

Short Paper

A New Two-Path Search Algorithm for Block Motion Estimation of Video Data

CHIN-CHEN CHANG, LIN-LI CHEN AND TUNG-SHOU CHEN*

Institute of Computer Science and Information Engineering

National Chung Cheng University

Chiayi, Taiwan 621, R.O.C.

E-mail: {ccc, cll84}@cs.ccu.edu.tw

* *Department of Information Management*

National Taichung Institute of Technology

Taichung, Taiwan 404, R.O.C.

E-mail: ts.chen@taiwan.com

In this paper, we propose a new fast search algorithm for block motion estimation (BME). This algorithm employs two techniques: the center-biased technique and the two-path technique. The former is applied to stop the search procedure if the block currently being processed is stationary. Otherwise, the proposed algorithm conducts the two-path search to perform BME. The search result of the traditional one-path search will probably not reach the global minimum. It is clear that two-path search can reduce this bias. The experimental results show that the computational complexity of the proposed algorithm is low and that its image quality is acceptable no matter whether the input video contains slow motion only or a large amount of activity.

Keywords: two-path search, center-biased concept, motion estimation, block matching process, video coding

1. INTRODUCTION

The amount of video size is always large. The limited network bandwidth cannot support the transmission of video data. It is a serious problem, especially for transmitting video data on a low-bit-rate network. Now many standards have been developed for different applications in an attempt to solve the above problem, such as CCITT H.261, H.263, and MPEG [1]. These standards have a common feature with regard to the reduction of the temporal redundancies. We call this feature block motion estimation (BME). BME plays a very important role in video compression [1].

Consider the t -th frame of a video. It is denoted as frame(t). A BME search algo-

Received October 4, 1999; revised April 17 & July 10, 2000; accepted September 1, 2000.
Communicated by Yung-Nien Sun.

rithm first divides this frame (i.e., the image) into a large number of small blocks of size $n \times n$ pixels. For each block x in frame(t), BME searches the previous frame, i.e., frame($t-1$), for the block closest to x . The search space of x in frame($t-1$) is called the *search window* of x , and the displacement between the position of x in frame(t) and the position of the closest block in frame($t-1$) is called the *motion vector* of x . The goal of a fast search algorithm for BME is to find the motion vector as quickly as possible.

The most intuitive algorithm for BME is FS. It employs exhaustive search to check every block in the corresponding search window of frame($t-1$), and calculates the distortion of each block from x . The distortion between two image blocks could be measured by means of their *mean absolute error* (MAE) or *mean squared error* (MSE). To accelerate the whole process, most approaches apply MAE to perform measurement.

Let $[i, j]$ denote the coordinate in the upper-left corner of x in frame(t), and let y^{uv} indicate the checked block in frame($t-1$) whose coordinate in the upper-left corner is $[i+u, j+v]$. To simplify the following description, we regard $[i, j]$ as the position of x and (u, v) as a *checkpoint* for x . The corresponding checked block of (u, v) is y^{uv} . Assume that the maximal movement in both the vertical and horizontal directions is $\pm w$ pixels. That is, $-w \leq u \leq w$ and $-w \leq v \leq w$. Then, in FS, there are totally $(2w+1)^2$ blocks to be checked. Suppose the block closest to x is $y^{u'v'}$; that is, $\text{MAE}(x, y^{u'v'})$ is the minimum of all $\text{MAE}(x, y^{uv})$ s. The displacement (u', v') is, then, the *motion vector* of x . BME uses this motion vector to replace x .

Full search (FS) is the most intuitive algorithm for BME. Since FS is applied to each pixel in each block and conducts exhaustive search to find the motion vector, FS can obtain the best image quality. Its computational complexity is, however, relatively high. This phenomenon limits the performance of video coding. Many fast search algorithms have been proposed to solve the above problem in recent years, such as three-step search (TSS) [2], new three-step search (NTSS) [3], four-step search (4SS) [4], and block-based gradient descent search (BBGDS) [5]. These algorithms always divide the whole BME process into a number of steps. In each step, only a fixed number of blocks are checked in the search window of the previous frame, and one of these blocks is chosen as the origin of the next step. A block checked in the above process is called a *checked block*. The block chosen in the above process is called the *candidate block*. The candidate block is, in fact, the closest one among all the checked blocks to the block currently being processed, moreover, the candidate block in the last step is, then, the result of the BME search algorithm. Note that all the above search algorithms choose only one candidate block in each step. Hence, we call them *one-path search algorithms*.

The selection of candidate blocks is critical for one-path search algorithms. If the selection is inappropriate, then the next step taken by of the search algorithm may be far off track. In fact, we find that, in many cases, the closest checked block and the second closest one are very similar, but that their search directions are totally opposite. Thus, our new fast search algorithm chooses two candidate blocks in each step and we call our new idea the *two-path search* (2PS) algorithm. We take these two search paths to avoid possible incorrect selection of candidate blocks that may occur in one-path search algorithms. In addition, 2PS also employs the center-biased concept proposed by Zeng and Liou [6] to encode an inactive block. 2PS makes different stop decisions in different situations based on the two-path search and center-biased concept. Therefore, 2PS has the ability to find a balance between image quality and computational complexity.

2. Two-Path Search Algorithm (2PS)

All the traditional algorithms have one thing in common — they choose only one candidate point, i.e., the closest checkpoint in each step. In fact, it is possible for the closest and the second closest checkpoints to be very close to each other. In this case, it is a pity to consider only the first candidate point while ignoring the second one. Of course, more candidate points can be considered in each step to remedy the above problem. It is true that considering more candidate points is helpful for image quality, but choosing too many candidate points in each step may be going overboard. Thus, in this paper, a two-path search (2PS) algorithm is proposed. The number of checkpoints is also nine for each step and each path in 2PS. Among these checkpoints, 2PS selects the closest and the second closest checkpoints as the basis for the next step. We call the closest and the second closest checkpoints the first and second candidate points, respectively. 2PS endeavors to find a compromise between less computational complexity and better image quality based on the choice of these two points.

2PS also employs the center-biased concept to select the motion vector for inactive blocks. In previous works, experiments have shown that most blocks in a frame are stationary. This means that most motion vectors are near the center of the search window. Thus, in 2PS, the checkpoints of each path are always within a 3×3 window as with BBGDS. In the first step, 2PS verifies the nine checkpoints in the 3×3 window located near the center of the search window. If the closest checkpoint is found to be exactly at the center, then the procedure stops and outputs the displacement (0,0). This is called the *first-step-stop*. The steps in 2PS are described in detail in the following.

2.1 2PS Step One

Consider an image block x of frame(t). Let $[i, j]$ denote the coordinate in the upper-left corner of x , and let y^{uv} denote a checked block of x in frame($t-1$) whose coordinate in the upper-left corner is $[i+u, j+v]$. That is, (u, v) is a checkpoint of x . (0,0) denotes the center of the search window. It is also a checkpoint. Assume that $-w \leq u \leq w$ and $-w \leq v \leq w$. In the first step, 2PS examines the nine checkpoints in the 3×3 window whose center overlaps with the center of the search window of x . The nine checkpoints are (0,0), (1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), and (1,-1). Each checkpoint represents a search direction. If the minimum MAE for x among these checkpoints is found to be at (0,0), then 2PS stops and takes (0,0) as the motion vector of x . Otherwise, 2PS locates the closest and the second closest checkpoints for x as the first and the second candidate points F_1 and S_1 , respectively. F_1 and S_1 denote the directions of the two-path search. Here, the subscript of F_1 and S_1 indicates that they belong to the first step.

2.2 2PS Step Two

In the second step, 2PS checks two 3×3 windows each with nine points, with one window centered on F_1 and the other centered on S_1 . 2PS checks the eighteen points and finds the closest and the second closest checkpoints (i.e., the first and the second candidate points) F_2 and S_2 , respectively. Note that, though eighteen checkpoints need to be

verified in the second step, some of the checkpoints have already been examined in the first step. Those MAE's do not need to be calculated again.

The checkpoints of each path in 2PS are always within a 3×3 window. This is due to application of the center-biased concept. 2PS is, thus, suitable for searching the motion vectors of stationary blocks. For the active blocks, 2PS employs another strategy. This strategy is called *jump-stop* and is described as follows. Based on the first two steps, 2PS has four candidate points, F_1 , S_1 , F_2 , and S_2 . F_1 and S_1 are the two points in the 3×3 window which is located at the center of the search window. Let B_0 denote $(0,0)$. It is the base point of F_1 and S_1 . Thus, F_1-B_0 and S_1-B_0 represent the two search directions generated in the first step. Let d'_1 and d''_1 indicate F_1-B_0 and S_1-B_0 , respectively. Similarly, consider the second step in 2PS. Suppose F_2 is established in the 3×3 window centered on B'_1 , and that S_2 is settled in the 3×3 window centered on B''_1 . Here, B'_1 is F_1 or S_1 , and B''_1 is also F_1 or S_1 . B'_1 and B''_1 are, in fact, the base points of F_2 and S_2 . Let d'_2 and d''_2 indicate $F_2-B'_1$ and $S_2-B''_1$, respectively. They are the two search directions generated in the second step. For the block currently being processed, x , these four search directions show the trace of the motion direction.

There are four possible combinations (d'_1, d'_2) , (d'_1, d''_2) , (d''_1, d'_2) , and (d''_1, d''_2) for these four search directions according to the search order. The priorities of the four combinations are in descending order. We check d'_1 and d'_2 first. If d'_1 and d'_2 have the same direction, i.e., d'_1 equals d'_2 , then the motion of x has higher probability in this direction. Hence, 2PS sets F_2 to be the origin and employs TSS to discover the motion vector of x . Note that, in this situation, TSS focuses on the new origin. 2PS applies TSS to find the motion vector of x since x is an active block if it satisfies the above condition; moreover, TSS is an effective algorithm for searching the motion vector for active blocks. The result of TSS is, then, the final result of 2PS. In the same way, if the two directions of any one of the four pairs are the same, then 2PS selects the corresponding candidate point (i.e., F_2 or S_2) as the new origin and employs TSS to find the motion vector of x . If none of the jump-stop cases happen, then 2PS will proceed to Step 3.

2.3 2PS Step Three

The third step of 2PS is similar to the second step of 2PS. 2PS also checks two 3×3 windows in the third step. One is clustered around F_2 , and the other is collected around S_2 . 2PS verifies the eighteen checkpoints and finds the first and the second candidate points, F_3 and S_3 , respectively. Among these eighteen checkpoints, some have been examined in the first or second steps. We do not need to calculate their MAE's for x again. 2PS has four candidate points according to the second and the third steps. They are F_2 , S_2 , F_3 , and S_3 . Using the same strategy as in the second step, 2PS checks the different cases of jump-stop in the third step. Let d'_2 and d''_2 be the search directions of the second step, which have been defined in Section 3.2. Consider the third step. Suppose F_3 is located in the 3×3 window centered on B'_2 , and that S_3 is located in the 3×3 window centered on B''_2 . Here, B'_2 and B''_2 must be F_2 or S_2 . Then $F_3-B'_2$ and $S_3-B''_2$ are the two search directions generated in the third step. Let d'_3 and d''_3 denote $F_3-B'_2$ and $S_3-B''_2$, respectively. Now we have four search directions d'_2 , d''_2 , d'_3 , and d''_3 . There are also four possible combinations among them according to the search order. They are (d'_1, d'_2) , (d'_1, d''_2) , (d''_1, d'_2) , and (d''_1, d''_2) . If the directions of one of the four pairs are identical, then 2PS

will choose the corresponding candidate point (i.e., F_3 or S_3) as the origin and employ TSS to find the motion vector of x . This is the jump-stop of the third step, and it is similar to that of the second step. If no jump-stop cases occur, 2PS takes F_3 as the motion vector of x .

3. EXPERIMENTAL RESULTS

To illustrate the efficiency of 2PS, we have tested some BME search algorithms using videos. Five CIF video sequences ‘Claire,’ ‘Football,’ ‘Table Tennis,’ ‘Salesman,’ and ‘Miss American’ were used in our experiments. Each test video contains 29 frames numbered from 0 to 28. Each frame had 352×288 pixels. Our experiments considered the similarities in adjacent frames only. The BMEs among skipped frames were not estimated in this paper. The block size was 8×8 pixels in our experiments, and the size of search window was defined as 15×15 pixels, i.e., $w = 7$. We conducted MAE to measure block matching. All these experiments were performed on an IBM personal computer with an Intel Pentium II 233 CPU.

In our experiments, we compared 2PS with five traditional search algorithms, FS, TSS, NTSS, 4SS, and BBGDS. We focused on both the computational complexity and the image quality obtained using the BME search algorithms. There are two kinds of measurements for the estimation of computational complexity. One is based on the number of checkpoints for each processed block, and the other is based on the execution time. We list the average number of checkpoints in our experiments in Table 1 and show the average execution time in Table 2. From Table 1, it is observed that 2PS always used fewer checkpoints than FS, TSS, NTSS, and BBGDS for the same video. As for 4SS, the number of checkpoints in 2PS was also smaller than that for 4SS except for the videos ‘Football’ and ‘Miss American.’ The results show that 2PS was more efficient than the other algorithms. This phenomenon is also shown in Table 2 in terms of the execution time. As Table 2 shows, the execution time for 2PS was always less than that for FS, TSS, or NTSS. Compared with 4SS and BBGDS, 2PS had less execution time in most cases.

To evaluate the image quality, we compared FS, TSS, NTSS, 4SS, BBGDS, and 2PS in terms of the mean squared error (MSE). The experimental results for averaged image quality are listed in Table 3. From this table, we find that 2PS did not have the best image quality in all cases. The image quality of 2PS was always poorer than that of FS, TSS, or NTSS for the same video. However, 2PS usually had a lower MSE value than did 4SS and BBGDS.

Table 1. The average numbers of checkpoints for the videos ‘Claire,’ ‘Football,’ ‘Miss American,’ ‘Table Tennis,’ and ‘Salesman’ obtained by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

	FS	TSS	NTSS	4SS	BBGDS	2PS
Claire	225	25	18.327	4.188	9.849	2.643
Football	225	25	23.163	4.895	12.238	5.878
Table Tennis	225	25	19.327	4.385	10.640	3.672
Salesman	225	25	19.150	4.260	10.636	3.869
MissAmerica	225	25	23.506	4.735	12.127	6.029

Examining the data in Tables 1, 2, and 3 carefully, we find that the numbers of checkpoints of FS, TSS, and NTSS were always greater than those of 4SS, BBGDS, and 2PS. This phenomenon also applied to execution time. On the other hand, the image quality of FS, TSS, and NTSS was always better than that of 4SS, BBGDS, and 2PS for the same video. Based on these observations, it can be concluded that FS, TSS, and NTSS are designed to maintain the image quality of videos. We call them *quality-oriented* BME algorithms. As for 4SS, BBGDS, and 2PS, they pay more attention to the efficiency of BME search. Hence, we call them *speed-oriented* BME algorithms.

Table 2. The average execution times (second) for the videos ‘Claire,’ ‘Football,’ ‘Miss American,’ ‘Table Tennis,’ and ‘Salesman’ used by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

	FS	TSS	NTSS	4SS	BBGDS	2PS
Claire	21.192	0.204	0.179	0.116	0.100	0.063
Football	22.862	0.205	0.213	0.133	0.116	0.131
Table Tennis	22.750	0.205	0.184	0.121	0.103	0.085
Salesman	22.751	0.205	0.182	0.118	0.105	0.090
MissAmerica	22.236	0.204	0.217	0.134	0.117	0.137

Table 3. The average MSE values for the videos ‘Claire,’ ‘Football,’ ‘Miss American,’ ‘Table Tennis,’ and ‘Salesman’ obtained by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

	FS	TSS	NTSS	4SS	BBGDS	2PS
Claire	2.846	3.155	2.932	3.495	2.940	3.443
Football	221.481	290.516	288.014	418.324	530.199	392.250
Table Tennis	42.775	54.236	49.239	65.867	74.975	67.384
Salesman	15.589	17.050	16.654	21.807	18.385	21.148
MissAmerica	6.563	7.136	6.973	8.737	7.180	8.246

Consider the motion activity on videos. We know that ‘Claire’ contains slow motion only, and that ‘Football’ contains fast motion. Both ‘Claire’ and ‘Football’ are special types of videos. All the other videos are typical compared with ‘Claire’ and ‘Football.’ Thus, in the following discussion, we will focus on these two videos. For ‘Claire,’ the image quality of 2PS was close to that of FS. Thus, in this situation, the computational complexity is an important factor when comparing BME search algorithms. 2PS required the minimum execution time for ‘Claire’ in our experiments. Its execution time was only 0.30% (i.e., $0.063/21.192$) of that for FS, 30.88% (i.e., $0.063/0.204$) of that for TSS, 35.20% (i.e., $0.063/0.179$) of that for NTSS, 54.31% (i.e., $0.063/0.116$) of that for 4SS, and 63% (i.e., $0.063/0.100$) of that for BBGDS. We can saved more than 0.037 (i.e., $0.100-0.063$) seconds per frame when we applied 2PS to encode a video. Suppose there is a video containing slow motion only like ‘Claire,’ and that it consists of 86400 frames. If we broadcast this video at 24 frames per second, then it takes one hour to play. It is not

a long video in general. We will save more than 50 minutes if we apply 2PS to encode this video. This is a significant contribution of 2PS.

In addition, for ‘Football,’ the image quality of 2PS was the best of all the speed-oriented algorithms. Here the speed-oriented algorithms are 4SS, BBGDS, and 2PS. Note that the MSE values of ‘Football’ were much larger in our experiments. This means that the encoded image quality was extremely poor. In this situation, the image quality of an encoded video is most important. None will interest a video coding method whose image quality is unacceptable, even if its execution time is the shortest. For ‘Football,’ 2PS had the best image quality of all the speed-oriented algorithms although the execution time of 2PS was longer than that of BBGDS. This shows that, for sport videos, 2PS has the ability to maintain image quality by checking more checkpoints.

To compare the BME search algorithms in more detail, we show the experimental results for the frames in the videos ‘Claire’ and ‘Football’ in Figs. 1-6. Fig. 1 plots the numbers of checkpoints of each frame in ‘Claire.’ It can be found that 2PS had the smallest number of checkpoints in each frame. Fig. 2 displays the execution times of search algorithms for frames in ‘Claire.’ It is apparent that 2PS was always the quickest algorithm in Fig. 2. The MSE values of the frames in ‘Claire’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS are shown in Fig. 3. From this figure, we note that all the results are very close. It is difficult to differentiate between FS, TSS, NTSS, 4SS, BBGDS, and 2PS in this situation. Therefore, from Figs. 1-3, it is observed that, for slow motion videos, 2PS is an efficient search algorithm for BME and, moreover, that the image quality of 2PS is acceptable.

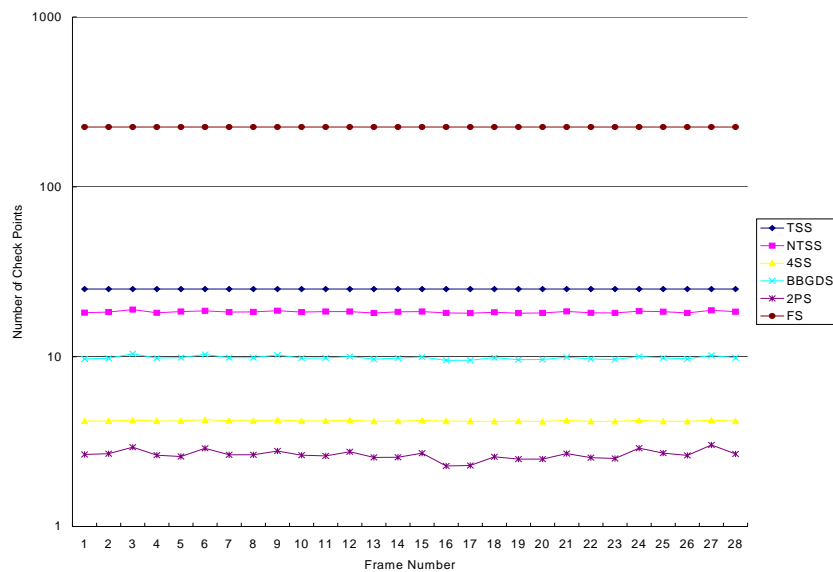


Fig. 1. The numbers of checkpoints for the frames in the video ‘Claire’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

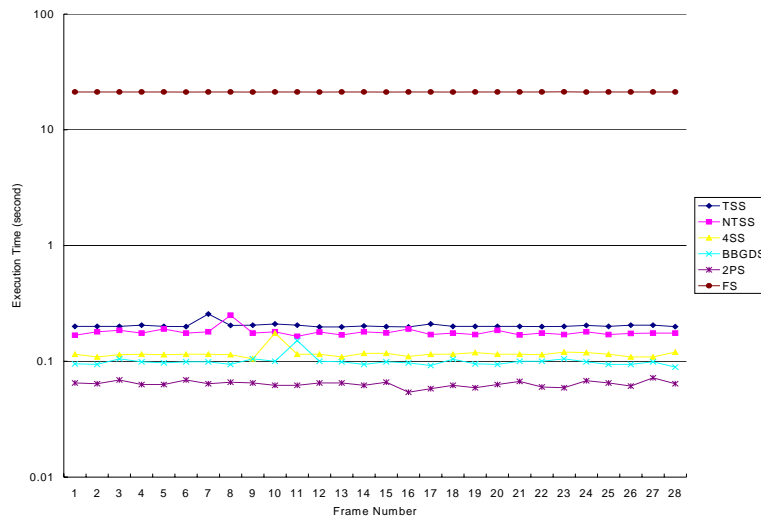


Fig. 2. The execution times for the frames in the video ‘Claire’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

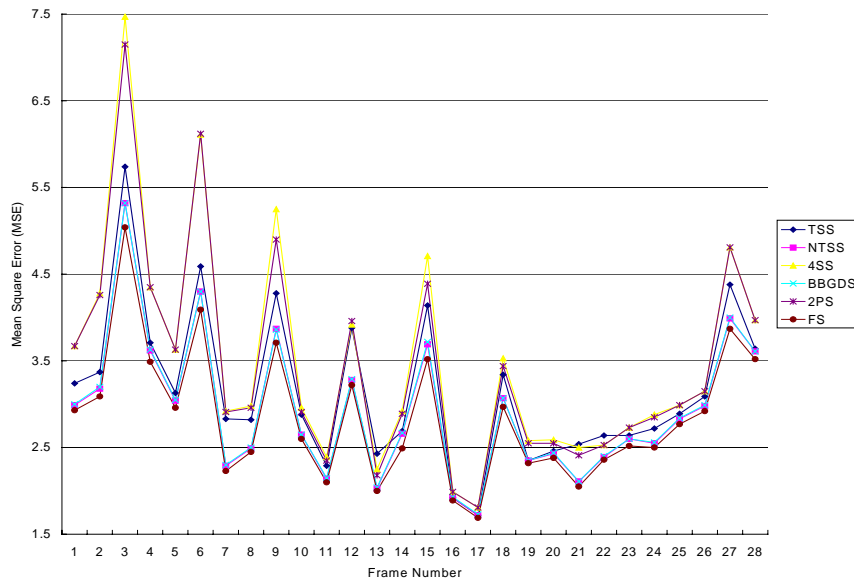


Fig. 3. The mean squared errors (MSE) of the frames in the video ‘Claire’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

‘Football’ is a sports video. The number of checkpoints for each frame is shown in Fig. 4. The number of checkpoints for 2PS is close to that for 4SS, and is smaller than that for FS, TSS, NTSS, or BBGDS. Fig. 5 displays the execution times of the BME

search algorithms for frames in ‘Football.’ 4SS, BBGDS, and 2PS had similar execution times as shown in this figure. 2PS was usually slower than 4SS and BBGDS. This is because 2PS utilizes more complex computational to maintain image quality. This phenomenon can be observed in Fig. 6. In Fig. 6, the MSE values of the frames in ‘Football’ are plotted. Based on these data, we note that 2PS has the best image quality among the speed-oriented algorithms in the experiments.

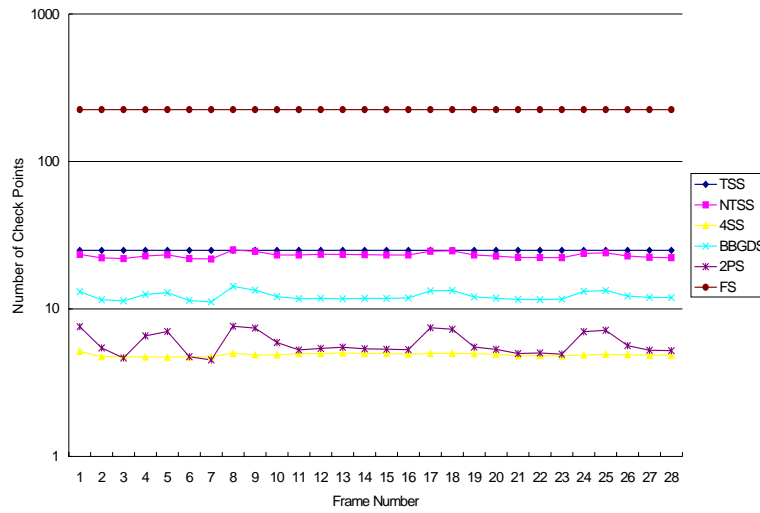


Fig. 4. The numbers of checkpoints of the frames in the video ‘Football’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

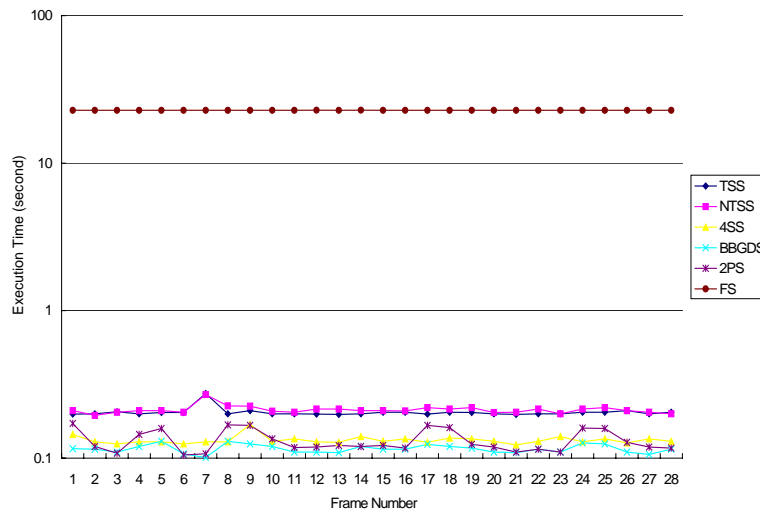


Fig. 5. The execution times for the frames in the video ‘Football’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

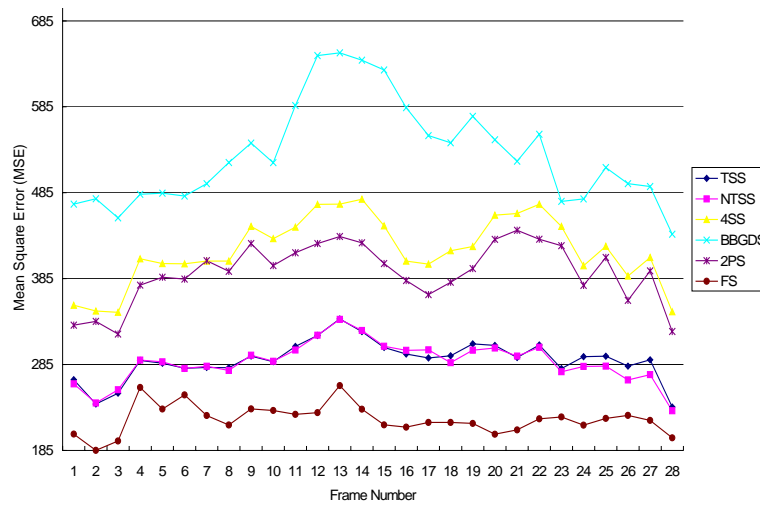


Fig. 6. The mean squared errors (MSE) of the frames in the video ‘Football’ processed by FS, TSS, NTSS, 4SS, BBGDS, and 2PS.

4. CONCLUSIONS

A new fast search algorithm has been proposed for BME in this paper. The traditional BME algorithms use only one candidate point in each step. The proposed algorithm employs two candidate points, namely, the closest and the second closest check-points, to reduce the possibility of making an incorrect choice of the candidate point for the traditional BME algorithms. We call this new algorithm the two-path search (2PS) algorithm. Besides the two search paths, 2PS also employs the center-biased concept to differentiate between stationary and active blocks. Thus, 2PS is more flexible than the previously proposed algorithms. Based on our experimental results, it can be observed that, for slow motion videos, the execution time for 2PS is the shorter than that for FS, TSS, NTSS, 4SS, or BBGDS and, furthermore, the image quality for 2PS is acceptable. The MSE values of 2PS are close to those of the other BME algorithms. As for large motion videos, 2PS has the ability to maintain acceptable image quality automatically by applying more complex computation. In conclusion, 2PS is a speed-oriented algorithm. Its computational complexity is low, and its image quality is always acceptable no matter whether the input video is composed of slow motion or fast motion activity.

REFERENCES

1. Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s Video, ISO/IEC 11172-2 (MPEG-1 Video), 1993.
2. T. Koga, K. Iinuma, A. Hirano, and T. Ishiguro, “Motion-compensated interframe coding for video conferencing,” *National Telecommunication Conference*, 1981, pp.

G5.3.1-G5.3.5.

3. R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 4, No. 4, 1994, pp. 438-442.
4. L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 3, 1996, pp. 313-317.
5. L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 4, 1996, pp. 419-422.

Chin-Chen Chang (張真誠) was born in Taichung, Taiwan, the Republic of China, on November 12, 1954. He received his B.S. degree in Applied Mathematics in 1977 and his M.S. degree in Computer and Decision Sciences in 1979 from National Tsing Hua University, Hsinchu, Taiwan. He received his Ph.D. in Computer Engineering in 1982 from National Chiao Tung University, Hsinchu, Taiwan. From 1983 to 1989, he was among the faculty of the Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan. Since August 1989, he has worked as a professor of the Institute of Computer Science and Information Engineering at National Chung Cheng University, Chiayi, Taiwan. Dr. Chang is a Fellow of IEEE and a member of the Chinese Language Computer Society, the Chinese Institute of Engineers of the Republic of China, and the Phi Tau Phi Society of the Republic of China. His research interests include computer cryptography, data engineering, and image compression.

Lin-Li Chen (陳林立) was born in Taiwan on September 30, 1972. He received the B.S. degree in Business Mathematics from Soochow University and the M.S. degree in Computer Science and Information Engineering from National Chung Cheng University in 1995 and 1997, respectively. His research interests include image processing, image compression and video compression.

Tung-Shou Chen (陳同孝) was born in Taichung, Taiwan, Republic of China, on October 14, 1964. He received the B.S. and Ph.D. degrees from National Chiao Tung University in 1986 and 1992, respectively, both in Computer Science and Information Engineering. He served at the computer center, Chinese Army Infantry School, Taiwan, from 1992 to 1994. During the academic years 1994-97, he was on the faculty of the Department of Information Management at National Chin-Yi Institute of Technology. From August 1998 to July 2000, he was a professor of the Department of Computer Science and Information Management at Providence University. Since August 2000, he has been a professor of the Department of Information Management at National Taichung Institute Technology, Taichung, Taiwan. His current research interests include data structure, steganography, and image compression.