (digital video), the Internet or a mass memory device such as a DVD or flash memory card.

To protect digital content against unauthorized use, it must be distributed to users in encrypted or scrambled form. User entitlements, which define what a user may do with content, must also be distributed in a similar form. When a user attempts to access protected digital content, a software rendering and access control engine is used to determine whether the user is entitled to do what they are trying to do (play, forward, copy, etc.). This may be based on factors such as being a valid subscriber to a satellite TV service, or confirmation that a user is an employee of company XYZ and that they are entitled to view certain content (authentication and authorization). The decision is made by the *copyright enforcement/entitlements management system* as to whether the user can do what they request. Once that decision is made, it sends the appropriate commands to the *content decryption* engine.

There are three key weaknesses to this problem that can be addressed using Cloakware code transformation technology:

- Protecting the copyright enforcement/entitlements management function against tampering. For example, a hacker might change the software from:

  "If CEO then give general access to content"

  to

  "If anyone then give general access to the content"
- Protecting the content decryption key against reverse engineering attacks or tampering attacks designed to separate it from the software player. For example, the DVD Content Scrambling System (CSS) was subjected to this type of attack in 2000.
- Protecting attacks to spool unprotected and decompressed content to a storage device or to the Internet for rebroadcast to the general Internet population.

The alternative to Cloakware code protection technology is to implement these functions in hardware on closed platforms, or to use some other form of tamper resistant software. The third problem indicated in Figure 1 is easier to address for some content types such as video, books and documents, due to the large size of the decompressed or rendered content. For content types such as music, it is difficult to prevent this type of attack.

These attacks on digital content protection applications cannot be protected with cryptography alone. Cryptographic protocols generally assume the user can be trusted (i.e., not to act in a hostile manner). In the case of digital content protection, the user can't be assumed to be trusted and may in fact attempt to modify the software used to protect digital content.

MARKET DYNAMICS

The market for digital content protection is varied. Although some markets are just emerging, others can immediately take advantage of the benefits of code transformation technology. For example, the content protection market for digital video delivered via satellite and cable system is well developed, with content protection being provided by Conditional Access systems. Companies such as NDS and Philips provide these, primarily via smart cards. Smart cards are used in set top boxes to store and decipher user entitlements, and they are frequently hacked. Once a gray market for the hacked smart cards has developed, the direct broadcast satellite provider has no alternative but to change the installed base of cards at a cost that can exceed $100 million. Code protection technology offers a software-only alternative that can be "refreshed" online easily in the event of an attack.

In the video streaming market, several players that provide products for satellite and digital cable are looking to leverage their Conditional Access systems. Since these systems are targeted at PCs, they can no longer rely on smart cards to decipher and store user entitlements. Code protection technology provides the same substitute for smart cards in these video streaming applications and markets.

Another market segment that could benefit greatly is that of content privacy. The system architecture used to manage content privacy is almost identical to Figure 1, and has the same security challenges. The same challenges addressed by code transformation technology. The difference is content can originate from an enterprise or a single end user. For example, a user may wish to send an email or document to a group of employees within a company, and specify that the email or document cannot be forwarded or read by other users. This can only be enforced using code transformations and the appropriate application on the user's computing platform.

DCP SECURITY MODEL

The security model for digital content protection differs substantially from the typical Internet security model, which relies on software operating in trusted environments. For example, companies rely on web servers and applications for e-commerce because they deploy these in trusted environments on trusted servers. Hostile users or hackers are prevented from accessing the trusted software both physically (by locked doors) or electronically (by firewalls). Hence, they cannot change the behavior of the trusted software to suit their purpose.

Recent research in security has also focused on the protecting benign hosts against malicious or virus-infested software that may be downloaded from the Internet. For example, the Java sandbox was developed specifically to address the risk of malicious software downloaded to the user's computer. User's can trust the downloaded software since the Java sandbox protects the user's computer against malicious or virus-infested code.

Digital content protection systems require a different security model. Content providers must deliver content across a hostile network to a user and host platform that cannot be trusted. To control how that content is used and by whom, digital content protection systems must rely on a software enforcement agent that executes in a hostile environment.

The trusted agent must represent and enforce the interests of the content owner, including the terms and conditions under which the content may be used, and by whom, in an environment where the user has complete control and access to the hardware and software that provides access to the content. Users must also be assumed to have an unlimited amount of time and resources to attack and bypass any content protections mechanisms.

This is a unique security model, requiring a unique set of technology.

Traditional Assumptions Won't Work

Historically, cryptography adopted the security model where Alice sends data to Bob, and Eve tries to attack the communication. This model assumes that both Alice and Bob can be trusted. For example, they can be trusted to maintain the secrecy of the cryptographic key(s) used to decrypt messages sent to each other, because they have no incentive to disclose that information to other individuals.

A problem arises when this thinking is applied to digital content protection.

First, digital content protection systems must rely on cryptographic keys to decrypt streamed or persistent (stored locally) content. However, the possibility must be considered that Bob could be a hostile party, and cannot be trusted to maintain the secrecy of the cryptographic keys or to use those keys in a manner that supports the interests of Alice (the content provider). Since Bob cannot be trusted, a trusted agent must be introduced and be present on Bob's computer or player to decrypt the content in a manner that reflects Bob's authorization to access or use that content.

Second, cryptography protects data, not the trusted agent. The trusted agent must withstand focused reverse engineering attacks to discover the cryptographic keys, and in the case of the software, tampering attacks to change the behavior of the trusted agent in favor of the attacker. Digital signatures could be used to ensure the integrity of a trusted software agent, but again we can't rely on Bob to verify the digital signature and the integrity of the agent. If we rely on a trusted software agent to verify its own integrity, the software again is exposed to tampering attacks to eliminate this check.

The failure of the DVD (Digital Video Disk) Forum's CSS (Content Security System) and recent attacks such as on the player used to distribute Stephen King's latest e-book were the result of cryptographic technologies alone being applied to protect digital content in a hostile environment.

While cryptography is an important element of DCP, full content protection will hinge on the ability of content providers and distributors to develop confidence in these trusted agents on users' computers or content players. To achieve this, the trusted agents must be made able to withstand focused tampering and reverse engineering attacks.

REQUIREMENTS FOR TRUSTED SOFTWARE

Trusted software agents can be deployed and upgraded across the Internet at relatively low cost, avoiding problems with installed base compatibility and the time and cost to deploy a hardware solution.

While some may consider a software-based approach to security to be less effective than one delivered in hardware, trusted software agents offer the advantage of easy renewability. As new attacks are found, they can be countered with advances in architecture, and with improvements to the technologies used to implement them. Neither hardware nor software will withstand a focused attack lasting months or years, so digital content protection security models cannot be predicated on this assumption.

Trusted software agents also offer greater user convenience since they can be readily deployed on existing player devices without requiring the user to upgrade their player or purchase a new player.

Trusted software incorporated in digital content protection solutions must have the following properties:

- **Resistance to Reverse Engineering Attacks:** Trusted software must protect cryptographic keys used for content decryption, against extraction through attacks that rely on static analysis or tracing the execution of code.

  Simple code obfuscation technologies do not address this requirement; these typically obfuscate software through symbol renaming, stripping out debug information and other obfuscation techniques that are easily defeated.

- **Resistance to Tampering Attacks:** Trusted software must be resistant to tampering so that functionality that is critical to the content provider cannot be bypassed or changed.

  Standard hacking techniques, such as modifying crucial branch instructions, are easy to apply to these software programs. In the early days of the PC, computer game manufacturers made many efforts to prevent hackers from bypassing their copy protection code but all these were readily defeated by hackers using these techniques.

  Code obfuscation tools do nothing to protect software against tampering; a user that can reverse engineer the software can also, with relative ease, make useful changes to the software.

- **Portability:** Portability is another key requirement for trusted software. With the rise in new consumer devices that will be used to access the Internet and digital content, the X86 architecture is unlikely to be the dominant platform for the viewing and use of digital content.

  A number of techniques to render software tamper-resistant have been implemented or proposed. In general, these rely on encryption and decryption of the binary file, or manipulation of the binary to make it very difficult to reverse engineer. These "wrapper" technologies are, in general, limited in their portability.

CLOAKWARE CODE TRANSFORMATION TECHNOLOGY

Cloakware has developed a revolutionary new security technology that converts normal software into a tamper-resistant form. Cloakware code transformation technology can be applied to trusted software agents used in digital content protection systems to harden them against focused attacks.

This technology is based on a family of compiler-derived techniques and the innovative application of mathematical and cryptographic principles to prevent reverse engineering and tampering with software.

This conversion of software to a tamper-resistant form is accomplished through a process of "cloaking" crucial pieces of the software. The cloaking process, which occurs at source level, resists traditional techniques for key extraction - cryptographic keys are mapped using non-invertible functions in the cloaked code and never appear as "clear text" in the protected software during execution.

Tamper-resistant properties are achieved by injecting a measured and appropriate degree of entropy into the code. If the cloaked code is tampered with, the code will cascade into failure during execution. This means the attacker cannot make useful changes (from the hacker's perspective) to the code.

Since the technology works at source code level, it offers portability advantages over tamper resistant technologies that operate at binary level.

Cloakware is working with major digital content protection system providers and content providers to implement our code transformation technology into their products. This technology offers the convenience of a software-only offering, with a high-level of resistance to tampering and reverse engineering attacks. These are essential to the digital content protection security model and ultimately to the success of digital content protection and the use of the Internet for the distribution of digital content.

CONCLUSION

The success of the Internet as a distribution channel for digital content will depend not only on providing users with convenient and cost effective access to digital content, but also on providing assurances to content providers that their copyrights will be enforced.

Digital Rights Management (digital content protection) systems offer a technological means of enforcing the copyrights of content owners. Trustworthiness and convenience will be crucial to the broad acceptance of digital right management (digital content protection) across the Internet.

The digital content protection security model depends on trusted software agents that can be downloaded to user's content players or PCs and be trusted to enforce the interests of content owners. These trusted software agents must be made tamper-resistant to withstand attacks to extract cryptographic keys and to change the functionality of the trusted software agents.

Cloakware has developed a revolutionary new code transformation technology that converts ordinary software into tamper-resistant software. The technology offers digital content protection system and service providers, tamper resistance with the convenience and cost-effectiveness of a portable, software-only product.

Cloakware is currently working with leading digital content protection vendors to embed this technology into their DCP systems and to manufacture renewable trusted software agents. The application of Cloakware's security technology to digital content protection offers content providers assurances that their copyrights will be enforced while offering content users the convenience of a portable, software-only solution.

© Cloakware Corporation, 2002

This document is provided "as is" with no warranties, expressed or implied, including but not limited to any implied warranty of merchantability, fitness for a particular purpose, or freedom from infringement. Cloakware Corporation may have patents or pending patent applications, trademarks, copyrights or other intellectual property rights that relate to the described subject matter. The furnishing of this document does not provide any license, expressed or implied, by estoppels or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights. Cloakware Corporation assumes no responsibility for error or omissions in this document; nor does Cloakware Corporation make any commitment to update the information contained herein. This document is subject to change without notice.

Safari Power Saver
Click to Start Flash

CYBERSECURITY theStand
>>> READ THE REPORT
SPONSORED BY NORTHROP GRUMMAN AND FIREEYE
What's the password?
What's the password?

Tool speeds info to vehicles

- Jul 25, 1999

CellPort Labs Inc. has developed a tool for law enforcement agencies, ambulance corps and other groups that need to distribute information quickly and securely across a fleet of vehicles.

The CP2100 mobile network server supports two-way data communications between a vehicle-based unit and the Internet, a corporate network or an intranet. The system also provides data communications between vehicles.

The product eliminates a lot of the hassles associated with remote-access technology by simply turning a vehicle system into a node on the Internet, company officials said. Such an approach makes it easy to give vehicles access to centralized applications and to give central offices access to information on the vehicles.

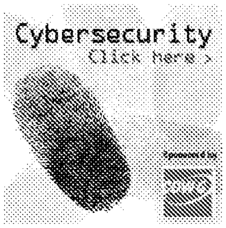
Boulder, Colo.-based CellPort, a developer of wireless connectivity products for cellular phones, has been contacted by officials from law enforcement and other public-safety agencies that are looking to better monitor their fleet of vehicles, said Tom Kubancik, the company's director of business development.

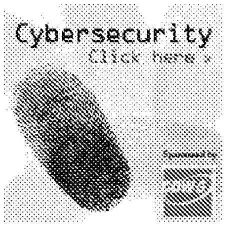
The company is in discussions with several state transportation departments about using the technology to assist their emergency response vehicles on major highways. The product can be customized to monitor road temperature, air temperature and vehicle speeds. "Without any human interaction, the server grabs all the information and sends it back," he said. "It would allow the [transportation departments] to use their vehicles as probes."

The CP2100 is available directly from the company and starts at $995.

Related Articles

- VA prepared for new employee mobile devices on Oct. 1 09/29/2011
- DOD, FEMA team on Irene recovery 08/29/2011
- Earthquake: Another missed opportunity for telework? 08/24/2011


Cybersecurity
Click here >
Sponsored by


Cybersecurity
Click here >
Sponsored by

HOT TOPICS

2012 Budget
Agency News
Cloud / Virtualization
Contracts and Procurement
CXOs
Defense
Enterprise Architecture
Government 2.0
Health IT
Homeland Security
Management and Workforce
Policy and Funding

Most Popular Articles | Most Emailed Articles

Budget battles reshape debate about military strategy
USAJobs: Lesson or warning?
VA official outlines 'ruthless' reductions in IT budget cut plan
Organizations fail on use of technology tools, ELC keynote says
Air Force urged to move to social-media style communications

Safari Power Saver
Click to Start Flash Plug-in

IBM® i2® National Security and Defense Intelligence is a cost-effective solution that helps organizations generate the intelligence they need to make faster, better-informed decisions for confident and effective actions.
To learn more, Watch this demo ▸ IBM.
LEARN MORE ABOUT WIRELESS SOLUTIONS FOR GOVERNMENT ▸ verizon

Program Management

Security / Cybersecurity

State and Local

Telework

Training and Certification

FCW OPINION

Blogs

Columns

Resource Center

New! Special Report: Infrastructure Optimization

New! Insights Special Report: Health IT

New! The STAND: Teleworking

Special Report: Information Sharing

Special Report: Infrastructure Hardware

The STAND: Cloud Computing

Special Report: Cybersecurity

Special Report: Virtualization and Consolidation

New! The Stand: C4ISR

The STAND: Rugged IT

NASA SEWP IV Contract Guide

Special Report: CHESS

GSA IT Schedule 70 Special Report

Insights Special Report: IT Modernization

The STAND: Information Security

Special Report: Client Computing

2010-2011 Library

Virtualization & Consolidation: Planning, savings & efficiency

Click here

Sponsored by IRON BOW

- Federal 100: Susie Adams 03/28/2011
- DHS buying SBInet-like system for border despite uncertainties, GAO says 03/16/2011

Share this Page

LinkedIn Facebook Twitter Google Slashdot

Reader comments

Please post your comments here. Comments are moderated, so they may not appear immediately after submitting. We will not post comments that we consider abusive or off-topic.

*(optional)*Your Name:

*(optional)*Your Email:

*(optional)*Your Location:

Comment:

Please type the letters/numbers you see above

Submit

MARKETPLACE: Products and Services from our Sponsors

DOI Contractors' Ratings
Complimentary Summary: First-Time Analysis Of DOI Vendors' Past Performance
www.INPUT.com

GSA Schedules 101
Free Whitepaper - Understanding the Greatest Challenges of the GSA Schedule.
www.INPUT.com/GSA-101

2012 State/Local Contracts
Opportunity Overview - $754M in Opps With Key Contacts/Timeframes. Free!
www.INPUT.com/Top-Opps

Buy a Link Now

MARKETPLACE: Products and Services from our Sponsors

DOI Contractors' Ratings
Complimentary Summary: First-Time Analysis Of DOI Vendors' Past Performance
www.INPUT.com

LT E
MI S N-C I ICA
I F

LEARN MORE ABOUT
WIRELESS SOLUTIONS FOR GOVERNMENT ›
verizon

LT
M S NC I A
I F

LEARN MORE ABOUT
WIRELESS SOLUTIONS FOR GOVERNMENT ›
verizon

Related Webcasts

- Why Your Cloud-First Step Should Be IT Service Management
- How to Accelerate Your Enterprise Resource Planning (ERP) Without Replacing It
- DOD's 5 Strategic Initiatives for Cyberspace Operations

Related White Papers

- Building An Assured Network
- Storage Efficiency for Dummies -- eBook Guide
- The STAND: C4ISR

Related Podcasts

- Risk Management
- Integrated Program Management
- Unified Computing: An Agile Platform for Your Next Gen Virtualized Data Center
- Counter the Insider Threat with Security Intelligence

Related Resources

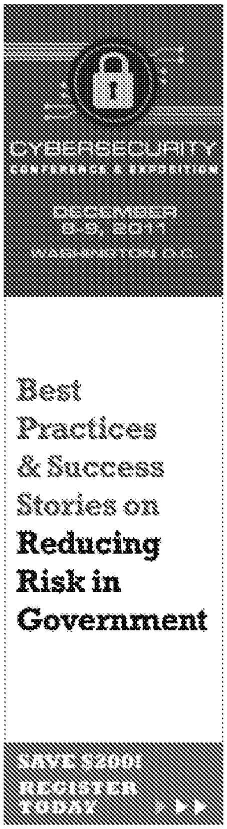
- Mobile movement infiltrates the health IT world
- DOD, VA recalibrate the model for health records
- Telehealth boom could break down distance barriers
- Test
- Four advantages to using CHESS for your hardware, software purchases

Virtualization
& Consolidation:
Planning, savings & efficiency
Click here
Sponsored by
IRON BOW

GSA Schedules 101
Free Whitepaper - Understanding the Greatest Challenges of the GSA Schedule.
www.INPUT.com/GSA-101

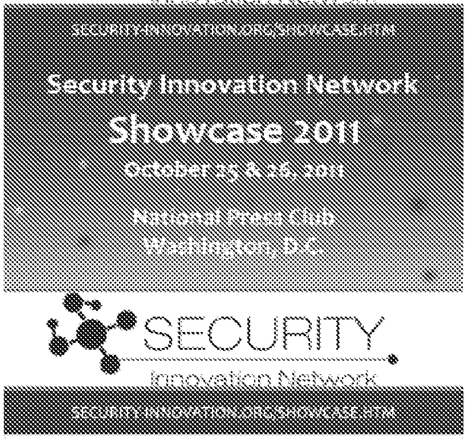
2012 State/Local Contracts
Opportunity Overview - $754M in Opps With Key Contacts/Timeframes. Free!
www.INPUT.com/Top-Opps

Buy a Link Now


CYBERSECURITY
CONFERENCE & EXPOSITION
DECEMBER 8-9, 2011
WASHINGTON D.C.
Best Practices & Success Stories on Reducing Risk in Government
SAVE $200! REGISTER TODAY

SPONSORED BY NORTHROP GRUMMAN AND FIREEYE
theStand
CYBERSECURITY: Rising to the Challenge
READ THE REPORT

Security Innovation Network
Showcase 2011
October 25 & 26, 2011
National Press Club
Washington, D.C.
SECURITY Innovation Network
SECURITY-INNOVATION.ORG/SHOWCASE.HTM


Security Innovation Network
Showcase 2011
October 25 & 26, 2011
National Press Club
Washington, D.C.
SECURITY Innovation Network
SECURITY-INNOVATION.ORG/SHOWCASE.HTM

Editorial Webcasts

Improve Operational Efficiencies---Evolving Approaches for Agency Correspondence

Register Now

Many Federal Government enterprises are focused on programs to consolidate data centers and introduce cloud computing to deliver significant efficiencies to daily operations. While it is important to implement these options for saving valuable resources, it is also possible to gain measurable improvement by reviewing traditional internal processes that also can be optimized to leverage today's technologies, information-sharing opportunities, and cost-reduction objectives. Read more

More Editorial Webcasts

GovCareerNetwork

Featured Jobs

CYBERSECURITY
CONFERENCE & EXPOSITION
DECEMBER
5-8, 2011
WASHINGTON D.C.

Best Practices & Success Stories on **Reducing Risk in Government**

SAVE $200! REGISTER TODAY ▶▶

Governance Analyst IV - Fannie Mae - Washington

Governance Analyst IV - Fannie Mae - Washington

Premier Field Engineer - Microsoft - Washington

Consultant - 774079 Infrastructure/National Security Group - Microsoft - Washington

Senior Consultant - 774071 .NET application development - Microsoft - Washington

Find more great jobs on GovCareerNet

Federal Computer Week eNewsletters

SUBSCRIBE

Subscribe to Newsletters

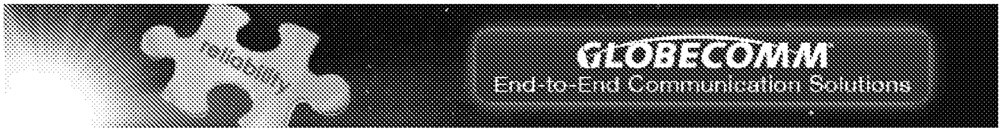
Federal Computer Week's eNewsletters deliver the latest policy and management news to your inbox.

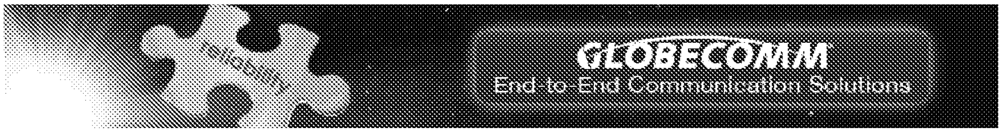
Sa

Cl

SPONSORED BY NORTHROP GRUMMAN AND FIREEYE
theStand
CYBERSECURITY: Rising to the Challenge
>>> READ THE REPORT

CYBERSECURITY
>>> READ THE REPORT
theStand
SPONSORED BY NORTHROP GRUMMAN AND FIREEYE





ABOUT ADVERTISE CONTACT CUSTOM MEDIA EDITORIAL STAFF EVENTS LIST RENTAL PRIVACY POLICY SITEMAP SUBSCRIBE

Defense Systems FCW Federal Employee News Digest FederalSoup Government Career Network GCN Washington Technology

1105 GOVERNMENT
Information Group

8609 Westwood Center Drive, Suite 500
Vienna, VA 22182-2215
703-876-5100

© 1996-2011 1105 Media, Inc. All Rights Reserved.

HTTP Live Streaming Overview

Networking & Internet

2011-04-01

Apple Inc.
© 2011 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

App Store is a service mark of Apple Inc.

Apple, the Apple logo, Apple TV, Final Cut, Final Cut Studio, iPhone, iPod, iPod touch, Mac, Mac OS, QuickTime, and Safari are trademarks of Apple Inc., registered in the United States and other countries.

iPad is a trademark of Apple Inc.

DEC is a trademark of Digital Equipment Corporation.

IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

Intel and Intel Core are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction **Introduction 7**

At a Glance 8
    You Can Send Audio and Video Without Special Server Software 8
    You Can Send Live Streams or Video on Demand, with Optional Encryption 9
Prerequisites 9
See Also 9

Chapter 1 **HTTP Streaming Architecture 11**

Overview 11
Server Components 12
    Media Encoder 12
    Stream Segmenter 13
    File Segmenter 13
Media Segment Files 13
Index Files (Playlists) 13
Distribution Components 14
Client Component 15

Chapter 2 **Using HTTP Live Streaming 17**

Download the Tools 17
    Media Stream Segmenter 17
    Media File Segmenter 17
    Media Stream Validator 17
    Variant Playlist Creator 18
    Metadata Tag Generator 18
Session Types 18
Content Protection 18
Caching and Delivery Protocols 19
Stream Alternates 19
Video Over Cellular Networks 21
Requirements for Apps 21
Failover Protection 22
Adding Timed Metadata 23
Adding Closed Captions 24
Preparing Media for Delivery to iOS-Based Devices 24
Sample Streams 26

Chapter 3 Deploying HTTP Live Streaming 27

Creating an HTML Page 27
Configuring a Web Server 28
Validating Your Streams 28
Serving Key Files Securely Over HTTPS 29

Chapter 4 Frequently Asked Questions 31

Document Revision History 37

Figures, Tables, and Listings

Chapter 1 **HTTP Streaming Architecture 11**

Figure 1-1 A basic configuration 12

Chapter 2 **Using HTTP Live Streaming 17**

Figure 2-1 Stream alternates 20
Table 2-1 Encoder settings for iPhone, iPod touch, iPad, and Apple TV, 16:9 aspect ratio 25
Table 2-2 Encoder settings for iPhone, iPod touch, iPad, and Apple TV, 4:3 aspect ratio 25
Table 2-3 Additional encoder settings for iPad and Apple TV only, 16:9 aspect ratio 25
Table 2-4 Additional encoder settings for iPad and Apple TV only, 4:3 aspect ratio 26

Chapter 3 **Deploying HTTP Live Streaming 27**

Listing 3-1 Serving HTTP Live Streaming in a webpage 27

2011-04-01 | © 2011 Apple Inc. All Rights Reserved.

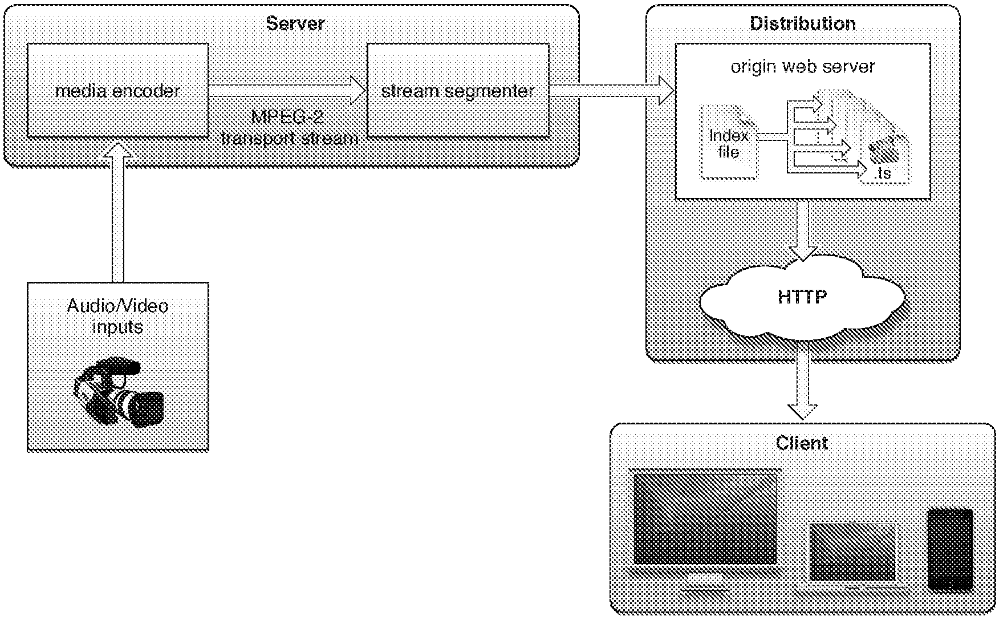
Introduction

If you are interested in any of the following:

- Streaming audio or video to iPhone, iPod touch, iPad, or Apple TV
- Streaming live events without special server software
- Sending video on demand with encryption and authentication

you should learn about HTTP Live Streaming.

HTTP Live Streaming lets you send audio and video over HTTP from an ordinary web server for playback on iOS-based devices—including iPhone, iPad, iPod touch, and Apple TV—and on desktop computers (Mac OS X). HTTP Live Streaming supports both live broadcasts and prerecorded content (video on demand). HTTP Live Streaming supports multiple alternate streams at different bit rates, and the client software can switch streams intelligently as network bandwidth changes. HTTP Live Streaming also provides for media encryption and user authentication over HTTPS, allowing publishers to protect their work.



All devices running iOS 3.0 and later include built-in client software for HTTP Live Streaming. The Safari browser can play HTTP streams within a webpage on iPad and desktop computers, and Safari launches a full-screen media player for HTTP streams on iOS devices with small screens, such as iPhone and iPod touch. Apple TV 2 and later includes an HTTP Live Streaming client.

2011-04-01 | © 2011 Apple Inc. All Rights Reserved.

Important: iPhone and iPad apps that send large amounts of audio or video data over cellular networks are *required* to use HTTP Live Streaming. See "Requirements for Apps" (page 21).

Safari plays HTTP Live streams natively as the source for the `<VIDEO>` tag. Mac OS X developers can use the QTKit and AVFoundation frameworks to create desktop applications that play HTTP Live Streams, and iOS developers can use the MediaPlayer and AVFoundation frameworks to create iOS apps.

Important: Always use the `<video>` tag to embed HTTP Live Streaming. Do not use the `<object>` or `<embed>` tags (except to specify fallback content).

Because it uses HTTP, this kind of streaming is automatically supported by nearly all edge servers, media distributors, caching systems, routers, and firewalls.

Note: Many existing streaming services require specialized servers to distribute content to end users. These servers require specialized skills to set up and maintain, and in a large-scale deployment this can be costly. HTTP Live Streaming avoids this by using standard HTTP to deliver the media. Additionally, HTTP Live Streaming is designed to work seamlessly in conjunction with media distribution networks for large scale operations.

The HTTP Live Streaming specification is an IETF Internet-Draft.

At a Glance

HTTP Live Streaming is a way to send audio and video over HTTP from a web server to client software on the desktop or to iOS-based devices.

You Can Send Audio and Video Without Special Server Software

You can serve HTTP Live Streaming audio and video from an ordinary web server. The client software can be the Safari browser or an app that you've written for iOS or Mac OS X.

HTTP Live Streaming sends audio and video as a series of small files, typically of about 10 seconds duration, called media segment files. An index file, or playlist, gives the clients the URLs of the media segment files. The playlist can be periodically refreshed to accomodate live broadcasts, where media segment files are constantly being produced. You can embed a link to the playlist in a webpage or send it to an app that you've written.

Relevant Chapter: "HTTP Streaming Architecture" (page 11)

You Can Send Live Streams or Video on Demand, with Optional Encryption

For video on demand from prerecorded media, Apple provides a free tool to make media segment files and playlists from MPEG-4 video or QuickTime movies with H.264 video compression, or audio files with AAC or MP3 compression. The playlists and media segment files can be used for video on demand or streaming radio, for example.

For live streams, Apple provides a free tool to make media segment files and playlists from live MPEG-2 transport streams carrying H.264 video, AAC audio, or MP3 audio. There are a number of hardware and software encoders that can create MPEG-2 transport streams carrying MPEG-4 video and AAC audio in real time.

Either of these tools can be instructed to encrypt your media and generate decryption keys. You can use a single key for all your streams, a different key for each stream, or a series of randomly generated keys that change at intervals during a stream. Keys are further protected by the requirement for an initialization vector, which can also be set to change periodically.

Relevant Chapter: "Using HTTP Live Streaming" (page 17)

Prerequisites

You should have a general understanding of common audio and video file formats and be familiar with how web servers and browsers work.

See Also

- *iOS Human Interface Guidelines*—how to design web content for iOS-based devices.
- HTTP Live Streaming protocol—the IETF Internet-Draft of the HTTP Live Streaming specification.

2011-04-01 | © 2011 Apple Inc. All Rights Reserved.

HTTP Streaming Architecture

HTTP Live Streaming allows you to send live or prerecorded audio and video, with support for encryption and authentication, from an ordinary web server to any device running iOS 3.0 or later (including iPad and Apple TV), or any computer with Safari 4.0 or later installed.

Overview

Conceptually, HTTP Live Streaming consists of three parts: the server component, the distribution component, and the client software.

The **server component** is responsible for taking input streams of media and encoding them digitally, encapsulating them in a format suitable for delivery, and preparing the encapsulated media for distribution.

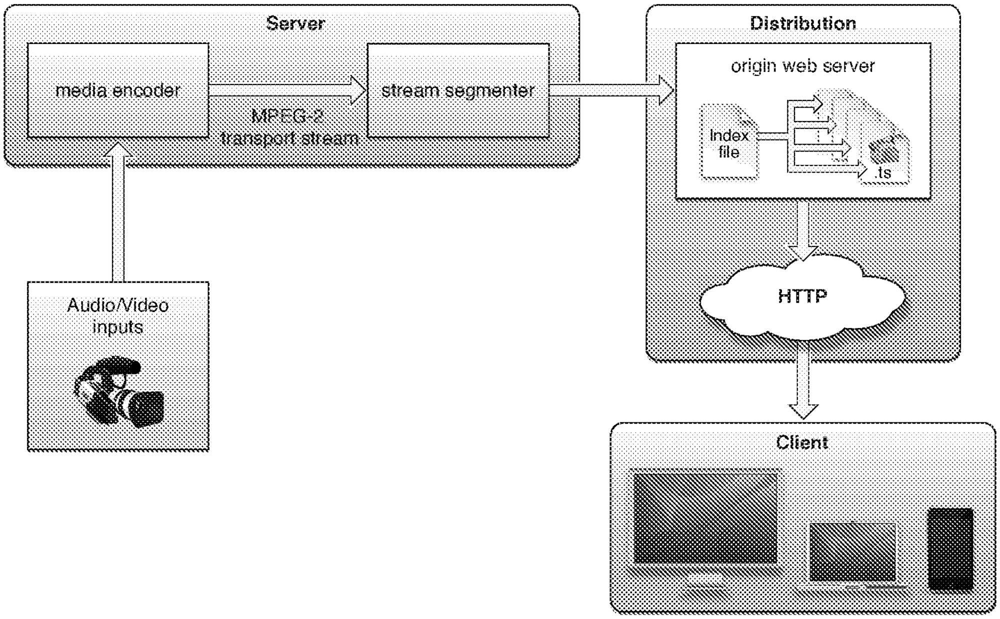
The **distribution component** consists of standard web servers. They are responsible for accepting client requests and delivering prepared media and associated resources to the client. For large-scale distribution, edge networks or other content delivery networks can also be used.

The **client software** is responsible for determining the appropriate media to request, downloading those resources, and then reassembling them so that the media can be presented to the user in a continuous stream. Client software is included on iOS 3.0 and later and computers with Safari 4.0 or later installed.

In a typical configuration, a hardware encoder takes audio-video input, encodes it as H.264 video and AAC audio, and outputs it in an MPEG-2 Transport Stream, which is then broken into a series of short media files by a software stream segmenter. These files are placed on a web server. The segmenter also creates and maintains an index file containing a list of the media files. The URL of the index file is published on the web server. Client software reads the index, then requests the listed media files in order and displays them without any pauses or gaps between segments.

An example of a simple HTTP streaming configuration is shown in "A basic configuration."

Figure 1-1 A basic configuration



Input can be live or from a prerecorded source. It is typically encoded as MPEG-4 (H.264 video and AAC audio) and packaged in an MPEG-2 Transport Stream by off-the-shelf hardware. The MPEG-2 transport stream is then broken into segments and saved as a series of one or more `.ts` media files. This is typically accomplished using a software tool such as the Apple stream segmenter.

Audio-only streams can be a series of MPEG elementary audio files formatted as either AAC with ADTS headers or as MP3.

The segmenter also creates an index file. The index file contains a list of media files. The index file also contains metadata. The index file is an `.M3U8` playlist. The URL of the index file is accessed by clients, which then request the indexed files in sequence.

Server Components

The server requires a media encoder, which can be off-the-shelf hardware, and a way to break the encoded media into segments and save them as files, which can be software such as the media stream segmenter provided by Apple.

Media Encoder

The media encoder takes a real-time signal from an audio-video device, encodes the media, and encapsulates it for transport. Encoding should be set to a format supported by the client device, such as H.264 video and HE-AAC audio. Currently, the supported delivery format is MPEG-2 Transport Streams for audio-video, or MPEG elementary streams for audio-only.

2011-04-01 | © 2011 Apple Inc. All Rights Reserved.