AN EFFICIENT MOTION ESTIMATION ALGORITHM FOR REAL-TIME MPEG-4 VIDEO ENCODING ON MULTIMEDIA PROCESSORS

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

National Taiwan University, Taipei, Taiwan

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 position 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 SAD calculations. Experimental results show that the performance of the predictive line search is very close to the full search algorithm with a speed up of 10. Compared with the well-known diamond search, the predictive line search shows better performance and robustness especially for high motion sequences. A prototype MPEG-4 encoding system is implemented on a 216MHz multimedia processor with VLIW (very long instruction word) architecture to verify the effectiveness of the predictive line search. Real-time encoding of MPEG-4 Simple Profile Level 3 (CIF, 30fps) can be achieved with only 63% of the processor load.

1. 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 is powerful enough to achieve the real-time requirements. Therefore, multimedia processors [2] are the nature 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.

Almost in all video compression standards, including the MPEG-4 visual part, the block-matching motion estimation is the dominant part of the computation load. 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 algorithm. The distortion measure is usually the sum of absolute difference (SAD) for its simplicity. If the maximum allowable displacement for 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 macroblcok is $N \times N$. Thus, the full search 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 full search approach. Many fast algorithms for block-matching motion estimation [3] [4] [5] have been proposed. These algorithms are designed to search as few candidates as possible without significant drop in quality. However, the feature 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 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.

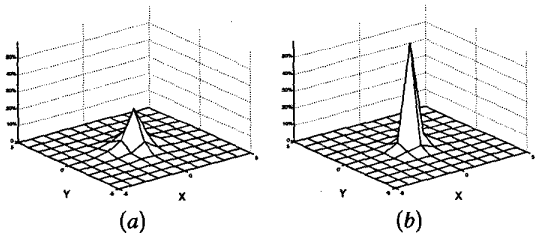
2. PREDICTIVE LINE SEARCH ALGORITHM

2.1. 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 predictive line search (PLS), starts at the motion vector predictor to exploit the characteristics of motion field in nature video and the feature of MPEG-4

Chun-Fu Shen is with Vivotek Inc., Taipei County, Taiwan.



(a) (b)

Figure 1: Motion vector distribution for Foreman sequence: (a) the distribution of motion vector (b) the distribution of motion vector residue after prediction.

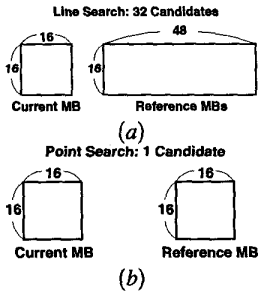


Figure 2: Memory access comparison: (a) one line search for the predictive line search (PLS) versus (b) one point search for the diamond search (DS).

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. 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 [6], therefore, the correlation between motion vectors of neighboring macroblocks is very strong.

Fig. 1 shows the distribution of motion vectors and the distribution of motion vector residues after prediction for Foreman sequence. These distributions are obtained by applying the full search algorithm. The search range is $(-16, 16)$ and the macroblock size is 16×16 in this case. As we can see in this figure, about 24% of the motion vectors locate 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 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, starting from the predictor has a higher probability of getting shorter motion vector codes.

2.2. Considerations for Multimedia Processors

There are three main features of multimedia processors [2] that may impact the performance of motion estimation, they are wider data path compared with general purpose processors, sub-word parallel architecture (SWP) to deal with multiple pixels simultaneously, and 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. 2 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. 2(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. 2(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.

2.3. The Proposed Algorithm

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

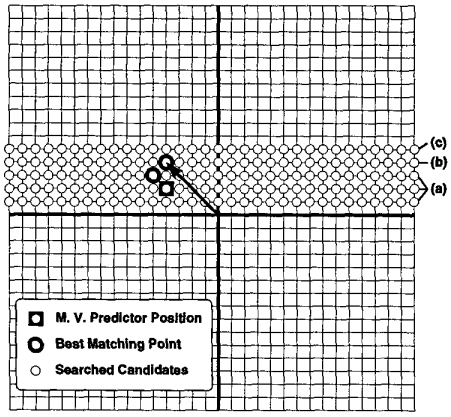


Figure 3: The predictive line search (PLS) procedure. The search range is (-16,16), the motion vector predictor is (-4, -2), and the best-matching point is (-4,-4) in this example.

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 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 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. 3. Assume that the motion vector predictor is $(-4,-2)$, the true motion vector for this macroblock is $(-4,-4)$, and the search range is $(-16,16)$. First, the y-value of 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.

Table 1: MSE Performance Comparison

Sequences	PLS	DS	FS
Children	52.8	54.8	49.2
Coastguard	48.5	51.5	48.4
Container	9.8	9.9	9.5
Foreman	43.0	56.5	39.3
Hall Monitor	22.5	22.6	22.2
News	2.7	2.8	2.7
Silent Voice	20.1	21.0	17.7
Stefan	284.2	507.5	265.1
Average	60.4	90.8	56.8

3. EXPERIMENTAL RESULTS

3.1. Simulation Results

In order to evaluate the performance of the predictive line search (PLS), we apply it on several standard MPEG-4 test sequences. We use the mean square error (MSE) for measuring the performance of motion estimation The MSE compares the motion compensated image frame with the original image frame and calculates the mean square error. The lower the MSE, the fewer bits will be used to represent the predictive error, and therefore the more effective the motion estimation algorithm is.

The results of center-biased diamond search (DS) [4] 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 [7] has adopted it as an alternative to full search algorithm.

Table 1 shows the MSE performance for predictive line search (PLS), the diamond search (DS), and the full search (FS) algorithms on eight standard MPEG-4 test sequences. The search range is (-16,16) in all cases. For sequences where only small motions are involved, such as News and Container, the MSE performance of three algorithms are very close. Full search always has the smallest MSE values, while the predictive line search is better than the diamond search in all cases. On the other hand, for sequences with large motions, such as Foreman and Stefan, predictive line search outperform diamond search significantly with slightly higher MSE values than the result of full search. This means that the predictive line search is very robust even

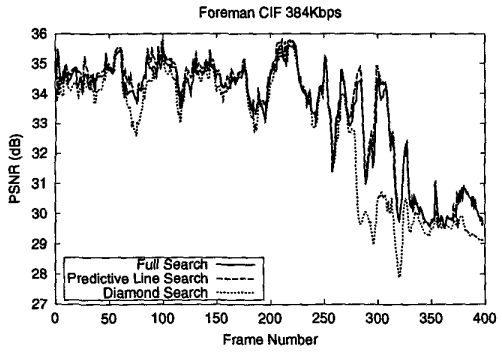


Figure 4: PSNR comparison for the MPEG-4 encoder using three different motion estimation algorithms.

when very large motion is involved.

3.2. System Performance

We have implemented an MPEG-4 encoder on a multimedia processor, the Equator MAP-CA, which has a VLIW (very long instruction word) core running at a clock frequency of 216MHz. 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 (352x288) format at 30 frames per second, only 63% of the processing power of the multimedia processor is consumed. The predictive line search motion estimation is responsible for 54% of the total computation load. Since the speed up for the predictive line search is about 10 compared with the full search algorithm, it is not possible to run the full search algorithm in real time even in such a powerful multimedia processor. The proposed predictive line search is a very good alternative.

Fig. 4 shows the PSNR (peak signal-to-noise ratio) of the Foreman sequence encoded at a target bit rate of 384Kbps. As shown in the figure, the PSNR results of the predictive line search are very close to the results of full search through out the whole sequence. On the other hand, the results of the diamond search deviate from the full search results when large motions are involved. The predictive line search can achieve the performance of the full search algorithm even when large motions are involved in the scene.

4. CONCLUSION

An efficient motion estimation algorithm, the predictive line search (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 motion vector predictor because there exists strong correlation 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 performance of the predictive line search is very close to that of the full search approach with a speed up of 10. It is also shown that the predictive line search is more robust than the diamond search, 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 predictive line search as the motion estimation algorithm. The encoding system consumes 63% of the processing power of a 216MHz VLIW processor core while the predictive line search is responsible for 54% of the computation load.

5. REFERENCES

[1] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 19–31, Feb. 1997.

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

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

[4] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 369–377, Aug. 1998.

[5] 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 *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. III–674–III–677, 2000.

[6] 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, pp. 438–442, Aug. 1994.

[7] T. Chiang, H.-J. Lee, and H. Sun, "An overview of the encoding tools in the MPEG-4 reference software," in *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. I–295–I–298, 2000.