Advanced Macro-block Entropy Coding in H.264

Chi-Wah Wong, Oscar C. Au, Raymond Chi-Wing Wong+*
Hong Kong Univ. of Science and Technology, Hong Kong +the Chinese University of Hong Kong, Hong Kong
Email: {dickywcw, eeau}@ust.hk Email: {cwwong@cse.cuhk.edu.hk}*

Abstract

Existing video encoders, such as MPEG4, H.263 and H.264, adopt variable length coding (VLC) as entropy coding method, focusing on residue data coding. In low bit-rate video coding, larger quantization parameters are used to give a smaller number of bits spent on residue data. As a result, the header overhead is a dominant factor of yielding the overall bit rates. In this paper, macro-block (MB) header behavior is investigated. It is found that the relative percentage of the number of bits of MB header increases with quantization parameter (*QP*) and header correlation between neighboring MBs is high. Based on these MB header properties, we propose an advanced coding algorithm for entropy coding of MB header. The experimental results suggest that our proposed algorithm achieves lower total number of encoded bits over JM10.2, up to 10% bit reduction, with the same PSNR quality. At the same bit rates, our algorithm has about 0.3-0.4dB PSNR gain.

Keywords: video coding, entropy coding, H.264

1. Introduction

Video coding achieves compression by eliminating redundancy in video data. There are two kinds of redundancy in video data. They are spatial and temporal redundancies. Removal of spatial redundancy and temporal redundancy involves looking within a frame through the use of transform coding techniques and between frames through the use of motion estimation and compensation techniques respectively.

MPEG4[7], H.263[6] and H.264[3] adopt block-based coding schemes, in which the pictures are divided into smaller units called blocks processed one by one in raster-scan order by both the encoder and decoder. A block is defined as a set of 4x4 pixels in H.264, and 8x8 pixels in H.263 and MPEG4. The group of blocks with total size 16x16 is called a macro-block (MB). A number of consecutive macro-blocks are grouped into slices, representing independent coding units to be decoded without referencing other slices of the same frame. Through DCT transformation and quantization, a set of data samples of the block is first linearly transformed and quantized into a set of transform coefficients, resulting in concentrating the energy of input samples into a small number of low-frequency coefficients. Variable length coding (VLC) is adopted for coding resulting residue data. In low bit-rate video coding, larger quantization parameters are used to give a smaller number of bits spent on residue data. As a result, the header overhead is a dominant factor of yielding the overall bit rates, especially in high-complexity video standards such as H.264. In low bit-rate video coding, it is very critical for low-speed network applications [1], [2] including 20-40kbps 2.5G wireless network, 24-64kbps videophone and 64kbps ISDN.

In this work, we propose an advanced coding algorithm for entropy coding of MB header, focusing inter-coded frames (i.e. P-frames), which are used mostly in real-time video streaming applications. We first investigate macro-block (MB) header behavior. It is found that the relative percentage of the number of bits of MB header is dominant in low bit-rate video coding (i.e. *QP* is large) and header correlation of each individual elements (run-length, mode, motion vector, coded block pattern and quantization parameter) between neighboring MBs is high. Based on these MB header properties, we take advantage of the most probable value of MB header and propose an advanced coding algorithm for entropy coding of MB header.

This paper is organized as follows. MB header overview is described in section 2. In section 3, MB header characteristic, including effect of QP and MB header correlation, is discussed. In section 4 and 5, the proposed MB header grouping and the advanced MB coding algorithm are discussed respectively. In section 6, the experiments are conducted to evaluate the performance. Finally, the conclusion is made.

2. MB Header Overview

In H.264, frames are divided into macro-blocks of 16x16 luminance samples each, with two corresponding 8x8 chrominance samples. In QCIF picture format, each frame consists of 99 macro-blocks. A number of consecutive macro-blocks in raster-scan order can be grouped into slices, representing independent coding units to be decoded without referencing other slices of the same frame.

Given that the whole frame is adopted as a unit slice, the frame header (FH) is encoded and *N* MBs are processed one by one. The resulting MB syntax is MB header (MBH) followed by MB residue data (MBD) as shown in Figure 1. In P-frame, the MB header basically consists of elements of run-length (MBR), MB mode (MBM), motion vector data (MBV), coded block pattern (MBC) and change of quantization parameter (MBQ). When the MB header starts to be encoded, the run-length indicates the number of skipped macro-blocks that are made by copying the co-located picture information from the last decoded frame. With motion estimation, the best MB mode is determined and encoded by searching the best matches between the block made by the best mode and current block. To locate the reference inter-block made by the best mode,

two-dimensional motion vectors are specified. The coded block pattern (CBP) indicates whether each of 8x8 luminance sub-blocks and 4x4 chrominance sub-blocks of a MB contains non-zero residue coefficients. With non-zero coefficients, MBQ determines the change of the quantization parameter of a current MB from the previous one. After the MB header is encoded, the non-zero luminance and chrominance coefficients are entropy-coded by either UVLC or CABAC [3]. This encoding process repeats again for next and remaining MBs of the frame.

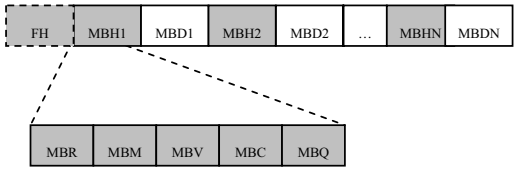

Figure 1: Basic H.264 MB syntax within a frame as slice

3. MB Header Characteristic

3.1. Effect of quantization parameter

Quantization parameter plays an important role in video coding. The number of encoded bits is generally determined by quantization parameter. The greater the quantization parameter is, the smaller the overall bit rate is. Interestingly, the frame and MB header overhead are more complex in the latest video standard H.264 than the previous standards such as H.263. The run-length and advanced MB mode are incorporated in the MB header of H.264, making its header size larger. Table 1 shows the percentage of the number of bits of MB header (P_{header}) and luminance & chrominance residue data (P_{data}) in H.264 and H.263. The experimental conditions are basically similar in H.264 (Baseline Profile[3]) and H.263. The picture format is QCIF, the encoded frame rate is 10fps, the structure of GOP is IPPP, maximum search range is 16, the number of reference frame is 1 and the entropy coding method is VLC. It is noted that among quantization parameter (*QP*) range [0…51], the *QP* values [12…43] in H.264 is similar to the *QP'* values [0…31] in H.263 [3]. Approximately, the *QP* =12, 37 and 43 in H.264 are nearly equal to the *QP'*=1, 16 and 32 in H.263 respectively. Every increment of 6 in *QP* of H.264 (e.g. 37→43) doubles *QP'* in H263 (e.g. 16→32).

Video Sequence	*QP*	H.264		H.263	
		P_{header} (%)	P_{data} (%)	P_{header} (%)	P_{data} (%)
Akiyo	12	6.48	93.52	5.93	94.07
	37	31.59	68.41	26.44	73.56
	43	79.45	20.55	67.74	32.26
Foreman	12	3.52	96.48	2.89	97.11
	37	27.36	72.64	21.11	78.89
	43	66.32	33.68	51.20	48.80
Stefan	12	2.09	97.91	1.84	98.16
	37	10.35	89.65	7.99	92.01
	43	38.88	61.12	22.75	77.25

Table 1: Percentage of the number of bits of MB Header and Residue data in H.264 and H.263

It is observed that for any given QP, P_{header} is higher in H.264 than H.263 because more run-length and advanced MB mode are encoded in H.264, resulting in smaller bits of residue data relatively. It is also noticed that P_{header} increases with *QP* in both H.264 and H.263. When *QP=43* in "Akiyo" and "Foreman", MB header is a dominant factor of yielding the overall number of encoded bits since most of coefficients of the residue data are quantized to zero, giving insignificant bit contribution. In addition, higher run-length in MB header is used for skipped MBs, giving larger run-length bits. In H.264, P_{header} is even higher at larger *QP* values (i.e. > 43) adopted in H.264. Therefore, MB header is a critical factor of yielding the encoded bits in low bit-rate video coding, especially in larger *QP* range from H.264.

3.2. MB Header Correlation

As described earlier, the MB header of each MB consists of its own elements, including MBR, MBM, MBV, MBC and MBQ. The MB header of each MB is encoded independently. Actually, there is MB header correlation between neighboring MBs. It is found that the most probable values of an element of neighboring MB headers are more likely repeated at larger *QP* and the most probable value of all the elements tends to be zero. Table 2 shows the percentage of getting the most probable value "zero" (P_{zero}) for the current element of MB header given that the previous value is also the most probable value under the same experimental conditions as H.264 mentioned above.

Video Seq.	QP	MBR (%)	MBM (%)	MBV_x (%)	MBV_y (%)	MBC (%)	MBQ (%)
Akiyo	12	80.6	74.2	10.6	13.2	0.0	100
	37	43.5	87.4	21.6	26.8	16.8	100
	43	35.5	98.8	41.0	55.1	27.8	100
Foreman	12	98.9	79.6	24.5	25.0	0.0	100
	37	80.2	91.2	33.9	33.5	27.6	100
	43	64.2	97.4	38.6	36.9	35.1	100
Stefan	12	98.8	81.3	25.1	14.2	0.0	100
	37	92.2	92.3	27.8	20.3	31.1	100
	43	78.6	96.4	35.1	29.3	36.5	100

Table 2: Percentage of the most probable value "zero" for the current element of MB header given "zero" previous value

Generally speaking, the goal of all the elements is to minimize both the bits of MB header and MB residue data in accordance with cost equations [3], [5]. It is noticed that P_{zero} of all the elements, except MBR and MBQ, increases with *QP*. When *QP=43*, P_{zero} ranges from 27.8% to 98.8%. MBM tends to give mode 0 for copying co-located reference MB with no MBV overhead bits. MBV_x (horizontal direction) and MBV_y (vertical direction) likely give zero components to minimize the overhead of motion vectors. MBC tends to contain "zero" CBP for indication of no luminance and chrominance coefficients to be encoded and no bits are spent on them. In case of MBR, non-zero run-length is preferred and more MBs are skipped to save bits at larger *QP*. Although P_{zero} of MBR decreases with *QP*, the resulting value is still high (i.e. 35.5% in "akiyo"). Since the experiment uses the fixed *QP*, the value of MBQ remains constant regardless of *QP* values. So, the high MB header correlation indicates higher header redundancies to achieve better coding efficiency for compression, importantly in low bit-rate video coding (i.e. larger *QP*).

4. Proposed MB Header Grouping

As described above, MB header is a dominant factor of yielding the overall bits at larger *QP* and high MB header correlation can achieve higher coding efficiency. Instead of encoding individual MB header independently, we propose that an element of MB header should be encoded based on its previous values in order to reduce overall bits by grouping the same element of various MBs together. It is known that entropy [8] of the current value *H(X)* is larger

than that of the current value *H(X')* given the previous value is known. For the sake of illustration and simplicity, the element "MBR" is chosen at *QP=37* in "Foreman" and two symbols, zero and non-zero values, are used. It is assumed that probabilities of previous zero and non-zero values are 0.7 and 0.3 respectively. Given the zero previous value, probabilities of current zero and non-zero values are 0.8 and 0.2 respectively. Given the non-zero previous value, probabilities of current zero and non-zero values are 0.6 and 0.4 respectively. Therefore, given the previous value, probabilities of current zero and non-zero values are 0.7x0.8 + 0.3x0.6=0.74 and 0.7x0.2 + 0.3x0.4=0.26 respectively. The entropy calculation is shown as follows:

$H(X)$	$H(X')$
$= -0.7\log_2 0.7 - 0.3\log_2 0.3$	$= -0.74\log_2 0.74 - 0.26\log_2 0.26$
$= 0.8813$	$= 0.8268$
	< 0.8813

It is noticed that $H(X) > H(X')$ since there is correlation between the previous and current values. The entropy of the next future value given the known previous value is even smaller (i.e. 0.8144). Therefore, it is better to encode the current or/and future value based on the previous value. The following figure depicts the proposed H.264 MB syntax within a frame as slice.

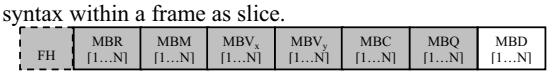

Figure 2: Proposed H.264 MB syntax within a frame as slice

In this proposed syntax, it is convenient for the video encoder to encode individual elements of MB header by placing the *similar elements together* among all the MBs within a frame. Instead of encoding the MB header followed by MB data of each MB one by one, the MB headers of all the MBs within a frame are placed and encoded first. After processing the MB headers, their remaining MB data is encoded. It is known that the header information normally has higher priority over data information since the use of data information mainly depends on the header. Data information is not identified in case of header corruption. In the proposed syntax, the header and data information are treated separately such that different protection methods can be adopted easily. For example, more robust channel coding can be used for coding header information. In addition, the video decoder is more efficient when this proposed syntax is used because the ineffective decision operation (e.g. if-then-else operation) [3], [4], existing mainly in MB header, is separated and the remaining MB data can be processed effectively nearly without decision operation. Also, values (e.g. function parameters) in low-level cache memory can be re-used in both header and data parts since the coding process is repeated one MB by one MB within a group of element of MB header or/and data. It is also observed that no time delay is made at the decoding side as the whole frame is displayed only when all the MBs in the frame (i.e. slice) are decoded completely. In other words, the decoding time to display each frame is the same regardless of the syntax of the slice.

5. Advanced MB Coding Algorithm

It is described in Section 3 that the most probable value of all the elements tends to be zero and the most probable values of an element of neighboring MB headers are more likely repeated at larger *QP*. To the best of our knowledge, it is the first time to propose an advanced algorithm in the area of MB header coding. By adopting run-length coding [8], our proposed algorithm for encoding MB header in a P-frame is shown in the following:

```
For each element of MB header {
  Element_Run-length = 0
  For each i-th non-skipped MB within a frame {
  If (i=0)
    Encode curr_value as normal
  Else {
    If (curr_value = "zero") {
      If (prev_value = "zero") {
        Element_Run-length++
         If (i=last MB)
           Encode Element_Run-length
       }
       Else
          Encode curr_value as normal
      }
    Else {
      If (prev_value ="zero") {
          (1)  Encode Element_Run-length
          (2)  Encode curr_value with index = curr_value-1
          (3)  Element_Run-length = 0
      }
      Else
        Encode curr_value as normal
    }
   }
  }
  If (last MB = skipped MB and element = MBR)
    If (Element_Run-length > 0)
      (1)  Encode Element_Run-length
      (2)  Encode curr_value   with index = curr_value-1
    Else
      Encode curr_value as normal
}
```

According to our proposed MB syntax, the group of each element, including MBR, MBM, MBVx, MBVy, MBC and MBQ, is encoded one by one. For the first non-skipped MB, the current value "*curr_value*" of an element is encoded as normal as original H.264 coding method [3]. In case of "zero" current value, the run-length is incremented by 1 when the previous value "*prev_value*" is also "zero". Otherwise, the "zero" current value is encoded as normal as H.264. In case of "non-zero" current value, the run-length is encoded when the previous value is "zero". At the same time, the "non-zero" current value is encoded with codeword index equal to (current value - 1). It is known that the current value is "non-zero" after "zero" previous value and its run-length are encoded. The "zero" index can be excluded in the codeword table used in H.264. In the same codeword table, the non-zero current value can be encoded with one lower codeword index to achieve smaller encoded bits. If the current and previous values are "non-zero", the current value is encoded as normal as H.264. When the processed non-skipped MB is the last in a frame and current value & previous value are equal to "zero", the run-length is encoded. When the last MB is the skipped MB, the run-length and "non-zero" current value is encoded with codeword index equal to (current value – 1) for non-zero run-length. For zero run-length, the current value is encoded as normal. The codeword of the element run-length is proposed in Table 3. For run-length value *K*, the codeword is either "0…01*X*" where *X=0* for odd *K* or *X=1* for even *K*. The corresponding number of bits is $\lceil K/2 \rceil + 2$ for *K>0*. In QCIF picture format, the

maximum run-length is 98, giving the codeword "0…011" and 51 codeword bits. In a decoder side, the codeword is decoded to the current value as normal as H.264 when the previous value is non-zero or the MB is the first non-skipped one during decoding groups of each element. Otherwise, the run-length and the next codeword, with one lower codeword index, are decoded within a frame. The following MBs are decoded in the same way until the last MB. This process repeats for next and following frames.

6. Experimental Results

We implemented our proposed algorithm in a JVT JM 10.2 version [4]. In the following experiments, we compare the proposed algorithm with the original H.264 JM10.2 encoder. Here are the test conditions used in the baseline profile. The picture format is QCIF, the encoded frame rate is 10fps , maximum search range is 16, the number of reference frame is 1 and the entropy coding method is UVLC, MV resolution is 1/4 pel, Hadamard is "OFF", RD optimization is "OFF", and restrict search range is "0". The first frame was intra-coded (I-frame) and other remaining frames are inter-coded (P-frame).

Table 4 shows the encoded bits and reduction percentage P_r of each element for P-frame in the proposed algorithm, compared with original H.264 MB header coding algorithm. P_r is defined as $(R_{old} - R_{new}) / R_{old} * 100$ where R_{old} and R_{new} are the number of bits of the original H.264 and our proposed algorithms respectively. Two algorithms give the same PSNR but different encoded bits as their MB header entropy coding methods are different. It is observed that the smaller number of header bits and total bits can be obtained in our proposed algorithm. P_r of the header bits ranges from 9.2% to 24.5%. P_r of the total bits is up to 10.3% in "Foreman" when $QP=43$. It can be seen that the proposed algorithm has an obvious bit reduction improvement in low bit-rate video coding (i.e. larger QP) because relatively larger MB header overhead is compressed effectively. It is noticed that P_r of each element has similar behavior to its P_{zero} depicted in Table 2. In case of MBM, MBV and MBC, P_r and P_{zero} increases with QP. It is observed that P_r is zero when $QP=12$ in case of MBC since non-zero (instead of zero) coefficients are preferred and encoded as normal. In case of MBR, P_r and P_{zero} decreases with QP. In case of MBQ, P_r is about 50% as the bit reduction in our proposed codeword (i.e. $\lceil K/2 \rceil + 2$) is nearly 50% for fixed QP. It is also observed that P_r decreases slightly with QP as the number of non-skipped MBs becomes smaller, resulting in insignificant effect of bit reduction relatively. The same experiments (not shown here due to the limited space) are also done in H.263, giving the similar results but slightly lower P_r. Figure 3 shows the graph of PSNR against bit rate in video sequence "Foreman" at low bit rates. At 10kbps, PSNR gain of the proposed algorithm is about 0.3-0.4dB over JM10.2.

7. Conclusion

In this paper, it is observed that the relative percentage of the number of bits of MB header increases with QP and header correlation between neighboring MBs is relatively high. Based on these MB header properties, we propose advanced coding algorithm for MB-header entropy coding. The experimental results suggest that our proposed algorithm achieves lower encoded bits, up to 10% bit reduction, over JM10.2 with the same PSNR quality. At the same bit rates, our algorithm has about 0.3-0.4dB PSNR gain.

8. References

[1] K. Yu, J. Li, C. Shi and S. Li, "A Novel Model-based Rate-Control Method for Portrait Video Coding", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 12, Dec 2005.

[2] X. Yang, W. Lin, Z. Lu, X. Lin, S. Rahardja, E. Ong, and S. Yao, "Rate Control for Videophone Using Local Perceptual Cues", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 4, April 2005.

[3] T. Wiegand, "Working Draft Number 2, Revision 8(WD-2 rev 8)", *JVT-B118r8, ISO/IEC MPEG & ITU-T-T VCEG*, Geneva, Switzerland, 29 Jan.-29 Feb., 2002.

[4] "JVT JM10.2", http://iphome.hhi.de/suehring/tml/, 2006

[5] F. Moscheni, F. Dufaux and H. Nicolas, "Entropy criterion for optimal bit allocation between motion and prediction error information", *Proc. SPIE Visual Commun. And Image Proc.*, pp. 235-242, Nov. 1993

[6] "H.263 Video Coding for Low Bit Rate Communication", *ITU-T Recommendation*, 1993.

[7] "Coding of Moving Pictures and Audio", *ISO/IEC JTC1/SC29/WG11 N3312*, Mar 2000.

[8] D. S. Taubman and M. W. Marcellin, JPEG2000 Image Compression Fundamentals, Standards and Practice, Norwell, *Kluwer Acad.*, 2002.

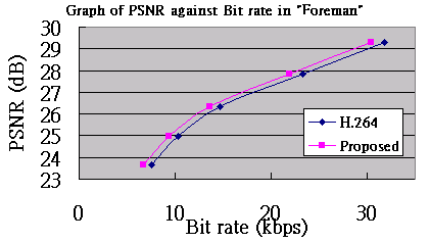

Figure 3: Graph of PSNR against bit rate in "Foreman"

Run-length	Codeword	No of bits
0	1	1
1	010	3
2	011	3
3	0010	4
4	0011	4
5	00010	5
6	00011	5
…		

Table 3: Codeword of Run-length in the proposed algorithm

Video Seq.	QP	MBR		MBM		MBV_x		MBV_y		MBC		MBQ		Header		Total Bits	
		Bits	P_r (%)	Bits	P_r (%)	Bits	P_r (%)	Bits	P_r (%)	Bits	P_r (%)	Bits	P_r (%)	Bits	P_r (%)	Bits	P_r (%)
Akiyo	12	9k	16.0	3k	43.6	5k	3.5	5k	4.1	37k	0.0	5k	47.0	54k	24.5	1154k	1.5
	37	6k	2.4	1k	45.3	2k	8.3	2k	9.4	8k	0.1	3k	45.1	20k	17.5	76k	4.8
	43	3k	1.2	1k	47.6	1k	15.9	1k	17.3	1k	6.4	1k	35.2	6k	9.2	12k	5.5
Foreman	12	5k	47.3	2k	45.9	35k	3.1	35k	3.0	77k	0.0	5k	47.0	157k	18.2	5393k	0.6
	37	8k	20.0	1k	46.3	28k	7.3	28k	7.1	38k	1.1	3k	45.4	105k	16.8	441k	4.5
	43	8k	11.1	1k	47.3	16k	12.6	17k	11.9	11k	9.9	2k	42.6	56k	16.4	91k	10.3
Stefan	12	5k	47.0	2k	44.1	36k	2.1	35k	1.9	73k	0.0	5k	47.0	155k	20.1	9110k	0.4
	37	7k	33.0	1k	46.7	24k	6.1	23k	5.3	54k	2.4	4k	46.3	121k	20.0	1422k	2.0
	43	8k	17.4	1k	48.4	19k	13.0	20k	12.5	26k	5.7	3k	45.7	76k	18.5	225k	6.9

Table 4: Encoded Bits and reduction percentage of each element in the proposed algorithm