Signal Processing: *Image Communication* 24 (2009) 630–636

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

Fast motion estimation for H.264

Canhui Cai [a,*], Huanqiang Zeng [b], Sanjit K. Mitra [c]

[a] School of Information Science and Technology, Huaqiao University, Quanzhou 362021, China
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore
[c] Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA

ARTICLE INFO

Article history:
Received 30 August 2008
Received in revised form
11 February 2009
Accepted 12 February 2009

Keywords:
H.264/AVC
Motion estimation
Video coding

ABSTRACT

Several specific features have been incorporated into *Motion estimation* (ME) in H.264 coding standard to improve its coding efficiency. However, they result in very high computational load. In this paper, a fast ME algorithm is proposed to reduce the computational complexity. First, a mode discriminant method is used to free the encoder from checking the small block size modes in homogeneous regions. Second, a condensed hierarchical block matching method and a spatial neighbor searching scheme are employed to find the best full-pixel motion vector. Finally, direction-based selection rule is utilized to reduce the searching range in sub-pixel ME process. Experimental results on commonly used QCIF and CIF format test sequences have shown that the proposed algorithm achieves a reduction of 88% ME process time on average, while incurring only 0.033 dB loss in PSNR and 0.50% increment on the total bit rate compared with that of exhaustive ME process, which is a default approach adopted in the JM reference software.

1. Introduction

H.264/AVC is the newest video coding standard developed by the Joint Video Team (JVT), which is composed of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG) [1]. Compared with its predecessors, H.264/AVC is a more "network-friendly" video coding standard with much higher coding efficiency. Hence, H.264 is expected to be employed in various video services and applications, such as Internet Protocol Television (IPTV), digital satellite broadcast, and so on [2].

Motion estimation (ME) is an important part of data compression methods adopted by all existing video coding standards, such as H.263, MPEG-4, etc. In H.264, the ME mechanism is designed with many new features, such as the variable block size, multiple reference frames, a quarter pixel accurate estimation, and so on. The new features obviously improve the coding efficiency. However, the gain of its high performance is at the price of heavy computational complexity [3]. Experimental results have shown that the ME process usually takes about 60% of the computational load for the case of the single reference frame, and 80% for the case of the multiple reference frames. Therefore, how to reduce the ME computational complexity while still maintaining almost the same coding efficiency has become an important issue in H.264.

Many improved ME algorithms have been proposed to quicken the ME process. Zhu et al. [4] suggested a *diamond search* (DS) method based on the characteristic of the center-biased *motion vector* (MV) distribution typically existed in the real-world video sequences. The DS greatly decreases the search points without sacrificing much coding efficiency, and has been accepted in the MPEG-4 Verification Model (VM) [5]. Nie et al. [6] presented an *adaptive rood pattern search* (ARPS) algorithm, which

* Corresponding author. Tel.: +86 595 22692840.
E-mail addresses: chcai@hqu.edu.cn (C. Cai), zeng0043@ntu.edu.sg (H. Zeng), skmitra@usc.edu (S.K. Mitra).

0923-5965/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2009.02.012

exploited the spatial inter-block correlation to adjust the size of the rood-shaped search pattern according to their motion status. Li et al. [7] proposed a flexible multiple reference frame ME algorithm with adaptive search strategies. With special consideration on the multiple reference frames and block sizes, flexible multiple frame selection and adaptive search strategies for selected frame have been utilized to significantly speed up the ME process. Chen et al. [8] proposed the *hybrid unsymmetrical-cross multi-hexagon-grid search* (UMHexagonS) algorithm, which consisted of four steps: unsymmetrical cross search, uneven multi-hexagon grid search, extended hexagon based search and a small diamond search. The UMHexagonS algorithm effectively reduces the number of candidate blocks within a searching window, and has been adopted by JVT. Kuo et al. [9] proposed a fast ME algorithm based on the motion field distribution and correlation inside a *macroblock* (MB). Tsai et al. [10] proposed a three-dimensional predict hexagon search method to reduce the computational load. Based on the characteristics of MV distribution, the object movement is predicted by a novel searching pattern, and a more accurate searching center is found before ME. Kuo et al. [11] proposed a simple yet effective reference frame selector to speed up the ME process. The suitable reference frames are chosen according to the initial search result of an 8×8 block size.

In this paper, a more efficient ME algorithm is proposed by composing several effective techniques. In our approach, a mode discriminant method is first used to skip checking the small block size modes in the homogeneous region. A condensed hierarchical block matching scheme for ME is then exploited to further lower the computational load. Considering that the first reference frame (previous frame) plays a dominant role in ME, a simplified reference frame decision strategy called spatial neighbor searching scheme, is employed to speed up the selection of the optimal reference frame. To quicken the sub-pixel ME process, direction-based selection rule is utilized to reduce searching points. Experimental results have shown that the proposed algorithm significantly reduces the computational complexity compared with the exhaustive ME algorithm adopted in H.264 reference software, while keeping almost the same coding efficiency.

The rest of this paper is organized as follows. An overview of ME algorithm in H.264 is described in Section 2. The proposed fast ME algorithm is presented in Section 3. The simulation results are shown in Section 4. Finally, the conclusion is drawn in Section 5.

2. Overview of H.264/AVC's motion estimation

Similar to the previous video coding standards, H.264 is also a block-based motion-compensated hybrid video coder. However, it possesses many specifics that considerably affect the coding efficiency and complexity of the ME process. In order to achieve higher coding efficiency, H.264 adopts multiple block sizes for ME. As specified in H.264, an MB can be partitioned into 7 different block sizes or *modes* for inter-frame prediction. The block sizes can be of 16×16, 16×8, 8×16, and 8×8. The 8×8 mode (denoted as $P8 \times 8$) can further lead to another 4 small block sizes, namely, 8×8, 8×4, 4×8, and 4×4, as illustrated in Fig. 1 [12]. Moreover, H.264 supports multiple reference frames. In other words, more than one previously decoded frame can be employed as the reference frames to code the current frame.

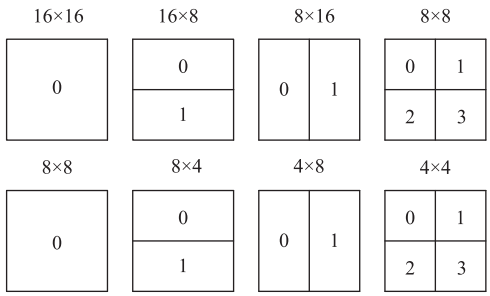

Fig. 1. The various H.264 block sizes.

In the JM reference software of H.264, the Lagrangian *rate distortion optimization* (RDO) function shown in Eq. (1) is exploited as its block size decision criterion [13].

$$J(m, \lambda_{MOTION}) = SAD(s, c(m)) + \lambda_{MOTION} \times R(m - p) \quad (1)$$

where $m = (m_x, m_y)^T$ is the current MV, $p = (p_x, p_y)^T$ is the predicted MV, and λ_{MOTION} is the Lagrangian multiplier. SAD is used for the distortion measurement, and $R(m-p)$ represents the number of bits required to code the difference between the current MV m and the predicted MV p.

All block sizes are investigated based on this criterion and the one with minimum cost is considered to be the optimal one. Consequently, the computational complexity resulted from such an exhaustive search is extremely high, and a *fast* ME algorithm to effectively reduce the computational complexity is therefore desirable.

3. The proposed fast motion estimation algorithm

3.1. Mode discriminant

In our approach, these 7 modes are classified into two categories: large block size class, which includes 16×16, 16×8 and 8×16, and small block size class, which includes 8×8, 8×4, 4×8 and 4×4. Intuitively, one can perceive that the large block size modes are suitable for coding homogenous regions in slow motion, while the small block size modes help to improve the coding efficiency for regions in fast motion. For most of natural videos, the background scenery is often kept still or is in fairly slow motion. Hence, the probability of using large block size modes is higher than that of using the small ones. Further investigations have indicated that the optimal mode is related to the SAD of an MB. Generally, an MB whose optimal mode belongs to the small block size class yields larger SAD. In other words, if the SAD of an MB is smaller than a threshold, its optimal mode will fall into the large block size class.

Based on the above observation, a mode discriminant method is suggested as follows:

(1) Calculate $SAD_{(0,0)}$, the SAD value of the current MB (16×16) via a *zero motion vector* (ZMV).
(2) If $SAD_{(0,0)}$ is smaller than a threshold T, skip the small block size modes; otherwise search all seven modes to find out the optimal one.

Note that the threshold T is Qp dependent, and its value can be determined by extensive experiments. For this, all MBs from a set of commonly used test sequences, Akiyo, Silent, News, Table, Foreman, and Mobile, are employed to empirically determine the reliable threshold values for T. Our experimental results have shown that with a goal of achieving 90% degree of confidence, the threshold T can be determined by the following equation:

$$T = 800 + (Qp - 24) \times 500; \text{ for } Qp = [24, 28, 32, 36, 40] \quad (2)$$

In other words, if the $SAD_{(0,0)}$ of the current MB is smaller than T, the probability that its optimal mode falls into the large block size class is 0.9.

The use of the mode discriminant method enables the proposed algorithm to eliminate the execution of time-consuming small block size mode investigation in the homogeneous regions. As a result, the computational complexity is considerably decreased.

3.2. Condensed hierarchical block matching

To speed up the optimal MV search process for a given mode, a condensed hierarchical block matching scheme is proposed based on two observations resulted from extensive experiments:

(1) Decision of the MB's optimal block size depends on its bandwidth. If the optimal block size of an MB is $M \times N$, then an $M \times N$ block in the MB can be well approximated by a *condensed block*, which is made up of its 16 sample pixels resulted from a down-sampling operation:

$$P_d(x, y) = P_0(mx, ny); \text{ for } 0 \leqslant x < 4, \quad 0 \leqslant y < 4 \quad (3)$$

where $m = M/4$, $n = N/4$, $P_d(x, y)$ denotes the pixel in the condensed block, and $P_0(x, y)$ represents the pixel in the original $M \times N$ block.
(2) Using not all pixels but only the 4×4 sample pixels to find the best matching block in a sampled searching window makes little difference in motion estimation and motion compensation.

As a result, MV searching for the $M \times N$ block can be simplified as using its 4×4 sample pixels for block matching in a sampled searching window. To speed up the search process, a hierarchical block matching algorithm [14] is employed in the 4×4 block matching in this work. Consequently, there are two down-sampling procedures in the hierarchical block matching: (1) down-sampling current block and its reference block to relevant 4×4 pixel blocks; (2) down-sampling each 4×4 block to a 2×2 pixel block. To ease following discussion, SAD of the 2×2 block and SAD of the 4×4 block are referred as 4SAD and 16SAD, respectively. Since not all pixels but only pixels in the condensed block are used in the block matching, the method is thus called *condensed hierarchical block matching* (CHBM). Fig. 2 shows coding results on a CIF test sequence "Foreman" by using CHBM and original H.264 ME in single reference frame and full-pixel accuracy, respectively. From the figure, one can see that there is little performance difference between two methods.

Early termination strategy is also used to further decrease the computational load before ME is performed on the given block. Seven threshold values, T_i, $i = (1,2,3,4,5,6,7)$, are used for the aforementioned seven block sizes, respectively. Similarly, all the MBs from a set of test sequences with different motion activities, including "Akiyo, Silent, News, Table, Foreman, Mobile", are employed to empirically determine the reliable threshold values T_i—with a goal of achieving 90% degree of confidence. The resulted threshold values are documented in Table 1.

In summary, the condensed hierarchical block matching method for a given block is described as follows:

(1) Early termination checking: select the first reference frame as the reference frame, and compute the SAD values of *prediction motion vector* (PMV) and ZMV for the given block, respectively. Denote the smaller one as SAD_{pmv}. If SAD_{pmv} is lower than the corresponding threshold T_i, the PMV is selected as the optimal one and the searching process for the optimal MV is

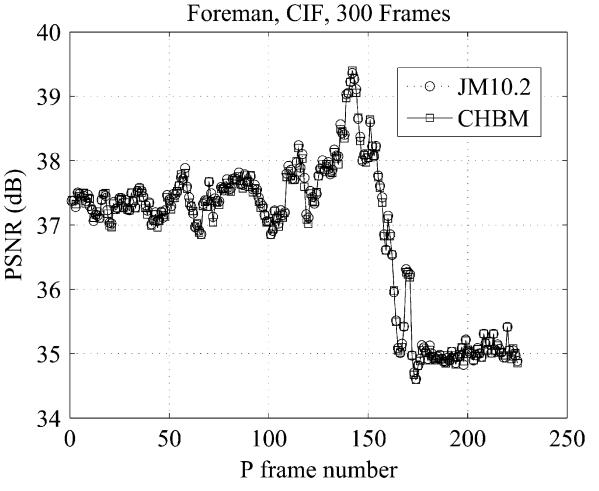

Fig. 2. The performances resulted from CHBM and original H.264 ME in single reference frame and full-pixel accuracy, respectively.

Table 1
The value of T_i.

T_1	T_2	T_3	T_4	T_5	T_6	T_7
2500	1450	1450	920	600	600	500

terminated. Otherwise, choose the MV with a smaller SAD as the search center and proceed to the next step.
(2) Employ full search strategy to find the reference block with the smallest 4SAD, to be called the initial matching point in this paper.
(3) Calculate the 16SADs of the initial matching point and its 8-neighbors.
(4) Adopt the one that yields the minimum 16SAD as the best MV of the first reference frame.

3.3. Simplified optimal reference frame decision scheme

In order to study the effect of varying the number of reference frames on H.264 encoding, extensive simulation experiments are conducted by exploiting the exhaustive ME of the H.264 reference software over a CIF format video sequence "Foreman", and the result is described in Fig. 3. This study clearly indicates that as the number of reference frames increase, the improvement of the coding efficiency is limited, but the computational complexity is dramatically increased. It implies that it is possible to reschedule the MV search procedure to greatly reduce ME time, with little loss in PSNR and increment in bit rate.

Table 2 shows the statistical distribution of the best prediction blocks from 5 reference frames, Ref1 to Ref5. The reference frames are numbered in temporal order, with the previous frame being the first reference frame Ref1. From Table 2, one can see that more than 80% of the optimal prediction blocks are from Ref1 and other four reference frames are seldom used.

Based on the perception that the best MV in the next reference frame is hopefully located in the neighborhood of the best MV in current one, a simplified optimal reference frame decision scheme, called spatial neighbor searching, is employed to find the best MVs of Ref2 Ref3 Ref4 and Ref5:

(1) For a given reference frame except Ref1, employ the best MV obtained from previous reference frame as the initial matching point.
(2) Calculate the 16SADs of the initial matching point and its 8-neighbors.
(3) Adopt the point that yields the minimum 16SAD as the best MV of this reference frame.
(4) Repeat Step 1–Step 3 to find out respective best MVs for these reference frames.
(5) Compare the 16SADs of the best MVs of all reference frames (e.g., from Ref1 to Ref5) and select the one with the minimum 16SAD as the best full-pixel MV.

3.4. Fast sub-pixel motion estimation by directional selection rule

Let solid circle points represent full-pixel position, point O be the best full-pixel MV, and triangles express its 8 1/2-pel neighbors, as shown in Fig. 4. In original H.264 reference software, the sub-pixel ME algorithm first searches the best 1/2-pel MV by checking 8 1/2-pel points around the best full-pixel MV, and then checks 8 1/4-pel point around the best 1/2-pel MV to find the best 1/4-pel MV. Although the time used in such optimal sub-pixel MV searching is only a small fraction of ME time in the original JM reference software, it is considerably higher than the best full-pixel MV searching time in the proposed algorithm. Hence, an efficient fast sub-pixel ME method is designed to further reduce related computational time.

Based on the observation that the 1/2-pel point locates between the best full-pixel MV and its neighbor with smallest SAD has the highest possibility to be the best 1/2-pel MV, we check only those most possible candidate points to reduce the searching range. Denote SADs of the four full-pixel points, H_1, H_2, V_1, and V_2 in Fig. 4, as SAD_{H_1},

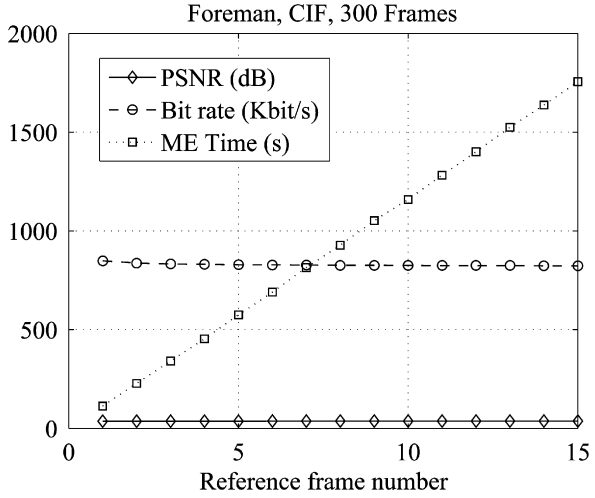

Fig. 3. The coding performance vs. reference frame number.

Table 2
Distribution of the best prediction blocks.

QCIF sequences	Ref1	Ref2	Ref3	Ref4	Ref5
Akiyo	99.5	0.2	0.1	0.1	0.1
Silent	99.2	0.4	0.2	0.1	0.1
News	98.8	0.5	0.3	0.2	0.2
Table	89.6	5.3	2.3	1.5	1.3
Foreman	85.7	6.2	3.6	2.5	2.0
Mobile	81.9	5.2	5.7	4.2	3.0

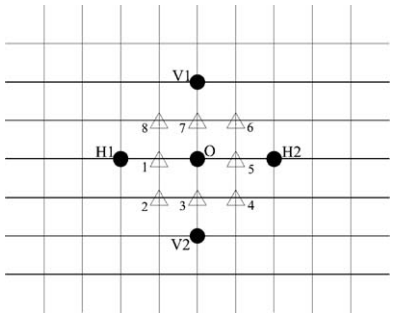

Fig. 4. Sub-pixel search pattern in H.264.

SAD_{H_2}, SAD_{V_1}, and SAD_{V_2} respectively. Then, the proposed fast sub-pixel ME method can be summarized as follows:

(1) Find a point with the smallest SAD from the H_1, H_2, V_1, V_2, and name it X.
(2) Check X's 2 one-pixel neighbors and denote the one with the smaller SAD as Y. For instance, if $X = H_1$, then V_1 and V_2 are its 2 one-pixel neighbors. If $SAD_{V_1} < SAD_{V_2}$ then $Y = V_1$ and vice versa.
(3) Choose the 1/2-pel point in the middle of O and X, the 1/2-pel point in the middle of Y and X, and point O as the candidates of the best 1/2-pel MV. For example, if $X = H_1$ and $Y = V_1$ then the candidates are points 1, 8 and O, respectively. Table 3 documents the candidate points except O in all possible cases.
(4) Decide the best 1/2-pel MV by checking the three candidates.
(5) Check 8 1/4-pel points around the best 1/2-pel MV to find the best 1/4-pel MV.

Compared with the original sub-pixel ME process in H.264 reference software, the proposed algorithm chooses three candidates of the 1/2-pel MV instead of nine. Note that sub-pixel ME takes considerable time in our approach, so the proposed fast sub-pixel ME method can further decrease the computational load.

3.5. The proposed fast motion estimation algorithm

Finally, the proposed fast ME algorithm for H.264 can be summarized as follows:

(1) Compute the SAD value of the current MB (16×16) with ZMV, $SAD_{(0,0)}$. If $SAD_{(0,0)} < T$, only the large block sizes are chosen as the candidate block sizes; otherwise, all the block sizes are selected as candidates.
(2) Select candidate block size and perform early termination checking. If SAD_{pmv} is lower than relative threshold, go to step 5, otherwise, proceed to next step.
(3) Find the best full-pixel MV by using condensed hierarchical block matching method and spatial neighbor searching scheme.
(4) Perform the fast sub-pixel ME around the best full-pixel MV obtained to find the best sub-pixel MV and calculate the relevant cost.
(5) Repeat Step 2–Step 4 likewise one by one for other candidate block sizes.

Table 3
The candidate points for fast sub-pixel search method.

X	Y	Candidate points
H_1	V_1	1, 8
H_1	V_2	1, 2
H_2	V_1	5, 6
H_2	V_2	4, 5
V_1	H_1	7, 8
V_1	H_2	6, 7
V_2	H_1	2, 3
V_2	H_2	3, 4

(6) Among all the block sizes checked in the previous steps, select the one that yields the minimum cost as the best block size, and corresponding best sub-pixel MV as the selected one.

4. Experimental results and discussion

In order to evaluate its performance, the proposed algorithm has been implemented on JM10.2 reference software provided by JVT [15], and experiments on multiple video sequences in both QCIF and CIF formats have been carried out. The test conditions are set as follows: (1) for each test sequence, 300 frames are encoded with IPPP… *group of picture* (GOP) structure; (2) quantization parameter Qp is set at 24, 28, 32, 36 and 40, respectively; (3) RD optimization is enabled; (4) the CABAC entropy coding is used; and (5) the maximum search range is ± 16 with 5 reference frames. The experimental platform is a 1.6 GHz Intel Pentium IV processor with 512 MB memory.

Table 4 shows the experimental results by only using fast full-pixel ME scheme based on condensed hierarchical block matching method and spatial neighbor searching scheme, compared with the exhaustive ME searching

Table 4
Performance comparison of only using fast full-pixel ME scheme and the exhaustive ME approach.

CIF sequences	$\Delta PSNR$ (dB)	ΔB (%)	ΔT (%)	ΔMET (%)
Akiyo	+0.00	−0.60	−54.28	−71.50
News	+0.00	+0.06	−51.56	−69.89
Foreman	−0.01	+0.10	−47.08	−63.80
Tempete	−0.01	+0.00	−41.13	−62.28
Mobile	−0.01	+0.30	−37.13	−60.03
Stefan	−0.01	+0.48	−40.73	−61.59

Table 5
Performance comparison of (A) Chen et al. [8] (UMHexagonS), (B) Chen et al. [16] (simplified UMHexagonS), and (C) our proposed method.

QCIF sequences	Method	$\Delta PSNR$ (dB)	ΔB (%)	ΔT (%)	ΔMET (%)
Akiyo	(A)	+0.02	+0.05	−54.92	−76.39
	(B)	+0.00	+0.03	−62.50	−87.11
	(C)	−0.03	−0.15	−70.42	−93.69
News	(A)	−0.01	+0.00	−51.70	−74.71
	(B)	+0.00	−0.09	−57.94	−84.23
	(C)	−0.04	+0.37	−66.12	−92.78
Container	(A)	+0.00	+0.02	−53.52	−76.70
	(B)	+0.01	+0.00	−59.31	−84.75
	(C)	−0.02	−0.36	−66.84	−92.76
Silent	(A)	−0.01	+0.16	−49.89	−70.41
	(B)	−0.02	+0.28	−55.64	−80.52
	(C)	−0.03	+1.20	−65.87	−91.25
Foreman	(A)	−0.01	−0.26	−45.75	−65.36
	(B)	−0.02	−0.15	−51.77	−73.86
	(C)	−0.05	+0.92	−61.88	−86.94
Mobile	(A)	+0.00	−0.13	−37.10	−61.36
	(B)	−0.01	−0.05	−42.68	−70.12
	(C)	−0.03	+0.60	−49.59	−81.12

All are compared with the exhaustive ME approach, individually.

approach in H.264 reference software. Tables 5 and 6 compare the outcomes between the proposed algorithm and the methods proposed in [8] and [16]. Due to the limited paper length, only experimental results with $Qp = 28$ are presented. In these tables, $\Delta PSNR$ is the average PSNR deviation; ΔB is the total bit rate change in terms of percentage; ΔT is the average time saving (in percentage) for encoding the entire sequence; ΔMET is the average time saving (in percentage) for the ME process; "+" means increase; and "−" means decrease. These performance indexes follow the specification suggested in [17].

From Table 4, one can see that the proposed algorithm efficiently decreases the computational load by only using fast full-pixel ME scheme based on condensed hierarchical block matching method and spatial neighbor searching scheme, while keeping PSNR and bit rate used almost unchanged. The experimental data shown in Tables 5 and 6 have demonstrated that the proposed algorithm achieves a very good trade-off between computational complexity and coding efficiency. By contrast with the outcomes of JM10.2, the proposed algorithm achieves consistent ME time saving (88% on average) and about 62% total time reduction for encoding the entire sequences with only 0.033 dB loss in PSNR and 0.50% increment in the total bit rate. Experimental results also show the proposed algorithm consistently outperforms the methods proposed in Refs. [8,16], with about 20%, 12% coding time saving, respectively, while keeping similar bit rate and PSNR. Fig. 5 demonstrates the 64th decoded frame of the CIF test sequence "Foreman". It can be seen that the image reconstructed by the proposed algorithm is with almost the same visual quality as that reconstructed by the exhaustive ME searching approach.

Table 6
Performance comparison of (A) Chen et al. [8] (UMHexagonS), (B) Chen et al. [16] (simplified UMHexagonS), and (C) our proposed method.

CIF sequences	Method	$\Delta PSNR$ (dB)	ΔB (%)	ΔT (%)	ΔMET (%)
Akiyo	(A)	+0.01	−0.04	−55.95	−76.38
	(B)	+0.01	−0.03	−63.96	−87.20
	(C)	−0.02	−0.45	−70.16	−93.58
News	(A)	−0.01	−0.11	−53.37	−75.08
	(B)	−0.01	−0.06	−60.45	−84.86
	(C)	−0.03	+0.32	−67.39	−92.69
Foreman	(A)	−0.01	−0.04	−47.06	−65.03
	(B)	−0.04	+0.03	−52.89	−72.93
	(C)	−0.05	+0.88	−62.86	−86.26
Tempete	(A)	−0.01	−0.08	−41.43	−63.11
	(B)	−0.01	−0.07	−46.92	−71.71
	(C)	−0.03	+0.30	−55.36	−84.55
Mobile	(A)	+0.00	−0.02	−38.21	−62.04
	(B)	−0.01	+0.01	−41.78	−67.67
	(C)	−0.03	+0.76	−49.22	−80.03
Stefan	(A)	−0.01	+0.14	−41.63	−63.35
	(B)	−0.03	+0.02	−44.94	−68.45
	(C)	−0.04	+1.55	−54.84	−83.38

All are compared with the exhaustive ME approach, individually.

5. Conclusion

In this paper, a fast ME algorithm for the H.264 encoder optimization is presented, including mode discriminant technique, the fast full-pixel ME scheme based on condensed hierarchical block matching measure and spatial neighbor searching scheme, and fast sub-pixel ME method by directional selection rule. Experimental results have verified that the proposed fast ME algorithm is consistently speeding up the ME process by achieving 88% reduction as compared with that of exhaustive ME searching approach in the JVT reference software (version JM10.2), at the expense of incurring a negligible loss of PSNR (about 0.033 dB on average) and slightly increment of the total bit rate (about 0.50% on average).

Acknowledgements

This work is partially supported by the National Natural Science Foundation of China under Grant 60772164 and the Fujian Province Natural Science Foundation under Grant A0710009.

Fig. 5. The 64th reconstructed frame of CIF test sequence "Foreman" resulted from (A) the exhaustive ME searching approach: 37.64 dB and (B) our proposed method: 37.60 dB.

References

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14 496-10 AVC, Joint Video Team, March 2003.

[2] T. Wiegand, G.J. Sullivan, G. Bjonte, A. Luthra, Overview of the H.264/AVC video coding standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 560–576.

[3] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, T. Wedi, Video coding with H.264/AVC: tools, performance, and complexity, IEEE Circuits and Systems Magazine 4 (1) (2004) 7–28.

[4] S. Zhu, K.-K. Ma, A new diamond search algorithm for fast block-matching motion estimation, IEEE Transactions on Image Processing 9 (2) (2000) 287–290.

[5] MPEG-4 Video Verification Model (Version 14.0), ISO/IEC JTC1/SC29/WG11 N2932, October 1999.

[6] Y. Nie, K.-K. Ma, Adaptive rood pattern search for fast block-matching motion estimation, IEEE Transactions on Image Processing 11 (12) (2002) 1442–1449.

[7] X. Li, E.Q. Li, Y.-K. Chen, Fast multi-frame motion estimation algorithm with adaptive search strategies in H.264, IEEE International Conference on Acoustics, Speech, and Signal Processing 3 (2004) 369–372.

[8] Z. Chen, P. Zhou, Y. He, Fast Integer Pel and Fractional Pel Motion Estimation for JVT, Documents JVT-F017, JVT 6th Meeting, Awaji Island, Japan, December 2002.

[9] T.-Y. Kuo, C.H. Chan, Fast variable block size motion estimation for H.264 using likelihood and correlation of motion field, IEEE Transactions on Circuits and Systems for Video Technology 16 (10) (2006) 1185–1195.

[10] T.-H. Tsai, Y.-N. Pan, A novel 3-D predict hexagon search algorithm for fast block motion estimation on H.264 video coding, IEEE Transactions on Circuits and Systems for Video Technology 16 (12) (2006) 1542–1549.

[11] T.-Y. Kuo, H.-J. Lu, Efficient reference frame selector for H.264, IEEE Transactions on Circuits and Systems for Video Technology 18 (3) (2008) 400–405.

[12] I.E.G. Richardson, H.264/MPEG-4 Part 10 White Paper: Inter Prediction, April 2003, 〈http://www.vcodex.com/h264.html〉.

[13] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, G.J. Sullivan, Rate-constrained coder control and comparison of video coding standards, IEEE Transactions on Circuits and System for Video Technology 13 (7) (2003) 688–703.

[14] M. Bierling, Displacement estimation by hierarchical block matching, Proceedings SPIE Visual Communications and Image Processing 1001 (1988) 942–951.

[15] Joint Video Team, Reference Software JM10.2, 〈http://iphome.hhi.de/suehring/tml/download/old_jm/〉.

[16] Z. Chen, P. Zhou, Y. He, Fast Motion Estimation for JVT, Documents JVT-G016, JVT 7th Meeting, Pattaya, Thailand, March 2003.

[17] G. Bjontegaard, Calculation of Average PSNR Differences between RD-Curves, Document VCEG-M33, VCEG 13th meeting, Austin, Texas, USA, April 2001.