ADVANCED BLOCK SIZE SELECTION ALGORITHM FOR INTER FRAME CODING IN H.264/MPEG-4 AVC

Andy C. Yu and Graham R. Martin

Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom
Email: {AndyCYu, Graham.Martin}@dcs.warwick.ac.uk

ABSTRACT

A fast inter-mode selection algorithm is proposed to improve the encoder efficiency of the H.264/MPEG-4 AVC standard, but with insignificant degradation in picture quality. The Modified Fast Inter-mode selection (MFInterms) algorithm extends previous work to provide a more efficient prediction of mode decision. The strategy incorporates temporal similarity detection and the detection of different moving features within a macroblock. Simulation results demonstrate a speed up in encoding time of up to 73% compared with the H.264 benchmark.

1. INTRODUCTION

The JVT (Joint Video Team) introduced a number of advanced features in H.264 or MPEG-4 AVC [6] to provide improvements in coding performance. One of the new features is multi-mode selection, which is the subject of this paper. In the H.264 coding algorithm, block-matching motion estimation is an essential part of the encoder to reduce the temporal redundancy between frames. The difference with other standards, however, is that the block size is no longer fixed, but ranges from 4×4 to 16×16 for inter-frame coding.

In order to choose the best block size for a macroblock, the H.264 standard makes use of computationally intensive Lagrangian rate-distortion (RD) optimization, the general equation of which is:

$$J_{\mathrm{mode}} = D + \lambda_{\mathrm{mode}} * R \qquad (1)$$

where J_{mode} is the rate-distortion cost (RD cost) and λ_{mode} is the Lagrangian multiplier. D is the distortion measure between the original macroblock and the reconstructed macroblock located in the previous coded frame, and R reflects the number of bits associated with choosing the mode and macroblock quantiser value, Qp, including the bits for the macroblock header, the motion vector(s) and all the DCT residue blocks. In inter-frame coding, possible modes are:

$$\mathrm{mode} \in \left\{ \begin{array}{l} SKIP, I4MB, I16MB, 16\times16, \\ 16\times8, 8\times16, 8\times8, 8\times4, 4\times8, 4\times4 \end{array} \right\} \qquad (2)$$

where *SKIP* is a direct copy from the previous frame; *I4MB* and *I16MB* are the intra-modes predicted from encoded adjacent pixels; and the others represent the inter-modes with the different block-sizes depicted in Fig.1.

The optimal mode (mode decision) for a macroblock is selected as that which produces the least RD cost. The H.264 standard employs a brute force algorithm to search through all possible block-sizes to find a motion vector for each macroblock. Thus, the computational burden of the searching process is far more demanding than any existing video coding algorithm.

Recently, a fast inter mode selection (FInterms) algorithm [1] was proposed to alleviate the encoder complexity due to mode decision, while maintaining picture quality and other coding efficiency. The FInterms algorithm makes use of a spatial content measure to assign different levels of inter-modes to each macroblock depending on its complexity. The lower the complexity of the macroblock content, the fewer inter-modes the encoder has to check, and vice versa.

In this paper, we propose a substantially improved algorithm, Modified FInterms (denoted MFInterms), to provide a more effective prediction for choosing the mode decision. The remainder of this paper is organized as follows: In Section 2, a review of the FInterms algorithm is provided, Section 3 describes the proposed modified inter mode selection (MFInterms) algorithm, and simulation results of FInterms and MFInterms are compared in Section 4.

2. REVIEW OF PREVIOUS WORK

Success of the FInterms algorithm is achieved by discarding the least possible modes according to the

0-7803-8554-3/04/$20.00 ©2004 IEEE.

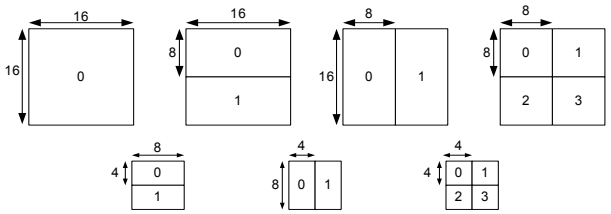

Fig. 1 Inter-prediction modes with 7 different block sizes ranging from 4×4 to 16×16.

spatial complexity of the macroblock's content and the mode knowledge of the previously encoded frame.

The FInterms algorithm makes use of a general tendency: a mode having a smaller partition size may benefit detailed areas, whereas a larger partition size is more suitable for homogeneous areas [2]. Thus, the first part of the FInterms algorithm is a complexity measurement defined as follows:

$$R_{\mathrm{C}} = \frac{\ln(E_{AC})}{\ln(E_{\max})} \tag{3}$$

where R_{C} is a complexity ratio; E_{AC} represents the total energy of the high-frequency component (AC component) in the current macroblock; $E_{\max}$ denotes the largest variance of a macroblock that comprises a checkerboard pattern with the maximum and minimum permissible pixel values. Compared to the edge map approach with Sobel operators [4-5], the proposed complexity measurement is easy to compute.

The FInterms algorithm assigns one of three categories, representing different levels of inter-mode examination, to each macroblock depending on its spatial complexity. Generally, a homogeneous macroblock is more likely to require examination of fewer inter-modes compared to a highly detailed macroblock. Nevertheless, the complexity measurement is not the only factor by which to decide whether a macroblock should undergo a different level of inter-mode examination. The assigned category is amended according to the mode information of the macroblock at the same location in the previous frame. For example, if the macroblock in the previous frame is encoded with the inter-mode smaller partition size, then the current macroblock is searched with more inter-modes. If the reverse situation is true, no amendment is done.

3. PROPOSED MODIFIED INTER-MODE SELECTION (MFINTERMS) ALGORITHM

High efficiency of the modified fast inter-mode selection algorithm (MFInterms) is achieved by introducing two additional measurements targeted at two kinds of encoded macroblocks: (a) macroblocks encoded with *SKIP* mode (direct copy from the corresponding macroblock located at the same position in the previous frame); (b) macroblocks encoded by the inter-modes with larger decomposed partition size (greater than 8×8 pixels). By successfully identifying these two kinds of macroblocks, the encoder is exempted from examining them with all possible inter-modes, which saves encoding time. Fortunately, the aforementioned macroblocks can be detected in terms of the temporal similarity between two macroblocks and the motion consistency of a macroblock. The following two subsections discuss these measurements.

3.1. Measurement of temporal similarity

The *SKIP* mode is normally assigned to a macroblock that comprises almost identical pixel information to that of the corresponding macroblock in same position in the previous frame, for example, in areas representing a static background. The macroblocks coded with *SKIP* mode (skipped macroblocks) can be easily detected by comparing the residue between the current macroblock and the previously encoded macroblock with a threshold as follows:

$$T(S_{\mathrm{residue}}) = \begin{cases} 1, & S_{\mathrm{residue}} < Th \\ 0, & S_{\mathrm{residue}} > Th \end{cases} \tag{4}$$

$$S_{\mathrm{residue}} = \sum_{m}\sum_{n}\left|\mathbf{B}_{m,n,t} - \mathbf{B}_{m,n,t-1}\right| \tag{5}$$

where S_{residue} is the sum absolute difference between $\mathbf{B}_{m,n,t}$ and $\mathbf{B}_{m,n,t-1}$, which represent current and previous macroblocks, respectively. If $T(S_{\mathrm{residue}}) = 1$, the current macroblock is a skipped macroblock. However, performing this calculation for every macroblock further increases the encoding time. Lim et al. [4] suggest performing temporal similarity checking if the current 16×16 macroblock has zero motion. This necessitates each macroblock, including skipped macroblocks, to undergo at least one complete cycle of motion estimation. If the encoder can detect the skipped macroblocks without a priori knowledge, then a significant proportion of the encoding time will be saved.

Generally, the skipped macroblocks tend to occur in clusters, such as in a patch of static background. Thus, we propose that the current macroblock has to undergo temporal similarity detection if one of the encoded neighbours is a skipped macroblock. The temporal similarity detection is implemented according to (4) and (5), but we propose an adaptive spatially varying threshold, Th_{ASV}, to replace Th.

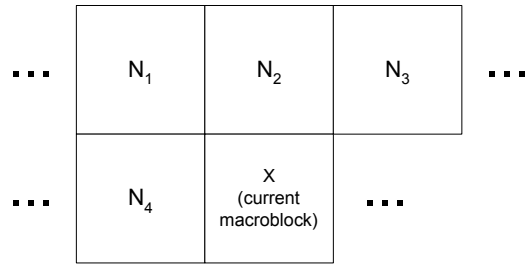

Fig. 2 The relative position of four nearest encoded neighbours of the current macroblock.

$$Th_{ASV} = C * \min(S_{N1}, S_{N2}, S_{N3}, S_{N4}) \quad (6)$$

where C is a constant; S_{N1}, S_{N2}, S_{N3}, and S_{N4} are the sum absolute difference of four nearest encoded neighbours, N_1, N_2, N_3, N_4, as shown in Fig. 2. They are valid and pre-stored in the system if and only if their corresponding macroblocks are skipped macroblocks. Thus, (6) reduces in size according to the number of skipped neighbouring macroblocks.

3.2. Measurement on block-based motion consistency

The tendency that the inter modes with larger partition size (of sizes 8×8, 8×16, 16×8, and 16×16 pixels) are more suitable to encode homogeneous macroblocks has been verified by a number of authors [1,2,4]. By contrast, macroblocks containing moving features appear more detailed and therefore require use of smaller block sizes. Thus, the proposed algorithm suggests checking the motion vector of each 8×8 block decomposed from a highly detailed macroblock. If consistency among motion vectors exists, the proposed algorithm checks the inter-modes with partition size greater than 8×8, otherwise, all possible inter modes are searched.

3.3. The algorithm of MFInterms

Fig. 3 shows a flowchart of the proposed MFInterms algorithm, which is summarized as follows:

A.1. Turn off all flags including SKIP, INTRA and all inter-modes

A.2. Check if one of the four nearest neighbours of the current macroblock is a skipped macroblock. Implement A.3 if the situation is true. If not, go to A4.

A.3. Obtain a threshold, T_{ASV}, from (6). Compare T_{ASV} with the sum absolute difference between the current macroblock and the previous macroblock at the same position. If the sum is smaller than the threshold, turn on the flag for SKIP only. Otherwise, continue to A.4.

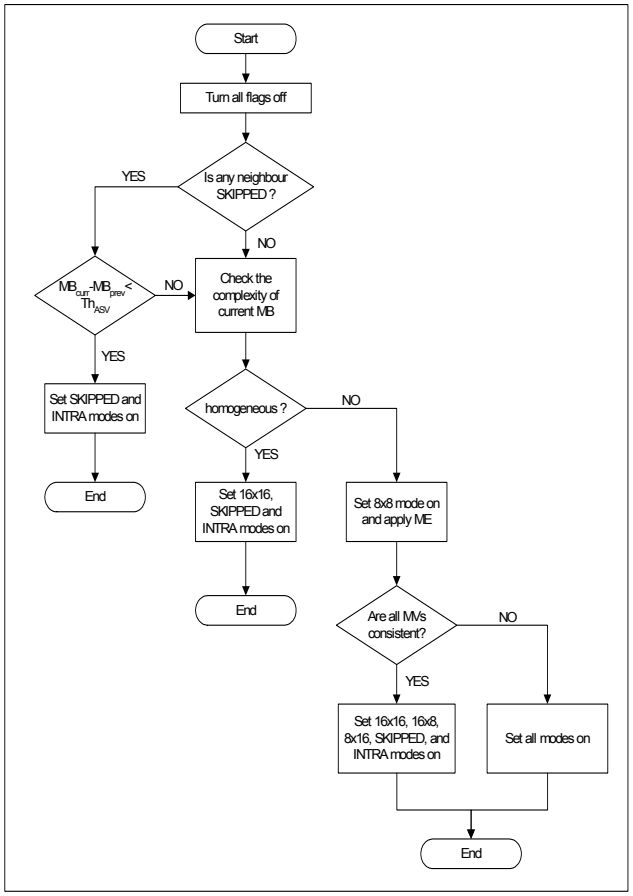

Fig. 3 Flowchart of the proposed Modified Fast Inter-Mode Selection (MFInterms) algorithm.

A.4. Check the complexity of the macroblock using equation (3). If the current macroblock is homogeneous, turn on the flag for INTRA and the inter-mode with partition size 16×16. Otherwise, continue to A.5.

A.5. Decompose the highly detailed macroblock into four non-overlapping 8×8 blocks. Check whether the motion vectors of the four blocks are consistent. If consistent, check the flags of INTRA, and the inter-modes with partition size 8×16, 16×8, and 16×16 and go to A.7. Otherwise, continue to A.6.

A.6. Turn on all flags and use sixteen motion vectors obtained from inter-modes with partition size 4×4 as searching points for the inter-mode with partition size 4×8 and 8×4 rather than performing full search. Then, continue to A.7.

A.7. Utilise the four motion vectors obtained from four 8×8 blocks as searching points for the inter-modes with partition size 8×16, 16×8, and 16×16 rather than performing full search.

4. SIMULATION RESULTS

This section compares the simulation results employing the proposed MFInterms and previously proposed FInterms algorithm [1]. All the simulations were programmed using C++. The computer used for the simulations was a 2.8GHz Pentium 4 with 1024MB RAM. The testing benchmark was the JM6.1e software version provided by the Joint Video Team (JVT) [6]. The selected sequences are of QCIF resolution (176×144 pixels) and classified into three different classes, i.e., Class A, B, and C according to their spatial correlation and motion information. The other settings are as follows: all the sequences are defined in a static coding structure, i.e., one I-frame is followed by nine P-frames (1I9P), with a frame rate of 30 frames per second and no skip frame throughout the 100 frames. The precision and search range of the motion estimation is set to ¼ pixel and ±8 pixels, respectively. Lastly, Context-based Adaptive Binary Arithmetic Coding (CABAC) is used to perform entropy coding and a static quantizer value, Qp = 29, is applied throughout the simulation.

The table summarises the simulation results of the two algorithms – FInterms and MFInterms in terms of PSNR difference, bit rate difference and speed up compared with JM6.1e, the testing benchmark. The general trends are identified as follows: both fast algorithms introduce less than 0.08 dB of PSNR degradation in Class A and Class B, and approximately 0.10 dB in Class C. Note that there is insignificant PSNR difference between the MFInterms and FInterms algorithms. As to compression ratio, the proposed MFInterms produces slightly higher bit rates than FInterms especially in the Class C sequences, however the bit differences for most test sequences are less than 5%. Nevertheless, the picture degradations and bit rate increase are generally considered within acceptable range as human visual perception is unable to distinguish a PSNR difference of less than 0.2dB.

Significantly, the new MFInterms algorithm provides a saving of 28-50% in encoding time for Class C sequences when compared with the JM6.1e benchmark. The saving for Class A and B sequences is 60-73%. The previously reported FInterms algorithm provided improvements of only 18-25% and 22-31%, respectively. The reason that a significant proportion of the encoding time is saved with the MFInterms algorithm is that skipped macroblocks are detected accurately and are encoded with *SKIP* mode, obviating the need for other mode examinations. As a result, the encoding time of a P frame could be shorter than that of an I frame if a sequence contains a significant number of skipped macroblocks.

5. CONCLUSIONS

In this paper, a fast algorithm for inter-mode selection, MFInterms has been proposed. The simulation results show that MFInterms has more than double the computational efficiency (up to 73%) and introduces insignificant picture degradation and bit rate increase compared to JM6.1e, the testing benchmark.

6. REFERENCES

[1] A. Yu, "Efficient block-size selection algorithm for inter-frame coding in H.264/MPEG-4 AVC," accepted by *ICASSP 2004*, May 2004, Montreal, Canada.

[2] I. Richardson, "H.264/MPEG-4 video compression: video coding for next-generation multimedia," published by Wiley, 2003.

[3] F. Pan, X. Lin, R. Susanto et al., "Fast mode decision for intra prediction," JVT-G013, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Mar. 2003, Pattaya, Thailand.

[4] K. Lim, S. Wu, J. Wu et al., "Fast inter mode selection," JVT-I020, ISO/IEC JTC1/SC29/WG11 and ITU-I SG16 Q.6, Sep. 2003, San Diego, U.S.

[5] JVT reference software, JM 6.1e, downloaded from http://bs.hhi.de/~suehring/tml.

[6] "Information technology – coding of audio visual objects – Part 10: advanced video coding," ISO/IEC 14496-10:2003, Dec. 2003.

	Sequences	PSNR Difference		Bit Rate Difference		Speed up cf. JM6.1e	
		FInterms	MFInterms	FInterms	MFInterms	FInterms	MFInterms
A	Ship Container	-0.02 dB	-0.06 dB	0.10 %	0.37%	30.85%	72.94%
	Sean	-0.05 dB	-0.07 dB	0.44 %	-0.11%	29.87%	69.63%
B	Silent	-0.07 dB	-0.08 dB	1.47 %	3.73%	26.64%	60.22%
	News	-0.07 dB	-0.07 dB	1.34 %	1.52%	22.04%	62.08%
C	Stefan	-0.09 dB	-0.13 dB	5.36 %	8.90%	18.34%	28.72%
	Table Tennis	-0.08 dB	-0.09 dB	5.26 %	6.57%	24.74%	49.41%