May 2006, Vol.21, No.3, pp.345–353 J. Comput. Sci. & Technol.

Low-Complexity Tools in AVS Part 7

Feng Yi (易 峰), Qi-Chao Sun (孙琦超), Jie Dong (董 洁), and Lu Yu (虞 露)

Institute of Information and Communication Engineering, Zhejiang University, Hangzhou 310027, P.R. China

E-mail: {hero519, sunqc, yul}@zju.edu.cn; agnese@163.com

Received October 15, 2005; revised March 13, 2006.

Abstract Audio Video coding Standard (AVS) is established by the AVS Working Group of China. The main goal of AVS part 7 is to provide high compression performance with relatively low complexity for mobility applications. There are 3 main low-complexity tools: deblocking filter, context-based adaptive 2D-VLC and direct intra prediction. These tools are presented and analyzed respectively. Finally, we compare the performance and the decoding speed of AVS part 7 and H.264 baseline profile. The analysis and results indicate that AVS part 7 achieves similar performance with lower cost.

Keywords AVS part 7, context-based 2D-VLC, deblocking filter, direct intra prediction

1 Introduction

Audio Video coding Standard (AVS) is established by the Working Group of China in the same name. Up to now, there are two separate video parts in this standard which target to different video compression applications: AVS part 2[1] for high-definition digital video broadcasting and high-density storage media and AVS part 7[2] for low-complexity, low-resolution mobility applications[3].

AVS part 7 (in this paper, it is referred as AVS for short) is based on the hybrid coding framework, including spatial and temporal predictions, integer transform and efficient entropy coding techniques. It aims at the applications of video communications on mobility devices which only have limited processing and memory resources. So when we constitute the standard, complexity consideration is always kept in mind. In [4], a complexity analysis of H.264 baseline profile (in this paper , it is referred as H.264 for short) shows that three most complex tools are: in-loop deblocking filter (33%), interpolation (25%) and entropy decoding (13%). In order to achieve high compression performance with relatively low complexity, AVS adopts some low-complexity tools. There are 3 main low-complexity tools: deblocking filter[5,6], Context-based Adaptive 2-Dimension Variable Length Coding (CA-2D-VLC)[7] and Direct Intra Prediction (DIP)[8]. In this paper, these three tools are presented and analyzed. And some comparisons with H.264 are shown. With these tools and some new techniques for transform and interpolation, AVS achieves similar performance to H.264 with lower cost.

We describe deblocking filter, CA-2D-VLC and DIP respectively in Sections 2, 3 and 4. In Section 5, we present the decoding speed comparison and the performance comparison with H.264. The summary and conclusion are given in the last section.

2 Deblocking Filter

2.1 Background

In order to remove blocking artifacts in low bit-rate block-based video coding, a method named "in-loop deblocking filter" is applied in H.264[9]. In this method, every boundary between two 4×4 blocks is assigned a boundary strength from five different levels. According to different strengths, different filters are applied[9,10]. This filtering method may need at most 4 pixels on each side of the boundary, and change at most 6 pixels totally (Pixels used for each sample-level boundary deblocking filter are shown in Fig.1, and all pixels except p_3 and q_3 can be updated after deblocking filter). Because of high adaptive structure, the deblocking filter becomes the most complex component at the decoder side, which consumes more than 30% decoding time[4,10]. Thus, developing a simple but effective debloking filter is useful and necessary for AVS.

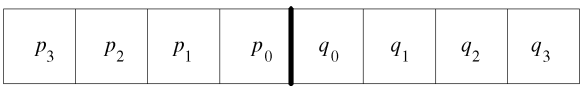


Fig.1. Sample-level boundary for deblocking filter in H.264.

To determine which pixels to update without destroying real edges, Marta Karczewicz made two assumptions[11]. First, the gray-level differences of real edges are larger than those caused by blocking artifacts. Second, real edges with only small gray level changes will not be visually destroyed if smoothed by the filter. The deblocking filtering method of H.264 is developed mainly based on these two assumptions. Taking Human Visual System (HVS) into consideration, Sung Deuk Kim made an assumption that the blocking artifacts in flat region are more sensitive to HVS, and a strong filter must be applied in that case. While for the areas with rich details, a sophisticated smoothing filter is better[12].

Considering all these three assumptions together, we can develop a better deblocking filter. To simplify the existing deblocking filter techniques, a deep analysis of blocking artifacts has also been done. We find that the essential reason causing the blocking artifacts is quantization, which can also be found in [10, 12, 13]. We also

Regular Paper
Supported by the National Natural Science Foundation of China under Grant Nos. 60333020 and 90207005.

find that the Motion Vectors (MV) only induce blocking artifacts indirectly. Whether MV of two neighbor blocks are the same does not necessarily lead to blocking artifacts. What is more, at the same Quantization Parameter (QP), the distortions of quantization are the same in statistic. So the condition of whether the blocks contain coefficient is not needed for a simpler filter mode decision in deblocking filter. Only MB types and QP values should be involved in deciding which filter to be applied. The deblocking filter can be adaptively selected at MB level instead of 4-pixel edge level that is used in H.264. By reducing the number of comparisons and judgments, the total complexity of deblocking filter decreases a lot. To further simply the filtering process, the number of pixels involved in filter is reduced from 8 (Fig.1) to 4 (Fig.2). By doing this, it is much easier to do parallel operations in filtering process.

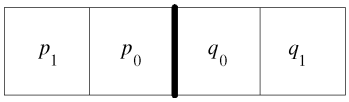


Fig.2. Sample-level boundary for deblocking filter in AVS.

2.2 Deblocking Filter in AVS Part 7

2.2.1 Overview of Deblocking Filter

When we apply a deblocking filter, there are two main steps for each MB. First, a filter mode decision is made using the coding information. Then, different filtering processes are applied to each sample-level boundary (as shown in Fig.2) under different filter modes. The values of some pixels are updated, thus artifact blocking will reduce and improve both subjective and objective results.

2.2.2 Filter Mode Decision

Intra-prediction MB usually has more and bigger residuals than that of inter-prediction MB, which will lead to stronger blocking artifacts at the same QP. So a stronger filter should be applied for an intra-prediction MB, while a weaker filter should be applied to an inter MB. When the MB type is P_Skip, there is no coded residual. When QP is not very large, the distortion caused by quantization is relatively small, so at this time filtering process may not necessarily need. Based on different MB types and QPs, three filter modes are used.

First, for an intra MB, Intra Filter Mode is used to each block boundary in an MB and the MB's up and left boundaries. (In Fig.3, the lines indicate the boundaries which will be filtered.)

Second, for an inter MB which is not P_Skip or the QP is larger than a certain threshold (in AVS the threshold is set to 40 in default condition. But it can be changed by the encoder with an offset transmitted in bitstream), Inter Filter Mode is used.

Third, for an inter MB whose type is P_Skip and the QP is smaller than the threshold, the filtering process is bypassed.

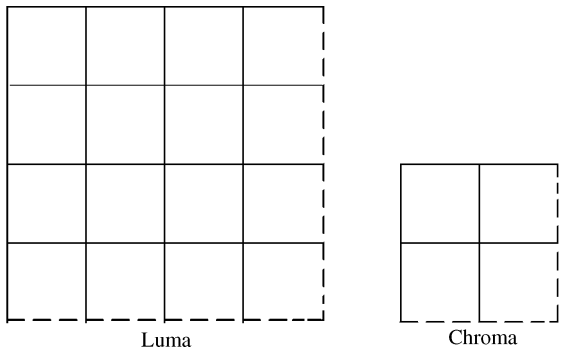


Fig.3. Block boundaries needed to apply deblocking filter.

For both Inter and Intra Filter Mode edge, sample-level filtering decision is made. If the following three conditions hold true, then the filtering process is applied, otherwise, the filtering process is bypassed.

$$|p_0 - q_0| < \alpha \ (IndexA) \tag{1}$$

$$|p_1 - p_0| < \beta \ (IndexB) \tag{2}$$

$$|q_1 - q_0| < \beta \ (IndexB) \tag{3}$$

where α and β can be calculated by *IndexA*, *IndexB*. p_1, p_0, q_1 and q_0 are samples across every sample-level boundary, shown in Fig.2. For detailed information, please refer to [2].

2.2.3 Filtering Process

The filtering processes for Intra and Inter Filter Mode are different.

For Intra Filter Mode, at first, Δ_0 and Δ_1 are calculated for each block to be filtered, using (4) and (5).

$$\Delta_0 = \mathrm{Clip}(-CI, CI, ((q_0 - p_0) \times 4 + (p_1 - q_1) + 4) \gg 3) \tag{4}$$

$$\Delta_1 = \Delta_0 \gg 1, \tag{5}$$

where function Clip is an operation defined as:

$$\mathrm{Clip}(a, b, c) = \min(\max(a, c), b). \tag{6}$$

CI is the upper limit of the Clip function for I slice, which can be obtained by looking up tables defined in the standard.

Then, the following formulas are used to get the values of p_1', p_0', q_1' and q_0'.

$$p_0' = p_0 + \Delta_0, \tag{7}$$

$$p_1' = p_1 + \Delta_1, \tag{8}$$

$$q_0' = q_0 - \Delta_0, \tag{9}$$

$$q_1' = q_1 - \Delta_1. \tag{10}$$

Finally, the values of p_1, p_0, q_1 and q_0 are replaced by p_1', p_0', q_1' and q_0', which have been clipped into the range of 0–255.

For Inter Filter Mode, according to the content of the block, a more sophisticated filter process is applied. First, Δ_0 and Δ_1 are calculated, using (4) and (5) with CI replaced by CP. CP is the upper limit of the clip function for P slice, and can be induced from CI (in AVS, $CP = CI \gg 1 + CP\_Offset$, where CP_Offset is an offset value transmitted in bitstream). Then, the values of p_1', p_0', q_1' and q_0' are calculated by the procedure below.

```
if (abs(p0 − p1) < (β ≫ 1)) {
    p0' = p0 + Δ0;
    p1' = p1 + Δ1;
}
else {
    p0' = p0 + Δ1;
    p1' = p1;
}
if (abs(q0 − q1) < (β ≫ 1)) {
    q0' = q0 − Δ0;
    q1' = q1 − Δ1;
else {
    q0' = q0 − Δ1;
    q1' = q1;
}
```

Similar to the Intra Filter Mode, a clip to p_1', p_0', q_1' and q_0' must be done before updating the values of p_1, p_0, q_1 and q_0.

2.2.4 *Difference Between Deblocking Filter of AVS and H.264*

There are 6 main differences as follows.

a) Boundary strength decision of AVS is made at MB level, but in H.264 is at 4-pixel edge level.

b) The number of strength levels of AVS is 3, while in H.264 it is 5.

c) No motion vector information and coefficient information are required for filter mode decision in AVS.

d) For AVS, at most 2 pixels on each side of the boundary are involved and updated in filtering process, so the pixels processed between two neighboring boundaries will not be overlapped. Therefore, parallel operations can be achieved in filtering process. But in H.264, because of the overlapped pixels of neighboring boundaries, parallel operations are hard to be implemented.

e) There are fewer logic control operations of AVS than that of H.264.

f) Fewer tables are required for AVS deblocking filter.

2.3 Complexity Analysis

The key advantage of AVS deblocking filter is of low complexity. So here we compare its complexity with that of H.264. The comparison includes two parts: complexity of mode decision, and complexity of filtering process.

2.3.1 *Complexity of Filter Mode Decision*

H.264's deblocking filter calculates Boundary strength (Bs) at 4-piexl edge level. Fig.4 shows the Bs tree.

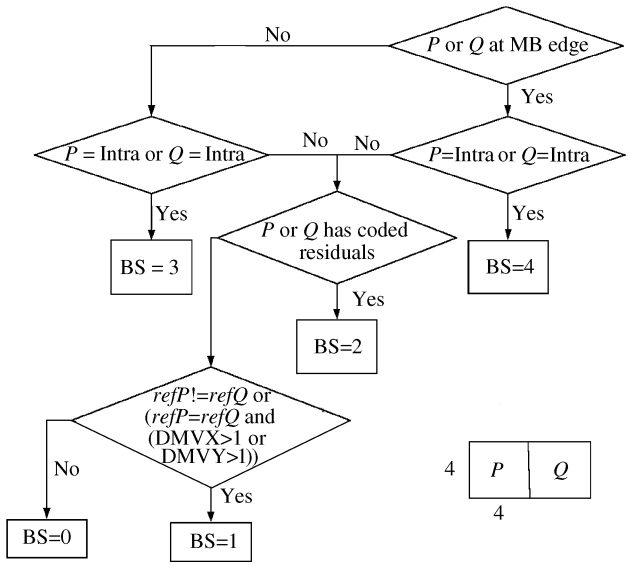


Fig.4. Bs tree of H.264 deblocking filter.

In the Bs tree, there are totally 5 logical decision blocks. For the worst case, 4 logical decisions are required before the Bs is determined for one edge, which means that for an MB, the maximum number of logical decisions is $4 \times 16 \times 2 = 128$. Additionally, many data fetch processes and logic control operations are required, because the data of Coded Block Pattern (CBP), MV and reference frame index will be used for the decision.

With the AVS method, mode decision is made at MB level. The tree structure is shown in Fig.5.

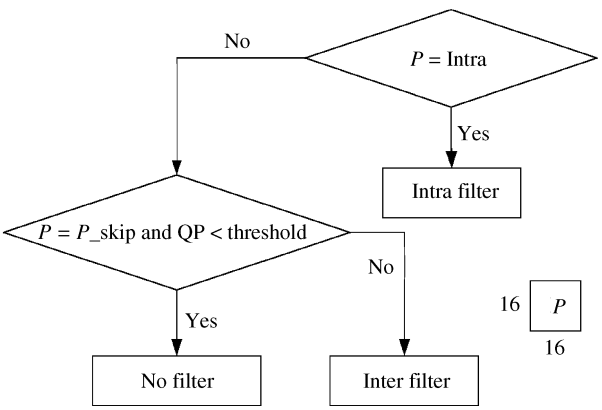


Fig.5. Bs tree of AVS deblocking filter.

In the Bs tree, there are totally 2 logical decision blocks. For the worst case, 2 logical decisions are re-

quired before the Bs is determined for one MB.

The sample-level filter mode decisions are almost the same as that of H.264.

It is obvious that the complexity of the AVS filter mode decision is simplified a lot.

2.3.2 *Complexity of Filtering Process*

The filtering process of AVS is similar to that of H.264 with Bs ranging from 1 to 3. Because fewer branches, fewer content-adaptive operations (in H.264 the upper limit of the Clip function is adaptive by the content, but in AVS the value remains the same under certain QP) and fewer pixels are involved and modified in the proposed method. Additionally, fewer mapping tables are required in AVS filtering process.

2.4 Subject Performance

Fig.6 shows the remarkable subjective picture quality improvement, which compares output images using this low complexity deblocking filter (b) with the old deblocking filter in WM2.0 (a).

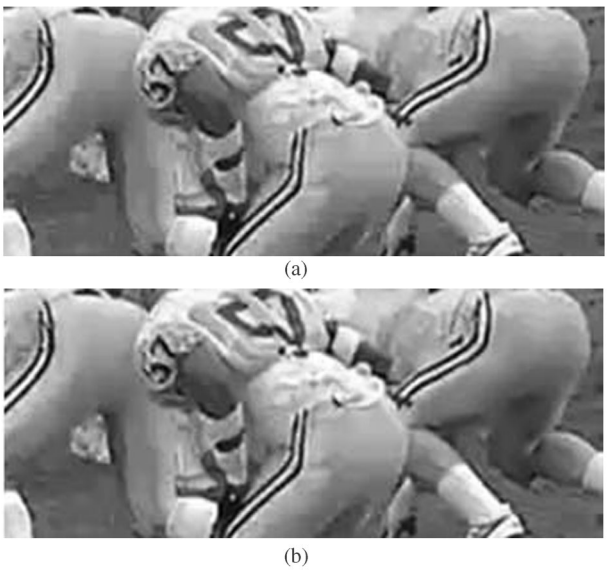


Fig.6. Details of the output images.

3 Context-Based Adaptive 2-Dimension Variable Length Coding

3.1 Background of Entropy Coding

In former standard MPEG2/4[14,15], different VLC tables are used for inter and intra predicted blocks, but only one VLC table is used to code a DCT block. In H.264, an efficient Context-based Adaptive Variable Length Coding (CAVLC)[9] is designed to make use of the context relationship among the transform coefficients in one block. But CAVLC is complex, because of its high adaptivity and irregular codewords. In AVS, an efficient Context-based Adaptive 2-Dimension Variable Length Coding (CA-2D-VLC) is designed for coding transform coefficients in a 4×4 block. High performance can be achieved with relatively low complexity.

3.2 Description of CA-2D-VLC

The transform coefficients are mapped into one-dimensional (*level*, *run*) sequence by the reverse zigzag scan. Depicted in Fig.7, the coding process is as follows.

Step 1. Transform coefficients are classified into three categories, intra, inter and Chroma, for the Luma components of intra MB, Luma components of inter MB and Chroma components for both kinds of MB, respectively.

Step 2. Set $tablenum = 0$ and use the first VLC table to code the first (*level*, *run*) instance.

Step 3. If the (*level*, *run*) can be coded in the current table, code the (*level*, *run*) with Exp-Golomb code.

Step 4. If the (*level*, *run*) is out of the current table's range, code the (*level*, *run*) with escape coding method.

Step 5. Using the coded information to choose the table, represented as tablenum, for the next (*level*, *run*); then jump to Step 3. If all the (*level*, *run*)s in the transform block are coded, code the EOB.

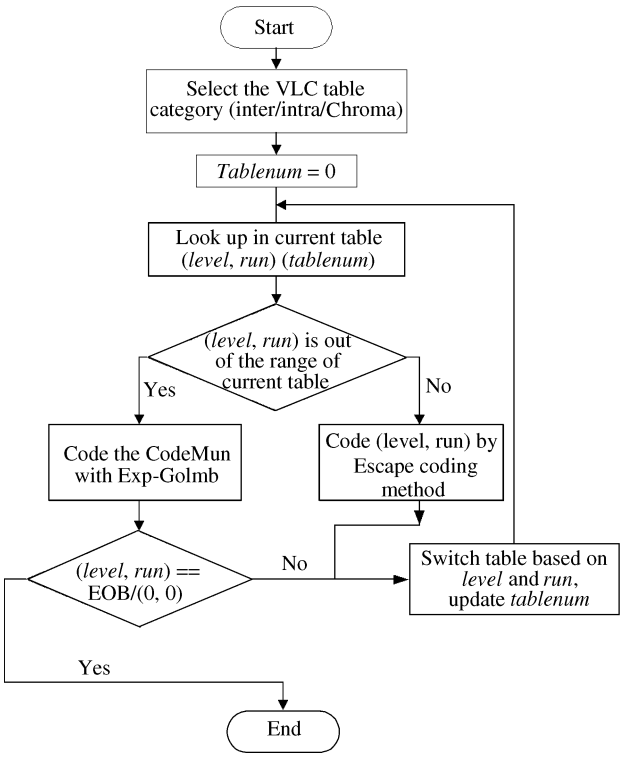


Fig.7. Flow diagram of the entropy coding in AVS.

3.3 Main Features of the CA-2D-VLC

CA-2D-VLC has four main features as follows.

First, it employs the 2D joint VLC to remove the redundancy between the levels and runs in transform

coefficients block. It regards the level and run as a joint event, and estimates the joint probability distribution of (*level*, *run*) to design 2D-VLC tables accordingly.

Second, it employs multiple conditionally-trained 2D-VLC tables to better match different (*level*, *run*)s' probability distributions at different coding phases by automatic table switching. When transform coefficients in one block are mapped into one-dimensional (*level*, *run*) sequence by zigzag scan, the sequence exhibits decreasing tendency for level's magnitude and increasing tendency for run. This indicates a kind of useful context information. To fully exploit the context information and reduce inter-coefficient redundancy, multiple 2D-VLC tables may be used while coding one transform block.

Third, it employs an improved table switching method and an improved escape coding method. In H.264 and AVS Part 2, table switch is based only on the magnitude of last coded level information. For a 4×4 transform block, the percentage of levels whose abstract values equal 1 is very large. When the abstract value of the level equals 1, the probability distribution of (*level*, *run*) is different. In H.264, TrailingOnes is used to deal with this special case. In AVS, table switch is well designed to adapt to the distribution of (*level*, *run*) of 4×4 transform blocks by using both of the last coded level information and run information when the magnitude of last coded level equals 1. Taking the inter-luma coefficients for example, the table switch method is described as follows:

```
int incLevel_inter = {0, 1, 1, 2, 3, 5, 3000};
tablenum = 0; //initialize
for (totalcoeff; totalcoeff > 0; totalcoeff--) {
        //code DCT block
encode_symbol (tablenum, totalcoeff, level, run);
        //code a (level, run) pair
if (abs(level) ⩾ incLevel_inter[tablenum]) {
        //table switch
    if(abs(level) == 1) {
      if (run < 4)
        tablenum = 1;
      else if (run < 7)
        tablenum = 2;
      else
        tablenum = 3;
      }
    else if(abs(level) == 2)
      tablenum = 4;
    else if (abs (level) ⩽ 4)
      tablenum = 5;
    else
      tablenum = 6;
    }
}
```

In AVS, some (*level*, *run*)s are not coded using the 2D-VLC tables directly, but coded with the escape coding method and an improved escape coding method is adopted to avoid the too long codewords and the start code emulation[16].

Fourth, it employs a new CBP_4×4 to be compatible with the 4×4 block-size transform better. The transform block size in AVS is 4×4, so a new 4-bit syntax element CBP_4×4[7] is introduced to represent whether all transform coefficient levels in the corresponding 4×4 block are equal to zero. If the bit for the current 8×8 block of the CBP is "0", all transform coefficient levels of the four 4×4 blocks in the current 8×8 block are equal to zero, so there is no other information needed. If the bit for the current 8×8 block of the CBP is "1", it is possible that one or more transform coefficient levels of one or more of the 4×4 blocks in the 8×8 block shall be nonzero valued. Then the CBP_4×4 needs to be coded.

3.4 Comparison with CAVLC

3.4.1 Computing Complexity

In H.264, the CAVLC method codes level and run separately using corresponding multiple VLC table and makes table selection based on already coded information. In CAVLC, there are some new syntaxes such as TotalCoeff and TrailingOnes to gain better compression. But the decoder should decode the coeff_token to generate TotalCoeff and TrailingOnes, then decode trailing_ones_sign_flag, level_prefix, level_suffix to calculate level, and decode the total_zeros and run_before to calculate run. Because of the separate coding method, it is difficult to implement the parallel CAVLC decoding process. And because of the complex calculation and judgment, the computing complexity is high. In an H.264 decoder, CAVLC takes about 13%[4] of the decoding time. It is possible to reduce the calculation burden by making up pre-calculated tables, but it will use more memories to store the tables of pre-calculated judgments and the memory requirement becomes higher. In AVS, the CA-2D-VLC method firstly maps (*level*, *run*) to CodeNum, a non-negative integer, by the corresponding VLC table, and then the CodeNum is coded with Exp-Golomb codes[20]. The Exp-Golomb codes have regular structures, which mean that any non-negative CodeNum can be mapped to a unique binary codeword using the regular code-constructing rule. Due to the regular codeword structure, the decoder can decode an Exp-Golomb codeword to obtain the CodeNum with very low complexity, and from the CodeNum, level and run can be mapped simultaneously. This process can be done without involving high computational complexity and it is very easy to implement the parallel decoding process.

3.4.2 Memory Requirement

In CAVLC, the binary codes are stored in the VLC tables. Although some binary codewords are very short,

some are long. Using looking-up table in a hardware decoder means that the short and long binary codewords in the same table will use the same bit length memory. Taking the first coeff_token mapping table ($0 \leqslant nC < 2$) for instance, when *TotalCoeff*(*coeff_token*) equals 0 and *TrailingOnes*(*coeff_token*) equals 1, the mapping bitstream is "1", and when *TotalCoeff*(*coeff_token*) equals 3 and *TrailingOnes*(*coeff_token*) equals 16, the mapping bitstream stored in the table is "0000000000001000". Because of the structure of ROM, the amounts of memory to store the two mapping bitstreams are both 16 bits. So it will waste some memory for the short binary codewords and it requires a lot of memories to store all mapping tables. The amount of memories to store the tables of coeff_token mapping to *TotalCoeff*(*coeff_token*) and *TrailingOnes*(*coeff_token*), total_zeros table for 4×4 blocks, total_zeros tables for Chroma DC 2×2 and 2×4 blocks and the tables for run_before is about 4379 bits. The biggest tables are the tables of coeff_token, which need about 3354 bits. It is possible to calculate the long binary codes instead of using looking-up tables to reduce the memory requirement, but it will increase the computational complexity. In AVS, because of the regular structures and low computational complexity of the Exp-Golomb codes, the CodeNum can be stored in VLC tables instead of the real binary codewords. It is a useful feature that resolves the problem of high memory requirement for multiple VLC tables. In AVS, the amount of memories to store the 7 intra multiple VLC tables, 6 inter multiple VLC tables and 4 Chroma VLC tables is only 1650 bits.

3.4.3 CBP_4 × 4

One important factor is that CBP shall contain information about coefficients/no_coefficients in a transform block[18]. In H.264, CBP is a 6-bit integer with every bit representing whether the corresponding 8×8 block contains coefficients or not. CBP tells if there are coefficients or not in an 8×8 block but not on the 4×4 block level, and in the case of 16×16 intra mode, some CBP information is contained in "Mode", but the unit of the CAVLC is a 4×4 block. While the CBP represents that the corresponding 8×8 block contains transform coefficient levels, it is also possible that all transform coefficient levels of one or more 4×4 blocks in the 8×8 block are equal to zero. So in CAVLC the condition of full-zero 4×4 block is coded with *TotalCoefficients*=0. The *TotalCoefficients* specifies the following two meanings: whether the coded 4×4 block is full-zero and the number of the non-zero transform coefficient levels in the coded 4×4 block. Only the latter one is the information that need to be coded. Doing the statistic with two information combined together will not be accurate, and the VLC table will not be efficient enough. Meanwhile, although all transform coefficient levels of the 4×4 block are equal to zero, it will calculate the context to code the TotalCoefficients. This will introduce the unnecessary computational complexity. The transform block size in AVS is 4×4, so the CBP_4×4 is introduced. By using the CBP_4×4, EOB only represent whether a 4×4 block is coded or not, there is no need to represent whether the block is full-zero or not. As a result, the statistical probability of the EOB will be more accurate. Using the new VLC tables adapting to the new CBP_4×4, the coding efficiency will be higher[7].

3.5 Performance of CA-2D-VLC

Experimental results (see Table 1) show that CA-2D-VLC has similar performance with CAVLC. The two methods are both integrated into WM2.0 platform, the AVS reference software.

Table 1. Performance Comparison of CA-2D-VLC and CAVLC on WM2.4

Test sequence	AVSNR (dB)[18]	
	CIF 15Hz	CIF 30Hz
Bus	−0.021	−0.045
Football	0.118	0.068
Foreman	0.042	−0.024
Mobile	−0.063	−0.100
News	−0.084	−0.160
Paris	−0.078	−0.087
Tempete	−0.009	−0.053
Average	−0.013	−0.057

The test condition is using IPPP bitstream structure, frame coding, and 2 reference frames with Fast ME, RD optimization and deblocking filter on. The bitrate is about 50–700kb/s. The AVSNR shows the performance gain of using CA-2D-VLC, compared with CAVLC.

4 Direct Intra Prediction

4.1 Background of Intra Prediction

Intra prediction is a technique of extrapolating the edges of the previously-decoded parts of the current picture[3]. There are two types of intra prediction in H.264: Intra_4×4 and Intra_16×16[9]. Intra_4×4 has 9 spatial prediction modes, while Intra_16×16 has 4 prediction modes. Because Intra_16×16 contributes little to the performance, but needs nearly half of the encoding time for the intra coding, AVS adopts only Intra_4×4 modes.

However, Intra_4×4 is not efficient enough, especially at low bit rate. For some case that the prediction modes for 16 4×4 blocks in an MB are all the most probable modes, 16 bits must be transmitted. In some low bit-rate applications, 16 bits to indicate prediction mode information may be more than the bits to code the quantized coefficient data. The inefficient mode information coding will lead to relatively low coding efficiency at low bit rate.

4.2 Algorithm of Direct Intra Prediction

Direct Intra Prediction (DIP) was firstly proposed in [8], and adopted by AVS in that meeting.

It is well known that direct mode in B slice is a very efficient coding tool. Because it does not need to transmit any MV in bitstream, this mode can save some bits when the MV can be calculated using neighboring and co-located MVs. Intra prediction mode is also predicted from the neighboring blocks as MV does, so a similar idea can be used in intra prediction. There is an MB level flag to indicate whether this MB is DIP coded or not. If the MB is DIP coded, no intra prediction mode information is transmitted, and all blocks use the predicted mode as the intra prediction mode. If the MB is not DIP coded, the MB is encoded and decoded in the ordinary way. In AVS, there are two cases: when the MB is DIP coded, Intra_4 × 4 prediction modes of all 16 blocks are their Most Probably Mode (MPM); when the MB is not DIP coded, the MB is coded with ordinary Intra_4×4 modes. This algorithm greatly improves AVS intra coding performance. At the same time, implementation complexity in both encoder and decoder is similar to that when only Intra_4 × 4 is used. The details of encoding and decoding using DIP are presented in Subsections 4.2.1 and 4.2.2.

4.2.1 Encode an AVS Intra MB

It is known that Rate-Distortion Cost (RDCost) is used to apply rate-distortion optimization when choosing a best mode for the MB. It is defined as:

$$RDCost(mode) = D(mode) + \lambda \cdot R(mode). \qquad (11)$$

In (11), $D(mode)$ is the Sum of Square Distortion (SSD) between reconstructed MB and original MB under this mode, $R(mode)$ is the number of bits required to code this MB, λ specifies the relative importance of the distortion D and rate R.

Encoding an AVS intra MB has five major steps:

Step 1. All 16 4×4 blocks in an MB use their MPMs to do Intra_4×4 prediction and calculate *RDCost*(DIP) of this MB.

Step 2. For every block, try the 9 Intra_4 × 4 prediction modes one by one and find the best, and then calculate the *RDCost*(Intra_4 × 4) for this MB.

Step 3. Compare *RDCost*(DIP) and *RDCost*(Intra_4×4). If *RDCost*(DIP) is less than *RDCost* (Intra_4×4), DIP flag is set to 1 and go to Step 4, otherwise DIP flag is set to 0 and go to Step 5.

Step 4. Encode the MB using DIP and finish encoding of this MB.

Step 5. Encode the MB using ordinary Intra_4 × 4 and finish encoding of this MB.

4.2.2 Decode an AVS Intra MB

Decoding an intra MB in AVS takes the following steps:

Step 1. Read the DIP flag in MB head. If the flag indicates the MB is DIP coded, go to Step 2, otherwise go to Step 3.

Step 2. Let intra prediction mode of each block be its MPM, then go to Step 5.

Step 3. Read the MB's prev_intra4 × 4_pred_mode_flag and rem_intra_4 × 4_pred_mode from bit stream.

Step 4. Calculate intra prediction mode of each block.

Step 5. Decode the MB using Intra_4 × 4, with the intra prediction mode calculated in Step 2 or Step 4.

Table 2. Decoding Speed Ratios for H.264 Baseline and AVS Part 7 on Xeon 3.2GHz PC

Sequence	H.264 (JM10.2)			AVS part 7 (WM3.3)			
	QP	Bit rate (kbps)	Decoding speed (fps)	QP	Bit rate (kbps)	Decoding speed (fps)	AVS/H.264 speed ratio
Football CIF 30Hz	28	1499	18	32	1511	34	1.85
	32	932	20	37	956	35	1.73
	36	565	22	43	542	35	1.57
	40	332	25	48	327	37	1.48
Foreman CIF 30Hz	28	388	23	33	368	35	1.52
	32	224	25	37	242	40	1.61
	36	138	26	43	142	37	1.45
	40	89	28	49	89	39	1.41
Paris CIF 30Hz	28	580	25	32	562	48	1.91
	32	329	26	37	332	49	1.86
	36	182	29	43	175	48	1.65
	40	104	29	48	102	48	1.62
Foreman QCIF 30Hz	28	111	92	32	113	149	1.63
	32	60	102	38	59	145	1.42
	36	36	106	43	37	164	1.54
	40	24	110	49	24	169	1.54
Tempete QCIF 30Hz	28	330	70	32	317	137	1.95
	32	155	88	38	140	139	1.58
News QCIF 30Hz	28	75	112	32	76	222	1.98
	32	45	116	37	47	227	1.95
	36	27	125	42	29	244	1.95
	40	17	130	48	17	263	2.03
Average							**1.69**

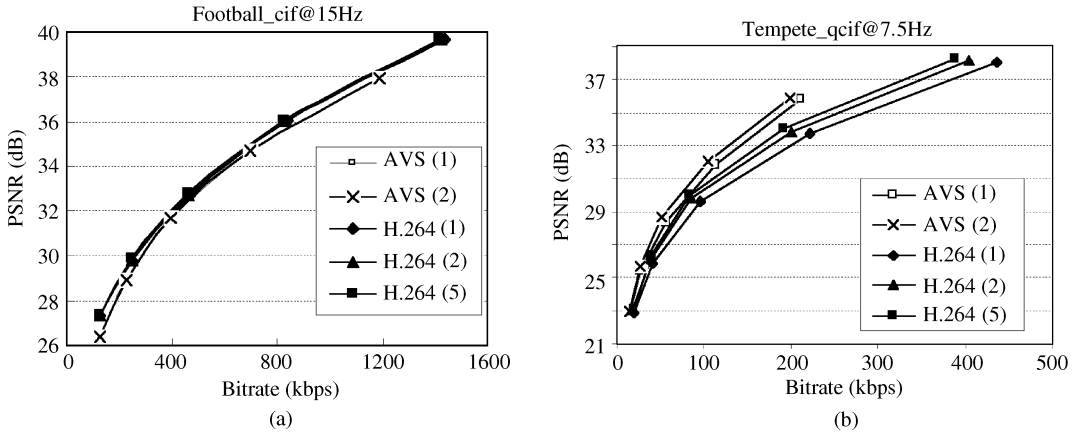


Fig.8. Performance comparison of AVS over H.264. (a) CIF sequence Football coded at 15Hz. (b) QCIF sequence Tempete coded at 7.5Hz.

4.3 Complexity Analysis

On encoder side, to calculate the *RDCost*(DIP), there is no need to do mode search. To calculate the *RDCost*(Intra_4 × 4), best mode will be selected after all 9 modes are tried. So the complexity increase of using DIP to encoder is only about 10% of Intra_4 × 4. On decoder side, there is no additional work compared with Intra_4 × 4, so the total complexity of DIP and Intra_4 × 4 in AVS remains almost the same as that of only Intra_4 × 4.

4.4 Performance of Direct Intra Prediction

The average PSNR gain of DIP+Intra_4 × 4 vs. Intra_4 × 4 on WM2.0 is 0.1dB[8], under the common test condition of AVS[19].

5 Performance of AVS vs. H.264

The speed of AVS part 7 and H.264 baseline decoder are tested. Our comparison is based on non-optimized newest reference software WM3.3 and JM10.2' decoder running on Xeon 3.2GHz workstation. The encoders use similar motion estimation and mode decision processes to generate bitstreams compliant with the respective standards. 2 reference frames are used for both, 100 frames are coded for each sequence and only the first frame is coded as *I* frame. The decoding speed of both and the speed ratio are presented in Table 2. For all six coded bitstreams, the ratio of decoding speed between AVS part 7 and H.264 baseline lies in the range from 1.4 to 2.0, with an average of 1.69. Therefore, our conclusion is that the time complexity of AVS part 7 decoder is about 59% of that of H.264 baseline decoder.

Performance comparison is also done. Experimental results (Fig.8) show that AVS part 7 has similar performance with H.264/AVC baseline profile. The test platform of AVS is WM2.7, and the test platform of H.264 is JM96. The test conditions are listed in Table 3.

Table 3. Test Conditions of Performance Comparison

Option	AVS	H.264
Bitstream structure	IPPP (only 1 *I* frame)	IPPP (only 1 *I* frame)
Entropy coding	CA-2D-VLC	CAVLC
RDO	On	On
Reference frames	1, 2	1, 2, 5
Deblocking filter	On	On
QP	28, 34, 40, 46, 52	28, 34, 40, 46, 52

6 Conclusion

AVS part 7 is developed and standardized by the AVS working group of China. It aims at low-complexity, low-resolution mobility applications. Because it adopts some low complexity tools, such as deblocking filter, CA-2D-VLC and DIP, it achieves the similar compression performance to H.264 baseline profile with lower cost.

References

[1] AVS Video Expert Group. Information technology—Advanced audio video coding standard part 2: Video. Audio video coding standard group of China (AVS), Doc. AVS-N1063, Dec. 2003.

[2] AVS Video Expert Group. Information technology—Advanced audio video coding standard Part 6: Mobility video. Audio video coding standard group of China (AVS), Doc. AVS-N1151, Dec. 2004.

[3] Lu Yu, Feng Yi *et al.* Overview of AVS-video: Tools, performance and complexity. In *Proc. Visual Communications and Image Processing* (*VCIP*), Beijing, July 2005, pp.679–690.

[4] Horowitz M, Joch A, Kossentin F, Hallapuro A. H.264/AVC baseline profile decoder complexity analysis. *IEEE Trans. Circuits Syst. Video Technol.*, July 2003, 13(7): 704–716.

[5] Feng Yi, Jie Dong, Lu Yu. An improvement of intra-frame loop filter. AVS Doc. AVS-M1517, 2004.

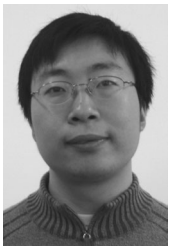
[6] Feng Yi, Jie Dong, Lu Yu. An improvement of inter-frame loop filter. AVS Doc. AVS-M1518, 2004.

[7] Jie Dong, Jianpeng Wang, Lu Yu. An entropy coding method for 4 × 4 coefficient and new CBP coding method. AVS Doc. AVS-M1455, 2004.

[8] Feng Yi, Jie Dong, Lu Yu. A method to improve intra prediction mode coding in AVS. AVS Doc. AVS-M1456, 2004.

[9] Joint Video Team. Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC). Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, March 2003.

[10] List P, Joch A, Lainema J, Bjotegaard G, Karczewicz M. Adaptive deblocking filter. *IEEE Trans. Circuits Syst. Video Technol.*, July 2003, 13(7): 614–619.
[11] Karczewicz M. A filter for removing blocking artifacts. ITU-T SG16 Doc. q15a50 1997.
[12] Sung Deuk Kim, Jaeyoun Yi, Hyun Mun Kim, Jong Beom Ra. A deblocking filter with two separate modes in block-based video coding, I. *IEEE Trans. Circuits Syst. Video Technol.*, Feb. 1999, 9(1): 156–160.
[13] Pang K K, Tan T K. Optimum loop filter in hybrid coders. *IEEE Trans. Circuits Syst. Video Technol.*, Apr. 1994, 4(2): 158–167.
[14] Generic coding of moving pictures and associated audio information—Part 2: Video, ITU-T and ISO/IEC JTC 1, ITU-T recommendation H.262 and ISO/IEC 13 818-2 (MPEG-2). 1994.
[15] Coding of audio-visual objects—Part 2: Visual, in ISO/IEC 14 496-2 (MPEG-4 visual version 1). Apr. 1999.
[16] Lim Chong Soon, Shen Sheng Mei, Li Min. The problem in current escape code and solution. AVS Doc. AVS- M1506r1, 2004.
[17] Gisle Bjontegaard, Arild Fuldseth. Simplified VLC coding of transform coefficients ITU-T SG16 Doc. VCEG-Y08, 2005.
[18] Bjontegaard G. Calculation of average PSNR differences between RD-curves, VCEG-M33, Apr. 2001.
[19] AVS Video Expert Group. AVS-M common test condition. AVS Doc. AVS- N1074, 2004.
[20] Teuhola J. A compression method for clustered bit-vectors. *Information Processing Letteres,* Oct, 1978, 7(6): 308–311.



Feng Yi received the B.S. degree from Zhejiang University in 2003. He is currently pursuing the M.S. degree in electrical engineering at Zhejiang University. His research interests are video coding and implementation of video coding standard. He has been engaged in the AVS standardization effort since 2003. He is author and co-author of many technical contributions of the AVS standard. He was awarded AVS proposal award in 2004. He has published several papers in this field.

Qi-Chao Sun received the B.S. degree from Zhejiang University in 2005. He is currently pursuing the M.S. degree in electrical engineering at Zhejiang University. His research interests are scalable video coding and implementation of video coding standard.

Jie Dong received the B.S. and M.S. degrees in electronic engineering from Zhejiang University, Hangzhou, China in 2002 and 2005, respectively. She is currently working toward the Ph.D. degree at the Chinese University of Hong Kong. Her research interests include video compression and communication.

Lu Yu received her B. Eng. degree (Honor.) in radio engineering and Ph.D degree in communication and electronic systems from Zhejiang University, China in 1991 and 1996, respectively. Now she is a professor in the Institute of Information and Communication Engineering, Zhejiang University. She is the chair of AVS Video-Subgroup and also the co-chair of AVS Implementation-Subgroup. Her research interests include video coding, multimedia communication, ASIC and SoC design.