

# HYBRID VARIABLE LENGTH CODING IN VIDEO COMPRESSION USING VARIABLE BREAKPOINT

*Dihong Tian, Pi Sheng Chang, and Wen H. Chen*

Cisco Systems, Inc.  
{dtian, pschang, wenc}@cisco.com

## ABSTRACT

Hybrid variable length coding (HVLC) was recently proposed as a novel entropy coding scheme for block-based image and video compression, in observation of the inefficiency of the conventional run-level variable length coding scheme in coding consecutive nonzero transform coefficients. In HVLC, each transform block is partitioned into low-frequency and high-frequency regions, and the coefficients in the two regions are coded separately by different schemes. The partition of the transform block was performed based on a pre-defined, constant breakpoint. In this paper, we propose to partition the transform block using a variable breakpoint that is adaptive to the local context. We present a method to find one optimal breakpoint per transform block or per multi-block partition efficiently, and we show that by using variable breakpoint, the efficiency of HVLC can be improved considerably compared to the constant breakpoint case.

*Index Terms*— Image coding, Video coding, Data compression.

## 1. INTRODUCTION

Prevailing video coding standards such as MPEG-2/4 and H.26x commonly adopt a codec model that uses a block-based transform, quantization, and entropy coding. Variable length coding (VLC), for its nice tradeoff in efficiency and simplicity, is widely deployed for the entropy coding, particularly when the codec is desired to have low computational complexity. Conventionally, the coefficient array is represented by a series of (run, level) symbols, where “run” indicates the number of zeros preceding a nonzero coefficient and “level” indicates the magnitude of the nonzero coefficient. The entropy encoder assigns one variable length codeword to each of the symbols<sup>1</sup>, and VLC tables are designed such that symbols appearing more often are encoded by shorter codewords, thus resulting in a compressed bitstream.

The VLC based on the run-level representation, referred to as RL-VLC, is efficient in coding scattered nonzero coefficients. Nevertheless, it is inefficient in coding clustered nonzero coefficients, due to the fact that  $n$  separate codes are required to represent  $n$  consecutive nonzero coefficients, each of which has a run equal to zero. Using multiple VLC tables is a solution to improve the efficiency of RL-VLC for DCT blocks, and its extreme case has been studied in [3], where one VLC table is generated for each frequency index of the transform block. The local context of DCT blocks was not considered in [3].

<sup>1</sup>H.264/AVC [1] deploys a more sophisticated VLC scheme which is specifically suitable for small transform blocks ( $4 \times 4$  and  $2 \times 2$ ). In this paper, we mainly consider  $8 \times 8$  transform blocks, and for comparison we refer to the conventional VLC scheme in H.263 [2] for its simplicity. More remarks regarding the proposed coding scheme in comparison with H.264 will be presented at the end of the paper.

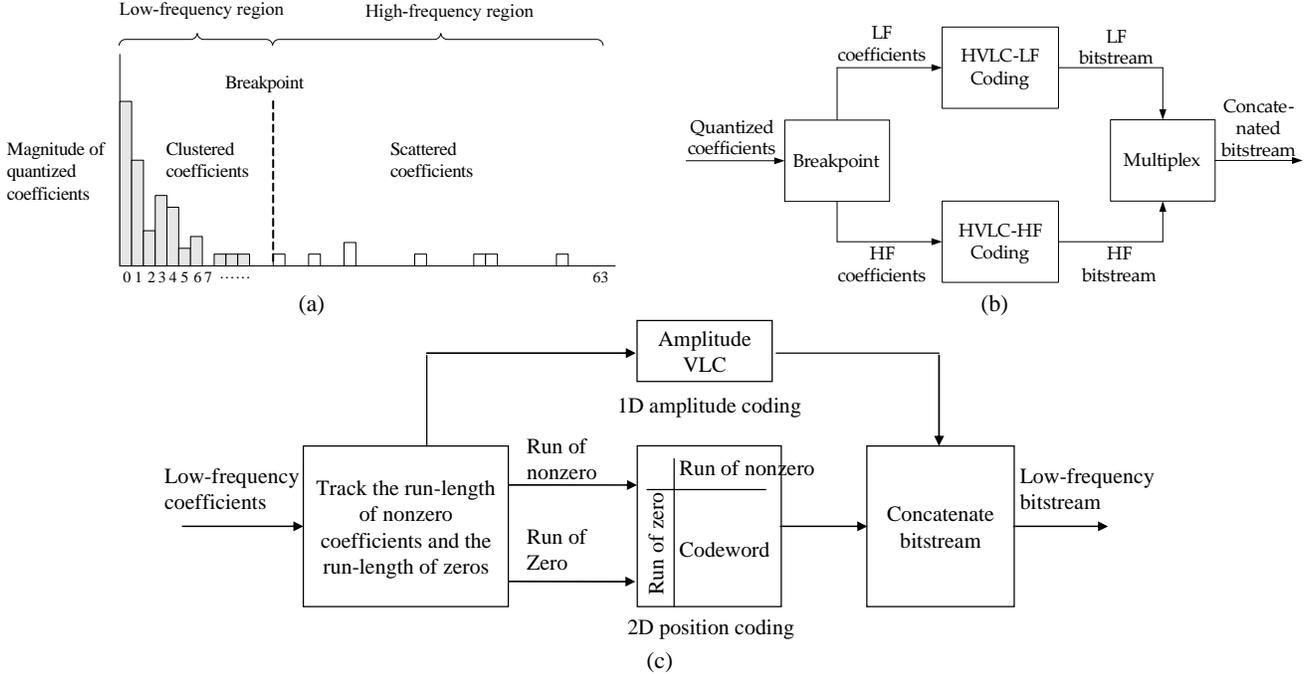
Different from the sole use of RL-VLC, hybrid variable length coding (HVLC) was recently proposed [4], which accounts for the clustered nature of the quantized nonzero coefficients in the low-frequency (LF) region and their scattered nature in the high-frequency (HF) region by employing two position and amplitude coding schemes. In the LF region, the runs of consecutive zero-value coefficients and the runs of consecutive nonzero coefficients are coded as a pair using a two-dimensional VLC table. The amplitudes of the nonzero-valued coefficients are then coded by an independent, one-dimensional VLC table. This coding scheme is referred to as 2DP1DA in [4]. In the HF region, RL-VLC is retained to code the position and amplitude of each nonzero coefficient as a pair. A detailed review of HVLC is presented in Section 2.

The previous work on HVLC also did not take into account the local context of DCT blocks. The partition of a transform block was performed based on a constant, pre-defined breakpoint. Although using a constant breakpoint avoids introducing additional overhead to the bitstream, it does not account for the local statistics of the current block(s). The most efficient partition of a transform block strongly depends on the context of the block. Coding scattered nonzero coefficients by 2DP1DA or clustered nonzero coefficients by run-level coding would not be efficient. Empirical studies also confirmed the difficulty of finding a common “optimal” breakpoint for given coding parameters that would be applicable to generic video sequences. To improve the efficiency of HVLC, in this paper we propose the use of a variable breakpoint in partitioning the transform block into low-frequency and high-frequency regions. In doing so, a two-pass coding algorithm is developed, which finds one optimal breakpoint for each transform block. The breakpoint itself is coded by a separate VLC table. In order to reduce the overhead of coding one breakpoint per block while taking advantage of the statistical correlation of neighboring blocks, the algorithm is extended to find one optimal breakpoint for a frame partition that consists of an arbitrary number of blocks. We provide preliminary experimental results to show that, despite the overhead introduced by coding the variable breakpoint, using variable breakpoint can improve the efficiency of HVLC considerably compared to the constant breakpoint case.

The rest of the paper is organized as follows. Section 2 provides an overview of HVLC. In Section 3, we present the methods that find the variable breakpoint based on the local statistics of a transform block or a multi-block partition. We present preliminary results in Section 4 to evaluate the performance of the proposed methods and provide concluding remarks in Section 5.

## 2. HYBRID VARIABLE LENGTH CODING

The principal idea of HVLC is illustrated in Figure 1. In HVLC, a breakpoint along the coefficient scan path is first defined, as shown



**Fig. 1.** Illustration of HVLC: (a) Coefficient scan of an  $8 \times 8$  block along a pre-defined path, e.g., zigzag; (b) The block diagram of HVLC; (c) The block diagram of the 2DP1DA coding for LF coefficients.

in Figure 1(a). The coefficients below and above the breakpoint are considered as LF and HF coefficients, respectively. The LF coefficients are coded by a two-dimensional position and one-dimensional amplitude (referred to as 2DP1DA) coding scheme, which is illustrated in Figure 1(c), while the HF coefficients are coded by RL-VLC or its equivalent. Figure 1(b) shows a block diagram of the complete HVLC scheme.

It should be noted that in 2DP1DA, a run of nonzero coefficients implies that the following coefficient is a zero-value coefficient, as otherwise it would have been counted into the nonzero cluster and a larger cluster would have been obtained. Therefore, each run of zero-value coefficients can be reduced by 1 before it is coded, with the exception of the first run at the beginning of a block. This results in a shorter codeword to encode the positions of LF coefficients. Each symbol coded by 2DP1DA therefore has the following form:

$$\begin{cases} (R_z, R_n, last), & \text{for the first symbol in a block,} \\ (R_z - 1, R_n, last), & \text{otherwise,} \end{cases} \quad (1)$$

where  $R_z$  and  $R_n$  denote the run of zeros and the run of nonzeros, respectively, and *last* is a binary symbol indicating the “end-of-block” information, similar to H.263 [2].

The breakpoint, denoted by  $N$  hereinafter, is a coefficient index that divides the coefficient sequence into LF and HF regions along the reordering path. The breakpoint must be known to the decoder to properly decode the coefficients. To avoid using two codewords for one coefficient around the breakpoint, the breakpoint is extended beyond the LF region to the last coefficient coded by the LF coding scheme. This extended breakpoint is termed a *soft* breakpoint and is denoted by  $N_s$  for differentiation.

Determining a proper breakpoint is crucial for the coding efficiency of HVLC. On one hand, given the LF and HF coding schemes, there exists an optimal breakpoint within each block of coefficients,

which results in the minimum number of bits in the coded coefficients. On the other hand, this optimal breakpoint varies among blocks and therefore needs to be included in the bit stream, which may introduce a considerable overhead if it is not efficiently coded. Preceding studies, as presented in [4], used a constant breakpoint for the entire video sequence, which is not optimal. In this paper, an approach that allows HVLC to be performed with a variable breakpoint is proposed, which will be detailed in the following sections.

Before proceeding, we present an example to end our summary on HVLC in this section, with the following coefficient sequence:

Index:	1	2	3	4	5	6	7	8	9	10	11	...
Coeff:	2	3	2	0	0	1	-2	1	0	0	-1	...

All the remaining coefficients in the sequence are zeros, and a constant breakpoint,  $N = 6$ , is assumed. Using 2DP1DA for LF coding and RL-VLC for HF coding, the coefficient sequence will be coded as

$$C_P(0, 3, 0) C_A(2) S(0) C_A(3) S(0) C_A(2) S(0) C_P(1, 3, 0) C_A(1) S(0) C_A(2) S(-1) C_A(1) S(0) C_{RL}(1, 1) S(1)$$

In the codeword stream above,  $C_P$ ,  $C_A$  denote the position and amplitude codewords, respectively, for 2DP1DA, and  $C_{RL}$  denotes the codeword of RL-VLC.  $S$  indicates the positive or negative sign of a nonzero amplitude. Note that the soft breakpoint for this example is  $N_s = 9$ , i.e., the ending position of the second 2DP1DA symbol.

### 3. VARIABLE BREAKPOINT

There exists an optimal breakpoint for each block of coefficients that results in the minimum number of bits in coding the block with given LF and HF coding schemes. In this section, we present the method to find the optimal breakpoint for each block. Furthermore, we generalize the method to determine one optimal breakpoint for a frame partition that consists of an arbitrary number of blocks.

### 3.1. Optimal Breakpoint per Block

Finding the optimal breakpoint for a block of coefficients comprises a two-pass coding process. In the first pass, the entire block is coded by LF coding, i.e., 2DP1DA. The coding process starts from the lowest-frequency coefficient. During the process, a table of candidate breakpoints is constructed, which records three quantities for each symbol that is coded by 2DP1DA: the starting position of the symbol, the ending position of the symbol, and the accumulated number of bits that has been consumed to code the coefficients.

As an example, consider the same coefficient sequence in the preceding section. The HVLC table constructed by the first-pass coding process will have the following entries:

Symbol index	Start	End	Number of bits
0	0	0	$n_L(0)$
1	1	4	$n_L(4)$
2	5	9	$n_L(9)$
3	10	12	$n_L(12)$

The first entry in the table (symbol index = 0) is a null entry indicating the state prior to the coding process. The ending position of each 2DP1DA symbol is the frequency index of the zero-value coefficient that follows the run of consecutive nonzero coefficients. For instance, the first 2DP1DA symbol starts at frequency index 1 and ends at frequency index 4 instead of 3. The accumulated number of bits for coding the coefficients is denoted by  $n_L(x)$ , where  $x$  is the ending position of the present symbol. According to the definition,  $n_L(0) = 0$ , and  $n_L(12)$  is the total number of bits for coding the entire block by the LF coding scheme.

In the second pass, the entire block is re-encoded by the HF coding scheme. Unlike the first pass, the coding process is performed in a reverse order starting from the nonzero coefficient that has the highest frequency index. Similar to the first pass, the coding process counts the accumulated number of bits after coding each symbol. Once the coding process reaches an ending position that was recorded in the first pass, it adds the accumulated number of bits consumed by RL-VLC to the corresponding entry in the table. For the above example, the updated table after the second-pass coding is

Symbol index	Start	End	Number of bits
0	0	0	$n_L(0) + n_H(0)$
1	1	4	$n_L(4) + n_H(4)$
2	5	9	$n_L(9) + n_H(9)$
3	10	12	$n_L(12) + n_H(12)$

where  $n_H(x)$  denotes the accumulated number of bits when RL-VLC proceeds to position  $x$  in the reverse order. In contrast to the first-pass coding,  $n_H(12) = 0$  this time, and  $n_H(0)$  is the total number of bits for coding the entire block by RL-VLC.

The optimal breakpoint for coding the block by HVLC is then obtained by finding the entry with the minimum number of bits in the resulting table, which will be coded using a separate VLC table. This optimal breakpoint, as mentioned in Section 2, is a ‘‘soft’’ position, meaning that it can be represented by any coefficient index between the starting position and the ending position of the last symbol that is coded by the LF coding scheme.

### 3.2. Optimal Breakpoint per Multi-Block Partition

Coding one optimal breakpoint per block introduces a considerable overhead to the bitstream. On the other hand, considering the correlation of local context, the soft breakpoints of adjacent blocks may partially ‘‘overlap’’, in which case choosing a single breakpoint from the overlapped coefficient indices will provide optimal coding performance for all blocks. When such an overlap does not exist, us-

ing a single breakpoint for multiple blocks may result in degraded performance for a particular block within the multi-block partition. Nevertheless, it is expected to provide suboptimal performance in a statistical sense, while it significantly reduces the overhead of coding breakpoints. As a result, it achieves a reduced number of bits in coding the entire partition.

Finding the optimal breakpoint for a multi-block partition consists of three steps. The first step closely follows the process in the previous subsection to generate an HVLC table for each block of coefficients. For presentation convenience, in here we use  $nbits_i(x)$  to denote the total number of bits when coding block  $i$  with a breakpoint  $x$ . In other words, we define

$$nbits_i(x) = n_{L,i}(x) + n_{H,i}(x). \quad (2)$$

The property of soft breakpoint, namely a breakpoint can be denoted by any coefficient index between the starting position and the ending position of the corresponding 2DP symbol, suggests that the following equality holds:

$$nbits_i(N_{s,0}) = nbits_i(N_{s,1}) = \dots = nbits_i(N_{s,k}), \quad (3)$$

where  $N_{s,0}$  and  $N_{s,k}$  represent the starting and ending positions of a soft breakpoint, respectively.

Using this important equality, individual HVLC tables resulting from multiple blocks can be merged into a single table for the entire multi-block partition. The optimal breakpoint for the partition can then be obtained based on the merged HVLC table.

To illustrate the table-merging process, let us resume the example in Section 3.1 and refer to the corresponding transform block as Block 1. We assume that the HVLC table for one of its adjacent blocks (referred to as Block 2) has been constructed as follows.

Symbol index	Start	End	Number of bits
0	0	0	$nbits_2(0)$
1	1	6	$nbits_2(6)$
2	7	9	$nbits_2(9)$

According to (3), the following equalities hold for Block 2:

$$\begin{aligned} nbits_2(1) &= nbits_2(2) = \dots = nbits_2(6), \\ nbits_2(7) &= nbits_2(8) = \dots = nbits_2(9). \end{aligned}$$

Similar equalities can also be derived for Block 1. To merge the two individual HVLC tables into a single table, the process starts with combining the first entries of the two tables, which have both starting and ending positions equal to 0. The combined entry corresponds to the case of coding both blocks solely by HF coding:

Symbol index	Start	End	Number of bits
0	0	0	$nbits_1(0) + nbits_2(0)$

The first entries are then removed from both the individual tables. Next, the process inserts one entry at a time to the new table by choosing the ending position in the two tables that has a lower coefficient index. The starting position of the new entry is always the ending position of the last entry plus 1. Every time an ending position is selected and added to the new table, the corresponding entry in the old table is removed. In this example, for the second entry, the starting position is 1, and the lower ending position is 4, chosen from the HVLC table of Block 1:

Symbol index	Start	End	Number of bits
0	0	0	$nbits_1(0) + nbits_2(0)$
1	1	4	$nbits_1(4) + nbits_2(4)$

Note that there is no computation needed for  $nbits_2(4)$ , as  $nbits_2(4) = nbits_2(6)$  according to the equalities obtained for Block 2.

The corresponding entry in the HVLC table of Block 1 is then removed, and the table-merging process continues until all the entries are removed from the two individual HVLC tables. The final merged table for the two blocks is

Symbol index	Start	End	Number of bits
0	0	0	$nbits_1(0) + nbits_2(0)$
1	1	4	$nbits_1(4) + nbits_2(4)$
2	5	6	$nbits_1(6) + nbits_2(6)$
3	7	9	$nbits_1(9) + nbits_2(9)$
4	10	12	$nbits_1(12) + nbits_2(12)$

Finally, the optimal breakpoint for coding the two adjacent blocks is obtained by finding the entry in the table that has the minimum number of bits. Repeating the foregoing process will allow us to find the optimal breakpoint for a multi-block partition with an arbitrary number of blocks.

#### 4. EXPERIMENTAL RESULTS

We present preliminary test results for the performance evaluation of the presented approaches. Following the same setup as in [4], the empirical evaluation is performed with H.263 [2] using the reference software TMN 3.0. H.263 deploys  $8 \times 8$  DCT and run-level VLC for the quantized transform coefficients. For multi-block partition we refer to a  $16 \times 16$  macroblock (MB), which consists of four  $8 \times 8$  transform blocks.

The test video sequences contain three resolutions: QCIF, CIF, and 4CIF, with two sequences for each resolution. Each test sequence has 300 frames with a frame rate of 30 frames per second (fps). In every 15 frames 1 frame is enforced to be coded as an INTRA-frame. For both INTRA and INTER-coded frames, a quantization parameter (QP) of 6 is used. All VLC tables are Huffman code tables constructed based on the measured statistics of symbols and are generated separately for INTRA and INTER coding modes.

Tables 1–3 present the bit-rate results for the methods described in Section 3.1 and Section 3.2, which are referred to as BPP (one breakpoint per block) and BPM (one breakpoint per macroblock), respectively. In the BPP method, we encode the optimal breakpoints of the blocks in a macroblock in a joint manner to reduce the overhead. The results are compared to those obtained by H.263 and by HVLC with a constant breakpoint [4] ( $N = 20$  in the presented case) determined based on the quantization parameter. A minus percentage indicates the bit-rate reduction (performance improvement) by HVLC over H.263, while a positive percentage indicates that the bit rate is increased by HVLC. All the bit-rate results are in kbits/sec.

From the results, it can be seen that HVLC greatly outperforms H.263, and the improvement is especially significant for blocks that are coded in the INTRA mode. This is not difficult to understand, as more nonzero coefficients result when a small quantization parameter is used and/or a block is coded as INTRA. The most significant improvement is observed for the STEFAN and HARBOUR sequences. In Table 2, for example, the INTRA bit rates of STEFAN and HARBOUR are reduced by more than 15%. Those two sequences are noticed to contain large motion, which results in more clustered nonzero coefficients in the low-frequency region.

Using variable breakpoint further improves the coding performance of HVLC considerably on both INTRA and INTER-coded blocks, compared to the case of constant breakpoint. Between the two approaches of variable breakpoint, using one optimal breakpoint per macroblock outperforms the use of one optimal breakpoint per block, as the former provides a better tradeoff of coding both the coefficients and the breakpoints.

**Table 1.** Overall bit-rate results

Sequence	H.263	Constant	BPP	BPM
Carphone	268.56	-3.58%	-4.25%	-5.07%
Pingpong	273.21	-5.97%	-7.86%	-7.59%
Foreman	970.34	-1.47%	-2.18%	-3.27%
Stefan	2713.26	-9.32%	-10.12%	-10.64%
Soccer	4187.94	-4.78%	-5.01%	-6.79%
Harbour	5657.80	-5.13%	-5.93%	-7.77%

**Table 2.** Bit rates for INTRA-coded blocks

Sequence	H.263	Constant	BPP	BPM
Carphone	268.56	-13.24%	-13.76%	-14.49%
Pingpong	273.21	-12.98%	-15.31%	-15.17%
Foreman	970.34	-9.58%	-10.29%	-11.11%
Stefan	2713.26	-17.48%	-19.16%	-19.60%
Soccer	4187.94	-5.76%	-6.43%	-7.86%
Harbour	5657.80	-15.11%	-16.34%	-17.45%

**Table 3.** Bit rates for INTER-coded blocks

Sequence	H.263	Constant	BPP	BPM
Carphone	268.56	-0.41%	-1.13%	-1.97%
Pingpong	273.21	-4.40%	-5.13%	-4.80%
Foreman	970.34	+0.98%	+0.30%	-0.91%
Stefan	2713.26	-7.89%	-8.53%	-9.07%
Soccer	4187.94	-4.59%	-4.74%	-6.59%
Harbour	5657.80	-2.92%	-3.62%	-5.62%

#### 5. CONCLUSIONS AND FUTURE WORK

The proposed use of variable breakpoint in HVLC allows the encoder to seek the best tradeoff in coding both coefficients and breakpoints. Preliminary results showed that the proposed method improves the efficiency of HVLC considerably compared to constant breakpoint, while both of them outperform H.263 significantly. Extensive empirical study is ongoing to provide a thorough evaluation of the performance.

Future research efforts will be devoted to the further improvement of HVLC and its performance comparison with the 2D-VLC schemes used in the newest video codec, e.g., H.264 [1] and China's AVS standard [5]. The improvement of HVLC is anticipated from the introduction of context adaptivity in optimizing VLC tables and the development of an effective method that determines the variable breakpoint in a predictive fashion, eliminating the overhead of breakpoint at a minor compensation of coding efficiency.

#### 6. REFERENCES

- [1] ISO/IEC 14496-10, *Draft of Version 4 of H.264/AVC*, 2005.
- [2] Draft ITU-T Recommendation H.263, *Video coding for low bitrate communication*, 1996.
- [3] G. Lakhani, "Optimal Huffman coding for DCT blocks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 4, pp. 522–527, 2004.
- [4] D. Tian, W. H. Chen, P. S. Chang, G. AlRegib, and R. M. Mersereau, "Hybrid variable length coding for image and video compression," in *Proc. of IEEE ICASSP*, Hawaii, HI, April 2007, vol. 1, pp. 1133–1136.
- [5] Q. Wang, D.-B. Zhao, and W. Gao, "Context-based 2D-VLC entropy coder in AVS video coding standard," *J. Comput. Sci. Technol.*, vol. 21, no. 3, pp. 315–322, 2006.