Front. Electr. Electron. Eng. China (2006) 4: 431–436
DOI 10.1007/s11460-006-0082-4

RESEARCH ARTICLE

LIN Wei-yao, Fang Xiang-zhong, HUANG Xiu-chao, LI Dian, LIU Xiao-feng

A fast block mode selection approach for H.264 visual coding

Abstract In this paper, a new fast mode-selection approach is proposed. This algorithm combines the proposed approaches of mode pre-decision and precise large-small mode decision, by selecting the best mode efficiently. Experimental results show that the proposed approach can reduce the computational cost of full search and fast multi-block motion estimation by 80 % and 45 %, respectively, with similar visual quality and bit rate. The proposed algorithm also reduces by 75 % the computational cost of the large-small mode isolation algorithm for low-motion sequence coding, and with 0.06 PSNR gain and 3.7 % reduction in bit rate.

Keywords video coding, H.264 inter mode, optimization algorithm

1 Introduction

H.264/AVC is a new video coding standard established by the ITU-T and ISO/IEC. Compared with the old video coding standards, H.264 uses many new techniques, which greatly increase its video coding efficiency. According to Ref. [1], H.264 can save 50 % of the bit rate while keeping the same video coding quality as other standards.

H.264 uses 7 modes for inter-frame prediction, i.e., 16 × 16, 16 × 8, 8 × 16, 8 × 8, 8 × 4, 4 × 8 and 4 × 4. The usage of multiple inter-modes greatly improves the coding efficiency. However, the computation cost will be very high if the best mode is obtained by fully searching all 7 modes. According to Ref. [2], the motion estimation for all 7 modes accounts for about 80 % of the total encoding time. The encoding complexity of H.264 is approximately 4–5 times larger than H.263. Therefore, the development of a fast inter-mode selection algorithm is a very important research topic for H.264 video coding.

Nowadays, there have been many algorithms proposed for H.264 inter-mode selection. Reference [3] analyzes the object motion and texture feature, proposing a fast multi-block motion estimation algorithm (FMBME), which first searches the 8 × 8 mode and then selects the 16 × 16 mode according to the consistency of the four 8 × 8 motion vectors (MVs). FMBME can save 58 % computation time by quickly selecting the most frequent mode 16 × 16 while keeping the bit rate and visual quality almost the same. The method of Ref. [4] first tests the MV relationship between neighboring macroblocks (MBs), and then separates MBs into large mode and small mode. The method in Ref. [4], when combined with some fast motion estimation algorithms, can reduce computation by 77.96 % compared with the 7-mode full search. However, both algorithms above have some limitations. Reference [3] only considers the fast algorithm for 16 × 16 mode with no algorithm for the other modes. Therefore, for large motion sequences, which may have fewer 16 × 16 modes, the computation savings will be reduced. Although the method of Ref. [4] is almost 4 times faster than the 7-mode full search, there are also some decreases in PSNR and bit rate performance.

In this paper, we propose a new fast inter-mode selection algorithm, which quickly selects the best mode on the basis of mode pre-decision, spatial-temporal relationship of MBs and the consistency of 8 × 8 MVs. The simulation results show that our method can greatly reduce computation complexity while keeping PSNR and bit rate performance almost the same. The remainder of the paper is organized as follows: the following section presents our approach in detail. Section 2 describes the algorithm procedure. Some discussion and experiments are given in Sect. 3 and Sect. 4. We conclude this paper in Sect. 5.

2 Mode pre-decision and all-zero DCT coefficients detection

In Ref. [3], a fast 16 × 16 mode selection algorithm

Translated from *Journal of Shanghai Jiao Tong University,* 2006, 40(1): 1–6 (in Chinese)

LIN Wei-yao (✉), FANG Xiang-zhong, HUANG Xiu-chao, LI Dian, LIU Xiao-feng
Institute of Image Communication and Information Processing, Shanghai Jiao Tong University, Shanghai 200030, China
E-mail: hellomike@sohu.com

(FMBME) was proposed, which can greatly reduce the computation cost by quickly selecting the 16 × 16 mode. Since the 16 × 16 mode usually accounts for the largest portion in video sequences, the FMBME algorithm can reduce the computational cost by 58 % on average. However, the FMBME algorithm needs to search the 8 × 8 mode first and the 16 × 16 mode is then selected according to the correlation of the four MVs of 8 × 8 blocks. The additional 8 × 8 mode search will still waste computational time (we need to search the 8 × 8 mode four times for four 8 × 8 blocks before the 16 × 16 mode-decision). We consider whether the 16 × 16 mode can be searched and selected directly. Therefore, we propose the mode pre-decision algorithm, which combines the neighboring macroblock information and all-zero DCT coefficients detection.

2.1 All-zero DCT coefficients detection and modepre-decision

2.1.1 All-zero DCT coefficients detection

The all-zero DCT coefficient detection detects the DCT blocks, whose coefficients are all zero, without DCT transform and quantization. Reference [5] proposes that in most cases, the DC coefficient is the largest among all DCT coefficients. Therefore, when $F(0,0)< 2Q$, i.e.,

$$\left|\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\right| < 16Q \tag{1}$$

All coefficients will become zero after quantization, where Q is the quantization parameter, $f(x, y)$ is the coefficients of a block.

Reference [6] gives another all-zero detection algorithm based on 8 × 8 DCT transform. The two-dimension DCT transform is:

$$\begin{aligned} F(u,v) = &(\frac{2}{N})k_u k_v \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y) \\ &\cdot\cos(\frac{(2x+1)u\pi}{2N})\cos(\frac{(2y+1)v\pi}{2N}) \end{aligned} \tag{2}$$

where N is the size of the block, $F(u,v)$ is the DCT coefficients of a block, when $u = v = 0$, $k_u = k_v = 1/\sqrt{2}$; when u， $v \neq 0$, $k_u = k_v = 1$.

According to Eq. (2), when $N = 8$, we have

$$\begin{aligned} F(u,v) &\leqslant \frac{1}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}\left|f(x,y)\right|\cos\left(\frac{\pi}{16}\right)\cos\left(\frac{\pi}{16}\right) \\ &< \frac{1}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}\left|f(x,y)\right| \end{aligned} \tag{3}$$

From Eq. (3) and $F(u,v) < 2Q$, the all-zero DCT coefficient algorithm is defined as:

$$\sum_{x=0}^{7}\sum_{y=0}^{7}\left|f(x,y)\right| < 8Q$$

The algorithm in Ref. [6] is conservative, which will miss many all-zero blocks (separating all-zero blocks as non-all-zero). According to this, Ref. [7] makes an improvement and gives the detection-equation as follows:

$$\max\left(\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f_p(x,y), \left|\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f_q(x,y)\right|\right) < \left\lfloor\frac{2Q}{\alpha}\right\rfloor \tag{4}$$

where，

$f_p(x,y) \in \{f(x,y) > 0,\ f_p(x,y) = f(x,y);\ f(x,y) < 0,\ f_p(x,y) = 0\}$,

$f_q(x,y) \in \{f(x,y) < 0,\ f_q(x,y) = f(x,y);\ f(x,y) > 0,\ f_q(x,y) = 0\}$,

$\alpha = \frac{2}{N}\cos^2(\frac{\pi}{2N})$

The algorithm in Ref. [7] greatly reduces the probability of missing all-zero blocks with no increase in the probability of wrong-decision (recognizing a non-all-zero block as all-zero). The above algorithm is based on 8 × 8 DCT transform. Reference [8] gives another algorithm based on 4 × 4 DCT.

2.1.2 Mode pre-decision

Based on the all-zero DCT coefficient detection introduced above, we propose a novel mode pre-decision algorithm.

In H.264 video coding standard, the R-D cost used for mode decision is as follows:

$$J(\boldsymbol{m},\lambda) = \mathrm{SAD}(s,c(\boldsymbol{m})) + \lambda R(\boldsymbol{m}-\boldsymbol{p}) \tag{5}$$

where $\boldsymbol{m} = (m_x, m_y)^{\mathrm{T}}$ is the motion vector (MV), $\boldsymbol{p} = (p_x, p_y)^{\mathrm{T}}$ is the prediction MV, $R(\boldsymbol{m}-\boldsymbol{p})$ represents the bit cost used to encode motion information and λ is the Lagrange coefficient. The sum of absolute differences (SAD) is computed as:

$$\mathrm{SAD}(s,c(\boldsymbol{m})) = \sum_{x=1,y=1}^{B,B}\left|s[x,y]-c[x-m_x,y-m_y]\right|,$$

where B is the size of the block, s represents the original video signal and c is the video signal after coding.

However, if we consider the all-zero condition after DCT and quantization, the R-D cost in Eq. (5) is not precise. When SAD is smaller than the all-zero detection threshold, the motion compensation residue will become all-zero after DCT and quantization, no matter how large the SAD value is. In this case, the first item in Eq. (5) should be zero. Therefore, the R-D cost can be revised as:

$$J(\boldsymbol{m},\lambda) = \begin{cases} \mathrm{SAD}(s,c(\boldsymbol{m})) + \lambda R(\boldsymbol{m}-\boldsymbol{p}) & \\ \qquad\qquad\qquad\qquad \mathrm{SAD}(s,c(\boldsymbol{m})) \leqslant T \\ \lambda R(\boldsymbol{m}-\boldsymbol{p}) \qquad \mathrm{SAD}(s,c(\boldsymbol{m})) \leqslant T \end{cases} \tag{6}$$

where T is the all-zero detection threshold. According to Eq. (6), we can see that when SAD is smaller than T, the R-D cost will be decided by the second item. Since the 16 × 16 mode only needs to encode the motion information of one block while the other modes require encoding the motion information of no less than two blocks, the second item of 16 × 16 mode is usually the smallest. Therefore, we can reach the conclusion that the 16 × 16 mode will be the best mode when its SAD is smaller than the all-zero detection

threshold. According to the above conclusion, we can carry out the motion search for 16 × 16 mode first and select 16 × 16 as the best mode directly when its SAD is smaller than the all-zero detection threshold.

Since H.264 uses 4 × 4 block DCT, it is very difficult to determine a precise all-zero detection threshold T for 16 × 16 MB, with all 4 × 4 sub-blocks within the 16 × 16 MB being zero after DCT transform when SAD < T. Furthermore, due to different features for each MB, not all MBs are suitable for 16 × 16 mode pre-decision. Therefore, it is not reasonable to search 16 × 16 mode first for each MB. In our proposed algorithm, we use neighboring (MB) information to predict the mode of the current MB. If the current MB is predicted to be 16 × 16 mode, the mode pre-decision algorithm is used. Otherwise, the mode pre-decision will not be utilized.

Furthermore, we also expand the all-zero detection threshold to our proposed mode pre-decision threshold (MPT), so that when SAD is smaller than MPT, two situations will happen as follows:

1) The mode of the current MB is 16 × 16 and the motion-compensation-residue coefficients will be all-zero.

2) The motion-compensation-residue coefficients of the 16 × 16 block are not all-zero. However, since the value of the first item in Eq. (5) is very small, the R-D cost is mainly decided by the second item. Therefore, 16 × 16 is still the best mode.

In reference to Eq. (1) and (4), we set MPT as 64max [(Q−12), 1]. Since our mode pre-decision algorithm is combined with the information of the neighboring MB, experiments show that this threshold can select the 16 × 16 mode with high accuracy. Furthermore, we also use the similar mode pre-decision algorithm before the decision of modes smaller than 8 × 8 (8 × 8, 8 × 4, 4 × 8 and 4 × 4) and quickly select the 8 × 8 mode, thus further reducing the computation cost.

2.2 Precise large-small mode decision

Reference [4] proposes a large-small mode division algorithm, which divides the seven modes into two groups: large modes (16 × 16, 16 × 8, 8 × 16 and 8 × 8) and small modes (8 × 8, 8 × 4, 4 × 8 and 4 × 4). The algorithm predicts whether the current MB is a large or small mode by testing the correlation of neighboring MBs. If the current MB is predicted to be a large mode, the four large modes are searched first; if the resulting best large mode is not an 8 × 8 mode, then the small mode will not be searched, otherwise, the four small modes will be searched. Similarly, if the current MB is predicted to be a small mode, the four small modes are firstly searched and the large modes will not be searched if the resulting mode is not 8 × 8.

This separation algorithm provides a good idea for fast mode decision. However, the separation result of Ref. [4] is not precise. The wrong decision of large-small mode will lead to the increase of computation cost as well as the decrease of coding performance. The algorithm in Ref. [4] has a large decrease in PSNR and bit rate performance. In order to solve this problem, we propose a precise large-small mode decision algorithm.

The algorithm in Ref. [4] only considers the information of neighboring MBs and the information of the current MB is not known before separation. This is the main reason for its low-separation-accuracy. In this paper, we combine the information of neighboring MBs with that of the current MB to make the separation. Since both large mode and small mode include the 8 × 8 mode, our proposed algorithm searches the 8 × 8 mode first and tests the correlation of the four 8 × 8 MVs. The separation is then made with the help of the neighboring MB information. At the same time, we use the cautious small mode decision algorithm as well as large-mode-tendency decision algorithm in mixed area (see the next section) during our large-small mode decision process, which is based on two reasons:

1) The wrong decision into small mode will easily lead to a best small mode far from the 8 × 8 mode, and thus cannot be corrected to the large mode. This will lead to a great decrease in coding performance. On the other hand, the wrong decision into the large mode usually leads to the best large mode of 8 × 8, and thus can be corrected through small mode search. The resulting coding performance remains unchanged.

2) Small modes have more sub-blocks than large modes; the computational cost to search small modes is 4–10 times that of large modes. The total computational cost to search the large mode for correction after the wrong decision into small mode will be much larger than searching the large mode directly. On the contrary, the computational cost to search the small mode for correction after the wrong decision into the large mode increases slightly compared with searching the small mode directly.

3 The fast block mode selection approach

The description of our proposed algorithm is as follows:

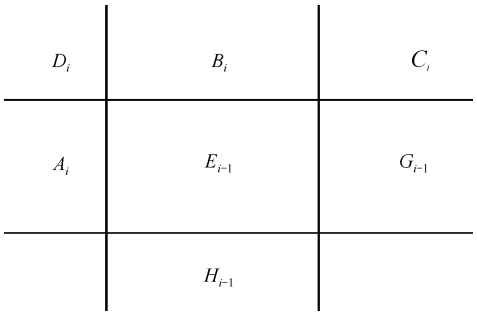

Fig. 1 Neighboring block mode

1) Predicting whether the current MB is 16 × 16 mode. As shown in Fig. 1, A_i, B_i, C_i, D_i represent the neighboring

MB of the current MB, while $E_{i-1}, G_{i-1}, H_{i-1}$ are MBs of the previous frame, which has the same location as the current MB. We define M_{16} as the decision variable of 16 × 16 mode and we make the prediction as follows:

Set M_{16} to be zero, if B_i or A_i exists and has the mode of 16×16, then corresponding to this MB: $M_{16} = M_{16} + 3$; if B_i or A_i does not exist, then $M_{16} = M_{16} + 1$;

If H_{i-1} or G_{i-1} exists and has the mode of 16 × 16, then corresponding to this MB: $M_{16} = M_{16} + 2.5$;

If H_{i-1} or G_{i-1} does not exist, then $M_{16} = M_{16} + 1$;

If D_i or C_i exists and has the mode of 16 × 16, then corresponding to this MB: $M_{16} = M_{16} + 2$;

If D_i or C_i does not exist, then $M_{16} = M_{16} + 0.5$;

If E_{i-1} exists and it has the mode of 16 × 16, then $M_{16} = M_{16} + 3$;

If $M_{16} > 9$, the current MB is predicted to be 16 × 16 mode, then 16 × 16 mode is firstly searched. Otherwise, do not search 16 × 16 mode and go to step 3.

In order to use the information of neighboring MBs more reasonably, we set different weights to different MBs. If a MB does not exist, it is shown that the current MB is on the edge of the image. Since MBs on the edge are more likely to be 16 × 16 mode, the decision variable M_{16} will also add a certain weight value in such case.

2) 16 × 16 mode pre-decision. If 16 × 16 is searched in step 1, the resulting SAD will be compared with MPT. If SAD is smaller than MPT, the best mode is 16 × 16 and the whole motion search process is stopped. Otherwise, go to step 3.

3) 8 × 8 mode search. Get four 8 × 8 motion vectors.

4) Large-small mode decision and motion search. We use the neighboring MB mode as shown in Fig. 1. Suppose M_{big} is the decision variable for large mode and M_{small} is the decision variable for small mode. The decision process is as follows:

i) Processing the information of neighboring MB

Set M_{big}, M_{small} to be zero.

If A_i or G_{i-1} exists and has the mode of 16 × 16 or 8 × 16, then corresponding to this MB: $M_{\text{big}} = M_{\text{big}} + 2$;

If A_i or G_{i-1} exists and has small mode or mode 16 × 8, then corresponding to this MB: $M_{\text{small}} = M_{\text{small}} + 2$;

If A_i or G_{i-1} does not exist, then $M_{\text{big}} = M_{\text{big}} + 1$;

If B_i or H_{i-1} exists and has the mode of 16 × 16 or 16×8, then corresponding to this MB: $M_{\text{big}} = M_{\text{big}} + 2$;

If B_i or H_{i-1} exists and has small mode or mode 8×16, then corresponding to this MB: $M_{\text{small}} = M_{\text{small}} + 2$;

If B_i or H_{i-1} does not exist, then $M_{\text{big}} = M_{\text{big}} + 1$;

If E_{i-1} exists and has the mode of 16 × 16, then $M_{\text{big}} = M_{\text{big}} + 1$;

If E_i–1 exists and it has a small mode, then $M_{\text{small}} = M_{\text{small}} + 2$;

ii) Testing the correlation of the four 8 × 8 MVs.

$$D_x = \sum_n | V_x(n) - \overline{V_x} |$$

$$D_y = \sum_n | V_y(n) - \overline{V_y} |$$

where $\overline{V_x}, \overline{V_y}$ are the average values of MVs in x and y direction, respectively. n is the number of sub-block.

iii) Large-small mode decision

If the following four conditions are satisfied simultaneously: (a) $M_{16} > 11$ (b) $D_x < 3$ (c) $D_y < 3$ (d) $\text{COST}_{8\times8} > \text{COST}_{16\times16}$, the current MB will be 16 × 16 and the whole motion search process is stopped. M_{16} is the 16 × 16 decision variable of step 1 and $\text{COST}_{16\times16}$, $\text{COST}_{8\times8}$ represent the R-D cost of 8 × 8 and 16 × 16 modes, respectively. This process further picks out the 16 × 16 blocks from the MBs left by 16 × 16 mode pre-decision in step 2.

If the following conditions are satisfied simultaneously: (a) $M_{\text{big}} > 6$ (b) $(D_x + D_y) < 7$ (c) $\text{COST}_{8\times8} > \text{BCOST}_{16\times16}$, the current MB is predicted as large mode. The 81 points around the point of $(\overline{V_x}, \overline{V_y})$ are searched for each large mode and the resulting best large mode will be the best mode. $\text{BCOST}_{16\times16}$ is defined as follows: if 16 × 16 is searched in step 1, $\text{BCOST}_{16\times16}$ is just the R-D cost of the 16 × 16 mode; Otherwise, $\text{BCOST}_{16\times16}$ is set a very large value. This process further picks out the large-mode blocks from the MBs left after the 16 × 16 mode pre-decision in step 2.

If $(D_x + D_y) < 13$, the 121 points around the point of $(\overline{V_x}, \overline{V_y})$ are searched for each large mode and the resulting R-D cost is compared with the 8 × 8 mode. If the resulting best mode is not 8 × 8, then the search process is stopped. Otherwise, continue to search the small modes.

If the following conditions are satisfied simultaneously: (a) $M_{\text{small}} > 3$ (b) $(D_x + D_y) > 15$ (3) $\text{COST}_{8\times8} < \text{BCOST}_{16\times16}$, a cautious small mode decision algorithm is utilized, which firstly searches all of the small modes and then searches one large mode of 8 × 16. If the resulting best mode is a small mode and at least one 8 × 8 sub-block is not an 8 × 8 mode, the resulting small mode will be the best mode and the search process is stopped. Otherwise, continue to search the other large modes. Since the wrong decision of small mode will greatly reduce the coding performance, one large mode is searched during the small mode decision to reduce the wrong decision.

In other situations, the large-mode-tendency decision is utilized, which firstly fully searches large modes and then compares the result with 8×8 mode. If the resulting best mode is not 8×8, the search process is stopped. Otherwise, the small modes will be searched to make a refinement.

5) 8 × 8 mode pre-decision. Before searching the small modes, the SADs of each 8 × 8 sub-blocks are compared with the 8 × 8 MPT. If the SAD is smaller than MPT, the best mode will be 8 × 8 and the small modes will not be searched. In reference to Eqs. (1) and (5), we set the 8 × 8 MPT as 16 max [(Q–12), 1], and the simulation results show that such threshold can select the 8 × 8 mode accurately. Since 8 × 8-mode block accounts for over 40 % among all the small mode blocks, such operation will further reduce the computational cost.

4 Discussion of some problems

4.1 Separation of 8 × 8 mode

In the conventional H.264 standard, it searches all small modes of one 8 × 8 sub-block before searching the next 8 × 8 block. If we want to use the 8×8 information, it is necessary to separate 8 × 8 mode from the small mode search cycle. However, such separation will have some impact on coding performance, just as shown in Fig. 2.

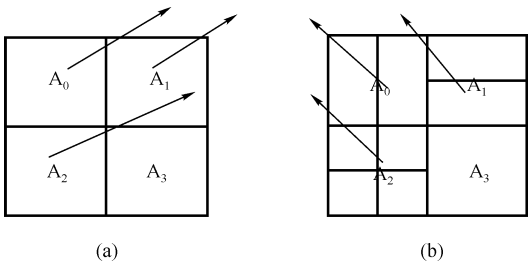

Fig. 2 The figure for 8×8 mode isolation

We use 8 × 8 block A_3 as an example. The MV prediction of A_3 is calculated by the best-mode MVs of its neighboring blocks: A_0, A_1 and A_2. If 8 × 8 mode is separated from the small mode searching cycle, the 8 × 8-mode prediction MV of A_3 is calculated by the 8 × 8 MVs of its neighboring blocks (as shown in Fig. 2 (a)). On the other hand, if 8 × 8 mode is among the small mode searching cycle, there are chances that the best modes of A_0, A_1, A_2 are not 8 × 8. Therefore, the 8 × 8 prediction MV of A_3 may be calculated by other small-mode MVs of its neighboring blocks (as shown in Fig. 2(b)). In such case, the prediction MV of A_3 may have a large deviation from that calculated from 8 × 8-mode-reference MVs, which may which may lead to the decrease of coding performance. To avoid such deviation, one 8 × 8 mode has to be searched again during small mode searching cycle, which leads to the increase of computational cost. In our algorithm, we proposed the following solution:

For A_0 block, since its MV prediction does not use the MVs of the current MB, we do not need to search 8 × 8 mode again for this block. As for other blocks, the neighboring block MVs are used as reference MV (A_1 uses A_0; A_2 uses A_0, A_1; A_3 uses A_0, A_1, A_2). In these cases, we test the differences between the best-mode MV and the 8 × 8 mode MV of the reference-blocks. If the difference is smaller than 3, there is no need to search 8 × 8 mode once more. Otherwise, a large deviation is predicted, and the 8 × 8 mode needs to be searched again with the new prediction MV. According to our statistics, this algorithm only re-searches less than 2% of the 8 × 8 mode while ensuring the coding quality at the same time.

4.2 Prediction of neighboring blocks

The texture information is considered when making large–small mode decisions. During the large mode prediction, if the up- or down-neighboring MBs of the current MB are 16 × 8 modes, or if the left- or right-neighboring MBs are 8 × 16 modes, there are high probabilities that the current MB will be a large mode. Therefore, some weights will be added to the corresponding decision variable. During the small large mode texture mode prediction, if the up- or down-neighboring MBs of the current MB are 8 × 16 modes, or if the left- or right-neighboring MBs are 16 × 8 modes, there are high probabilities that the current MB will be a small mode, therefore, the corresponding decision variable (M_{big}, M_{small}) will be added by some weight values. Furthermore, in consideration of the stripe and edge features, the threshold for M_{small} is set as 3, that is, the current MB may be a small mode if two of its neighboring MBs are small modes (as shown in Fig. 3). The simulation result shows that our setting is reasonable.

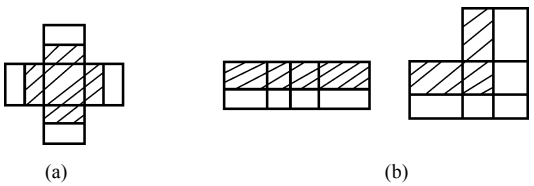

Fig. 3 Large and small mode textures. (a) Large mode texture; (b) Small mode texture

5 Experiment results

We implemented our proposed algorithm on JVT JM version

73 and tested four QCIF sequences “Foreman”, “Stefan”, “Container” and “Akiyo”. We compared the results with full search (FS), and the algorithms of Refs. [3] and [4]. The QP changes from 12 to 40. Comparison result is listed in Table 1.

Table 1 The encoding performance comparison of different algorithm

		Proposed	Ref. [3]	Ref. [4]
Foreman	PSNR decrease	0.041	0.035	0.129
	Bit rate increase/%	1.47	1.23	4.11
	Computation saving/%	72.25	52.87	72.80
Stefan	PSNR decrease	0.025	0.020	0.091
	Bit rate increase/%	0.22	0.35	3.17
	Computation saving/%	72.66	59.22	76.23
Container	PSNR decrease	0.020	0.020	0.070
	Bit rate increase/%	1.19	1.17	5.37
	Computation saving/%	90.29	80.90	76.31
Akiyo	PSNR decrease	0.023	0.025	0.069
	Bit rate increase/%	1.60	1.59	6.01
	Computation saving/%	92.37	80.02	75.44

In our simulation, the sequences are encoded at 30 fps (frame rate). Table 1 compares the whole search points, PSNR decrease and bit rate increase of the three algorithms. We use total search point (TSP) as a measure of computational cost. According to the table, our proposed algorithm can save about 72 % computational cost for the ‘active’ sequences such as Foreman and Stefan. It can save over 90 % computational cost for those ‘quite’ sequences such as Akiyo and Container. At the same time, the coding performance decrease of our proposed algorithm is very low (only a 0.028 dB PSNR decrease and bit rate 1.1 % increase). Compared with the algorithm in Ref. [3], our proposed algorithm can reduce the computational cost of FMBME by about 45 % with similar visual quality and bit rate. Compared with the approach in Ref. [4], our proposed algorithm has the same computation complexity as the Ref. [4] approach for the high-motion sequences such as Foreman and Stefan. When coding low-motion sequences such as Akiyo and Container, our proposed algorithm can reduce the computational cost of the Ref. [4] approach by about 75 %. The proposed algorithm also has 0.06 dB PSNR gain and 3.7 % bit rate savings over the Ref. [4] approach. It is necessary to notice that some of the computational cost saving in the Ref. [4] algorithm is gained through motion search algorithm (such as diamond search and three-step search) while our proposed algorithm mainly focuses on mode selection with few revisions on the motion search algorithm. The algorithm can be improved further with the combination of motion search algorithm.

6 Conclusions

In this paper, a novel fast mode selection algorithm is proposed, which can reduce the computational cost while keeping the visual quality and bit rate performance almost the same.

References

1. Wiegand T., Sullivan G. T., Overview of the H.264/AVC video coding standard, IEEE Trans. Circuit System Video Technology, 2003, 13(7): 560−576
2. Zhang J., He Y., Performance and complexity joint optimization for H.264 video coding, ISCAS2003, Bangkok, Thailand: IEEE, 2003(2): 888−891
3. Chang A., Wong H. W., Yeung Y. M. et al., Fast multi-block selection for H.264 video coding, ISCAS2004, Vancouver, British Columbia, Canada: IEEE, 2004(3): 817−820
4. Xue Jin-zhu, Shen Lan-sun, An efficient block-matching motion estimation algorithm for H.264/AVC, Acta Electronica Sinica, 2004, 32(4): 583−587 (in Chinese)
5. Yu A., Lee R., Early detection of all-zero coefficients in H.263, PCS’97. Berlin, Germany: IEEE, 1997: 159–164
6. ZHOU Xuan, TAN Jing-wei, YU Song-yu, A new method for method for early detection of all-zero DCT coefficients in H.263, Journal of Shanghai Jiao Tong University, 1998, 32(9): 107−109 (in Chinese)
7. Shi J., Yu S., Efficient method for early detection of all-zero DCT coefficients, Electronics Letters of IEE, 2001, 37(3): 160–161
8. Wang Wen-sheng, Yang Ming, Cui Hui-juan et al., An adaptive early termination algorithm for H.264 motion estimation, Journal of Tsinghua University, 2004, 44(10): 1426−1430 (in Chinese)
9. Chang A., Au C., Yeung Y. M. et al., A novel approach to fast multi-block motion estimation for H.264 video coding, Proc. of IEEE Conf. On Multimedia and Expo, Baltimore, Maryland, USA: IEEE, 2003: 405−508
10. H.264, Drift ITU-T recommendation and final draft international standard, Pattaya, Thailand: ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC, 2003
11. JVT, JVT reference software version JM73, http://iphome.hhi.de/suehring/tml/download/old_jm/, Jul. 2003