Efficient Fast Motion Estimation Method for H.264

Hyung-Su Jeon and Cheol-Jung Yoo

Chonbuk National University, Jeonju, Jeonbuk, South KOREA

Summary
Although motion compensation for H.264 produces high coding efficiency through its use of multiple reference frames and macro blocks, it also increases the overall coding complexity due to the motion search for various cases. This study proposes a fast motion estimation method to obtain a motion vector able to reduce the overall coding complexity and obtain the required computation in motion search by reusing the sum of absolute difference. Experimental results show that the proposed fast motion estimation method significantly reduces the performance time, with negligible degradation in video quality.

Key words:
motion vector, H.264, motion predication, motion estimation.

1. Introduction

In recent years, with the sharply increasing demand for real time information about digital video, the capacity of communication media has also been increasing greatly but not at a sufficient rate for transferring and storing high image quality video. Accordingly, a new H.264 coding standard, with better coding performance than existing MPEG-2 or H.263, has emerged[1][2].

H264's compression method has the advantage of providing various multimedia services, but it achieves motion vector predication of the macro block unit through compression by loss and damage of the image data. With its advantage of providing diversified multimedia services, the H.264 coding method estimates the motion vector in macro blocks for coding without the loss of image data.

The motion vector predication significantly increases the number of operations by repeating the motion predication in each mode of the macro block, so it is unreasonable to use the H.264 coding method in digital communication media such as mobile handsets. Therefore, a new decoding method with low complexity of memory performance is required to reduce power consumption in mobile handsets by decreasing the number of operations arising out of the H.264 motion predication. Several methods have been proposed to estimate the motion of current blocks by using the spatiotemporal characteristics among the motion vectors and deciding the search origin or the search pattern for the current blocks according to the estimated motion to increase the speed of the motion estimation[3].

These methods can obtain a suitable motion vector by estimating the pixels having small block matching errors with the motion vector via a check of all pixels in the search region of the reference frame for motion estimation, because they can increase the coding efficiency by decreasing the spatiotemporal overlapping; however, they involve excessive computations. Fast block matching methods have been suggested with spatiotemporal characteristics, but these introduce several problems such as improperly estimating the current motion block by erroneous judgment of motion information and selecting unsuitable search patterns in addition to excessive search points.

With this background of the problem of the demand for many operations in the course of carrying out an operation of the sum of absolute difference and determining the optimal motion vector for all cases in blocks to increase the coding efficiency in H.264, this study proposes to improve the motion search speed by reusing the sum of absolute difference for various blocks and decreasing the computations required for calculating the sum of absolute difference in the motion search of pixels with spatiotemporal characteristics.

This study comprises the following sections. Section 2 introduces the block size decision method and motion estimation methods. Section 3 proposes the new motion estimation method according to pixel and sub-pixel. Section 4 describes the result of comparative analysis on performance time and the image quality between the proposed method and the existing methods. Section 5 presents the study conclusion.

2. Background

2.1 Block Size Decision Method

In the H.264 standard, motion predication is conducted in block modes of various forms (16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4), as shown in Fig. 1. In H.264, repeated motion predication is performed in the same manner for these block modes. For the motion vector, the sum of absolute difference and the value of the motion vector are calculated, as is the position of the minimum value. The

This work was supported by grant No. R01-2004-000-10730-0 from the Basic Research Program of Korea Science & Engineering Foundation.

rate of distortion(RD) is calculated for mode selection from all block modes, including the intra mode and skip mode. Then, the block mode with the lowest RD is decided as the mode for the macro block [4][5].

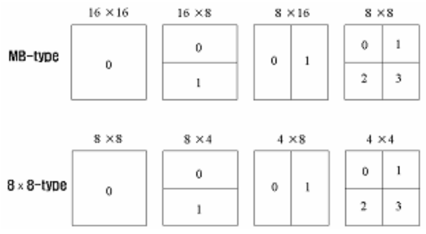


Fig. 1. Block Mode for Motion Predication

The macro block is calculated according to the following equations:

$$RD - value = Distortion + \lambda \times rate \quad (1)$$

$$Distortion = \sum_{i=1}^{16}\sum_{j=1}^{16}\left|IF(i,j) - PF(i,j)\right|^2 \quad (2)$$

Here, *IF(i,j)* is pixel value for the time of *(i,j)* for the current macro block, while *PF(i,j)* is that for the time of *(i,j)* for the restored macro block. In (1), *Distortion* is the sum of the square of the difference between the macro block for the original image and the restored macro block, as shown in (2). λ is a constant to be decided by quantization parameter, and *rate* is a necessary number of bits to represent the information of the macro block such as CBP (Coded Block Pattern), motion vector, block mode, etc. As described above, H.264 uses block modes, so one macro block may be divided into several sub-blocks. While small blocks can be used in parts with detailed and substantial motion, large ones are necessary in parts with simple and little motion.

2.2 Existing Motion Estimation Methods

The motion predication method is broadly used in video coding on the basis of the block matching method of dividing one image frame into the same blocks in certain size and finding the most similar block to the current block in the search region of the reference frame[6][7]. That is, the motion vector is the difference in position between the current block and the block having the smallest matching error from the current block.

For motion estimation, the following fast block matching methods have been suggested to estimate the motion vector: the diamond search (DS) method, adaptive search method for motion vector field, adaptive search method for estimated motion vector field, and low complexity motion estimation method using spatiotemporal correlation.

The DS method is based on the fact that the motion vector distribution of pixels forms a diamond shape centered on the search origin of the search region, which determines the center of the search pattern through the diamond pattern and forms a new diamond pattern, i.e., a large or small diamond pattern to the estimate motion vector[8].

The adaptive search method for the motion vector field uses as many search points in its predication of the motion vector for pixels as do the existing search methods which estimate the motion vector for pixels with a fixed search pattern and a fixed search origin, irrespective of the degree of motion of the current block. To solve this problem, this method estimates the motion vector for integer-pixels by estimating the motion of the current block and adaptively selecting a search pattern and search origin suitable for the estimated motion[9].

The adaptive search method for estimating the motion vector field uses temporal correlation and spatial correlation between the motion vectors for integer-pixels. This method decides the search origin and search pattern to estimate the integer-pixel motion of the current blocks by using motion vectors and the information of motion vectors. It improves the motion estimation speed with various threshold (*T)* values[3].

The low complexity motion estimation method using spatiotemporal correlation estimates the motion of the current block with spatiotemporal correlation between the motion vectors and adaptively decides the search origin and search pattern according to the estimated degree of motion to estimate the motion vector since it is difficult to identify whether the current block of the current frame is more influenced by temporal correlation or spatial correlation. To complete the motion estimation in the early stage, a simple *T* value is used to improve the motion estimation speed[10].

Regarding the problem of existing fast block matching methods, they start the predication of the integer-pixel motion vector from the search origin by always using a fixed search pattern, irrespective of the degree of integer-pixel motion for the current block since they cannot estimate the integer-pixel motion for the current block. Therefore, they require numerous search points in estimating the integer-pixel motion vector. If only spatial correlation is used, when the current block is largely influenced by the spatial correlation, the integer-pixel motion vector can be exactly estimated with a small number of search points. Otherwise, the motion of the current block cannot be exactly estimated, and therefore many integer-pixel search points have to be checked and a poor result is obtained in terms of the image quality in the predication of the integer-pixel motion vector. For methods which use only one of temporal correlation and spatial correlation among the motion vectors, an error may arise in estimating the degree of the integer-pixel motion for the current block due to the difficulty in identifying whether the current block is more influenced by temporal correlation or spatial correlation. In addition, though methods based on spatiotemporal correlation can estimate the motion of the current block more correctly, the number of computations is increased because complicated numerical expressions and conditions of calculation are used to estimate the degree of integer-pixel motion for the current block. Therefore, this study decreases the number of computations for various blocks by reusing the sum of absolute difference to solve the problem of excessive computations.

3. Proposed Motion Estimation Method

The goal of this method is to use the least number of computations and maintain the best condition of the current image quality in estimating the pixel motion for the current block through simple numerical expressions. Furthermore, this method compensates and restores the image based on the motion estimation result. As matching criteria to measure the degree of similarity to a certain block in the current block of the current frame and the search region of the previous frame, Mean Square Error (MSE), Mean Absolute Error (MAE), and Sum of Absolute Difference(SAD) are used.

$$MSE(u,v)=\left(\frac{1}{MN}\right)\sum_{u=1}^{M}\sum_{v=1}^{N}\left|X_t(x,y)-X_{t-1}(x+u,y+v)\right|^2 \quad (3)$$

$$MAE(u,v)=\left(\frac{1}{MN}\right)\sum_{u=1}^{M}\sum_{v=1}^{N}\left|X_t(x,y)-X_{t-1}(x+u,y+v)\right|^2 \quad (4)$$

$$SAD(u,v)=\left(\frac{1}{MN}\right)\sum_{u=1}^{M}\sum_{v=1}^{N}\left|X_t(x,y)-X_{t-1}(x+u,y+v)\right|^2 \quad (5)$$

In (3), (4), and (5), M, N indicate the width and length of the block, X_t the current frame, and X_{t-1} the previous frame.

3.1 Integer-pixel Motion Estimation Algorithm

Regarding the motion search for various block sizes of the H.264 standard, one 16×16 macro block is gradually divided in detail to investigate the motion vector. Therefore, 4 motion vectors in the 4×4 block are highly correlated with the motion vectors in the 8×8 block for each suitable position. Similarly, 4 motion vectors in the 8×8 block are highly correlated with motion vectors in the 16×16 macro block. Therefore, this study presents a search method able to get a motion vector in the macro block of such a structure.

If one same object is present in the macro block and the motion vectors in the 4×4 block are similar to each other, the motion compensation efficiency is increased in the upper blocks due to relative decrease in the weight of the motion vector overhead. On the contrary, if the motion vectors in the macro block are differently distributed, the motion compensation efficiency is decreased due to the increase of distortion in the upper blocks and the lower similarity between the motion vectors. Thus the proposed method changes the motion vector search method for the upper blocks by using such a distribution of motion vectors in the 4×4 block. The method is described in detail below.

The motion vector is obtained by using the existing fast motion search method in the 4×4 macro block, and the calculation is conducted in the manner shown in (6) for the upper blocks.

$$\left.\begin{aligned} d &= \frac{1}{n}\Big(\sum_{i=1}^{n} mv_b(i)-\overline{mv_b}\Big)^2 \\ \overline{mv_b} &= \frac{1}{n}\sum_{i=1}^{n} mv_b(i) \end{aligned}\right\} \quad (6)$$

Here, $mv_b(i)$ is the unit vector for the time of i in the given block and $\overline{mv_b}$ is the mean vector for n unit vectors to compose the given block. The d value reflects the distribution of unit vectors, and the similarity to unit vectors can be judged, such that with a larger d value the unit vectors give a better indication of each different direction. That is, the origin and pattern of the motion vector search are decided with the variance value d among the unit vectors.

If the values of all unit vectors are the same, d becomes 0, the unit vector is decided as the motion vector for the given block, and the motion search is not conducted. If d is less than the T, the motion vectors in the given block are apparently uniformly distributed and, therefore, the point of the lowest distortion is investigated through a SDSP search while $\overline{mv_b}$ of the mean vector for the unit vectors is the origin of SDSP. If d is larger than T, then LDSP and SDSP searches are conducted. That is, if d is larger than T, the motion vector is searched in the same search manner as in the DS method, except for the origin. With a larger dispersion of unit vectors, the similarity decreases between $\overline{mv_b}$ of the mean vector of the unit vectors and the motion vector, which necessitates a broader search.

In addition, upon searching in the various blocks, the number of operations for the sum of absolute difference can be greatly decreased by saving and reusing the sum of absolute difference to avoid repeated operation of such a sum. Therefore, upon completion of the motion search from the 4×4 block to the 16×16 block, operation of the sum of the absolute difference according to each position is saved and reused in the 4×4 unit block. Prior to calculating the sum of absolute difference for the given block, it is checked whether or not the sum of absolute difference saved in the 4×4 block is present in the block; if it is, the value is used, and the sum of absolute difference is operated and saved only for the other 4×4 blocks inside. To minimize the additionally memory capacity required by this operation of the sum of absolute difference to some degree, an integer-pixel motion search algorithm is used to limit the search region and thereby save the sum of absolute difference to a certain range, as shown in Fig. 2.

```
Input : Reference picture
Output : Integer-pixel motion vector
begin
{
  if(mode==4×4)) {      // For 4×4 block
    fact_motion_search();      // Fast motion search
  }
  else if(mode==4×8 or 8×4 or 8×8) {
                              // For 4×8, 8×4, 8×8 block
    submacroblock_mode_decision();
      // Use motion vector(unit vector) in 4×4 block.
      // Calculate d value of motion vector dispersion by using (6).
    if(D==0)  mv = average_vector;
           // Decide motion vector as $\overline{mv_b}$ .
   else {
     start_point = average_vector;  // Decide the origin as $\overline{mv_b}$ .
     diamond_search();          // Diamond search
     }
  }
   else if(mode==8×16 or 16×8 or 16×16)
       // For 8×16, 16×8, 16×16 block
   {
     submacroblock_mode_decision();
       // Use motion vector(unit vector) in 8×8 block.
      // Calculate d value of motion vector dispersion by using (6).
   if(D==0)  mv = average_vector;
         // Decide motion vector as $\overline{mv_b}$ .
   else {
       start_point = average_vector;
                              // Decide the origin as $\overline{mv_b}$ .
       diamond_search();   // Diamond search
     }
   }
}
end.
```

Fig. 2. Proposed Integer-pixel Motion Search Algorithm

3.2 Pixel Motion Estimation Algorithm

As the motion vector of the H.264 standard has an accuracy level to 1/4 pixel, a sub-pixel motion search is additionally required. Before the sub-pixel motion search is conducted, each sub-pixel at the 1/2 and 1/4 positions is obtained through image interpolation. Then, centered on the motion vector at the obtained integer-pixel position, the position having the minimum sum of absolute difference among the sub-pixels in the surroundings is searched to decide the final motion vector.

For this sub-pixel motion search, in JM (Joint Model) of the current reference codec for H.264, a two-step search is conducted, as shown in Fig. 3. A point having the minimum sum of absolute difference is searched for among 8 surrounding 1/2 pixel positions, including the present position centered on the integer-pixel unit motion vector (step 1). A point having the minimum sum of absolute difference is searched for again among 8 surrounding 1/4 pixel positions centered on the position having the minimum sum of absolute difference to fix it as final motion vector (step 2).

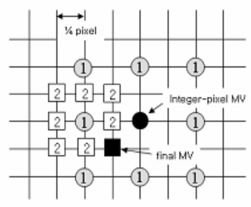


Fig. 3. Two Step Search Method

As shown in Fig. 3, the total number of search positions necessary for sub-pixel motion search is 17 in JM. Therefore, operation of the transformed sum of absolute difference in the 4×4 block is conducted 17 times according to the mode of each macro block. Thus the total required number of operations per macro block for the transformed sum of absolute difference to all kinds of block mode in a macro block is 1904.

Total operations for the transformed sum of absolute difference
= 17 × 16(MB16×16) + 8×2(MB16×8) + 8×2(MB8×16) + 4×4×4(MB8×8sub)
= 1904 times

Consequently, a fast motion search method for the sub-pixel motion search is necessary. Currently, the H.264 motion vector enhances the entropy coding efficiency by taking advantage of the fact that blocks around the left, upside and right top have a high spatial similarity to each other. According to the statistics, the probability that the motion vector is the same as the predicted motion vector (PMV) is 50%, and the probability that the motion vector is present at the SDSP position, centered on and including PMV, is about 70%.

Based on this probability model, an SDSP search centered on PMV is proposed for the sub-pixel motion search in this study. First, if PMV is present in a region ±0.7 from the integer-pixel unit motion vector point having the minimum sum of absolute difference, a comparison on the sum of absolute difference or the transformed sum is made between PMV and the integer-pixel unit motion vector, and the point having a minimum value is set as the origin of search. Otherwise, the integer-pixel motion vector is set as the origin of the search, the SDSP search is repeatedly conducted, and the final motion vector is decided. The search process for the proposed method is shown in Fig. 4.

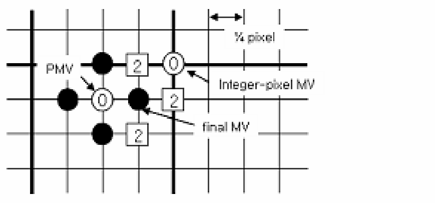


Fig. 4. SDSP-based Search Process

The proposed sub-pixel motion search algorithm, as shown in Fig. 5.

```
Input : Reference picture
Output : sub-pixel motion vector
// Obtain motion vector(mv) through integer-pixel motion search.
begin
{
  if(abs(mv-pmv)<= 0.7)     // For less than ±0.7 from mv and
                            // pmv(predicted motion vector)
  {
    pt=sum_abs_diff();
      // Calculate sad or transformed sad between pmv and mv.
    start_point = pt;
      // Set the position having a minimum value as the origin.
  }
  else
    start_point = mv;
      // Set the integer-pixel motion vector as the origin.
  do {
      small_diamond_search_pattern();    // Conduct SDSP.
  } while( (pt=sum_abs_diff()) == center of SDSP);
        // Repeat until pt becomes the center of SDSP.
  mv = center_point;
      // Decide the position of the center with motion vector.
}
end.
```

Fig. 5. Proposed Pixel Motion Search Algorithm

4. Experiment and Comparative Analysis

For comparison of the method proposed in this study with the experiment on the H.264 reference software, an original video is finally converted into a digital signal (values of 0 and 1) through encoding.

In this experiment, the source code for the H.264 Reference Software Version JM10.2 was used [11], and 6 kinds of video (Claire, Carphone, Foreman, News, Salesman, Silent) sized QCIF(176X144) were used in a Pentium-4 PC running at 3.0GHz with 2.0GB RAM. As shown in the experimental conditions of Table 1, the result was obtained with digital signals of the encoding results through the experiment on the proposed method, in addition to the reference software

Table 1: Encoder Experiment Conditions

Parameter	Value	Parameter	Value
Profile	77 (Main)	YUV Format	YUV 4:2:0
Level IDC	30 (Level 3.0)	B-Frame	Used
Entropy Coding	CABAC	Frame Skip	1
Reference Picture	5	Search range	16

4.1 Comparative Analysis Based on the Search Method

The existing DS and the improved method of MVFAST and its performance were compared on the proposed fast motion search method. For comparison of the performance of each method, a speed improvement was performed for the sum of absolute difference or the transformed sum in the 4×4 blocks in terms of the computations in comparison with a full search (FS), as shown in Table 2. S_{SAD} indicates the speed improvement for the integer-pixel motion search, while the transformed SAD indicates that for the sub-pixel motion search.

Table 2. Comparison of Speed Improvement according to Method

Method / Image	QP	DS		MVFAST		Proposed method	
		S_{SAD}	$S_{T\text{-}SAD}$	S_{SAD}	$S_{T\text{-}SAD}$	S_{SAD}	$S_{T\text{-}SAD}$
Claire	28	63.4	1	144.2	1	856.1	2.2
	32	63.8	1	142.4	1	869.9	2.2
Container	28	80.7	1	168.4	1	1134.8	3.1
	32	82	1	173.3	1	1167.8	3.1
Foreman	28	47.4	1	95.5	1	254.8	2.1
	32	48.5	1	98.2	1	275.4	2.1
News	28	75.2	1	152.1	1	752.2	2.8
	32	76.2	1	153.9	1	810.8	2.8
Salesman	28	66.8	1	139.1	1	722.7	2.2
	32	67.8	1	140.2	1	758.6	2.2
Silent	28	72.4	1	136.9	1	574	2.5
	32	72.6	1	139.3	1	621.1	2.5
Average		68.1	1	140.3	1	733.2	2.5

Also, concerning the transformed sum of absolute difference necessary for the sub-pixel motion search, the proposed method showed an average 2.7-fold speed improvement compared to the two step search, contrary to existing methods.

To examine the actual speed improvement when the proposed method is implemented in the H.264 coder, *TS* (Time Saving) is measured for the method subject to compare on the basis of fast FS (FFS) being implemented in JM, as shown in equation (7).

$$TS = \frac{T_{ME}(FFS) - T_{ME}(Selected\ method)}{T_{ME}(FFS)} \qquad (7)$$

Therefore increasing *TS* indicates that the performance time is relatively more reduced compared to FFS. Table 3 shows the speed improvement for the proposed method on this FFS. To compare the performance of the proposed method with the existing fast motion search method, it was compared with the fast motion search method [12] adopted in the JVT (Joint Video Team) standardization meeting (hereafter referred as JM_FME) and the DS method. JM_FME is already implemented in JM, and it minimizes the image degradation by applying a global search upon the integer-pixel motion search in order to solve the problem of a local minimum value for the existing fast motion search algorithms.

Table 3. Comparison of Performance Time between the Proposed Method and the Existing Algorithms

Pixel / Image	QP	Integer-pixel			Sub-pixel	
		DS	JM_FME	Proposed method	JM_FME	Proposed method
Claire	28	92.3	88.8	97.99	29.14	49.21
	32	91.8	89.44	97.84	34.02	54.97
Carphone	28	75.64	85.11	96.75	54.6	79.69
	32	73.68	87.17	95.15	36.58	69.84
Foreman	28	55.89	66.78	91.79	35.19	51.48
	32	60.51	72.6	88.14	23.82	43.6
News	28	77.8	77.15	88.54	33.13	69.42
	32	74.1	83.82	90.76	38.7	59.64
Salesman	28	91.21	93.65	97.49	44.35	56.78
	32	81.87	91.89	98.67	36.34	57.5
Silent	28	68.54	89.08	82.76	31.89	54.89
	32	72.54	77.15	86.94	23.2	52.31
Average		74.4	83.08	92.71	31.39	55.61

4.2 Comparative Analysis on the Image Quality

The video quality measurement method is divided into subjective and objective measurement methods. Significant factors for recognition of the video quality for subjective video quality measurement include time attention and up-to-date effects [13]. These factors involve difficulty in the correct and mathematical measurement of the visual video quality. The complexity and cost of the subjective video quality measurement method has led researchers to take an interest in automatic video quality measurement methods using algorithms. Video coding and video processing system developers depend upon an objective video quality measurement method. For comparison of the video quality, between coded and restored images, according to the popular video quality measurement method guaranteeing easy and rapid calculation, equation (9) PSNR(Peak Signal to Noise Ratio) is frequently used.

$$MAE(u,v) = \left(\frac{1}{MN}\right)\sum_{u=1}^{M}\sum_{v=1}^{N}\left|X_t(x,y) - X_{t-1}(x+u,y+v)\right|^2 \qquad (8)$$

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \qquad (9)$$

Here, *MSE* means the mean square error, *M* and *N* each indicates the width and length of the block, *Xt* the current frame, and *Xt-1* the previous frame.

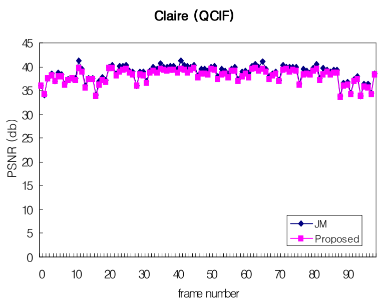
Table 4 shows the comparison between PSNR and bit rate according to each method. Since the rate control algorithm was not applied, each PSNR and bit rate based on each quantization coefficient was represented by image. For both DS and MVFAST, 0.04~0.08dB of loss was found on average with regard to PSNR, compared to FS, while the bits were increased by 3% and 1%, respectively, regarding the bit rate. On the contrary, a PSNR loss of about 0.07dB and a bit rate increase of about 2% were observed for the proposed method, compared to FS. For the Foreman image, the greatest performance degradation was found concerning the coding efficiency in the proposed method, as well as the existing one. This originated from the heterogeneous characteristic of the image, i.e., the faster motion than the other images and the *T* value for the proposed method needs to be adjusted to be further suited to such characteristic.

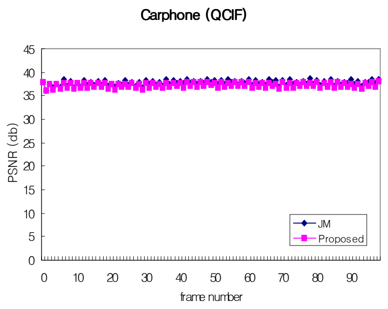
Table 4. Comparison on PSNR and Bit Rate between the Proposed Method and the Existing Algorithms

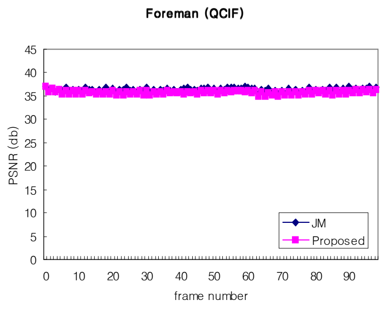
Method / Image	QP	FS		MVFAST		Proposed method	
		PSNR [db]	bitrate [Kbps]	PSNR [db]	bitrate [Kbps]	PSNR [db]	bitrate [Kbps]
Claire	28	33.86	55.78	33.84	55.69	33.84	55.84
	32	30.44	27.11	30.12	27.17	30.28	27.64
Container	28	35.97	23.59	35.89	23.54	35.87	24.13
	32	33.19	13.23	33.14	13.1	33.24	13.29
Foreman	28	35.94	70.28	35.78	72.25	35.84	74.12
	32	33.16	43.07	33.16	45.16	33.12	46.51
News	28	36.64	44.57	36.57	45.75	36.61	45.1
	32	33.66	27.54	33.62	28.44	33.54	27.94
Salesman	28	34.78	57.48	34.86	57.45	34.78	57.37
	32	31.4	27.44	31.41	27.51	31.56	27.38
Silent	28	35.6	57.84	35.16	58.9	35.74	59.03
	32	32.95	34.95	33.42	35.36	32.89	35.49

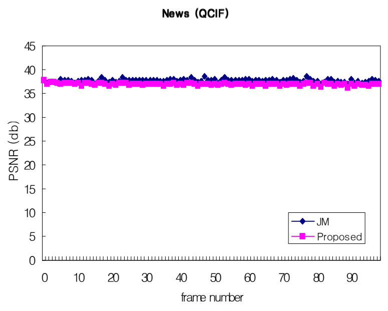
In Tables 3 and 4, the PSNR value is higher in either the reference software or the proposed method. The PSNR value showed little variation. Specially, the image quality was improved compared to that of the existing method when there was little change in picture motion.

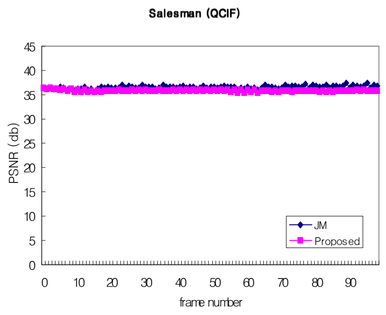
Fig. 6 graphically presents the FS measured for 6 experimental images and the proposed method. The image quality shows little variation because of the similar PSNR values.











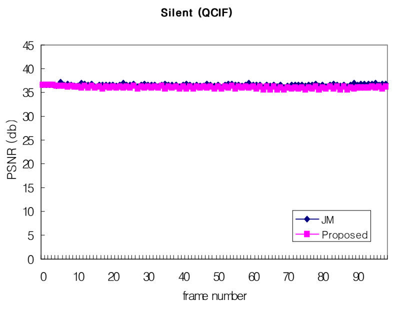


Fig. 6. Comparison on PSNR among Images

5. Conclusion

Although H.264 is used for digital data broadcasting, mobile handsets, etc., due to its better coding rate and image quality in comparison with existing coding standards, the encoding and decoding methods have been complicated by the development of new coding methods. H.264, however, introduces the problem of a bottleneck state to increase the overall encoding complexity, requires excessive decoding time, increases memory bandwidth, and ultimately impairs system efficiency because the sum of absolute difference is obtained for all cases in blocks in order to determine the optimal motion vector as a means of increasing the coding efficiency.

Therefore, this study attempted to estimate the motion vector with the spatiotemporal characteristics of a video frame and to propose a motion search method to reduce the number of repeated operations by reusing the sum of absolute difference as calculated in various blocks upon the motion search.

In the experimental results, the proposed method reduced the repeated calculation of the sum of absolute difference and accordingly the performance time, played an important role in decreasing the complexity of the H.264 encoder, and displayed great speed improvement, without image degradation, while searching for the motion vector, compared to the existing fast motion search method.

These results confirm the suitability and efficiency of this fast motion estimation method for application to VOD and image communication. Furthermore, the method may be offered in mobile settings, by encoding the video with high efficiency and reducing the motion compensation time, and implemented in the broadcasting of mobile settings such as ground wave DMB and satellite DMB. Finally, it will enable video conferencing, VOD and streaming service by using wired networks (ISDN, Ethernet, LAN, DSL, MODEM, etc.) with high efficiency.

Although the motion search method proposed in this study shows a better result in terms of the image quality and speed in small images or images with regular motion, it causes degradation in images with large motions and estimates the motion vector too slowly. Further studies are therefore necessary to solve this problem. For high-speed video decoding in digital communication media like mobile handsets, continuous research on efficient memory control in mobile handsets is required to solve the low memory problem.

References

[1] C. Kim and J-N Hwang, "Fast and Automatic Video Object Segmentation and Tracking for Content-Based Applications," IEEE Transaction on Circuits and Systems for Video Technology, Vol. 12, No. 2, pp. 122-129, 2002.

[2] M D Walker, M Nilsson, T Jebb, and R Turnbull, "Mobile Video-Streaming," BT Technology Journal, Vol 21 No 3, pp. 192-202, 2003.

[3] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation," Visual Communication and Image Processing, Proceeding of SPIE, Vol. 4310, pp. 883-891, 2001.

[4] Jeyun Lee, Woongil Choi, Byeung Woo Jeon, and Minsoo Suk, " Fast Motion Estimation and Mode Decision for Variable Block Sizes Motion Compensation in H.264," The Institute of Electronics Engineers of Korea, Journal of IEEK, Vol. 40, No. 4, pp. 49-59, 2003.

[5] In-Cheol Jeong and Jong-Ki Han, " An Efficient Algorithm for Motion Estimation in H.264," Journal of KICS, Vol. 29, No. 12C, pp. 1669-1676, 2004.

[6] P. Kuhn, Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwer Academic Publishers, 1999.

[7] HyoSun Yoon, GueeSang Lee, " An Adaptive Block Matching Algorithm Based on Temporal Correlations," Journal of KIPS, Vol. 9-B, No. 2, pp. 199-204, 2002.

[8] J. Y. Than, S. Raganath, and A. A. Kassim, "A Novel Unrestricted Center-Based Diamond Search Algorithm for Block Motion Estimation," IEEE Transaction on Circuits and Systems for Video Technology, Vol. 8, No. 4, pp. 369-377, 1998.

[9] Prabhudev Irappa Hosur and K. K. Ma, "Report on Performance of Fast Motion Using Motion Vector Adaptive Search Technique," ISO/IEC/JTC1/SC29/WG11 M5453, 1999.

[10] Hyo-Sun Yoon, Mi-Young Kim, and Guee-Sang Lee, "Low Complexity Motion Estimation Based on Spatio-Temporal Correlations," Journal of KISS, Vol. 31, No. 9, pp. 1142-1149,, 2004.

[11] H.264/AVC Reference Software Version JM10.2, http://bs.hhi.de/～suehring/tml/index.htm, Joint Video Team, 2003.

[12] Zhibo Chen, Peng Zhou, Yun He, and Yidong Chen, "Fast Integer Pel and Fractional Pel Motion Estimation for JVT," JVT Document, JVT-F017, Dec. 2002.

[13] N. Wade and M. Swanston, Visual Perception: An Introduction, 2nd Edition, Psychology Press, 2001.

[14] H-S Jeon, H-M Noh, C-J Yoo, and O-B Chang, "Design of H.264/AVC-Based Software Decoder for Mobile Phone," ICCSA 2006, LNCS 3482, pp. 188-197, 2006.

[15] H-S Jeon, C-J Yoo, and O-B Chang, "On Motion Compensation of H.264/AVC Software Decoder," MITA 2006, pp. 402-405, 2006.

[16] ITU-T, ITU-T Rec. H.264 Advanced Video Coding, 2003.

[17] ISO/IEC 14496-10, Information Technology - Coding of Audio-Visual Objects - Part 10: Advanced Video Coding, 2003.

Hyung-Su Jeon received the B.S. degree in Mathematics from Chonbuk National University, Jeonju, Korea in 1992, and M.S. and Ph.D. degrees in Computer Science from Chonbuk National University, Jeonju, Korea in 1997 and 2007. His research interests include multimedia application, image processing, and transcoding etc.

Cheol-Jung Yoo received the B.S. degree in Computer and Statistics from Chonbuk National University, Jeonju, Korea, in 1982, M.S. degrees in Computer and Statistics from Chonnam National University, Kwangju, Korea, in 1985, and Ph.D. degree in Computer and Statistics from Chonbuk National University, Jeonju, Korea, in 1994. He is currently an associate Professor, Department of Computer Science, Chonbuk National University, Jeonju, Korea. His research interests are software development process, software quality, component software, software metrics, software agent, GNSS, GIS, education engineering, and recognition science etc.