A Fast Motion Vector Search Algorithm for Variable Blocks

Yung-Lyul Lee[1], Yung-Ki Lee[1], and HyunWook Park[2]

[1] Sejong University, Department of Internet Engineering, DMS Lab.,
98 Kunja-dong, Kwangjin-gu, Seoul, Korea
yllee@sejong.ac.kr
[2] Department of Electrical Engineering
KAIST, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea

Abstract. A fast motion estimation (ME) algorithm is proposed to search motion vectors for variable blocks. The proposed method is based on the successive elimination algorithm (SEA) using sum norms to find the best estimate of the motion vectors and to implement efficient calculations for variable blocks. The proposed ME algorithm is applied to the Joint Video Team (JVT) encoder that performs a variable-block ME. In terms of computational complexity, the proposed ME algorithm with limited search range searches motion vectors at about 6.3 times as fast as the spiral full search and 5.5 times as fast as the fast full search using the hierarchical sum of absolute difference (SAD), while the PSNR (peak signal-to-noise ratio) of the reconstructed image is slightly degraded with only 0.1~0.4 dB.

1 Introduction

Most video compression standards, including ITU-T H.263 [1], MPEG-4 [2], and the JVT codec [3] which is a joint standard of ITU-T Recommendation H.264 and ISO/IEC MPEG-4 Part 10, use a block-based motion estimation/compensation. Motion estimation (ME) plays an important role in saving bit-rates by reducing temporal redundancy in moving picture coding, but it requires intensive computations in the encoder. In particular, the JVT codec adopts the variable block-based motion estimation/compensation for each 16×16 macroblock (MB), whose motion vectors have quarter pixel resolution. It improves the peak signal to noise ratio (PSNR) and subjective quality, but it requires heavy computations for motion estimation.

Several ME algorithms have been proposed to save computation time. The well-known approaches to speed up motion estimation are the three-step search [4], the 2-D logarithmic search [5], the one-at-a-time search (OTS) [6], and the new diamond search [7]. These methods estimate integer motion vector with approximately 3%~5% of the computational complexity compared to the full search by sampling of the search positions for ME. However, these approaches cannot provide as high PSNR as the full search. The successive elimination algorithm (SEA) [8] shows the same PSNR as the full search with 13% computational complexity of the full search. The enhanced SEA (ESEA) [9] further reduces computational complexity, while maintain-

J. Blanc-Talon et al. (Eds.): ACIVS 2006, LNCS 4179, pp. 311 – 322, 2006.

ing the PSNR. The spiral full search [10] in the JM (Joint Model) of JVT performs a full search in a way that an initial motion vector is predicted by median value of motion vectors of the adjacent blocks and the motion vector search is performed in the spiral sequence from the median-predicted motion vector to ±16 motion vector. The fast full search [10] in the JM performs SAD calculations of sixteen 4×4 blocks in an MB over ±16 search ranges and the SAD values of 4×4 blocks are used hierarchically to estimate the motion vectors of larger blocks.

In order to apply the concept of SEA to the JVT codec, a new fast motion vector search algorithm is proposed for integer-pixel unit. We also use the rate-distortion optimization (RDO) [11] that was adopted in JVT as an informative module. The RDO is applied to select the optimum variable block size, with which we have both the minimum mean-square error and the minimum bit allocations for each 16×16 MB.

This paper consists of the following three sections. In Section 2, a detailed proposed method is described. Experimental results are shown to evaluate the proposed method in Section 3. Finally, conclusions are given in Section 4.

2 Motion Estimation Algorithm in Integer-Pixel and Quarter-Pixel Units

2.1 Motion Estimation Algorithm in Integer-Pixel Unit

The JVT codec uses the motion estimation of the variable blocks such as 16×16, 16×8, 8×16, and 8×8 blocks for each 16×16 MB and 8×4, 4×8 and 4×4 blocks for each 8×8 block. In the hierarchical motion vector search algorithm of the variable blocks, the best motion vector is found in consideration of the number of bits to represent motion vectors and the SAD of the motion-compensated MB. In this paper, sum norms are calculated in advance on the reference frame for fast motion search.

Assume that the current frame is denoted by $f(i,j)$ with spatial coordinates (i,j), and the reference frame is $r(i,j)$. Motion estimation of S×T blocks, such as 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4, is performed by using the inequality equation [8, 12] as follows:

$$\left|\sum_{i=1}^{S}\sum_{j=1}^{T}|r(i-x,j-y)|-\sum_{i=1}^{S}\sum_{j=1}^{T}|f(i,j)|\right|\le\sum_{i=1}^{S}\sum_{j=1}^{T}|r(i-x,j-y)-f(i,j)| \tag{1}$$

where (x, y) represents a motion vector. For simplicity, eq. (1) is described as follows:

$$\left|R_{S\times T}(x,y)-F_{S\times T}\right|\le SAD_{S\times T}(x,y) \tag{2}$$

$$R_{S\times T}(x,y)=\sum_{i=1}^{S}\sum_{j=1}^{T}|r(i-x,j-y)|$$

where $F_{S\times T}=\sum_{i=1}^{S}\sum_{j=1}^{T}|f(i,j)|$

$$SAD_{S\times T}(x,y)=\sum_{i=1}^{S}\sum_{j=1}^{T}|r(i-x,j-y)-f(i,j)|$$

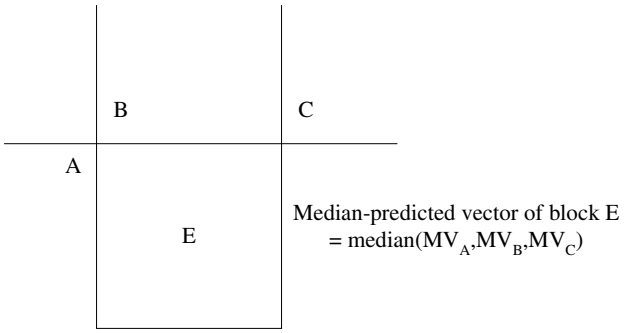


Fig. 1. Neighboring blocks for the motion vector prediction of current block. E is the current block and A, B, and C are the neighboring blocks. MVA, MVB, and MVC are motion vectors of the neighboring blocks A, B, and C, respectively.

In eq. (2), $R_{S\times T}(x, y)$ is the sum norm of $S\times T$ block of the reference frame with the motion vector (x,y), $F_{S\times T}$ is the sum norm of $S\times T$ block of the current frame and $SAD_{S\times T}(x, y)$ is the sum of absolute difference (SAD) between the current $S\times T$ block and the reference $S\times T$ block with the motion vector (x,y). If the motion vector (x,y) is a better estimation than the other motion vector (m,n) in terms of SAD, then the inequality is given as follows:

$$SAD_{S\times T}(x, y) \le SAD_{S\times T}(m, n) \tag{3}$$

Eqs. (2) and (3) can be combined as follows:

$$| R_{S\times T}(x, y) - F_{S\times T} | \le SAD_{S\times T}(m, n) \tag{4}$$

Eq. (4) can be rearranged without absolute terms as follows:

$$F_{S\times T} - SAD_{S\times T}(m, n) \le R_{S\times T}(x, y) \le F_{S\times T} + SAD_{S\times T}(m, n) \tag{5}$$

If eq. (5) is satisfied on the motion vector (x,y), $SAD_{S\times T}(x, y)$ is calculated and $SAD_{S\times T}(m, n)$ is replaced with $SAD_{S\times T}(x, y)$. To obtain the best estimation of the current $S\times T$ block, computation of SAD is performed only for the motion vector (x,y) satisfying eq. (5). The number of motion vectors satisfying eq. (5) is obviously less than the total search range. Therefore, the proposed algorithm can reduce the computational complexity without degradation of the PSNR. The efficiency of the algorithm depends on the fast calculation of the sum norms for each block and the initial motion prediction, $SAD_{S\times T}(m, n)$. Fig. 1 illustrates the relative location of the current block E and the neighboring blocks of A, B, and C, whatever the block size is. The median-predicted motion vector PMV is used as the initial motion vector of the current block, which is predicted from the upper block B, the upper right block C, and the left block A of the current block E as follows:

$$PMV=median(MV_A,MV_B,MV_C),$$

where MV_A, MV_B, and MV_C are motion vectors of A, B, and C, respectively in Fig. 1.

Since JVT adopts variable blocks for motion estimation/compensation, there are various numbers of motion vectors for each MB according to the block size. As an example, if all 4×4 blocks in an MB are selected, 16 motion vectors should be encoded and transmitted to receiver. Including the motion vector cost as suggested in the JVT encoder, eq. (3) is modified as follows:

$$\begin{aligned} &SAD_{S\times T}(x,y)+\lambda\times MVbits(PMV-(x,y)) \\ &\quad\le SAD_{S\times T}(m,n)+\lambda\times MVbits(PMV-(m,n)) \end{aligned} \tag{6}$$

where the Lagrangian multiplier λ is set to $\sqrt{0.85\times 2^{(QP-12)/3}}$ as defined in the JVT encoder [3, 10, 11], $MVbits(PMV\text{-}(x,y))$ is the bit amount to represent the difference between the motion vector (x,y) and the predicted-motion vector PMV by Exp-Golomb code, and QP is the quantization parameter of the JVT codec. Then, eq. (6) is rearranged to be used for the proposed fast motion estimation as follows:

$$\begin{aligned} &F_{S\times T}-(SAD_{S\times T}(m,n)+\lambda\times(MVbits(PMV-(m,n))-MVbits(PMV-(x,y))) \\ &\le R_{S\times T}(x,y) \\ &\le F_{S\times T}+(SAD_{S\times T}(m,n)+\lambda\times(MVbits(PMV-(m,n))-MVbits(PMV-(x,y))) \end{aligned} \tag{7}$$

Eq. (7) is the proposed sum norm inequality which considers both SAD and the motion vector cost. Weighting of SAD and the motion vector cost can be adjusted by the Lagrangian multiplier λ, which is the same value as JVT encoder in this paper.

2.2 Efficient Calculation of the Hierarchical Sum Norms of Variable Blocks

The sum norms of the variable blocks are computed hierarchically. First of all, the sum norms of the smallest blocks of 4×4 among variable blocks are first calculated in the reference frame. In order to reduce the computation amount, an efficient procedure to compute the sum norms of 4×4 blocks is described. The first row strip, $C(0, j)$, that is the sum of four elements of j-th row is computed for reference frame as follows:

$$C(0,j)=\sum_{k=0}^{3} r(k,j) \quad for \quad 0\le j<W \tag{8}$$

Then, next row strip, $C(i,j)$ for $i\ge 1$, is computed by using $C(i\text{-}1,j)$ as follows:

$$C(i,j)=C(i-1,j)-r(i-1,j)+r(i+3,j) \ \ for\ 1\le i<H-3, \ \ 0\le j<W \tag{9}$$

From $C(i, j)$, $R_{4\times4}(x,y)$ (x=0,1,2,…,H-4, y=0,1,2,…,W-4) is derived in the same way as the sum of four column elements as follow:

$$\begin{aligned} &R_{4\times4}(i,0)=\sum_{k=0}^{3} C(i,k) \ \ for\ 0\le i<H-3 \\ &R_{4\times4}(i,j)=R_{4\times4}(i,j-1)-C(i,j-1)+C(i,j+3) \ \ for\ 0 \\ &\qquad\le i<H-3, \quad 1\le j<W-3 \end{aligned} \tag{10}$$

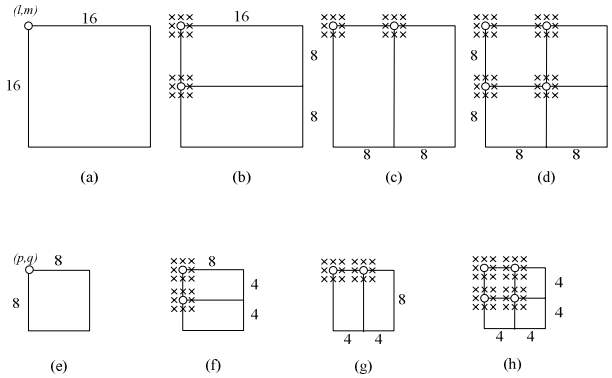


Fig. 2. Limited search ranges for each variable block matching: (a) matching position of the 16×16 macroblock in the reference frame, (b) search positions in the reference frame of the two 16×8 blocks, (c) search positions in the reference frame of the two 8×16 blocks, (d) search positions in the reference frame of four 8×8 blocks, (e) matching position of the 8×8 block in the reference frame, (f) search positions in the reference frame of two 8×4 blocks, (g) search positions in the reference frame of two 4×8 blocks, (h) search positions in the reference frame of four 4×4 blocks

From the above sum norms of 4×4 blocks, the sum norms of 4×8 and 8×4 blocks are calculated as follows:

$$R_{4\times8}(i,j) = R_{4\times4}(i,j) + R_{4\times4}(i,j+4) \;\; for \;\; 0 \le i < H-3, \quad 0 \le j < W-7 \tag{11}$$

$$R_{8\times4}(i,j) = R_{4\times4}(i,j) + R_{4\times4}(i+4,j) \;\; for \;\; 0 \le i < H-7, \quad 0 \le j < W-3 \tag{12}$$

Also, $R_{8\times8}(i,j), R_{8\times16}(i,j), R_{16\times8}(i,j),$ and $R_{16\times16}(i,j)$ can be derived in the same way as $R_{4\times8}(i,j)$ and $R_{8\times4}(i,j)$. The proposed hierarchical sum norm calculation is based on 4×4 sum norm and expanded to the sum norms calculation of lager blocks using 4×4 sum norm in eqs. (11) and (12). The main contributions on the proposed method are that the proposed sum norm inequality of eq. (7) considers both SAD and the motion vector cost and the proposed hierarchical sum norms calculations of eqs. (8)~(11) enable to find the motion vector very fast.

2.3 Motion Estimation Procedure for Variable Blocks in Integer-Pixel Unit

The hierarchical calculation of sum norms can reduce the computation time. The sequence of variable block matching is executed from the 16×16 block to the 4×4 blocks to find the best estimates of the motion vectors and block sizes. After obtaining the seven sum-norm sets of $R_{4\times4}(i,j)$, $R_{4\times8}(i,j)$, $R_{8\times4}(i,j)$, $R_{8\times8}(i,j)$, $R_{8\times16}(i,j)$, $R_{16\times8}(i,j)$, and $R_{16\times16}(i,j)$ in the reference frame, the motion estimation of variable blocks is performed by eq. (7) for 16×16 MBs. Once we find the motion vector (l,m) that has a minimum SAD for 16×16 MB as shown in Fig. 2(a), motion estimation of16×8, 8×16, 8×8 blocks are performed for the motion vector(l,m) and its eight-neighbor search ranges as shown in Fig. 2(b)-(d), in which the eight- neighbor search

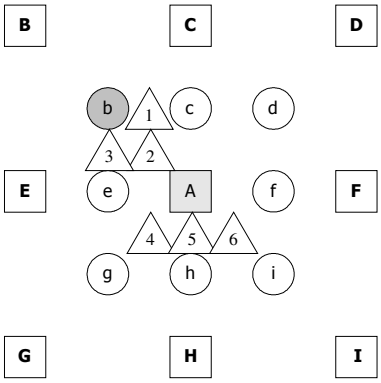


Fig. 3. A fast quarter pixel search method: the upper-case letters are integer-pixel unit, the lower case letters are half-pixel unit, and the numbers in triangle are quarter-pixel unit

ranges mean ±1 integer search range around the motion vector(l,m) for estimating 16×8, 8×16, 8×8 block motion vector. This limited search range can reduce the computation amount, whereas image quality is slightly degraded. After we find an optimum motion vector for each 8×8 block, the 8×8 block is divided into two 8×4 blocks, two 4×8 blocks and four 4×4 blocks. If the optimum motion vector of the 8×8 block is (p,q), the motion estimation of 4×8, 8×4, and 4×4 blocks is performed for the motion vector (p,q) and its eight neighbor motion vectors. In order to retain the image quality, the search range can be ± 16 for all variable blocks, which requires a longer computation time compared to the limited search range.

The proposed algorithm adopts the early termination to reduce the computations of $SAD_{S\times T}(x, y)$. $SAD_{S\times T}(x, y)$ should be calculated only if eq. (7) is satisfied. When $SAD_{S\times T}(x, y)$ is calculated, the intermediate result of $SAD_{S\times T}(x, y)$ between the current block and the reference block after 75% calculations of S×T block is compared with $SAD_{S\times T}(m, n)$. If the intermediate result is greater than $SAD_{S\times T}(m, n)$, the remaining calculation is not necessary. Therefore, it slightly contributes to the reduction of computation amounts.

2.4 Fast Quarter-Pixel Search

For fast quarter-pixel motion estimation, three points quarter-pixel search algorithm is developed. An example of quarter-pixel search is shown in Fig. 3, in which the upper-case letters are integer-pixel unit, the lower case letters are half-pixel unit, and the numbers in triangle are quarter-pixel unit. When an integer pixel is found as the matching block of a block that can be one of 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 blocks according to the value of S and T, the eight-neighbor positions of b, c, d, e, f, g, h, and i in half pixel unit are investigated by full search and the candidate position that has the minimum MAD is selected as the best half pixel motion vector position. If the half-pixel motion vector position is b, only positions of 1, 2, and 3 in quarter-pixel unit are investigated for the quarter-pixel search. When the half-pixel

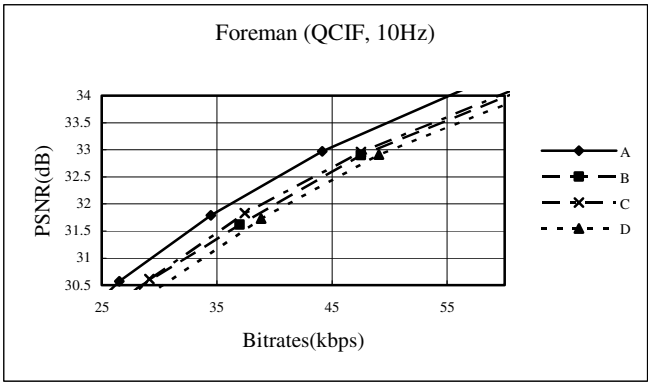
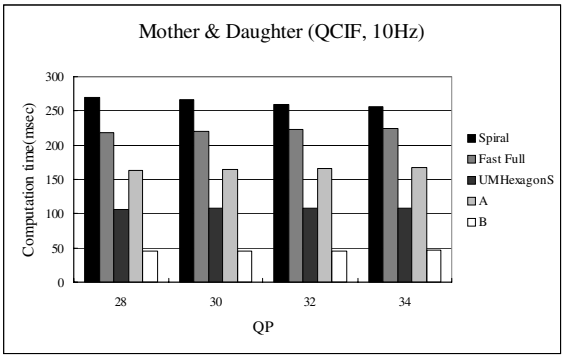


Fig. 4. Rate-distortion plots of the proposed methods "**A**" and "**B**", the methods "**C**", and "**D**" to compare the performance of eq. (5) and eq. (7)

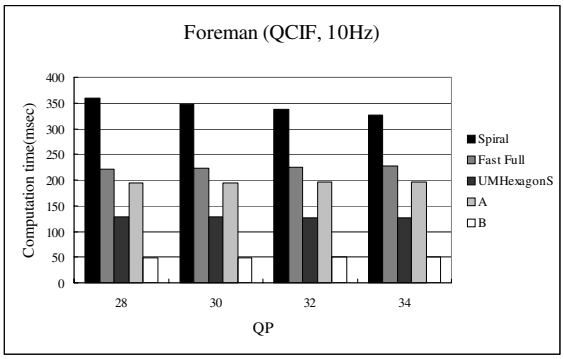
motion vector position is h, only positions of 4, 5, and 6 are investigated for the quarter-pixel search. That is, only three quarter pixel positions between the integer motion vector position and the half-pixel motion vector position are investigated as the candidate of quarter-pixel motion vector, because the possibility of the best quarter pixel candidate is very high within three points in quarter pixel unit between the best integer and best half pixel position.

3 Experimental Results

The proposed motion vector search algorithm is applied to the JVT encoder [3,10] with the Exp-Golomb code, variable-block ME/MC having 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, and 4×4 blocks, ±16 motion search range, quarter pixel interpolation, 4×4 DCT (Discrete Cosine Transform), and rate-distortion optimization. Several image sequences, each of which has 300 frames, were used for this experiment. Each sequence is compressed with the scheme of I,P,P,P,P… , i.e., only the first frame is INTRA frame, while the others are all INTER frames without the B frame. The proposed method is compared to the spiral full search, the fast full search and UMHexagon search (Unsymmetrical-cross Multi-Hexagon grid Search) [13], which are implemented in JM (Joint Model) source code [10], with respect to PSNR and computation time. In the comparison study, the "A" method is the proposed method using eq. (7) with ±16 search ranges for all variable blocks that gives the same PSNR as the spiral full search, and the "B" method is the proposed method using eq. (7) with the limited search ranges described in Section II. The fast full search in the JM source code performs SAD calculations of 4×4 blocks for variable-block motion estimation, in which SAD values of 4×4 blocks are used hierarchically to estimate motion vectors of larger blocks. Also, the same rate distortion optimization technique, which was introduced in the references [10, 11], is used for fair comparison.



(a)

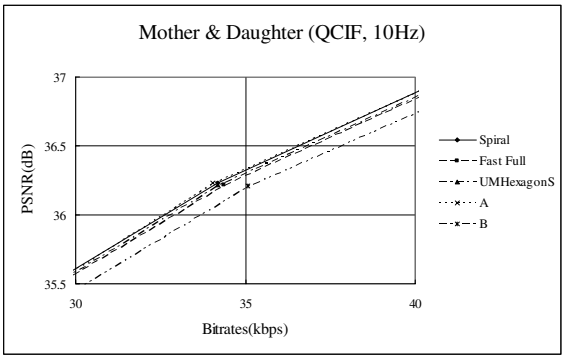


(b)

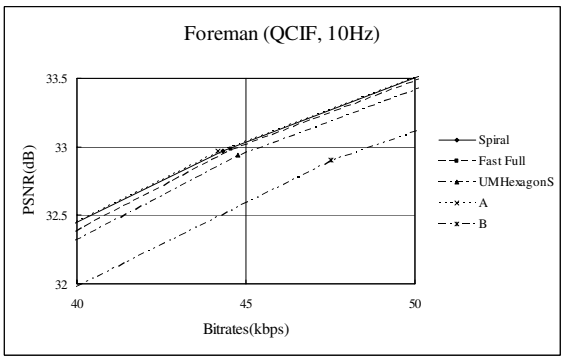
Fig. 5. The computation time of the spiral full search, the fast full search, UMHexagonS, the proposed method "**A**", and "**B**". Only the first frame is Intra-coded: (a) "Mother & Daughter" sequence with a frame rate of 10Hz and QCIF resolution, (b) "Foreman" sequence with a frame rate of 10Hz and QCIF resolution.

In order to show that eq. (7) has PSNR improvement in comparison to eq. (5), the motion estimation using eq. (7) is compared to that using eq. (5) in terms of PSNR as shown in Fig. 4. "C" and "D" are the methods using eq. (5) with ± 16 and the limited search ranges, respectively. The result of eq. (7) is approximately 0.4~0.5 dB better than that of eq. (5) in the Foreman QCIF sequence with a frame rate of 10 Hz.

All experiments were carried out on a Pentium IV, 1.7 GHz, using the JVT (Joint Video Team) codec [10] for various video sequences with QCIF and CIF size. The computation times of four different motion estimation methods are compared for two QCIF sequences in Figs. 5(a) and (b), where the horizontal axis represents the quantization parameters of the JVT codec. The computation time(msec) is the measured average time per a frame on the Pentium IV-1.7GHz for only motion estimation in JM73. The "**A**" method shows that the proposed method results in the same PSNR as the spiral full search if it is applied to ± 16 search ranges for all variable-blocks whereas computation amount increases in Figs. 5 and 6. UMHexagonS, which is



(a)



(b)

Fig. 6. Rate-distortion plots: (a) "Mother & Daughter" rate-distortion plot corresponding to Fig. 5(a), (b) "Foreman" rate-distortion plot corresponding to Fig. 5(b)

adopted as an informative part of JVT, is the motion estimation method using initial search point prediction and early termination. In the "Mother & Daughter" and "Foreman" sequences, Figs. 5(a) and (b), show that the proposed search method is approximately 6.3 times faster than the spiral full search method, 5.5 times faster than the fast full search, and 2.4 times faster than the UMHexagon search in the JM source code. Figs. 6(a) and (b) show the rate distortion curves of the "Mother & Daughter" and "Foreman" sequence of QCIF, in which a frame rate of 10 Hz was used in this experiment. In terms of rate-distortion curve, the PSNR of the proposed method is approximately 0.4dB degraded in comparison to that of the fast full search, the spiral full search and the UMHexagon search methods. If the image quality is more important than the computation time, we can increase the search range into 16×16 in the proposed method, which improves the PSNR. Figs. 7 and 8 show the computation times and the rate distortion curves of four motion estimation methods for the two CIF sequences, "Paris" and "Foreman", with frame rates of 10 Hz. In the CIF sequences, the proposed method shows similar results to that of the QCIF sequences.

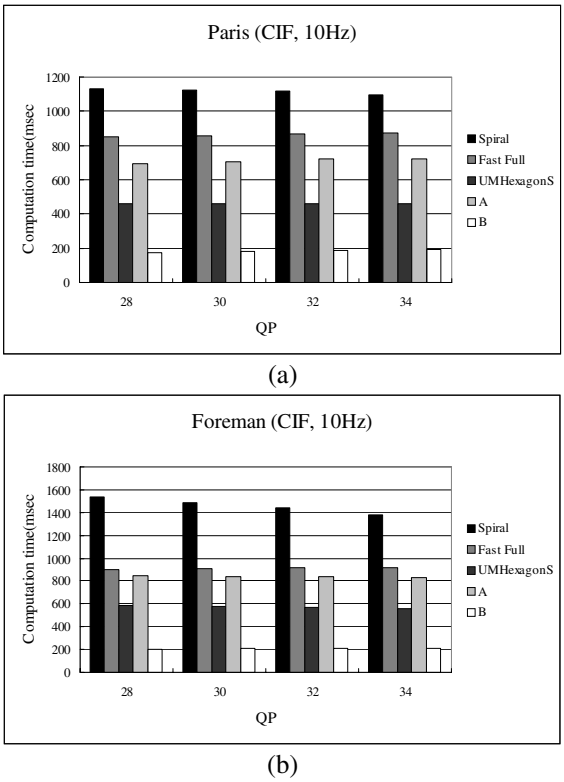
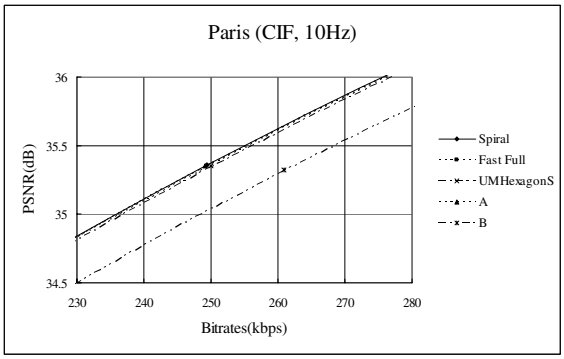


Fig. 7. The computation time of the spiral full search, the fast full search, UMHexagonS, and the proposed methods "**A**" and "**B**". Only the first frame is Intra-coded: (a) "Paris" sequence with a frame rate of 10Hz and CIF resolution, (b) "Foreman" sequence with a frame rate of 10Hz and CIF resolution.

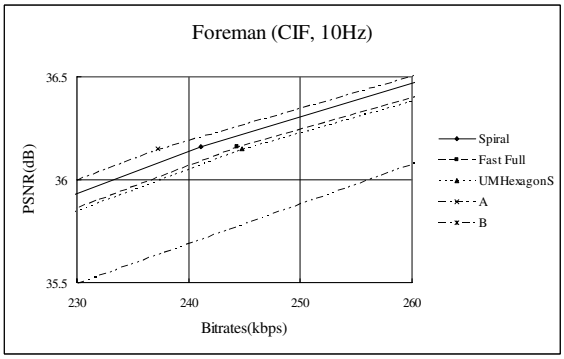
The "A" method always shows the best PSNR as shown in Figs. 6 and 8. The "B" method requires the smallest computation amount as shown in Figs. 5 and 7, whereas PSNR is degraded. Therefore, methods "A" or "B" can be applied alterenatively according to the importance of the image quality and the computation time.

4 Conclusions

The proposed motion vector search method utilizing hierarchical sum norm, which is developed through eqs (6)~(12) in this paper, can be easily applied to the JVT codec. The JVT codec, which is H.264 and MPEG-4 part-10 video coding standard, adopted variable blocks and quarter-pixel motion estimation/compensation in its codec. Therefore, the proposed method can be applied to the JVT codec to reduce the computational complexity of the search process with a very small loss in PSNR. The proposed one can also be applied to multiple reference frames without a serious loss of video quality. The hierarchical sum norm and the fast three points search for quarter-pixel motion estimation contribute to the reduction of computational complexity.



(a)



(b)

Fig. 8. Rate-distortion plots: (a) "Paris" rate-distortion plot corresponding to Fig. 7(a), (b) "Foreman" rate-distortion plot corresponding to Fig. 7(b)

References

[1] ITU Telecom. Standardization Sector, "Video Codec Test Model Near-Term, Version 10 (TMN10) Draft 1," *H.263 Ad Hoc Group*, April 1998.

[2] "Information Technology - Coding of Audio-Visual Objects Part2: Visual Amendment 1: Visual Extensions," *ISO/IEC JTC1/SC29/WG11 N3056*, Dec. 1999.

[3] T. Wiegand, Final draft international standard for joint video specification H.264, *in JVT of ISO/IEC MPEG and ITU-T VCEG*, JVT-G050, Mar. 2003.

[4] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommunications Conf.*, New Orleans, LA, pp. G.5.3.1-G.5.3.5., Nov. 1981.

[5] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, pp.1799-1808, Dec. 1981.

[6] R. Srinivansan and K. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun., vol. COM-33*, pp. 1011-1015, Sept. 1985.

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

[8] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation," *IEEE Trans. Image Processing*, vol. 9, no. 1, pp.105-107, Jan. 1995.
[9] M. Brunig and W. Niehsen, "Fast Full-Search Block Matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 241-247, Feb. 2001.
[10] http://bs.hhi.de/~suehring/tml/download/old_jm/jm73.zip
[11] G. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," *IEEE Signal Processing Magazine*, pp. 74-90, Nov. 1998.
[12] D. M. Young and R. T. Gregory, A Survey of Numerical Mathematics, New York: Dover, vol. 2, pp. 759-762, 1988.
[13] Z. Chen, P. Zhou, and Y. He, "Fast Motion Estimation for JVT," *JVT of ISO/IEC MPEG & ITU-T VCEG,* JVT-G016, Mar. 2003.