A SPIRAL SEARCH ALGORITHM FOR FAST ESTIMATION OF BLOCK MOTION VECTORS

Th. Zahariadis[1] *and D.* Kalivas[2]
[1] National Technical University of Athens, Heroon Polytechniou 9, 157 73 Athens, Greece
[2] Develop. Programmes Dept., Intracom S.A., P.O. Box 68, 19002 Peania, Greece
Tel. +30 1 686000 (X-4483); Fax. +30 1 6860312
e-mail: dkal@intranet.gr

ABSTRACT

The most important fast block matching algorithms are analysed and evaluated. Then a new fast search method, the "Spiral Search Algorithm" (SSA), is introduced. It is a three step algorithm which follows a spiral path searching outwards for candidate locations that satisfy the matching criterion. The efficiency of the SSA arises from: (1) the reduction of the candidate locations without leaving out zones of pixels where the mean absolute difference is not evaluated, and (2) the reduction of computations since many candidate locations are being bailed out. A comparison of fast search methods and the Full Search (FS) approach is presented for a number of video sequences. The SSA is proven to be an excellent compromise between quality and speed.

1 INTRODUCTION

Many video applications, including standard and high-definition television (HDTV), video conference and CD-ROM archiving, demand video compression. The inherent redundancy in time that characterises successive frames of a video sequence, makes possible the achievement of high compression ratios when motion compensated video coding is used. In a video coder the motion estimator is the most computationally intensive part. For this reason, Fast Block Matching Algorithms (BMA) for motion estimation are used in many practical video coding schemes. The *Logarithmic Search* (LS) [3], the *Three-Step hierarchical Search* (TSS) [4] and the *Binary Search* (BS) used in MPEG-Tool are considered to be among the best fast BMAs. They have decreased computational complexity with respect to the FS and produce decoded video sequences of high quality. In section 2, these BMAs are described and compared to the full search approach. In section 3, a new algorithm, the *Spiral Search Algorithm* (SSA), is presented. The SSA provides quality and complexity comparable to TSS and BS respectively. In section 4, the BMAs algorithms have been incorporated into the second public release of the MPEG-2 Video Encoder offered by the MPEG Software Simulation Group and their performance is being analysed. Finally, in section 5, the conclusions of the comparison of the SSA to the other fast BMAs and the FS are presented.

2 BLOCK MATCHING ALGORITHMS

A BMA removes the temporal redundancy within a frame sequence on a block by block analysis. Each frame is divided into blocks of *MxN p*ixels. For each block in the current frame, a matched displaced block from a previous and/or a future frame is found. If x and y are the maximum absolute displacement values, there will be (2x+1)(2y+1) locations to search for the best match. Various criteria can be used as measures of the match between two blocks. The Mean Absolute Difference (MAD) is favoured, because it requires no multiplication and gives similar performance to the Mean Squared Error (MSE). If blocks are referred by the co-ordinate of their upper left corner, the MAD between the block *(k,l)* of the present frame and the block *(k+x,l+y)* of the previous frame is :

$$MAD_{(k,l)}(x,y) = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left|F_n(k+i,l+j) - F_{n-1}(k+x+i,l+y+j)\right|$$

where the (k+x,l+y) are valid block co-ordinates [2].

The simplest search is the *exhaustive* or *full search* (FS), where all possible displacements are evaluated within a specific range. Thus, the FS computes the MAD at all (2x+1)(2y+1) locations of the searching area in order to find the best motion vector. For many applications the FS may be time consuming, so fast search methods have been proposed [1]-[6]. The main principle of these algorithms is the reduction of the computation complexity by decreasing the number of the searching locations. This principle is based on the assumption that the MAD is monotonically decreasing around the location of the optimal motion vector. However, the MAD surface may include local minima, where the algorithms may be trapped. This will create final motion-vectors different than optimal ones and a larger residual error in the motion prediction. In the case of variable bit rate applications this will cost increased bandwidth requirements and in the case of constant bit rate applications quality degradation in the decoded sequences. However, the significant gain in the complexity and the rather acceptable quality constitute algorithms of great interest. The requirements of an application may each time define the threshold between required picture quality and consumed encoding time.

2.1 The Three-Step hierarchical Search (TSS)

In order to reduce the heavy computational cost implied by the large number of candidate locations, the TSS algorithm searches for the best motion vector in a coarse-to-fine manner [5]. An example of the searching procedure of the TSS with a searching window of 15x15 is shown in Fig. 1.

The following steps describe the main computations :
Step 1: The MAD is evaluated on a coarse grid of 9 pixels. The pixel with the smallest MAD is chosen.
Step 2: A less coarse grid of 9 pixels is used, centred at the pixel with the best match in the previous step.
Step 3: Finally, a full search around the winner of the second step provides the final motion vector.

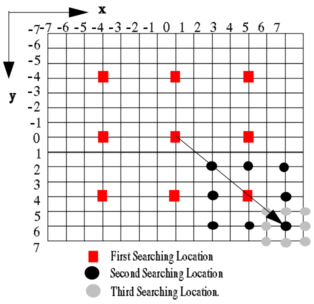

Fig. 1. The TSS algorithm.

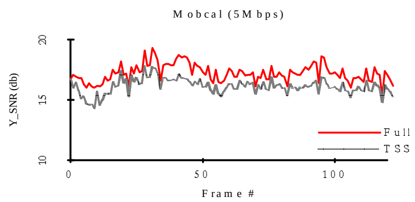

Fig. 2 Luminance SNR of the motion-compensated frames for the FS and the TSS algorithms in mobcal video series.

The TSS algorithm performs a great reduction in number of candidate locations. For a 15x15 searching window, the MAD is evaluated on 225 locations in the FS and only on 25 locations in the TSS. Experimental results show that the TSS provides a robust and close to optimal performance. The performance of the FS and the TSS for the mobcal video sequence (15x7 searching window, 5 Mbits/s) is shown in Fig.2.

2.2 The Logarithmic Search Algorithm

The *Logarithmic Search* (LS) is closely related to the TTS algorithm. In each step the MAD is evaluated on a grid of five pixels. From step to step the distance between the search pixels is reduced in a logarithmic way. Using this technique, the LS requires more steps, searching in more candidate locations and consuming more time. However, it may be more accurate than the TSS, especially when the searching window is quite large.

2.3 The Binary Search (BS)

The MPEG-Tool uses a Binary Search method for the estimation of the motion vectors (Fig. 3) [6]. The algorithm is executed in the following two steps:

Step 1: The MAD is evaluated on a grid of 9 pixels, one at the center, 4 at the boundaries and 4 at the corners of the searching window. The pixel with the smallest MAD is chosen.
Step 2: A full search is focused on the area centered at the pixel of the best match in the first step providing the final motion vector.

If the pixel with the smallest MAD is located at the center of the searching window, BS will evaluate the MAD in 33 candidate locations. This is the worst case scenario and it will occur with probability 1/9. If it is located at the boundaries, the MAD will be evaluated in 23 locations with probability 4/9. Finally, if it is located at the corners, the MAD will be evaluated in only 17 locations (best case scenario), with probability 4/9. So the average number of the MAD calculations are 21.445, less than the 25 in the TSS and the 225 in the FS. A comparison of the performance of the BS and FS algorithms is shown in Fig. 4. The main reason of the inferior performance of the BS algorithm is the existence of a zone of pixels where MAD is never evaluated. This zone is analogue to the searching window.

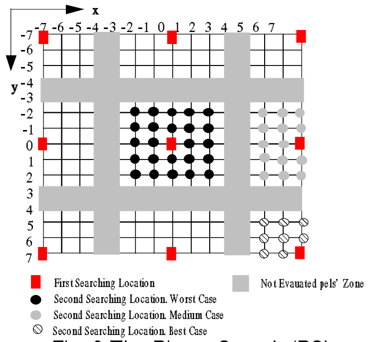

Fig. 3 The Binary Search (BS)

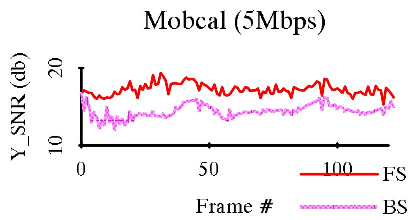

Fig. 4 Luminance SNR of the motion-compensated frames provided by FS and BS algorithms in mobcal video series (5Mbits/s, 7x3 searching window).

3 THE SPIRAL SEARCH ALGORITHM

The Spiral Search Algorithms (SSA) combines the TSS and BS principles to find the motion vectors. The appropriate routines are sped up and there is not a zone of pixels where the MAD is not evaluated. The algorithm is executed in the following steps (Fig. 5):

Step 1: The MAD is evaluated on a grid of 4 pixels, which are the ends of a cross (+). The dimensions of the cross is equal to the half of the length of the searching window.

Step 2: The MAD is evaluated on a grid of 4 pixels, which are located at the corners of the searching window. The pixel with minimal MAD in both step 1 and step 2 is selected.

Step 3: The search is focused on the area centered at the pixel of the best match in the previous steps. A less coarse grid of 9 pixels is used for the MAD evaluation.

Step 4: Finally, a full search around the winner of the third step provides the final motion vector.

The SSA has gained its name by the spiral created in the first two steps.

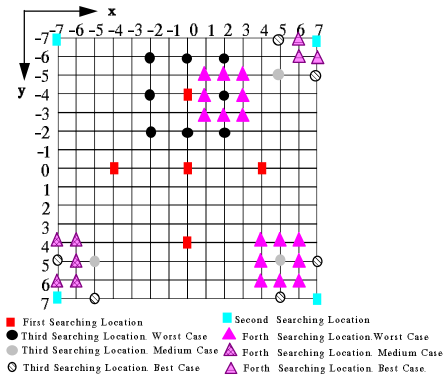

Fig. 4 The Spiral Search Algorithm(SSA).

The total number of MAD calculations in the first and second step is 9. If a pixel from the first step is selected, then 16 (8+8) MAD calculations are performed in the third and fourth steps. This is the worst case scenario with 25 candidate locations and a probability of 5/9. If a pixel from the second step is selected, then 3 MAD calculations are performed in the third step. In the fourth step, the number of MAD calculations may be 8, 5 or just 3 with probability 1/9, 2/9 or 1/9 respectively. So the average number of MAD calculations is 21.556, which is less than the 25 in the TSS and the 225 in the FS and slightly more than the 21.445 in the BS. Further computational reduction is achieved by the following technique: While searching candidate locations, SSA stores the minimum MAD that has been so far evaluated in the current searching window (Current MAD). During the evaluation of the MAD in a location, after each increment, its value is compared to the CMAD. If the MAD exceeds the CMAD, the evaluation does not need to continue for the particular location and this location is bailed out. So, many locations are aborted with just a few comparisons. This technique combined with the spiral order of MAD calculations leads to a further computational reduction. If there are objects with no or small displacement between two frames (e.g. background), the global minimum MAD or a good approximation is quickly found, so the rest of the locations are bailed out with just a few comparisons and the total computation complexity is significantly reduced. Then, the SSA follows a spiral path and greater displacements are evaluated. This is an important feature of the SSA. Finally, the SSA does not leave out zones of pixels where MAD is not evaluated and the optimum MAD is found with a satisfactory probability.

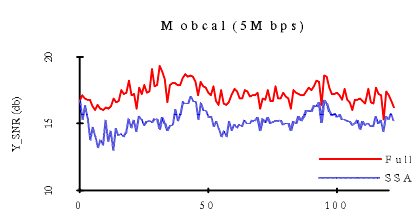

Fig. 5 Luminance SNR of the decoded frames for the FS and the SSA using the mobcal sequence (5Mbits/s, 7x3 searching window).

4 PERFORMANCE EFFICIENCY OF SPIRAL SEARCH ALGORITHM

In Figures 6-9, a comparison between the fast algorithms and the FS is presented. Generally, the FS algorithm provides the best quality (maximum SNR and minimum MSE), and TSS, SSA and BS algorithms follow in decreasing quality. Though, especially in the football sequence, the SNR and the MSE graphs from TSS and SSA algorithms are almost identical. In the mobcal sequence there are significant performance differences, with TSS and SSA graphs lying between FS and BS graphs. However, as can be noticed in tables 1-4, there is a remarkable difference in computation complexity and required encoding time between algorithms. Each table provides:

1. The number of times the MAD subroutine was called. This number is directly connected to the number of candidate locations that each algorithm examines.
2. Response time (process end time minus start time).
3. The CPU time consumed by each process.
4. The number of direct I/O performed.
5. The corresponding number of Page Faults.

First measure is quite accurate as it is independent of the CPU characteristics and refers absolutely to the algorithm under consideration.

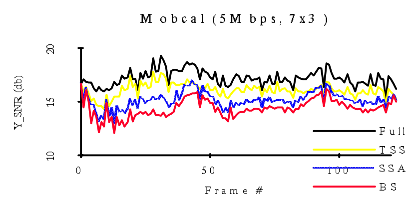

Fig. 6. Luminance SNR of the decoded frames for the FS, the BS, the TSS and the SSA using the mobcal sequence (5Mbits/s, 7x3 searching window).

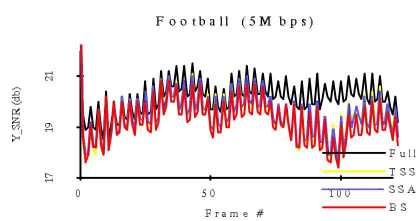

Fig. 7. Luminance SNR of the decoded frames for the Full search, the BS, the TSS and the SSA using the football sequence (5Mbits/s, 15x7 searching window).

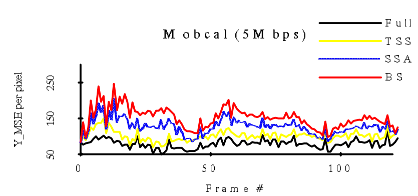

Fig. 8. Luminance MSE of the decoded frames for the Full search, the BS, the TSS and the SSA using the mobcal sequence (5Mbits/s, 7x3 searching window).

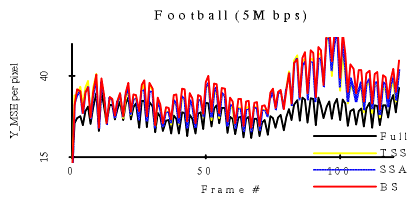

Fig. 9. Luminance MSE per pixel of the decoded frames for the Full search, the BS, the TSS and the SSA using the football sequence (5Mbits/s, 15x7 searching window).

Mobcal Sequence 5 Mbits/s, 7x3 searching window, 123 frames					
	Candidate Loc.	Resp. Time	CPU Time	Direct I/O	Page Faults
Full Search	311,175,457	1:08:24	9.78	2555	3013
BS	45,403,155	0:31:55	7.49	2422	1930
TSS	61,263,163	0:49:17	8.37	2488	1829
SSA	43,445,391	0:42:07	8.20	2524	1904

Table 1. Comparison of algorithms using the mobcal sequence.

Football Sequence 5 Mbits/s, 15x7 searching window, 123 frames					
	Candidate Loc.	Resp. Time	CPU Time	Direct I/O	Page Faults
Full Search	1,389,811,862	3:37:14	19.70	2644	3853
BS	52,353,824	0:32:44	7.40	2420	1745
TSS	65,096,295	0:44:28	8.99	2625	1990
SSA	52,669,576	0:35:58	7.93	2355	1735

Table 2. Comparison of algorithms using the football sequence.

Tennis Sequence 5 Mbits/s, 7x3 searching window, 123 frames					
	Candidate Loc.	Resp. Time	CPU Time	Direct I/O	Page Faults
Full Search	311,175,457	2:36:49	12.72	2588	2014
BS	46,141,993	0:55:44	8.80	2782	2009
TSS	60,366,191	1:15:33	9.16	2764	2112
SSA	43,386,611	0:45:27	8.31	2632	1903

Table 3. Comparison of algorithms using the tennis sequence.

Mobcal Sequence 7 Mbits/s, 7x7 searching window, 123 frames					
	Candidate Loc.	Resp. Time	CPU Time	Direct I/O	Page Faults
Full Search	311,175,457	3:03:17	13.14	2720	4332
BS	47,403,155	0:53:44	7.23	2472	1791
TSS	61,263,163	0:58:30	7.0	2741	1882
SSA	43,445,391	0:44:27	7.16	2564	1763

Table 4. Comparison of algorithms using the mobcal sequence.

5 CONCLUSIONS

An overview of fast block matching algorithms for motion estimation has been presented. Additionally, a new fast algorithm, the Spiral Search Algorithm (SSA), has been proposed. The SSA has a very good performance in terms of SNR and achieves significant computation time redundancy. The SSA evaluates the MAD only in 1/30 to 1/7 of the candidate locations resulting in a computation time smaller by a factor of 4 to 6 with respect to the time required by a full search algorithm. Besides, the performance of a full search is usually only 1 to 3 db better than the performance of the SSA. In comparison with other fast block matching algorithms, the SSA is faster than the TSS (by a factor of 1.3 to 1.5) and occasionally even faster than the BS. In terms of the quality of the decoded video sequences, the SSA is always better than BS and a little worse than TSS. Finally, the SSA algorithm is a very good compromise between quality of the decoded video sequence and computational complexity..

REFERENCES

[1] M.Ghanbari, “The Cross-Search Algorithm for Motion Estimation”, *IEEE Trans. on Commun.,* Vol.38, No.7, pp. 950-953, July 1990.

[2] Bee Liu, and Andre Zaccarin, “New Fast Algorithms for the Estimation of Block Motion Vectors”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.3, No.2, pp. 148-157, April 1993.

[3] MPEG, “ISO/IEC 13818-2, Generic Coding of Moving Pictures and Associted Audio at up to 1.5 Mbits/s”, *Recommendation H.262*, Committee Draft (pg. D-31).

[4] T. Koga, K. Iinuma, A,Hirano, Y. Iijima and T. Ishiguro, “Motion comprensated interframe coding for video conferencing,” in *Proc. Nat. Telecommun. Conf.,* New Orleans, LA, pp. G5.3.1-5.3.5, Nov. 29-Dec. 3, 1981

[5] Her-Ming Jong, Liang-Gee Chen and Tzi-Dar Chiueh, “Parallel Architecture for 3-Step Hierarchical Search Block-Matching Algorithm”, *IEEE Trans. on Circuits and Systems for Video Technology,* Vol. 4, No. 4, August 1994.

[6] Toshiyuki Urabe, Hassan Afzal, Grace Ho, Pramod Pancha and Magda El Zarki, “MPEGTool Version 1.03. README File”, Department of Electrical Engineering, University of Pennsylvania, Philadelpia.