AN ADAPTIVE BLOCK-MATCHING ALGORITHM FOR MOTION ESTIMATION

Vasily G. Moshnyaga

Department of Electronics and Computer Science, Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, JAPAN

ABSTRACT

A new adaptive algorithm for the block matching motion estimation is presented. The algorithm works in the full-search fashion but unlike the FSBMA it adjusts the number of computations dynamically to picture variation. Due to incorporated mechanism of data-driven thresholding, the proposed approach performs as four times as less operations comparing to the FSBMA while maintaining the same quality of results. Its hardware implementation is simple and compact. A supportive hardware design as well as simulation results on benchmarks are outlined.

1. INTRODUCTION

1.1. Motivation

Motion estimation is a basic bandwidth compression task utilized in video-coding systems. Among various computation methods[1], the *Full Search Block Matching Algorithm* (FSBMA) is most popular. Having successive video frames divided into blocks of $(N \times N)$ pixels, the FSBMA determines a *motion vector* (v) for every *reference block* (X) of the current image by comparing it with all *candidate blocks* $(Y(0,0), Y(0,1), ...,)$ within the search area surrounding the position of the reference block in the previous frame. Let $x(i,j)$ be the luminance value of the reference block pixel, $y(i,j)$ the luminance value of the candidate block pixel, p the maximum displacement allowed in both vertical and horizontal directions. Then, the position (m,n) of a candidate block $Y(m,n)$ that results in the minimum distortion D denotes the motion vector v:

$$D(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |y(i+m, j+n) - x(i,j)|, \quad -p \le m,n \le p-1 \qquad (1)$$

$$v = \arg \min_{-p \le m,n \le p-1} D(m,n) \qquad (2)$$

The FSBMA provides optimal precision, regular data flow as well as higher parallelism, a characteristic that is advantageous for VLSI implementation. However, it is extremely time consuming because $N^2 \times (2 \times p)^2$ computations of the distortion (D) have to be performed per each reference block. If a frame has 720×480 pixels, $p = 16$, $N = 16$ (MPEG2, MP@ML complexity level), the FSBMA requires over 11 GOPS.

Due to such an enormous computational rate, existing hardware implementations of the FSBMA[2] are extremely power hungry: over a half of energy dissipated in a modern encoder is burned in the motion estimation hardware! As result, algorithms and architectures which ensure low power operation have become very important, especially for portable video application.

1.2. Related research

Over the years, a large variety of fast and computationally inexpensive block-matching algorithms have been proposed. Examples include the 2D-logarithm search[3], the three-step search[4] and its modification[5], the conjugate direction search [6], the cross-search[7], etc. These algorithms search only a small subset of available candidate blocks and consequently execute less computations. However, they lack in terms of Peak Signal to Noise Ratio (PSNR), i.e. subjective picture resolution. Therefore, recent research attempts have been put on decreasing the operational complexity of the FSBMA.

An approach to reduce the FBMA complexity is to transform the 8-bit gray-scale data into binary numbers and then use binary level distortion metric instead of multi-bit arithmetic, as proposed in [8],[9]. This approach, as well as the LSB-bit truncation scheme[10], minimizes the amount of energy dissipating switches during the distortion calculation. However, it does not affect the amount of operations involved in the FSBMA. To lower the number of operations in the FSBMA, work [11] suggest the dynamic search range adjustment to the picture content variation. The idea is to run the FSBMA using a large p over a number of frames and then shrink p, if possible, to accommodate 95% of motion vectors. The technique achieves a quadratic reduction in operations but restricts itself to the highly correlated video sequences. Moreover, it challenges the FSBMA's optimality which is not acceptable. Up to our knowledge there have been reported only one method[12] capable of reducing the FSBMA complexity without sacrificing its accuracy. Before computing the exact distortion, the method performs a conservative estimation of the distortion value. If the estimate is larger than the minimum distortion found so far, the exact distortion value is not calculated and the corresponding computations not performed. The approach can halve the computational count, but at the cost of extra large hardware overhead.

1.3. Contribution

In this paper, we propose a novel algorithm for adaptive elimination of unnecessary computations in the full-search block matching. In contrast to FSBMA, we dynamically

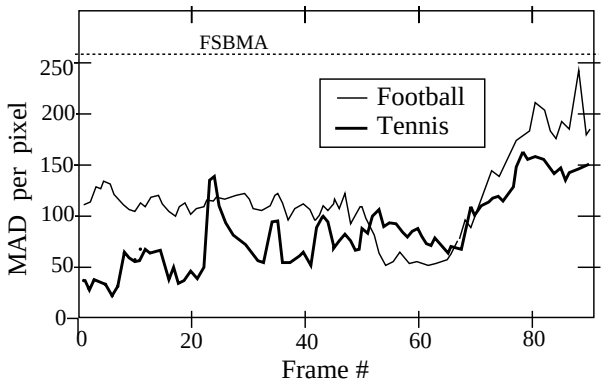

Figure 1. MAD per pixel of the motion compensated frames produced by FSBMA for the first 90 frames of the 'Tennis' and 'Football' sequences

adjust the number of computations required per block to the picture content. Comparing to the FSBMA, the proposed algorithm performs as less as 1/4 of the total number of operations while maintaining the highest level of PSNR. Unlike the related research[12], our algorithm does not require a large number of extra computations to eliminate the redundant search candidates. Also it is more compact in implementation.

2. ADAPTIVE FSBMA

2.1. Main idea

Figure 1 shows the number of displacement computations per pixel obtained for 90 frames of two video sequences 'Tennis'(10-s segment) and 'Football'(7 segment) for the MPEG2, MP@ML complexity. As it indicates, the picture content significantly varies with frames, i.e. in time. The FSBMA, however, does not take it into account, executing the fixed maximum of MAD-operations per pixel, block and frame. (see the dotted line). In order to increase the efficiency of the FSBMA, we propose to adjust dynamically the number of computations to the picture content variation. Namely, if the picture is changing slowly, i.e. the number of motion vectors is small, the total number of computations should be low. From the other hand, when the picture motion is large an extra amount of computations has to be performed in order to preserve the quality of the results.

As stated in Equations (1) and (2), the main goal of the FSBMA is search for a candidate block which has the lowest accumulated distortion (D_{min}). Since one distortion value is computed for each candidate block in the search area, the minimum value, D_{min}, must be found from the pool (S) of $(2 \times p)^2$ candidates. Clearly, the less is the size of the pool, the less computations necessary for the block matching.

Let $D^i(m,n) = \sum_{j=0}^{N-1} |y(i+m, j+n) - x(i,j)|$ be the partial distortion accumulated in row i of block $Y(m,n)$. Then Eq.(1) can be rewritten as

$$D(m,n) = \sum_{i=0}^{N-1} D^i(m,n) \qquad (3)$$

In the standard block matching we have the following assumptions: (1) the content of a block remains unchanged among the adjacent frames; (2) the pixels enter the system in a row (or column) based fashion and (3) the accumulation in Eq.(1) is done sequentially. Based on them, one can assume a candidate block $Y(m,n)$ with a high matching capability to the reference block X has a small distortion $D^i(m,n)$ at the accumulation step i, while a large distortion $D^i(u,v) > D^i(m,n)$ reflects the block $Y(u,v)$ with a picture pattern different to X. Consequently those candidates, $Y(u,v)$, whose distortion values $D^i(u,v)$ exceed a given threshold T, $(D^i(u,v) > T)$, can be eliminated from the search at the steps $i+1, i+2, ...$, with omitting all the corresponding operations.

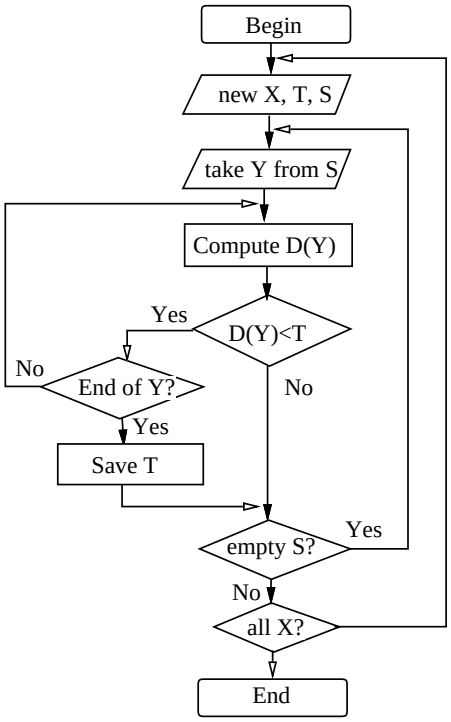

Figure 2. Flow-chart of the adaptive FSBMA

Figure 2 shows the flow chart of the proposed algorithm. Starting with a reference block X, a given threshold T and a full search pool S of the candidate blocks $Y(u,v)$, it then iteratively computes the partial distortion $D^i(Y)$ for every candidate Y, comparing it to the threshold T after each iteration (i). If $D^i(Y) > T$, the block Y is excluded from the search pool of X. The iteration cycle for X continues till all the pixels of the candidate blocks are processed. A new reference block X initiates a new iteration cycle.

Figure 3 illustrates the algorithm on a simple example of matching the 3×3 reference block in the 4×4 search area ($p = 1$). For the simplicity, assume that all 6 candidates block are processed in parallel and the dotted lines show the rows processed at each step and the corresponding partial distortion values. During the first iteration, the algorithm compares the upper row of the reference block to a corresponding upper row of all the six candidate blocks. The candidate blocks 5 and 6 have distortions larger than the threshold (T=5). So they are excluded from the search.

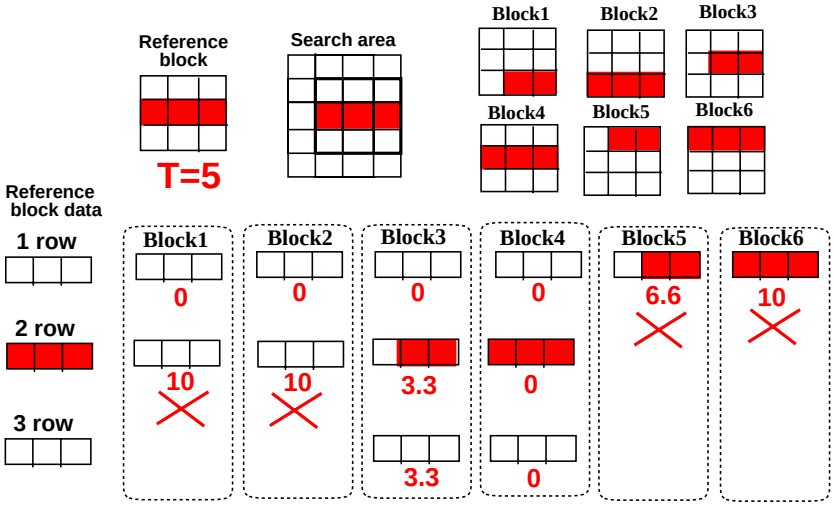

Figure 3. An illustration of the algorithm operation

Similarly, the second iteration stops computations in blocks 1 and 2 because both of them have the distortion of 10. Thus, we have only two blocks at the final iteration, and determine the best match in the block 4. The total number of operations was 12 instead of 18.

2.2. The threshold modification algorithm

The basic question is how large the threshold T should be. The choice of T involves a tradeoff between the quality of the motion estimation and the computational count. When T is large, the number of eliminated operations is small but the PSNR is high. Oppositely, when T is small, the PSNR value is decreasing, because the number of computations is small. For some video sequences and applications, where picture quality highly depends on the prediction error, a higher number of computations is preferred to achieve a lower prediction error. Hence, the T should be increased. On the other hand, for some video sequences when the picture content is more changing or the desired output PSNR is lower, the T can be reduced in order to achieve a larger energy saving. In order to cope with the picture quality requirements, we modify the threshold dynamically during the ME.

The threshold modification algorithm works as follows. Starting with $T(X) = \infty$, it computes motion vectors for all the blocks of the first frame (F_1). The values of distortions, D_{min} corresponding to each motion vector in the frame are then used as the thresholds, $T_i, (i = 1, 2, ...)$ for the blocks $X_i, (i = 1, 2, ...)$ of the next frame. If the threshold value used for the current block X_i are so small that all the candidates in S need to be eliminated from the consideration at a step i, the corresponding $T(X)$ is incremented by Δ an the process continues without stopping the computations at the current step i.

2.3. Implementation scheme

Figure 4 shows a possible implementation scheme on a linear array of PEs which broadcast current block pixels[1]. In this figure, the AD unit computes accumulated displacement, the L is latch, R is latch with multiplexor, M the minimum computation unit. Each AD unit works individually accumulating its own partial distortion value incrementally, in a step-by-step fashion. After t-steps, the values are compared to the threshold (T). Those blocks whose distortion values raised over the T are shut off, thus stopping operations. The process is repeated with selection and shut-down of the other AD-units, if any, in every t steps. Thus, only units with the accumulated values below the threshold will continue working while all the others stop. With a new reference block X, all the AD-units "wake up" and begin computing. The arithmetic cost of the method can be easily shown to involve one comparison unit per AD-unit, one reg-

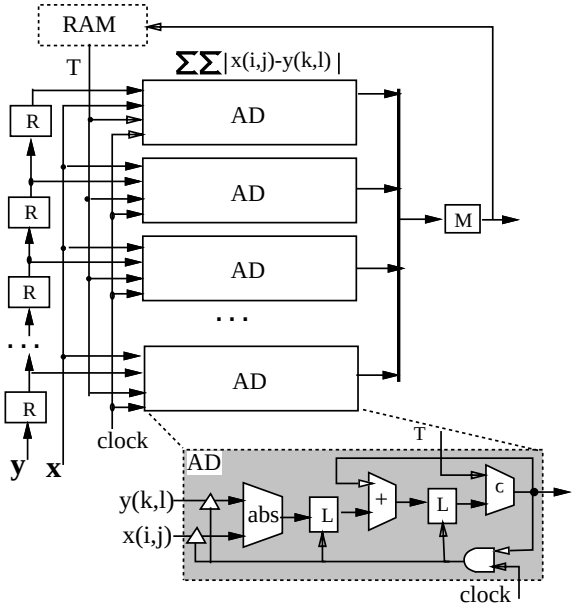

Figure 4. An implementation of the adaptive FS-BMA

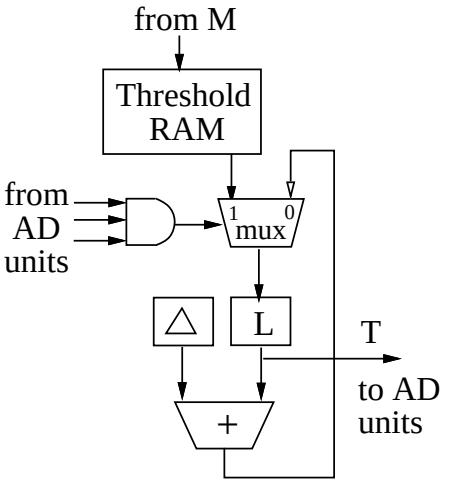

Figure 5. The threshold modification circuit

Table 1. PSNR

Video sequence	PSNR Full	PSNR Adapt.	Error (%)	Save (%)
Carousel	35.227	35.219	0.3	73
Football	40.338	38.902	0.1	76
Bicycle	39.822	36.590	0.8	60
Fl.garden	37.803	37.610	0.5	56

ister per block to store the threshold and a simple control. Figure 5 shows the control circuit structure. The threshold RAM takes the minimum displacements (from M in Fig.2) calculated for each block in the current frame and outputs them as the block thresholds for the next frame. Since one threshold is used per block, the required capacity of the threshold RAM is $M1 \times M2/N^2$, where $M1$ and $M2$ are the frame sizes, respectively. For the MP@ML level it is only 1350 words.

3. EXPERIMENTAL RESULTS

The proposed algorithm has been tested on the following video sequences: *Carousel, Football, Bicycle, Flow Garden*, picture size: 352× 240. In each video sequence, the reference frame is the first frame (001); three search frames are 002,003 and 004. Table 1 lists the Peak Signal to Noise Ratio for the conventional FSBMA and the adaptive FSBMA, the average error picture and the percentage of operations saved do to adaptive processing. Note that the error image is very low (less than 1%) while the amount of computations was reduced to 1/4 of the traditional FSBMA. Table 2 outlines the effect of N and p on the results for the Football benchmark. As we expected, the larger p the larger is saving factor.

4. CONCLUSIONS

We presented a new algorithm that effectively reduces the number of operations in full-search block-matching motion estimation without sacrificing the quality of the results. Due to adaptive adjustment of computations to the picture variation, the algorithm is able to save up to 3/4 of the total operations required by the FSBMA, while preserving high quality of the results. Future research will be dedicated to a detailed hardware design.

Table 2. Number of operations per frame as a function of N and p (Football, Picture size: 352x240)

N	p	No.operations Full	No.operations Adapt.	Save (%)	Error (%)
8	8	20,504,640	7,005,512	66	0.3
8	16	81.870,912	24,644,032	70	0.4
16	8	19,411,200	6,480,352	65	0.7
16	16	77,357,312	22,217,167	72	0.8

REFERENCES

[1] M.Sung, "Algorithms and VLSI architectures for motion estimation", *VLSI Implementations for Image Communications*, P.Pirsch (Ed.), 1993, pp.251-2281.

[2] M.Yoshimoto, et al., "ULSI realization of MPEG2 Real-time Video Encoder and Decoder - An Overview", *IEICE Trans.Electron.*, Vol.E78-C, No.12, pp.1668-1681, Dec. 1995.

[3] J.Jain, and A.Jain, "Displacement measurement and its application in internal image coding", *IEEE Trans. Commun.*, vol.29, No.12, pp.1799-1808, 1981.

[4] T.Koga, K.Linuma, A.Hirano, Y.Lijima and T.Ishiguro, "Motion-compensated interframe coding for video conferencing", *Proc.NTC'81*, pp.G5.3.1-G5.3.5, 1981.

[5] H.Jong, L.Chen, and T.Chieuh, "Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding", *IEEE Trans. Circuits Syst. Video technol.*, vol.4, no.1, pp.88-91, Jan. 1994.

[6] R.Srinivasan and K.Rao, "Predictive coding based on efficient motion estimation", *IEEE Trans. Commun.*, vol.38, No.9, pp.950-953, 1990.

[7] M.Ghanbari, "The cross-search algorithm for motion estimation", *IEEE Trans. Commun.*, vol.38, No.9, pp.950-953, 1990.

[8] M.Mizuki, U.Desai, I.Masaki, and A.Chandrakasan, "A binary block matching architecture with reduced power consumption", *IEEE ICASSP'96*, vol.6, pp.3248-3251, 1996.

[9] B.Natarajan, V.Bhaskaran, and K.Konstantinides, "Low complexity block-based motion estimation via one-bit transform", *IEEE Trans. Circuits Syst. Video technol.*, vol.7, no.4, pp.702-706, Aug.1997.

[10] Z-L.He, K-K.Chan, C-Y.Tsui, and M.L.Liou, "Low-Power Motion Estimation Design Using Adaptive Pixel Truncation ",*IEEE ISLPD'97*, 1997.

[11] S.Park and W.Burleson, "Reconfiguration for power saving in real-time motion estimation", *IEEE ICASSP'98*, vol.5, pp.3037-3040, 1998.

[12] V.L.Do and K.Y.Yun, "A low-power VLSI architecture for fuul-search block matching motion estimation", *IEEE Trans. CAS for Video Technology*, Vol.8, No.4, Aug. 1998, pp.393-398.