FAST BLOCK SIZE PREDICTION FOR MPEG-2 TO H.264/AVC TRANSCODING

Qiang Tang, Panos Nasiopoulos, Rabab Ward

Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada

ABSTRACT

One objective in MPEG-2 to H.264 transcoding is to improve the H.264 compression ratio by using more accurate H.264 motion vectors. Motion re-estimation is by far the most time consuming process in video transcoding, and improving the searching speed is a challenging problem. We introduce a new transcoding scheme that uses the MPEG-2 DCT coefficients to predict the block size partitioning for H.264. Performance evaluations have shown that, for the same rate–distortion performance, our proposed scheme achieves an impressive reduction in the computational complexity of more than 82% compared to the full range motion estimation used by H.264.

***Index Terms*—**MPEG-2, H.264, Transcoding, Block Size

1. INTRODUCTION

Video coding standards have been constantly evolving during the last two decades. MPEG-2 is presently the dominant video coding standard in DTV broadcasting and DVD applications [1]. However, H.264 is quickly gaining ground mainly due to its higher compression efficiency [2]. Since the two standards are destined to coexist for some time, providing universal media access between them is becoming a hot research area. Transcoding from one standard to the other is the most cost effective and most attractive way to achieve such universal media access [3].

Since H.264 provides better compression performance than MPEG-2, one objective of MPEG-2 to H.264 transcoding is to improve the compression ratio of the resulting H.264 video. This may be achieved by taking advantage of some of the new features introduced by the H.264 standard such as variable block-size motion estimation (VBSME), multiple reference pictures in motion estimation (ME), and the ME quarter pixel accuracy. These features manage to significantly improve compression performance by providing more accurate motion vectors for the inter-frame prediction stage. Thus, one type of MPEG-2 to H.264 transcoding is to take advantage of the existing MPEG-2 video coding information in order to efficiently choose the coding parameters for H.264. Such research efforts have been presented in [4]-[7]. In all these cases, however, the number of calculations involved in finding the best match is relatively high. In [4]-[5], for instance, once a threshold is reached, a large amount of calculations is needed during the H.264 encoding process. In [6], machine learning techniques are used to predict 16x16 and 8x8 modes by calculating the mean and variance for all the 16 (4x4) blocks within each (16x16) MB. In addition to the above, the proposed schemes in [4]-[6] use information in the pixel domain, a disadvantage since spatial correlations between pixels is very high. This problem is addressed in [7] which uses information based on DCT coefficients to accelerate the macroblock mode selection.

We propose an efficient MPEG-2 to H.264 transcoding scheme which predicts the block size partition in H.264 using the MPEG-2 DCT coefficients. However, instead of using all 64 coefficients within each 8x8 block as in [7], we only use the first 3 AC low-frequency coefficients of each 8x8 block. Our method predicts the block size partition to be used in H.264 from the set {16x16, 16x8, 8x16, 8x8} directly without the use of full-search. Experimental results show that the proposed algorithm yields a rate-distortion performance that is almost the same as that resulting from the exhaustive full search. At the same time, the computational complexity is significantly reduced,, requiring less than 18% of the calculations used by the exhaustive full search method.

2. PROBLEM DESCRIPTION

In video coding, motion estimation is the process that searches for the block in the reference frame that best matches the current block. After the best match is found, the current block is subtracted from the best match and the difference is coded. In MPEG-2, motion estimation (ME) is implemented using only 16x16 pixel block size. On the other hand, H.264 performs motion estimation for a variety of block sizes which, in addition to 16x16, includes 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4 blocks (see Fig. 1). It has been shown that this H.264 feature yields optimized motion vectors with better matching accuracy which results in higher compression performance [8]. In general, MPEG-2 to H.264 transcoding schemes which re-use the MPEG-2 motion vectors result in high computational efficiency, but their matching accuracy is well below what could be achieved by H.264. In transcoding applications with emphasis on

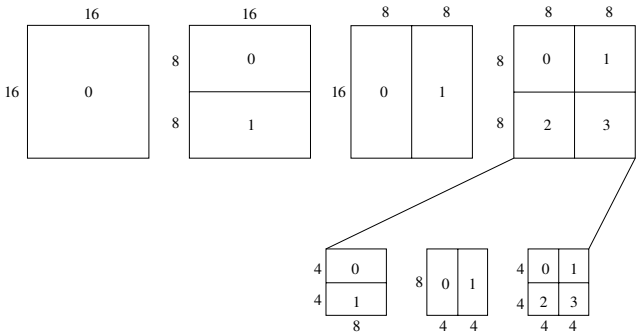


Fig. 1 Variable Block Sizes in H.264 Motion Compensation

improving the bit-rate, the main objective is to efficiently determine more accurate motion vectors, and take advantage of H.264's new motion estimation features. To this end, the following problem has to be addressed: how to efficiently determine the block sizes that will be chosen by the H.264 encoding process.

3. PROPOSED MB MODES PREDICTION ALGORITHMS

Addressing the problems described above, we designed a transcoding scheme that 1) limits the range of the available block sizes in the H.264 encoding process, and 2) predetermines the sizes of the blocks H.264 would use for every 16x16 MB that is not intra-frame coded in MPEG-2, without engaging any best match searching. An efficient method that exploits the relationship between the MPEG-2 DCT coefficients and the H.264 block size partitions is proposed.

3.1. Choosing only 16x16, 16x8, 8x16 and 8x8 block sizes

All the presently available MPEG-2 to H.264 transcoding schemes search the entire range of block sizes (i.e., 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4). This approach is counterproductive since one of our main objectives in designing an efficient transcoding scheme is to reduce the computational complexity of the overall system. It is thus desirable to limit the amount of the search effort but without sacrificing the compression ratio. For this reason, we compressed a large number of previously coded MPEG-2 video streams using H.264 to determine how different block sizes affect the compression performance. Our tests showed that block sizes below 8x8 resulted in negligible compression improvements which by no means justified the extra computational complexity involved. The main reason behind this founding is the fact that the MPEG-2 video streams had already gone through a quantization process which removed a significant amount of video details. Combining this finding with the fact that block sizes smaller than 8x8 need many more bits for storing the corresponding motion vectors, the overall cost savings for using these blocks becomes insignificant. Based on these observations,

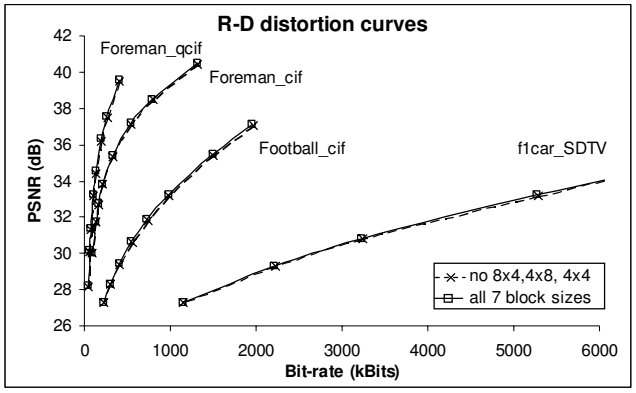


Fig. 2 Rate-Distortion curves of full range search and partial range search

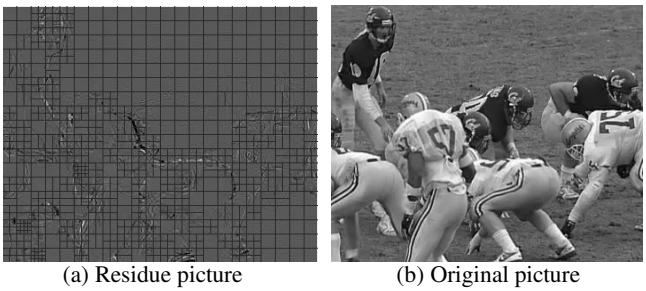
we choose to consider only the 16x16, 16x8, 8x16, and 8x8 block sizes for our proposed transcoding scheme. The more detailed experimental results are shown below.

For video test sequences, three picture resolutions we chose: 2 sequences in QCIF (176x144), 10 sequences in CIF (352x288), 2 sequences in SDTV (720x576). The bit-rates of MPEG-2 video streams for these three resolutions are 1M, 2M and 6M bits/sec, respectively. The MPEG-2 video streams were decoded back into the pixel domain and re-encoded using H.264. Fig. 2 shows the rate-distortion when all the block sizes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4) are used and also when only {16x16, 16x8, 8x16, 8x8} are employed. Only four video sequences are shown in Fig. 2 (1 SDTV, 2 CIF, 1 QCIF sequences). We observe that the difference between the R-D performances resulting from the partial range search and those from the full range search is negligible. Based on this observation, in our proposed transcoding coding scheme, we consider only the 16x16, 16x8, 8x16 and 8x8 block size partitions.

3.2. Proposed block size prediction

In MPEG-2 to H.264 transcoding, to accelerate the H.264 block size partition selection, we propose to use the information contained in the DCT coefficients of the residue picture resulting from the MPEG-2 motion compensation process. The residue picture is formed from the difference between the original picture and the reference one during the motion compensation process. Referring to Fig. 3, the left picture is the residue picture after MPEG-2 motion compensation, and the right one is the corresponding MPEG-2 encoded-decoded picture. The grid on Fig. 3(a) shows the block size partition chosen by H.264.

Fig. 3(a) clearly shows the correlation between the H.264 block sizes and motion activity. Whenever the 16x16 MB residues resulting from the MPEG-2 motion compensation are large, the H.264 block sizes are smaller than 16x16 since the MPEG-2 16x16 block size motion compensation did not yield good performance. Thus, to determine whether a 16x16 MB should be encoded as 16x16



(a) Residue picture (b) Original picture

Fig. 3 One frame of (a) residue picture and (b) original picture (Football CIF)

MB in H.264, the energy of the MPEG-2 motion compensated MB residues could be used. The following proposed method determines how such a MB should be partitioned using the information carried in already calculated MPEG-2 DCT coefficients.

Let X_{ij} denotes a pixel inside an 8x8 block ($0 \le i \le 7$; $0 \le j \le 7$), and F_{mn} denotes the DCT coefficients of the corresponding 8x8 block ($0 \le m \le 7$; $0 \le n \le 7$). Then F_{mn} is expressed as:

$$F_{mn} = C_m C_n \sum_{i=0}^{7} \sum_{j=0}^{7} X_{ij} \cos\frac{(2j+1)n\pi}{16} \cos\frac{(2i+1)m\pi}{16} \quad (1)$$

$$C_m = C_n = \sqrt{\frac{1}{8}} \ \ (m, n = 0), \ \ C_m = C_n = \frac{1}{2} \ \ (m, n > 0)$$

The basis patterns of F_{01}, F_{10} and F_{11} are shown in Fig. 4.

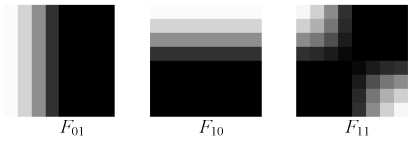


Fig. 4 The basis patterns of the first 3 low frequency AC coefficients

Our proposed algorithm is composed of two steps. The first step uses the energy of DCT coefficients in the current MB to estimate whether or not this MB should be partitioned. If the 16x16 MB is to be partitioned, we then find the directions of the different motions inside that block. If there are two different motion directions inside the block, the edge separating the pixels belonging to different motion directions will result in large DCT coefficients. Referring to Fig. 4, the directions of motion boundaries can be found from the edge information of the AC coefficients in each 8x8 block. Based on the directions of motion boundaries, we further estimate which block size partition among {16x8, 8x16, 8x8} to use in H.264. To determine which combination of DCT coefficients indicates the directions of motion boundaries inside a block, we implemented a large amount of experiments using different sub-sets of the DCT coefficients, such as middle-frequency coefficients or the first 6 AC low frequency coefficients. These performance evaluations have shown that using the first 3 AC low frequency coefficients offers the best trade-off between bit-rate performance and computational complexity.

Therefore, our proposed algorithms are based on the values of F_{01}, F_{10} and F_{11} in each 8x8 MPEG-2 block. A threshold is defined (T_{active}) to determine the active extent of AC coefficients. If the value of F_{01}, F_{10} or F_{11} is above T_{active}, there is visible vertical, horizontal or diagonal edge in the 8x8 block which corresponds to vertical, horizontal or diagonal motion boundaries, respectively. Similar to the coarse edge detection in [9], the following criteria are used to determine the dominant edge (motion boundaries) for each 8x8 block (see Table I).

Table I. Vertical, horizontal or diagonal dominant edges

Vertical-dominant	$\|F_{01}\| > \|F_{10}\|$, $\|F_{01}\| > T_{\text{active}}$ and $\|F_{11}\| \le T_{\text{active}}$
Horizontal-dominant	$\|F_{10}\| < \|F_{01}\|$, $\|F_{10}\| > T_{\text{active}}$ and $\|F_{11}\| \le T_{\text{active}}$
Diagonal-dominant	$\|F_{01}\| > T_{\text{active}}$, $\|F_{10}\| > T_{\text{active}}$ and $\|F_{11}\| > T_{\text{active}}$

There are four 8x8 blocks within one 16x16 MB. Based on the dominant edge of each 8x8 block, the following steps are applied to predict the block size partition for every 16x16 MB in H.264 encoding.

a) When all four 8x8 blocks have no dominant edge, the block partition is chosen to be 16x16.
b) When only one 8x8 block has dominant edge, if the dominant edge has diagonal directions, the block size partition is chosen to be 8x8. Otherwise, it is chosen to be 16x16.
c) When two 8x8 blocks have dominant edges, if these two blocks are not adjacent, the block size partition is chosen to be 8x8. Otherwise, if the motion boundaries in these two adjacent 8x8 block have the same direction (not diagonal), the block size partition is chosen to be 16x8 (horizontal) or 8x16 (vertical). Otherwise, the block size partition is chosen to be 8x8.
d) When three or four 8x8 blocks have dominant edges, the block size partition is chosen to be 8x8.

Note that, as in the case of full-search, our algorithm still checks the skip and intra-frame modes for every MB.

4. EXPERIMENTAL RESULTS AND COMPUTATIONAL COMPLEXITY ANALYSIS

The cascaded pixel-domain transcoding structure is used in our experiments. The TM5 and JM12.4 reference software codecs are used in our implementation [10][11]. Performance evaluations are carried out using a large number of different video test sequences and a wide variety of content. For instance, ten MPEG-2 videos with CIF resolution are used which are encoded at 2 Mbits/sec. The total frame number for each video is 300. The same quantization factor (QF) is used for every frame during H.264 encoding. Finally, the H.264 deblocking filter is turned on for all the tested transcoding schemes.

4.1. Experimental results of the proposed fast MB mode selection transcoding scheme

We compared our proposed MPEG-2 to H.264 transcoding scheme with two other schemes. One scheme re-uses the MPEG-2 MVs without further motion re-estimation during H.264 encoding. This represents the fastest transcoding scheme but compromises the bit-rate savings. The second scheme implements the full motion search using all 7 block size partitions. Evaluations over a large number of possible threshold values for T_{active} showed that the proposed algorithm achieves the best performance for T_{active} = 12. The threshold is adaptive with different sequences but only slightly changed. Fig. 5 shows the rate-distortion curves of four video sequences for all the tested transcoding schemes. We observe that the proposed method achieves almost the same rate-distortion performance as that of the full range motion estimation. Compared to the transcoding scheme which re-uses the MPEG-2 MV, the rate-distortion is significantly improved.

4.2. Computational complexity analysis

Table II shows the PSNR, bit-rate and executing time of motion estimation process of the proposed and full-search transcoding schemes. The results show that our proposed algorithm can reduce the computational complexity by 82%. The bit-rate only increases 1.5% on average. The full-range of motion estimation needs to search all 7 block size partitions. The proposed transcoding scheme, on the other hand, directly decides which block size mode to use and only implements motion estimation for the chosen mode. Thus, the proposed method needs only 1/7 of the computations involved in the full search method. Since the proposed algorithm needs to use the 3 AC coefficients of each 8x8 block to predict the chosen block size partition, the actual computational savings should a little bit smaller than 6/7 (approximately 86%). However, these extra computations are very small since we only use 3 out of the 63 AC coefficients and thus are expected to have a limited impact on the overall computational performance. This is in accordance with the 82% computational reduction achieved in our experiments.

Table II. Comparison of PSNR (dB), bit-rate (kbits/sec) and motion estimation time (s)

Sequences (quantization factor is equal to 27)	PSNR		Bit-rate		Motion estimation time	
	Full search	Proposed	Full search	Proposed	Full search	Proposed
Akiyo	39.43	39.33	166.89	170.59	34.7	5.6
Foreman	36.34	36.22	674.67	697.51	47.0	7.2
Football	35.88	35.81	1351.75	1355.39	59.5	10.5
Mobile	33.75	33.62	2814.09	2830.41	59.7	10.3

5. CONCLUSION

Motion re-estimating in MPEG-2 to H.264 transcoding significantly improves the resulting H.264 video's bit rate. This process however requires a high computational effort when full search is implemented. Our method uses only the first 3 AC coefficients within each 8x8 block to predict the block size partition mode in the H.264 encoding process. The experimental results show that our proposed method achieves very close rate-distortion performance as the full search. At the same time, the proposed transcoding scheme saves the computation efforts by more than 82%.

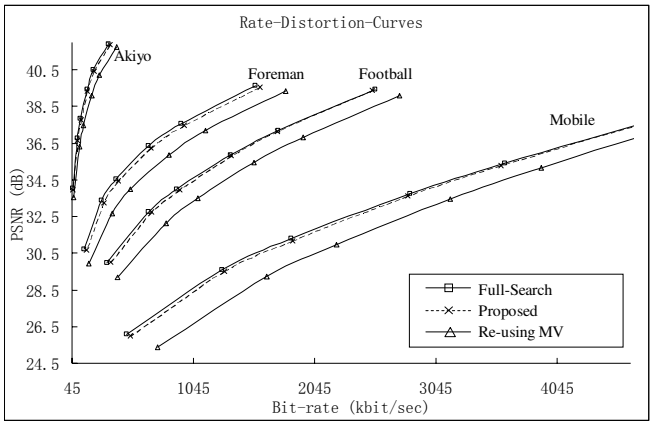


Fig. 5 Rate-distortion curves of different transcoding schemes

6. REFERENCES

[1] "Information technology – Generic coding of moving pictures and associated audio information: Video," ITU-T Recommendation H.262, International Standard 13818-2, ed. Feb. 2000.

[2] "Advanced video coding for generic audiovisual services, " ITU-T Recommendation H.264, International Standard 14496-10, ed. Mar. 2005.

[3] I. Ahmad, X. Wei, Y. Sun, Y-Q. Zhang, "Video Transcoding: An Overview of Various Techniques and Research Issues," *IEEE Trans. on Multimeida*, pp. 793-804, Oct. 2005.

[4] X. Lu, A. Tourapis, P. Yin and J. Boyce, "Fast Mode Decision and Motion Estimation for H.264 with a Focus on MPEG2/H.264 Transcoding," *ISCAS 2005*, pp. 1246–1249, May 2005.

[5] Z. Zhou, S. Sun, S. Lei, and M.T. Sun, "Motion Information and Coding Mode Reuse for MPEG-2 to H.264 Transcoding," *ISCAS 2005*, pp. 1230-1233, May 2005.

[6] G Fernández, H Kalva, P Cuenca, LO Barbosa, "Speeding-up the Macroblock Partition Mode Decision in MPEG-2/H.264 Transcoding," *ICIP 2006*, pp.869-872, Oct. 2006.

[7] G. Chen, Y. Zhang, S. Lin, F. Dai, "Efficient Block Size Selection for MPEG-2 to H.264 Transcoding," *Proceedings of the 12th annual ACM international conference on Multimedia*, pp. 300-303, 2004.

[8] M. Wien, "Variable Block-Size Transforms for H.264/AVC," *IEEE Tran. on CSVT*, pp. 604-613, Jul. 2003.

[9] B. Shen, I.K. Sethi, "Direct Feature Extraction from Compressed Images," *SPIE*, pp.404-414, 1996.

[10] MPEG-2 reference software, Test Model 5 (TM5). Available: http://www.mpeg.org/MPEG/MSSG/.

[11] H.264/AVC Reference software JM10.2. Available: http://iphome.hhi.de/suehring/tml/download/.