# AVS - The Chinese Next-Generation Video Coding Standard

**Wen Gao[1], Cliff Reader[2], Feng Wu[3], Yun He[4], Lu Yu[5],**
**Hanqing Lu[6], Shiqiang Yang[7], Tiejun Huang[1], Xingde Pan[8]**

**Joint Development Lab., Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.[1]**
**Consultant, Chinese Academy of Sciences.[2] Microsoft Research Asia, Beijing, 100080, China.[3]**
**Dept. of Electronic Eng. , Tsinghua University. [4]**
**Institute of Information and Communication Engineering, Zhejiang University.[5]**
**NLPR, Institute of Automation, Chinese Academy of Sciences. [6]**
**Department of Computer Science and Technology, Tsinghua University.[7]**
**Beijing Media Works Co. Ltd, Beijing, China.[8]**

## INTRODUCTION

The Audio Video Coding Standard of China (AVS) video standard is a streamlined, highly efficient video coder employing the latest video coding tools and dedicated to coding HDTV content. All video coding algorithms comprise an optimization between absolute coding performance and complexity of implementation. Compared with other standards, AVS has been designed to provide near optimum performance and a considerable reduction in complexity. AVS will therefore provide low-cost implementations.

AVS has also been designed in such a way that its technology can be licensed without delay and for a very reasonable fee. This has required some compromises in the design but the benefits of a non-proprietary, open standard, and the licensing cost savings easily outweigh the small loss in efficiency.

The AVS 1.0 specification was finalized in December 2003 and will be extensively tested with laboratory verification tests and field tests in 2004. It is expected to be issued as a Chinese National Standard in Summer 2004.

AVS applications include broadcast TV, HD-DVD, and broadband video networking.

## DATA FORMATS

### Progressive scan

AVS codes video data in progressive scan format. This format is directly compatible with all content that originates in film, and can accept inputs directly from progressive telecine machines. It is also directly compatible with the emerging standard for digital production – the so-called "24p" standard. In the next few years, most movie production and much TV production will be converted to this new standard. It will also be the standard for digital cinema, so there is convergence in the professional film and TV production industry toward a single production format offering the highest original quality. AVS also codes progressive content at higher frame rates. Such rates may be necessary for televised sports.

A significant benefit of progressive format is the efficiency with which motion estimation operates. Progressive content can be encoded at significantly lower bitrates than interlaced content with the same perceptual quality. Furthermore, motion compensated coding of progressive format data is significantly less complex than coding of interlaced data. This is a significant component of the reduced complexity of AVS coding.

### Interlaced scan

AVS also provides coding tools for interlaced scan format. These tools offer coding of legacy interlaced format video.

### Picture format

AVS is primarily focused on broadcast TV applications with an emphasis on HDTV and therefore will be used principally with a format of 1920 x 1080 pixels. Nevertheless AVS is a generic standard and can code pictures with a rectangular format up to 16K x 16K pixels in size. Pixels are coded in Luminance-Chrominance format (YCrCb) and each component can have precision of 8 bits. AVS supports a range of commonly used frame rates and pixel aspect ratios AVS supports 4:2:0, and 4:2:2, Chroma formats. Chromaticity is as defined by international standards.

## LAYERED STRUCTURE

AVS is built on a layered data structure representing traditional video data. This structure is mirrored in the coded video bitstream. Figure 1 illustrates this layered structure.

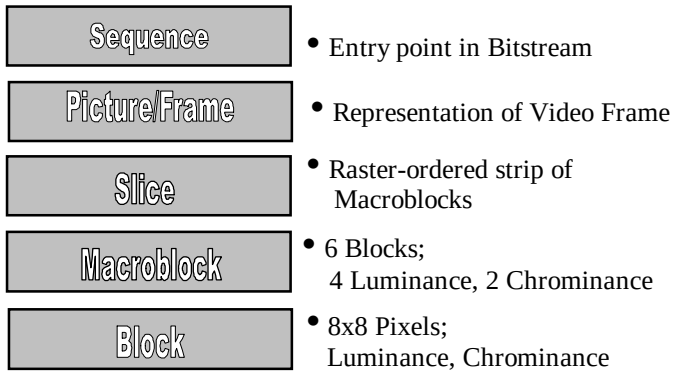


Figure 1. AVS Layered Data Structure

At the highest layer, sets of frames of continuous video are organized into a Sequence. The Sequence provides an opportunity to download parameter sets to decoders. Video frames comprise the next layer, and are called Pictures to avoid any ambiguity between legacy video fields and frames. Pictures can

optionally be subdivided into rectangular regions called Slices. Slices are further subdivided into square regions of pixels called Macroblocks. These are the fundamental coding units used by AVS and comprise a set of luminance and chrominance blocks of pixels covering the same square region of the Picture.

The Sequence, Picture and Slice layers begin with unique Start Codes that allow a decoder's parser to find them within the bitstream. An example of a Sequence of Pictures is shown in Figure 2.

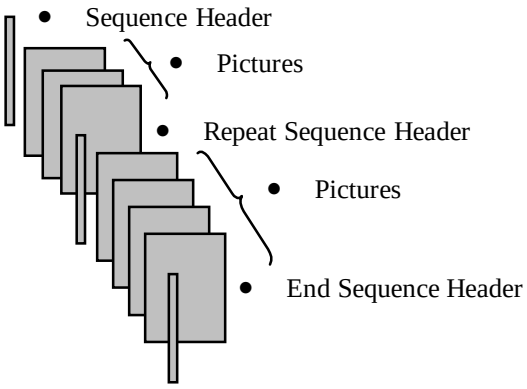


Figure 2. Video Sequence Example.

### Sequence

The sequence layer comprises a set of mandatory and optional downloaded system parameters. The mandatory parameters are necessary to initialize decoder systems. The optional parameters can be used for other system settings at the discretion of the network provider. In addition, user data can optionally be contained in the Sequence header.

The Sequence layer provides an entry point into the coded video. Sequence headers should be placed in the bitstream to support user access appropriately for the given distribution medium. For example, they should be placed at the start of each chapter on a DVD to facilitate random access. Alternatively they should be placed every ½-second in broadcast TV to facilitate changing channels.

Repeat Sequence headers may be inserted to support random access. Sequences are terminated with a Sequence End Code.

### Picture

The Picture layer provides the coded representation of a video frame. It comprises a header with mandatory and optional parameters and optionally with user data. Three types of Picture are defined by AVS:

- Intra Pictures (I-pictures)
- Predicted Pictures (P-pictures)
- Interpolated Pictures (B-Pictures)

### Slice

The Slice structure provides the lowest-layer mechanism for resynchronizing the bitstream in case of transmission error. Slices comprise an arbitrary number of raster-ordered rows of Macroblocks as illustrated in the example of Figure 3. Slices must be contiguous, must begin and terminate at the left and right edges of the Picture and must not overlap. It is possible for a single slice to cover the entire Picture. The Slice structure is optional. Slices are independently coded – no slice can refer to another slice during the decoding process.

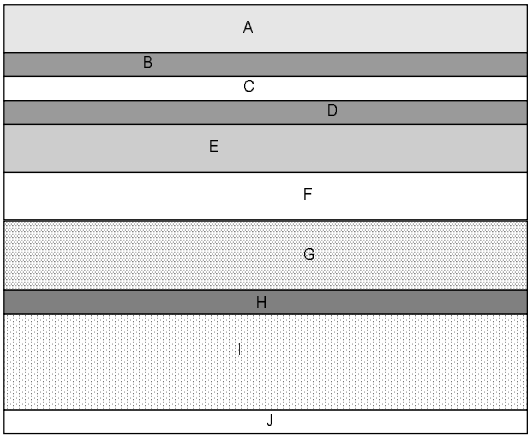


Figure 3. Slice Layer example.

### Macroblock

A Macroblock includes the luminance and chrominance component pixels that collectively represent a 16x16 region of the Picture. In 4:2:0 mode, the Chrominance pixels are subsampled by a factor of two in each dimension; therefore each chrominance component contains only one 8x8 block. In 4:2:2 mode, the Chrominance pixels are subsampled by a factor of two in the horizontal dimension; therefore each chrominance component contains two 8x8 blocks. This is illustrated in Figure 4.

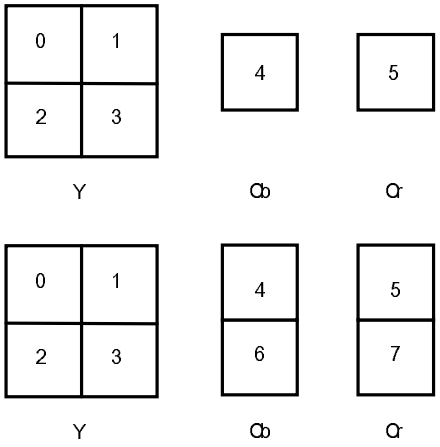


Figure 4. Macroblock formats.

The Macroblock layer is the primary unit of adaptivity in AVS and the primary unit of motion compensation. The Macroblock header contains information about the coding mode and the motion vectors. It may optionally contain the quantization parameter.

### Block

The Block is the smallest coded unit and contains the transform coefficient data for the prediction errors. In the case of Intra-coded blocks, Intra prediction is performed from neighboring Blocks.

## OVERVIEW

The AVS video coding standard is based on the classic hybrid DPCM-DCT coder, which was first introduced by Jain and Jain in 1979[1]. Temporal redundancy is removed by motion-compensated DPCM coding. Residual spatial redundancy is removed first by spatial prediction, and finally by transform coding. Statistical redundancy is removed by entropy coding.

These basic coding tools are enhanced by a set of minor coding tools that remove any remaining redundancy, code side information efficiently and provide syntax for the coded bitstream. The algorithm is highly adaptive, since video data statistics are not stationary and because perceptual coding is also used to maximize perceived quality. The adaptivity is applied at both the Picture layer and the Macroblock layer.

Figure 5 shows block diagrams of the AVS encoder and decoder. The encoder shown in Figure 5a accepts input video and stores multiple frames in a set of frame buffers. These buffers provide the storage and delay required by multi-frame motion estimation. The motion estimation unit can accept original frames from the input buffers or reconstructed coded frames from the forward and backward reference frame stores in the encoder. The motion estimation unit can perform motion estimation in the following ways:

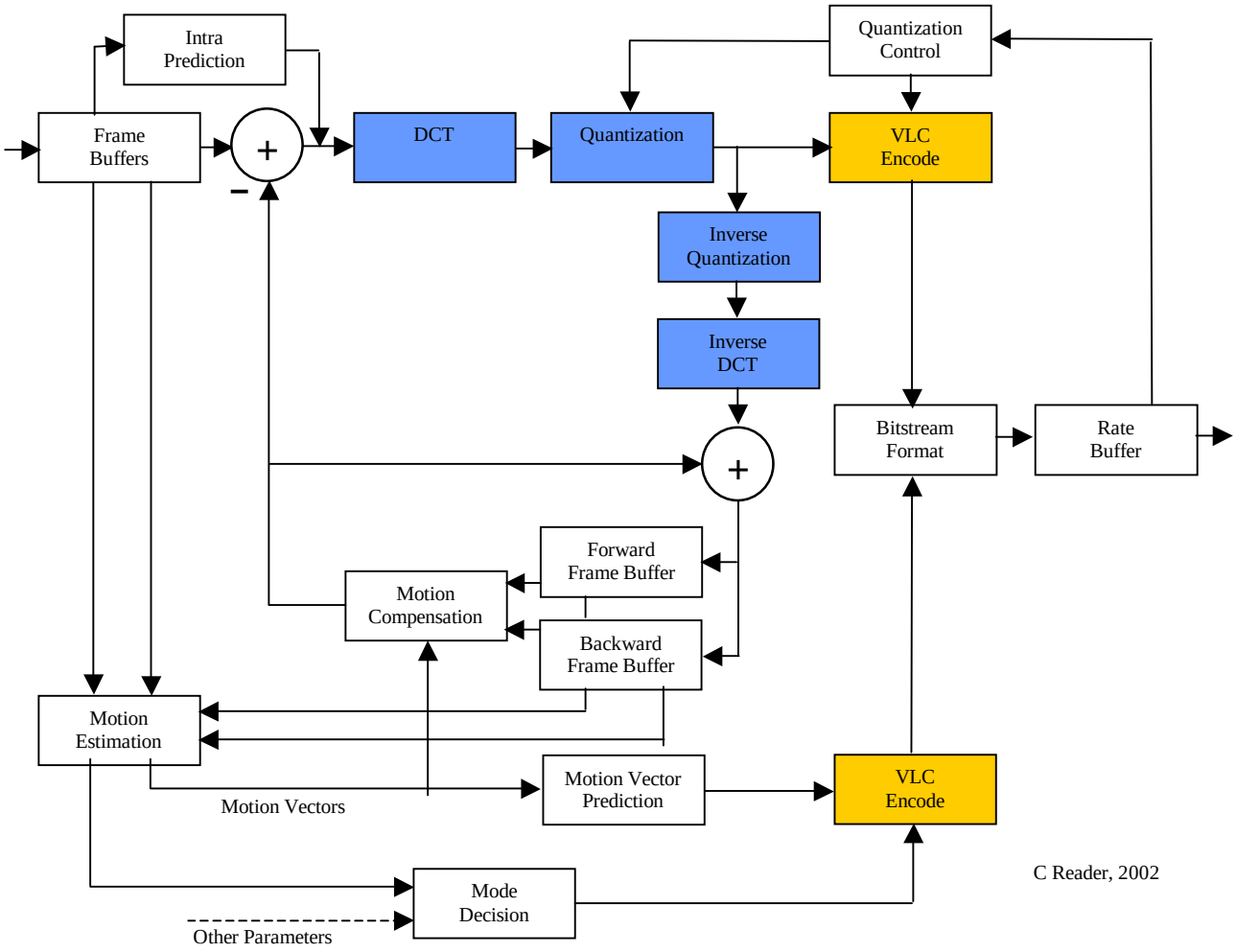


Figure 5a. AVS Encoder

Forward prediction from the most recent reference frame

- Forward prediction from the second most recent prediction frame
- Interpolative prediction between the most recent reference frame and a future reference frame.

Motion estimation produces motion vectors used by the motion compensation unit to produce a forward prediction or interpolated prediction for the current frame. Motion vectors are coded for transmission first by a predictive encoder, and then by entropy encoding.

The prediction produced by the motion compensation unit is subtracted from the current frame and the difference signal, i.e., the prediction error, is coded by the DCT and quantization units. In the case of intra-coded macroblocks, the data passes through the intra prediction process to the DCT. The signal is then VLC encoded, formatted with the motion vectors and other side information and stored temporarily in the rate buffer. The signal is also decoded by the inverse quantizer and inverse DCT, and stored in the forward or backward frame buffers for subsequent use in motion compensation. The rate buffer smooths the variable data rate produced by coding into a constant rate for storage or transmission. A feedback path from the rate buffer controls the quantizer to prevent buffer overflow. A mode decision unit selects the motion compensation mode for pictures and macroblocks.

The decoder shown in Figure 7b accepts the constant rate signal from the storage or transmission and stores it temporarily in a rate buffer. The data is read out at a rate demanded by the decoding of each macroblock and picture. The signal is parsed to separate the quantization parameter, motion vectors and other side information from the coded data signal. The signal is then decoded by the inverse quantizer and inverse DCT to reconstruct the prediction error or intra coded data. The quantizer is controlled by the extracted parameter.

The motion vectors are decoded, reconstructed and used by the motion compensation unit to produce a prediction for the current picture. This is added to the reconstructed prediction error to produce the output signal. In the case of intra-coded macroblocks, the data passes from the DCT through the intra prediction process.

### Coding tools

The main tools used in the AVS coder are summarized in Table 1.

| |
|---|
| Motion compensated prediction |
| Motion compensated interpolation |
| Intra prediction |
| DCT coding and linear quantization |
| Deblocking (loop filter) |
| VLC coding |
| Rate buffering |

Table 1. Major Coding Tools

The minor tools used in the AVS coder are summarized in Table 2.

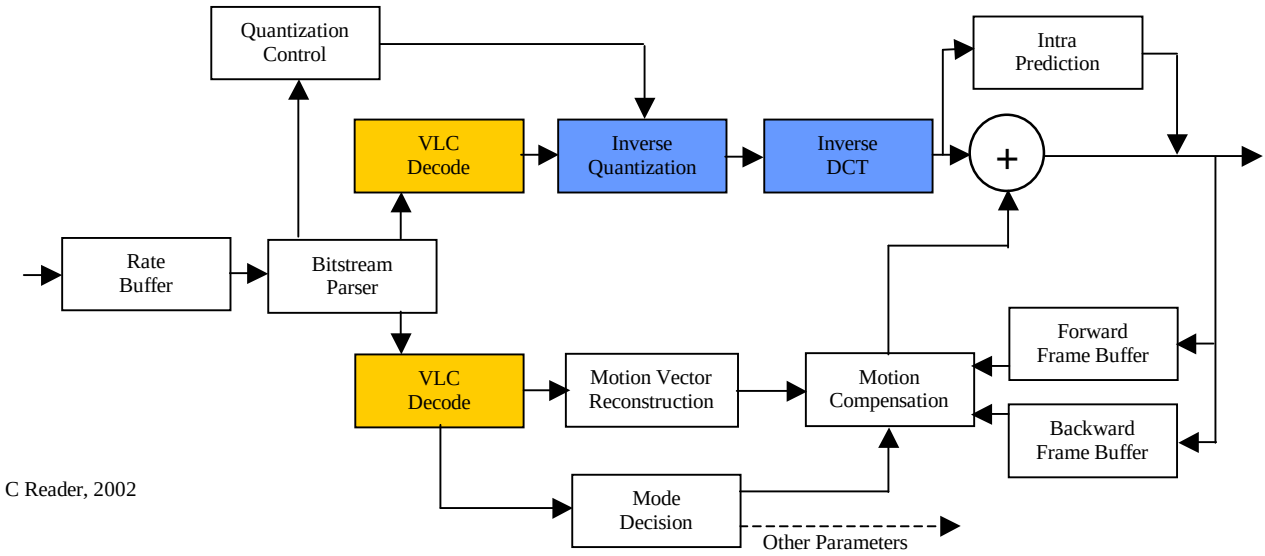


Figure 5b. AVS Decoder.

| |
|---|
| Motion vector prediction |
| Skipped macroblocks |
| Coded block pattern |
| Low delay mode |
| Weighted prediction |
| Inverse 3:2 pulldown |

Table 2. Minor Coding Tools

### Modes

AVS uses adaptive modes for motion compensation at the picture layer and macroblock layer. At the picture layer, the modes are

- Forward prediction from the most recent reference frame
- Forward prediction from the second most recent prediction frame
- Interpolative prediction between the most recent reference frame and a future reference frame.
- Intra coding

At the macroblock layer, the modes depend on the picture mode.

- In Intra pictures, all macroblocks are intra coded.
- In Predicted pictures, macroblocks may be forward predicted or intra coded.
- In interpolated pictures, macroblocks may be forward predicted, backward predicted, interpolated or intra coded.

### Profiles and levels

AVS 1.0 has defined only one profile containing all AVS coding tools. It is called Jizhun (Base Reference) Profile.

4 levels are defined in AVS 1.0.

- Level 4.0: Standard Definition 4:2:0,
- Level 4.2: Standard Definition 4:2:2
- Level 6.0: High Definition 4:2:0
- Level 6.2: High Definition 4:2:2

### MOTION COMPENSATED PREDICTION

In AVS there three types of pictures:

- Forward predicted: P-Pictures
- Interpolated: B-Pictures
- Intra coded: I-Pictures

## Predicted Pictures; P-Pictures

The forward prediction process is illustrated in Figure 6. Prediction of a Macroblock or block in the current picture may be from the most recent reference picture or from the second most recent reference picture. There may be any number of Interpolated pictures in between the current picture and the most recent reference picture. There may be any number of pictures in between the current picture and the second most recent reference picture. Prediction for interlaced format is illustrated in Figure 7. In I-Pictures, the second field may be predicted from the first field. In P-Pictures, prediction of the current field may be made from the four most recent fields. As shown, these fields may be in the current frame, most recent frame or second most recent frame.

## Interpolated Pictures; B-Pictures

The interpolated process is shown in Figure 8. A Macroblock or block in the current picture is predicted by the average of the macroblocks or blocks in the most recent and future P-Pictures that are selected by the motion vector. Prediction for interlaced format is illustrated in Figure 9. Two modes are supported for the motion vector selection.

In Direct Mode, the motion vectors for a macroblock in the interpolated frame are derived from the motion vector of the co-located macroblock in the future P-frame. A separate motion vector is not transmitted, and the forward and backward motion vectors are derived by scaling the P-frame motion vector according to the temporal distance between the interpolated frame and the past and future P-frames.

In Symmetric Mode, a single motion vector is calculated and transmitted that passes through the macroblock in the interpolated frame, pointing to the past and future P-frames.

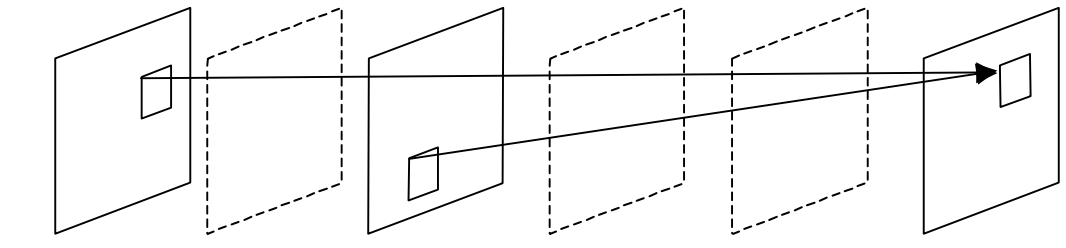


Figure 6. P-Pictures

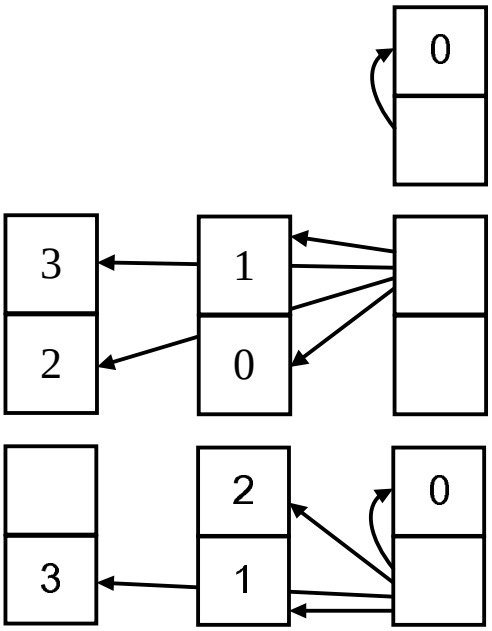


Figure 7. Interlaced P-Frame Prediction.

AVS supports a wide range of prediction modes using adaptive block sizes. Motion estimation may be applied collectively to all luminance blocks in a macroblock, or to pairs of luminance blocks, or individually to each luminance block. Block formats are illustrated in Figure 10.

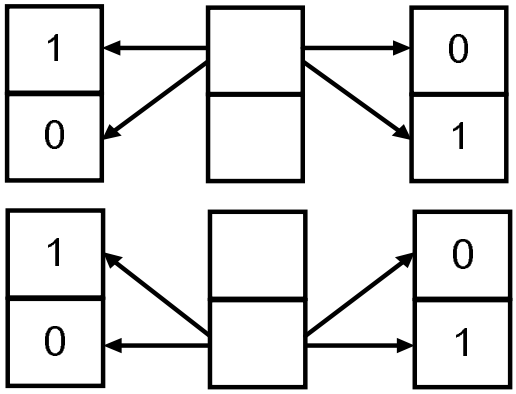


Figure 9. Interlaced B-Frame Prediction.

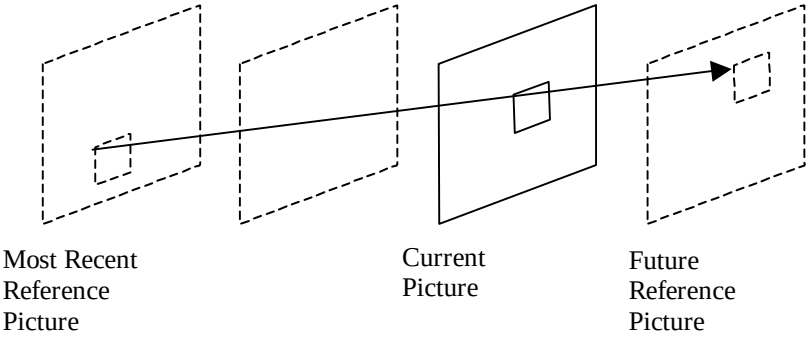


Figure 8. Interpolated Pictures

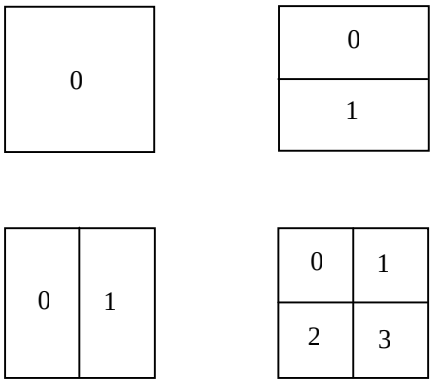


Figure 10. Motion Compensated Prediction Modes.

### Weighted Prediction

Predicted pictures (including P-fields in IP frames) can be predicted from reference pictures that have been weighted. This provides a mechanism for low-residual error prediction when there is a lighting change in the scene, notably when there is a fade or cross-fade. The encoder can signal this to the decoder by optionally transmitting scale and shift parameters for both luminance and chrominance components. These parameters are used in the following equations to compute the weighted prediction matrices:

predMatrix[x,y] =
(predMatrix[x,y] × luma_scale + 16)>>5 + luma_shift

PredMatrix [x,y] =
(PredMatrix[x,y] × chroma_scale + 16)>>5 + chroma_shift

### Out-of-Order Coding

Interpolated frames cannot be encoded or decoded until after the respective reference frames have been encoded or decoded. Therefore the coded representations of the reference frames are placed in the bitstream before those of the interpolated frames.

### Motion vector precision and range

Motion vectors are computed with ¼-pixel accuracy and with unlimited range.

### Sub-Pixel Interpolation

Luminance values at sub-pixel locations are interpolated by a process in which the values at ½-pixel locations are calculated using a 4-tap filter: {-1, 5, 5, -1}, and the values at ¼-pixel locations are calculated using a 4-tap filter {1, 3, 3, 1}. For simplicity, only one-dimensional filters are used. The nomenclature for the locations is shown in Figure 11. Shaded locations represent the original (full-pixel) locations.

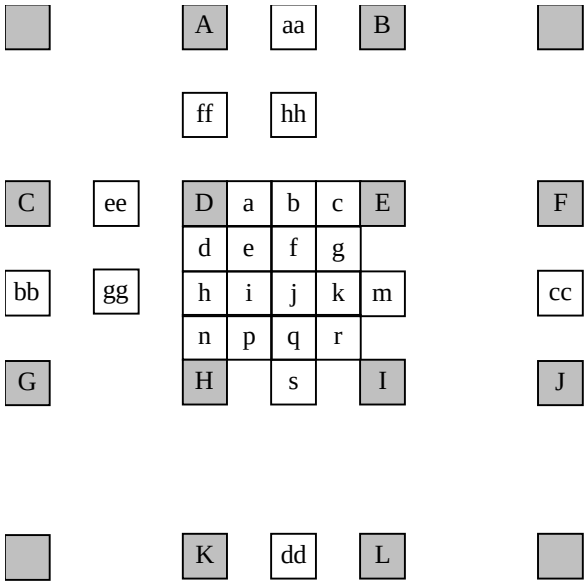


Figure 11. Sub-Pixel Interpolation

The following equations describe the process for each sub-pixel location:

B: b'=(-C+5D+5E-F); b=Clip1((b'+4)>>3)
H: h'=(-A+5D+5H-K); h=Clip1((h'+4)>>3)

Similarly, aa, s and dd are computed using the first 4-tap filter horizontally on the top, third and bottom rows of the array in Figure 11, while bb, m and cc are computed using the left, third and right columns of the array. Then j can be computed using either the available row or column of corresponding locations.

J: j'=(-bb+5h+5m-cc) or j'=(-aa+5b+5s-dd);
j=Clip1((j'+32)>>6)

Then a, d, i and f are computed using the second 4-tap filter:

a'= (ee+7D+7b+E); a=Clip1((a'+64)>>7)
d'= (ff+7D+7h+H); d=Clip1((d'+64)>>7)
i'= (gg+7h+7j+m); i=Clip1((i'+512)>>10)
f'= (hh+7b+7j+s); f=Clip1((f'+512)>>10)

c and n are computed using the second 4-tap filter during subsequent cycles of the process. The value of c is computed during the calculation of sub-pixel values between pixels E, F, I, J, i.e., one pixel-space to the right in Figure 11. The value of n is computed during the calculation of sub-pixel values between pixels H,I, K, and L, i.e., one pixel-space below in Figure 11.

Finally, e, g, p, r are computed by simply averaging their diagonal neighbor pixels:
e = ( D + j + 64 ) >> 7
g = ( E + j + 64) >> 7
p = ( H + j + 64 ) >> 7
r = ( i + j + 64 ) >> 7

Chrominance values at sub-pixel locations are computed from the 4 neighborhood pixels according to the following equation and Figure 12.

$v = ( (8-d_x)\times(8-d_y)\times A + d_x\times(8-d_y)\times B + (8-d_x)\times d_y\times C + d_x\times d_y\times D + 32 ) >> 6$

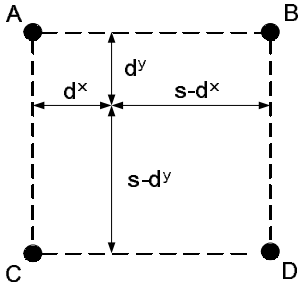


Figure 12. Chrominance Sub-Pixel Interpolation.

### Motion vector prediction

Motion vectors are predicted using the motion vectors for blocks in their neighborhood. The nomenclature is shown in Figure 13. A, B, C are the motion vectors of left, above and above-right blocks.

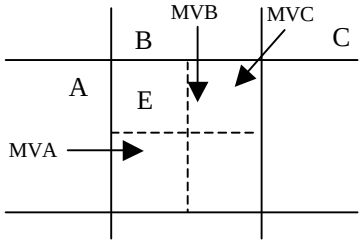


Figure 13. Motion Vector Neighborhood.

The motion vectors are scaled according to the temporal distance from the reference frame (most recent or second most recent).

If prediction is being performed in 16x8 or 8x16 format, the motion vectors used for prediction are as follows:

16x8 Mode:
- Top subblock – use MVB
- Bottom Subblock – use MVA

8x16 Mode:
- Left subblock - use MVA
- Right subblock – use MVC

Otherwise, a spatial distance measure is computed:

Dist12 (MV1, MV2) = $|x_1 - x_2| + |y_1 - y_2|$
Where (x,y) are the components of MV

between pairs of neighboring blocks DistAB; DistBC, DistCA and the median distance is found. The prediction motion vector is then found according to the following rules:
- If median = DistAB, use MVC
- If median = DistBC, use MVA
- If median = Dist CA, use MVB

## INTRA-PREDICTION

In Intra frames and intra-coded macroblocks in P- and B-frames, intra blocks are spatially predicted from neighboring intra-coded blocks. Figure 14 illustrates the process, from which it can be seen that prediction is made using 16 pixels above and to the right of the block, and 16 pixels to the left and below-left of the block. There are 5 directional modes and a DC mode (mode 2).

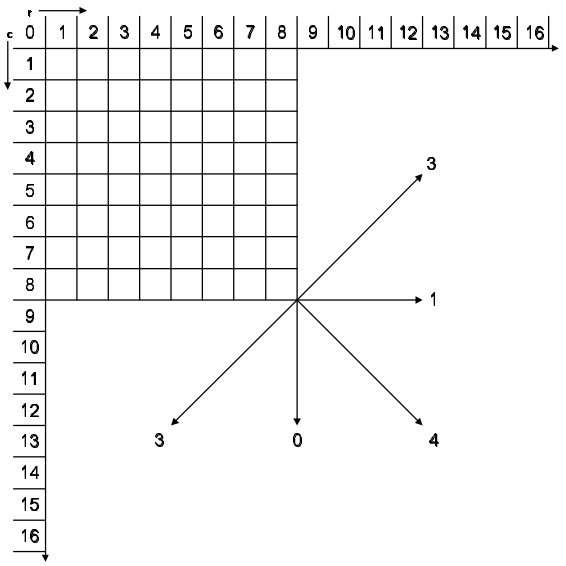


Figure 14. Intra Prediction Modes.

Signaling of the prediction mode to the decoder can be explicit, in which case this is signaled using a prediction mode flag, or it can be implicit, following a set of rules so the decoder can make the same decision as the encoder. In the latter case, selection of the prediction mode is based on the prediction modes for the blocks above (Mode A) and to the left (Mode B) of the current block. The lower numbered mode for these two blocks is selected for the current block. If either invalid, pick mode 2. In the former case, the mode in the bitstream is used if it is smaller than the mode above, otherwise, add 1 to the bitstream mode.

## DCT CODING

### Transform

AVS uses a separable, integer-precise, 8x8 discrete cosine transform (DCT). The inverse transform is shown in Figure 15. The transform is designed in conjunction with the quantization to minimize decoder implementation complexity. This operation will be described here. The separable inverse transform is applied to the rows of the coefficient matrix data:

$$H' = \text{CoeffMatrix} \times T_8^T$$

The elements of this intermediate transform are then scaled with rounding:

$$H''_{ij} = (h'_{ij} + 4) >> 3$$

Then the columns are transformed:

$$H = T8 \times H'' \qquad i, j = 0..7$$

Finally the residual matrix is formed by scaling (with rounding):

$$r_{ij} = [h_{ij} + 26] >> 7 \qquad i, j = 0..7$$

$$T_8 = \begin{bmatrix} 8 & 10 & 10 & 9 & 8 & 6 & 4 & 2 \\ 8 & 9 & 4 & -2 & -8 & -10 & -10 & -6 \\ 8 & 6 & -4 & -10 & -8 & 2 & 10 & 9 \\ 8 & 2 & -10 & -6 & 8 & 9 & -4 & -10 \\ 8 & -2 & -10 & 6 & 8 & -9 & -4 & 10 \\ 8 & -6 & -4 & 10 & -8 & -2 & 10 & -9 \\ 8 & -9 & 4 & 2 & -8 & 10 & -10 & 6 \\ 8 & -10 & 10 & -9 & 8 & -6 & 4 & -2 \end{bmatrix}$$

Figure 15. DCT-based integer Transform

### Coefficient Scan Order

The 2D coefficients are converted into a 1D sequence for quantization and coding using a zigzag scan for progressive data and an alternate scan for interlaced data. These are shown in Figure 16.

| j \ i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 5 | 6 | 14 | 15 | 27 | 28 |
| 1 | 2 | 4 | 7 | 13 | 16 | 26 | 29 | 42 |
| 2 | 3 | 8 | 12 | 17 | 25 | 30 | 41 | 43 |
| 3 | 9 | 11 | 18 | 24 | 31 | 40 | 44 | 53 |
| 4 | 10 | 19 | 23 | 32 | 39 | 45 | 52 | 54 |
| 5 | 20 | 22 | 33 | 38 | 46 | 51 | 55 | 60 |
| 6 | 21 | 34 | 37 | 47 | 50 | 56 | 59 | 61 |
| 7 | 35 | 36 | 48 | 49 | 57 | 58 | 62 | 63 |

Figure 16a. Zigzag Scan.

| j \ i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 11 | 16 | 22 | 32 | 38 | 55 |
| 1 | 1 | 6 | 12 | 20 | 25 | 33 | 42 | 57 |
| 2 | 2 | 7 | 15 | 21 | 28 | 37 | 43 | 58 |
| 3 | 4 | 10 | 19 | 27 | 31 | 39 | 47 | 59 |
| 4 | 5 | 14 | 24 | 30 | 36 | 44 | 50 | 60 |
| 5 | 8 | 17 | 26 | 35 | 41 | 48 | 52 | 61 |
| 6 | 9 | 18 | 29 | 40 | 46 | 51 | 54 | 62 |
| 7 | 13 | 23 | 34 | 45 | 49 | 53 | 56 | 63 |

Figure 16b. Alternate Scan

### Quantization

Quantization of the transform coefficients is performed with an adaptive linear quantizer. The step size of the quantizer can be varied to provide rate control. In constant bitrate operation, this mechanism is used to prevent buffer overflow. The transmitted step size quantization parameter is used directly for luminance coefficients. For chrominance coefficients it is modified on the upper end of its range.

The quantization parameter may optionally be fixed for an entire picture or slice. If it is not fixed, it may be updated differentially at every macroblock.

The linear quantization process is modified to work together with the transform in order to provide low complexity decoder implementation. The resulting non-linear quantization and scaling coefficients are held in look-up tables, “DequantTable”, “ShiftTable” and the inverse quantization and scaling process is as follows:

$$w_{ij} = (\text{QuantCoeffMatrix}[i,j] \times \text{DequantTable}(QP) + 2^{\text{ShiftTable}(QP)-1}) >> \text{ShiftTable}(QP) \quad i,j=0..7$$

### ENTROPY CODING

The quantized coefficients are coded using a 2D variable length code for run and level. Sets of kth-order exponential Golomb codes are used, with a code number (“CodeNumber”) range of 0 to 59. The code number 59 is used as an escape code. A set of 7 tables is used for intra-coded luminance coefficients and a set of 7 tables for inter-coded luminance coefficients. A set of 5 tables is used for both intra- and inter-coded chrominance coefficients. An “EOB” (End of Block) code is used to terminate coding of a given block. (However tables VLC0_Intra, VLC0_Inter and VLC0_Chroma are used only to decode the first coefficient, so that in these tables there is not EOB, as it is obvious that if a block has non-zero coefficients, the first coefficient cannot be EOB). In all these tables, the assignment of code numbers to level and run is specified only for positive levels. Run and negative levels are assigned as follows:

CodeNumber (-abs(level), run ) =
CodeNumber (abs(level), run ) + 1

In the decoding process, 2D-VLC tables are switched based on the size of the previously decoded level. A set of threshold level values determines the switching point. The first coefficient in a block is decoded using the first table. The absolute value of that coefficient is compared against the set of thresholds to determine which table will be used for the next coefficient. This process continues recursively until the EOB is decoded. Decoding of that block then terminates and the process is reset for the next block.

When the code number is equal to 59 in the parsing process, which means (level, run) is beyond the scope of the table, another two CodeNumbers are parsed from the bitstream, and decoded to produce level and run values.

Entropy coding is also used for other elements of the bitstream, notably the motion vectors.

### DEBLOCKING FILTER

The deblocking filter or loop filter is a non-linear 1D filter selectively applied across the edges of blocks in order to smooth block artifacts. The filter is turned on and off, and applied with varying strength according to local conditions in the block neighborhood. The filter parameters can be signaled explicitly in the bitstream on a picture basis, or they can be deduced by the decoder based on the local conditions. The

deblocking filter is applied recursively to the upper edge and left-hand edges of blocks. Essentially, the filter is adaptive to whether adjacent blocks are intra coded or whether the reference frames and motion vectors are significantly different. Further, the filter is adaptive to local pixel gradients across the block boundary and to the state of the quantization parameter. Experience has shown that all these factors influence discontinuities across block boundaries and the visibility of such discontinuities.

## RATE BUFFERING

AVS supports constant bitrate (CBR) operation using traditional coding tools.

## SYNTAX

The AVS syntax supports a number of additional features, some of which are listed here:

- Separate I and PB picture headers
- Progressive sequence and frame
- Interlaced parameters
- Low -delay mode incl. buffer management
- Sequence extension data
- SMPTE timecode in I-pictures
- Picture distance
- Fixed QP at frame or slice level
- Skip coding mode
- Picture weighting selection and parameters

## PERFORMANCE

The performance of AVS has been measured for a set of HDTV test sequences. The format of these sequences is 720p, and the range of bitrates is from approximately 1Mbit/s to approximately 20 Mbit/s. A comparison has been made with H.264. The results are shown in Figure 17, from which it can be seen that for these sequences and bitrates, the performance of AVS 1.0 is within 0.1dB in almost all cases.

## SUMMARY

The AVS 1.0 standard combines a set of traditional video coding tools with new coding tools into an efficient new algorithm. Compared with other standards, AVS is significantly less complex. However, the performance of AVS for its target applications is very high. This indicates that by focusing on particular applications, and using innovative algorithm design, it is possible to simultaneously achieve high coding efficiency and low implementation cost.

## Reference

J R Jain, A K Jain, "Interframe Adaptive Data Compression Techniques for Images", Signal and Image Processing lab., Dep. Electrical & Computer Eng., UC Davis Tech Rep., Aug 1979.

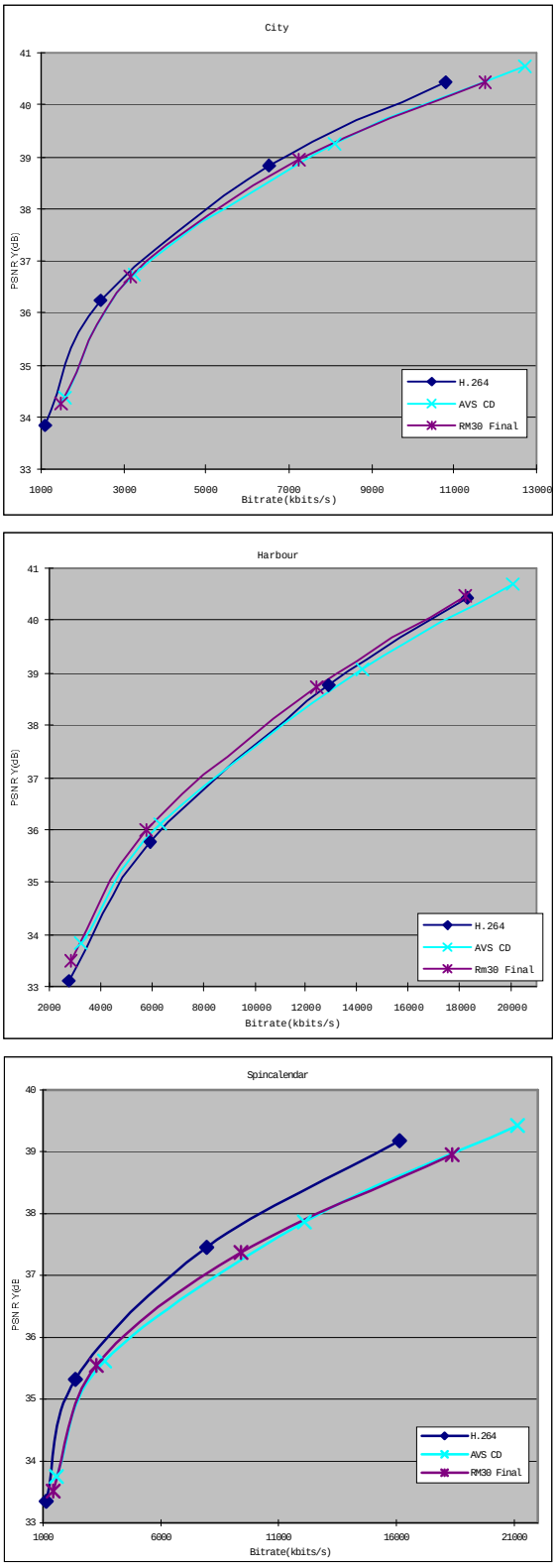


Figure 17. AVS Test Results.
[AVS CD, AVS Ref. Model 30, H.264]
[H.264 Main Profile with CABAC and Loop Filter on]