A Computation-Aware Scheme for Motion Estimation in H.264

Fu-Chieh Chuang and Jin-Jang Leou

Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi, Taiwan 621, Republic of China
E-mail: jjleou@cs.ccu.edu.tw

Abstract

In this study, a computation-aware scheme for motion estimation (ME) in H.264 is proposed. The objective of the proposed scheme is to properly distribute the available ME computations of the H.264 encoder. Here, the temporal motion vector prediction technique is used to get the predicted motion vector (PMV) of each macroblock (MB). The sum of absolute components of the PMVs of all the MBs in a video frame is used as the measure to allocate the target computation to a video frame. The proposed scheme contains four phases: 1) frame-level computation allocation, 2) MB-level computation allocation, 3) mode-level computation allocation, and 4) pixel-level computation allocation. As compared with the comparison scheme, the proposed computation-aware schemes for motion estimation in H.264 usually have the better performance (average PSNR improvement and bit rate increment) in most simulation cases.

Keywords: H.264 video coding, computation-aware video coding, motion estimation.

1. Introduction

In general, the computational complexity of the H.264 encoder is higher than that of the H.264 decoder and the most time-consuming portion of the H.264 encoder is ME (motion estimation) [1]. Because fast search ME operations may consume over 80% of the computational power of the H.264 encoder, conventional fast ME algorithms [2-4] usually focus on reducing computations, i.e., reducing the number of checking points required for each MB or reducing the computation complexity of a checking point. However, if the total computations are not sufficient to finish the whole video coding (or ME) procedure, the encoding process may be forced to terminate suddenly.

Tai, et al. [5] proposed a computation-aware scheme, which contains two phases: (1) frame-level computation allocation and (2) block-level computation allocation. Within the frame-level phase, two parameters, C^i_{BP} and C^i_{FP} , stand for backward prediction and forward prediction, respectively, when the *i*th frame is handling. C^i_{BP} is the average of remaining computation power given to the (n-i+1) frames that are not coded, whereas C^i_{FP} is got from the relationship between the amount of computation power which has been used and the mean square error (MSE) gained by using a block matching algorithm (BMA). Based on the two parameters, C^i_{BP} and C^i_{FP}, a smooth section will be decided to give an acceptable range of computation power of the *i*th frame. Within the block-level phase, a so-called predicted computation-distortion benefit (PCDB) list is established, according to the order of initial mean square error (MSE) values at *mv*(0, 0) of all blocks in the current video frame. Once the PCDB list is initialized, their approach greedily allocates the computation power (in unit of checking points) to the first block on the PCDB list in order to perform the next step. The PCDB list is updated by the returned MSE value, which corresponds to the new motion vector found in the search step. This process is repeated until the target computation is exhausted.

In the frame level, the initial MSE of a frame should be calculated before the ME process can start. In the block level, all initial MSEs of all blocks in a frame should be got first to establish the PCDB list, and at each PCDB list updating, one more MSE value will be calculated and if the order of the PCDB list changes, the unfinished ME of the current block will be forced to stop its ME process. The initial MSE calculations of all blocks in a frame and switching the ME process due to changing of the PCDB list are some kind of redundancy. To overcome the weaknesses of the existing approach, in this study, a

$^+$ This work was supported in part by National Science Council and Ministry of Economic Affairs, Taiwan, Republic of China under Grants NSC 94-2213-E-194-006 and 94-EC-17-A-02-S1-032.

computation-aware scheme for motion estimation (ME) in H.264 is proposed.

2. Proposed scheme

2.1. Overview

The proposed computation-aware scheme for ME in H.264 is shown in Fig. 1. For a video sequence containing *n* frames, let C_{total} be the total computation allocated to the *n* frames, f_i be the *i*th frame, and MB_j be the *j*th MB. In this study, there are four phases: (1) frame-level computation allocation, (2) MB-level computation allocation, (3) mode-level computation allocation, and (4) pixel-level computation allocation. Frame-level computation allocation determines the target computation (CF_i) for f_i, MB-level computation allocation determines the target computation (CMB_j) for MB_j of f_i, mode-level computation allocation distributes CMB_j to two different ME modes (MB and block), and pixel-level computation allocation distributes each mode to two different pixel ME modes. In the frame and MB levels, the remaining computations are fed backward to update the available computations for the remaining frames and MBs, respectively.

In general [2], the number of checking points is treated as the measurement unit of computation power. In H.264, the *SAD* (sum of absolute differences) calculation is performed in a row-by-row manner and each partial *SAD* value containing a new row will be compared with the minimal *SAD* value, SAD_{min}, at that time. Based on the row-based partial *SAD* computation, *SAD* computations of different search positions may be different so that computation allocation in terms of the number of checking points is not adequate. Therefore, in this study, computation allocation is in terms of the row-based partial *SAD* (SADR). To unify two different block sizes, i.e., 8×8 and 16×16, in H.264, the one-row partial *SAD* computation of 8×8 blocks, $SADR_8$, is used as the computation measure.

In this study, temporal motion vector (MV) prediction [6] is employed to allocate total (available) computation (C_{total}) at the frame and MB levels. Temporal MV projection tracks the motion vector of a moving object from one frame to the next frame. For an MB in frame f_i, the corresponding temporally neighboring MVs in the previous frame f_{i-1} whose "motion-projected" MBs in the current frame will overlap the MB in frame f_i. Here, if an MB in frame f_i has no temporally motion-projected overlapping MBs, the predicted MV (PMV) of such an MB is set to (0,0). If an MB in frame f_i has only one temporally motion-projected overlapping MB, the PMV of such an MB is set to the corresponding MV of the temporally motion-projected overlapping MB in frame f_{i-1}. If an MB in frame f_i has two or more temporally motion-projected overlapping MBs, the PMV of the MB is set to the corresponding MV of the temporally motion-projected overlapping MB which has the longest MV_{SAC} value. Here if the two components of an MV is (MV_x, MV_y), the sum of absolute components (SAC) of the MV is defined as

$$MV_{\text{SAC}} = |\text{MV}_x| + |\text{MV}_y| \tag{1}$$

and the total sum of absolute components of all the *K* MVs in frame f_i is given by

$$TMV_{\text{SAC},i} = \sum_{k=1}^{K} MV_{\text{SAC},k}\ , \tag{2}$$

where $MV_{\text{SAC},k}$ is the MV_{SAC} value of the *k*th MV in f_i.

2.2. Frame-level computation

For frame f_i, let CBP_i, CFP_i, and CR_i be the backward PMV computation, forward PMV computation, and real MV computation for all MBs within frame f_i, respectively. The average computation cost (in terms of $SADR_8$) per MV_{SAC} for the pervious *i*-1 frames is defined as

$$\mu_{\text{ave}} = \sum_{l=1}^{i-1} CR_l \Big/ \sum_{l=1}^{i-1} TMV_{\text{SAC},l}\ . \tag{3}$$

If the predicted $TMV_{\text{SAC},i}$ value for frame f_i is denoted as $PTMV_{\text{SAC},i}$, the forward PMV computation for frame f_i can be given by

$$CFP_i = \mu_{\text{ave}} \times PTMV_{\text{SAC},i}\ . \tag{4}$$

Because the related information for the "remaining" frames is not available for frame f_i, the backward PMV computation for frame f_i is set to the remaining computations divided by the number (*n*-*i*+1) of remaining frames within the video sequence, i.e.,

$$CBP_i = \left(C_{\text{total}} - \sum_{l=1}^{i-1} CR_l \right) \Big/ (n-i+1)\ . \tag{5}$$

To avoid over-use of the available computation, it is claimed [2] that CFP_i smaller than or equal to CBP_i implies that insufficient computations are allocated to the previous *i*-1 frames and thus more computation can be allocated to frame f_i. On the contrary, CFP_i larger than CBP_i indicates that too much computation is allocated to the previous *i*-1 frames and the average computation allocated to each of the remaining *n*-*i*+1 frames should be gradually decreased. In this study, the target computation CF_i for frame f_i is given by

$$CF_i = \begin{cases} \alpha \times CBP_i, & \text{if } CFP_i \le CBP_i, \\ \min(\alpha \times CBP_i, CFP_i), & \text{otherwise,} \end{cases} \tag{6}$$

where α is a parameter for the backward PMV computation (CBP_i), which is defined as

$$\alpha = \begin{cases} \dfrac{PTMV_{\mathrm{SAC},i}}{\sum_{l=1}^{i-1} TMV_{SAC,l} \Big/ (i-1)}, & \text{if } \dfrac{PTMV_{\mathrm{SAC},i}}{\sum_{l=1}^{i-1} TMV_{SAC,l} \Big/ (i-1)} > 1, \\ 1, & \text{otherwise.} \end{cases} \quad (7)$$

After the MV search processes of all the MBs in frame f_i are finished, the remaining computation of CF_i will be fed backward to the next frame f_{i+1} using Eq. (5) to adjust the target computation CF_{i+1} for the next frame f_{i+1}.

2.3. MB-level computation

After the target computation CF_i for frame f_i has been determined, if there are $P\times Q$ MBs in frame f_i, let the current macroblock of frame f_i be $MB_{(p,q)}$, $PMV_{\mathrm{SAC},(p,q)}$ be the predicted MV_{SAC} of $MB_{(p,q)}$ and $PTMV_{\mathrm{SAC},i}$ be the predicted $TMV_{\mathrm{SAC},i}$ value for frame f_i. The target computation for $MB_{(p,q)}$, $CMB_{(p,q)}$, can be allocated in compliance with the $PMV_{\mathrm{SAC},(p,q)}$ proportion taken in $PTMV_{\mathrm{SAC},i}$, which is given by

$$CMB_{(p,q)} = CF_i \times \frac{PMV_{SAC,(p,q)}}{PTMV_{SAC,i}}. \quad (8)$$

To avoid the situation that $PMV_{SAC,(p,q)}= 0$ would make $CMB_{(p,q)}=0$, a basic value is added to $PMV_{SAC,(p,q)}$ when storing PMV_{SAC} to each MB in f_i. In this study, the basic value is 1. Moreover, to prevent another situation that the allocated computation for $MB_{(p,q)}$ has not been exhausted, the remaining computations of $MB_{(p,q)}$ would be fed backward to update $CMB_{(p,q+1)}$.

2.4. Mode-level computation

Table 1 shows the percentage of blocks whose number of search positions for finding the best MV in each Inter-mode of the QCIF "Akiyo," "Silent," and "Foreman" sequences. We can find that over 75% of blocks of the four prior modes (Inter16×16, Inter16×8, Inter8×16, and Inter8×8) search less than 30 positions, and 20% of blocks search over 110 positions. In Inter8×4 and Inter4×8 modes, the percentage of blocks searching less than 20 positions is 55%, and 40% of blocks search over 110 positions. 99% of the blocks of the last mode (Inter4×4) search less than 20 positions. Based on this observation, in this study, after the target computation *CMB* for an MB is allocated for the first-grade mode, the remaining computations will be allocated for the second-grade mode until *CMB* is exhausted.

2.5. Pixel-level computation

In H.264 JM96, motion estimation (ME) is conducted into integer-pixel ME and then fractional-pixel ME around the position obtained by the integer-pixel ME. For integer-pixel ME, motion vector prediction is used to select a start search position for ME search in H.264. The distance between the best MV (BMV) determined by the fast search strategy in H.264 and the predicted motion vector (PMV) in H.264 is defined as *MVD*:

$$MVD = \left|BMV_{\mathrm{x}} - PMV_{\mathrm{x}}\right| + \left|BMV_{\mathrm{y}} - PMV_{\mathrm{y}}\right|. \quad (9)$$

As shown in Table 2, about 90% of *MVD* is smaller than 1, i.e., the predicted MV is very close to the best MV.

Because fractional-pixel ME can provide significantly better compression performance than integer-pixel ME, in this study, the fractional-pixel ME has the higher priority to the integer-pixel ME.

3. Simulation Results

Six QCIF video sequences, "*Akiyo*," "*Salesman*," "*News*," "*Silent*," "*Foreman*," and "*Stefan*," are used to evaluate the performance of the proposed scheme. Each sequence consists of 100 frames, in which the first video frame is an I-frame, followed by 99 P-frames. The search range is set to 16 and quantization parameter (QP) is set to 28. Four performance measures are employed in this study. They are: (1) the average *PSNR* improvement (dB) with respective to JM_FME, (2) the average bit rate per second (kbps), (3) the total number of search positions, and (4) the total encoding time (s).

Here, one existing fast search algorithm and three comparison computation-aware schemes are implemented. They are: (1) JM_FME, which is the fast search algorithm in H.264 JM96, (2) CA_FME [5], which is the computation-aware scheme for software-based ME with JM_FME, (3) PRO FME, which is the proposed computation-aware scheme with JM_FME, (4) PRO DC, which is the proposed computation-aware scheme with adaptive dual-cross search algorithm. Some simulation results are listed in Table 3.

4. Concluding remarks

Based on the simulation results obtained in this study, given different available motion estimation computations (MECs), both the average PSNR and bit rate increment of the two proposed schemes are better than that of the comparison scheme CA_FME.

5. References

[1] ITU-T Recommendation H.264/ISO/IEC 11496-10, "Advance Video Coding," Final Committee Draft, Document JVT-F100, Dec. 2002.

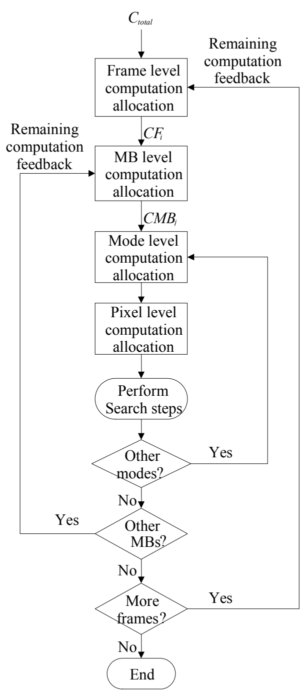
[2] C. Zhu, X. Lin, and L. P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349–355, May 2002.

[3] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. on Image Processing*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[4] X. Q. Banh and Y. P. Tan, "Adaptive dual-cross search algorithm for block-matching motion estimation," *IEEE Trans. on Consumer Electronics*, vol. 50, no. 2, pp. 766–775, May 2004.

[5] P. L. Tai, S. Y. Huang, C. T. Liu, and J. S. Wang "Computation-aware scheme for software-based block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 9, pp. 901-913, Sept. 2003.

[6] I. Ismaeil, A. Docef, F. Kossentini, and R. Ward, "Efficient motion estimation using spatial and temporal motion vector prediction," in *Proc. of IEEE Int. Conf. on Image Processing*, vol. 1, pp. 70-74, Oct. 1999.



C_{total}
Remaining computation feedback
Frame level computation allocation
CF_i
Remaining computation feedback
MB level computation allocation
CMB_i
Mode level computation allocation
Pixel level computation allocation
Perform Search steps
Other modes?
Yes
No
Other MBs?
Yes
No
More frames?
Yes
No
End

Fig. 1. The proposed computation-aware scheme.

Table 1. The relationship between the number of search positions (SP) and percentages of modes chosen after ME.

Mode	Sequence	SP≤20	21< SP ≤30	31< SP ≤40	41< SP ≤110	111< SP ≤120	SP>120
Inter 16×16	Akiyo	90.12	2.49	0.00	0.00	0.00	7.39
	Silent	81.11	9.42	0.07	0.06	0.32	9.02
	Foreman	43.59	30.22	0.04	0.01	6.11	20.03
Inter 16×8	Akiyo	61.36	13.64	0.00	0.00	0.57	24.43
	Silent	43.52	36.27	1.81	1.30	5.96	11.14
	Foreman	32.50	38.77	0.98	0.34	8.30	19.11
Inter 8×16	Akiyo	60.65	11.54	0.00	0.00	0.30	27.51
	Silent	45.79	27.59	0.77	1.52	3.64	20.69
	Foreman	38.10	40.42	0.27	0.10	5.36	15.75
Inter 8×8	Akiyo	72.85	0.38	0.00	1.01	25.76	0.00
	Silent	70.47	2.34	0.52	5.12	21.27	0.28
	Foreman	71.81	1.01	0.10	6.64	20.39	0.05
Inter 8×4	Akiyo	57.13	0.43	0.00	1.15	41.29	0.00
	Silent	53.52	2.97	0.39	8.88	33.75	0.49
	Foreman	50.35	1.43	0.02	15.04	33.01	0.15
Inter 4×8	Akiyo	54.52	0.33	0.00	1.80	43.35	0.00
	Silent	52.93	2.50	0.52	10.79	32.74	0.52
	Foreman	50.72	1.39	0.07	13.87	33.90	0.05
Inter 4×4	Akiyo	92.91	7.09	0.00	0.00	0.00	0.00
	Silent	78.60	20.52	0.88	0.00	0.00	0.00
	Foreman	92.56	7.27	0.17	0.00	0.00	0.00

Table 2. Percentages of different MVDs.

Sequence	0	1	2	3	4	5	6	7	>8
Akiyo	93.77	5.69	0.49	0.04	0.01	0.00	0.00	0.00	0.00
Salesman	85.59	11.29	2.38	0.39	0.19	0.05	0.03	0.02	0.06
News	81.59	13.88	2.88	0.83	0.27	0.15	0.04	0.07	0.29
Silent	75.65	15.83	5.02	1.41	0.71	0.44	0.27	0.11	0.55
Foreman	55.96	34.61	7.36	1.21	0.46	0.25	0.04	0.04	0.07
Carphone	59.12	30.56	7.26	1.50	0.58	0.27	0.15	0.17	0.37
Average	75.28	18.65	4.23	0.90	0.37	0.20	0.09	0.07	0.23

Table 3. Simulation results for the "*Akiyo*" sequence with different percentages of available search positions of three comparison schemes.

	Average PSNR improvement (dB)			Average bit rate increment (kbps)			Average search positions per frame (k)			Time speed up		
JM FME	38.50			33.70			80.02			1 (25.795 s)		
	CA FME	PRO FME	PRO DC	CA FME	PRO FME	PRO DC	CA FME	PRO FME	PRO DC	CA FME	PRO FME	PRO DC
90%	-0.03	-0.00	+0.01	+0.50	+0.08	-0.16	71.16	71.21	43.14	0.99	1.00	1.17
80%	-0.10	-0.00	+0.01	+1.37	+0.08	-0.16	63.26	63.34	43.14	1.01	1.05	1.17
70%	-0.15	-0.01	+0.01	+2.83	+0.00	-0.16	55.36	55.46	43.14	1.04	1.09	1.17
60%	-0.30	-0.00	+0.01	+4.41	+0.28	-0.16	47.45	47.56	43.08	1.04	1.04	1.17
50%	-0.36	-0.03	-0.00	+6.35	+0.17	-0.15	39.54	39.61	39.52	1.05	1.06	1.23
40%	-0.44	-0.03	-0.00	+8.06	-0.04	+0.39	31.63	31.69	31.68	1.07	1.10	1.31
30%	-0.49	-0.06	-0.08	+9.91	+0.06	-0.16	23.73	23.76	23.75	1.10	1.12	1.23
20%	-0.58	-0.06	-0.07	+14.90	-0.10	+0.07	15.82	15.83	15.83	1.17	1.19	1.49
10%	-0.63	-0.13	-0.10	+23.92	+0.44	+0.14	7.91	7.91	7.91	1.32	1.35	1.60