A TWO-STAGE VARIABLE BLOCK SIZE MOTION SEARCH ALGORITHM FOR H.264 ENCODER

Tomoyuki SHIMIZU Akio YONEYAMA Hiromasa YANAGIHARA Yasuyuki NAKAJIMA

KDDI R&D Laboratories Inc.
2-1-15 Ohara, Kamifukuoka-shi, Saitama 356-8502, Japan
E-mail: {tomoyuki, yoneyama, yanap, nakajima}@kddilabs.jp

ABSTRACT

We propose a fast motion search algorithm for H.264 motion estimation with variable block sizes. Motion estimation of H.264 encoder has larger computation complexity than existing video compression standards such as MPEG-4 and H.263 since search steps increase in proportion to the number of block sizes. Here, we employed two-stage motion estimation. In the first stage, 8x8 to 16x16 motion estimation is conducted where only limited areas are searched in 8x16, 16x8, and 16x16 block sizes using the results of 8x8 block search. Then in the second stage, 4x4/8x4/4x8 block size motion search is performed only when 8x8 block mode is chosen in the first stage. Using the above two approaches, limited search area and conditional smaller block size search, very fast and highly accurate variable block size motion estimation has been achieved. The experimental results also confirmed that the proposed algorithm can greatly reduce computational complexity while fully using seven block sizes and maintaining motion estimation efficiency.

1. INTRODUCTION

H.264/MPEG-4 Part 10 (AVC) [1] is a video compression standard which can achieve about twice higher coding efficiency than that of existing standards such as MPEG-4 and H.263. However, most of the coding tools adopted in H.264 involve very large computation complexity, e.g. variable block size motion estimation, quarter-pel motion compensation, in-loop deblocking filter, and multiple-frame inter prediction.

As for motion estimation, every macroblock (MB) can be divided into blocks in the size of 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 or 4x4. Here, 8x8/8x4/4x8/4x4 block modes are defined as subblock modes of an 8x8 block. Although smaller block size can reduce prediction error, it will increase the number of motion vectors (MVs), which may result in an increase of total code length of MVs. Therefore, H.264 encoder should choose the best block size according to a balance of minimizing prediction error and minimizing code length.

As for computational complexity, this multiple block size motion estimation is one of the most computationally intensive tasks among the above coding tools. For example, if the same motion search algorithm is employed for each block size, search steps will increase in proportion to the number of block sizes. On the other hand, when the fewer block sizes are employed, the lower coding efficiency may be expected.

Recently, several algorithms have been proposed to decrease search steps of variable block size motion search [2, 3, 4, 10]. For example, in [2] motion estimation of block sizes smaller than 8x8 is discarded and MVs of 8x8 blocks are merged for larger block sizes when a distance between the 8x8 block MVs is shorter than a threshold. However, from our preliminary experiments, it was observed that 4x4/4x8/8x4 block size motion search contributes to motion estimation efficiency of several sequences with non-uniform motion fields such as Mobile & Calendar. On the other hand, the motion estimation method in [3] always begins with 4x4 block size motion search and then performs larger block size motion search hierarchically, while most of MBs are determined to be one of 16x16-8x8 block size modes. Therefore, redundant 4x4/4x8/8x4 block size motion search may be conducted. The other method determines initial block size mode according to the result of motion search in a low resolution frame and discards several block size modes [4]. However, this algorithm involves computation besides motion search, such as creating low resolution frame and estimating SAD from low resolution motion search.

In this paper, we propose a two-stage variable block size motion search algorithm. The first stage consists of a motion search with 8x8/8x16/16x8/16x16 block sizes with limited search areas for 8x16/16x8/16x16 block search based on the results of 8x8 block search. In the second stage, a motion search with 8x4/4x8/4x4 block size is performed only when 8x8 block mode is chosen at the first step since further block size division may provide further improvement in coding efficiency. In the following, we describe the proposed motion estimation method followed by experimental results using several fast motion estimation methods.

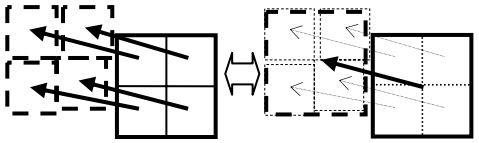
2. PROPOSED ALGORITHM

As stated earlier, although smaller block size motion search can reduce prediction error, it may increase code length of motion vectors. If further block size division introduces a slight decrease of prediction errors but results in a large increase of code length, larger block size should be chosen.

In the following, we describe typical two cases; Case 1 where MVs of neighboring two or more MBs are correlated, and Case 2 where MVs of neighboring four MBs are uncorrelated.

Case 1: if two or more neighboring 8x8 block MVs are correlated (e.g. Fig. 1), then the prediction error of the combined block (i.e. 8x16, 16x8, or 16x16) and the sum of the prediction

0-7803-8554-3/04/$20.00 ©2004 IEEE.



(a) 8x8 block search (b) 16x16 block search

Fig. 1: Relation of correlated 8x8 Block MVs and a 16x16 Block MV

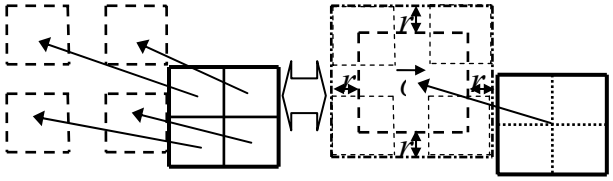


Fig. 2: Larger Block Size Search Area Estimation Using Distance and a Mean Vector of 8x8 MVs

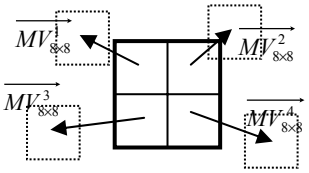


Fig. 3: Uncorrelated 8x8 Block MVs

errors of corresponding 8x8 blocks after motion estimation will be very close. In this case, the combined block size should be chosen since total code length of MV in the combined block size is shorter than that of 8x8 block MVs. Here, we employ the motion estimation results of 8x8 block size to determine search areas (search center and search range) of larger block sizes. In this way, motion estimation search points for other block sizes can be greatly reduced without losing searching accuracy.

Case 2: if all of the neighboring four 8x8 block MVs are different in direction or magnitude and therefore they are uncorrelated (Fig. 3), 8x8 block size may be the best suitable size. However, total code length of 8x8 block MVs might be much longer than that of a 16x16 block or other combined block MV in spite of PSNR gain. In this case, larger block size motion search should be also performed. Otherwise 8x8 block size or smaller size can be used.

2.1. Proposed Algorithm Overview

The basic concept of the proposed motion estimation algorithm is that (1) motion search block size modes and search range are reduced based on distance and dispersion of smaller block MVs and (2) motion search of smaller than 8x8 block sizes is performed only when 8x8 block mode is chosen at 8x8-16x16 block size motion search. The following shows the search steps for the two-stage variable block size motion estimation.

Stage 1: 8x8 to16x16 block size motion search

- Step 1: Perform 8x8 block size motion estimation
- Step 2: Determine search areas for 16x16/16x8/8x16block sizes using the results in Step 1
- Step 3: Perform motion estimation for 16x16/16x8/8x16 block sizes for the search areas obtained in Step 2
- Step 4: Select the best suitable block size among 16x16/16x8/8x16/8x8

Stage 2: 4x4 to 8x8 block size motion search

- Step 5: Perform motion estimation for 4x4/4x8/8x4 subblocks when 8x8 block size mode is chosen at Step 4, and determine the best subblock size

In every block size x, motion estimation is performed by searching $\overrightarrow{MV}$ which gives the minimum value of rate-distortion function $RDCost(x)$ [8]. Let $D(x, \overrightarrow{MV})$ and $R(x, \overrightarrow{MV})$ denote prediction error and code length of all motion vector differentials (MVDs) at the position $\overrightarrow{MV}$, respectively. Then, $RDCost(x)$ is given by

$$RDCost(x) = \min(D(x,\overrightarrow{MV}) + \lambda(QP)R(x,\overrightarrow{MV})) \quad (1)$$

$$where \begin{cases} \lambda(QP) = \text{int}[2^{(QP-12)/6} + 0.5] & for \quad QP \geq 12 \\ \lambda(QP) = 1 & for \quad QP \leq 12 \end{cases}$$

In the following subsections, we describe each step in detail.

2.2. Step 1: 8x8 Block Size Motion Search

At first, 8x8 block size motion search is performed for every four 8x8 blocks independently. Any existing integer-pel search algorithms can be applied here. We employ diamond search (DS)[6] and ripple-shaped search (RS)[7] algorithms, in order to evaluate the effects of both methods on the proposed algorithm.

All the SADs are cached per 4x4 block and reused for the other block size motion search.

2.3. Step 2: Search Area Decision

Let $\overrightarrow{MV}^i_{8\times8} = \left(x^i_{8\times8}, y^i_{8\times8}\right)$ (i=1,2,3,4) denote four 8x8 block MVs as shown in Fig. 3, and $\overrightarrow{Mean}^{1234}_{8\times8} = (x^{1234}_{8\times8}, y^{1234}_{8\times8})$ denote a mean vector of four 8x8 block MVs. D_{ij} is the distance between $\overrightarrow{MV}^i_{8\times8}$ and $\overrightarrow{MV}^j_{8\times8}$. Then the dispersion of four 8x8 block MVs $\sigma^2_{8\times8}$ can be defined as

$$\sigma^2_{8\times8} = 1/4 \sum_{n=1}^{4} \left\{(x^n_{8\times8} - x^{1234}_{8\times8})^2 + (y^n_{8\times8} - y^{1234}_{8\times8})^2\right\} \quad (2)$$

In the following subsections, we describe detailed search area decision for various block sizes reflecting the case study shown in the previous section.

2.3.1. 16x16 Block Size Motion Search Range

If all the distances between neighboring 8x8 block MVs are small as shown in Eq.(3), then the four 8x8 block MVs are estimated to be close to each other.

$$D_{12}, D_{13}, D_{24}, D_{34} < Thr1 \quad (3)$$

In this case, it is sufficient to perform 16x16 block size motion estimation within the area which covers all of four corresponding 8x8 block MVs (Fig. 2). Thus, 16x16 block motion search area can be limited to be within a range of ±max(D_{12}, D_{13}, D_{24}, D_{34}) centered by a mean vector of the 8x8 MVs. Since we observed in the preliminary experiments that even if the search area is limited to ±min(D_{12}, D_{13}, D_{24}, D_{34}), almost the same coding efficiency can be obtained, we apply the following search area for 16x16 block search.

$$\overrightarrow{C_{1234}} = \overrightarrow{Mean}^{1234}_{8\times8} \quad (4)$$

$$r_{1234} = \min(D_{12}, D_{13}, D_{24}, D_{34}) \quad (5)$$

If the above Eq.(3) is not satisfied, then the search area of 16x16 block is set to default range (e.g. ±32) centered in $\overrightarrow{C_{1234}}$ after further investigation of 8x8 block MV as describe in Section 2.5.

2.3.2. 16x8 Block Size Motion Search Range

If the distances between two horizontally adjacent MVs are small as shown in Eq.(6), these MVs can be regarded as correlated.

$$D_{12} < Thr2 \quad or \quad D_{34} < Thr2 \quad (6)$$

In this case, the upper and lower 16x8 block size motion searches can be performed within the limited search area which can be determined, in the same manner as 2.3.1.

$$\overrightarrow{C_{12}} = \frac{1}{2}\left(\overrightarrow{MV^{1}_{8\times8}} + \overrightarrow{MV^{2}_{8\times8}}\right), \overrightarrow{C_{34}} = \frac{1}{2}\left(\overrightarrow{MV^{3}_{8\times8}} + \overrightarrow{MV^{4}_{8\times8}}\right) \quad (7)$$

$$r_{12} = D_{12} \quad or \quad r_{34} = D_{34} \quad (8)$$

where $\overrightarrow{C_{12}}$ and $\overrightarrow{C_{34}}$ are centers of upper and lower 16x8 search blocks and r_{12} and r_{34} are search areas of upper and lower 16x8 search blocks, respectively. The same search procedure as the previous section is applied when the above Eq. (6) is not satisfied.

2.3.3. 8x16 Block Size Motion Search Range

In a similar manner to the previous section, two vertically adjacent 8x8 block motion vectors are estimated to be close to each other if D_{13} or D_{24} is small as described in the following.

$$D_{13} < Thr2 \quad or \quad D_{24} < Thr2 \quad (9)$$

Then, the search areas of the left and right 8x16 block size motion searches can be limited to the following.

$$\overrightarrow{C_{13}} = \frac{1}{2}\left(\overrightarrow{MV^{1}_{8\times8}} + \overrightarrow{MV^{3}_{8\times8}}\right), \overrightarrow{C_{24}} = \frac{1}{2}\left(\overrightarrow{MV^{2}_{8\times8}} + \overrightarrow{MV^{4}_{8\times8}}\right) \quad (10)$$

$$r_{13} = D_{13} \quad or \quad r_{24} = D_{24} \quad (11)$$

where $\overrightarrow{C_{13}}$ and $\overrightarrow{C_{24}}$ are centers of left and right 8x16 search blocks and r_{13} and r_{24} are search areas of left and right 8x16 search blocks, respectively.

2.3.4. Four Uncorrelated 8x8 Block MVs

When the dispersion of 8x8 block MVs are high as described in Eq. (12), four 8x8 block MVs can be regarded as uncorrelated.

$$\sigma^2_{8\times8} > Thr3 \quad (12)$$

In this case, as stated in Case 2, since larger block sizes than 8x8 may provide the best size considering the balance between code length and prediction error, motion searches are conducted within the default search range centered in the averaged 8x8 block MVs (i.e. for 16x16, $\overrightarrow{C_{1234}}$).

When none of the conditions stated above is satisfied, 8x8 block size is determined as the best suitable mode.

2.4. Step 3: Larger Block Size Motion Search

Motion estimation in 16x16/16x8/8x16 block sizes is conducted using DS for search areas determined in Step 2.

2.5. Step 4: 8x8-16x16 Block Size Decision

The best suitable block size among 8x8 to 16x16 block sizes is determined by comparing the motion estimation results obtained in the above steps. Here, we use low-complexity mode of rate-distortion optimization [8]. The best block size is given as the block size x which minimizes $RDCost(x)$.

2.6. Step 5: 4x4/4x8/8x4 Subblock Size Motion Search

This step is performed only if 8x8 block MV mode is chosen in Step 4. At first, 4x4 subblock MVs are searched by using DS. Search center is determined to a MV of the corresponding 8x8 block. By applying a similar manner to Step 2 and 3, search areas for 4x8 and 8x4 block sizes are determined based on the motion search results of 4x4 block search and motion search is performed for these blocks. Note that 8x8 block size motion search is skipped since it is already performed in Step 1.

Then, the subblock size of the 8x8 block is by comparing all sizes chosen according to the above step. Here, the same manner as section 2.5 is also applied to the 8x8-4x4 subblock size decision.

3. EXPERIMENTAL RESULTS

We implemented the proposed algorithm in H.264 reference model JM 7.3 encoder and compared JM full search (FS) with each of DS, RS, DS with two-stage block size decision (DS+2Stage), DS with the proposed algorithm (DS+Proposed) , RS with the proposed algorithm (RS+Proposed). The experimental conditions are shown in Table 1. Thresholds *Thr1*, *Thr2* and *Thr3* are determined from our preliminary experiments.

In integer and fractional pel motion search, SAD and SA(T)D are respectively used as prediction error. Early elimination [5] is also adopted in order to decrease SAD calculation times. Table 2 shows the experimental results of the proposed algorithm. BDBR and BDPSNR [9] indicate difference of rate-distortion characteristic between FS and each of given methods. S_{SAD} and $S_{SA(T)D}$ respectively indicate ratio of the number of SAD and SA(T)D calculation in 4x4 block unit given by Eqs. (13) and (14).

$$S_{SAD} = \#SAD(FS) / \#SAD(\text{given method}) \quad (13)$$

$$S_{SA(T)D} = \#SA(T)D(FS) / \#SA(T)D(\text{given method}) \quad (14)$$

As shown in Table 2, DS algorithm is about 194 times faster than FS, while its PSNR penalty is about 0.07 dB. DS+2Stage algorithm is about 276 times faster than FS, while its PSNR penalty is about 0.12dB. Therefore, variable block size motion estimation becomes about 1.4 times faster by adopting two-stage block size decision, while its PSNR penalty is about 0.05 dB.

As for the whole proposed algorithm, DS+Proposed is about 1374 times faster than FS, while its PSNR penalty is about 0.19 dB. Therefore, DS+Proposed is about 7.1 times faster than DS. In [10], a fast motion estimation method for H.264 is proposed and compared with DS. In this report, about 2 times faster speed is achieved with almost no PSNR degradation but with up to a few percent bit rate increase. When compared with these results, the proposed method can provide sufficiently high speed motion search with very small PSNR penalty.

As for RS based search, it is about 40% slower than DS based search with almost the same PSNR. Although RS always

Table 1: Simulation Conditions

Profile	Baseline Profile
QP	28, 32, 36, 40
Thresholds	*Thr1*=4, *Thr2*=6, *Thr3*=24
Search Range	±32
Optimization	R-D Optimization with Hadamard Transform
No. of Reference Frame(s)	1

Table 2: RD & Calculation Performance Comparison

		BDBR [%]	BDPSNR [dB]	S_{SAD}	$S_{SA(T)D}$
Container (QCIF, 10fps)	DS	0.10	0.00	187.68	1.00
	RS	0.23	-0.01	209.57	1.00
	DS+2Stage	1.19	-0.06	263.77	1.76
	DS+Proposed	1.60	-0.08	1649.20	1.76
	RS+Proposed	1.69	-0.09	1206.06	1.76
Foreman (QCIF, 10fps)	DS	5.19	-0.25	173.28	1.00
	RS	7.81	-0.37	201.98	1.00
	DS+2Stage	5.87	-0.28	249.00	1.61
	DS+Proposed	11.59	-0.55	911.81	1.53
	RS+Proposed	10.03	-0.47	697.56	1.53
News (QCIF, 10fps)	DS	0.82	-0.05	183.40	1.00
	RS	1.79	-0.11	222.56	1.00
	DS+2Stage	2.05	-0.13	246.90	1.67
	DS+Proposed	2.44	-0.15	1422.89	1.67
	RS+Proposed	2.42	-0.15	1056.09	1.67
Mobile & Calendar (CIF, 30fps)	DS	-0.12	0.01	226.91	1.00
	RS	2.16	-0.10	294.41	1.00
	DS+2Stage	1.04	-0.05	336.51	1.60
	DS+Proposed	2.77	-0.13	1558.71	1.44
	RS+Proposed	3.03	-0.14	1101.03	1.44
Silent Voice (QCIF, 15fps)	DS	1.04	-0.05	187.67	1.00
	RS	1.76	-0.09	217.29	1.00
	DS+2Stage	1.72	-0.09	266.60	1.65
	DS+Proposed	2.55	-0.13	1229.49	1.64
	RS+Proposed	2.33	-0.12	913.89	1.64
Paris (CIF, 15fps)	DS	1.43	-0.08	192.78	1.00
	RS	2.77	-0.16	252.03	1.00
	DS+2Stage	2.69	-0.15	270.42	1.70
	DS+Proposed	3.78	-0.21	1431.82	1.67
	RS+Proposed	3.77	-0.21	1064.39	1.68
Tempete (CIF, 30fps)	DS	0.93	-0.04	206.13	1.00
	RS	2.50	-0.11	271.37	1.00
	DS+2Stage	1.73	-0.07	300.34	1.61
	DS+Proposed	2.33	-0.10	1414.55	1.59
	RS+Proposed	2.39	-0.10	1035.72	1.59
average	DS	1.34	-0.07	193.98	1.00
	RS	2.72	-0.14	238.46	1.00
	DS+2Stage	2.33	-0.12	276.22	1.66
	DS+Proposed	3.86	-0.19	1374.07	1.62
	RS+Proposed	3.67	-0.18	1010.68	1.61

searches a MV at scattered points within a wide search area during the first ripple-shaped coarse search step in order to avoid falling into local minimum, the number of calculation steps is increased since the cached 8x8 motion search results will not be reused in the scattered points

On the other hand, since search points of DS are more center-biased, cached SADs results in DS are frequently reused for larger block size motion search. As a result, DS+Proposed becomes about 1.4 times faster than RS+Proposed.

As for SA(T)D calculation at sub-pel motion search refinement, the proposed algorithm is about 1.6 times faster than the full search. This indicates that 40% of variable block size modes are decided to be unnecessary to perform motion search.

The actual encoding time of original JM and DS+Proposed on 3.2GHz Pentium 4 processor with 1GB memory, using the sample sequence "Tempete," is 903.04 sec and 317.42 sec, respectively. Therefore, DS+Proposed is about 2.8 times faster than original JM.

4. CONCLUSIONS

We proposed a fast search algorithm for variable block size motion estimation of H.264. We employ a two-stage motion search starting with 8x8 block size. By analyzing motion search results at 8x8 block size, search ranges and block sizes are adaptively selected and motion search for other block sizes is efficiently conducted under the obtained conditions. Since any fast algorithm can be used for 8x8 block size motion search, further improvement can be expected using such algorithm as MVFAST [5].

The proposed algorithm can be adopted for multiple reference frame motion search by using the same algorithm for every reference frame. Computation time of motion search increases in proportion to the number of reference frames. However, its search steps can be further reduced by incorporating correlation among motion vectors of reference frames. Therefore, we are planning to optimize the proposed algorithm for multiple reference frame motion search.

5. REFERENCES

[1] ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, 2003.

[2] Yu-Kuang Tu, *et al*, "Fast Variable-size Block Motion Estimation Using Merging Procedure with an Adaptive Threshold," *IEEE Proc. ICME*, vol. II, pp.789-792, 2003.

[3] Woong Il Choi, *et al*, "Fast Motion Estimation with Modified Diamond Search for Variable Motion Block Sizes," *IEEE Proc. ICIP*, vol. II, pp.371-374, 2003.

[4] C.-H Kuo *et al*, "A Fast Variable Block Size Motion Compensation Algorithm for H.264 Video Coding," *Proc. of SPIE ITCom 2003*, 5241-12, 2003.

[5] P. Hosur and K. Ma, "Motion Vector Field Adaptive Fast Motion Estimation," *ICICS '99*, Singapore, 7-10, 1999.

[6] S. Zhu and K. Ma, "A New Diamond Search Algorithm for Fast Block Matching," *IEEE Trans. Image Processing*, vol. 9, No. 2, pp. 287-290, 2000.

[7] Y. Nakajima, A. Yoneyama, *et al*, "A Fast Motion Estimation Algorithm for MPEG2 Video Using Ripple Shaped Search," *IEEE Proc. ISCAS*, vol. 4, pp.207-210, 1999.

[8] ISO/IEC | ITU-T VCEG, JVT-B118r8, 2002.

[9] G. Bjontegard, "Calculation of Average PSNR Differences between RD-curves," ITU-T, VCEG-M33, 2001.

[10] K.-K. Ma and G. Qiu, "Unequal-Arm Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation in the JVT/H.26L," *IEEE Proc. ICIP*, vol. I, pp.901-904, 2003.