Available online at www.sciencedirect.com
SCIENCE DIRECT®
J. Vis. Commun. Image R. 17 (2006) 217–242
JOURNAL OF VISUAL Communication & IMAGE Representation
www.elsevier.com/locate/jvci

Fast motion search with efficient inter-prediction mode decision for H.264

Chih-Hung Kuo [a], Meiyin Shen [b], C.-C. Jay Kuo [b,*]

[a] *Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan*
[b] *Department of Electrical Engineering and Integrated Media Systems Center, University of Southern California, Los Angeles, CA 90089-2564, USA*

Received 7 February 2004; accepted 10 May 2005
Available online 6 July 2005

Abstract

A fast inter-prediction mode decision and motion search algorithm is proposed for the H.264 video coding standard. The multi-resolution motion estimation scheme and an adaptive rate-distortion model are employed with early termination rules to accelerate the search process. With the new algorithm, the amount of computation involved in the motion search can be substantially reduced. Experimental results show that the proposed algorithm can achieve a speed-up factor ranging from 60 to 150 times as compared to the full-search algorithm with little quality degradation.

Keywords: H.264; Motion estimation; Fast motion search; Video coding

1. Introduction

H.264 is the latest video coding standard jointly developed by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC MPEG [1]. It is widely regarded as the state-of-the-art video coding standard since it is able to save up

* Corresponding author. Fax: +213 740 4651.
E-mail addresses: chkuo@ee.ncku.edu.tw (C.-H. Kuo), meiyinsh@sipi.usc.edu (M. Shen), cckuo@sipi.usc.edu (C.-C. Jay Kuo).

1047-3203/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2005.05.001

to 50% in the file size (or the bandwidth consumption) while delivering the same visual quality as compared to existing standards. The excellent coding gain of H.264 is achieved at the expense of the high encoder complexity. How to reduce the complexity while preserving good coding performance is an interesting research problem. In this work, we will investigate methods to reduce the complexity of the H.264 encoder with special attention on fast variable block-size motion search.

As a block-based motion-compensated predictive coder, H.264 is similar to its prior standards in the general framework yet with improvements on some coding modules such as variable block-size motion estimation, intra-frame prediction, in-loop deblocking filter, and context-adaptive arithmetic coding, etc. The main reason for H.264 to outperform other standards is that it allows the choice among multiple modes in several coding components as long as this freedom can provide a substantial coding gain. However, this freedom also implies more computation since one has to select the best mode among all possible modes to give the best rate-distortion tradeoff.

For inter-frame prediction, H.264 allows blocks of variable sizes and shapes. To be more specific, seven modes of different sizes and shapes, i.e., 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, are supported in H.264. Smaller blocks, which intend to better characterize the motion behavior of a region, can reduce the prediction error and provide better visual quality due to the less visible blocking artifact. Besides variable block sizes and shapes, the use of higher precision in the motion vector representation also improves the coding gain. H.264 supports 1/4-pel motion accuracy. In addition, H.264 allows the use of multiple reference frames, which is useful in dealing with periodic motion in the sequence. To achieve the highest coding gain, an optimized H.264 encoder will estimate the best motion vector (MV) by searching among multiple reference frames and multiple block modes with 1/4-pel motion vector precision. With all these modes in place, the computational complexity of motion estimation increases dramatically as compared with previous standards. This is one major bottleneck for the H.264 encoder.

Let us briefly review motion vector search algorithms. The full search algorithm checks every displacement inside the designated search window. It is the most straightforward way to find the optimal motion vector. Many sub-optimal algorithms have been proposed to reduce the number of block matching operations in the search process. Some fast search algorithms were developed in the past under the assumption of the unimodal error surface. They include: the three-step search (3SS) [2], the new three-step search (N3SS) [3], the four-step search (4SS) [4], the block-based gradient descent search (BBGS) [5], and the diamond search [6]. In these algorithms, the search process is divided into several steps, where several possible displacements were checked at each step and the one with the minimum distortion will be picked as the center for the search at the next step. Although the search speed can be improved, they may result in significant quality degradation in comparison with the full search scheme at the same bit rate.

A better strategy to achieve fast search is to predict the good initial displacement of the motion vector, and perform early termination with reliable stopping criteria to avoid unnecessary block matching. Chalidabhongse and Jay Kuo [7] first investigated a fast search algorithm that predicts initial motion vectors from neighboring MBs in both multiresolution and spatial–temporal dimensions. Although the scanning order of macroblocks may be non-causal, their proposed algorithm can speed-up by a factor of 100–300 with little quality degradation due to better prediction of the initial motion vector. Recently, two fast motion estimation algorithms, i.e., the Motion Vector Field Adaptive Search Technique (MVFAST) [8] and the Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) [9] were adopted by MPEG-4 Part-7 as an optimization model [10]. They both attempted to exploit more correlations from spatial–temporal neighboring macroblocks. The basic ideas of these two algorithms are: (i) select initial MV predictors from spatially and temporally adjacent blocks to perform the diamond search (DS); (ii) adaptively choose small or large diamonds as the search pattern based on the local motion activity; (iii) apply the early termination principle to avoid inefficient SAD matching operations. These two algorithms provide a significant improvement over traditional fast search algorithms in terms of visual quality and complexity reduction. The obtained visual quality from MVFAST and PMVFAST is very similar to that obtained by the full search scheme. However, it considers motion search for a fixed-size block only.

Since H.264 allows motion estimation and compensation with variable block sizes and shapes, some research effort has to be made to enhance the efficiency of the variable block-size motion search. Various approaches have been proposed to speed up the decision process in the literature recently, e.g. [11–20]. For example, some criteria for mode decision based on the texture homogeneity were investigated to reduce the number of searching modes [11–13]. The partition type of a MB was predicted in [14] according to the modes of its surrounding MBs. A fixed mode was searched in advance and the preliminary distortion was computed in [15–17] to decide the most probable modes for the following searching steps. Xu et al. [18] integrated their own UMHexagonS algorithm with an early termination strategy as well as some motion vector predictions to improve coding efficiency. In [19,20], the Enhanced Predictive Zonal Search (EPZS) [21] algorithm was applied in the context of H.264 coding with early termination criteria and motion vector predictions.

In this work, we provide a more comprehensive scheme by introducing a rate-model as a basis of the fast mode selection technique. The fast motion search algorithm is integrated so that the best motion candidate can be found without searching all modes exhaustively. Therefore, our algorithm can perform much faster search than previous work and achieve up to a speed up factor of 150 as compared to the full search algorithm with good implementation. The rest of this chapter is organized as follows. An overview of H.264 motion estimation is given in Section 2. The proposed fast searching algorithm is described in Section 3. Experimental results are presented in Section 4. Finally, concluding remarks are given in Section 5.

2. Overview of H.264 motion estimation

2.1. Tree-structured motion estimation

For better adaptation to motion details, H.264 adopts a tree-based decomposition to partition a macroblock (MB) into smaller sub-blocks of specified sizes. For example, one MB of size 16×16 may be kept as is, decomposed into two rectangular blocks of size 8×16 or 16×6, or decomposed into four square blocks of size 8×8. If the last case is chosen (i.e. four 8×8 blocks), each of the four 8×8 blocks can be further split to result in more sub-macroblocks. There are four choices again, i.e., 8×8, 8×4, 4×8, and 4×4. These partitions result in a large number of possible block decompositions for each MB. An example of an MB with tree decomposition is shown in Fig. 1.

If an MB is divided, each sub-macroblock inside the MB requires a separate motion vector. For example, if an MB is coded using Inter-8×8, and each 8×8 sub-macroblock is coded using Inter-4×4, 16 motion vectors will be transmitted for this MB. All motion vectors as well as the partition information should be coded and transmitted. In general, a partition with larger block sizes requires fewer bits to represent the associated motion vectors and the partition type, but it may need more bits to encode motion compensation residuals if encoded areas contain high motion details. On the contrary, a partition of smaller block sizes may give smaller residuals after motion compensation but requires more bits to represent motion vectors and partition types. Thus, an optimal tree-structured motion compensation is to find the best combination of partition block sizes to minimize the final coded bits for each macroblock.

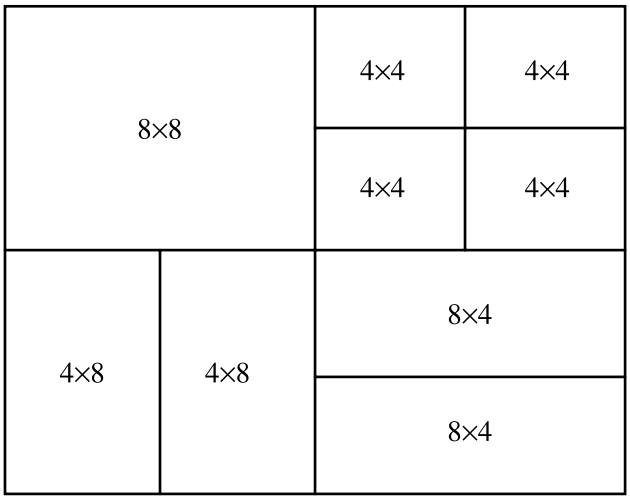

Fig. 1. Partitioning of a MB with tree-structured decomposition.

2.2. Notations

We use P to denote the partition type of a macroblack and V the set of motion vectors of all divided sub-blocks. Let $E(\mathrm{MB}_i; P, V)$ represent the number of encoded bits for the ith macroblock with partition P and V be the set of motion vectors associated with each sub-MB. Then, the encoding process is to find the best partition P and motion vectors V so that the bit number b_i of the ith MB is minimized:

$$B_i = \min_{P,V} E(\mathrm{MB}_i;P,V).$$

Among the bits to encode an MB, those represent motion vectors and residual signals are most critical in the motion search component. Hence, the motion search process is equivalent to finding

$$\min_{P,V} E_{\mathrm{M}}(\mathrm{MB}_i;P,V) + E_{\mathrm{R}}(\mathrm{MB}_i;P,V), \tag{1}$$

where E_{M} and E_{R} denote the bit numbers required to encode the motion vectors and residual signals, respectively. Usually, there exists a tradeoff between these two terms. When the MB is partitioned into more smaller blocks, E_{M} will increase while E_{R} will decrease. On the other hand, if the MB is not partitioned, then E_{M} is the smallest while E_{R} becomes the largest. The search process is to find the best tradeoff between these two terms to keep the best overall rate-distortion performance.

2.3. Distortion measurement for block matching

To find the optimal solution for Eq. (1), one should perform the whole encoding process for every combination of all modes and their possible associated motion vectors. This extremely large amount of computation is usually not acceptable in practice. The practical method is to search for the best displacement in prior to the operations of DCT transform and entropy coding. Block-matching techniques that minimize a cost function measuring the mismatch between the current block and the candidate block within the search area are commonly adopted due to its simplicity. The most widely used distortion measure is the sum of absolute difference (SAD), which is defined as

$$\mathrm{SAD}(m_x, m_y) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |F_k(x+i, y+j) - F_{k-1}(x+i+m_x, y+j+m_y)|, \tag{2}$$

where $F_k(x,y)$ is the (x,y)th pixel in the current frame and F_{k-1} is the reconstruction of the previous frame, (m_x,m_y) is the displacement relative to the block with size $M \times N$. In this work, the mean of absolute difference (MAD), defined as

$$\mathrm{MAD}(m_x, m_y) = \frac{\mathrm{SAD}(m_x, m_y)}{M \times N}, \tag{3}$$

is adopted for the distortion measure.

Most practical motion estimation components search for an MB's motion vector with the smallest associated MAD. This process can be done without performing

operations of transform and entropy coding. However, in some rare cases the searched motion vectors with least MAD may be not exactly the same as the solution of Eq. (1) and hence encoded bit numbers are not always optimal.

2.4. Fast full search in H.264

In the motion estimation for H.264, there are total seven possible block sizes to be searched for each MB. Following the naming convention of the reference software, we let modes 1–7 denote block sizes of 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, respectively. The most straightforward way to find the optimal motion vectors of an MB is to use the full search (FS) over all seven modes. The FS algorithm, which evaluates MAD at all locations of a given search window, requires a large amount of computation. Therefore, in the reference software of H.264, an alternative implementation of the FS algorithm is provided.

The idea to speed up the full search algorithm is to use the bottom-up merge process. First, an FS is performed for all 4×4 partitions. The corresponding MADs and motion vectors are stored for later used. Then, the MAD of a larger partition (e.g., 4×8, 8×4, or higher) can be obtained simply by averaging the MADs of all 4×4 sub-blocks inside this partition. With this approach, block-matching operations for larger partitions can be saved. The search result is the same as exhaustively searching all seven modes while the amount of MAD computation can be reduced to 1/7. The major drawback of this method is the MADs for all displacements of all 4×4 blocks need to be stored in calculating the MAD of a larger partition. Therefore, it cannot take advantage of many fast search algorithms developed in the past, which reduce the number of search positions but cannot guarantee the global minimum MAD. The MAD values of some displacements of 4×4 blocks may be not available in the fast algorithms. As a result, this bottom-up merge process still demands a considerable amount of computation and memory storage in finding MAD of the displacements which are not yet visited.

Another way to speed up the motion search of H.264 is directly applying a fast search algorithm to each mode. This approach will sacrifice some coding performance since most fast search algorithms only find sub-optimal solutions to trade for more computational saving. However, these fast search algorithms cannot be as effective for H.264 as in prior standards because of more modes. For example, in prior coding standards, the MVFAST algorithm can achieve a speed-up factor of around 100 as compared to the FS algorithm. However, in H.264, this method can only speed up by a factor of 14. The speed-up factor is reduced to 1/7 since there are seven times of locations to be searched in this fast algorithm.

3. The proposed algorithm

Fig. 2 depicts the block diagram of the proposed algorithm. First, the original image frame is low-pass filtered and sub-sampled to get a lower resolution image. Then, motion estimation is performed on the lower resolution image, and the result is used

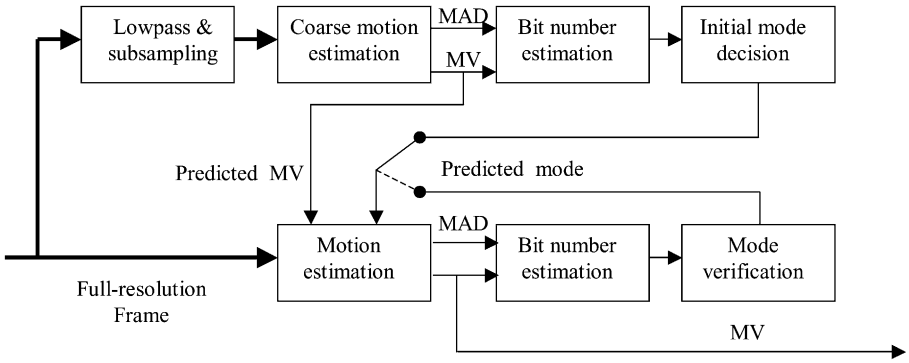

Fig. 2. The block-diagram of the proposed motion search algorithm.

to predict the initial motion vector and the mean of absolute differences (MADs) of the corresponding MB in the original image. With the MAD information, an initial search mode (of a certain block size) is determined using the estimated encoded bits and some predefined threshold. After the initialization, the motion vector of an MB in the original resolution image is refined by performing fast motion search with the initial block size. The number of encoded bits is then estimated again using the MAD of the refined motion vector. The updated encoded bits are used to verify the mode decision and determine the next mode to search. In addition, motion search will be terminated if the number of estimated encoded bits is smaller than an adaptive threshold. As a result, by reducing number of modes to be searched, the amount of computation can be saved remarkably. In the following, we will describe each component of our algorithm in detail.

3.1. Estimation of bits for residual coding

In the reference software of H.264, the encoder finds the best MB partition by minimizing the cost function:

$$J_{\text{mode}} = \text{SAD} + \lambda_{\text{motion}} \times R, \tag{4}$$

where R represents the bits used to encode the motion information, λ_{motion} is the Lagrangian multiplier set according to the quantization step [22]. In spite of its simple computation, this model is found not accurate enough for our fast mode decision algorithm. It is necessary to develop a more elaborated rate-distortion model to estimate the number of bits $E_{\text{R}}(\text{MB}_i; P, V)$ for the residual signal coding based on the pixel variance information in an MB. As a matter of fact, several rate-distortion models have been proposed for the frame level, e.g. [23,24]. In the current context, we are concerned with the coding at the MB level and the distortion will be about the same with respect to the same quantization step. The rate-distortion optimization process can be greatly simplified since we only have to minimize the encoding bit rates.

The estimation of encoded bits is derived with the assumption that the residual signal is a zero-mean i.i.d source with the Laplacian distribution. The entropy of a pixel can be obtained explicitly via [23]

$$B(Q,\sigma) = H_2(\sqrt{\gamma}) + \sqrt{\gamma}\left(1 + \frac{H_2(\gamma)}{1-\gamma}\right), \tag{5}$$

where the binary entropy function $H_2(y) = -y\log_2 y - (1-y)\log_2\ (1-y)$ and $\gamma = \mathrm{e}^{-\alpha Q}$, where Q is the quantization step and σ is the variance of pixels in an MB. The entropy $B(Q,\sigma)$ can be viewed as the average bits to represent a symbol if a perfect entropy encoder is applied. In later sections, we will use it to estimate the encoded bit number $E(\mathrm{MB}_i; P, V)$ for the MB residual provided that the variance σ and quantization step Q is known.

In Fig. 3, examples are given with various quantization steps. The solid curve is the theoretical result computed from Eq. (5) while small dots represent actual experimental results conducted by encoding a 300-frame CIF video 'Forman' with H.264 JM software. Since the practical real-world encoder may not be an ideal entropy coder and signals inside a single MB may have a wide distribution, it is reasonable that the number of bits to represent an MB may deviate from the theoretical model value.

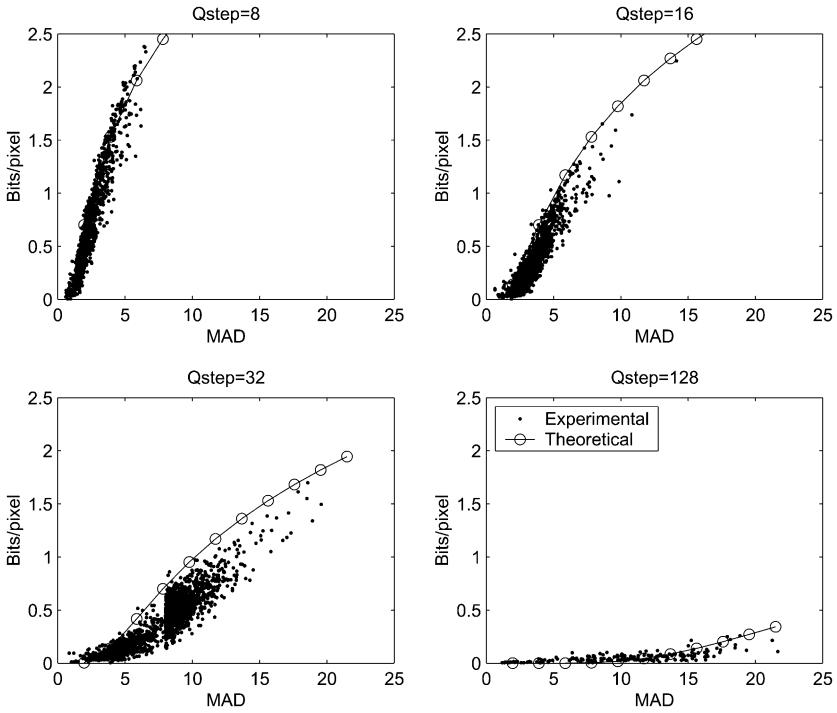

Fig. 3. Bits per pixel vs MAD with quantization steps 8, 16, 32, and 128 using H.264 quantization parameters QP = 22, 28, 34, and 46, respectively.

However, these figures indicate that the derived theoretical curves fit the general trend of experimental results well. Since we only use this model for mode selection, it is accurate enough for this purpose. In the practical implementation, the values of $B(Q,\sigma)$ can be computed off-line and stored in a lookup table. The estimated bits can be simply found from the lookup table after finding the nearest index of the value Q/σ.

Furthermore, we improve the model to make it an adaptive one so that it can adjust to the encoding context. The adaptive model for rate control in the frame level has been studied by researchers. For example, Lee et al. [25] proposed a second order polynomial for the rate model, and its coefficients are updated by linear regression. However, we observe that the polynomial of order 2 is not enough for the MB-level rate approximation. It demands a third-order polynomial to approximate Eq. (5), i.e.

$$B(\gamma) = w_0 + w_1\gamma + w_2\gamma^2 + w_3\gamma^3 + o(\gamma^4),$$

where $o(\gamma^4)$ represents negligible higher-order terms. Fig. 4 shows the theoretical curve as well as the second-order and the third-order approximations as a function of γ. We see that when γ approaches one, which corresponds to the case of higher MAD values, the second-order curve deviates a lot from the theoretical one. The proposed third-order polynomial provides a better estimation.

To be adaptive to the context of the encoding process, the coefficients of the polynomial are updated after the encoding of each MB. We use the least mean square (LMS) algorithm to update the coefficients after the nth MB is encoded:

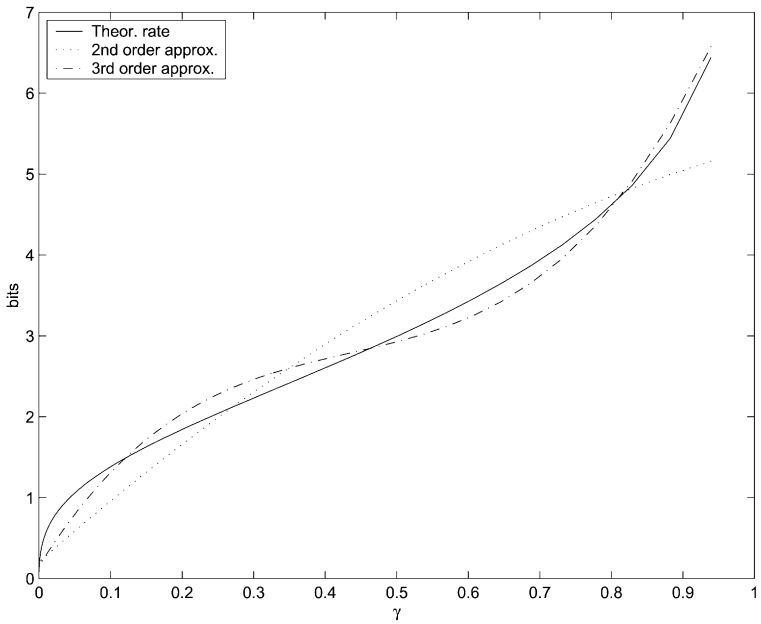

Fig. 4. The approximate bit number vs $\gamma = e^{-\alpha Q}$.

$$w_i(n+1) = w_i(n) + \mu\gamma(n)^i e(n), \quad i = 0, 1, 2, 3,$$

where μ is the weighting parameter and

$$e(n) = R(n) - \sum_{i=0}^{i=3} w_i(n)\gamma(n)^i$$

is the prediction error. The LMS algorithm is a well-known technique for adaptive filtering. It is simple and robust and we find it suitable for the MB level adaptation.

In the simulation, we use the coefficients of the Taylor series expansion of Eq. (5) as initial values, and set weighting parameter μ to be as small as 0.001. It is confirmed by experimental results, as compared to the fixed model, the adaptive scheme achieves more stable performance under various quantization steps.

3.2. Multiresolution motion search

Before motion search is applied to the full resolution video, the MAD of an MB is estimated using a multi-resolution approach as shown in Fig. 5 in the proposed algorithm. This approach can help predict the initial motion vector as a starting point for the search in the full resolution.

First, we represent the original image frame with two resolutions: the original one and the lower resolution one. The lower resolution image is obtained by averaging 4×4 pixels and performing the 4×4 to 1×1 down-sampling. Thus, each MB in the original frame becomes a block of size 4×4 in the low resolution frame. A diamond search is then performed to find the motion vector and MAD of each 4×4 block in the low resolution level. The motion vector of an MB in the original level can be predicted by an interpolation technique to be detailed in the next section.

The multiresolution motion search does not cost much computation, since the block-size and the search range are scaled down greatly in the lower resolution level. Besides, the search at the lower resolution level only has to be done once for each frame. The information of motion vectors and MAD can be stored for initial estimates of motion vectors in the original resolution level.

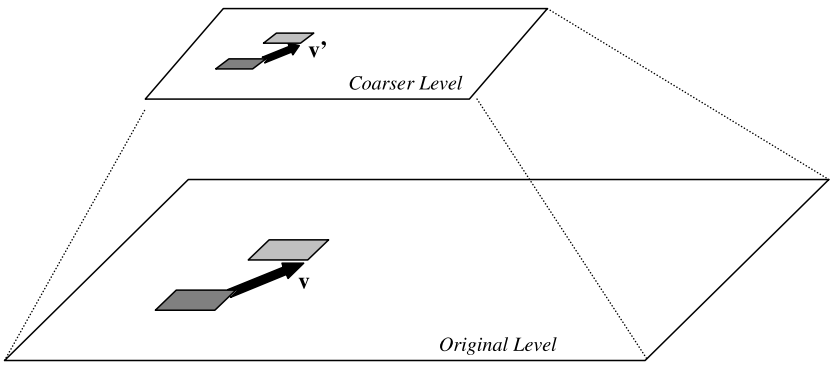

Fig. 5. Multi-resolution motion search.

3.3. Prediction of motion vectors

A quadratic model is used to predict the motion vector in the original resolution video from the low resolution video. The model was first proposed in [26] for the sub-pixel motion vector interpolation from integer-pixel motion vectors. We modified it for the prediction of motion vectors in the full resolution from those in the coarse resolution.

Let $\mathbf{v} = (p_x, p_y)$ be the motion vector of some MB in the original resolution video. To predict $\mathbf{v}$, we first model the SAD in the low resolution level, denoted as $S(x, y)$, as a second-order polynomial of the motion vector in the original resolution:

$$S(x,y) = k_1\left(x - \frac{p_x}{4}\right)^2 + k_2\left(y - \frac{p_y}{4}\right)^2 + k_3, \tag{6}$$

where k_1, k_2, and k_3 are characteristic parameters of the SAD function. Let $\mathbf{v}' = (v_x, v_y)$ be the motion vector of the corresponding 4×4 block in the coarse resolution with minimum SAD denoted as S_0. The minimum SADs of its four neighboring motion vectors $(v_x, v_y - 1)$, $(v_x + 1, v_y)$, $(v_x, v_y + 1)$, and $(v_x - 1,\ v_y)$ in the coarse level are denoted by S_1, S_2, S_3, and S_4, respectively. By substituting five known SADs, $S_0, \ldots, S_4$, into Eq. (6), the optimum predicted MV (p_x, p_y) can be solved by

$$p_x = \text{Round}\left[4 \times \left(v_x + \frac{S_4 - S_2}{2(S_2 + S_4 - 2S_0)}\right)\right],$$
$$p_y = \text{Round}\left[4 \times \left(v_y + \frac{S_1 - S_3}{2(S_1 + S_3 - 2S_0)}\right)\right].$$

For boundary MBs, where some of SADs may not be available, we can simply choose $(4v_x, 4v_y)$ as the prediction in the full-resolution video.

This model is different from that given in [26] since we predict motion vectors from the coarse level to the full-resolution level (rather than from the integral pixel level to the fractional pixel level). The accuracy of our model depends on the subsampling operation, while that of [26] depends on the interpolation operation. Thus, our model may not be as accurate as the original scheme in [26] since some details could be lost in the down-sampling process. Note that macroblocks with smoother contents can be better predicted more accurately than those with complicated textures.

The motion vector predicted by this model only serves as a rough initial candidate. It should be compared with other candidates obtained using different prediction methods. Only the winner will serve as the start point in the diamond search process. When early termination conditions are not met, the prediction error will be corrected by the refinement provided by the diamond search.

3.4. Prediction of residual variance from coarse level

In the proposed approach, we use the MAD information in the full resolution to estimate the number of bits required to encode an MB. In this subsection, we show

that the value $\mathrm{MAD}_{\mathrm{full}}$ of an MB in the full resolution level can also be predicted from $\mathrm{MAD}_{\mathrm{lev2}}$ in the coarse level by a simple relation

$$\mathrm{MAD}_{\mathrm{full}} = c \times \mathrm{MAD}_{\mathrm{lev2}}, \tag{7}$$

where c is a constant.

It is assumed that the distribution of predicted residual signals of an MB can be approximated by the Laplacian distribution with zero mean and a separable covariance

$$R(m,n) = \sigma_{\mathrm{f}}^2 r^{|m|} r^{|n|},$$

where m and n are the horizontal and vertical distances between two pixels, respectively, σ_{f}^2 is the variance of these pixel values, and r is the correlation coefficient. Under the assumption of Laplacian distribution, the variance of residual signals in the compensated blocks can be approximated by

$$\sigma_{\mathrm{f}} \approx \sqrt{2} \times \mathrm{MAD}_{\mathrm{full}}.$$

Similarly, the variance of residual signals in the coarse level can be approximated by $\sigma_{\mathrm{lev2}} \approx \sqrt{2} \times \mathrm{MAD}_{\mathrm{lev2}}$. We observe that each pixel in the coarse level is the average of all 4×4 pixels in the original level. Therefore, it is actually the DC value of the 4×4 DCT transform in the original level. Since we have modeled the image to have correlation coefficients r, the variance of the (u,v)th component of DCT coefficients in the transform domain can be derived as [27]

$$\sigma_{\mathrm{F}}^2(u,v) = \sigma_{\mathrm{f}}^2 [ARA^{\mathrm{T}}]_{u,u} [ARA^{\mathrm{T}}]_{v,v},$$

where R is the correlation matrix with coefficients r, A is 4×4 DCT transformation matrix, and the operation $[\cdot]_{u,v}$ means the (u,v)th component of a matrix. Hence $\sigma_{\mathrm{lev2}}^2 = \sigma_{\mathrm{F}}^2(1,1)$. Let the ratio of $\sigma_{\mathrm{F}}^2(u,v)/\sigma_{\mathrm{f}}^2$ be denoted by $W(u,v)$. By using the energy conservation property, we obtain

$$\sigma_{\mathrm{f}}^2 = \sigma_{\mathrm{lev2}}^2 \times \frac{\sum_{u,v} W(u,v)}{W(1,1)}.$$

Take the correlation factor $r = 0.6$ as an example, which was investigated in [27] to be a good approximation for most pictures. By applying the transformation and the quantizer of H.264 to the above equations, the variance of residual signals can be approximated by

$$\sigma_{\mathrm{f}} \approx \sqrt{2} \times \mathrm{MAD}_{\mathrm{lev2}} \times 1.689. \tag{8}$$

Therefore, the constant c in Eq. (7) can be set to 1.689.

In this model, the matched block in the reference frame at the coarse level is assumed to have the scaled version of the motion vector searched in the full-resolution level, and residual signals in the coarse level are approximately the same as the down-sampled versions of residual signals in the full-resolution image. In practice, this approximation may be not accurate, since many details are lost in the sub-sampling process. Thus, it is better to update constant c after the motion search of each macroblock. In addition, the correlation factor r may change from

frame to frame, and consequently, the value of c may vary with the picture. In other words, we can replace the constant c to be a function of the form $c(k)$, where k denotes the kth MB. The value of $c(k)$ is updated right after the encoding of an MB by

$$c(k) = \frac{A_f(k)}{A_2(k)},$$

where $A_f(k)$ and $A_2(k)$ are averaged MADs from the full and the coarse resolutions. They are estimated, respectively, by

$$A_f(k) = \beta A_f(k-1) + (1-\beta)\mathrm{MAD}_{full}(k),$$
$$A_2(k) = \beta A_2(k-1) + (1-\beta)\mathrm{MAD}_{lev2}(k),$$

where β is a forgetting parameter with a value between $[0,1]$.

This approximation can help make the initial mode decision for an MB, and it is used before any motion search in the original level. After mode selection, a fast motion search is applied to the original level with the selected mode to obtain a more reliable MAD_{new}, and the estimated variance can be updated to $\sqrt{2} \times \mathrm{MAD}_{new}$.

3.5. Search with fast mode detection

Since motion search in a low resolution frame is done before the actual encoding of any MB in the full-resolution frame, the estimated motion vector and MAD are both available for initial mode decision of each MB. Furthermore, the predicted bit number b_i of the ith MB can be estimated from Eq. (5).

From extensive experimental results, we find some empirical rules for mode selection in H.264. For example, mode 1 (of size 16×16) is the most popular one among all modes. Therefore, this should be the first one to search if the predicted MAD and estimated encoded bits b_i are both small.

Since large MAD will result in large b_i, we can simply use b_i for mode decision. If the estimated number of coding bits b_i is much greater than $\bar{b}$, which represents the average bits to code an MB, it is probable that the MB should be further decomposed into smaller partitions so that they can be searched in different directions to reduce the motion compensation residual and, therefore, the number of bits required to encode the residual. On the other hand, if b_i is small enough to meet a distortion criterion, there is no need to do further search. This early termination rule will reduce the number of modes to be searched with little performance degradation.

In the reference software of H.264, there is a fixed search pattern to follow for partitions of all modes. In the proposed algorithm, we start the search with the initial mode predicted by b_i. Fig. 6 illustrates the flowchart of the proposed mode decision process. The initial modes are chosen between two levels of square partitions, i.e., 16×16 and 8×8. For a smaller number of b_i, the mode with block 16×16 is used. if b_i is greater than the threshold θ_1, then an MB is split into four smaller blocks of size 8×8.

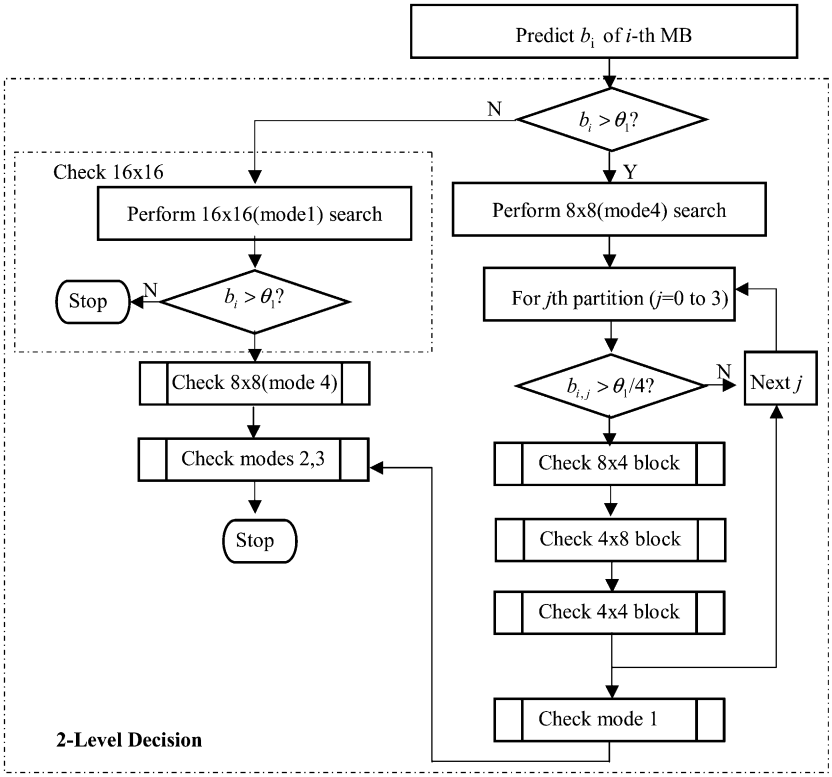

Fig. 6. The flowchart of 2-level mode decision.

After the initial mode is determined, motion search is performed with blocks under this new partition, and MAD and b_i are updated with more accurate values. If b_i meets a predefined threshold, then the search will stop immediately to save computation. Otherwise, the search should continue with other modes. In the case of using the 16×16 block as an initial mode, the next search mode is the 8×8 block. If the early termination criterion is still not met, we continue to check modes with rectangular shapes of 16×8 and 8×16 blocks. The modes with partitions smaller than 8×8 would not be checked since they are less probable to happen while the value of b_i is small.

On the other hand, if the initial mode is the 8×8 block and the threshold is not met, the predicted bit number $b_{i,j}$ is estimated for the jth 8×8 partition ($j = 0,1,2,3$) with the same method to predict residual bits of a macroblock. The jth partition will be split only if $b_{i,j} \geqslant \theta_1/4$. For split 8×8 partitions, their 4×4 sub-blocks will be checked first. If the early termination condition is still not satisfied, other modes such as 4×8 and 8×4 will be checked consequently.

For motion search under a particular partition, the MVFAST algorithm is adopted with modifications in several aspects. First, the fixed threshold is replaced by

adaptive thresholds derived from MAD in the low resolution image. Second, the initial motion vectors to be checked are also changed. The predicted vectors in MVFAST are taken from the neighboring MB's motion vectors. In the proposed algorithm, we predict the motion vector using a merge-and-split process. The merge process means that the predicted vector of a partition are taken from its square sub-partitions. The split process means that the predicted vector of a sub-partition are taken from its parent square partition. These motion vectors are first checked, and the one with the minimum SAD is examined furthermore. If the resulting bit length does not exceed the specified threshold, the search stops immediately. Otherwise, we will check the motion vectors in neighboring MB's as specified by MVFAST. The same threshold of early termination applies to these candidates, too. If the threshold is not met, the winner among candidates is served as the new search center and a local search of the diamond pattern is performed. In our implementation, only the small diamond pattern is used since it works well for almost all cases tried.

The above 2-level decision process can be further enhanced by including one more decision level. It is observed in many cases that, when the initially predicted bit number is large, it is better to go directly to search for all 4×4 blocks under the search mode for 8×8 partitions. The flowchart of the 3-level decision process is depicted in Fig. 7.

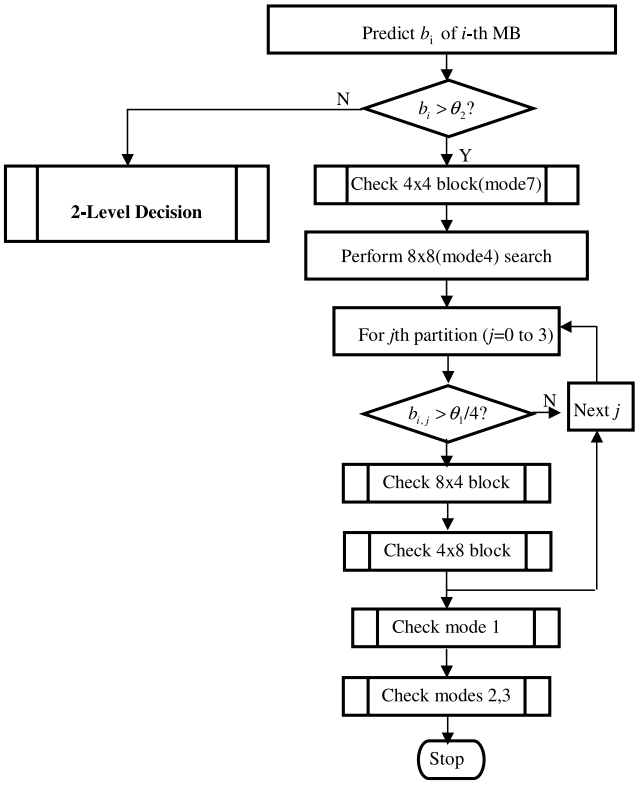

Fig. 7. The flowchart of the 3-level mode decision.

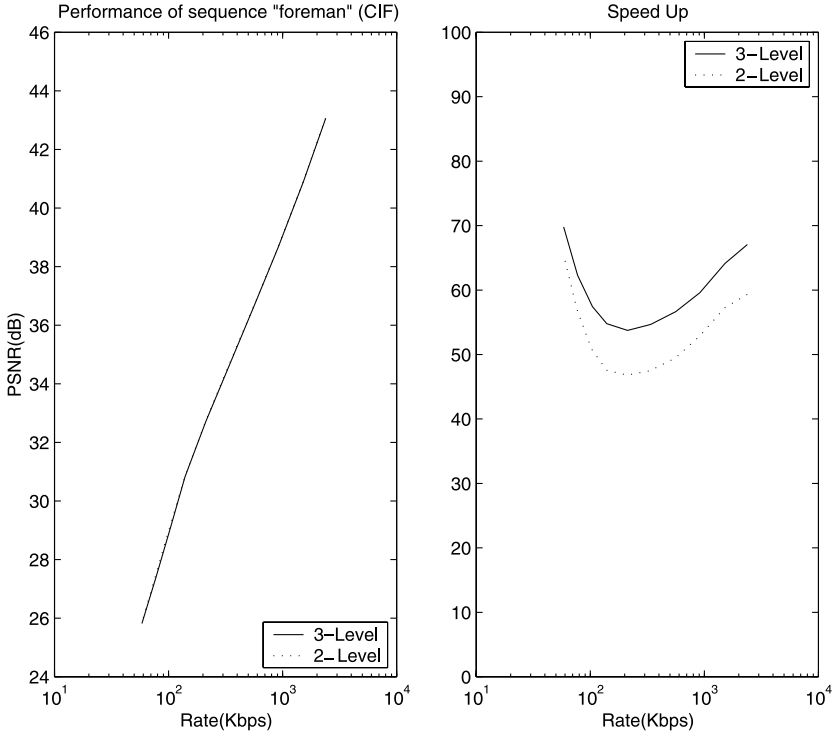

Fig. 8. The performance comparison of the 2-level and the 3-level mode decision.

Fig. 8 shows a comparison for the two proposed schemes by encoding test CIF sequence 'Foreman.' We see that the 3-level decision scheme is always faster than the 2-level one. The 3-level algorithm is 13% faster than the 2-level one. However, the coding rate may increase a little bit since the mode of 8×8 partitions may be omitted. When compared to the 2-level algorithm, the bit rate of the 3-level algorithm increase by 0.8% while the PSNR quality remains about the same in this example. One may make a tradeoff between the coding bit rate and the speed-up factor by selecting the 2-level or the 3-level decision process adaptively. In the following simulation results, the 3-level scheme is adopted since it is much faster while the increased bit rate is insignificant.

3.6. Thresholds setting

Here, we provide more details about the principle in setting the thresholds. Suppose that the variance σ_f of the motion compensated residual in one MB has been predicted from motion vectors in the coarse resolution by Eq. (8). Then, we can determine the averaged MAD $\bar{\sigma}_f$ of frame residuals over all MB's σ_f. The bit number to represent a single motion vector Γ_{mv} can also be estimated by averaging the bits

representing the motion vectors of all MBs in the previous frame. From Eq. (5), the average bit number to encode an MB's residual signal is about $\bar{b} = 256B(Q, \bar{\sigma}_f)$.

Let $\sigma_{f,i}$ be the MAD for the ith MB and, consequently, the bit number b_i can be approximated by $b_i = 256B(Q,\sigma_{f,i})$. The values of l_{mv} and b_i can be viewed as initial estimations of $E_M(MB_i; P, V)$ and $E_R(MB_i; P, V)$ in Eq. (1), respectively. In other words, the total number of bits to represent an MB can be approximately by $b_i + l_{mv}$, where the header bits are excluded. It is expected that b_i should be close to the average number $\bar{b}$. If b_i is much larger than the average bit number, then it is likely that a further partition into smaller blocks may help reduce the number of coding bits, and the minimum of Eq. (1) can be approached.

In the threshold selection for the proposed mode decision process, one may trade the computational complexity for coding efficiency. If the threshold is set in favor of coarser modes, it is less likely for a macroblock to be split into partitions to search for individual motion vectors, thus requiring less SAD operations and resulting in a lower computational complexity. On the other hand, this threshold reduces the probability of finding the correct mode in the search process, thus leading to larger residual signals.

Fig. 9 gives an example of the trade-off by encoding the CIF 'Foreman' sequence with reference software JM 4.0d. The horizontal axis indicates the ratio $q = (\theta - \bar{b})/\bar{l}_{mv}$, where θ is the threshold for the predicted bit number b_i to determine

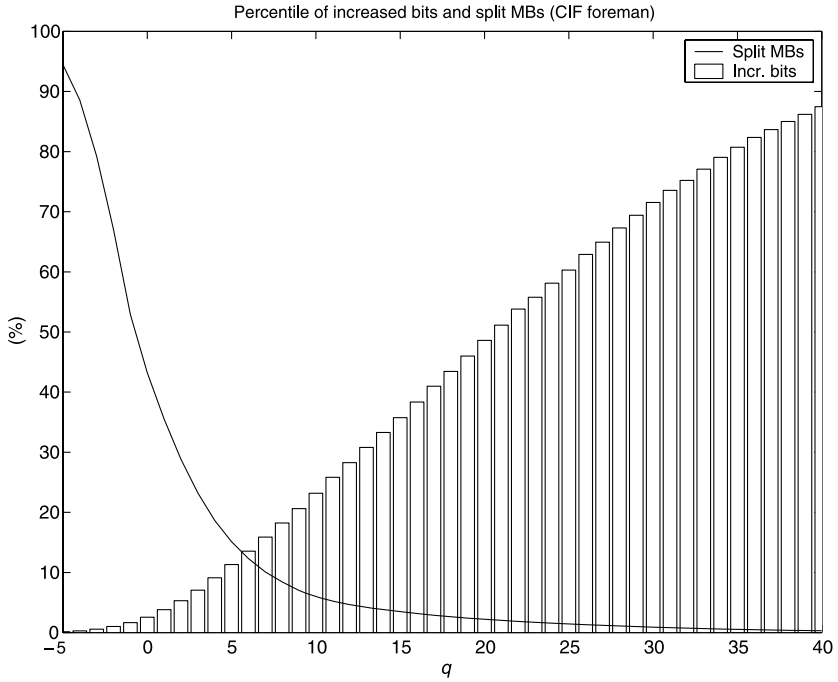

Fig. 9. Percentile of increased bits and split MBs for test sequence "Foreman" (CIF).

whether the ith MB should be split or not. The solid curve represents the percentage of macroblocks to be split and the split scheme needs more SAD operations than the non-split one. The bars represent the percentages of increased bits with respect to the one using the exhaustive search algorithm to encode a macroblock, when some threshold θ is applied for early termination. For instance, if $q = 3$, the bit number is increased by 8%, while only 20% of the macroblocks are split. In this proposed algorithm, we choose $q = 3$ to set the first threshold to decide whether a macroblock should be split or not.

Alternatively, we can heuristically set thresholds in the following way. For an MB with four 8×8 blocks, the expected number of bits to encode the MB is roughly $\bar{b} + 4\bar{l}_{mv}$. Therefore, if $b_i > \bar{b} + 3\bar{l}_{mv}$, the mode with four 8×8 blocks in one MB is more likely to happen and should be checked before the 16x16 MB. Therefore, we set the first threshold to

$$\theta_1 = \bar{b} + 3\bar{l}_{mv}.$$

In the case where b_i is even larger, we set a more conservative criterion. That is, if $b_i > \bar{b} + 7\bar{l}_{mv}$, we go directly to the partition with the smallest block size 4×4. This gives the second threshold

$$\theta_2 = \bar{b} + 7\bar{l}_{mv}.$$

After motion search in the full-resolution frame, the predicted bit number b_i is replaced with the number estimated by the resulting MAD found in the specified mode. It is compared with thresholds θ_1 and θ_2 for possible early termination. That is, the search stops immediately if b_i is smaller than these threshold values for the 8×8 and 4×4 cases, respectively. Otherwise, the motion search process will continue.

4. Experimental results

We implemented the proposed algorithm and integrated it with the H.264 reference software JM4.0d. The exhaustive search (fast FS) and MVFAST search algorithms were tested for comparison. The exhaustive search was executed by enabling the option of *FAST_FULL_SEARCH* in reference software, which computed SADs of 4×4 blocks first and merged SADs of larger blocks with a bottom-up merging process as described in Section 2.4. The MVFAST algorithm was implemented by following the description in [10]. Only the P-frame prediction with one reference frame was tested as inter-frame prediction. The search range was set to ± 16 for all cases.

Several video sequences of different formats were tested in our experiment, and 10 quantization parameters ranging from 18 to 45 were selected for each sequence to cover a wide range of bit rates. Table 1 gives the results of the average performance for these sequences.

We used the fast FS algorithm provided in the H.264 reference software as a benchmark. Two fast search algorithms, including the proposed algorithm and the

Table 1
Comparison of the proposed algorithm and other algorithms for H.264 encoding

Sequence	Format	BPS increase(%)		Speed up	
		New method	Diamond	New method	Diamond
Susie	D1	6.02	9.43	71.60	23.96
Football[a]	D1	5.20	2.90	47.01	14.71
Mobile	D1	2.21	4.39	64.28	14.89
Mobile	CIF	2.68	4.85	69.49	17.55
Foreman	CIF	3.73	4.93	60.01	17.98
Foreman	QCIF	4.77	4.47	65.88	20.71
Akiyo	CIF	−0.09	4.01	156.82	50.32
Akiyo	QCIF	0.04	3.91	147.85	49.63
Salesman	CIF	1.35	2.92	92.15	29.63
Salesman	QCIF	0.80	3.01	110.51	36.94
Missam	CIF	0.68	6.59	104.24	31.32
Missam	QCIF	0.08	3.78	136.62	44.66
Coastguard	CIF	2.66	6.11	66.65	16.98
Coastguard	QCIF	3.46	6.21	79.15	23.44
Hall	CIF	0.88	1.46	121.31	33.69
Hall	QCIF	0.75	2.14	145.97	40.18
Flower	CIF	2.14	2.19	70.67	20.79

[a] Interlaced coding mode is enabled.

fast search algorithm with MVFAST directly applied to all seven modes, were compared to the benchmark algorithm. In Table 1, the columns labeled by "BPS Increase" are the bit-rate increase in terms of the percentage. As expected, the fast FS algorithm generates the lowest bit-rates in most cases. The results also show that the proposed method has higher coding efficiency than the MVFAST algorithm in general. This means that our algorithm can determine the proper mode and the associated motion vector better than MVFAST. It is worth mentioning that the proposed algorithm even outperforms fast FS in coding bit rates for the "Akiyo" CIF sequence. The reason is that, although FS can find the MV candidate with the minimum SAD, the minimum SAD value does not always guarantee the smallest number of coding bits, as stated in Section 2.3.

The right two columns of the table show the speed-up factors. The factors are calculated by counting the number of SAD operations. In the proposed algorithm, the counting includes the sub-sampling operation and SAD operations of motion search in both the original and the coarse levels. The values given in the table are ratios of the SAD operation compared to the benchmark, i.e., the fast FS algorithm. It is clear that the proposed algorithm can provide a speed-up factor in the range of 60–150. Note that the speed-up factor may vary for different bit rates even for the same test video. Compared to MVFAST, our method can still improve the search speed by 3–4.3 times while providing better coding efficiency in most test sequences.

In general, the proposed algorithm has better performance in sequences with smoother motion since fewer motion vector candidates need to checked.

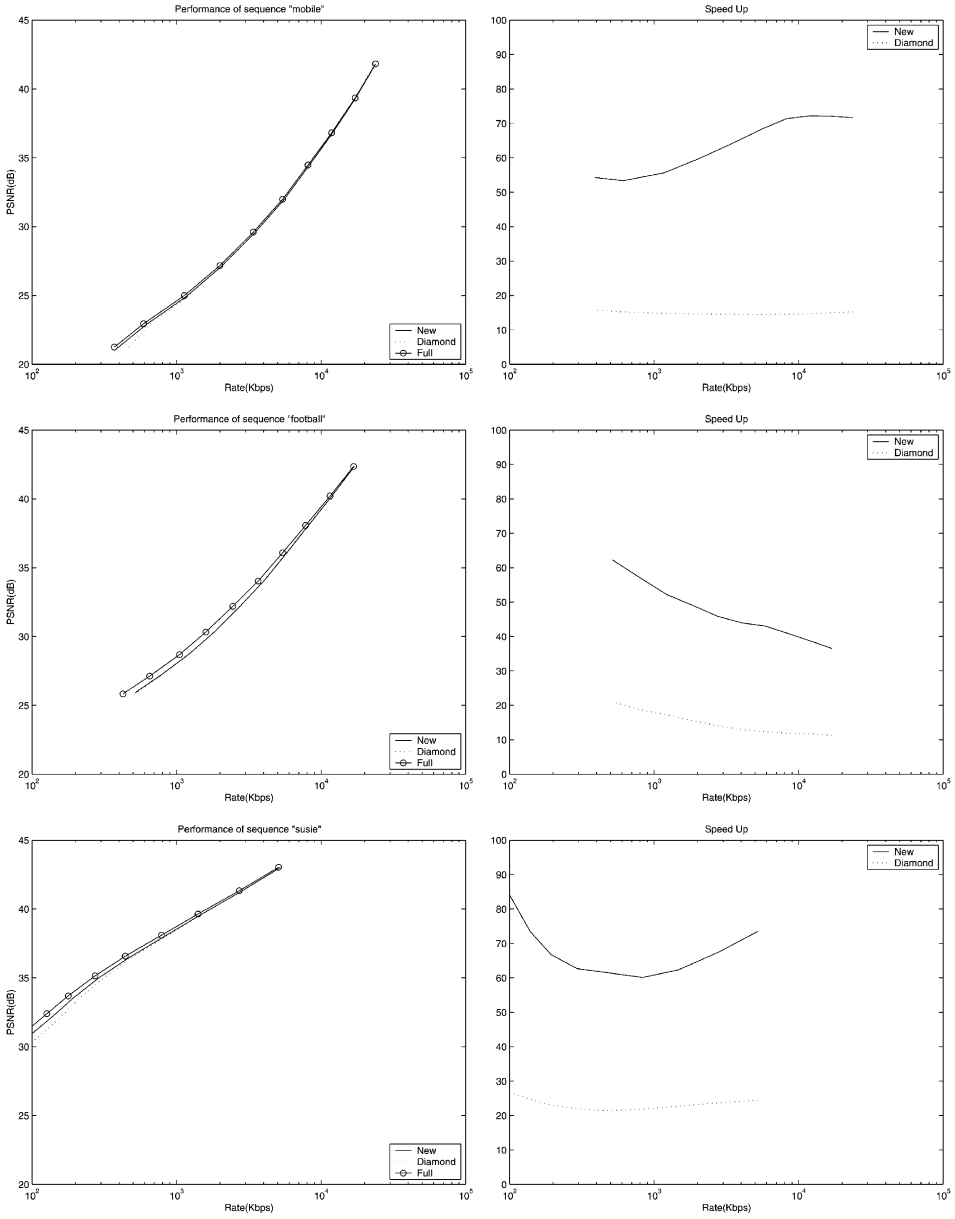

Fig. 10. The performance of test sequences with a frame size of 720×480.

Other video characteristics may also influence the performance of the proposed algorithm. For example, the test sequence 'football' has a lot of movement and obviously interlaced effects. If we use the frame coding mode as adopted in other test

sequences, the coding performance of the proposed algorithm will degrade severely. The resultant bit rate is 10% higher than the full search algorithm. In this case, the interlaced coding option has to be turned on, and the performance becomes more reasonable as shown in the table.

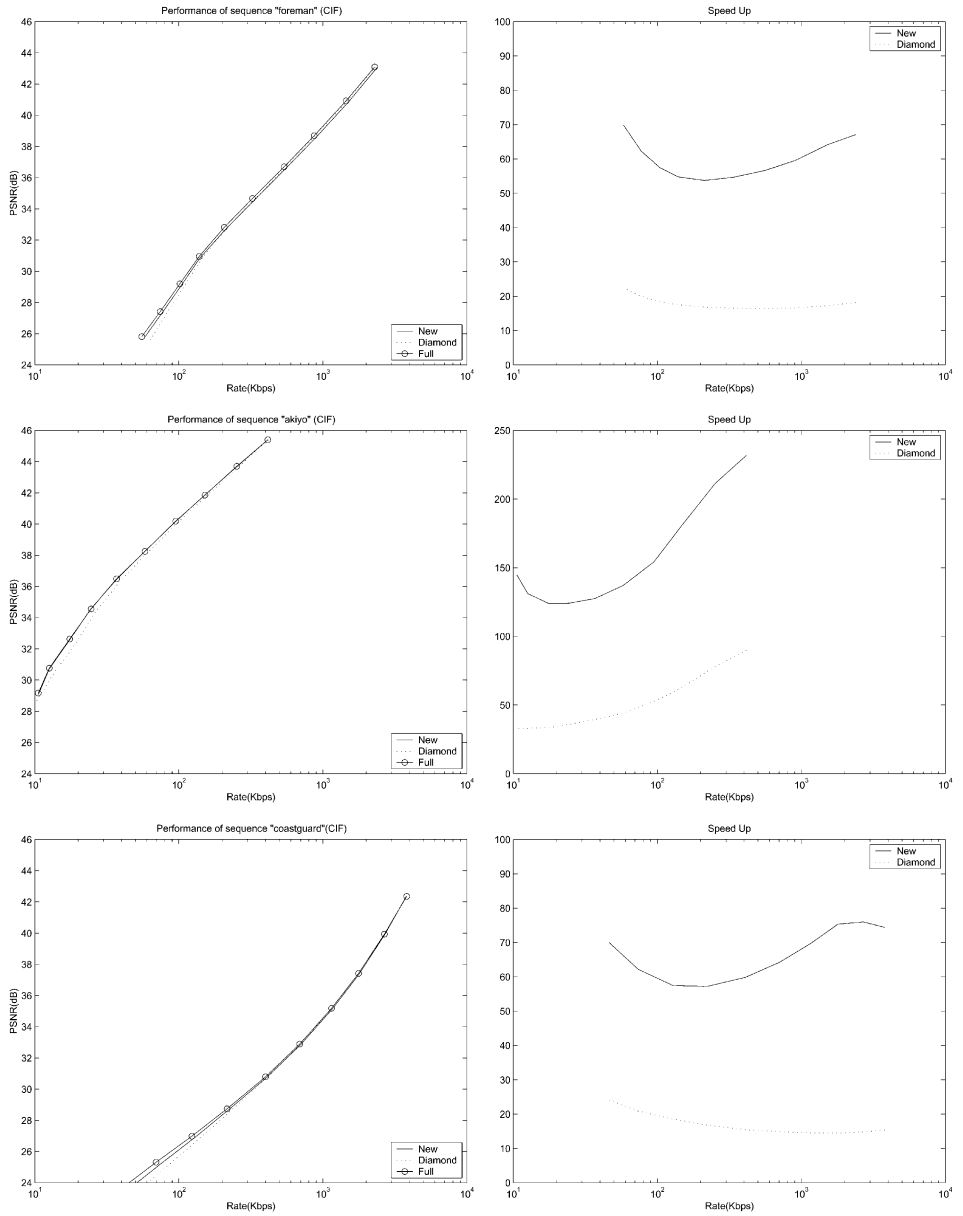

Fig. 11. The performance of test sequences of the CIF size.

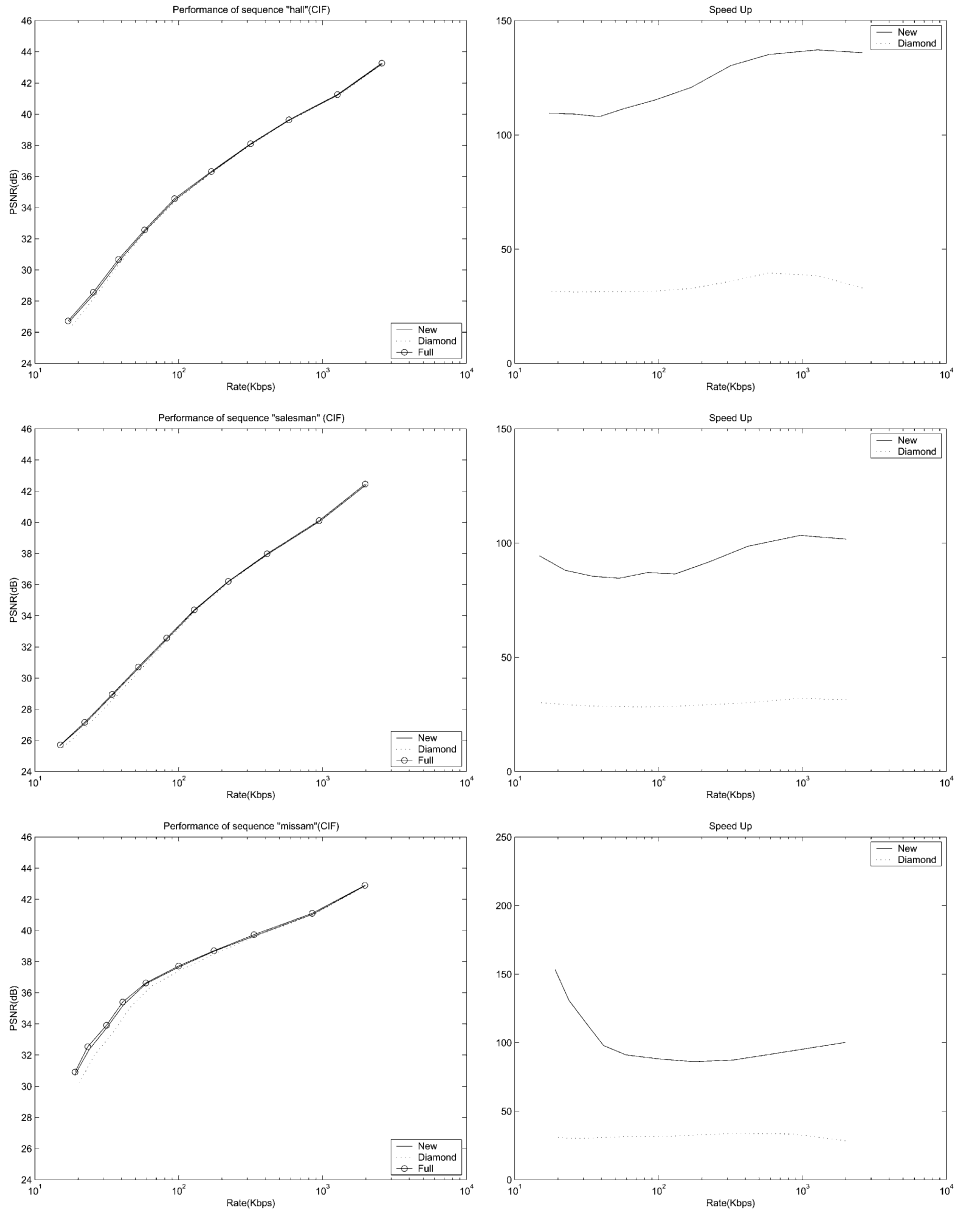

Fig. 11. (*continued*)

The comparison of coding performance among three codecs are shown in Figs. 10–12. The PSNR and speed-up factors vs bit rates are plotted at various quantization steps with format D1, CIF, and QCIF, respectively. The proposed algorithm, MVFAST, and FS are marked as 'new,' 'diam,' and 'full' in the legend. In most cases,

the PSNR curves of the proposed algorithm are close to those of the FS algorithm. This means that the proposed fast algorithm has little visual quality degradation as compared with the FS algorithm. Note that the visual quality using MVFAST degrades significantly in the low bit rates.

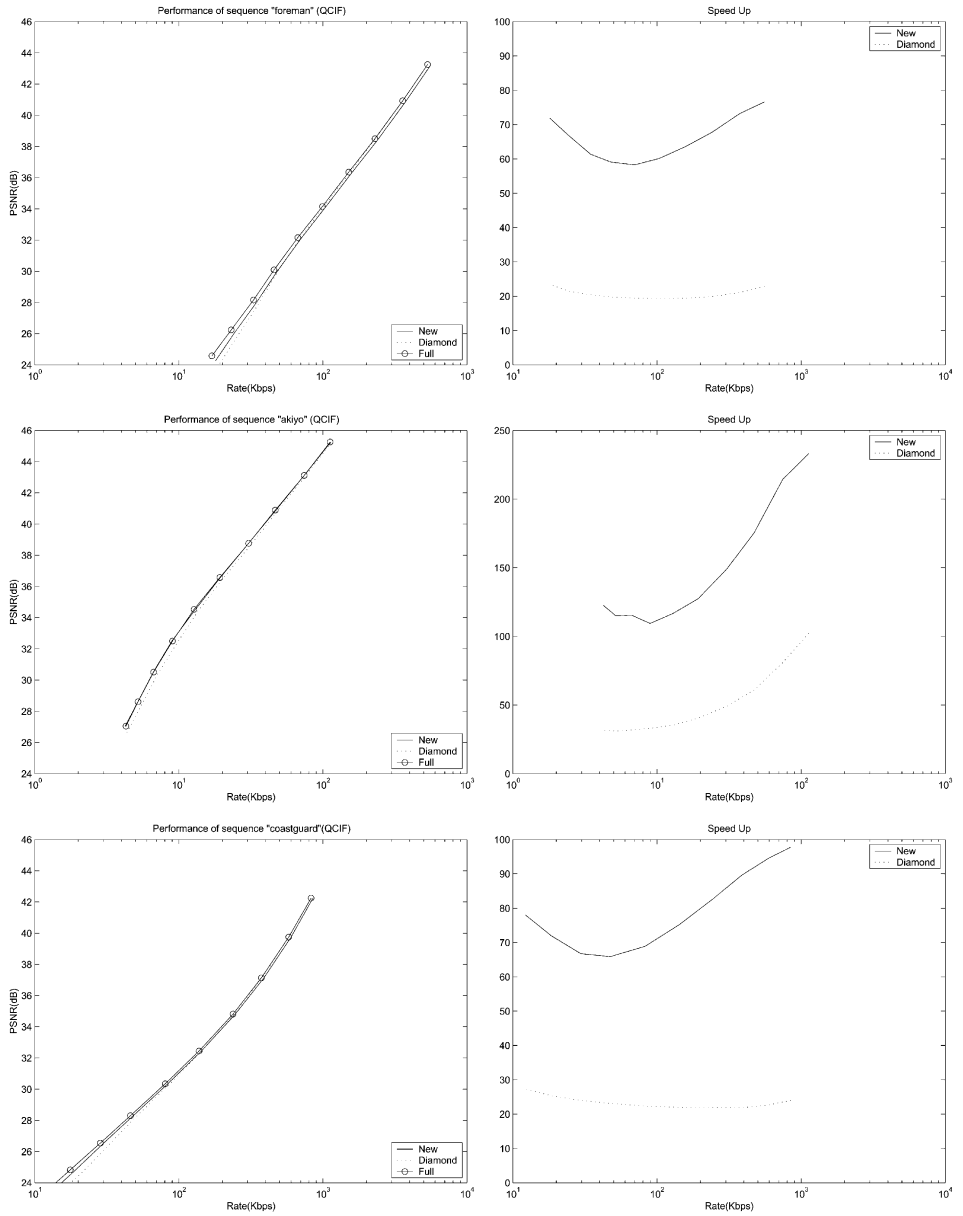

Fig. 12. The performance of test sequences of the QCIF size.

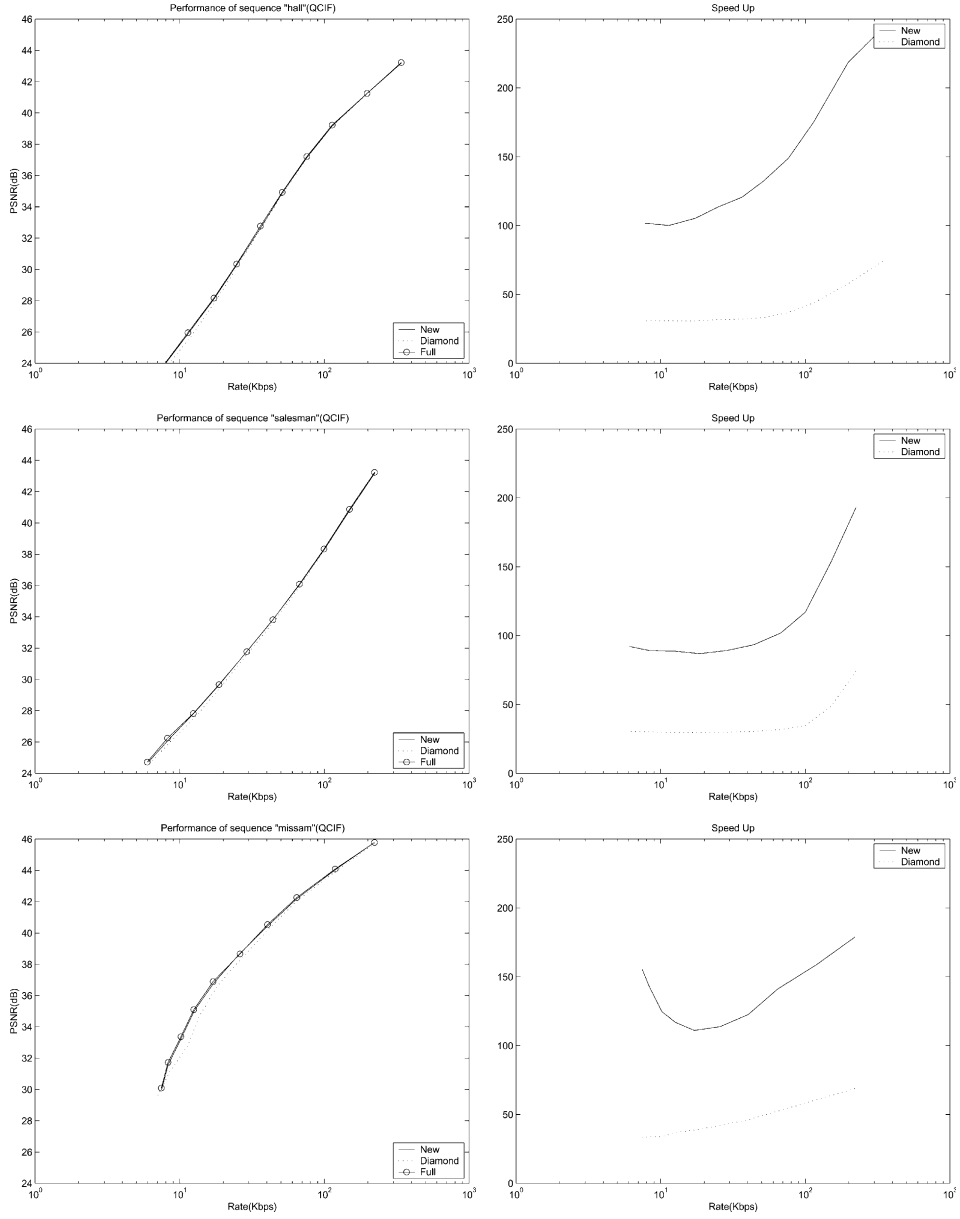

Fig. 12. (*continued*)

5. Conclusion

We have developed a fast motion search algorithm for H.264 by introducing a rate model for effective initial mode selection, a method for initial motion vector

prediction and early termination rules to avoid unnecessary computation. Experimental results were given to demonstrate that our method can speed up the search up to a factor of 150 times with little visual quality degradation. The proposed method outperforms several fast search algorithms and provides the best tradeoff between coding efficiency and the speed. With our approach, a real-time H.264 encoder for high quality video becomes more feasible.

References

[1] Study of Final Committee Draft of Joint Video Specification, Joint Video Team (JVT) of ITU-T Rec. H.264 |ISO/IEC 14496-10 AVC, February 16, 2003.
[2] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion compensated interframe coding for video conferencing, in: IEEE Proceedings of the National Telecommunication Conference, December 1981, vol. 4. pp. G5.3.1–G5.3.5.
[3] Renxiang Li, Bing Zeng, Ming L. Liou, A new three-step search algorithm for block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 4 (4) (1994) 438–442.
[4] Lai-Man Po, Wing-Chung Ma, A novel four-step search algorithm for fast block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 6 (3) (1996) 313–317.
[5] Lurng-Kuo Liu, Ephraim Feig, A block-based gradient descent search algorithm for block motion estimation in video coding, IEEE Trans. Circuits Syst. Video Technol. 6 (4) (1996) 419–422.
[6] Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath, Ashraf Ali Kassim, A novel unrestricted center-biased diamond search algorithm for block motion estimation, IEEE Trans. Circuits Syst. Video Technol. 8 (4) (1998) 369–377.
[7] Junavit Chalidabhongse, C.-C. Jay Kuo, Fast motion vector estimation using multiresolution-spatio-temporal correlations, IEEE Trans. Circuits Syst. Video Technol. 7 (3) (1997) 477–488.
[8] A.M. Tourapis, O.C. Au, M.L. Liou, Predictive motion vector field adaptive search technique (PMVFAST)-enhancing block based motion estimation, in: SPIE Proceedings of the Visual Commun. Image Proc., January 2001.
[9] Shan Zhu, Kai-Kuang Ma, A new diamond search algorithm for fast block-matching motion estimation, IEEE Trans. Image Process. 9 (2) (2000) 287–290.
[10] Optimization Model Version 3.0, ISO/IEC JTC1/SC29/WG11 N4344, Sydney, July 2001.
[11] A.C. Yu, Efficient block-size selection algorithm for inter-frame coding in H.264/MPEG-4 AVC, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 2004, vol. 3, May 2004, pp. 169–172.
[12] D. Wu, S. Wu, K.P. Lim, F. Pan, Z.G. Li, X. Lin, Block inter mode decision for fast encoding of H.264, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 2004, May 2004, vol. 3, pp. 181–184.
[13] X. Jing, L.-P. Chau, Fast approach for H.264 inter mode decision, IEEE Electron. Lett. 40 (17) (2004) 1050–1052.
[14] A. Ahmad, N. Khan, S. Masud, M.A. Maud, Efficient block size selection in h.264 video coding standard, IEEE Electron. Lett. 40 (1) (2004) 19–21.
[15] Woong Il Choi, Byeungwoo Jeon, Jechang Jeong, Fast mode estimation with modified diamond search variable motion block sizes, in: Proceeings of the IEEE International Conference on Image Processing 2003, September 2003, vol. 3, pp. 371–374.
[16] Andy Chang, P.H.W. Wong, Y.M. Yeung, O.C. Au, Fast multi-block selection for H.264 video coding, in: Proceedings of the IEEE International Symposium on Circuits and Systems 2004, May 2004, vol. 3, pp. 817–820.
[17] Zhi Zhou, Ming-Ting Sun, Yuh-Feng Hsu, Fast variable block-size motion estimation algorithms based on merge and split procedures for H.264/MPEG-4 AVC, in: Proceeings of the IEEE International Symposium on Circuits and Systems 2004, May 2004, vol. 3, pp. 725–728.

[18] Jianfeng Xu, Zhibo Chen, Yun He, Efficient fast ME prediction and early-termination strategy based on H.264 statistical characters, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 2004, May 2004, vol. 3, pp. 181–184.
[19] Hye-Yeon Cheong Tourapis, Alexis Michael Tourapis, Fast motion estimation within the H.264 CODEC, in: Proceedings of the IEEE International Conference on Multimedia and Expo, July 2003, vol. 3, pp. 517–520.
[20] Peng Yin, Hye-Yeon Cheong Tourapis, Alexis Michael Tourapis, Jill Boyce, Fast mode decision and motion estimation for JVT/H.264, in: Proceedings of the IEEE International Conference on Image Processing 2003, September 2003, vol. 2, pp. 853–856.
[21] A.M. Tourapis, Enhanced predictive zonal search for single and multiple frame motion estimation, in: SPIE Proceedings of the Visual Comm. Image Proc., 2002.
[22] G.J. Sullivan, Thomas Wegand, Rate-distortion optimization for video compression, IEEE Signal Process. Magazine 15 (6) (1998) 74–90.
[23] Hsueh-Ming Hang, Jiann-Jone Chen, Source model for transform video coder and its application—part 1: fundamental theory, IEEE Trans. Circuits Syst. Video Technol. 7 (2) (1997) 287–298.
[24] E.Y. Lam, J.W. Goodman, A mathematical analysis of the DCT coefficient distributions for images, IEEE Trans. Image Process. 9 (10) (2000) 1661–1666.
[25] Hung-Ju Lee, Tihao Chiang, Ya-Qin Zhang, Scalable rate control for MPEG-4 video, IEEE Trans. Circuits Syst. Video Technol. 10 (6) (2000) 878–894.
[26] Xiaoming Li, Cesar Gonzales, A locally quadratic model of the motion estimation error criterion function and its application to subpixel interpolations, IEEE Trans. Circuits Syst. Video Technol. 6 (1) (1996) 118–122.
[27] I-Ming Pao, Ming-Ting Sun, Modeling DCT coefficients for fast video encoding, IEEE Trans. Circuits Syst. Video Technol. 9 (4) (1999) 608–616.