INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG2005/N7573
October 2005, Nice, F

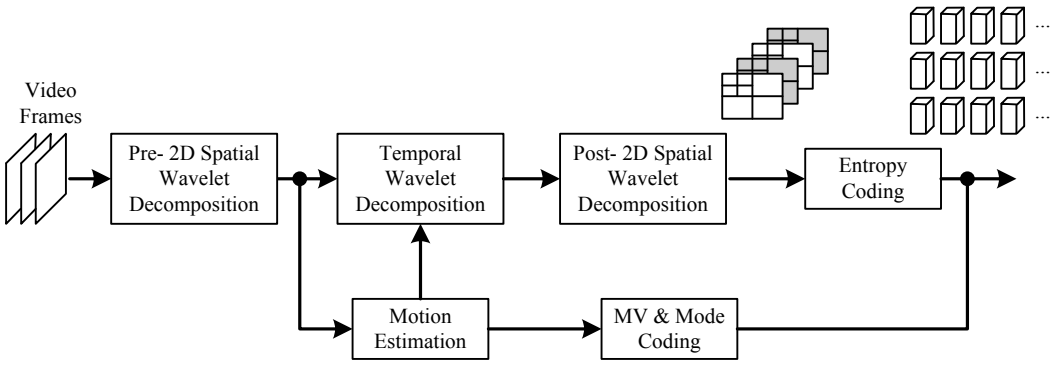
Source	University of Brescia (Italy) – Signals & Communications Laboratory ENST (France) – Signal and Image Processing Laboratory
Title	**Edited version of the document SC 29 N 7334**
Status	Contribution
Sub group	Video Group
Authors	Riccardo Leonardi, Sébastien Brangoulo Contacts: riccardo.leonardi@ing.unibs.it sebastien.brangoulo@enst.fr

Table of contents

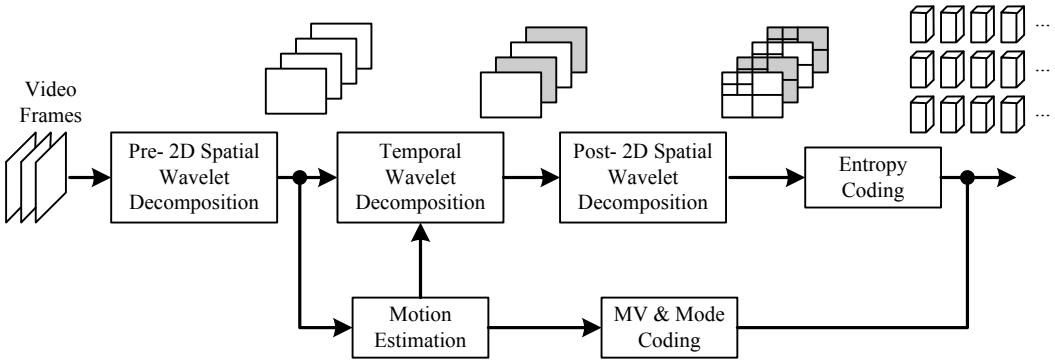
1 Framework

Current 3D wavelet video coding schemes with Motion Compensated Temporal Filtering (MCTF) can be divided into two main categories. The first performs MCTF on the input video sequence directly in the full resolution spatial domain before spatial transform and is often referred to as spatial domain MCTF. The second performs MCTF in wavelet subband domain generated by spatial transform, being often referred to as in-band MCTF. Figure 1(a) is a general framework which can support both of the above two schemes. Firstly, a pre-spatial decomposition can be applied to the input video sequence. Then a multi-level MCTF decomposes the video frames into several temporal subbands, such as temporal highpass subbands and temporal lowpass subbands. After temporal decomposition, a post-spatial decomposition is applied to each temporal subband to further decompose the frames spatially.

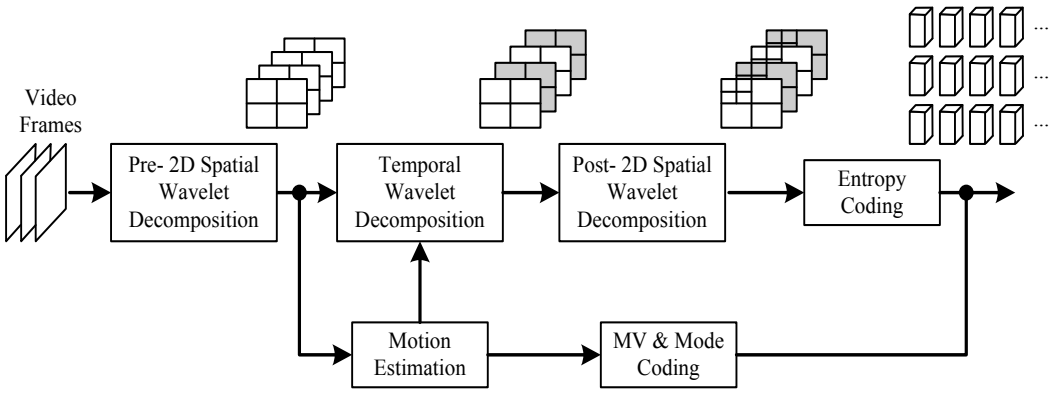
In the framework, the whole spatial decomposition operations for each temporal subband are separated into two parts: pre-spatial decomposition operations and post-spatial decomposition operations. The pre-spatial decomposition can be void for some schemes while non-empty for other schemes. Figure 1(b) shows the case of the T+2D scheme where pre-spatial decomposition is empty. Figure 1(c) shows the case of the 2D+T+2D scheme where pre-spatial decomposition is usually a multi-level dyadic wavelet transform. Depending on the results of pre-spatial decomposition, the temporal decomposition should perform different MCTF operations, either in spatial domain or in subband domain.



(a) The general coding framework



(b) Case for the T+2D scheme (Pre-spatial decomposition is void)



(c) Case for the 2D+T+2D scheme (Pre-spatial decomposition exists)

Figure 1: Framework for 3D wavelet video coding.

A deep analysis on the difference between schemes is here reported.

A simple T+2D scheme acts on the video sequences by applying a temporal decomposition followed by a spatial transform. The main problem arising with this scheme is that the inverse temporal transform is performed on the lower spatial resolution temporal subbands by using the same (scaled) motion field obtained from the higher resolution sequence analysis. Because of the non ideal decimation performed by the low-pass wavelet decomposition, a simply scaled motion field is, in general, not optimal for the low resolution level. This causes a loss in performance and, even if some means are being designed to obtain a better motion field, this is highly dependent on the working rate for the decoding process, and is thus difficult to estimate it in advance at the encoding stage. Furthermore, as the allowed bit-rate for the lower resolution format is generally very restrictive, it is not possible to add corrective measures at this level so as to compensate the problems due to inverse temporal transform.

In order to solve the problem of motion fields at different spatial levels a natural approach has been to consider a 2D+T scheme, where the spatial transform is applied before the temporal one. Unfortunately, this approach suffers from the shift-variant nature of wavelet decomposition, which leads to inefficiency in motion compensated

temporal transforms on the spatial subbands. This problem has found a partial solution in schemes where the motion estimation and compensation take place in an overcomplete (shift-invariant) wavelet domain.

From the above discussion it comes clear that the spatial and temporal wavelet filtering cannot be decoupled because of the motion compensation. As a consequence it is not possible to encode different spatial resolution levels at once, with only one MCTF, and thus both lower and higher resolution sequences must be MCTF filtered.

In this perspective, a possibility for obtaining good performance in terms of bit-rate and scalability is to use an Inter-Scale Prediction scheme. What has been proposed in the literature is to use prediction between the low resolution and the higher one before applying spatio-temporal transform. The low resolution sequence is interpolated and used as prediction for the high resolution sequence. The residual is then filtered both temporally and spatially. This architecture has a clear basis on what have been the first hierarchical representation technique, introduced for images, namely the Laplacian pyramid. So, even if from an intuitive point of view the scheme seems to be well motivated, it has the typical disadvantage of overcomplete transforms, namely that of leading to a full size residual image. This way the information to be encoded as refinement is spread on a high number of coefficients and coding efficiency is hardly achievable.

A 2D+T+2D scheme that combines a layered representation with interband prediction in the MCTF domain appears now as a valid alternative approach. The proposed method called “STool” efficiently combines the idea of prediction between different resolution levels within the framework of spatial and temporal wavelet transforms. Compared with the previous schemes it has several advantages. First of all, the different spatial resolution levels have all undergone an MCTF, which prevents the problems of T+2D schemes. Furthermore, the MCTF are applied before spatial DWT, which solves the problem of 2D+T schemes.

Moreover, the prediction is confined to the same number of transformed coefficients that exist in the lower resolution format. So, there is a clear distinction between the coefficients that are associated to differences in the low-pass bands of high resolution format with respect to the low resolution ones and the coefficients that are associated to higher resolution details. This constitutes an advantage between the prediction schemes based on interpolation in the original sequence domain. Another important advantage is that it is possible to decide which and how many temporal subbands to use in the prediction. So, one can for example discard the temporal high-pass subbands if when a good prediction cannot be achieved for such “quick” details. Alternatively this allows for example a QCIF sequence at 15 fps to be efficiently used as a base for prediction of a 30 fps CIF sequence.

2 Main Modules

2.1 Motion

2.1.1 Modes/Scale

2.1.1.1 Motion Compensation Block Partition

A picture is divided into macroblocks of 16×16 pixels. The macroblock partition patterns like H.264/AVC were adopted. The tree-structured macroblock partitions are applied to motion compensation. As shown in Figure 2, the luma component for each macroblock contains four patterns, namely one 16×16 partition, two 16×8 partitions, two 8×16 partitions or four 8×8 partitions. If the 8×8 partition is chosen, each of the four 8×8 sub-macroblocks may be further divided into 4 patterns, namely one 8×8 sub-partition, two 8×4 partitions, two 4×8 partitions or four 4×4 partitions.

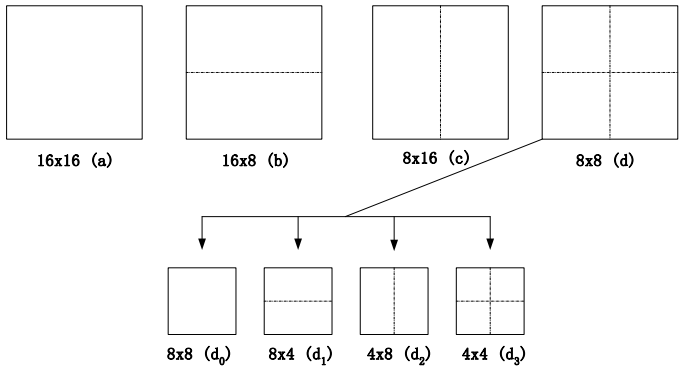


Figure 2: Macroblock and sub-macroblock partitions of the tree-structured motion compensation

In the "T+2D" scheme, if no spatial scalability is supported, the macroblocks are of size 16×16. To support spatial scalability functionality, macroblock size is scaled according to the ratio between the spatial resolution of the decoded video and the original full resolution. For example, when two-level spatial scalability with QCIF and CIF formats is required, the macroblock size is equal to 32×32 for CIF and 16×16 for QCIF decoding respectively; when three-level spatial scalability with QCIF, CIF and 4CIF is required, the macroblock size is 64×64, 32×32 and 16×16 for 4CIF, CIF and QCIF decoding.

2.1.1.2 Interpolation Accuracy

In the default case, quarter pixel interpolation accuracy is used for motion compensation. In the "T+2D" scheme, quarter pixel interpolation accuracy for the lowest spatial resolution is guaranteed and the higher spatial resolution has lower pixel interpolation accuracy to keep the motion compensation effective for each spatial resolution video decoding. For example, when three-level spatial scalability with QCIF, CIF and 4CIF is required, reference pixel interpolation accuracy is quarter, half and integer pixel for QCIF, CIF and 4CIF video decoding respectively.

2.1.2 Modes/MVs Coding

The motion information to be coded for each macroblock includes the mode type and the motion vectors for each partition.

2.1.2.1 Universal Variable Length Coding (UVLC)

Universal Variable Length Coding (UVLC) is adopted for entropy coding. It's codewords can be written in the following compressed form.

$$1$$
$$0\ 1\ x_0$$
$$0\ 0\ 1\ x_0\ x_1$$
$$0\ 0\ 0\ 1\ x_0\ x_1\ x_2$$
$$0\ 0\ 0\ 0\ 1\ x_0\ x_1\ \ x_2\ \ x_3$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

where x_n is equal to the binary value 0 or 1. We will sometimes refer to a codeword with its length in bits (L) and INFO = x_n .. x_1 x_0 . Notice that the number of bits in INFO is L/2 (division by truncation). The codewords are numbered from 0 and upwards. The definition of the numbering is:

Code_number = $2^{L/2}$ + INFO-1 (L/2 uses division with truncation. INFO = 0 when L = 1).

Some of the first code numbers and codewords are written explicitly in the table below. As an example, for the code number 5, L = 5 and INFO = 10 (binary) = 2 (decimal).

2.1.2.2 Mode Type Coding

There are five modes including DIRECT, INTER 16x16, INTER 16x8, INTER 8x16, INTER 8x8 and INTRA. Each partition inter-block has forward, backward and bi-directional prediction direction for temporal 5x3 high-pass subband and only forward direction prediction for Haar temporal high-pass subband. Their corresponding codeword numbers are shown in Table 1 and

Table 2.

Table 1: Connection between mode type and codeword number

Mode Type	Partition Size	5x3		Haar	
		Reference Direction	UVLC Code Number	Reference Direction	UVLC Code Number
DIRECT	16x16	Dependent on spatial predictors	0	Dependent on spatial predictors	0
INTER16x16	16x16	Forward	1	Forward	1
	16x16	Backward	2		
	16x16	Bi-directional	3		
INTER16x8	16x8	Forward, Forward	4	Forward	2
	16x8	Forward, Backward	8		
	16x8	Forward, Bi-directional	12		
	16x8	Backward, Forward	10		
	16x8	Backward, Backward	6		
	16x8	Backward, Bi-directional	14		
	16x8	Bi-directional, Forward	16		
	16x8	Bi-directional, Backward	18		
	16x8	Bi-directional, Bi-directional	20		
INTER8x16	8x16	Forward, Forward	5	Forward	3
	8x16	Forward, Backward	9		
	8x16	Forward, Bi-directional	13		
	8x16	Backward, Forward	11		
	8x16	Backward, Backward	7		
	8x16	Backward, Bi-directional	15		
	8x16	Bi-directional, Forward	17		
	8x16	Bi-directional, Backward	19		
	8x16	Bi-directional, Bi-directional	21		
INTER8x8	8x8		22	Forward	4
INTRA	16x16	None	23	None	5

Table 2: Connection between sub-mode type (each inter 8x8 block) and codeword number

Sub-mode Type	Partition Size	5x3		Haar	
		Reference Direction	UVLC Code Number	Reference Direction	UVLC Code Number
DIRECT	8x8	Dependent on spatial predictors	0	Dependent on spatial predictors	0
sub8x8	8x8	Forward	1	Forward	1
	8x8	Backward	2		
	8x8	Bi-directional	3		
sub8x4	8x4	Forward	4	Forward	2
	8x4	Backward	6		
	8x4	Bi-directional	8		
sub4x8	4x8	Forward	5	Forward	3
	4x8	Backward	7		
	4x8	Bi-directional	9		
sub4x4	4x4	Forward	10	Forward	4
	4x4	Backward	11		
	4x4	Bi-directional	12		

2.1.2.3 Motion Vectors Coding

Only the differences between true motion vectors and their predictors from the neighboring blocks are coded. The predictors can be generated using *median prediction* or Directional *segmentation prediction.*

Median prediction

For partition size with square shape (16×16, 8×8 and 4×8), median prediction technique is used. The spatial predictor is the median of the three previous coded motion vectors from the left block A, the above block B, and the above-right block C (or the above-left block D if the motion vector of the right block is not available), as shown in Figure 3. If a motion vector is not available (outside the picture or coded as intra block), it is not considered candidate motion.

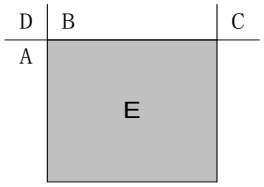
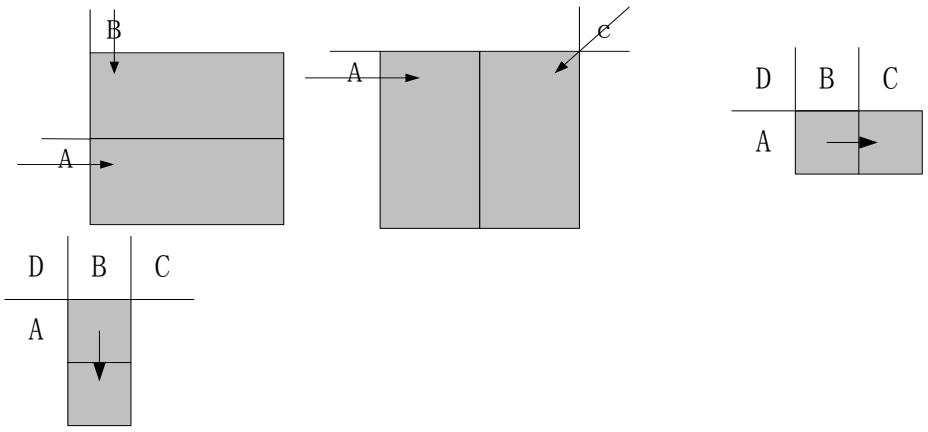


Fig.3.

Figure 3: Median prediction of motion vectors

Directional segmentation prediction

Directional segmentation prediction is used for partition sizes with non-square shapes (16 ×8, 8×16, 8×4 and 4×8).



16x8 8x16

8x4 4x8

Figure 4: Directional segmentation prediction

For 16x8 partition:

- Upper block: motion vector from block B is used as predictor if it is available, otherwise, median prediction is used.
- Lower block: motion vector from block A is used as predictor if it is available, otherwise, median prediction is used.

For 8x16 partition:

- Left block: motion vector from block A is used as predictor if it is available, otherwise, median prediction is used.
- Right block: motion vector from block C is used as predictor if it is available, otherwise, median prediction is used.

For 8x4 partition:

- Left block: "Median prediction" is used.
- Right block: A is used as predictor.

For 4x8 partition

- Upper block: "Median prediction" is used.
- Lower block: A is used as predictor.

2.1.3 Motion Estimation

For each block or macroblock the motion vector is determined by full search on integer-pixel positions followed by sub-pixel refinement. The search positions are organized in a spiral structure around a prediction vector. To reduce the computational complexity, half-pixel search is performed on 8 reference pixels around the best searched point in integer search step. Quarter-pixel search is furthermore performed on 8 reference pixels around the previous best searched point in half-pixel search step.

2.1.3.1 Finding the Best Motion Vector

Rate-constrained motion estimation for a given block S_i is performed by minimizing the Lagrangian cost function:

$$J(mv, \lambda_{SAD}) = D_{SAD}(S_i, mv) + \lambda_{SAD} \times R(mv - mvp)) \quad (1)$$

with the distortion term being given as $D_{SAD}(mv) = \sum_{(i,j)\in X} \left| S_{cur}[i,j] - S_{ref}[i + mv(x), j + mv(y)] \right|$, where *mv, mvp* denote the estimated motion vector and its corresponding motion vector predictor for the current block S_i, $R(mv - mvp)$ denotes the number of bits to transmit *mv*, S_{cur} and S_{ref} represent the current original block and referenced original block respectively.

2.1.3.2 Finding the Best Mode

$$J(\mathrm{mode\_type}, \lambda_{SAD}) = (\sum_{\substack{each \\ block}} D_{SAD}(S_i, mv) + \lambda_{SAD} \times R(mv - mvp))) + \lambda_{SAD} \times R(\mathrm{mode\_type}) \quad (2)$$

Where *R (mode_type)* represents bits to code mode type, see tables 1 and 2.

2.1.4 Convert to 4x4 Motion Fields

To support the generic motion process, the motion field of the luma component for each temporal high-pass subband is organized in 4x4 units. Components for each 4x4 unit include connected/disconnected flag and the corresponding motion vectors. Motion vectors for chroma components are derived from luma component. In the 4:2:0 YUV format picture, motion vectors of the chroma components are the half of the luma ones.
For the "T+2D" scheme, motion vectors at higher spatial resolution have lower fractional pixel accuracy and larger macroblock size. So the motion vectors should be scaled according to the resolution of decoded video. Its algorithm is described as follows:

```
for(int block_y = 0; block_y < nImgHeight / 4; block_y++)
 for(int block_x = 0; block_x < nImgWidth / 4; block_x++){
      // For left reference motion info
       if(m_nBlockRef[REF_LIST_FWD][block_y ][block_x ] >= 0){
              pFrameBuffer->m_pLeftYMVBuf[block_y][block_x].flag = 0;
    pFrameBuffer->m_pLeftYMVBuf[block_y][block_x].mv_x
           = m_nBlockMVs[REF_LIST_FWD][block_y][block_x][0] / nFactor;
    pFrameBuffer->m_pLeftYMVBuf[block_y][block_x].mv_y
         = m_nBlockMVs[REF_LIST_FWD][block_y][block_x][1] / nFactor;
    }
    else{
    pFrameBuffer->m_pLeftYMVBuf[block_y][block_x].flag = 1;
   }
```

```
  // For right reference motion info
    if(m_nBlockRef[REF_LIST_BWD][block_y ][block_x ] >= 0){
      pFrameBuffer->m_pRightYMVBuf[block_y][block_x].flag = 0;
      pFrameBuffer->m_pRightYMVBuf[block_y][block_x].mv_x
          = m_nBlockMVs[REF_LIST_BWD][block_y ][block_x][0] / nFactor;
      pFrameBuffer->m_pRightYMVBuf[block_y][block_x].mv_y
          = m_nBlockMVs[REF_LIST_BWD][block_y ][block_x][1] / nFactor;
    }
    else{
      pFrameBuffer->m_pRightYMVBuf[block_y][block_x].flag = 1;
    }
 }
```

Note: nFactor is the motion vectors scaling factor that is calculated according to ratio between full resolution and target resolution of decoded video and flag represent connected (0) or disconnected (1).

2.2 Temporal Transform

2.2.1 Motion Aligned Temporal Lifting

For practical description of the motion alignment across frames, the lifting structure of wavelet transform is incorporated into temporal filtering. With the lifting structure, any traditional motion model that establishes a pixel-mapping relationship between two adjacent frames can be easily adopted by the motion aligned temporal filtering. Especially the lifting structure facilitates exploiting fractional pixel motion vector and flexible frame prediction pattern with perfect reconstruction.

The key idea of the lifting scheme mainly contains the following steps: 1) Split the data $\{F_i\}$ into two subsets: $\{A_i\}$ for elements with even index and $\{B_i\}$ for elements with odd index; 2) Predict B_i from the subset $\{A_i\}$ and calculate the prediction error as the high-pass wavelet coefficient H_i; 3) Update $\{A_i\}$ with the high-pass wavelet coefficients $\{H_i\}$ to ensure preservation of moments in the low-pass coefficients L_i. These steps are briefly referred as *Split*, *Predict* and *Update* step, respectively. The Figure 5(a) illustrates the general lifting scheme for motion aligned temporal filtering. The lifting steps for forward MATF can be formulated as:

$$\textit{Split}: \quad A_i = F_{2i}, \; B_i = F_{2i+1}$$

$$\text{Motion Aligned } \textit{Predict}: \quad H_i = B_i - P\{A_i\}$$

$$\text{Motion Aligned } \textit{Update}: \quad L_i = A_i - U\{H_i\}$$

It is noteworthy that for motion aligned temporal filtering both the operator Π and Υ are performed according to motion alignment so that they can be referred as *Motion Aligned Predict* (MAP) and *Motion Aligned Update* (MAU). In this paper, the $P_i = P\{A_i\}$ in the *Predict* step is always referred as prediction signal or prediction image. In the same way, the $U_i = U\{H_i\}$ in the *Update* step is always referred as update signal or update image. Both P_i and U_i are referred as lifting signal or lifting image. The motion aligned temporal filtering with lifting structure can be easily reversed by reversing each of the lifting steps, as shown in Figure 5(b).

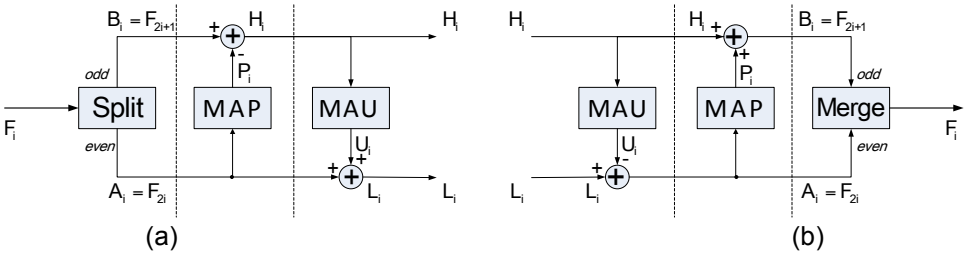


Figure 5: The general lifting scheme for MATF and inverse MATF

Figure 6 shows the temporal lifting steps of 5/3 biorthogonal wavelet that is the most popular structure adopted in temporal transform. In both the *Predict* and *Update* steps, only two directly neighboring frames are considered to form a bi-directional prediction or update signal. Block based bidirectional motion estimation is firstly performed at each B_i frame, using its two neighboring A_i frames as references. The obtained motion vectors establish a pixel mapping relationship between adjacent A-frames and B-frames. This mapping is exploited both in the *Predict* and the *Update* step. In the *Predict* step, the motion vectors are used directly. However, in the *Update* step, the derived inverse motion vectors are usually used as simple approximation. The 5/3 temporal filtering can be formulated as follows:

$$H_i = B_i - \frac{1}{2}[MAP_{2i+1\to 2i}(A_i) + MAP_{2i+1\to 2i+2}(A_{i+1})]$$

$$L_i = A_i + \frac{1}{4}[MAU_{2i\to 2i-1}(H_{i-1}) + MAU_{2i\to 2i+1}(H_i)]$$

Beside the above bidirectional-connected blocks, there can be other cases. For a block in a B-frame that cannot be well matched from one or both of its neighboring A-frames, the block can be marked as disconnected in one direction or in both directions. Its prediction mode can be changed from bidirectional prediction to unidirectional prediction or Intra block. Similarly, the disconnected A-frame will not be updated by the above block.

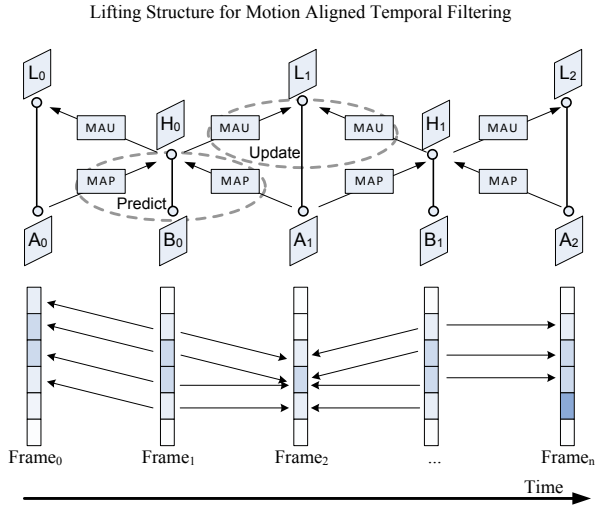


Figure 6: The 5/3 lifting structure for motion aligned temporal filtering.

2.2.2 Prediction Step

In the *Predict* lifting step, prediction lifting frames are generated from A-frames through motion compensation. At the encoder, these lifting frames are scaled with a proper lifting step tap value and added to B-frames to generate the H-frames. At decoder, these lifting frames are scaled with an inverse lifting step tap value and added to H-frames to reconstruct the B-frames.

2.2.2.1 Bidirectional and Unidirectional Connection

Currently, only the two nearest frames are considered as references. That is, only Haar, 5/3 and combinations of them are adopted in the prediction lifting step. Basically, prediction lifting frame is generated block by block. For each block in a B-frame, it can be bidirectional connected or unidirectional connected. The motion information of each block tells the connectivity of this block with its two neighboring reference frames.

- Bidirectional connection

  Bidirectional connection is a usual status for a block that can be well-matched with blocks in both of its neighboring reference frames. For these blocks, bidirectional prediction is used:

$$P[n,m] = \{P_{Left}[n,m] + P_{Right}[n,m]\} / 2$$

  where P_{Left}[n,m] and P_{Right}[n,m] are the left and right prediction values for the pixel at the [n,m] position, respectively.

- Unidirectional connection

  For a block which cannot be well-matched with a block in neighboring frames or which belongs to a frame having only one reference frame, it should be marked as unidirectional connected. For these blocks, unidirectional prediction is used:

$$P[n,m] = P_{Left}[n,m] \text{ when left-unidirectional connected}$$

$$P[n,m] = P_{Right}[n,m] \quad \text{when right-unidirectional connected}$$

  where P_{Left}[n,m] and P_{Right}[n,m] are the left and right prediction values for the pixel at the [n,m] position, respectively.

2.2.2.2 Padding

Before interpolation and motion compensation, video frames are padded to handle the out-of-bound reference:

```
// upper padding
for(j = 0; j < m_nPaddingSize; j++)
    for(i = 0; i < mwidth; i++)
        y[j][i] = y[m_nPaddingSize][i];

// bottom padding
for(j = mheight - m_nPaddingSize; j < mheight; j++)
    for(i = 0; i < mwidth; i++)
        y[j][i] = y[mheight - 1 - m_nPaddingSize][i];

// left padding
for(j = 0; j < mheight; j++)
    for(i = 0; i < m_nPaddingSize; i++)
            y[j][i] = y[j][m_nPaddingSize];
```

```
// right padding
for(j = 0; j < mheight; j++)
    for(i = mwidth - m_nPaddingSize; i < mwidth;i++)
            y[j][i] = y[j][mwidth - 1 - m_nPaddingSize];
```

2.2.2.3 Interpolation

The following interpolation filter is used:

```
float filtertap[7][8] =
{
    {-0.0072F,  0.0284F,  -0.0902F,  0.9742F,  0.1249F,  -0.0380F,  0.0105F,
-0.0026F},
    {-0.0110F,  0.0452F,  -0.1437F,  0.8950F,  0.2777F,  -0.0812F,  0.0233F,
-0.0053F},
    {-0.0117F,  0.0505F,  -0.1624F,  0.7713F,  0.4465F,  -0.1224F,  0.0363F,
-0.0081F},
    {-0.0105F,  0.0465F,  -0.1525F,  0.6165F,  0.6165F,  -0.1525F,  0.0465F,
-0.0105F},
    {-0.0081F,  0.0363F,  -0.1224F,  0.4465F,  0.7713F,  -0.1624F,  0.0505F,
-0.0117F},
    {-0.0053F,  0.0233F,  -0.0812F,  0.2777F,  0.8950F,  -0.1437F,  0.0452F,
-0.0110F},
    {-0.0026F,  0.0105F,  -0.0380F,  0.1249F,  0.9742F,  -0.0902F,  0.0284F,
-0.0072F}
};
```

Assuming the integer-pixel frame is represented by F[n,m], the 2D interpolation process has the following two steps:

- Horizontal interpolation is done with:

$$F[n, m+\frac{x}{8}] = \sum_{i=-3}^{4} F[n, m+i] \cdot filtertap[x-1][3+i] \qquad (1 \le x \le 7,\ n, m \in \text{¢})$$

- Vertical interpolation is done with:

$$F[n+\frac{y}{8}, m+\frac{x}{8}] = \sum_{i=-3}^{4} F[n+i, m+\frac{x}{8}] \cdot filtertap[y-1][3+i]$$
$$(1 \le y \le 7,\ 1 \le x \le 7,\ n, m \in \text{¢})$$

2.2.2.4 Motion Compensation

With the fractional-pixel accuracy motion vectors and interpolation, the motion compensation can be implemented as simple pixel fetching from the interpolated reference frames:

$$P_{Left}[n,m] = R_{Left}[n + \Delta n_{Left}[n,m], m + \Delta m_{Left}[n,m]]$$
$$P_{Right}[n,m] = R_{Right}[n + \Delta n_{Right}[n,m], m + \Delta m_{Right}[n,m]]$$

where $[\Delta n_{Left}[n,m], \Delta m_{Left}[n,m]]$ and $[\Delta n_{Right}[n,m], \Delta m_{Right}[n,m]]$ are the left and right motion vectors at the position [n,m]. $R_{Left}[n,m]$ and $R_{Right}[n,m]$ are the left and right reference frames, respectively. When the index [n,m] is not integer, interpolation as described in the previous section is used to give the "virtual pixel" value.

2.2.2.5 Overlapped Block Motion Compensation (OBMC)

To reduce the high-frequency component at block boundaries in prediction frames, OBMC techniques are used during block based motion compensation. In OBMC techniques, the motion vectors of neighboring blocks are considered and the final prediction for a pixel is a weighted combination of several predictions generated by motion vectors of itself and its neighboring blocks.

$$P_{Left}[n,m] = \sum_{\Upsilon \in B} \omega_{\Upsilon}[F_{pm}[n,m]] R_{Left}[n + \Delta n_{Left,\Upsilon}[n,m], m + \Delta m_{Left,\Upsilon}[n,m]]$$

$$P_{Right}[n,m] = \sum_{\Upsilon \in B} \omega_{\Upsilon}[F_{pm}[n,m]] R_{Right}[n + \Delta n_{Right,\Upsilon}[n,m], m + \Delta m_{Right,\Upsilon}[n,m]]$$

where B={center, left, right, up, down } is the label set of involved blocks, $[\Delta n_{Left,\Upsilon}[n,m], \Delta m_{Left,\Upsilon}[n,m]]$ and $[\Delta n_{Right,\Upsilon}[n,m], \Delta m_{Right,\Upsilon}[n,m]]$ are the forward and backward motion vectors of the Υ block of the pixel [n,m]. The $F_{pm}[n,m]$ is a position map function which converts the position in a frame to the relative position in its block. Simply we have $F_{pm}[n,m] = [n \bmod S,\ m \bmod S]$ where S is the block size.

For the following possible weighting window (S=4):

		1	2	2	1		
		3	4	4	3		
1	3	5	6	6	5	3	1
2	4	6	7	7	6	4	2
2	4	6	7	7	6	4	2
1	3	5	6	6	5	3	1
		3	4	4	3		
		1	2	2	1		

the $\omega_{\Upsilon}[n,m]$ is defined as follows:

5	6	6	5
6	7	7	6
6	7	7	6
5	6	6	5

$\omega_{center}[n,m]$

3	1	0	0
4	2	0	0
4	2	0	0
3	1	0	0

$\omega_{left}[n,m]$

0	0	1	3
0	0	2	4
0	0	2	4
0	0	1	3

$\omega_{right}[n,m]$

3	4	4	3
1	2	2	1
0	0	0	0
0	0	0	0

$\omega_{up}[n,m]$

0	0	0	0
0	0	0	0
1	2	2	1
3	4	4	3

$\omega_{down}[n,m]$

2.2.3 Update Step

In the *Update* lifting step, update lifting frames are generated from H-frames through motion compensation. At the encoder, these lifting frames are scaled with a proper lifting

step tap value and added to A-frames to generate the L-frames. At the decoder, these lifting frames are scaled with an inverse lifting step tap value and added to L-frames to reconstruct the A-frames.
Let HC[n, m] denote the update signals generated by motion compensated high-pass frames. It can be directly derived along the motion vectors used in the prediction step and motion inversion can then be avoided. The procedure to compute U[n, m] is described:

For all [n, m],
{
Let HC[n, m] = 0;
Get the motion vector [Δn, Δm];
If [n+ Δn, m+ Δm] is within the image
{

$$HC[n,m] = HC[n,m] + \omega_1 \times H[\lfloor n+\Delta n \rfloor, \lfloor m+\Delta m \rfloor] + \omega_2 \times H[\lfloor n+\Delta n \rfloor, \lceil m+\Delta m \rceil] + \omega_3 \times H[\lceil n+\Delta n \rceil, \lfloor m+\Delta m \rfloor] + \omega_4 \times H[\lceil n+\Delta n \rceil, \lceil m+\Delta m \rceil]$$

}
}
Where H[] is the corresponding high-pass frame. The weights $\omega_1, \omega_2, \omega_3, \omega_4$ are calculated according to the bilinear weights:

$$\omega_1 = \left[1-(n+\Delta n-\lfloor n+\Delta n \rfloor)\right] \times \left[1-(m+\Delta m-\lfloor m+\Delta m \rfloor)\right]$$
$$\omega_2 = \left[1-(n+\Delta n-\lfloor n+\Delta n \rfloor)\right] \times (m+\Delta m-\lfloor m+\Delta m \rfloor)$$
$$\omega_3 = (n+\Delta n-\lfloor n+\Delta n \rfloor) \times \left[1-(m+\Delta m-\lfloor m+\Delta m \rfloor)\right]$$
$$\omega_4 = (n+\Delta n-\lfloor n+\Delta n \rfloor) \times (m+\Delta m-\lfloor m+\Delta m \rfloor)$$

Once HC[n,m] is available, the update signals to be added into the high-pass frames U[n,m] can be got by:

$$U[n,m] = \begin{cases} T & HC[n,m] \geq T \\ HC[n,m] & -T < HC[n,m] < T \\ -T & HC[n,m] \leq -T \end{cases}$$

Where T is a threshold to keep the update signals from being too large and introducing ghosting effects. Note that it can be different for each level of the temporal transform.

2.3 Spatial Transform

2.3.1 Spatial Decomposition Structure Description

Current 3D wavelet video codecs with MCTF can be divided into two categories. The first one performs MCTF on the original video sequence before spatial transform, category often referred to as image domain MCTF. On the contrary, second category

codecs perform MCTF after spatial transform, which is often referred to as wavelet domain MCTF or In-Band MCTF.
The MSRA codec supports both of the two schemes. Before MCTF, a pre-spatial decomposition can be done to the original video sequence. Depending on the pre-spatial decomposition structure, the MCTF can be applied in the image domain or in the wavelet subband domain. After MCTF, a post-spatial decomposition can then be performed to each temporal subband to further split the frames spatially.
The whole wavelet decomposition structure includes pre-spatial decomposition, multi-level temporal decomposition and post-spatial decomposition, as illustrated by Figure 7.

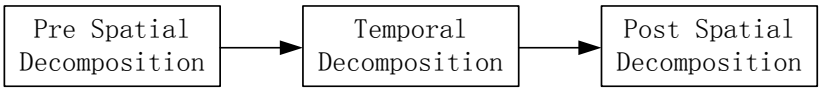


Figure 7: Decomposition Structure

Figure 8 illustrates the module list for the scheme.

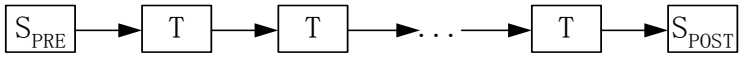
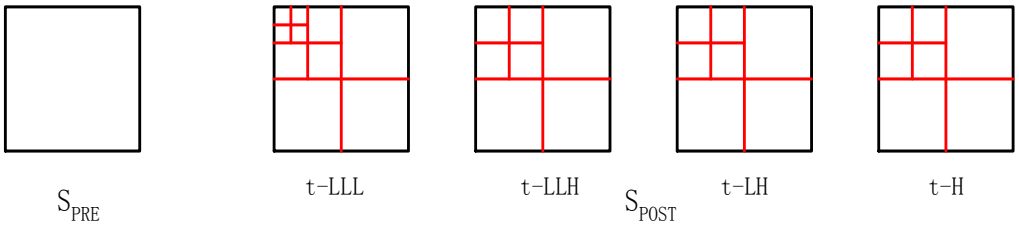


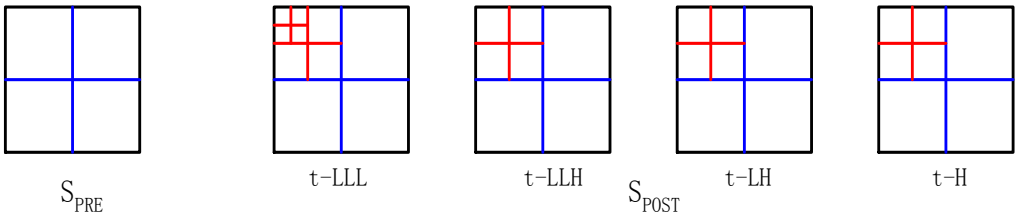
Figure 8: Module list for the scheme.

We first give some examples of possible pre- and post- spatial decomposition structures. Let us assume the number of temporal DWT levels is 3.
Example 1: As in Figure 9(a), the pre-spatial decomposition is void.
Example 2: As in Figure 9(b), the pre-spatial decomposition is a one-level spatial transform.
Example 3: As in Figure 9(c), the pre- spatial decomposition is a two-level spatial transform.

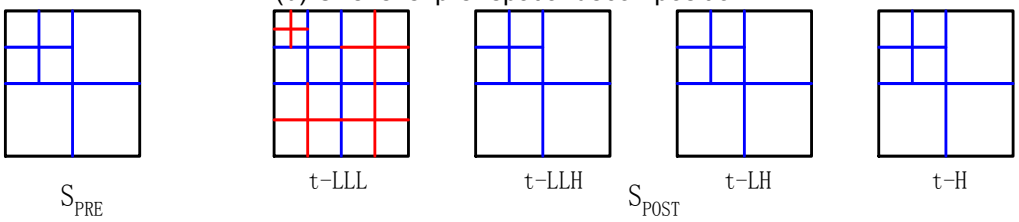
Pre-spatial transform is illustrated with the blue lines in the Figure 9. After the 3-level temporal DWT, the resulting four temporal subbands t-LLL, t-LLH, t-LH, t-H are further splitted spatially, as shown by Figure 9 with the red lines. Furthermore, we can notice that the decomposition structures for the four temporal subbands can be different. They can be adjusted according to the signal properties of the corresponding subbands.



(a) Empty pre- spatial decomposition



(b) One level pre- spatial decomposition



(c) Two level pre- spatial decomposition

Figure 9: Three examples of pre- and post- spatial decomposition.

To describe the spatial decomposition structure, we use a recursive description method. As shown in Figure 10, one spatial decomposition operation splits band B into four subbands B1, B2, B3 and B4, which correspond to the LL, HL, LH and HH subband of the band B, respectively. The four bands B1, B2, B3 and B4 can be further decomposed. We define the decomposition string for band B as DecomStr(B):

DecomStr (B) = S<> DecomStr (B1) DecomStr (B2) DecomStr (B3) DecomStr (B4)

where S<> stands for a spatial transform. We can specify the property of the spatial transform operation inside "<>". For instance, to specify the wavelet filter used in a spatial transform, we can use "S<Filter=W9x7>" or "S<Filter=W5x3>". W9x7 is the default 9/7 filter type value for spatial transform and can be omitted in the decomposition strings.

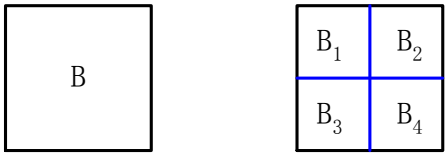


Figure 10: One spatial decomposition operation

If the band B is not further decomposed, we have

DecomStr (B) = E<>

where E<> can be understood as "end of further spatial decomposition".

For example, the S_{PRE} decomposition in the Figure 9(a)(b)(c) can be expressed as:

S_{PRE} = E<> for Figure 9(a)

S_{PRE} = S<> E<> E<> E<> E<> for Figure 9(b)

S_{PRE} = S<> S<> E<> E<> E<> E<> E<> E<> E<> for Figure 9(c)

The S_{POST} decomposition four each temporal subband in the Figure 9 should be expressed as following:

S_{POST} = S_{POST}(t-H), S_{POST}(t-LH), S_{POST}(t-LLH), S_{POST}(t-LLLH), … , S_{POST}(t-LL...L)

where S_{POST}(t-H) should describe the complete spatial decomposition operation, including the pre-spatial decompositions, for convenient parsing of the decomposition strings. For example, for Figure 9(c) we have

SPOST(t-H) = S<> S<> E<> E<> E<> E<> E<> E<> E<>

SPOST(t-LLL) = S<> S<> S<> E<> E<> E<> E<>E<> E<> E<> S<> E<> E<> E<> E<> E<> S<> E<> E<> E<> E<> S<> E<> E<> E<> E<>

2.4 Entropy Coding

After temporal and spatial sub-band transforms, the coefficients are coded with 3D Embedded Block Coding with Optimal Truncation (3D EBCOT). Basically, 3D EBCOT follows the idea and coding procedure of EBCOT in JPEG2000, which deals with 2D image coding. We extend the coding scheme into 3D dimensional signal coding.

2.4.1 3D EBCOT

Each sub-band, generated by temporal and spatial transforms, is divided into 3D coding blocks, which are coded independently. For each coding block, bit-plane coding and context-based arithmetic coding are used.

Let $\sigma[i,j,k]$ be a binary-valued state variable, which illustrates the significance of the sample of a coding block at position [i,j,k]. $\sigma[i,j,k]$ is initialized to 0 and toggled to 1 when the relevant sample's first non-zero bit-plane value is encoded. And $\chi[i,j,k]$ is defined as the sign of the sample, which is 0 when the sample is positive and 1 when the sample is negative. Three coding operations are used to code the samples in a bit-plane.

Zero Coding: When a sample is not yet significant in the previous bit-plane, i.e. $\sigma[i,j,k]=0$, this primitive operation is utilized to code the new information of the sample. It tells whether the sample becomes significant or not in the current bit-plane. The ZC operation uses the information of the current sample neighbor as a context to code the current sample significant information. Three categories of a sample neighbor (see Figure 11) are considered:

- Immediate horizontal neighbors. We denote the number of these neighbors that are significant by h, $0<h<2$.
- Immediate vertical neighbors. We denote the number of these neighbors that are significant by v, $0<v<2$.
- Immediate temporal neighbors. We denote the number of these neighbors that are significant by a, $0<a<2$.
- Immediate temporal neighbors. We denote the number of these neighbors that are significant by d, $0<d<12$.

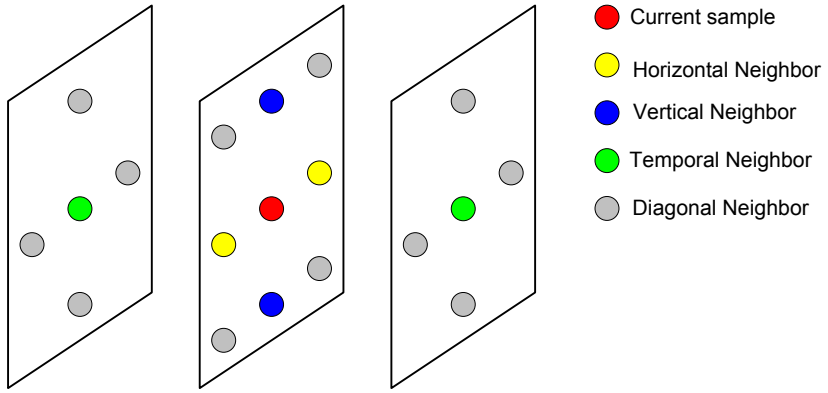


Figure 11: Coding neighbors for zero coding

Table 3 lists context assignment map for Zero Coding. If the conditions of two or more rows are satisfied in the same time, the low-numbered context is selected. An adaptive context-based arithmetic coder is used to code the significant symbols of Zero Coding.

Table 3: Context assignment map for Zero Coding.

LLL and LLH sub-band	h	2	1	1	1	0	0	0	0	0	0	0
	v	x	≥1	0	0	2	1	0	0	0	0	0
	a	x	x	≥1	0	0	0	≥1	0	0	0	0
	d	x	x	x	x	x	x	x	3	2	1	0
	context	0	0	1	2	3	4	5	6	7	8	9
LHH sub-band	h	2	1	1	1	1	1	0	0	0	0	0
	v+a	x	≥3	≥1	≥1	0	0	≥3	≥1	≥1	0	0
	d	x	x	≥4	x	≥4	x	x	≥4	x	≥4	x
	context	0	0	1	2	3	4	5	6	7	8	9
HHH sub-band	d	≥6	≥4	≥4	≥2	≥2	≥2	≥0	≥0	≥0	≥0	
	h+v+a	x	≥3	x	≥4	≥2	x	≥4	≥2	1	0	
	context	0	1	2	3	4	5	6	7	8	9	

Sign Coding: Once a sample becomes significant in the current bit-plane, Sign Coding operation is called to code the sign of the sample. Sign Coding also uses an adaptive context-based arithmetic coder to compress sign symbols. Similarly, we define three quantities, h, v and a, by

$$h=\min\{1, \max\{-1, \sigma[i-1,j,k] \times (1-2\chi[i-1,j,k])+ \sigma[i+1,j,k]\times(1-2\chi[i+1,j,k])\}\}$$

$$v=\min\{1, \max\{-1, \sigma[i,j-1,k] \times (1-2\chi[i,j-1,k])+ \sigma[i,j+1,k] \times (1-2\chi[i,j+1,k])\}\}$$

$$a=\min\{1, \max\{-1, \sigma[i,j,k-1] \times (1-2\chi[i,j,k-1])+ \sigma[i,j,k+1] \times (1-2\chi[i,j,k+1])\}\}$$

Table 4 shows context assignments. $\hat{\chi}$ denotes the sign symbol prediction in the given context. The symbol sent to the arithmetic coder is $\hat{\chi}$ XOR χ.

Table 4: Context assignment and sign prediction map for Sign Coding

h=-1	v	-1	-1	-1	0	0	0	1	1	1
	a	-1	0	1	-1	0	1	-1	0	1
	$\hat{\chi}$	0	0	0	0	0	0	0	0	0
	context	0	1	2	3	4	5	6	7	8
h=0	v	-1	-1	-1	0	0	0	1	1	1
	a	-1	0	1	-1	0	1	-1	0	1
	$\hat{\chi}$	0	0	0	0	0	1	1	1	1
	context	9	10	11	12	13	12	11	10	9
h=1	v	-1	-1	-1	0	0	0	1	1	1
	a	-1	0	1	-1	0	1	-1	0	1
	$\hat{\chi}$	1	1	1	1	1	1	1	1	1
	context	8	7	6	5	4	3	2	1	0

Magnitude Refinement: Magnitude Refinement is used to code the new information of a sample that has already become significant in the previous bit-plane. This operation has three contexts. If the MR operation is not yet used in this sample, context value is 0. If MR has been used in the sample and the sample has at least one significant neighbor by now, then the context value 1, otherwise the context is 2.

Following the EBCOT algorithm, for each bit-plane, the coding procedure consists of three distinct passes, applied in turn. Each pass processes a "fractional bit-plane". Moreover, in each pass, the scanning order is first done along the i-direction, then along the j and k-directions. The three passes are described here:

Significant propagation pass: In this pass, the samples that are not yet significant but have "preferred neighborhood" are processed. A sample has "preferred neighborhood" if and only if the sample has at least a significant immediate diagonal neighbor for HHH sub-band or horizontal, vertical, temporal neighbor for the other types of sub-band. For the sample that satisfies these conditions, we apply the ZC primitive to code the symbol of current bit-plane of this sample. If the sample becomes significant in the current bit-plane, then the SC primitive is used to code the sign.

Magnitude Refinement pass: We code the samples that have been significant in this pass. And the symbols of these samples in the current bit-plane are coded by MR primitive.

Normalization pass: During this pass, those samples that are not yet coded in the previous two passes are coded. These samples are insignificant. So ZC and SC primitives are applied in this pass.

Then R-D optimal truncation is used among coding blocks to decide how many bits are allocated for each coding block.

2.4.2 Bitplane Bitstream and R-D information

After the above 3D EBCOT entropy coding, we have the following information for each coding block:

- The number *N* of total bitplanes in the block
- The coded bitstream of the block, which consists of *3N-2* segments, each segment corresponding to the output bitstream of one coding pass.
- The length of each bitstream segment and/or the position of end-point of each coding pass.
- The R-D slope information at the end of each coding pass.

2.5 Bitstream Formation

Through the 3D EBCOT entropy coding, an embedded bitstream is generated for every coding block so the whole video sequence contains many embedded bitstreams.

To generate a bitstream of specified resolution and bit rate, it should be decided for each block how many bytes of its embedded bitstream or how many coding passes are included into the output bitstream. The decision raises a competition among all the coding blocks and it is arbitrated by an equal-slope criterion.

When resolution scalability is not considered, a fully embedded total bitstream can be produced by generating a multiple-layer bitstream, with each layer assigned a progressively increasing bit rate. The equal-slope criterion should be applied at each bitstream layer to make the bitstream selection decision.

2.5.1 Weights of Subbands

For each subband, a weight is assigned to show the importance of coefficients in this subband. The weight actually reflects the synthesis gain of the subband. The value quantifies, when there is a unit distortion in the subband domain, the distortion amount produced in the reconstructed final video.

For one level DWT transform, the synthesis gain is calculated by its synthesis filters. For multi-level DWT transform and multiple dimensional transform, simple multiplication is adopted as an approximation.

For the 5/3 filter implemented in lifting structure without normalization step, we have

Wlowpass=1.5 Whighpass=0.7188

For the 9/7 filter implemented in lifting structure without normalization step, we have

Wlowpass= 1.29906 Whighpass= 0.787261

2.5.2 Bitstream Selection

According to the equal-slope criterion, at each bitstream layer, an R-D slope threshold is used to select bitstream pieces from the numerous coding blocks. If the R-D slope at the end of a coding pass of a block is larger than the threshold, the related bitstream at and before that coding pass will be included into the output bitstream. On the contrary, if the R-D slope of a coding pass is smaller than the threshold, it will not be included into the output bitstream.

According to the above rule, the bitstream segments in each block to be included into the output bitstream are completely decided when the R-D slope threshold is given. Furthermore, we can know the size of the total output bitstream for each given threshold. To meet the given spatio-temporal resolution and bit rate constraints, a proper threshold value is obtained by bisection searching.

The following is a demonstrative code for the dichotomy searching.

```
void CBitStreamPacker::RDThresholdSelect(LAYER_INFO *pLayer,
     int previous_cumulative_bytes)
{
 rd_slope_type max_rd_threshold, min_rd_threshold, rd_threshold;

 /* Start by saving the status of previous layer. */
 BackupRDSelectionStatus(pLayer);

 /* The initial searching range and initial threshold. */
 max_rd_threshold=SHRT_MAX;
 min_rd_threshold=0;
 rd_threshold = (rd_slope_type)((((int) min_rd_threshold)+
              ((int) max_rd_threshold) + 1) >> 1);

 /* Now for the main iterative loop. */
 do {
     int_32u actual_bytes = previous_cumulative_bytes;

     /* Use the rd_threshold to simulate the packets forming
          process, which will change the status of related
          blocks and return the packet size. */
     actual_bytes += FormPackets(rd_threshold, … );

     /* Adjust the searching range and threshold value accordingly. */
     if (actual_bytes > pLayer->max_cumulative_bytes)
         min_rd_threshold = rd_threshold;
     else
         max_rd_threshold = rd_threshold;
     rd_threshold = (rd_slope_type)((((int) min_rd_threshold)+
                     ((int) max_rd_threshold) + 1) >> 1);

     /* Restore the status of previous layer. */
 RestoreRDSelectionStatus(pLayer);

 } while (rd_threshold < max_rd_threshold);

 /* Now the rd_threshold is what we want. Store it for later use. */
 pLayer->rd_threshold_Y  = rd_threshold;
 pLayer->rd_threshold_UV = rd_threshold;
}
```

In the case of generating multiple-layer bitstreams, the selection process of the coding passes of one layer should have the knowledge of previous layers. That is, it should know how many coding passes in current block have already been included in previous layers. This information is stored in the status of each coding block. It is noteworthy that the bisection searching simulates the packet formation process many times, it backups and restores the coding block status in the same time to make sure that all status before each simulation are the same.

The following code simulates the whole encoding process of a GOP and searches for the thresholds for each layer according to bytes-allocation:

```
//Reset all status for bitstream forming
ResetAllStatus();

//bytes counter of current GOP
actual_bytes=0;
//code GOP header
actual_bytes += m_pStream->WriteGOPHeader(
(unsigned char*)&(GOP_Header),sizeof(GOP_HEADER),SIMULATION);

for(layer_idx=0; (int) layer_idx < m_PackerInfo.nNum_Layers; layer_idx++)
{
 pLayer_Header[layer_idx].uIdx_Layer = layer_idx;
 pLayer = & (m_PackerInfo.LayerInfos[layer_idx]);

 //code bitstream layer header
 actual_bytes += m_pStream->WriteLayerHeader(
(unsigned char*)&(pLayer_Header[layer_idx]),
sizeof(LAYER_HEADER),SIMULATION);

 //mark the selection range of this layer for resolution scalability
 SetBandInclusion(layer_idx);

 // search for the appropriate rd_threshold for Y and UV component
// to meet the bit-rate constraint
 RDThresholdSelect(pLayer,actual_bytes);

 // simulate the process of FormPacket to obtain necessary packet
//  size information
// and accomplish the status transition
pLayer_Header[layer_idx].uLayer_Bytes = FormPackets( … );
 actual_bytes += pLayer_Header[layer_idx].uLayer_Bytes;

}
```

The final real coding of these multiple bistream layers of the current GOP is nearly the same with the above code. The only difference is that the rd_threshold information of each layer and the size information of each packet are already available to guide the formation of the final bitstream.

2.5.3 Syntax

Structure of a whole sequence:

```
SequenceBitstream =
{
GlobalHeader
+ GOPBitstream       // bitstream for GOP 0
+ GOPBitstream       // bitstream for GOP 1
+ GOPBitstream       // bitstream for GOP 2
+
…
+ GOPBitstream       // bitstream for GOP nGOP-1
}
```

Description: the whole sequence consists of a number of GOPs. The GOP sizes do not necessary always have the same value. For example, for a 300-frame sequence, it can consist of four 64-frame GOPs and a 44-frame GOP.

A GOP contains frames of various temporal subbands. For example, when four-level temporal transform is applied at the encoder, there are five possible temporal subbands,

t-H, t-LH, t-LLH, t-LLLH and t-LLLL. For a GOP size of 64, it consists of 32 t-H frames, 16 t-LH frames, 8 t-LLH frames, 4 t-LLLH frames and 4 t-LLLL frames.

Structure of a GOP:

```
GOPBitstream =
{
GOPHeader
+ LayerBitstream     // bitstream for Layer 0
+ LayerBitstream     // bitstream for Layer 1
+ LayerBitstream     // bitstream for Layer 2
+
…
+ LayerBitstream     // bitstream for Layer nLayer-1
}
```

Description: The bitstream of a GOP consists of several layers to support quick bitstream truncation. The bitstream with the higher layer index can be dropped directly when necessary. Each bitstream layer has a specified parameter set of spatial, temporal resolution and target bit rate. The bitstream layers with higher index should always have equal or higher resolution and equal or higher bit rate than the lower layers.
The total number of bitstream layers in the current bitstream is coded in GlobalHeader and this value is the same for all GOPs. During truncation, when some bitstream layers are dropped, the number of layers remained in a GOP decreases and the GlobalHeader is modified accordingly. It is possible that only one layer exists in a bitstream.

Structure of a bitstream layer:

```
LayerBitstream =
{
LayerHeader
+ PacketBitstream        // bitstream for Packet 0
+ PacketBitstream        // bitstream for Packet 1
+ PacketBitstream        // bitstream for Packet 2
+
…
+ PacketBitstream        // bitstream for Packet nPacket-1
}
```

Description: the LayerBitstream codes the current GOP with a specified parameter set (*S*, *T*, *R*) where *S* is the spatial resolution parameter, *T* is the temporal resolution parameter and *R* is the target bit rate. The parameters *S* and *T* indicate which spatial-temporal subbands that should be included in current layer and which subbands are not necessary.
The LayerBitstream consists of a number of PacketBitstream(s), with each PacketBitstream corresponding to one component of a temporal subband. For example, for four-level temporal transform encoding and *T=2* (1/4 of input frame rate), we have the following 3*3=9 packets: P(t-LLH,Y), P(t-LLH,U), P(t-LLH,V), P(t-LLLH,Y), P(t-LLLH,U), P(t-LLLH,V), P(t-LLLL,Y), P(t-LLLL,U) and P(t-LLLL,V), in their correct orders.

Structure of a packet:

```
PacketBitstream =
{
PacketHeader
PacketBody
}
```

Description: as mentioned above, a PacketBitstream corresponds to one component (Y or U or V) of one temporal subband in the current GOP. The temporal subband may have several frames in depth and it may consist of many small spatial-temporal subbands produced by the spatial decomposition. All these subbands are considered as coded in the same packet.
Every PacketBitstream belongs to a bitstream layer and it uses the *S* parameter of the layer to select the spatial subbands it should cover.

Structure of a packet header:

```
PacketHeader =
{
PacketNonEmptyFlag      //(1 bit )
+ TemporalBandIndex     //(3 bits)
+ ComponentIndex    //(2 bits)
+ PacketSize        //(28 bits)
+ PacketHeaderSize      //(22 bits)

+ SideInformation       //mainly the motion information

+ SubbandBitplaneSelectionInformation       // for subband 0
+ SubbandBitplaneSelectionInformation       // for subband 1
+ SubbandBitplaneSelectionInformation       // for subband 2
+
…
+ SubbandBitplaneSelectionInformation       // for subband nBand -1

}
```

Description: the packet header mainly contains the following information:

- Properties of the packet, such as the temporal subband index, component index, …
- SideInformation: mainly the motion bitstream. It is optional and only available for Y component packet
- Bitplane coding passes selection information: It tells how many coding passes are included in the current bitstream layer for each covered spatial subband.

Structure of a packet body:

```
PacketBody =
{
SubbandBPBitstreamSegment               // for subband 0
+ SubbandBPBitstreamSegment         // for subband 1
+ SubbandBPBitstreamSegment         // for subband 2
+
…
+ SubbandBPBitstreamSegment         // for subband nBand -1
}
```

Description: one SubbandBPBitstreamSegment for each covered spatial subband

Structure of a SubbandBPBitstreamSegment:

```
SubbandBPBitstreamSegment =
{
BlockBPBitstreamSegment             // for Block 0
+ BlockBPBitstreamSegment               // for Block 1
+ BlockBPBitstreamSegment               // for Block 2
+
…
```

```
+ BlockBPBitstreamSegment                // for Block nBlock -1
}
```

Description: a spatial-temporal subband consists of a number of coding blocks. Accordingly, a SubbandBPBitstreamSegment consists of a number of BlockBPBitstreamSegment, one BlockBPBitstreamSegment for each coding block.

Structure of a BlockBPBitstreamSegment:

```
BlockBPBitstreamSegment =
{
Appended Bytes of this block in current bitstream layer
}
```

Description: the appended bitstream bytes of current coding block to current bitstream layer.

Structure of a SubbandBitplaneSelectionInformation:

```
SubbandBitplaneSelectionInformation =
{
BlockBitplaneSelectionInformation       // for block 0
+ BlockBitplaneSelectionInformation     // for block 1
+ BlockBitplaneSelectionInformation     // for block 2
+
…
+ BlockBitplaneSelectionInformation     // for block nBlock-1
}
```

Description:

Structure of a BlockBitplaneSelectionInformation:

```
BlockBitplaneSelectionInformation =
{
if (!IncludedInPreviousLayer)
{
EncodeTagTree()      //code the inclusion flag,variable bits
If (IncludedInCurrentLayer)
{
BitplaneDepth        // 5 bits, number of bitplanes
}
}
else
{
 InclusionFlag          //1 bits
}

If (IncludedInCurrentLayer)
{
 CodeNumExtraPasses()       //variable bits
 CodeNumAppendedBytes()  //variable bits
}
}
```

Description: code the following information to bitstream

- Whether there are extra coding passes of current block included in current bitstream layer
- If there are coding passes included and this is the first inclusion of this block, the number of total bitplanes is coded.

- If there are coding passes included, code the number of appended coding passes and the number of appended bitstream bytes.

2.5.4 Extractor

Two kinds of functionalities are provided to extract a sub bitstream from an existing bitstream. They are simple bitstream layer dropping and bitstream layer regeneration.

2.5.4.1 Simple Bitstream Layer Dropping

Usually, a bitstream generated by this codec may have multiple bitstream layers, each layer with a progressively increasing resolution and/or bitrate. If the setting of one of these layers is suitable for our application, the simplest way to generate the sub-bitstream is to drop all the unnecessary bitstream layers directly in each GOP and trans-write the remained layers to the target bitstream.

2.5.4.2 Bitstream Layer Regeneration

When the existing bitstream keeps the R-D information in itself, the relative importance of each bitstream pieces is kept. More flexible bitstream regeneration can be performed again at the extractor, with any new bitstream layer settings, including number of bitstream layers and the (S,T,R) settings for each layer.

The process consists of two parts for each GOP: parsing all the bitstream layers in the existing bitstream and regenerating the multiple-layer output bitstream. The regenerating process of the multiple-layer bitstream is the same with that process at the encoder.

3 Additional Modules

3.1 Base layer

The minimum spatial, temporal and SNR resolution bitrate point represents a Base Layer on which additional Enhancement Layers (3D subband video coding) are built to provide scalability up to a desired maximum spatial, temporal and SNR resolution. The introduction of a Base Layer in 3D subband scalable video coding can efficiently improve the coding performance of the lowest quality point and it also provides compatibility with the existing video standard for the lowest quality bit-rate point.
H.264/AVC is adopted as the Base Layer Codec and hierarchical B coding structure, which can be implemented with stored B techniques, is enabled. Hierarchical B coding structure allows each B picture of Base Layer to match temporally corresponding high-pass subbands in Enhancement Layers. It therefore enables the frames to share the similar MCTF structure decomposition and this correlation can be efficiently exploited.

Figure 12 shows the diagram embedding the Base Layer codec into 3D subband video coding. When a 5-level temporal decomposition is used for input video with CIF and 30HZ format, input video for the lowest quality point with QCIF and 7.5HZ format will be introduced after a 2-level temporal transform.

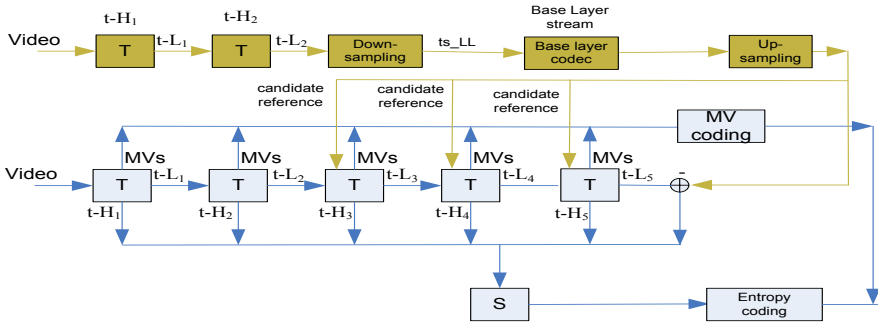


Figure 12: 3D subband video encoder with Base Layer codec embedded

Three kinds of correlations between Enhancement Layer and Base Layer need to be exploited.

3.1.1 Motion Information Prediction from Base Layer

In 3D subband coding with lifting structure, Haar and 5/3 temporal transforms are similar to the motion compensation of conventional P and B pictures respectively. The temporal high-pass subbands within the MCTF decomposition, which are beyond the maximum temporal level of those decomposition used to generate the input video of the Base Layer codec, can always be temporally matched with a corresponding inter-picture in Base Layer video.

- Motion information derived from the Base Layer codec can be used as additional candidate predictors for estimating motion information of the blocks in the temporally corresponding high-pass subband at Enhancement Layer.
- On the other hand, this motion information can also be used as the true motion vectors of the blocks in those temporally corresponding high-pass subbands when the current subband is of the same spatial resolution as Base Layer's.

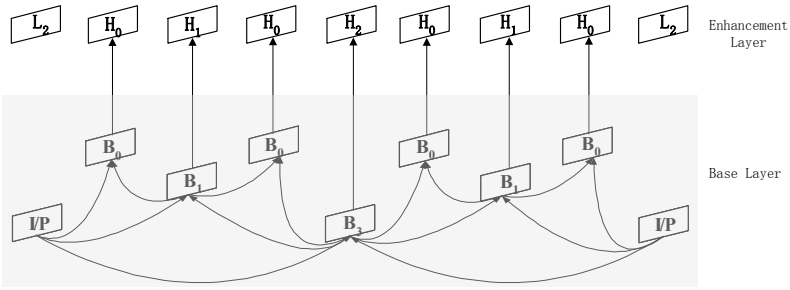


Figure 13: The motion information for B pictures in Base Layer provide extra candidate predictors or true motion information for the temporally corresponding high-pass subbands at Enhancement Layer.

A new mode "BASESKIP" is introduced to efficiently take advantage of motion information predictors from the Base Layer. For this mode, macroblock partition mode and motion vectors are all derived from the corresponding mapped macroblock partition and motion vectors in the Base Layer and no extra information is to be appended in bit-stream. When "BASESKIP" mode is enabled, the corresponding UVLC code number for mode type is increased by 1. The code number for "BASESKIP" is set to 0.

When tow-level spatial scalability is supported, the 4x4, 4x8, 8x4, 8x8 sub-macroblock partitions for 8x8 macrobock partition are converted into 8x8, 8x16, 16x8, 16x16 macroblock partition predictors respectively and all larger macroblock partitions for 8x8, 16x8, 8x16 and 16x16 are converted into 16x16 macroblock partition mode predictors. Its corresponding map rule can be illustrated in Figure 14. This feature can be enabled by *BL_MotionInfoPred* flag.

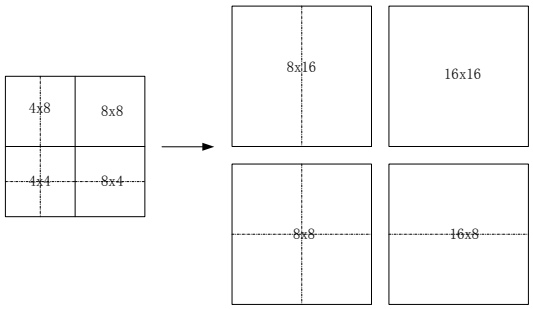


Figure 14: Macroblock partitions mode map rule to generate the predictors for higher spatial resolution

3.1.2 Predicting the Intra Region from Base Layer

The reconstructed video from Base Layer codec can be taken as predictors, which are up-sampled with DWT when the Enhancement Layer has higher spatial resolution, for those temporally poorly matched blocks of high pass subbands and the whole low-pass subbands. This feature can be enabled by *BL_RecPred* flag.

3.1.3 Residual Texture Prediction from Base Layer

The reconstructed residue from Base Layer can also be taken as predictors for temporal high pass subband after prediction step, which are also up-sampled with DWT when the Enhancement Layer has higher spatial resolution, for the whole temporally corresponding high-pass subband except for some blocks that have taken the reconstructed video from Base Layer as predictors. This feature can be enabled by *BL_ResPred* flag.

3.2 In-band (T+2D+T) Coding

The codec supports the 2D+T+2D scheme, which is activated when the pre-spatial transform is set as non-empty in figure 2. In the 2D+T+2D scheme, MCTF is applied to each spatial sub-band generated by pre-spatial transform. It can be considered like applying motion compensation on different resolutions. Compared to motion prediction in image domain or at full resolution, the in-band scheme suffers a loss in coding performance at full resolution, one reason being that signals of low resolution use only low resolution references as prediction, which has comparatively low quality. However, if high resolution signals are involved in the prediction for low resolution, drifting occurs when decoding low resolution video, because in that case high resolution signals are not available at the decoder. A leaky motion compensation method, based on frame level and mode-based temporal filtering method based on macroblock level, is used to make a trade-off between drifting reduction and improving coding performance.

3.2.1 The Interpolation Reference Frames with Different Quality for Inband MCTF

When coding the lower resolution sub-band, the interpolation reference frames with different quality can be formed with the additional higher resolution sub-bands. With the example of two-level pre-spatial transform, we will get seven sub-bands after pre-spatial transform, showed as Figure 15. Suppose that the original video is of 4CIF format. Then sub-band 0 corresponds to QCIF video, and sub-bands 1, 2,3 are of CIF video respectively. The reference of a sub-band can be directly formed with the current or lower resolution's information. We call it low quality reference. To improve the coding efficiency, besides the sub-bands of current or lower resolutions, sub-bands of higher resolutions can also be used to generate the high quality references. But using higher resolution's information will bring mismatch error when decoding lower resolution's video. For sub-band 0, a parameter in the configuration file, Num_AddedLevel_SBand0 is used to control how many levels of higher resolution information are used to generate the prediction. For example, Figure 16(a)(b)(c) demonstrates how reference of sub-band 0 is generated when Num_AddedLevel_SBand0 is set as 0, 1, and 2 respectively. Note that when Num_AddedLevel_SBand0 is 0, the high quality reference actually equals the low quality reference. Similarly, for the sub-bands 1, 2 and 3, the MCTF can be done with the information of four sub-bands 0, 1, 2 and 3 (The Num_AddedLevel_SBand in the configuration file is set as 0 in configure file), or with the information of all seven sub-bands (The Num_AddedLevel_SBand is set as 1 in configure file). Then two different quality interpolation reference frames of each band with can be formed. For the sub-bands 4, 5 and 6, the high quality interpolation reference frames of each band can be formed with all seven bands that are available at the decoder for decoding full resolution video. In the software, each spatial sub-band reference frame has two 1/4 interpolation buffers. When Num_AddedLevel_SBand0 and Num_AddedLevel_SBand are set non-zero, these two interpolation reference frames have different quality. For the highest level resolution sub-bands 4, 5 and 6, the high quality interpolation reference frames are always used for motion compensation. For other level resolution sub-bands, both of two interpolation reference frames are used for motion compensation.

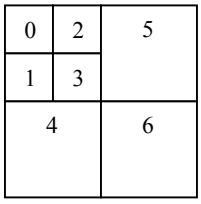
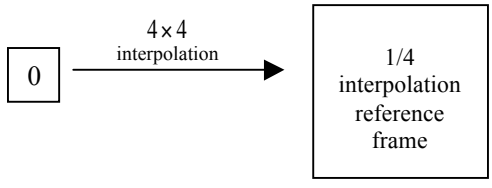
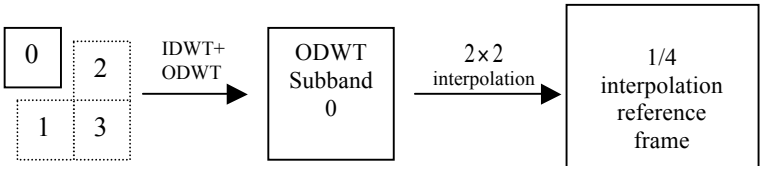


Figure 15: Spatial sub-bands after pre-spatial transform



(a)



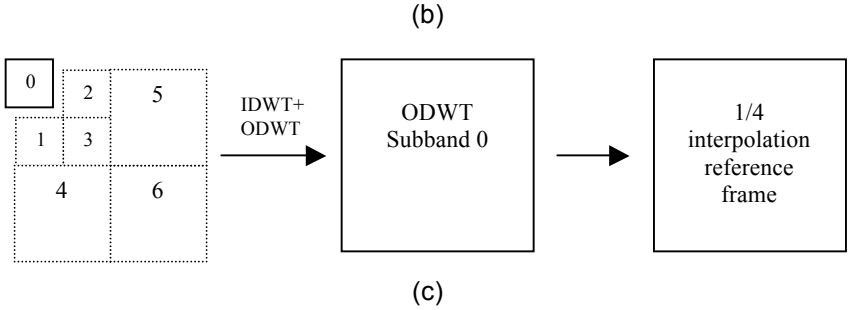


Figure 16: the interpolation reference frame forming of sub-band 0

3.2.2 Motion Estimation

Even though the independent motion estimation can be applied to each spatial sub-band, to save the coded bits for motion information, in the software all sub-bands use one set of motion vector, which is searched in the highest level spatial sub-band. This set of motion vector needs to be scaled according to the spatial resolution of each sub-band. With the example of two-level pre-spatial transform, we will get seven sub-bands after pre-spatial transform. The motion vector of sub-band 0 will be used to other six sub-bands. Since the sub-bands 1, 2 and 3 have the same resolution as the sub-band 0, the motion vector came from sub-band 0 can be used directly without scaling. Sub-bands 4, 5 and 6 have larger resolution compared with sub-band 0, so the scaling with the factor 2 is needed. If the block size for the sub-band 0 motion estimation is from 4×4 to16×16, the block size for sub-bands 4, 5 and 6 is from 8×8 to32×32.

3.2.3 Leaky Motion Compensation

When the Num_AddedLevel_SBand0 is set as non-zero in the configure file, the leaky motion compensation is used to the spatial sub-bands at the lower resolution. When encoding these sub-bands, the difference between the high quality interpolation reference and the low quality interpolation reference will be attenuated by a leaky factor. The high quality interpolation reference will be partly used to form a better interpolation reference frame of sub-band 0. For the spatial sub-bands at the highest resolution, the high quality interpolation references are always used for motion compensation at both encoder and decoder. Supposing that the N-level temporal filtering is used, the leaky motion compensation can be described as follow.

At encoder we have equations (3) and (4):

$$
\begin{aligned}
H_n^{\,i+1} &= L_{2n+1}{}^{i} - P(L_{2n}{}^{i} + L_{2n+2}{}^{i}) \\
P(L_{2n}{}^{i}, L_{2n+2}{}^{i}) &= \frac{1}{2} MC((1-\alpha_i)\times R_l(L_{2n}{}^{i}) + \alpha_i \times R_h(L_{2n}{}^{i}), MV_{2n+1->2n}) \\
&\quad + \frac{1}{2} MC((1-\alpha_i)\times R_l(L_{2n+2}{}^{i}) + \alpha_i \times R_h(L_{2n+2}{}^{i}), MV_{2n+1->2n+2}) \\
&\quad i = 0,\ldots, N-1
\end{aligned}
\qquad (3)
$$

$$L_n^{i+1} = L_{2n}^{\ i} + U(H_{n-1}^{\ i+1} + H_{n+1}^{\ i+1})$$
$$U(H_{n-1}^{\ i+1} + H_{n+1}^{\ i+1}) = \frac{1}{4}((MC(H_{n-1}^{\ i+1}, MV_{2n->2n-1}) + MC(H_{n+1}^{\ i+1}, MV_{2n->2n+1})) \quad (4)$$
$$i = 0,..., N-1$$

Where, $\{L_n^{\ 0}\}$ denotes the sequence of the sub-band at the lower resolution after pre-spatial transform. $\{L_n^{\ i}\}$ and $\{H_n^{\ i}\}$ denote the low-pass sequence and high-pass sequence after i -level temporal filtering. $\{\tilde{L}_n^{\ i}\}$ and $\{\tilde{H}_n^{\ i}\}$ are the corresponding reconstructed low-pass and high-pass sequences. $U()$ and $P()$ denote the prediction and update step. $R_l()$ and $R_h()$ are the operations generating the low quality interpolation reference and the high quality interpolation reference respectively. $MC()$ means motion compensation process that generates the current frame's prediction from its consecutive frame. $MV_{2n->2n-1}$ and $MV_{2n->2n+1}$ are the sub-pixel motion vectors from an even frame to the forward and backward adjacent odd one based on the high quality interpolation references. $\alpha_i (0 \le \alpha_i \le 1)$ is the leaky factor for the i[th] level temporal filtering.

At the decoder, we have equations (5), (6) for decoding low resolution video and equation (5), (7) for decoding full resolution video.

$$\tilde{L}_{2n}^{\ i} = \tilde{L}_n^{\ i+1} - U(\tilde{H}_{n-1}^{\ i+1} + \tilde{H}_{n+1}^{\ i+1})$$
$$U(\tilde{H}_{n-1}^{\ i+1} + \tilde{H}_{n+1}^{\ i+1}) = \frac{1}{4}((MC(\tilde{H}_{n-1}^{\ i+1}, MV_{2n->2n-1}) + MC(\tilde{H}_{n+1}^{\ i+1}, MV_{2n->2n+1})) \quad (5)$$
$$i = 0, ..., N-1$$

$$\tilde{L}_{2n+1}^{\ i} = \tilde{H}_n^{\ i+1} + P(\tilde{L}_{2n}^{\ i} + \tilde{L}_{2n+2}^{\ i})$$
$$P(\tilde{L}_{2n}^{\ i}, \tilde{L}_{2n+2}^{\ i}) = \frac{1}{2}MC(R_l(\tilde{L}_{2n}^{\ i}), MV_{2n+1->2n}) + \frac{1}{2}MC(R_l(\tilde{L}_{2n+2}^{\ i}), MV_{2n+1->2n+2}) \quad (6)$$
$$i = 0, ..., N-1$$

$$\tilde{L}_{2n+1}^{\ i} = \tilde{H}_n^{\ i+1} + P(\tilde{L}_{2n}^{\ i} + \tilde{L}_{2n+2}^{\ i})$$
$$P(\tilde{L}_{2n}^{\ i}, \tilde{L}_{2n+2}^{\ i}) = \frac{1}{2}MC((1-\alpha_i) \times R_l(\tilde{L}_{2n}^{\ i}) + \alpha_i \times R_h(\tilde{L}_{2n}^{\ i}), MV_{2n+1->2n}) \quad (7)$$
$$+ \frac{1}{2}MC((1-\alpha_i) \times R_l(\tilde{L}_{2n+2}^{\ i}) + \alpha_i \times R_h(\tilde{L}_{2n+2}^{\ i}), MV_{2n+1->2n+2})$$
$$i = 0,..., N-1$$

If the value of α_i is close to 1, more high band information is used for a certain level temporal motion compensation of LL band, which leads to the best coding efficiency and on the same time, the less drifting error reduction. On the other hand, for α_i that is close to zero, the drifting error reduction is enhanced significantly at the cost of the reduced coding efficiency. The value of α_i can be set by the item of *Weight_For_TLevel* in the configure file.

3.2.4 Mode-Based Temporal Filtering

As an alternative of leaky motion compensation, we also introduce a method, mode-based temporal filtering, into in-band MCTF scheme. In this method, each macro-block uses either the low quality reference or the high quality reference to make the prediction, which is defined as a mode of macro0block. When employing MCTF on the macroblock

of lower resolution sub-band, Coding mode can be adaptively selected among the references at different layer. In the software, both of the different quality interpolation references enter the motion estimation module. Each reference is used to separately estimate one set of motion information. In order to save bits, one of them for motion compensation is selected for each macroblock.
Define

$$RD\_Ratio = \frac{Min(RD\_Cost_{low})}{Min(RD\_Cost_{high})} \tag{8}$$

$$RD\_Cost_{low} = \lambda_{motion} \cdot (R_{mv} + R_{\mathrm{mod}e}) + SAD_{L}(\mathrm{mod}e, MV)$$

$$RD\_Cost_{high} = \lambda_{motion} \cdot (R_{mv} + R_{\mathrm{mod}e}) + SAD_{H}(\mathrm{mod}e, MV)$$

Where, R_{MV} and $R_{\mathrm{mod}e}$ are the needed bits for coding the predicted motion vector and the partition mode of one macroblock, respectively. λ_{motion} is the Lagrange multiplier for searching motion vectors. $SAD_{L}(\mathrm{mod}e, MV)$ is the sum absolute difference between the original macroblock and its low quality prediction reference. $SAD_{H}(\mathrm{mod}e, MV)$ is the sum absolute difference between the original macroblock and its high quality prediction reference.
When RD_Ratio is greater than the threshold value, which can be set different value for different temporal level by the item Ratio_For_TLevel in the configuration file, the motion information based on the high quality interpolation reference will be chosen and the corresponding flag in the software will be set as 1. Otherwise, the motion information based on the low quality interpolation reference will be used and the corresponding flag will be set as 0. Both the motion information and the flag will be coded into the bit stream.

3.3 Wavelet Ringing Reduction

DCT-based coders suffer from blocking-artefacts which can be reduced by application of appropriate filters. E.g. in AVC, an in-loop de-blocking filter is specified. This filter improves the quality of the prediction signal as well as the reconstructed video. For MCTF schemes, similar artefact-adapted concepts are applied. AVC based as well as Wavelet based MCTF approaches are already using de-blocking filters for reconstruction. In Wavelet based video compression, ringing is the prominent artefact.

3.3.1 Description

In every reconstruction step of the inverse Wavelet-transform, quantization-noise of lossily-coded Wavelet-coefficients is convolved with the reconstruction filters. The resulting oscillating shape of these filters at different reconstruction scales becomes observable especially in regions of flat intensities (luma as well as chroma). Its magnitude is related to the quantization step-size.

In order to reduce these oscillating shapes, a 2D bilateral filter applied to each component of the output video is used. The bilateral filter is related to other powerful de-noising filters, but it has the advantage of being non-iterative. The output of the bilateral filter is a weighted average given by

$$\hat{x}[n,m] = \frac{\sum_{(k,l)\in\chi} W_1\left(\sqrt{(n-k)^2+(m-l)^2}\right) W_2\left(|x[n,m]-x[k,l]|\right)\cdot x[k,l]}{\sum_{(k,l)\in\chi} W_1\left(\sqrt{(n-k)^2+(m-l)^2}\right) W_2\left(|x[n,m]-x[k,l]|\right)},$$

where $x[n,m]$ and $\hat{x}[n,m]$ are the input and output image of the filter, respectively, χ is a neighbourhood around (n,m) and $W_1(\cdot)$ and $W_2(\cdot)$ are weight functions. Here, $W_1(\cdot)$ accounts for the spatial distance of the considered pixel to the filter-output location (n,m). $W_2(\cdot)$ gives weight according to the distance in magnitudes of the considered pixel and the one at the output location. These weight functions are

$$W_1(d) = \exp\left(\frac{d^2}{{\sigma_S}^2}\right) \text{ and } W_2(d) = \exp\left(\frac{d^2}{{\sigma_V}^2}\right).$$

In the bit-stream extraction step, the least significant bit-planes of subbands are disregarded. The most significant disregarded bit-plane b_n of each non-zero subband is used, to estimate the quantization noise in the reconstructed sequence. The information on the quantization is derived in the entropy decoding stage and passed through the MCTF reconstruction. The de-ringing filter is applied to the reconstructed video. The amplitude weight is determined by ${\sigma_V}^2 = c_2 \cdot (1.5133)^{-2s} \cdot \frac{1}{N}\sum_{n=1}^{N} 2^{2\bar{b}_n}$, where $\bar{b}_n$ is the information on most significant disregarded bit-planes propagated through the MCTF and s is the spatial scaling factor ($s = 0$: reconstruction to full resolution, $s = 1$: half resolution, …). The parameters c_2 is the filter strength specified in the configuration file by PostFilterStrength, see 3.9.7.1. The spatial parameter of the post-filter ${\sigma_S}^2$ is given by PostFilterSize, see 3.9.7.2. The neighbourhood χ of the summations of the de-ringing filter is a square filter-mask of $\left(2\cdot\text{round}(1.5\cdot\sigma_S)+1\right)^2$ pixels.

3.4 Intra mode prediction

The intra-prediction exploits the spatial correlation within the frame to predict intra areas before entropy coding. Intra-block prediction functions are interleaved in the encoder as shown at Fig. 1.

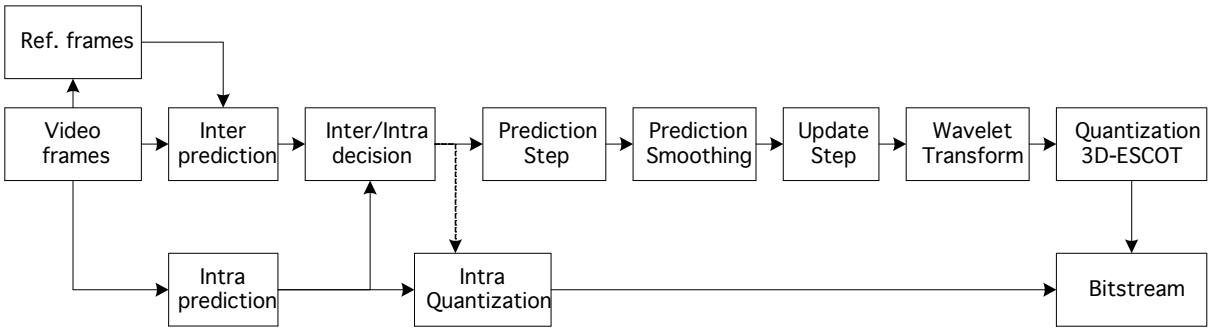


Fig. 1 Encoder with intra prediction

In intra-prediction module, an intra-predicted macroblock is obtained using data from the current frame. A macroblock is divided into sixteen subblocks for luma component and four subblocks for chrominance component. The scheme utilizes a pixel value at left-topmost position for each subblock as shown in Fig. 2. All pixels inside the subblock are generated by interpolating from four nearest pixel values, e.g. pixels *a, b, e*, and *f* from Fig. 2.

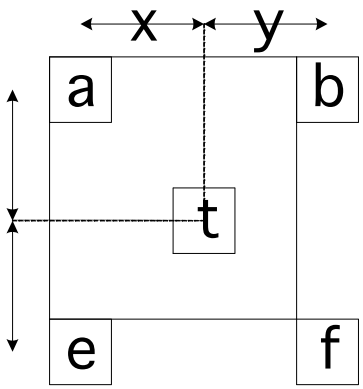


Fig. 2 Simple diagram of pixel interpolation for a subblock.

The interpolated pixels are used for the prediction and the original pixel values are subtracted from the predicted ones. The values are further transformed by Hadamard transform for additional reduction of spatial correlation. Finally, the intra-predicted pixels are quantized by a quantisation parameter specified in a configuration file
After intra-/inter-prediction, the lower-cost predicted modes are selected.
Sixteen luma values and eight chroma values for each macroblock of intra mode have to be included in the final bit-stream.

At the decoder side, block artifacts between inter-predicted blocks and intra-predicted blocks are additionally by application of a bi-linear 1:2:1 filter over the block boundaries.

3.5 STool scheme

STool scheme was proposed by University of Brescia at the Palma meeting and it has been implemented as an extension of the original MSRA software.

3.5.1 General description

The main characteristic of this (SNR-spatial-temporal) scalable video coding scheme is its native ability to deal spatial scalability. This in turns implies a spatial resolution driven complexity scalability. This native spatial scalability is implemented within a 2D+T+2D approach (a mixture of the 2D+T and T+2D approaches) where the lower spatial resolution information (at spatial level s) is used as a base-layer on which to predict the finer resolution spatial level s+1. For a 4CIF-CIF-QCIF implementation three different coding-decoding chains are used (Figure 17). Each chain acts at a different spatial scale level and presents temporal and SNR scalability. The idea is to use the decoded information (at a suitable quality) at a lower spatial level in order to predict the higher resolution info. In principle this can be done in different ways. In STool the idea consists in possibly predict the MCTF temporal subbands at spatial level s+1, f_{s+1}, starting from the decoded MCTF subbands (before MCTF^{-1}) at spatial level s, dec(f_s). In order to work at the same spatial level s we perform the prediction on the low-pass subbands extracted after 1 DWT level on f_{s+1}, i.e. $\mathrm{dwt}_L(f_{s+1})$. In fact at that point the predicted subbands and the predicting ones have been subject to the same number and type of transformations, but in a different order (t+2D and 2D+t respectively). The prediction error Δf_s = dec(f_s) –$\mathrm{dwt}_L(f_{s+1})$ can now substitute $\mathrm{dwt}_L(f_{s+1})$ in the coding scheme (Figure 18). The produced frame at spatial level s+1 is called a *delta*-frame, and the approach is called "Substi-Tool", or more concisely STool.

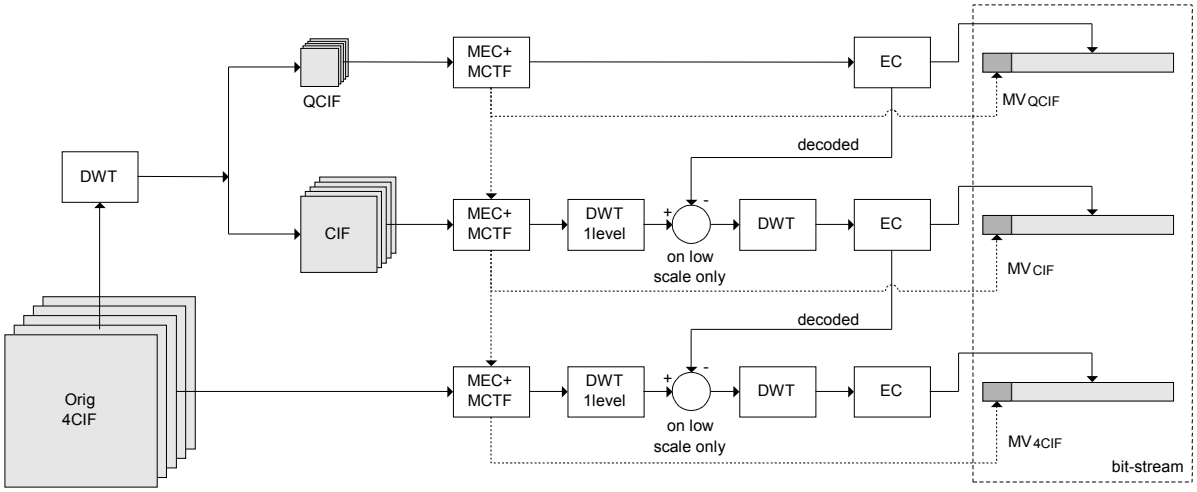


Figure 17 STool: overall coding scheme

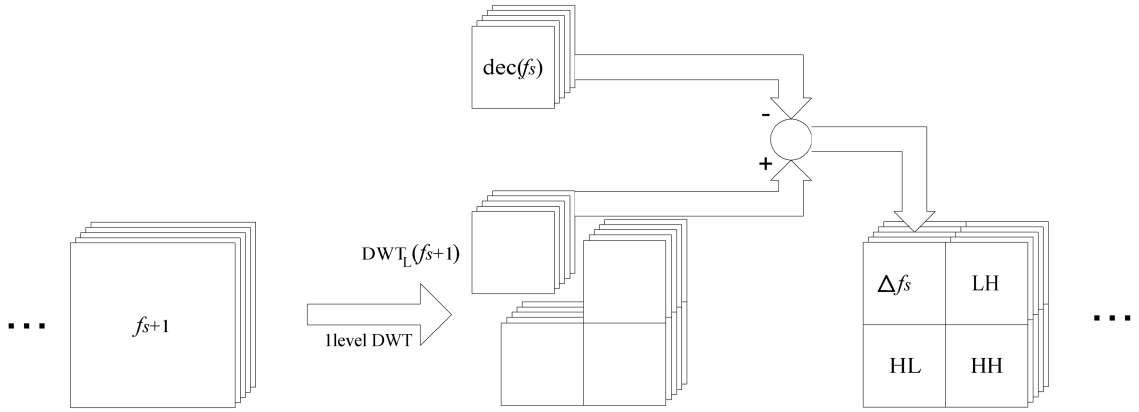


Figure 18 STool: prediction detail

For more details, refer to input documents m11368 and m11378, ISO/MPEG70, Palma de Mallorca, and.Vidwav evaluation software manual

During the #73 MPEG meeting in Poznan (Poland), it has been decided (see N7333) to have the vidwav reference software incorporate the extensions of the multiresolution layered architecture proposed in Palma (m11368) . The new reference software has been released under the new MPEG license rule two weeks after Poznan meeting. A technical description of this software has been provided to VidWav AHG three weeks after MPEG meeting. The Software is available via Aachen's CVS repository and the access is limited to MPEG members.
The purpose of this session is to provide a first description of the usage of the Vidwav evaluation software. This includes information about the encoder and decoder input parameters, syntax, compilation issues, and additional information with regards to best usage and configuration of this software.

3.5.2 How STool works

All the new part can be plugged and unplugged from the original version (released after Busan meeting) by activating/deactivating the following defines in module "EncDec_PreCom.h":

```
#define __ENABLE_THREE_LIFTING_STEPS_IRISA__
#define __DEFAULT_SPATIAL_FILTER_CHROMA_IRISA__
#define __STOOL
```

If not available, the spatially sub-sampled versions of the video must be generated starting from the original video. In this implementation case, we used the 3LS wavelet filter described in m10904 since it introduces less aliasing with respect to 9x7.

The real encoding process starts from the lowest spatial resolution (e.g. QCIF) by running the MSRA encoder, T+2D scheme, with only one spatial resolution (the original one) and the desired temporal scalability. From the compressed stream the point which will be used as prediction for the upper spatial layer is extracted and decoded. During the decoding process the temporal subbands are saved in a file identified by the configuration parameter "pred_band_fileD".

The encoding of the next spatial level is exactly as the previous one with only one difference. In fact, after the first spatial decomposition, the LL spatial sub-band is predicted by using those saved during the last decoding process. The file name from which the subbands are loaded is specified by the configuration parameter "pred_band_fileE".

Note: In the current implementation, it is required to decode, for example, all QCIF video before encoding the CIF one. This is simply an implementation issue; it is not an architectural problem. A better implementation should enable a parallel coding scheme for QCIF and CIF sequences (and, eventually, for 4CIF ones), using the decoded QCIF temporal subbands to predict the CIF subbands.

3.6 General information

3.6.1 Project References

For further information, it is recommended that the users access the ISO web site at www.iso.int. The evaluation software described in these pages can be downloaded from the the Aachen CVS server, which is hosted at garcon.ient.rwth-aachen.de. It can be accessed using WinCVS or any other CVS client using the "pserver" method. The path is "/cvs/vidwav". For example, with a command line client, you can checkout the software using the following commands:
cvs -d :pserver:vidwavuser@garcon.ient.rwth-aachen.de:/cvs/vidwav login
cvs -d :pserver:vidwavuser@garcon.ient.rwth-aachen.de:/cvs/vidwav co <module>
(you can use "co" or "checkout")
The passwd for vidwavuser is: ViDwAv05
The server is configured to allow read access only to vidwavuser. Currently, write access is provided for Mr. Jizheng Xu (jzxu@microsoft.com) , Dr.-Ing Mathias Wien (wien@ient.rwth-aachen.de), Michele Brescianini (michele.brescianini@ing.unibs.it), Sebastian Brangoulo (sebastien.brangoulo@enst.fr) and Beatrice Popescu (pesquet@tsi.enst.fr). Anyone who requires write access should please contact Dr.-Ing Mathias Wien (wien@ient.rwth-aachen.de) who will provide a personal login ASAP. Currently, the repository hosts the module "main2", which corresponds to the current available version of the Vidwav evaluation software.

3.6.2 Authorized Use Permission

The software package contains source code header comments disclaimer text that describes the terms associated with the use of the software and clarifying its copyright and patent right issues.

3.6.3 Points of Contact

3.6.3.1 Information

For general inquiries with regards to the AHG on Exploration in Wavelet Video Coding, users may contact Mr. Jizheng Xu (t-jzhxu@microsoft.com) and Dr. Béatrice Pesquet-Popescu (beatrice.pesquet@enst.fr) for further information. Certain information can also be provided through the ITU (www.itu.int) and ISO (www.iso.int) websites.

3.6.3.2 Coordination

Software coordination is performed by Mr. Jizheng Xu (jzxu@microsoft.com). For further information on key contributors to the wavelet evaluation software implementation please check the file "ChangeList.txt" within the evaluation software package.

3.6.4 Organization of the manual

In Section 3.7, a list of acronyms and abbreviations concerning the Vidwav evaluation software is provided. This is followed by instructions of how to install and compile the evaluation software under the Windows environments (Unix/Linux based platforms are not yet officially supported). The use of the encoder is described in Section 3.9, while all encoder/extractor/decoder parameters are analyzed in Section 3.10. The use of the extractor is described in Section 3.10.10, while the use of the decoder is described in Section 0. Finally Section 3.14 presents some of the output reports generated by the different modules of this software distribution. Example of configuration files, which implement different encoding architectures, can be found in the document ConfigExample.doc in current version of VidWav evaluation software.

3.7 Acronyms and Abbreviations

MCTF: Motion Compensated Temporal Filtering
GOF: Group of Frames
UMCTF: Unconstrained MCTF
OBMC: Overlapped Block Motion Compensation
EBCOT: Embedded Block Coding with Optimal Truncation
BL: Base Layer
MGS: Medium Grain Scalability

3.8 Installation and compilation

3.8.1 Windows using MS Visual Studio 6

The software package contains a Visual Studio 6 workspace named "MCWCodec.dsw". This workspace includes four projects:

- Extractor: Vidwav evaluation extractor;
- MCWDEC: Vidwav evaluation decoder;
- MCWEnc: Vidwav evaluation encoder;

- ShareLib: Shared libraries between above-mentioned projects.

Select the desired project and “Debug” or “Release” mode. Compilation will respectively create the binaries “Extractor.exe”, “MCWDec.exe” or “MCWEn.exe” in the “bin” directory.

3.8.2 UNIX and Windows using gcc (GNU Compiler Collection)

For the time being the software package does not contain any specific scripts such as Makefiles.

3.9 Using the encoder module

This section provides a detailed description of the Vidwav evaluation encoder's usage.

3.9.1 Encoder syntax

```
MCWEn.exe <configuration_file> <baselayer_flag>
```

Options

<configuration_file> Path to the configuration file, specified in section 3.10.
<baselayer_flag> 1 if the Base Layer feature is requested, 0 otherwise.

See section 3.10 for a description of all parameters.
Supported video file formats: RAW YUV 4:2:0 (.yuv).
Examples of usage:

```
MCWEn.exe NewCfg.cfg 0
MCWEn.exe baselayer\Foreman(CIF)_refine.cfg 1
```

3.9.2 Encoder ouput

When running the encoder, the encoder will display on screen general information about the encoding process. The output information generated may look as follows:

```
Scale for TBand 0: 0.718800
…
Scale for TBand 4: 5.062500
Displays temporal subband scaling.
//////////////////////////////////////////////////
//            START ENCODING                //
//////////////////////////////////////////////////
Total Operation 90
   SPATIAL_TRANS SBand:    Y Rect(    0,    0,  352,  288) TBandIdx:    2
…
   SPATIAL_TRANS SBand:    V Rect(   88,   72,   88,   72) TBandIdx:    9
Total Operation 19
  ENTROPY_CODING SBand:    Y Rect(    0,    0,   44,   36) TBandIdx:    2
  …
  ENTROPY_CODING SBand:    V Rect(  132,  108,   44,   36) TBandIdx:    9
Built T + S + E Scheme successfully
```

displays the building of the different encoding modules (temporal, spatial and entropy coding).

```
============  INSERT ONE ORIGINAL FRAME ============
(  0,  ORG::  0,  ORG) ===>||    Module( T, 1,  0)
    ME{S:0,E:0,V:0,}       NULL->       NULL<-(  0,  ORG::  0,  ORG)
      FrameUpdate :        NULL->       NULL<-       NULL
============  INSERT ONE ORIGINAL FRAME ============
(  1,  ORG::  1,  ORG) ===>||    Module( T, 1,  1)
…
```

displays the insertion of new frames into the encoder pipe (push\pull model). Original frame (0, ORG) is sent to transformation temporal module (T, 1, 0). Prediction and Update steps are called but not all needed frames are ready to be processed.

```
============  INSERT ONE ORIGINAL FRAME ============
(  2,  ORG::  2,  ORG) ===>||   Module( T, 1,  2)
   ME{S:0,E:0,V:0,}(  0,  ORG::  0,   L)->(  1,  ORG::  0,   H)<-(  2,  ORG::  2,  ORG)
Temporal Level: 0; Frame No. 1
Frame Motion Info Bits: 2042, cost: 208383.343750
Total Bits Length: 256 bytes(2048 bits)
     FrameUpdate :       NULL->(  0,  ORG::  0,   L)<-(  1,  ORG::  0,   H)
============  INSERT ONE ORIGINAL FRAME ============
…
```

displays the process of motion estimation on the triplet (0,1,2). Prediction and Update steps are called some motion information are shown.

```
Entropy Module Buffer Status.
  0 (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0),
Entropy Module Buffer Status.
  1 (B= 0, F= 1), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0),
Entropy Module Buffer Status.
  0 (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0),
…
(min_rd,max_rd,rd_threshold)=(0,32767,16384)
…
```

displays the entropy encoder module building (block by block).

```
FormPacket(0,2,0,16384)=(656,656)
…
 Trying rd_threshold=16384, actual_bytes=1633
(min_rd,max_rd,rd_threshold)=(0,16384,8192)
FormPacket(4,4,2,9403)=(28,479)
…
MGS overhead: 6920 bits.
 Frequency for Extra BlockBody Bytes:
    0     2    55    87    96    66    35    21     5     1     1     0     0     0     0     0
…
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
 Frequency for Delta Slope:
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
…
Press any key to continue
```

displays the process of layer building and bitstream construction.

3.10 Parameters

All parameters names are case independent. Comments starts with a semi-colon ";". Multi-lines parameters are specified by a final anti-slash "\". Please note that Vidwav evaluation encoder, extractor and decoder share the same configuration file. If not specified, quite universal (default) options are used.

3.10.1 File Input/Output Related Parameters

3.10.1.1 source_file

Class: Text
Description: Path of the RAW YUV 4:2:0 input source file, without the .yuv extension
Example: FOREMAN_352x288_30_orig_01

3.10.1.2 compress_file

Class: Text
Description: Path of the Wavelet evaluation software output compressed file
Example: compress

3.10.1.3 frame_width

Class: Integer
Description: Width in pixels of the input luminance frames
Example: 352

3.10.1.4 frame_height

Class: Integer
Description: Height in pixels of the input luminance frames
Example: 288

3.10.1.5 frame_number

Class: Integer
Description: Number of input frames of the video sequence to be coded
Example: 300

3.10.1.6 frame_rate

Class: Integer
Description: Input file frame rate, in frames per second
Example: 30

3.10.1.7 level_s

Class: Integer
Description: Number of supported dyadic spatial scalability levels
A value of 0 means that no spatial scalability is supported and the video can be decoded only at the original resolution.
Note: This value should be lower or equal to the minimum number of spatial transforms; see the spatial decomposition section 3.10.4.

Example: Typical values are 1 or 2. As an example, a value of 2 allows a 4CIF compressed video sequence to be decoded at 4CIF, CIF or QCIF resolution.

3.10.2 Temporal Filter and MCTF Related Parameters

The temporal filter cannot be changed with the configuration file; 5/3 MCTF is used by default.

3.10.2.1 temporal_number

Class: Integer
Description: Number of temporal decomposition levels
Allows a maximum temporal dyadic scalability corresponding to this value.
Example: Typical values are 4 or 5, for smooth-motion sequences.

3.10.2.2 MotionEstimation

Class: Structured text
Sequence of "ME<StartS=$\mathbf{ss}_i$ & EndS=$\mathbf{es}_i$ & lambdas=$\boldsymbol{\lambda}_i$ & SearchRange=$\mathbf{sr}_i$>" whitespace-separated patterns, where $\mathbf{ss}_i$, $\mathbf{es}_i$, $\boldsymbol{\lambda}_i$ and $\mathbf{sr}_i$ are integers. The sequence length is equal to the number of temporal levels, *temporal_number*.
Description: Description of the motion estimation parameters for each temporal level
For one temporal level, the motion estimation is done in a spatially layered way, from the coarsest $\mathbf{ss}_i$ spatial resolution to the finer one $\mathbf{es}_i$. The $\boldsymbol{\lambda}_i$ controls the block cost function $D + \lambda R$ used in the motion estimation algorithm. The search range in pixels is specified by $\mathbf{sr}_i$.
Note: The layered motion estimation is unlikely expected to work, since the introduction of the AVC fast ME algorithm in the Vidwav encoder; therefore $\mathbf{ss}_i$ should be equal to $\mathbf{es}_i$ in order to avoid crashes.
Example: The motion estimated with "ME<StartS=0 & EndS=0 & lambdas=16 & SearchRange=64>" is done at the original resolution with λ=16 and with a search range of 64 pixels. The motion estimated with "ME<StartS=1 & EndS=1 …" is done at the reduced resolution, QCIF for instance if the input video sequence is at CIF resolution.

3.10.2.3 OBMC_TYPE

Class: list of comma-separated Booleans of length *temporal_number*
Description: Enables OBMC for a particular temporal decomposition level.
Example: 1,1,1,1,1 is a default value
Options
0 Overlapped Block Motion Compensation deactivated
1 Overlapped Block Motion Compensation activated

3.10.2.4 edu_update

Class: Boolean
Description: Energy distributed Update method
Example: 1 is a default value.
Options
0 In this update technique, the inversion of one motion vector creates one motion vector.

Original motion vectors (dense motion field) are parsed in raster scan order. The motion vector is "applied" to get the reference point (in quarter pixel precision) reached by this motion vector. Then the nearest integer pixel position is set for the opposite motion vector.
If a motion vector has already been set for a point, the new motion vector is not taken into account.

1 In this update technique, the inversion of one motion vector generates one to four motion vectors (depending on the pixel precision of the reached sub-pixel position). Original motion vectors (dense motion field) are parsed in raster scan order. The (one, two or) four integer pixel positions around the point reached by the motion vector are affected. Weighting factors are applied, depending on the reached sub-pixel position.
This technique is the so-called “Barbell” update, using a bilinear interpolation.
OBMC is not applied in the update step, but still applied in the prediction step.

3.10.3 Bitstream Generation Related Parameters

3.10.3.1 MGS_SCALABLE

Class: Boolean
Description: The Vidwav evaluation software can work in two modes: Layered or Medium Grain Scalable (MGS). This flag toggles the MGS mode.
Example: 0 is a default value.
Options
0 Layered Mode: the video can only be decoded at a given (Spatial, Temporal, SNR) scalability point that belongs to the fixed set of predefined layers.
1 MGS Mode: the video can be decoded at almost any (S, T, R) point.

3.10.3.2 bitstream_layers

Class: Integer
Description: Number of bitstream layers in the output final bitstream
Note: If the *MGS_Scalable* parameter is set to true, this parameter should be equal to 1.

3.10.3.3 BitStrmLayers

Class: Structured text
Sequence of “Layer<S=$\mathbf{s_i}$ & T=$\mathbf{t_i}$ & Rate=$\mathbf{r_i}$ & EndSs=$\mathbf{ess_i}$>” white space-separated patterns, where $\mathbf{s_i}$, $\mathbf{t_i}$, $\mathbf{r_i}$ and $\mathbf{ess_i}$ are integers. The sequence length is equal to the number of bitstream layers, *bitstream_layers*.
Description: Description of each bitstream layer
A layer is a target decoding point at a spatial resolution $\mathbf{s_i}$ (0 is the original resolution), a temporal resolution $\mathbf{t_i}$ (0 is the original frame rate) and a bitrate $\mathbf{ess_i}$ in kbits/s. The $\mathbf{ess_i}$ parameter is a comma-separated list of integers, which refer to the spatial resolution of the motion layers used for the layer. The length of $\mathbf{ess_i}$ is equal to the number of temporal levels *temporal_number* minus the temporal resolution $\mathbf{t_i}$. In case of layered motion estimation, this parameter allows to choose which spatial motion layer to use for this bitstream.

Example: With an input CIF 30 Hz sequence, with 4 levels of decomposition: "Layer<S=0 & T=0 & Rate=512 & EndSs=0,1,1,1>" describes a CIF 30 Hz at 512 kbits/s layer, with first motion layer estimated at CIF resolution and others at QCIF resolution. With the same configuration, "Layer<S=1 & T=1 & Rate=128 & EndSs=1,1,1>" describes a QCIF 15 Hz at 128 kbits/s layer, where all motion layers are estimated at QCIF resolution.
Note: If the *MGS_Scalable* parameter is set to true, this parameter should be equal to the finest desired layer, for instance "Layer<S=0 & T=0 & Rate=8000 & EndSs=0,1,1,1>" to have an original spatial resolution and original frame rate 8000 kbits/s bitstream.

3.10.3.4 block_width

Class: Integer
Description: ESCOT coding block width
Example: 64 is a default value.

3.10.3.5 block_height

Class: Integer
Description: ESCOT coding block height
Example: 64 is a default value.

3.10.3.6 block_frame_number

Class: Integer
Description: ESCOT coding block depth
Example: 4 is a default value.

3.10.3.7 quantization_step_size

Class: Float
Description: Quantization step size
Example: 1.0 is a default value.

3.10.3.8 ESCOT_BlockDepth

Class: List of comma-separated Integers of length *temporal_number*
Description: EBCOT coding block depth for each temporal subband
Example: 4,4,4,4,4. If not specified, value block_frame_number is used as default.

3.10.3.9 GOP_Size

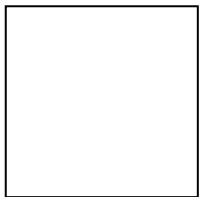
Class: Integer
Description: GOP window size, in frames used to perform the RD cut operation
Example: 64 is a default value.

3.10.4 Spatial Filter Related Parameters

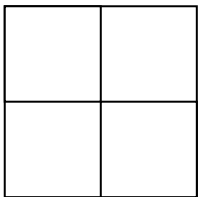
3.10.4.1 PRE_SPATIAL_DECOMPOSITION

Class: Structured text

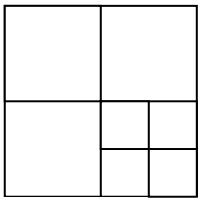
Recursive pattern P where P can be a terminal leaf “E<>” or a decomposition leaf “S<>PPPP”. Spatial filter can be chosen by using the parameter “Filter=W” inside brackets <> where W is a predefined spatial filter.
Description: Recursive quad-tree description of the pre-spatial 2D wavelet-packet decomposition
This spatial decomposition is done before the temporal decomposition and is used in Inband 2D+t+2D video coding mode. Filter is biorthogonal 9/7 by default, but can be chosen among “W9x7” (9/7), “W5X3” (5/3) and “WHaar” (Haar).
Note: In t+2D coding, the pre-spatial decomposition is unnecessary and this parameter should be equal to “E<>”.
Example: The following configurations are given as examples.



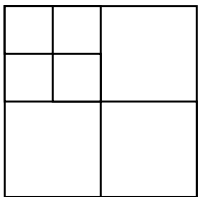
E<>



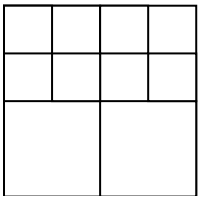
S<>E<>E<>E<>E<>



S<>E<>E<>E<>S<>E<>E<>E<>E<>



S<>S<>E<>E<>E<>E<>E<>E<>E<>



S<>S<>E<>E<>E<>E<>S<>E<>E<>E<>E<>E<>E<>

The spatial filter 5/3 can be used using the syntax “S<Filter=W5x3>E<>E<>E<>”, for instance.

3.10.4.2 POST_SPATIAL_DECOMPOSITION

Class: Structured text
Same as the parameter *PRE_SPATIAL_DECOMPOSITION*.

Description: Recursive quad-tree description of the post-spatial 2D wavelet-packet decomposition
This spatial decomposition is done after the temporal decomposition and must be a subtree of the *PRE_SPATIAL_DECOMPOSITION* parameter.
Example: Same as *PRE_SPATIAL_DECOMPOSITION*. A 4-level dyadic wavelet 2D decomposition (without wavelet packets) is done with:
"S<>S<>S<>S<>E<>E<>E<>E<>E<>E<>E<>E<>E<>E<>E<>E<>".

3.10.5 Inband Related Parameters

3.10.5.1 Inband

Class: Boolean
Description: Enables the Inband (2D+T+2D) video coding mode.
Note: All the parameters of this section should be correctly set. The *PRE_SPATIAL_DECOMPOSITION* parameter should be correctly set.
Example: 0 is a default value.
Options
0 (T+2D) coding mode.
1 In-Band (2D+T+2D) coding mode.

3.10.5.2 Pres_number

Class: Integer
Description: Number of pre-spatial decomposition levels
This parameter should be coherent with the *PRE_SPATIAL_DECOMPOSITION* parameter.
Example: 1 is a default value.

3.10.5.3 Pret_number

Class: Integer
Description: Number of pre-temporal decomposition levels
Example: 0 is a default value

4.5.5.3.1 SBandLeveli

3.10.5.4 Num_AddedLevel_SBand0

Class: Integer
Description: Number of added high levels for Subband0
The value should belong to [0, min (*Pres_number*, 2)]. This parameter is used to control how many levels of higher resolution information are used to generate the prediction.
Example: 1 is a default value.

3.10.5.5 Num_AddedLevel

Class: Integer
Description: Number of added high levels for other PreSubbands
The value should belong to [0, min (*Pres_number* - *SBandLevel*,1)].

Example: 0 is a default value.

3.10.5.6 TLevel_For_PFGS

Class: Integer
Description: Number of Temporal Level on which PFGS (or Leaky, according to *Leaky_PFGS_Flag*) is used.
Example: 3 is a default value.

3.10.5.7 Ratio_For_TLevel

Class: List of comma-separated floats of length *TLevel_For_PFGS*
Description: Threshold value of RD cost ratio for each temporal level in the PFGS scheme
Example: 1.2, 1.2, 1.2, 1.2, 1.2 are default values.

3.10.5.8 Weight_For_TLevel

Class: List of comma-separated Floats of length *TLevel_For_PFGS*
Description: Weight value for each temporal level in the leaky scheme
Example: 0.5 – 0.1**TLevel_For_PFGS* (0.5, 0.4, 0.3, 0.2 are default values).

3.10.5.9 Leaky_PFGS_Flag

Class: Boolean
Description: Enables the Leaky mode.
Options
0 Leaky Mode for motion compensation.
1 PFGS mode for motion compensation.
Examples: 0 is a default value.

3.10.5.10 ME_Phase

Class: Boolean
Description: Specifies if the phase should be considered during motion estimation.
Examples: 0 is a default value.
Options
0 Do not consider phase during motion estimation.
1 Do consider phase during motion estimation.

3.10.6 Baselayer Related Parameters

3.10.6.1 BL_Temporal_Level

Class: Integer
Description: Temporal resolution level of the base layer input file
BL_Temporal_Level is expressed compared to the original input file. This is also the lowest supported temporal resolution.
Example: 2 is a default value. With a 30 Hz input video sequence and a 15 Hz Base Layer one, this value should be 1.

3.10.6.2 BL_Spatial_Level

Class: Integer
Description: Spatial resolution level of the base layer input file
BL_Spatial_Level is expressed compared to the original input file. This is also the lowest supported spatial resolution.
Example: 1 is a default value. With a CIF input video sequence and a QCIF Base Layer one, this value should be 1.

3.10.6.3 BL_OrgSeqName

Class: Text
Description: Path of an output RAW YUV 4:2:0 video sequence that is used as original input sequence for the Base Layer (sequence downsampled to the *BL_Temporal_Level* temporal resolution and to the *BL_Spatial_Level* spatial resolution)
This sequence can be used as a reference.
Example: OrgBLSeq.yuv is a default value.

3.10.6.4 BL_RecSeqName

Class: Text
Description: Path of the reconstructed Base Layer RAW YUV 4:2:0 video sequence
Example: RecBLSeq.yuv is a default value.

3.10.6.5 BL_MCSeqName

Class: Text
Description: Path of the motion-compensated only residual RAW YUV 4:2:0 video sequence corresponding to the input video layer file
This sequence can be obtained by a H.264/MPEG-4 AVC decoder (_mc suffix).
Example: ResBLSeq.yuv is a default value.

3.10.6.6 BL_MotionInfoName

Class: Text
Description: Path of the motion information file radix produced by the H.264/MPEG-4 AVC encoder
Example: “BL_MotionInfo” is a default value. “Output_MEInfo_t” is OK to match Output_MEInfo_t0.bin, Output_MEInfo_t1.bin … files.

3.10.6.7 BL_RecPred

Class: Boolean
Description: Enables *BL_RecSeqName* as predictor for Enhancement Layer.
Example: 0 is a default value.
Options

0 Do not use Base Layer reconstructed RAW YUV 4:2:0 video pointed by *BL_RecSeqName* as predictor for Enhancement Layer.
1 Use Base Layer reconstructed RAW YUV 4:2:0 video pointed by *BL_RecSeqName* as predictor for Enhancement Layer.

3.10.6.8 BL_ResPred

Class: Boolean
Description: Enables *BL_ MCSeqName* as predictor for Enhancement Layer.
If this flag is set to 1, the residual motion-compensated RAW YUV 4:2:0 video pointed by *BL_MCSeqName* will be used as predictor for Enhancement Layer.
Example: 0 is a default value.

Options

0 Do not use Base Layer residual motion-compensated RAW YUV 4:2:0 video pointed by *BL_ MCSeqName* as predictor for Enhancement Layer.

1 Use Base Layer residual motion-compensated RAW YUV 4:2:0 video pointed by *BL_ MCSeqName* as predictor for Enhancement Layer.

3.10.6.9 BL_MotionInfoPred

Class: Boolean
Description: Enables MotionInfoName as predictor for Enhancement Layer.
Example: 0 is a default value.

Options

0 Do not use Base Layer motion information pointed by *BL_ MotionInfoName* as predictor for Enhancement Layer.

1 Use Base Layer motion information by *BL_ MotionInfoName* as predictor for Enhancement Layer.

3.10.7 Deringing Filter Related Parameters

These parameters control the decoder-side deringing filter.

3.10.7.1 PostFilterStrength

Class: Float
Description: Strength of the deringing post filter
PostFilterStrength scales the quantization noise estimate to specify the amplitude parameter of the de-ringing filter. The value 0.0 disables the deringing post filter. Larger values result in stronger filtering.
Example: 0.25 is a quite universal choice suitable for most sequences and rate-points. It is the default value

3.10.7.2 PostFilterSize

Class: Float
Description: Size of the deringing post filter; directly impacts the number of pixels that are included into the adaptive average of the de-ringing filter. Larger values increase the complexity of the decoder and result in stronger filtering.
Example: 2.5 is a default value, resulting in a 9x9 filter mask.

3.10.8 Intra-prediction Related Parameters

3.10.8.1 PIntra

Class: Boolean

Description: Enables use of intra-blocks and intra-prediction feature.
Example: 0 = deactivated, 1 = activated; default value = 0;

3.10.8.2 PIntraYQP

Class: Integer
Description: QP value for predicted Y component
Example: Range from [0..51]. 10 is a default value.

3.10.8.3 PIntraCQP

Class: Integer
Description: QP value for predicted U and V components.
Example: 10 is a default value. Range from [0..51]

3.10.8.4 PIntraDblk

Class: Integer
Description: Parameter controlling the strength of decoder-side deblocking filter.
Example: 30 is a default value. Range from [0..51]. The higher value, the stronger filtering is.

3.10.9 Optimized parameters

Although it would require an exhaustive investigation beyond the scope of this document, we try here to present an example of some optimized encoding parameters for test sequence Mobile (parameters that differ from their default value are highlighted in yellow):

```
; ////////////////////////////////////////////////////////////////////////////
; // Sequence dependent parameters
; ////////////////////////////////////////////////////////////////////////////

source_file = Mobile_cif           ; File name of source sequence(.yuv)
compress_file =   compress     ; File name of compressed bitstream
frame_width = 352                  ; Width of a frame
frame_height = 288                 ; Height of a frame
frame_number = 300                 ; Number of frames
frame_rate = 30                ; Frame rate
level_s = 1                    ; Number of supported spatial scalabilities
temporal_number = 5                ; Number of temporal transform

MotionEstimation = ME<StartS=0 & EndS=0 & lambdas=16 & SearchRange=32 >\
            ME<StartS=1 & EndS=1 & lambdas=32 & SearchRange=64 >\
            ME<StartS=1 & EndS=1 & lambdas=64 & SearchRange=128>\
            ME<StartS=1 & EndS=1 & lambdas=64 & SearchRange=128>\
           ME<StartS=1 & EndS=1 & lambdas=64 & SearchRange=128>

; ////////////////////////////////////////////////////////////////////////////
; // For Multi-Layer Bitstream
; ////////////////////////////////////////////////////////////////////////////
bitstream_layers = 5               ;Number of bitstream layers.
BitStrmLayers = Layer<S=1 & T=2 & Rate=48  & EndSs=1,1,1>\
```

```
          Layer<S=1 & T=1 & Rate=64  & EndSs=1,1,1,1>\
          Layer<S=0 & T=1 & Rate=128  & EndSs=1,1,1,1>\
          Layer<S=0 & T=1 & Rate=256  & EndSs=1,1,1,1>\
          Layer<S=0 & T=0 & Rate=384 & EndSs=0,1,1,1,1>

; ////////////////////////////////////////////////////////////////////////////
; // Inband parameters
; ////////////////////////////////////////////////////////////////////////////
Pres_number = 0
TLevel_For_PFGS = 4
ME_Phase = 1

; ////////////////////////////////////////////////////////////////////////////
; // Sequence independent parameters
; ////////////////////////////////////////////////////////////////////////////
PRE_SPATIAL_DECOMPOSITION = E<>
POST_SPATIAL_DECOMPOSITION= S<>E<>E<>E<>E<>,\
S<>E<>E<>E<>E<>,\
S<>S<>E<>E<>E<>E<>E<>E<>E<>,\
S<>S<>E<>E<>E<>E<>E<>E<>E<>,\
S<>S<>E<>E<>E<>E<>E<>E<>E<>,\
S<>S<>S<>E<>E<>E<>E<>E<>E<>S<>E<>E<>E<>E<>S<>E<>E<>E<>E<>S<>E<>E<>E<>E<>
GOP_Size = 128
PostFilterStrength = 0.0

; ////////////////////////////////////////////////////////////////////////////
; // Pixel-based intra-prediction
; ////////////////////////////////////////////////////////////////////////////
PIntra     = 1
PIntraYQP  = 10
PIntraCQP  = 10
PIntraDblk = 30
```

3.10.10 STool parameters

Hereafter is are described the parameters used to use Stool encoding scheme:

3.10.10.1 PIntraCQP

Class: Integer
Description: QP value for chrominance
Example: 10 is a default value. Range from [0..51]

3.10.10.2 differential_coded

Class: Boolean
Description: Stool activation flag (inter-layer prediction). If it is set to 0 (default) original MSRA implementation will run, if it is set to 1 STool scheme will be used.
Example: 1

3.10.10.3 all_band

Class: Boolean
Description: specifies the set of temporal subbands used to predict the upper layer. If it is set to 1 (default) all the temporal subbands will be predicted; if it is set to 0 only the L…L (the low-pass temporal subbands) will be predicted.
Example: 1 the spatial LL of all temporal subband are predicted using the quantized temporal subbands of lower spatial resolution.

3.10.10.4 down_band

Class: Integer , provided to give the corresponding subbands number.

H: 2
LH: 4
LLH: 6
LLLH: 8
…
LLLL…LLH: 2N

Description: specifies the shift applied to indexes of lower layer temporal subbands used as prediction (0 default).

3.10.10.5 *Example*: Consider the encoding of a CIF@30 fps video sequence, using the prediction signal (temporal subbands) obtained by decoding the QCIF@15 fps encoded version of the same video. If 3 temporal levels are used to encode the CIF layer (temporal subband indexes are H H H H LH LH LLH LLL) while 2 temporal levels have been used to encode the QCIF@15 fps (temporal number indexes are H H LH LL), there will be a mismatch between indexes values of subbands having the same temporal resolution. In fact LLL subbands of CIF@30 fps correspond to the CIF@3.75 fps similarly to LL subbands of QCIF@15 fps which represent the QCIF@3.75 fps. In this case *down_band* must be set to 2.pred_band_fileE

Class: String
Description: specifies the file name of temporal subbands used as prediction in coding/decoding process.

3.10.10.6 pred_band_fileD

Class: String
Descriptions: specifies the file name of temporal subbands saved during the decoding process.

3.10.10.7 lower_layer_rate

Class: Float
Description: specifies the rate associated with the temporal subbands of "base-layer", used to predict the upper layers.

3.11 Using the extractor module

This section provides a detailed description of the Vidwav evaluation extractor's usage.

3.11.1 Extractor syntax

```
Extractor.exe <baselayer_flag> <configuration_file> <input_bs> <nLayer> <output_bs>
```

The extractor is used to extract a layer from a compressed video file or to truncate it to a given (Spatial, Temporal, Bitrate) point according to the layered mode specified by <*baselayer_flag*>.

3.11.1.1 Layered mode

If the compressed bitstream has been made in the layered mode (see 3.10.3.1) (<*baselayer_flag*>==0), the following syntax should be used:

```
Extractor.exe 0 <configuration_file> <input_bs> <nLayer> <output_bs>
```

Options

<*configuration_file*>	Path to the configuration file, specified in section.
<*input_bs*>	Path to the input compressed video bitstream file.
<*nLayer*>	Layer index to be extracted.
<*output_bs*>	Path to the output desired layer video bitstream file.

Examples of usage:

```
Extractor.exe 0 NewCfg.cfg compress 1 compress_1
Extractor.exe 0 NewCfg.cfg compress 4 compress_4
```

3.11.1.2 MGS mode

If the compressed bitstream has been made in the Medium Grain Scale (MGS) mode (see 3.10.3.1) (<baselayer_flag>==1), the following syntax should be used:

```
Extractor.exe 1 <configuration_file> <input_bs> <S> <T> <R> <M> <output_bs>
```

Options

<*configuration_file*>	Path to the configuration file, specified in section 3.10.
<*input_bs*>	Path to the input compressed video bitstream file.
<*S*>	Integer that corresponds to the requested dyadic spatial scalability level. 0 means the original spatial size, 1 means half-resolution, etc…
<*T*>	Integer that corresponds to the requested dyadic temporal scalability level. 0 means the original framerate, 1 means half framerate, etc…
<*R*>	Integer that corresponds to the requested bitrate, in kilobits/second.
<*M*>	Boolean (0 or 1) that specifies if the output bitstream can be truncated again.
<*output_bs*>	Path to the output desired layer video bitstream file.

Examples of usage:

```
Extractor.exe 1 NewCfg.cfg compress_CIF@30Hz_1024kbs 1 1 256 0 compress_QCIF@15Hz_256kbs
Extractor.exe 1 NewCfg.cfg compress_CIF@30Hz_1024kbs 1 2 80 0 compress_QCIF@7.5Hz_80kbs
```

3.11.2 Extractor ouput

When running the extractor, the extractor will display on screen general information about the extraction process. The output information generated may look as follows:

```
////////////////////////////////////////////////////
//            START TRUNCATE                  //
////////////////////////////////////////////////////
Scale for TBand 0: 0.718800
…
Scale for TBand 4: 5.062500
…
```

displays temporal subband scaling.

```
////////////////////////////////////////////////////
//           Global Header ( 52 Bytes)          //
////////////////////////////////////////////////////
…
```

displays Global Header Reading.

```
////////////////////////////////////////////////////
//          START A NEW GOP ( 32 frame)         //
////////////////////////////////////////////////////
Begin to trans-copy layer 0:
Start at       62(bytes),   End at     6446(bytes)
…
```

displays Base Layer (0) trans-copy.

```
Begin to skip layer 1:
Start at     6454(bytes),   End at     8465(bytes)
…
```

displays Layer 1 truncation.

```
Start at    34180(bytes),   End at     68315(bytes)
Press any key to continue
```

displays last layer truncation and end of extraction process.

3.12 Referred documents

Ruiqin Xiong, Jizheng Xu Internet Media Group, Microsoft Research Asia, “Microsoft 3D Subband Video Coding Software”.

Ruiqin Xiong, Xiangyang Ji, Dongdong Zhang, Jizheng Xu Microsoft Research Asia, “MSRA 3D Wavelet Video Coding Specification”.

Ruiqin Xiong, Feng Wu and Shipeng Li, Zixiang Xiong, Ya-Qin Zhang, “EXPLOITING TEMPORAL CORRELATION WITH ADAPTIVE BLOCK-SIZE MOTION ALIGNMENT FOR 3D WAVELET”, Proceedings of SPIE -- Visual Communications and Image Processing 2004, Volume 5308, pp. 144-155, January 2004.

MSSVC_VCIP04

J Xu, Z Xiong, S Li, YQ Zhang, “Three-Dimensional Embedded Subband Coding with Optimized Truncation (3-D ESCOT)”, Applied and Computational Harmonic Analysis 10, 290-315 (2001).

MSSVC_ACHA01

Lin Luo, Feng Wu, Shipeng Li and Zhenquan Zhang, “Advanced Lifting-Based Motion-Threading (MTh) Technique for the 3D Wavelet Video Coding”, Proceedings of SPIE Visual Communications and Image Processing 2003, Volume 5150, pp. 707-718, June 2003.

MSSVC_VCIP03

ISO/IEC JTC1/SC29/WG11/N6914, “Exploration experiments on tools evaluation in Wavelet Video Coding”, #71 MPEG meeting Hong Kong, January 2005.

N. Adami, M. Brescianini, R. Leonardi and A. Signoroni, “SVC CE1: STool - a native spatially scalable approach to SVC”, ISO/IEC JTC1/SC29/WG11, M11368, Palma de Mallorca, October 2004

m11368

N. Adami, M. Brescianini, R. Leonardi and A. Signoroni, “Fully embedded entropy coding with arbitrary multiple adaptation capabilities,” ISO/IEC JTC1/SC29/WG11, M11378, Palma de Mallorca, October 2004

..\m11378.doc

Vincent Bottreau and Christine Guillemot, Rashid Ansari, Edouard François, “SVC technical Contribution to CE1b: Spatial Transform using Three Lifting Steps”, ISO/IEC JTC1/SC29/WG11, M10904, Redmond, July 2004

..\m10904.doc

3.13 Using the decoder module

This section provides a detailed description of the Vidwav evaluation decoder's usage.

3.13.1 Decoder syntax

```
MCWDec.exe <configuration_file> <bitstream_file> <baselayer_flag> <nLayer> <nViewS>
```

Options

<configuration_file>	Path to the configuration file, specified in section 3.10.
<bitstream_file>	Path to the input compressed video bitstream file.
<baselayer_flag>	1 if the Base Layer feature is requested, 0 otherwise.
<nLayer>	Layer index to be decoded. If omitted, the highest one is decoded.
<nViewS>	Resolution in which decoding operation is performed. If omitted, the layer resolution is chosen.

Examples of usage:

```
MCWDec.exe NewCfg.cfg compress 0 3
MCWDec.exe NewCfg.cfg compress_1 0
```

3.13.2 Decoder ouput

```
////////////////////////////////////////////////
//            START DECODING                  //
////////////////////////////////////////////////
Scale for TBand 0: 0.718800
..
Scale for TBand 4: 5.062500
…
```

displays temporal subband scaling.

```
Total Operation 30
   SPATIAL_TRANS SBand:    Y Rect(    0,    0,  176,  144) TBandIdx:    2
…
   SPATIAL_TRANS SBand:    V Rect(    0,    0,   44,   36) TBandIdx:    9
…
Total Operation 105
  ENTROPY_CODING SBand:    Y Rect(    0,    0,   44,   36) TBandIdx:    2
…
  ENTROPY_CODING SBand:    V Rect(   44,   36,   44,   36) TBandIdx:    9
Built T + S + E Scheme successfully
…
```

displays the building of the different encoding modules (temporal, spatial and entropy decoding).

```
============  START TO  DECODE ONE FRAME ============
Buffer Status.
  0 (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0),
Pull Frame for TBand 4.
Begin to decode layer 0:
start at 62 (bytes),    end at 6446 (bytes)
```

```
Buffer Status.
  8 (B= 0, F= 0), (B= 0, F= 0), (B= 4, F= 0), (B= 2, F= 0), (B= 2, F= 0),
…
```

displays the beginning of decoding process by feeding in entropy decoding buffers.

```
temporal level: 3
Total MVs Bits: 2067
motion information bits for s 0: 2067
     FrameUpdate :        NULL->( -1,  ORG::  0, LLLL)<-( -1,  ORG::  0, LLLH)
     FrameUpdate :        NULL->       NULL<-( -1,  ORG::  0, LLLL)
```

displays temporal synthesis.

```
============  START TO  DECODE ONE FRAME ============
Buffer Status.
  0 (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0), (B= 0, F= 0),
Pull Frame for TBand 2.
NULL              ===>||   Module( T, 3,  1)
     FrameUpdate :        NULL->       NULL<-       NULL
     FrameUpdate :        NULL->       NULL<-       NULL
Module( T, 3,  0)    ||===>              NULL
Press any key to continue
```

displays end of decoding process.

3.14 System generated report

The Vidwav evaluation software encoder and decoder generate several reports that could be used later for analysis of a simulation.

3.14.1 MCWLog.txt

This file contains the log of the all encoding process.

3.14.2 enc_motioninfo_mlp_tn.txt

This file contains the log of the Motion Estimation information for motion layer P and temporal decomposition level N.