Available online at www.sciencedirect.com
ScienceDirect
J. Vis. Commun. Image R. 19 (2008) 256–269
JOURNAL OF VISUAL Communication & IMAGE Representation
www.elsevier.com/locate/jvci

Content-Aware Fast Motion Estimation Algorithm

Yi-Wen Chen *, Ming-Ho Hsiao, Hua-Tsung Chen, Chi-Yu Liu, Suh-Yin Lee

College of Computer Science, National Chiao Tung University, 1001 Ta-Hsueh Road, Hsinchu, Taiwan

Received 31 July 2007; accepted 24 January 2008
Available online 19 February 2008

Abstract

In this paper, we propose the Content-Aware Fast Motion Estimation Algorithm (CAFME) that can reduce computation complexity of motion estimation (ME) in H.264/AVC while maintaining almost the same coding efficiency. Motion estimation can be divided into two phases: searching phase and matching phase. In searching phase, we propose the Simple Dynamic Search Range Algorithm (SDSR) based on video characteristics to reduce the number of search points (SP). In matching phase, we integrate the Successive Elimination Algorithm (SEA) and the integral frame to develop a new SEA for H.264/AVC video compression standard, called Successive Elimination Algorithm with Integral Frame (SEAIF). Besides, we also propose the Early Termination Algorithm (ETA) to early terminate the motion estimation of current block.

We implement the proposed algorithm in the reference software JM9.4 of H.264/AVC and the experimental results show that our proposed algorithm can reduce the number of search points about 93.1%, encoding time about 42%, while maintaining almost the same bitrate and PSNR.

Keywords: Motion estimation; Successive Elimination Algorithm; Integral frame; Search range; H.264/AVC; SAD; Motion vector; Computational complexity

1. Introduction

Block matching-based motion estimation (ME) and compensation is a fundamental process in international video coding standards, such as MPEG-1, MPEG-2, MPEG-4, ITU-T H.263, and H.264, which can efficiently remove temporal redundancy. Since an ME module is usually the most computational-intensive part in a typical video encoder (about 50–90% of the entire system), a efficient ME module is essential and vital.

In recent years, many fast motion estimation algorithms have been proposed. Some algorithms like Three-Step Search (TSS) [1] and Diamond Search (DS) [2], search the best matched blocks following a predefined search pattern to speed up the searching process. The Successive Elimination Algorithm (SEA) [3] is a lossless approach which can avoid unnecessary computation of the sum of absolute difference (SAD) and reduce the computation complexity while maintaining the same performance. The Window Follower Algorithm (WFA) [7] can dynamically adjust the size of the search window to avoid unnecessary computations. Since the video content varies dramatically, these algorithms do not always perform well for videos of various activities. The drawbacks and advantages of these algorithms are listed in Table 1.

In this paper, we propose the Content-Aware Fast Motion Estimation (CAFME) Algorithm to speed up the motion estimation considering the video content. The CAFME consists of the Simple Dynamic Search Range Algorithm (SDSR), Successive Elimination Algorithm with Integral Frame (SEAIF), and Early Termination Algorithm (ETA). The SDSR adjusts search range adaptively according to the motion activity of the video. The experiments show that the SDSR performs well for the video of different kinds of motion activity. The SEAIF is designed for saving the computing of block matching in

* Corresponding author. Fax: +886 3 5724176.
E-mail addresses: ewchen@csie.nctu.edu.tw (Y.-W. Chen), mhhsiao@csie.nctu.edu.tw (M.-H. Hsiao), huatsung@csie.nctu.edu.tw (H.-T. Chen), liucy@csie.nctu.edu.tw (C.-Y. Liu), sylee@csie.nctu.edu.tw (S.-Y. Lee).

1047-3203/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2008.01.002

Table 1
Advantages and drawbacks of fast motion estimation algorithms

Category	Advantage	Drawback
Follow certain search pattern	√ Number of SP is very small √ Reduce considerable computation	√ Local minimum problem √ Unsuitable for high motion √ Coding efficiency degradation
Adjust search window size	√ Number of SP is small √ Reduce considerable computation	√ Need thresholds √ Unsuitable for sudden motion change √ Substantial overhead √ Coding efficiency degradation
Reduce matching complexity	√ Reduce considerable computation √ Lossless approach	√ Substantial overhead √ Unsuitable for hardware √ Coding efficiency degradation

motion estimation of H.264/AVC and the ETA could skip some search points in motion estimation by early termination of the searching process. Although the CAFME consists of the SDSR, SEAIF, and ETA, these three algorithms can be used independently. The experimental result shows that the proposed algorithms could reduce the computing time-of-motion estimation and maintain almost the same coding efficiency compared with Full Search.

The paper is organized as follows: Section 2 introduces the related background knowledge. In Section 3, we present how the Content-Aware Fast Motion Estimation Algorithm is designed and developed. Section 4 reports the significant experimental results. Finally, the conclusions and future works are given in Section 5.

2. Related works

2.1. Matching criterion

Matching criterion is exploited as a quality evaluation metric for motion estimation algorithms to find out the best matched block. Mean square difference (MSD), mean absolute difference (MAD), and sum of absolute difference (SAD) are frequently used criteria. Their definitions can be described by the following equations.

$$\mathrm{MSD}(f_c,f_r(m,n)) = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(f_c(i,j)-f_r(i+m,j+n))^2 \quad (1)$$

$$\mathrm{MAD}(f_c,f_r(m,n)) = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|f_c(i,j)-f_r(i+m,j+n)| \quad (2)$$

$$\mathrm{SAD}(f_c,f_r(m,n)) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|f_c(i,j)-f_r(i+m,j+n)| \quad (3)$$

M and N are the width and height of a block, respectively. m and n are horizontal and vertical components of motion vector, respectively. f_c and f_r are the current and reference blocks, respectively. Among these matching criteria, SAD is a multiplication-free method which enables efficient implementations in hardware and software. Therefore, SAD is chosen as the criterion for block matching in the international video coding standards.

Unlike other video coding standards, H.264 uses the Lagrange multiplier to compute the rate distortion cost for selecting the best partition from seven block partitions for inter-prediction of a macroblock. The best matched block is selected by minimizing the following Lagrange cost.

$$J(\mathrm{MV},\lambda_{\mathrm{motion}}) = \mathrm{SAD}(f_c,f_r(m,n)) + \lambda_{\mathrm{motion}} \cdot \mathrm{Rate}(\mathrm{MV}-\mathrm{MV_P}) \quad (4)$$

$\mathrm{MV} = (m,n)$ is the motion vector, $\mathrm{MV_P} = (m_{Px}, n_P)$ is the prediction for motion vector, and $\lambda_{\mathrm{motion}}$ is the Lagrange multiplier. The function $\mathrm{Rate}(\mathrm{MV}-\mathrm{MV_P})$ represents the predicted motion error and is implemented by a look-up table [12].

2.2. Integral frame

Integral frame is proposed by Viola and Jones [13] to efficiently compute the sum of pixel values within any rectangle area in a frame. The main idea of integral frame is to first calculate the value of integral frame at pixel (p,q) in a frame f, denoted by $I_f(p,q)$ as defined in the Eq. (5), in which $f(i,j)$ represents the pixel value at position (i,j). From Eq. (5), we can see that the value of the integral frame $I_f(p,q)$ is the sum of the pixel values within the rectangle whose top left corner is $(0,0)$ and bottom right corner is (p,q). As Fig. 1 shows, the value of integral frame $I_f(p,q)$ is the sum of the pixel values within the gray area.

$$I_f(p,q) = \sum_{i=0}^{p}\sum_{j=0}^{q} f(i,j) \quad (5)$$

We could analyze the computational cost for an integral frame value with the following equations: let $R_f(p,q)$ be the sum of pixel values from pixel$(0,q)$ to pixel (p,q) in row q. By using Eqs. (9) and (10) recursively, one can compute all the values of integral frame I_f (i,j) at pixel (i,j)

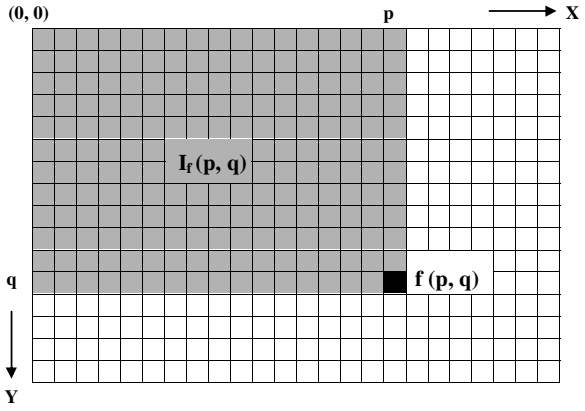

Fig. 1. Integral frame.

within a frame in one pass. For a frame of $W \times H$ pixels, $2WH$ additions are required to compute all the integral frame values.

$$R_f(p,q) = \sum_{i=0}^{p} f(i,q) \tag{6}$$

$$R_f(-1,q) = 0 \tag{7}$$

$$I_f(p,-1) = 0 \tag{8}$$

$$R_f(p,q) = R_f(p-1,q) + f(p,q) \tag{9}$$

$$I_f(p,q) = I_f(p,q-1) + R_f(p,q) \tag{10}$$

After we calculate all the integral frame values within a frame, the sum of pixel values in any rectangular block in the frame can be computed by three arithmetic operations. For example, as illustrated in Fig. 2, the block sum of block D, denoted as BS(D), could be acquired by only three operations as Eq. (11) shows:

$$\begin{aligned} \mathrm{BS}(D) &= \sum_{i=r+1}^{p} \sum_{j=s+1}^{q} f(i,j) \\ &= I_f(p,q) - I_f(r,q) - I_f(p,s) + I_f(r,s) \end{aligned} \tag{11}$$

Because integral frame speeds up the computation of the block sums, our proposed fast block-matching algorithm utilizes the concept of integral frame to further improve the efficiency of motion estimation in the process of video encoding.

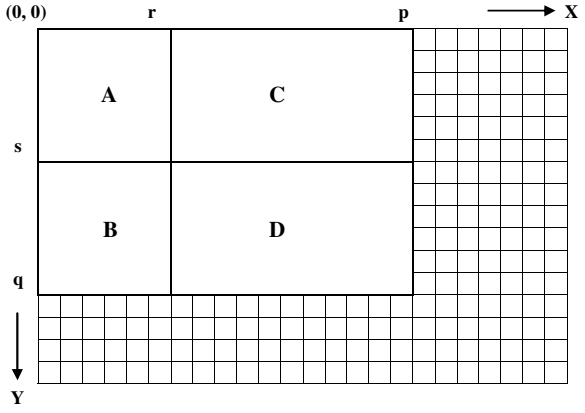

Fig. 2. Computation of block sum.

2.3. Successive Elimination Algorithm (SEA)

In motion estimation, once the SAD value between the current block and the candidate block is computed, it is compared with the current minimum SAD value. If the newly computed SAD value is smaller than the current minimum SAD, the candidate block is considered as the up-to-date best matched block. In order to reduce the computation of SAD, Successive Elimination Algorithm (SEA) [3] was proposed to speed up the motion estimation by pruning the unnecessary computation. The main idea of SEA can be shown in the Eq. (12), in which f_c and f_r represent the current block and the candidate block, BS_c and BS_r are the block sums of the current block and candidate block, respectively. sea(f_c,f_r) is a value computed by substracting the block sum BS_c from the block sum BS_r.

$$\begin{aligned} \mathrm{SAD}(f_c,f_r) &= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |f_c(i,j) - f_r(i,j)| \\ &\geqslant \left| \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f_c(i,j) - \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f_r(i,j) \right| \\ &\equiv |\mathrm{BS}_c - \mathrm{BS}_r| \\ &\equiv \mathrm{sea}(f_c,f_r) \end{aligned} \tag{12}$$

SAD (f_c,f_r) is equal to or larger than sea(f_c,f_r). If sea(f_c,f_r) is larger than the current minimum SAD, SAD(f_c,f_r) must be larger than the current minimum SAD, and therefore, computation of SAD(f_c,f_r) can be skipped. Besides, computing sea value is easier than computing SAD, because BS_c needs to be calculated only once and BS_r can be derived from the previous value of BS_r. Hence, SEA can efficiently reduce the computation of SAD.

Multilevel SEA (MSEA) proposed in [4] is a generalized SEA. MSEA partitions a block into several sub-blocks and calculates the BS for each sub-block to generate a tighter decision value. The block is partitioned in a multi-level manner. At the L-level partition, the block is divided into 2^{2L} sub-blocks of size $N/2^L \times N/2^L$. The msea(f_c,f_r) value for current block f_c and candidate block f_r is then computed by summing the absolute differences of the corresponding block sum (BS) of each sub-block. The mesa(f_c,f_r) is always equal to or larger than sea(f_c,f_r). Consequently, the msea(f_c,f_r) is a lower bound of SAD, as described in Eq. (13). In Eq. (13), k is the index of sub-block and L is the level of division. When the block size is 16×16 ($M = 16$, $N = 16$), MSEA with level $L = 0$ corresponds to SEA, and MSEA with level $L = 4$ corresponds to SAD. Obviously, the decision bound is tighter when the level L is larger; however, the computational cost is also higher.

$$\begin{aligned}\mathrm{SAD}(f_c, f_r) &= \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}|f_c(i,j) - f_r(i,j)| \\ &\geqslant \sum_{k=0}^{2^{2L-1}}|\mathrm{BS}_{ck} - \mathrm{BS}_{rk}| \\ &\equiv \mathrm{msea}(f_c, f_r) \\ &\geqslant |\mathrm{BS}_c - \mathrm{BS}_r| \\ &\equiv \mathrm{sea}(f_c, f_r)\end{aligned} \tag{13}$$

2.4. Modified Window Follower Algorithm (MWFA)

Search range is also a critical factor which influences the computational complexity of motion estimation. Small search range results in poor matching results while large search range produces higher computational load. A suitable search range can reduce the computation complexity and also maintain good coding performance. Window Follower Algorithm (WFA) [7] is proposed to adaptively adjust search range based on the following assumptions:

(1) The change of motion content between frames is gradual and not sudden.
(2) The motion content is constant over a large number of successive frames.

WFA takes the maximum displacement of MV in previous frame plus one unit as the search range for the current frame. The algorithm is presented in Table 2.

However, the characteristics of motion in natural video sequences vary a lot and is hardly predictable. The assumptions of WFA may not be true in natural video sequences. MWFA [8] modifies WFA by exploiting both temporal and spatial information and adopting SAD as a measure of accuracy of MV. MWFA algorithm is presented in Table 3.

$\mathrm{SAD}_{\mathrm{min}t-1}$ and d_{t-1} represent the minimum SAD and the maximum MV displacement for the $(t-1)$th block in the current frame, respectively. The flag F is set to zero at the beginning of each frame. When the flag F is set to zero, only temporal information is considered; when the flag F is set to one, both temporal and spatial information are taken into account. According to the experimental results from simulations of typical video sequences, the threshold TH_1 and TH_2 are set to 4096 and 2048, respectively.

Table 2
Window Follower Algorithm

Step 1: For the kth frame, compute the maximum horizontal and vertical displacement from all MVs in $(k-1)$th frame. The maximum value D is defined as Eq. (14). The d_t represents the maximum displacement of two components of MV of tth block

$$D = \max[d_t] \tag{14}$$

$$d_t = \max\left\lfloor \mathrm{MVt}_x, \mathrm{MVt}_y \right\rfloor \tag{15}$$

Step 2: Perform motion estimation for kth frame with search range $P = D + 1$. For the first frame, the search range P is set to the default max search range defined in sequence parameter set

Table 3
Modified Window Follower Algorithm

Step 1: For the kth frame, compute the value D as defined in WFA
Step 2: Perform motion estimation for each block in kth frame with search range P_t, for tth block. P_t is determined by the following mutually exclusive rules

(1)	If $(\mathrm{SAD}_{\mathrm{min}t-1} >= \mathrm{TH}_1)$	$P_t = P_{\max}, F = 1$
(2)	If $(\mathrm{SAD}_{\mathrm{min}t-1} <= \mathrm{TH}_1$ and $F == 1)$	$P_t = \max(D, d_{t-1}) + 1$
	If $(\mathrm{SAD}_{\mathrm{min}t-1} <= \mathrm{TH}_1$ and $F == 0)$	$P_t = D + 1$
(3)	If $(\mathrm{SAD}_{\mathrm{min}t-1} <= \mathrm{TH}_2$ and $F == 1)$	$P_t = \max(D, d_{t-1})$
	If $(\mathrm{SAD}_{\mathrm{min}t-1} <= \mathrm{TH}_2$ and $F == 0)$	$P_t = D$

3. Content-Aware Fast Motion Estimation Algorithm

In this paper, in order to reduce the computational complexity of motion estimation in .264/AVC, we analyze the correlations between search range and the motion activity of the video content and the correlations of the motion vectors between neighboring blocks. Based on these observations, these correlations are fully considered in the development of the Content-Aware Fast Motion Estimation Algorithm (CAFME). We first present some observations and analyses of search range in motion estimation in Section 3.1. Then, the details of the proposed algorithms, Simple Dynamic Search Range Algorithm (SDSR), SEA with integral frame (SEAIF) and Early Termination Algorithm (ETA) are described in Sections 3.2–3.4, respectively.

3.1. Analysis of search range

True MV is defined as the displacement of current block from the matched block in the reference picture with minimum SAD value. Search range constraints current block to search the best matched block within a predefined area in the reference picture. Therefore, exploring the effect of the coding parameters upon the search range in motion estimation helps to adaptively determine the search range for each coding block to retrieve true MV in a suitable search range. Some experiments have been made to observe and analyze the relationships between search range (SR) and frame rate, frame resolution, motion activity, quantization parameter (QP), and SAD of best matched block. The results and discussion of each experiment are introduced in the following subsections. The experimental environment is shown in Table 4.

3.1.1. Search range and frame rate

Since the frame rate affects the difference of successive frames, so we observe the relationship between SR and frame rate. The test data are foreman sequence with FPS = 30 and 15. The temporal distance of sequence with FPS = 30 and 15 are 1/30 and 1/15 s, respectively. In theory, when the frame rate is higher, the motion estimation needs smaller search range.

Table 4
Experimental environment for analysis of the correlations between search range and coding parameters

Encoder environment
Software: JM 9.4 [14]
RDO: Enabled
Number of reference frame: 1
Quantization parameter (QP): 36
GOP size: 15
Macroblock adaptive inter-layer prediction: Enabled
Machine: Athlon XP 1700+ with 512 MB memory
Profile: baseline
Prediction structure: IPPP
Fast ME (UMHexagonS) [15]: disable
Fast mode selection [16]: disable

The quantization parameter (QP) is mapped into quantization step and affects the bitrate significantly. In our experiments, the QP is fixed and Rate Control (RC) is disabled. Therefore, we only need to observe the bitrate for different search ranges. In Table 5, the gray areas indicate that the bitrates are stable, which means the search ranges are sufficient for most MBs to be coded with true MVs which mininimizes SAD value in motion estimation. We can observe that the bitrates are stable when search range (SR) is larger or equal to 4 with FPS = 30 and search range (SR) is larger than or equal to 8 with FPS = 15. It indicates that the sufficient search range (SR) is proportional to frame rate.

3.1.2. Search range and frame resolution

To find the relationship between search range and frame resolution, we run the testing sequence in QCIF and CIF resolution. The experimental results of coastguard sequence is shown in Table 6. The gray areas in QCIF resolution show that the bitrate is stable when the SR varies from 2 to 32. The gray areas in CIF resolution indicate the bitrate is stable when the SR varies from 4 to 32. These observations indicate that the search range equal to 2 is sufficient to find the true MVs in QCIF resolution and search range equal to 4 is sufficient to find the true MVs in CIF resolution. The search range is proportional to resolution, which means the search range should be adjusted adaptively based on the frame resolution.

Table 6
The relation between SR and resolution

SR (pixel)	Coastguard, QP = 36, FPS = 30, encoded frames = 90			
	QCIF (176 × 144)		CIF (352 × 288)	
	SNR (dB)	Bitrate (Kbps)	SNR (dB)	Bitrate (Kbps)
32	29.178	71.992	29.315	375.003
16	29.183	72.304	29.312	375.539
8	29.185	72.325	29.299	374.728
4	29.168	71.747	29.289	376.667
2	29.144	72.520	29.271	387.957
1	29.107	76.120	29.225	420.099
0	28.799	113.893	28.920	639.923

3.1.3. Search range and motion activity

To see the impact of motion activity on search range, we divide the foreman sequence into two sequences, in which one is of low motion and the other one is of high motion. The first sequence is clipped from the first 90 frames and the second one is clipped from frame 151 to 240. In Table 7, it is observed that the bitrates are approximately stable when the SR $\geqq$ 4 in low-motion sequence and the SR $\geqq$ 8 in high-motion sequence, which indicates the higher search range is needed for high-motion sequences to derive true motion vectors.

3.1.4. Search range, QP, and SAD of best matched block

In block matching, SAD is used as the matching criterion. In this experiment, we observe the impact of search range (SR) and quantization parameter (QP) on SAD value. As shown in Table 8, the experimental result shows the true MVs can be obtained as long as search range is larger or equal to 8 while QP only affects the magnitude of SAD. Fig. 3 depicts SAD average from frame 0 to 299 of Foreman sequence. In this figure, the vertical axis is the value of SAD average and the horizontal axis is frame index. And, the dotted line and solid line represent the average SAD value over frames in Forman sequence with SR = 4 and 32, respectively. We can see that the value of SAD average with SR = 4 and 32 are almost the same from frame 0 to 299 except for those frames from 150 to 220. The curve of SR = 4 is above the curve of SR = 32 from frame 150 to frame 220. This phenomenon is resulted from

Table 5
The relation between SR and FPS

SR (pixel)	Foreman, QCIF (176 × 144), QP = 36			
	FPS = 30, 100 frames		FPS = 15, 50 frames	
	SNR (dB)	Bitrate (Kbps)	SNR (dB)	Bitrate (Kbps)
32	31.478	69.528	31.582	46.851
16	31.469	69.365	31.561	46.640
8	31.448	69.400	31.561	46.733
4	31.443	69.224	31.555	47.187
2	31.424	69.771	31.495	48.627
1	31.406	71.155	31.465	50.547
0	31.227	79.122	30.077	60.005

Table 7
The relation between SR and motion activity

SR (pixel)	Foreman QCIF QP = 36 FPS = 30			
	Frame 0–89 (low motion)		Frame 151–240 (high motion)	
	SNR (dB)	Bitrate (Kbps)	SNR (dB)	Bitrate (Kbps)
32	31.478	69.528	31.245	84.747
16	31.469	69.365	31.242	84.949
8	31.448	69.400	31.223	85.307
4	31.443	69.224	31.166	89.931
2	31.424	69.771	31.024	112.832
1	31.406	71.155	30.954	130.109
0	31.227	79.122	30.633	185.883

Table 8
The relation between SR, QP, and SAD

SR (pixel)	Foreman QCIF 300 frames FPS = 30			
	SAD average			
	QP = 18	QP = 24	QP = 30	QP = 36
32	921.2	1024.0	1221.3	1577.5
16	924.7	1027.8	1226.0	1584.9
8	942.9	1045.6	1244.6	1605.3
4	1068.8	1165.5	1353.8	1702.9
2	1252.9	1344.9	1524.4	1860.9
1	1413.7	1503.7	1585.9	2001.9
0	1831.9	1911.4	2081.7	2330.1

the fact that the motion is higher than the rest of the sequence from frame 150 to 220. Since true MV always results in minimum SAD value, it means SR $\leqq 4$ is not sufficient to find the true MVs for those frames from 150 to 220, thus results in larger SAD value.

In summary, an appropriate SR can reduce unnecessary computation in motion estimation and still can find out true MVs. In our experiments, the search range should be adjusted adaptively according to motion activity of video and parameters of encoder. Hence, we propose a mechanism Simple Dynamic Search Range (SDSR) to adjust SR dynamically based on motion activity.

3.2. Simple Dynamic Search Range (SDSR)

In order to adaptively adjust search range for motion estimation, some approaches (DSWA [5], AFSBM [6], MWFA [8], and MAS [9]) have already been implemented. According to different measure criteria, these approaches could be classified into block-matching error-based and motion vector-based approaches.

The block-matching error, which represents the degree of matching between the current block and the candidate block, is usually measured in Mean of Squared Difference (MSD), Mean of Absolute Difference (MAD) or Summation of Absolute Difference (SAD). The value of block-matching error is determined considering many factors including motion activity, texture, and quantization parameter. See Fig. 4 for example. From frame 220, the values of SAD are much higher than the rest frames. The reason is the complicated video texture, not the motion activity. However, the values of SAD in frames from 150 to 170 rise sharply due to the sudden motion change instead of video texture. Consequently, the approaches based on block-matching error are usually unsuitable to evaluate the motion activity.

On the contrary, motion vector could represent the motion activity more precisely [9]. Since MV is the displacement between current block and the best matched block within the search range in reference picture, the magnitude of MV must be less or equal to that of search range. The relation between SR and MV can be model by Eq. (16).

$$\mathrm{SR} = \max\left[\mathrm{MV}_x, \mathrm{MV}_y\right] + D \tag{16}$$

D is an offset between the maximum component of MV and search range (SR) and it also can be viewed as a magnitude to measure the extra search points processed in motion estimation. Ideally, $D = 0$ represents that SR is set equal to the maximum component of MV, which means no extra search points are processed. An adaptive SR scheme is to set SR as close to MV as possible by minimizing D for each coding block. However, the MV of current block is unknown before the SR is determined. Therefore, in the proposed

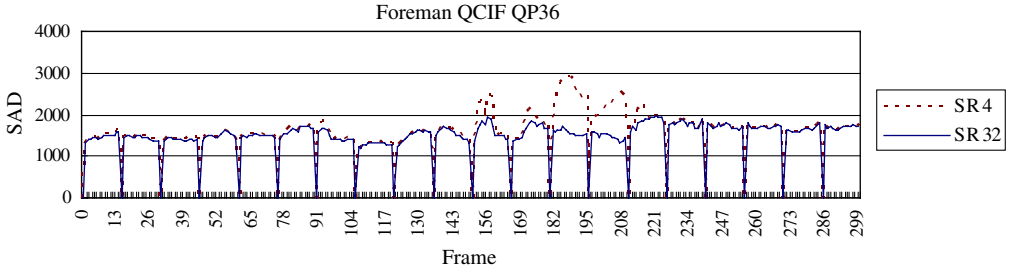

Fig. 3. Motion activity in foreman QCIF frame by frame with respective to different search range.

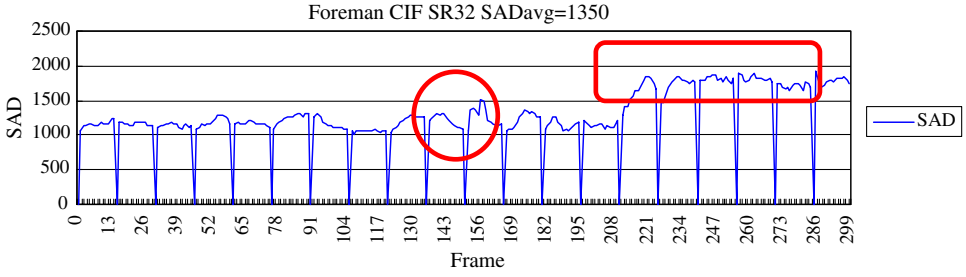

Fig. 4. SAD of foreman CIF frame by frame.

Simple Dynamic Search Range (SDSR) scheme, we try to set the SR according to the MVs of neighboring blocks because the MV of current block and those of neighboring blocks are highly correlated.

The proposed SDSR algorithm is described in Table 9. Due to the wide variations of motion activity in video sequences and different motion activity in various areas within a single frame, we like to adjust search range based on both temporal correlation and spatial correlation of motion field, respectively. The proposed SDSR scheme first uses the maximum component of MV in previous frame plus an offset γ as an upper bound for search range prediction. It can be described by Eq. (17), in which SR_FRAME_k represents search range in frame_level and $\max[\text{MV}_{xt}, \text{MV}_{yt}]$ represents the maximum horizontal and vertical displacement among all motion vectors in previous frame. In this equation, γ is used to enlarge the upper bound for search range prediction to avoid the bad matchings caused by small search range.

$$\begin{aligned} &\text{SR\_FRAME}_k = \max\left[\text{MV}_{xt}, \text{MV}_{yt}\right] + \gamma, \gamma \geqslant 0 \\ &t \in \{\text{all blocks in}(k-1)\text{th frame}\} \end{aligned} \tag{17}$$

Then the maximum displacement components of neighboring MVs of *current* block t, denoted as MV_MAX_t, is used as a lower bound for search range prediction.

Finally, the search range for each block is adjusted between the prediction upper bound, SR_FRAME_k, and the prediction lower bound,MV_MAX_t. As described in Eq. (18), once the MV_MAX_t are found to be larger than SR_FRAME_k, the final search range for current block, SR_BLOCK_t, will be set to MV_MAX_t plus an offset β which acts like γ in Eq. (17)

$$\text{SR\_BLOCK}_t = \text{MV\_MAX}_t + \beta, \beta \geqslant 0 \tag{18}$$

$$\text{SR\_BLOCK}_t = \alpha \cdot \text{MV\_MAX}_t + (1-\alpha) \cdot \text{SR\_FRAME}_k, \quad 0 \leqslant \alpha \leqslant 1 \tag{19}$$

Otherwise, the final search range will be calculated by Eq. (19). The control parameter α is used to adjust the weight of MV_MAX_t and SR_FRAME_k to calculate the final search range SR_BLOCK_t for each block.

Table 9
Simple Dynamic Search Range Algorithm

Step 1: Determine the search range in frame level. The search range called *SR_FRAME*$_k$ is computed by the maximum horizontal and vertical displacement from all MVs in $(k-1)$th frame plus γ. The definition is

$\text{SR\_FRAME}_k = \max\left[\text{MV}_{xt}, \text{MV}_{yt}\right] + \gamma, \quad \gamma \geqslant 0$

$t \in \{\text{all blocks in } (k-1)\text{th frame}\}$

Step 2: Adjust the search range in macroblock level. Let MV_MAX_t denote the maximum displacement of two components of MVs in neighbor blocks of tth block, as described in the following rules

$s \in \{\text{The left, above left, above, above right blocks of } t\text{th block}\}$

If any of neighbor blocks is not available

$\text{MV\_MAX}_t = \max[\max[\text{MV}_{xs}, \text{MVy}_s], \text{SR\_it FRAME}_k]$

Else

$\text{MV\_MAX}_t = \max[\text{MV}_{xs}, \text{MVy}_s]$

Step 3: Determine the final search range for tth block, called SR_BLOCK_t by the following rules

//Adjust SR in block level

If($\text{MV\_MAX}_t \geqslant \text{SR\_FRAME}_k$)

$\text{SR\_BLOCK}_t = \text{MV\_MAX}_t + \beta, \beta \geqslant 0$

Else

$\text{SR\_BLOCK}_t = \alpha \cdot \text{MV\_MAX}_t + (1-\alpha) \cdot \text{SR\_FRAME}_k, \quad 0 \leqslant \alpha \leqslant 1$

//SR constraint

If ($\text{SR\_BLOCK}_t \leqslant 1$)

$\text{SR\_BLOCK}_t = 1$

Else if ($\text{SR\_BLOCK}_t \geqslant$ max search range)

$\text{SR\_BLOCK}_t =$ max search range

3.3. Successive Elimination Algorithm with Integral Frame (SEAIF)

In H.264/AVC standard, the partition modes of each macroblock in motion estimation include nine intra-modes and seven inter-modes (see Fig. 5). In inter-coding, 41 motion estimations are required for a 16×16 macroblock while rate-distortion optimization (RDO) is enabled for mode selection (one for 16×16, two for 16×8, two for 8×16, four for 8×8, eight for 8×4, eight for 4×8, and 16 for 4×4). Due to the support of various partition modes, the ME cost in H.264/AVC increases dramatically compared to previous video coding standards. Therefore, it is essential to develop efficient algorithm to speed up the computation of ME.

In the H.264/AVC reference software JM 9.4 [14], in order to reduce the intensive computation caused by RDO, a Fast Full Pel Search algorithm is implemented by reusing SAD values of the smallest 4×4 block. At the beginning of the motion estimation of each macroblock, it first computes the SAD values for all 4×4 block at all search points within the search window. After that, it merges the SAD values to get the SAD values of larger blocks. In this way, computation of SAD for a macroblock with all block size enabled is about equal to the computation of SAD with only a 16×16 block.

We adopt the concept of reusing SAD and integrate it into our proposed algorithm. We integrate SEA and integral frame technique introduced in Sections 2.2 and 2.3 to form a new SEA called SEAIF for H.264/ACV standard. The main idea of the SEAIF for H.264/AVC is to reuse sea values and SAD values. The following sub-sections present the details of the design. Sections 3.3.1 and 3.3.2 present the techniques of reusing sea and SAD values. Finally, analysis of complexity for SEAIF is presented in Section 3.3.3.

3.3.1. Reusing of sea value

For each search point, calculate the sea values of 16 4×4 blocks of the current macroblock by using integral frame technique. These sea values of 4×4 blocks are the basis for sea values of larger blocks. Then the sea values of larger blocks are derived from thesesea values of 4×4 blocks, described as follows:

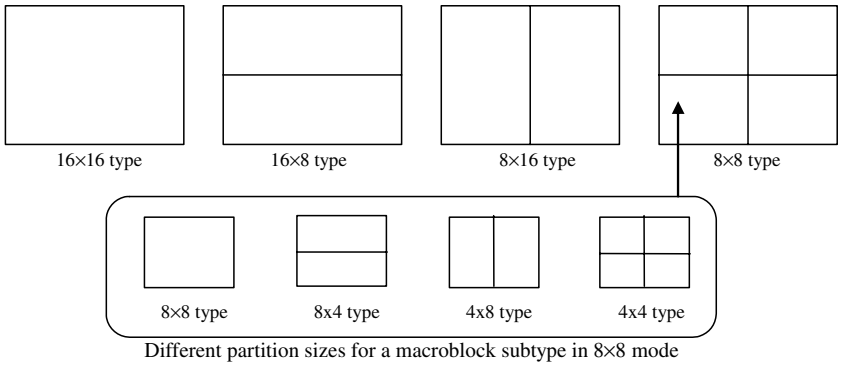

Fig. 5. Different partition sizes in a macroblock.

- For 8×4 or 4×8 block, sum up sea values of two 4×4 blocks.
- For 8×8 block, sum up sea values of two 8×4 blocks.
- For 16×8 or 8×16 block, sum up sea values of two 8×8 blocks.
- For 16×16 block, sum up sea values of two 16×8 blocks.

In this way, we can get all the sea values of all blocks of different partitions. These sea values of larger blocks are always equal to or larger than the sea values computed directly from block sums (BS) of corresponding blocks. Therefore, the sea values of larger blocks derived from 4×4 block sea values are lower bound of SAD and thus more computations of SAD can be skipped.

3.3.2. Reusing of SAD value

In SEAIF, if the sea value is less than the current minimum SAD value, complete calculation of SAD will be preformed. In H.264/AVC, overlapped blocks are used in motion estimation. In order to reduce the computations of SAD, we take the 4×4 block SAD values as the basis of the larger block SAD values. In this way, there is no redundant computation of SAD. The proposed approach is described in Table 10.

3.3.3. Analysis of complexity

The reason of adopting SEA is to reduce the computational cost in block matching. The overhead of SEA should be considered and analyzed. The overheads of SEA are mainly the computation of block sums. In SEA [3], Salari and Li proposed a fast algorithm to compute the block sums. The conventional approach, SEA approach, and Integral frame approach are compared and the analysis of the overhead of each approach is described as follows:

Let W denote image width, H image height, M block width, and N block height. Operations required for block sums of all $M \times N$ blocks in a reference frame for the approaches are as follows:

- *Straightforward approach:*
  Number of block sum in a frame: $(W - M + 1)(H - N + 1)$
  Operations required for a block sum: $MN - 1$
  Total cost: $(MN - 1)(W - M + 1)(H - N + 1)$
  Approximate cost: $MNWH$
- *SEA approach in [3]:*
  Total cost: $4WH - (H - N)(M + 3) - 3W(N + 1)$
  Approximate cost: $4WH$
- *Integral frame approach:*
  Operations required for an integral frame: $2WH$
  Operations required for all block sum: $\approx 2(W - M + 1)(H - N + 1)$[1]
  Total cost: $2WH + 2(W - M + 1)(H - N + 1)$
  Approximate cost: $4WH$

Although Integral frame approach and the SEA approach in [3] have approximately the same complexity, there is an advantage in Integral frame approach. In Integral frame approach, it is flexible to get block sum of any rectangle block.

For example, if we want to use the multilevel SEA for each block size in H.264/AVC, it will be easier to implement with integral frame approach (Note that our approach uses the tighter lower bound in SEA, not multilevel SEA). Computing msea value of 16×16 block with level $L = 0$ only needs five operations including three for getting BS, one for subtraction operation and one for absolute operation. Nevertheless, merging sixteen 4×4 sea values to get the sea value of 16×16 block with level $L = 0$ needs 15 addition operations while the sea value is a tighter lower bound. There

Table 10
Reusing SAD value algorithm

Regardless of block size, Calculation of SAD for the block is:
Step1: Find out all 4×4 blocks within the block
Step2: Check the SAD values of these 4×4 blocks. If any SAD value of 4×4 blocks is not available, compute the SAD value
Step3: Get the SAD value of the target block by adding up SAD values of these 4×4 blocks

[1] In [11], Viet Anh Nguyen and Yap-Pen Tan proposed a fast approach to calculate block sum by exploiting the adjacent property of the blocks.

is trade-off between the tighter lower bound and computational complexity.

3.4. Early Termination Algorithm (ETA)

In the H.264/AVC encoder, the most time-consuming component is variable block-size motion estimation. To reduce the complexity of motion estimation, we propose an Early Termination Algorithm (ETA) to predict the best motion vector by exploiting the correlation between the MVs of the current block and the neighboring blocks. With the proposed method, some of the search points can be discarded early to speed up the process of motion estimation.

Siou-Shen Lin et al. [10] show that the probability is about 79% in average when the variance of the current block and neighbor blocks is smaller than 3. They consider that it is of high probability that the variance of the motion vectors in the neighbor blocks is small, which means the difference between the MVs of current block and those of neighboring blocks might be small.

We exploit and modify the variance of motion vectors proposed in [10] to classify the motion activity of current block and neighbor blocks into simple motion and complex motion. The variance of motion vectors is defined in Eq. (21).

$$\mathrm{MVmean} = (\mathrm{MVa} + \mathrm{MVb} + \mathrm{MVc} + \mathrm{MVd})/4 \tag{20}$$

$$\mathrm{MVvar} = |\mathrm{MVa} - \mathrm{MVmean}| + |\mathrm{MVb} - \mathrm{MVmean}| + |\mathrm{MVc} - \mathrm{MVmean}| + |\mathrm{MVd} - \mathrm{MVmean}| \tag{21}$$

If any of the neighbor blocks is not available, $\mathrm{MV}_{\mathrm{var}}$ is set to a large value (999,999). As shown in Eq. (22), the threshold κ is used to classify each block into simple_motion or complex_motion by its MV variance. According to the experimental results, it is found that setting κ to "5" can bring more computation time saving while maintaining good coding performance.

$$\begin{aligned} &\mathrm{If}(\mathrm{MVvar} \leqq \kappa) \\ &\quad \mathrm{Mactivity} = \mathrm{simple\_motion} \\ &\mathrm{Else} \\ &\quad \mathrm{Mactivity} = \mathrm{complex\_motion} \end{aligned} \tag{22}$$

Once classified as simple motion, the SAD value of the block should be similar to those of neighboring blocks. On the contrary, the SAD values of blocks which are classified as complex_motion should be quite different from those of neighboring blocks. Based on this concept, the lower bound for the condition of termination is determined in Eq. (23).

$$\begin{aligned} &\mathrm{If}(\mathrm{Mactivity} = \mathrm{simple\_motion}) \\ &\quad \mathrm{SAD\_threshold} = \mathrm{SAD\_prediction} \\ &\mathrm{Else} \\ &\quad \mathrm{SAD\_threshold} = \mathrm{SAD\_prediction} \\ &\quad -\mathrm{SAD\_standard\_deviation} \end{aligned} \tag{23}$$

The SAD_prediction and SAD_standard_deviation represent the prediction of SAD of current block and the standard deviation of SAD of all blocks in the previous frame, respectively. The definitions are defined in Eqs. (24) and (25):

$$\mathrm{SAD\_prediction} = (\mathrm{SADa} + \mathrm{SADb} + \mathrm{SADc} + \mathrm{SADd})/4 \tag{24}$$

$$\mathrm{SAD\_mean} = \frac{1}{\mathrm{Number\_MB}} \sum_{t=0}^{\mathrm{Number\_MB-1}} \mathrm{SAD}_t \tag{25}$$

$$\mathrm{SAD\_standard\_deviation} = \left(\frac{1}{M-1} \sum_{t=0}^{M-1} (\mathrm{SAD}_t - \mathrm{SAD\_mean})^2 \right)^{1/2} \tag{26}$$

The SAD_t is the SAD value of tth block in a frame. Number_MB is the total number of MB in a frame. If there is no any neighbor block near the current block, SAD_prediction is set to a small value (−999,999). Note that the SAD_prediction and SAD_standard_deviation are calculated for 16×16 macroblock. In H.264/AVC standard, there are seven block sizes used in motion estimation. We determine the SAD_prediction and SAD_standard_deviation for other block size according to the area occupied by the block.

Finally, the condition of termination is tested when a new up-to-date best matched block is found. If the SAD value of the up-to-date block is equal to or smaller than SAD_threshold, the motion estimation is terminated.

4. Experimental results and discussions

In this section, we present the experimental results of the proposed approaches. We modify the H.264/AVC reference software JM 9.4 and implement the proposed algorithms on it. In the experiments, we observe the number of search points for each block to measure the performance of the proposed algorithms. We also measure the coding efficiency. In order to measure the coding efficiency, we compare the bitrates of encoded sequences with the same quantization parameter and disabling rate control. Besides, we exploit the SAD value as a criterion to measure whether the determined search range is large enough. Finally, we compare the total encoding time to measure the improvement in practical situation.

4.1. Experimental environment

Nine testing video sequences are taken into concern in the experiments. As shown in Table 12, these testing sequences includes video data of various resolutions and different motion activities. The experimental environment and some coding configurations are listed in Table 11. These parameters are applied to all experiments except for some circumstances which will be addressed later. Note that the maximum search range is set to 24.

Table 11
Testing conditions

Encoder configurations
Software: JM 9.4 [14]
ME search range: ±24 pixels
RDO: enabled
Number of reference frame: 1
Quantization parameter (QP): 36
GOP size: 15
Macroblock *Adaptive Inter-Layer Prediction*: enabled
Machine: Athlon XP 1700+ with 512 MB memory
Profile: baseline
Prediction structure: IPPP
Fast ME (UMHexagonS) [15]: disable
Fast mode selection [16]: disable

Table 12
Descriptions of test video sequences

ID	Name	Resolution	No. of frames	Motion activity
A	Foreman	QCIF	150	Medium
B	Mobile	QCIF	150	Slow
C	Coastguard	QCIF	150	Medium
D	Foreman	CIF	150	Medium
E	Tempete	CIF	150	Slow, zooming
F	Flower	CIF	90	Slow
G	Stefan	SIF	150	High
H	Football	CIF	90	Very high
I	Table tennis	SIF	90	Medium, scene change

Table 13
Search points of FFS and SDSR

Sequence name	Number of search points		Improvement (%)
	Fast Full Pel Search	SDSR	
Foreman QCIF	2401	144	−94
Mobile QCIF	2401	52	−98
Coastguard QCIF	2401	88	−96
Foreman CIF	2401	365	−85
Tempete CIF	2401	261	−89
Flower CIF	2401	563	−77
Stefan SIF	2401	860	−64
Football CIF	2401	1411	−41
Table tennis SIF	2401	497	−79
Average			−80

4.2. Fast full Pel Search

The proposed algorithms are compared with Fast Full Pel Search[2] which is an improved version of conventional Full Pel Search by reusing SAD values of the smallest 4×4 block. Fast Full Pel Search computes the SAD values for all 4×4 blocks in advance whenever a new macroblock begins the motion estimation. Then, it merges the SAD values to get the SAD values of larger blocks. In this way, computation of SAD for a macroblock with all block size enabled is about equal to the computation of SAD with only a 16×16 block.

Note that the performances of the Fast Full Pel Search and the conventional Full Search are the same but the Fast Full Pel Search is faster than the conventional Full Search in H.264/AVC. In the following context, we denote the Fast Full Pel Search as *FFS*.

4.3. Simple Dynamic Search Range

The experimental results of the proposed Simple Dynamic Search Range (SDSR) algorithm in Table 13 shows that the proposed SDSR algorithm outperforms the Fast Full Pel Search (FFS) greatly. Compared to FFS, SDSR reduces the number of search points to 80% in average. For the testing sequences of low and medium motions, SDSR could even reduces the number of search points over 90%. Table 14 shows the comparative results of coding bitrates of the testing sequences. We can observe that the bitrates increases slightly for each testing sequence. In Table 15 the total encoding time is reduced about 40–50%. The motion activity of Stefan and Football sequences are higher than others.

To evidence that SDSR could adaptively determine the search range in a reasonable size, we depict the search range (SR) determined in each frame as shown in Figs. 6

Table 14
Bitrates of FFS and SDSR

Sequence name	Bitrates (Kbps)		Improvement (%)
	Fast Full Pel Search	SDSR	
Foreman QCIF	69.203	68.858	−0.5
Mobile QCIF	173.016	173.250	+0.1
Coastguard QCIF	76.134	76.022	−0.1
Foreman CIF	188.773	188.490	−0.1
Tempete CIF	425.392	425.810	+0.1
Flower CIF	669.312	669.333	+0.003
Stefan SIF	505.450	505.693	+0.05
Football CIF	413.301	416.525	+0.8
Table tennis SIF	256.259	257.925	+0.65
Average			+ 0.11

Table 15
Total encoding time of FFS and SDSR

Sequence name	Total encoding time (s)		Improvement (%)
	Fast Full Pel Search	SDSR	
Foreman QCIF	156	74	−53
Mobile QCIF	151	75	−50
Coastguard QCIF	151	70	−54
Foreman CIF	602	319	−47
Tempete CIF	583	324	−44
Flower CIF	374	221	−41
Stefan SIF	508	340	−33
Football CIF	363	280	−23
Table tennis SIF	298	169	−43
Average			−43

[2] The Fast Full Pel Search is implemented in H.264/AVC Reference Software JM 9.4.

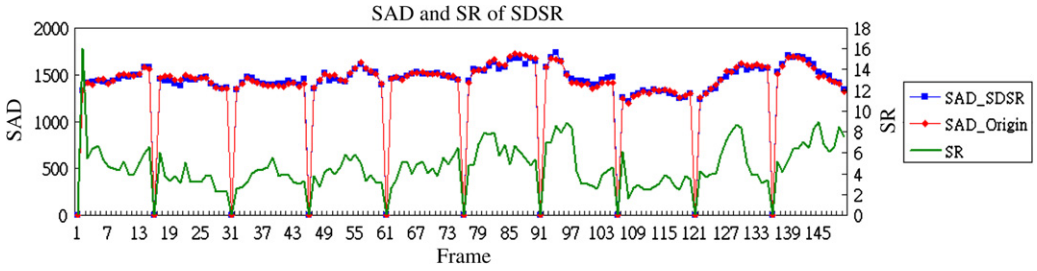

Fig. 6. SAD and SR of SDSR frame-by-frame in Foreman QCIF.

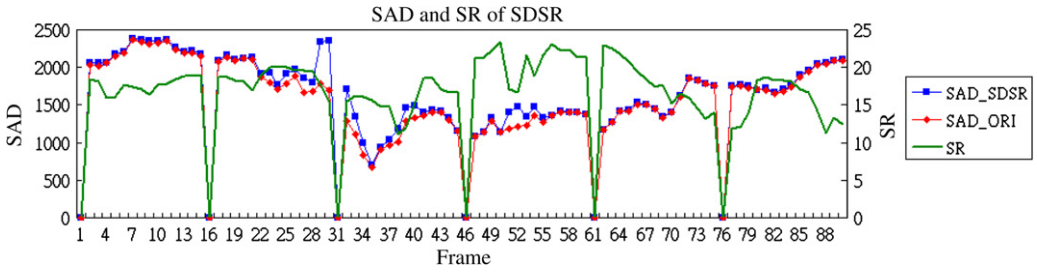

Fig. 7. SAD and SR of SDSR frame-by-frame in Football CIF.

and 7. In Figs. 6 and 7, we can also see the SAD values between the original frame and each reconstructed frame using the SDSR and the conventional method. It can be observed that the SAD values of SDSR and FFS are very close which means SDSR can find true MVs in most of the motion estimations except for the frames of higher motion activities. In average, the number of search points is reduced about 80%, bitrate increases about 0.11%, and total encoding time is reduced about 43%. Therefore, we could claim the proposed SDSR can reduce the coding complexity while maintaining almost the same coding efficiency.

4.4. Successive Elimination Algorithm with Integral Frame

The proposed Successive Elimination Algorithm with Integral Frame (SEAIF) is designed to reduces the number of search points in the process of motion estimation. Tables 16 and 17 show the experimental results of comparison of SEAIF and Fast Full Pel Search (FFS). In Table 16, we can see the search points using SEAIF is in average 95% less than those using FFS under the constraint that only 16×16 block size is enabled for motion estimation. Therefore, the encoding time of SEAIF is about 29% less than that of FFS as shown in Table 17.

Table 17
Total encoding time of FFS and SEAIF (16×16 block size only)

Sequence name	Total encoding time (s)		Improvement (%)
	Fast Full Pel Search	SEAIF	
Foreman QCIF	112	77	−31
Mobile QCIF	117	84	−28
Tempete CIF	458	332	−28
Stefan SIF	369	289	−27
Average			−29

4.5. Early Termination Algorithm

When reaching desired search point, the proposed Early Termination Algorithm (ETA) could early terminate the process of motion estimation to reduce the number of SP. As shown in Tables 18 and 19, about 44.5% SP are saving and the bitrate is nearly the same with the bit rate produced by FFS. However, Table 20 shows that the encoding time is not reduced as we expect. In the process of motion

Table 16
Search Points of FFS and SEAIF (16×16 block size only)

Sequence name	Number of search points		Improvement (%)
	Fast Full Pel Search	SEAIF	
Foreman QCIF	2401	61	−97
Mobile QCIF	2401	71	−97
Tempete CIF	2401	114	−95
Stefan SIF	2401	193	−92
Average			−95

Table 18
Search points of FFS and ETA

Sequence name	Number of search points		Improvement (%)
	Fast Full Pel Search	ETA	
Foreman QCIF	2401	1484	−38
Mobile QCIF	2401	1197	−50
Tempete CIF	2401	1306	−46
Stefan SIF	2401	1350	−44
Average			−44.5

Table 19
Bitrates of FFS and ETA

Sequence name	Bitrates (Kbps)		Improvement (%)
	Fast Full Pel Search	ETA	
Foreman QCIF	69.203	69.365	+0.2
Mobile QCIF	173.016	173.366	+0.2
Tempete CIF	425.392	424.898	−0.1
Stefan SIF	505.450	505.987	+0.1
Average			+0.1

Table 20
Total encoding time of FFS and ETA

Sequence name	Total encoding time (s)		Improvement (%)
	Fast Full Pel Search	ETA	
Foreman QCIF	156	140	−10.3
Mobile QCIF	151	152	+0.7
Tempete CIF	583	594	+1.9
Stefan SIF	508	498	−2.0
Average			−2.4

estimation, each search point is estimated in matching criterion, usually SAD. Although our ETA can skip a large number of search points, it cannot save the computations of SAD because FFS in JM9.4 calculates all SAD values in advance. Thus the encoding time can not be saved in this experiment.

4.6. Content-Aware Fast Motion Estimation Algorithm (CAFME)

The Content-Aware Fast Motion Estimation Algorithm (CAFME) is formed by integrating the Simple Dynamic Search Range (SDSR), Successive Elimination Algorithm with Integral Frame (SEAIF), and Early Termination Algorithm (ETA). Here we evaluate the performance of the proposed CAFME and give some discussions.

4.6.1. Performance compared to Fast Full Pel Search (FFS)

Compared to Fast Full Pel Search (FFS), as shown in Table 21, the number of search points used by CAFME can be reduced more than 90% for most of the testing sequences. For the sequences of slow and median motion, the reduced rates of search points could even reach about 99%. The reduced rate of search points is much lower (73.8%) for football sequence because of the very high motion characteristic. In average, the increment of bitrate of the video stream coded by CAFME is about 0.26% (Table 22), which is a slight increment. Furthermore, the total encoding time is reduced about 41.9% as shown in Table 23.

Table 21
Search points of FFS and CAFME

Sequence name	Number of search points		Improvement (%)
	Fast Full Pel Search	CAFME	
Foreman QCIF	2401	37	−98.5
Mobile QCIF	2401	12	−99.5
Coastguard QCIF	2401	29	−98.8
Foreman CIF	2401	100	−95.8
Tempete CIF	2401	69	−97.1
Flower CIF	2401	199	−91.7
Stefan SIF	2401	184	−92.3
Football CIF	2401	628	−73.8
Table tennis SIF	2401	224	−90.7
Average			−93.1

4.6.2. Performance compared to UMHexagonS

To compare the performance of the proposed CAFME scheme and UMHexagonS, experiments are done to evaluate the motion estimation time of both schemes under different search range (SR = 24, 48, 96, and 128). Both schemes are implemented based on JM9.4 and three testing sequences including Forman, Tempete, and Flower are used in the experiments.

Tables 24–26 are the results of total motion estimation time between proposed Content-Aware Fast Motion Estimation Algorithm (CAFME) and UMHexagonS method. As shown in these tables, although the motion estimation time of proposed CAFME scheme is higher than that of UMHexagonS when search range is small, CAFME can reduce more computation time when search range becomes

Table 22
Bitrates of FFS and CAFME

Sequence name	Bitrates (Kbps)		Improvement (%)
	Fast Full Pel Search	CAFME	
Foreman QCIF	69.203	69.118	−0.12
Mobile QCIF	173.016	173.285	+0.16
Coastguard QCIF	76.134	75.862	−0.36
Foreman CIF	188.773	188.784	+0.005
Tempete CIF	425.392	425.955	+0.13
Flower CIF	669.312	670.211	+0.13
Stefan SIF	505.450	504.782	−0.13
Football CIF	413.301	419.357	+1.5
Table tennis SIF	256.259	258.939	+1.04
Average			+0.26

Table 23
Total encoding time of FFS and CAFME

Sequence name	Total Encoding Time (Second)		Improvement (%)
	Fast Full Pel Search	CAFME	
Foreman QCIF	156	69	−56
Mobile QCIF	151	77	−49
Coastguard QCIF	151	68	−55
Foreman CIF	602	314	−48
Tempete CIF	583	318	−45
Flower CIF	374	224	−40
Stefan SIF	508	324	−36
Football CIF	363	325	−10
Table tennis SIF	298	184	−38
Average			−41.9

Table 24
Motion estimation time of UMHexagonS and CAFME over different search range for sequence Foreman_CIF

Search range	Motion estimation time (s)		Improvement (%)
	UMHexagonS	CAFME	
24	47	70	+48
48	57	70	+23
96	78	76	−3
128	92	83	−10
Average			+14.5

Table 25
Motion estimation time of UMHexagonS and CAFME over different search range for sequence tempete CIF

Search range	Motion estimation time (s)		Improvement (%)
	UMHexagonS	CAFME	
24	51	52	+2
48	69	58	−15
96	98	66	−32
128	119	70	−41
Average			−24

Table 26
Motion estimation time of UMHexagonS and CAFME over different search range for sequence Flower CIF

Search range	Motion estimation time (s)		Improvement (%)
	UMHexagonS	CAFME	
24	43	47	+8
48	52	51	−1
96	76	58	−23
128	90	65	−27
Average			−11

larger. It is believed that the proposed CAFME is more efficient under high search range scenarios like high profiles. It is noted that the differences in PSNR and BitRate between the two schemes at all testing sequences are found to be negligible. For example, from Tables 27 and 28, we can see the difference of PSNR and BitRate are not larger than 0.1% for Forman sequence.

UMHexagonS scheme searches the best matched blocks following a predefined search pattern to speed up the searching process while the proposed CAFME is a hybrid scheme consisting of three approaches to speed motion estimation. Here we combine UMHexagonS with the proposed scheme and compare it to UMHexagonS to see if the gain of CAFME could be added on top of UMHexagonS.

Compared to UMHexagonS, as shown in Tables 29–31, the total motion estimation time of hybrid approach (combination of UMHexagonS and CAFME) can be further

Table 27
PSNR of UMHexagonS and CAFME over different search range for sequence Foreman_CIF

Search range	PSNR (dB)		Improvement (%)
	UMHexagonS	CAFME	
24	32.58	32.61	+0.1
48	32.58	32.61	+0.1
96	32.58	32.61	+0.1
128	32.58	32.61	+0.1

Table 28
Bitrate of UMHexagonS and CAFME over different search range for sequence Foreman_CIF

Search range	Bitrate (bps)		Improvement (%)
	UMHexagonS	CAFME	
24	194.45	194.75	+0.15
48	194.53	194.68	+0.07
96	194.56	194.71	+0.07
128	194.71	194.71	+0

Table 29
Motion estimation time of UMHexagonS and hybrid approach over different search range for sequence Foreman_CIF

Search range	Motion estimation time (s)		Improvement (%)
	UMHexagonS	UMHexagonS + CAFME	
24	47	40	−15
48	57	41	−28
96	78	45	−42
128	92	48	−48
Average			−33

Table 30
Motion estimation time of UMHexagonS and hybrid approach over different search range for sequence Tempete CIF

Search range	Motion estimation time (s)		Improvement (%)
	UMHexagonS	UMHexagonS + CAFME	
24	51	40	−21
48	69	41	−40
96	98	46	−53
128	119	49	−58
Average			−43

Table 31
Motion estimation time of UMHexagonS and hybrid approach over different search range for sequence Flower CIF

Search range	Motion estimation time (s)		Improvement (%)
	UMHexagonS	UMHexagonS + CAFME	
24	43	35	−18
48	52	35	−33
96	76	39	−49
128	90	43	−52
Average			−38

reduced, which means the gain of CAFME can be added on UMHexagonS for all testing sequences.

4.7. Summary

The proposed Simple Dynamic Search Range (SDSR) can reduce the number of search points about 80% while sustaining the coding efficiency (bitrate increases 0.11% in average). We also integrate the Successive Elimination Algorithm with Integral Frame (SEAIF) and the Early Termination Algorithm (ETA) with SDSR to form the Content-Aware Fast Motion Estimation Algorithm (CAFME). The CAFME improves the SDSR and the number of search points is reduced 93.1% while the bitrate increases just a little (0.26%). The overall encoding time is reduced about 41.9% in our implementation.

5. Conclusions and future works

The motion estimation plays an important role in the video coding standard. Also, it is usually the most computational-intensive part in a typical video encoder. Hence, the efficient motion estimation algorithm is essential. We proposed a fast algorithm called Content-Aware Fast Motion Estimation Algorithm (CAFME). CAFME consists of the Simple Dynamic Search Range (SDSR), Successive Elimination Algorithm with Integral Frame (SEAIF), and Early Termination Algorithm (ETA). The SDSR adjusts the search range for every block adaptively and does not need any predefined thresholds and performs well for all the test sequences. The SEAIF utilizes reusing techniques for calculating SAD of overlapped blocks of variable size thus can reduces the number of computation of SAD without loss of coding efficiency. The ETA terminates the search process early when finding a good candidate block and the performance is stable for all kinds of testing sequence of different motion activity.

The experimental results show that CAFME can reduce the number of search point about 93.1% and the bitrate only increases 0.26% while sustaining the same PSNR. We modified H.264/AVC reference software JM 9.4 and implemented our proposed algorithms on it. The total encoding time reduces about 41.9%.

The motion search algorithm currently used in CAFME is Fast Full Pel Search (FFS). However it may be replaced by any fast motion estimation algorithm like TSS and DS. The future works will focus on developing a fast motion estimation algorithm suitable for dynamic search range, alleviate the overhead in implementation, and so on.

References

[1] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, Motion Compensated Interframe Coding for Video Conferencing, in: Proc. Nat. Telecommun. Conf., New Orleans, LA, November 29–December 3 1981, pp. G5.3.1–5.3.5.

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

[3] W. Li, E. Salari, Successive elimination algorithm for motion estimation, IEEE Trans. Image Process. 4 (1) (1995) 105–107.

[4] X.Q. Gao, C.J. Duanmu, C.R. Zou, A multilevel successive elimination algorithm for block matching motion estimation, IEEE Trans. Image Process. 9 (3) (2000) 501–504.

[5] L.-W. Lee, J.-F. Wang, J.-Y. Lee, J.-D. Shie, Dynamic search-window adjustment and interlaced search for block-matching algorithm, IEEE Trans. Circuits Systems Video Technol. 3 (1) (1993) 85–87.

[6] J. Feng, K.-T. Lo, H. Mehrpour, A.E. Karbowiak, Adaptive block matching motion estimation algorithm for video coding, IEE Electron. Lett. 31 (18) (1995) 1542–1543.

[7] J. Minocha, N.-R. Shanbhag, A low power data-adaptive motion estimation algorithm, in: IEEE Third Workshop on Multimedia Signal Processing, September 13–15 1999, pp. 685–690.

[8] S. Saponara, L. Fanucci, Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems, IEE Proc. Comput. Digital Techniques 151 (1) (2004) 51–59.

[9] P.-I. Hosur, Motion adaptive search for fast motion estimation, IEEE Trans. Consumer Electron. 49 (4) (2003) 1330–1340.

[10] S.-S. Lin, P.-C. Tseng, C.-P. Lin, L.-G. Chen, Multi-mode content-aware motion estimation algorithm for power-aware video coding systems, in: IEEE Workshop on Signal Processing Systems, 13–15 October 2004, pp. 239–244.

[11] V.-A. Nguyen, Y.-P. Tan, Fast block-based motion estimation using integral frames, IEEE Signal Process. Lett. 11 (9) (2004) 744–747.

[12] K.-P. Lim, G. Sullivan, T. Wiegand, Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods ITU-T, Doc. #JVT-N046, January 2005.

[13] P. Viola, M.-J. Jones, Robust Real-Time Object Detection" Cambridge Res. Lab., Tech. Rep. CRL 2001/01, February 2001.

[14] H.264/AVC reference software, <http://ftp3.itu.ch/av-arch/jvt-site/reference_software/> and <http://iphome.hhi.de/suehring/tml/>.

[15] Z. Chen, P. Zhou, Y. He, Y. Chen, Fast Integer Pel and Fractional Pel Motion Estimation for JVT" ITU-T, Doc. #JVT-F017, December 2002.

[16] B. Jeon, J. Lee, Fast Mode Decision for H.264 ITU-T, Doc. #JVT-J033, December 2003.