# Predictive Line Search: An Efficient Motion Estimation Algorithm for MPEG-4 Encoding Systems on Multimedia Processors

Yu-Wen Huang, Shyh-Yih Ma, Chun-Fu Shen, and Liang-Gee Chen

***Abstract*—This paper describes an efficient motion-estimation algorithm, the predictive line search (PLS), for real-time implementations of MPEG-4 encoder on multimedia processors. The motion-vector predictor is used as the starting point in the search process because the correlation between neighboring motion vectors is strong. The line search pattern is used in the proposed algorithm to reduce the memory access as well as to exploit the special multimedia processor instructions for sum of absolute difference calculations. Experimental results show that the performance of the PLS is very close to that of the full-search (FS) algorithm. Compared with the well-known diamond search and one-dimensional FS, the PLS shows better performance and robustness, especially for high motion sequences. A prototype MPEG-4 encoding system is implemented on a 216-MHz multimedia processor with very long instruction word architecture to verify the effectiveness of the PLS. Real-time encoding of MPEG-4 Simple Profile Level 3 (CIF, 30 fps) can be achieved with only 57% of the processor load.**

***Index Terms*—Motion estimation, multimedia processor, predictive line search (PLS), real-time MPEG-4 encoder.**

## I. Introduction

MPEG-4 [1] HAS become one of the dominant standards for multimedia communication. The main issues addressed by MPEG-4 are content-based interactivity, universal accessibility, and improved compression. In order to support these complex functionalities, the video-coding system must be built on a platform that is both flexible enough for various tools and powerful enough to achieve real-time requirements. Therefore, multimedia processors [2] are the natural choice to implement such a real-time video-coding system because they combine the flexibility of programmable processors and the processing power of parallel architectures.

In almost all video compression standards, including the MPEG-4 visual part, the block-matching motion estimation is the most computationally intensive part. The simplest and most effective method of motion estimation is to exhaustively search all the candidates in the search range and find a best-matching position with the lowest distortion; this is called the full search (FS) algorithm. The distortion measure is usually the sum of absolute difference (SAD) for its simplicity. If the maximum allowable displacement for a motion address is $p$ pixels, then there are $(2p+1)^2$ candidates to compare for each macroblock, and each comparison needs $N^2$ absolute-difference operations if the size of a macroblock is $N \times N$. Thus, FS motion estimation may consume as high as 80% of the total computational power in a typical video encoding system.

In order to reduce the extremely high complexity of the FS approach, many fast algorithms for block-matching motion estimation have been proposed. The three-step search [3], new three-step search [4], one-dimensional FS (ODFS) [5], four-step search [6], block-based gradient descent search [7], center-biased diamond search (DS) [8], and advanced diamond zonal search [9] are among the most famous fast algorithms.

These algorithms are designed to search as few candidates as possible without a significant drop in quality. However, the features of the MPEG-4 compression standard and the special architecture of multimedia processors are not considered in these algorithms. Therefore, the "fewest-search-point" criterion for optimization of the motion estimation may not be feasible for MPEG-4 video compression systems on multimedia processors.

The goal of this paper is to develop an efficient algorithm for block-matching motion estimation optimized for real-time MPEG-4 video coding systems on multimedia processors. The detailed algorithm is described in the next section, followed by experimental results, discussions, and a conclusion.

## II. Predictive Line Search (PLS) Algorithm

### A. Motion-Vector Prediction

Since fast motion-estimation algorithms will not search all the candidates in the search range, the distance between the starting point and the best-matching point is directly related to the total number of searched candidates and, therefore, to the complexity.

Many algorithms use the center-biased approach, which starts from the origin because it is the most probable position for the best-matching point. However, the algorithm proposed in this paper, the PLS, starts at the motion-vector predictor to exploit the characteristics of motion field in nature video and the feature of MPEG-4 motion-vector coding method.

The coding method for motion vectors in the MPEG-4 standard is predictive coding. The motion-vector predictor can be obtained from calculating the medium value of motion vectors of the three neighboring macroblocks as shown in Fig. 1. Only the error of motion-vector prediction is coded in the bitstream. The basic principle for motion-vector prediction is that the motion field of nature video is gentle, smooth, and varies slowly [4]. Therefore, the correlation between motion vectors of neighboring macroblocks is very strong.

Fig. 2 shows the distribution of motion vectors and the distribution of motion-vector residues after prediction for Foreman sequence. The search range is (−16, 15) and the macroblock

Manuscript received January 1, 2001; revised November 20, 2002. This paper was recommended by Associate Editor H.-F. Sun.

Y.-W. Huang and L.-G. Chen are with DSP/IC Design Laboratory, Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, 106 Taiwan, R.O.C. (e-mail: yuwen@video.ee.ntu.edu.tw; lgchen@video.ee.ntu.edu.tw).

S.-Y. Ma and C.-F. Shen are with Vivotek Inc., Taipei County, Taiwan, R.O.C. (e-mail: steve@vivotek.com; diego@vivotek.com).

Digital Object Identifier 10.1109/TCSVT.2002.808093

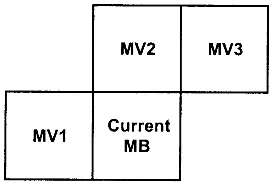

Fig. 1. Motion-vector prediction: the predictor for the current macroblock is the medium of MV1, MV2, and MV3.

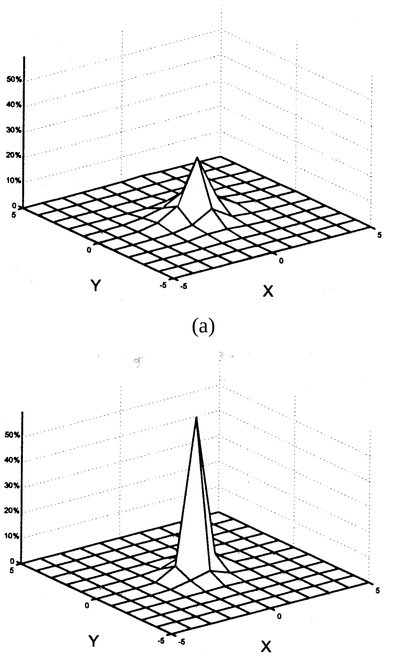

Fig. 2. Motion-vector distribution for the Foreman sequence. (a) Distribution of the motion vector. (b) Dtribution of the motion-vector residue after prediction.

size is 16 × 16, in this case. As we can see in this figure, about 24% of the motion vectors are located at the origin; this makes the center-biased approach feasible. However, after applying the MPEG-4 motion-vector prediction, more than 61% of the motion-vector residues are at the origin. Therefore, if we start our search from the position of the motion-vector predictor, it is very likely that the best-matching point can be obtained in the early stages of the search process and the complexity can be reduced significantly. Also, since the coded bit length of a motion-vector residue increases with the distance from the motion-vector predictor, there is a higher probability of getting shorter motion-vector codes by starting from the predictor. In fact, advanced diamond zonal search [9] also adopts this scheme to further improve the performance, and we will later show the difference between starting from the origin and starting from the motion-vector predictor.

## B. Considerations for Multimedia Processors

There are three main features of multimedia processors [2], [10], [11] that may impact the performance of motion estimation. They are: 1) wider data path compared with general purpose processors; 2) subword parallel architecture (SWP) to deal with multiple pixels simultaneously; and 3) special instructions for SAD calculation. Compared with general-purpose processors, the SAD can be calculated much more efficiently because, in one clock cycle, the processor is able to execute shift, subtract, absolute, and accumulation operations on many pixels in parallel.

However, this means the complexity weighting of control instructions in the motion-estimation algorithm increases in the multimedia processors because one control instruction now takes the same time as many SAD operations. For an algorithm to be efficiently executed on multimedia processors, the algorithm should be as simple as possible to reduce the control overhead.

Another issue for efficient motion estimation is data access. In most of the fast algorithms, the next search position depends on the result of current search step and can not be obtained in advance. Since the motion estimation requires massive memory access, if a fast algorithm has regular search pattern, data reuse can be applied, and the amount of memory access can be greatly reduced.

Fig. 3 shows an example of regular data access versus irregular data access. The macroblock size is 16 × 16 and the search range is (−16, 16). Fig. 3(a) shows the amount of data required for a line search pattern of 33 consecutive points. Most of the reference pixel data for the next candidate can be obtained by shifting the current reference pixel data. The total number of pixels loaded into register is $16 \times 16 + 16 \times 48 = 1024$. On the other hand, for an isolated search point, the data for the current macroblock and the reference macroblock are required as shown in Fig. 3(b). The total number of pixels loaded into registers is $16 \times 16 + 16 \times 16 = 512$. Compare the 1024 pixels for 33 candidates with the 512 pixels for only one candidate, the line search pattern is far more efficient in terms of memory access.

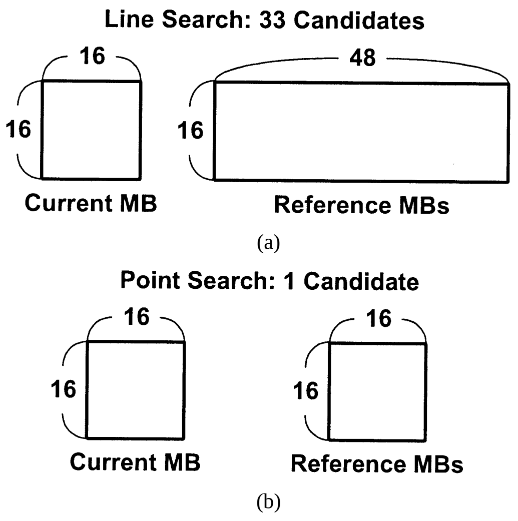

Fig. 3. Memory access comparison: (a) one-line search for PLS versus (b) one-point search for DS.

## C. The Proposed Algorithm

From the considerations of the above two subsections, we developed our fast algorithm, the PLS algorithm, with simplicity and regular search pattern in mind.

The PLS algorithm is summarized as follows:

**Step 1)** Search three consecutive lines of candidates centered at the motion-vector predictor. If the motion-vector predictor

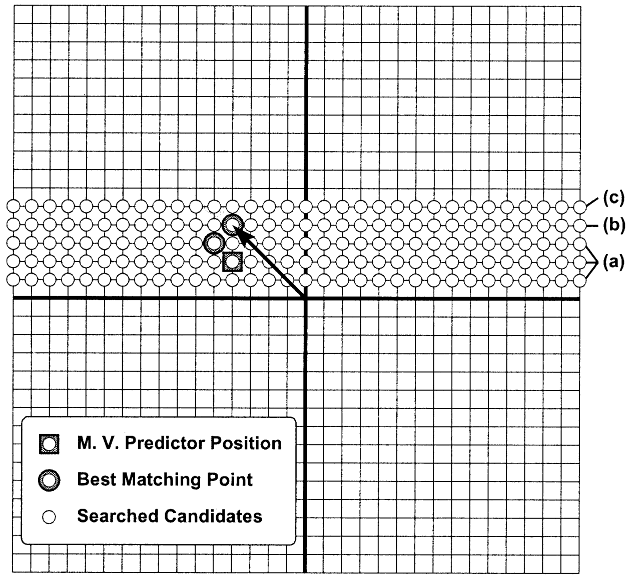

Fig. 4. PLS procedure. The search range is (−16, 15), the motion-vector predictor is (−4, −2), and the best-matching point is (−4, −4) in this example.

locates in line $p$, then all points in line $p-1$, line $p$, and line $p+1$ are tested. If the best-matching point calculated is located in line $p+1$, go to Step 2), if the best-matching point is in line $p-1$, go to Step 3), otherwise, go to Step 4).

**Step 2)** Let $p = p+1$, then test all points in line $p+1$. If the best matching point is in line $p$, go to Step 4), otherwise repeat the current step.

**Step 3)** Let $p = p-1$, then test all points in line $p-1$. If the best-matching point is in line $p$, go to Step 4), otherwise repeat the current step.

**Step 4)** Report the best-matching point as the position of the motion vector.

In short, this method starts from searching three lines around the motion-vector predictor, then searches additional lines in the direction of descending distortion, and stops when the best-matching point is not on the boundary of searched lines.

The search procedure is demonstrated by an example as shown in Fig. 4. Assume that the motion-vector predictor is (−4, −2), the true motion vector for this macroblock is (−4, −4), and the search range is (−16, 15). First, the $y$ value of the motion-vector predictor is −2, so all candidates in line −1, line −2, and line −3 are searched (a). The best-matching point in this step is at (−5, −3), which is on boundary of searched lines so an additional line is searched (b). The best-matching point after search line −4 is at (−4, −4), therefore, line −5 is also searched (c). Finally, since no candidates in line −5 has lower distortion than position (−4, −4), the procedure stops and the motion vector of (−4, −4) is found.

## III. Experimental Results

### A. Simulation Results

In order to evaluate the performance of the PLS, we apply it to several standard MPEG-4 test sequences. We use two criteria for measuring the performance of motion-estimation algorithms: the mean square error (MSE) and the motion-vector error rate. The MSE compares the motion-compensated image frame with the original image frame and calculates the MSE. The lower the MSE, the smaller the energy of the prediction error, and therefore the more effective the motion-estimation algorithm is.

TABLE I
MSE Performance Comparison

| Sequences | PLS | DS | PDS | ODFS | FS |
|---|---|---|---|---|---|
| Children | 52.8 | 54.8 | 52.6 | 70.7 | 49.2 |
| Coastguard | 48.5 | 51.5 | 48.7 | 52.6 | 48.4 |
| Container | 9.8 | 9.9 | 11.1 | 9.7 | 9.5 |
| Foreman | 43.8 | 56.6 | 44.6 | 54.1 | 39.7 |
| Hall Monitor | 22.5 | 22.6 | 22.6 | 22.4 | 22.2 |
| News | 2.7 | 2.8 | 2.8 | 2.8 | 2.7 |
| Silent Voice | 20.7 | 21.9 | 21.4 | 21.7 | 18.4 |
| Stefan | 287.2 | 507.7 | 342.1 | 290.7 | 268.1 |
| Average | 61.0 | 91.0 | 68.2 | 65.6 | 57.3 |

The motion-vector error rate of the fast algorithm is the percentage of motion vectors that are different from those obtained by the full-search algorithm. Since the FS algorithm generates the optimal results, the error rate shows how close the fast algorithm approaches the optimal solution. Therefore, an efficient and robust motion-estimation fast algorithm should have lower MSE and lower motion-vector error rate for all test sequences.

The results of center-biased DS [8] are also shown in the figures and tables for comparison. It is used for comparison not only because it has superior balance between simplicity and performance, but also because the MPEG-4 reference software [12] has adopted it as an alternative to FS algorithm. Predictive diamond search (PDS), which starts from the motion-vector predictor instead of the origin, is also tested to verify the effectiveness of motion-vector predictors and to fairly compare with the PLS. One-dimensional full search (ODFS), which is also efficient in memory access, is simulated as well.

Table I shows the MSE performance for PLS, DS, PDS, ODFS, and FS on eight standard MPEG-4 test sequences. The search range is (−16, 15) in all cases. For sequences where only small motions are involved, such as News, the MSE performance of the five algorithms are very close. FS always has the smallest MSE values, while PLS is better than DS in all cases. On the other hand, for sequences with large motions, such as Foreman and Stefan, PLS outperforms DS significantly, with slightly higher MSE values than the results of FS. This means that the PLS is very robust, even when very large motion is involved. If we compare the DS and the PDS, the effectiveness of choosing motion-vector predictors as starting points can be clearly seen. The prediction error of PDS is much smaller than that of DS for sequences with large motion, such as Foreman and Stefan. However, our PLS still significantly outperforms PDS for Stefan. As for the ODFS, it is better than PDS but worse than PLS, on average.

Figs. 5 and 6 show the MSE measure versus the frame number for Foreman and Stefan sequences. As we can see in these figures, the MSE values of results from the PLS stay very close to those from the FS all the time with only small deviations when

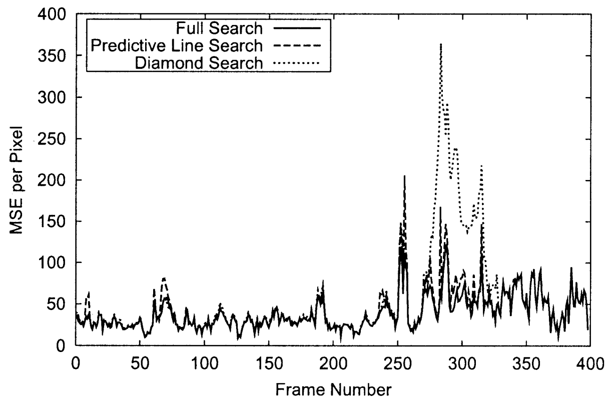

Fig. 5. MSE comparison between the PLS, the DS, and the FS algorithms. The input sequence is the Foreman sequence in CIF format

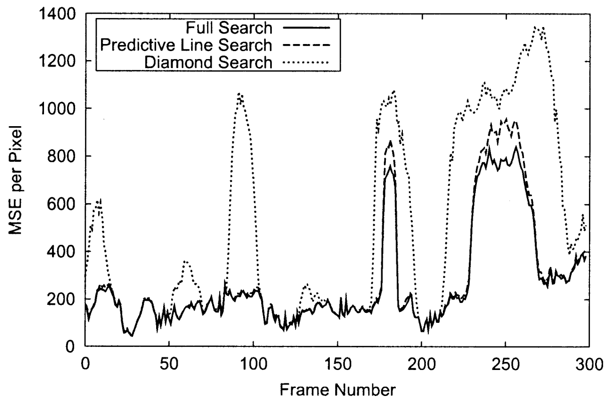

Fig. 6. MSE comparison between the PLS, the DS, and the FS algorithms. The input sequence is the Stefan sequence in CIF format.

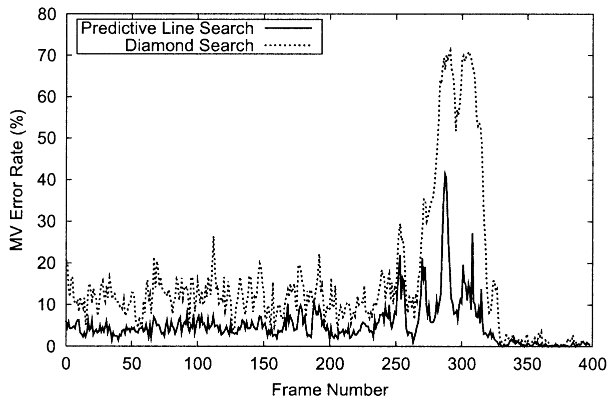

Fig. 7. Motion-vector error rate comparison between the PLS and the DS algorithms. The input sequence is the Foreman sequence in CIF format.

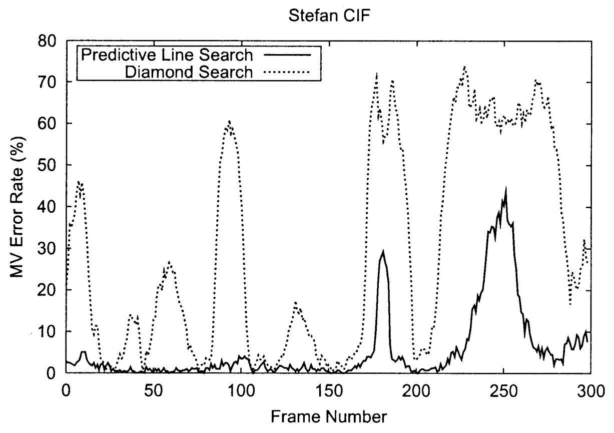

Fig. 8. Motion-vector error rate comparison between the PLS and the DS algorithms. The input sequence is the Stefan sequence in CIF format.

TABLE II
MOTION-VECTOR ERROR RATE

| Sequences | PLS | DS | PDS | ODFS |
|---|---|---|---|---|
| Children | 4.28% | 5.66% | 5.62% | 8.08% |
| Coastguard | 0.09% | 1.14% | 0.35% | 2.12% |
| Container | 0.70% | 3.90% | 4.29% | 1.54% |
| Foreman | 5.46% | 15.38% | 9.15% | 25.92% |
| Hall Monitor | 8.06% | 7.78% | 8.04% | 5.31% |
| News | 1.52% | 1.76% | 1.74% | 1.89% |
| Silent Voice | 2.35% | 3.50% | 3.42% | 5.14% |
| Stefan | 5.67% | 29.21% | 21.38% | 10.16% |
| Average | 3.52% | 8.54% | 6.75% | 7.52% |

the motion is very large. On the other hand, the MSE values of results from the DS rise significantly when sequences have large motions. Note that the curves of PDS and ODFS are omitted for clarity. In fact, for most of the frames, the two omitted curves lie between the curve of PLS and that of DS, and the curve of PDS is slightly higher than that of ODFS.

Table II shows the comparison of motion-vector error rates for various algorithms. From the table, we can see that the results of PLS is the best, especially in fast-moving sequences such as Foreman and Stefan.

Figs. 7 and 8 show the motion-vector error rates versus the frame number for the Foreman and Stefan sequences. Both sequences have large motion in the scene and, therefore, are used to test the robustness of motion-estimation fast algorithms. From the figures, we can see that the DS is not very reliable when the scene is moving fast, while the results of PLS stay very close those of FS. Superior robustness of the PLS is shown in these figures compared with the DS. Note that again the curves of PDS and ODFS are omitted for clarity. In fact, for most of the frames in Foreman, the ranks of MV error rate for these fast algorithms, from the best to the worst, are PLS, PDS, DS, and ODFS. For most of the frames in Stefan, the ranks are PLS, ODFS, PDS, and DS.

## B. Discussions

The two main features of the PLS are the predictive start point and the line search pattern. The effectiveness of these two methods are analyzed in this subsection.

Table III shows the comparison of the PLS with the center-biased line search (CBLS). The center-biased line search algorithm is the same as the PLS except that the starting point is always at the origin. Therefore, this comparison is used to show the enhancement of a predictive start point. As can be seen in the table, the MSE performance for the predictive approach is better than the center-biased approach. The motion-vector error rates and the search lines of PLS are lower than those of center-bi-

TABLE III
COMPARISON BETWEEN THE PLS AND THE CBLS

| | MSE | | MV Error Rate | | Searched Lines | |
|---|---|---|---|---|---|---|
| Sequences | PLS | CBLS | PLS | CBLS | PLS | CBLS |
| Children | 52.8 | 57.9 | 4.28% | 4.47% | 3.22 | 3.27 |
| Coastguard | 48.5 | 52.2 | 0.09% | 1.16% | 3.02 | 3.12 |
| Container | 9.8 | 9.7 | 0.70% | 0.67% | 3.03 | 3.02 |
| Foreman | 43.8 | 48.2 | 5.46% | 9.00% | 3.42 | 4.10 |
| Hall Monitor | 22.5 | 22.5 | 8.06% | 7.88% | 3.25 | 3.22 |
| News | 2.7 | 2.7 | 1.52% | 1.51% | 3.04 | 3.04 |
| Silent Voice | 20.7 | 21.2 | 2.35% | 2.53% | 3.24 | 3.28 |
| Stefan | 287.2 | 288.8 | 5.67% | 5.93% | 3.28 | 3.37 |
| Average | 60.4 | 62.9 | 3.52% | 4.14% | 3.19 | 3.30 |

ased line search. The use of predictive starting point is justified because it brings better performance and lower complexity.

Table IV shows the complexity comparison between the PLS and the other algorithms. The average number of searched lines by the PLS is 3.19. Compared with the FS algorithm, which searches all 32 lines in the search range, the speedup is about ten times faster.

Although the total number of candidates searched ($3.19 \times 32 = 102.1$) by the PLS is more than those by the DS (15.69) and those by the PDS (14.01), the memory access for PLS is only 40% of the memory access needed by the DS and 45% of the memory access needed by the PDS. The PLS has higher memory access efficiency than the DS, or than any other fast algorithm we are aware of.

The ODFS algorithm first searches a horizontal line, followed by a vertical line, and then a horizontal half line, and finally a vertical half line. Although the number of lines searched by ODFS is $3(1 + 1 + 0.5 + 0.5 = 3)$, which is lower than that of PLS (3.19), PLS is still more efficient in memory access. This is because the data reuse of one single line is more efficient than that of two separated half lines.

The total number of SAD operations that needs to be calculated is proportional to the total number of searched candidates, so the computational complexity of the PLS is about 6.51 times higher than that of the DS and 7.28 times higher than that of the PDS. However, since the PLS has lower memory access and smaller control overhead, the overall complexity comparison is platform dependent. On a platform that can calculate the SAD operations efficiently, the speed of the PLS can approach the speed of the DS and the PDS.

As for the ODFS, its required number of SAD operations is slightly lower than that of PLS, but its memory access is slightly higher than that of PLS. The complexities of these two algorithms are about the same. However, note that PLS has better performance in the quality of motion-compensated frames.

In Table IV, we assumed the cache is not used for multimedia processor. In fact, the item of memory access should be replaced by "cache and memory access" because multimedia processors are equipped with cache to facilitate higher speed of data transfer. However, even if the cache is considered, data transfer still leads to the processing bottleneck due to the high efficiency of SAD calculation in media processors. Furthermore, the size

TABLE IV
COMPLEXITY COMPARISON. (a) SEARCHED CANDIDATES PER MACROBLOCK. (b) MEMORY ACCESS PER MACROBLOCK

| | Searched Candidates per Macroblock | | | |
|---|---|---|---|---|
| Sequences | PLS (Lines) | DS (Points) | PDS (Points) | ODFS (Points) |
| Children | 3.22 | 14.10 | 13.93 | 96.00 |
| Coastguard | 3.02 | 17.63 | 13.45 | 96.00 |
| Container | 3.03 | 13.25 | 13.45 | 96.00 |
| Foreman | 3.42 | 19.13 | 14.69 | 96.00 |
| Hall Monitor | 3.25 | 13.70 | 13.84 | 96.00 |
| News | 3.04 | 13.07 | 13.08 | 96.00 |
| Silent Voice | 3.24 | 14.35 | 14.06 | 96.00 |
| Stefan | 3.28 | 20.28 | 15.56 | 96.00 |
| Average | 3.19 | 15.69 | 14.01 | 96.00 |

(a)

| | Memory Access per Macroblock | | | |
|---|---|---|---|---|
| | PLS | DS | PDS | ODFS |
| Sequences | | (Bytes) | | |
| Children | 3250.8 | 7217.1 | 7132.4 | 3520.0 |
| Coastguard | 3040.5 | 9025.0 | 6886.8 | 3520.0 |
| Container | 3052.0 | 6785.5 | 6886.8 | 3520.0 |
| Foreman | 3449.5 | 9794.6 | 7519.4 | 3520.0 |
| Hall Monitor | 3271.6 | 7015.8 | 7084.7 | 3520.0 |
| News | 3059.9 | 6694.2 | 6696.6 | 3520.0 |
| Silent Voice | 3264.7 | 7345.6 | 7119.2 | 3520.0 |
| Stefan | 3304.4 | 10384.9 | 7966.5 | 3520.0 |
| Average | 3211.7 | 8032.8 | 7161.6 | 3520.0 |

(b)

of the cache is limited. If the operands are not hit by the cache, the access time of memory is an order higher than that of the cache, which means that the reduction of memory access is still very important.

Due to the gravity, it is found that there is less significant motion in the vertical direction, so we rotated the standard sequences by 90° to test more cases. The results of MSE performance, MV error rate, and complexity are shown in Tables V–VII, respectively. Although the MSE performance and MV error rate of PLS for rotated sequences are not as good as those for original sequences, PLS is still significantly better than other fast algorithms. The complexity of PLS for rotated sequences rises a little (3.1%), while the other fast algorithms remain almost the same.

## C. System Performance

We have implemented an MPEG-4 encoder on a multimedia processor, the Equator MAP-CA, which has a very long instruction word (VLIW) core running at a clock frequency of 216 MHz. This processor can process the data of 32 pixels in parallel and has special instructions that can execute shift, subtract, absolute, and accumulation in a single clock cycle. When running a real-time encoder for MPEG-4 Simple Profile Level 3, which deals with CIF ($352 \times 288$) format at 30 frames per second, only 57% of the processing power of the multimedia

TABLE V
MSE PERFORMANCE COMPARISON FOR SEQUENCES ROTATED BY 90°

| Sequences | PLS | DS | PDS | ODFS | FS |
|---|---|---|---|---|---|
| Children | 51.5 | 54.8 | 56.3 | 55.8 | 49.2 |
| Coastguard | 48.6 | 51.5 | 48.7 | 52.3 | 48.4 |
| Container | 9.8 | 9.9 | 10.6 | 9.7 | 9.5 |
| Foreman | 43.4 | 56.6 | 46.4 | 58.7 | 39.7 |
| Hall Monitor | 22.9 | 22.6 | 22.6 | 22.8 | 22.2 |
| News | 2.7 | 2.8 | 2.8 | 2.8 | 2.7 |
| Silent Voice | 20.3 | 21.9 | 21.7 | 21.4 | 18.4 |
| Stefan | 340.2 | 507.7 | 352.5 | 470.7 | 268.1 |
| Average | 67.4 | 91.0 | 70.2 | 86.8 | 57.3 |

TABLE VI
MOTION-VECTOR ERROR RATE FOR SEQUENCES ROTATED BY 90°

| Sequences | PLS | DS | PDS | ODFS |
|---|---|---|---|---|
| Children | 4.09% | 5.65% | 6.24% | 6.63% |
| Coastguard | 0.41% | 1.13% | 0.38% | 1.42% |
| Container | 4.28% | 3.90% | 4.08% | 1.77% |
| Foreman | 7.70% | 15.40% | 9.83% | 26.60% |
| Hall Monitor | 4.16% | 7.74% | 7.96% | 4.89% |
| News | 1.06% | 1.76% | 1.76% | 1.82% |
| Silent Voice | 2.96% | 3.50% | 3.73% | 5.03% |
| Stefan | 26.44% | 29.21% | 20.05% | 24.45% |
| Average | 6.39% | 8.54% | 6.75% | 9.08% |

TABLE VII
COMPLEXITY COMPARISON FOR SEQUENCES ROTATED BY 90°. (a) SEARCHED CANDIDATES PER MACROBLOCK. (b) MEMORY ACCESS PER MACROBLOCK

| Searched Candidates per Macroblock | | | | |
|---|---|---|---|---|
| Sequences | PLS (Lines) | DS (Points) | PDS (Points) | ODFS (Points) |
| Children | 3.22 | 14.10 | 14.05 | 96.00 |
| Coastguard | 3.27 | 17.63 | 13.49 | 96.00 |
| Container | 3.07 | 13.25 | 13.31 | 96.00 |
| Foreman | 3.53 | 19.13 | 14.97 | 96.00 |
| Hall Monitor | 3.10 | 13.70 | 13.88 | 96.00 |
| News | 3.02 | 13.07 | 13.09 | 96.00 |
| Silent Voice | 3.25 | 14.35 | 14.14 | 96.00 |
| Stefan | 3.88 | 20.28 | 15.64 | 96.00 |
| Average | 3.29 | 15.69 | 14.07 | 96.00 |

(a)

| Memory Access per Macroblock | | | | |
|---|---|---|---|---|
| Sequences | PLS | DS (Bytes) | PDS | ODFS |
| Children | 3247.3 | 7217.2 | 7196.0 | 3520.0 |
| Coastguard | 3299.9 | 9025.0 | 6906.2 | 3520.0 |
| Container | 3098.9 | 6785.1 | 6813.8 | 3520.0 |
| Foreman | 3555.8 | 9794.2 | 7665.5 | 3520.0 |
| Hall Monitor | 3124.2 | 7016.0 | 7108.7 | 3520.0 |
| News | 3048.7 | 6694.1 | 6702.8 | 3520.0 |
| Silent Voice | 3279.0 | 7345.4 | 7238.0 | 3520.0 |
| Stefan | 3908.9 | 10384.5 | 8007.2 | 3520.0 |
| Average | 3320.3 | 8032.7 | 7204.8 | 3520.0 |

(b)

processor is consumed. The PLS motion estimation is responsible for 58% of the total computation load. Table VIII shows the run-time profiles for Foreman encoded at a target bit rate of 384 Kbits/s. On average, only 18.88 ms is required to encode one single frame.

Since the PLS is about ten times faster than the FS algorithm, it is not possible to run the FS algorithm in real time, even in such a powerful multimedia processor. The proposed PLS is a very good alternative.

Fig. 9 shows the peak signal-to-noise ratio (PSNR) of the Foreman sequence encoded at a target bit rate of 384 Kbits/s. As shown in the figure, the PSNR results of the PLS are very close to the results of FS throughout the whole sequence. On the other hand, the results of the DS deviate from the FS results when large motions are involved. The PLS can achieve the performance of the FS algorithm, even when large motions are involved in the scene.

Fig. 10 shows the rate-distortion curves of the Foreman sequence encoded at a target bit rate of 384 Kbits/s for various motion-estimation algorithms. As shown in the figure, the rate distortion curve of the PLS is very close to that of FS. On the other hand, the curve of the DS drops significantly from the FS results.

TABLE VIII
RUN-TIME PROFILES FOR FOREMAN

| Items | Cycles | Percentage |
|---|---|---|
| ME | 943,199,500 | 57.81% |
| MC | 17,912,126 | 1.10% |
| DCT + Quant + FCBP | 246,192,686 | 15.09% |
| IDCT + Dequant | 73,410,142 | 4.50% |
| Convert Input | 38,015,102 | 2.33% |
| Scan | 18,213,894 | 1.12% |
| MV Encode | 17,551,516 | 1.08% |
| Sync Cycles | 89,080,700 | 5.46% |
| Load Reference | 44,793,638 | 2.75% |
| Padding | 40,000,476 | 2.45% |
| Input Frame Cycles | 44,958,404 | 2.76% |
| Output Frame Cycles | 43,307,606 | 2.65% |
| Rate Control | 1,145,082 | 0.07% |
| Others | 13,734,698 | 0.83% |
| Total | 1,631,515,570 | 100.00% |

## IV. CONCLUSION

An efficient motion-estimation algorithm, the PLS, is described in this paper. The main features of PLS are the predictive starting point and the line search pattern. This search algorithm starts at the position of the motion-vector predictor because strong correlation exists between neighboring motion vectors. The line search pattern in PLS exploits the data reuse concept, so the memory access is very efficient compared with any other algorithm. From the experimental results, the PSNR performance of the PLS is very close to that of the FS approach, and the speed of the PLS is also ten times faster. It is

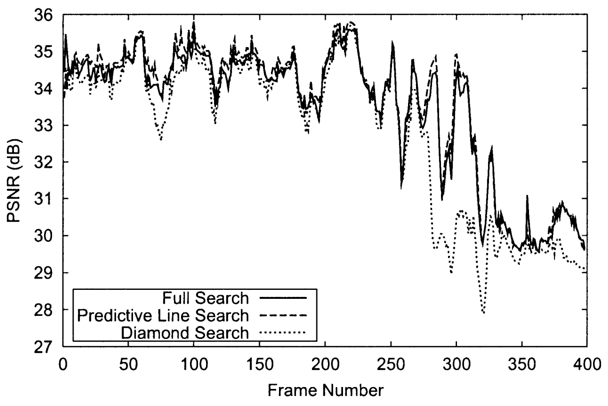

Fig. 9. PSNR comparison for the MPEG-4 encoder using three different motion-estimation algorithms: the FS, the PLS, and the DS algorithms. The input sequence is the Foreman sequence in CIF format and the target bit rate is 384 Kbits/s.

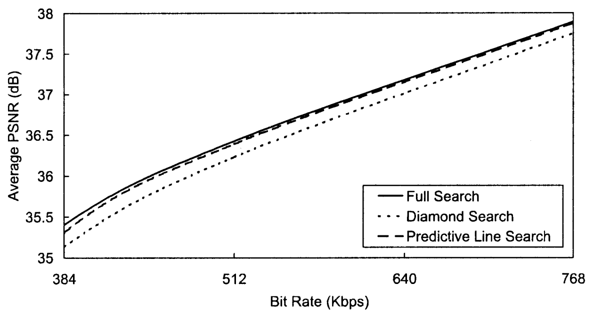

Fig. 10. Rate distortion curves for different motion-estimation algorithms: the FS, the PLS, and the DS algorithms. The input sequence is the Foreman sequence in CIF format and the target bit rate is 384 Kbits/s.

also shown that the PLS is more robust than the DS, which is a very good fast-algorithm adopted by the MPEG-4 reference software. A real-time encoder for MPEG-4 Simple Profile Level 3 is implemented on a multimedia processor with the PLS as the motion-estimation algorithm. The encoding system consumes 57% of the processing power of a 216-MHz VLIW processor core while the PLS is responsible for 58% of the computation load.

## REFERENCES

[1] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 19–31, Feb. 1997.

[2] I. Kuroda and T. Nishitani, "Multimedia processors," *Proc. IEEE*, vol. 86, pp. 1203–1221, June 1998.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proc. Nat. Telecommunication Conf.*, 1981, pp. C9.6.1–C9.6.5.

[4] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438–442, Aug. 1994.

[5] M.-J. Chen, L.-G. Chen, and T.-D. Chiueh, "One-dimensional full search motion estimation algorithm for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 504–509, Oct. 1994.

[6] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 313–317, June 1996.

[7] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 419–422, Aug. 1996.

[8] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IIEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 369–377, Aug. 1998.

[9] A. M. Tourapis, O. C. A. ad Ming, L. Liou, G. Shen, and I. Ahmad, "Optimizing the MPEG-4 encoder—Advanced diamond zonal search," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2000, pp. III-674–III-677.

[10] M. Budagavi, W. Rabiner Heinzelman, J. Webb, and R. Talluri, "Wireless MPEG-4 video communication on DSP chips," *IEEE Signal Processing Mag.*, pp. 36–53, Jan. 2000.

[11] S. Rathnam and G. Slavenburg, "An architectural overview of the programmable multimedia processor, tm-1," in *Proc. IEEE COMPCON*, 1996, pp. 319–325.

[12] T. Chiang, H.-J. Lee, and H. Sun, "An overview of the encoding tools in the MPEG-4 reference software," in *Proc. IEEE Int. Symp. Circuits and Systems*, 2000, pp. I-295–I-298.