RC23102 (W0402-023) February 5, 2004
Electrical Engineering

IBM Research Report

Motion Estimation Based on Spatio-Temporal Correlations and Pixel Decimation

V. Sheinin
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Motion Estimation Based on Spatio-Temporal Correlations and Pixel Decimation

V. Sheinin

Abstract— **Based on motion estimation using spatio-temporal correlations, an enhanced search algorithm is proposed to further improve the performance in terms of reducing total amount of computations while preserving the video quality. Our experimental results show that the proposed algorithm has substantially smaller computational complexity having some quality reduction. For sequences with low motion PSNR reduction is around 0.1 dB and computational complexity is around 1700 times less than that of a full search.**

***Index Terms*—Block matching, full search algorithm, motion estimation, search area determination**

I. Introduction

The full search (FS) block-matching algorithm is the simplest, but computationally very intensive approach. It provides an optimal solution by exhaustively evaluating all the possible candidates within the search range in the reference frame. In recent years there was substantial progress in block motion estimation algorithms. Important milestones on this path were such algorithms as two-dimensional logarithmic search [1], three-step search [2], four step search [3], diamond search (DS) [4]. More recently, even further improvements were made in the field. This can be explained by a very vivid evolution of software-based implementations of video decoders for such compression standards as MPEG-2/4, H.261, H.263, H.264 and others. For example, hexagonal search [5] has shown a significant improvement over other fast algorithms such as DS. The results of MPEG-4 encoder profiling [15] show that Motion Estimation may consume 40-60% of the total execution time. It only stresses even more the fact that complexity of motion estimation algorithms should be kept minimal for software implementations.

But we think that it is very interesting and promising to find many good search methods of last couple of years that use a prediction of initial search location. It is well known that there is a high correlation between current block motion vector and its neighbors. Various search methods define neighbors differently but all of them use this information gaining substantial reduction in computational complexity. All above mentioned methods try to minimize amount of search points applying sum of absolute differences (SAD) or equivalent metrics for each point:

$$SAD = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} abs(x(i,j) - y(i,j)) \qquad (1)$$

In Eq. (1) N is a block size, x(i,j) denotes the current block and y(i,j) denotes the candidate block.

There were some works, e.g. [6], that studied partial SAD (PSAD) but mostly concentrated on a constant decimation factor. The idea of PSAD is as follows: not all pixels are used to calculate the sum of absolute differences between the current block and the candidate one – but only particular set of pixels, e.g. every second, forth etc. What we tried to study in this work is the performance of one of the best block matching search algorithms in combination with adaptive PSAD as a matching metric. The idea is that we use original motion estimation based on spatio-temporal correlation method [7] but instead of using SAD as a matching metric we use PSAD with adaptively chosen decimation factor (DF). Computational complexity of such an approach is by far lower than that of the original algorithm. Additionally, there is no computational cost to determine the DF because this is done anyway by an encoder for the sake of bit-rate loop.

The remainder of the paper is organized as follows. In Section II we describe the proposed algorithm, Section III presents our simulation results and Section IV concludes the paper.

II. Enhanced Motion Estimation Based on Spatio-Temporal Correlation and Adaptive PSAD

In the original motion estimation based on spatio-temporal correlation method [7] there are three principles that make this scheme efficient. First principle uses prediction of initial search location based on three spatial neighbors and one temporal. The second principle can be called coarse or fine search procedures depending upon initial motion vector value. The third principle uses man square error (MSE) for the initial motion vector and if it's less than a threshold the initial motion vector is used as the final answer. Additional complexity of the algorithm is present due to the fact that MSE has to be calculated. In our experiment we didn't use

Manuscript received 9, 2004. V. Sheinin is with the IBM T.J Watson Research Center, Yorktown Heights NY 10598 USA phone: 914-945-1627; fax: 914-945-1602; e-mail: vadims@ us.ibm.com).

this condition partly because of the additional complexity and partly because of the unspecified threshold mechanism.

As shown in [6], using PSAD gives a very high gain in reducing computational complexity still having slight decrease of video quality for more static sequences and bigger decrease for fast motion ones. But basically if one accepts this quality decrease or uses the proposed method for compression in more static environments such as video surveillance or video conferencing, using decimation factor of 4 reduces computational complexity by factor of 4! This can be compared to trying to improve existing motion estimation methods and reduce complexity by 10-20%.

Since block matching is based on the idea that all pixels in a block move by the same amount, a good estimation of the motion could be obtained by employing only a fraction of these pixels. But for flat blocks it's enough to use only few pixels – and this fact allowed us to develop an adaptive scheme where the decimation factor for PSAD is defined based on the complexity of the block. In our simulation we used pixel variance (VAR) in a macroblock as a measure of a complexity as well as "modified variance" (MVAR) using absolute values instead of squared ones:

$$VAR = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(x(i,j) - mean)^2 \qquad (2)$$

$$MVAR = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}abs(x(i,j) - mean) \qquad (3)$$

In Eq. (1) and (2) mean denotes the mean pixel value of the current macroblock.

There was almost no difference in performance. Our final simulation results presented in Section III are based on absolute value. It is interesting to mention that an encoder calculates complexity of each block for the sake of bit-rate allocation. That is why in our complexity analysis we didn't add complexity calculation to the one of the proposed motion estimation method.

The proposed algorithm operates as follows. For each block we calculate its complexity as was described above. If this complexity value is less than a threshold, decimation factor (DF) is set to 8, otherwise it is set to 4. Threshold value was set to 60 but we saw that the algorithm is very robust with respect to variation of the threshold value. Then we perform the original motion estimation based on spatio-temporal correlations in which we use PSAD with decimation factor of DF as our block matching metric.

The proposed method is summarized as follows:

Step 1: Calculate current block complexity CMPL using MVAR (or use the complexity value from other encoder blocks)

Step 2: If CMPL < THRESHOLD, DF = 8, otherwise DF = 4

Step 3: Perform original motion estimation based on spatio-temporal correlations algorithm with PSAD instead of SAD and DF as defined in Step 2.

Moreover, we choose decimated pixels in such a way that the SIMD implementation of our algorithm is easily achieved. Fig.7 in [6] shows cases for DF = 4 and DF=8 when block size is a classical macroblock of 16x16 pixels. Colored pixels are used in PSAD calculation.

Other complexity metrics can be used depending on the availability of such metrics in a given encoder implementation.

The technique of adaptive PSAD can be used with any motion estimation scheme.

III. Simulation Results

The experimental set-up is as follows. MPEG-2 software encoder was used with the following parameters: open GOP with length of 14, 1 B frame, output bit-rate 12 Mbps, frame only encoding mode, motion estimation range +-25 H and +-16 V. We used 4:2:2 sequences of D1 resolution, each one of 20 frames, "Calendar", "Bike", "Basketball", and "Carrousel". These sequences are complex enough and have high motion. To test more static sequences we used "Lizard", "Sub", and "Free". MPEG-2 software encoder was able to accept motion vectors from external file where we put them as a result of our proposed method. We also calculated the full search results within the specified search window. The distortion measurement was PSNR, which is defined by

$$MSE = (\tfrac{1}{MN})\sum_{m=1}^{M}\sum_{n=1}^{N}[x(m,n) - y(m,n)]^2 \qquad (4)$$

$$PSNR = 10\log_{10}\frac{255}{MSE} \qquad (5)$$

In Eq. (1) x(m,n) denotes the original image, y(m,n) denotes the image after compression/decompression. M and N are horizontal and vertical dimensions of the image.

From Table 1 and Table 2 we see that for high motion sequences PSNR degradation between full search and the original motion estimation method [7] was around 0.1-0.7 dB. When we compare the original method with the proposed one the degradation increases to 0.1-0.6 dB. The computational complexity reduction of 650-1700 times (compared with the full search) and 9 times (compared to the original method) is pretty big and may be well worth this video quality decrease. In the case of more static sequences PSNR degradation between full search and the original motion estimation method [7] was around 0 dB. When we compare the original method with the proposed one the degradation increases to 0.1 dB. The computational complexity reduction was around 1600-1700 times (compared with the full search) and 5-7 times (compared to the original method).

IV. Conclusions

Based on the original motion estimation method with spatio-temporal correlations a new method is proposed in this paper. The proposed method uses partial sum of absolute differences as a matching metric whereas a decimation factor is adaptively selected based on the complexity of a processed block. Experiments show that the speed-up of the proposed method comparing to the original one is in a range of 7 whereas video quality reduction was about 0.1 dB. The proposed method presents a good trade-off between computational complexity and video quality and can be used in any encoding scheme.

References

[1] J. Jain, A. Jain, "Displacement measurements and its application in interframe image coding", *IEEE Trans. Commun., vol. COM-29, pp. 1799-1808, December 1981*.

[2] T. Koga, K. Iinuma, Y. Hirano, Y. Iijim, T. Ishiguro, "Motion compensated interframe coding for video conference", *In Proc. Nat. Telecommun. Conf., pp. G5.3.1-G5.3.5, November 1981*.

[3] P. Lai-man, M. Wing-Chung, " A Novel Four-Step Search Algorithm for Fast Block Motion Estimation", *IEEE Trans. Circuits Syst. Video technology*, vol. 6, No. 3, pp. 313-317, 1996.

[4] J. Tham, S. Ranganath, M. Ranganath and A. Kassim, " A Novel Unrestricted center-Biased Diamond Search Algorithm for Block Motion Estimation", *IEEE Trans. Circuits Syst. Video technology*, vol. 8, No. 4, pp. 369-377, August 1998.

[5] C. Zhu, X. Lin and L. Chau, " Hexagon-Based Search Pattern for Fast Block Motion Estimation", *IEEE Trans. Circuits Syst. Video technology*, vol. 12, No. 5, pp. 349-355, May 2002.

[6] F.Moschetti, M.Kunt and E.Debes, "A statistical adaptive block-matching motion estimation", *IEEE Trans. Circuits Syst. Video technology*, vol. 13, No. 4, April 2003.

[7] H.Yoon, G. Lee, "Motion Estimation Based on Spatio-Temporal Correlations", ICIP-2003, September 14-17, 2003, Barcelona, Spain.

[8] C. Zhu, X. Lin and L. Chau, "An Enhanced Hexagon-Based Search Pattern for Fast Block Motion Estimation", *IEEE Trans. Circuits Syst. Video technology*, vol. 12, No. 5, May 2002.

[9] Y. Chan, W. Hui and W. Siu, "A Block Motion Vector Estimation Using Pattern Based Pixel Decimation", *IEEE International Symposium on Circuits and Sysems*, June 9-12, 1997, Hong Kong.

[10] B. Liu, A. Zaccarin, "New fast Algorithms for the Estimation of Block Motion Vectors", ", *IEEE Trans. Circuits Syst. Video technology*, vol. 3, No. 2, pp. 148-157, April 1993.

[11] J. Xu, L. Po, C. Cheung, "Adaptive Motion Tracking Block Matching Algorithms for Video Coding", *IEEE Trans. Circuits Syst. Video technology*, vol. 9, No. 7, pp. 1025-1029, October 1999.

[12] K. Chung, L. Chang, "A New Predictive Search Area Approach for Fast Block Motion Estimation", *IEEE Trans. Circuits Syst. Video technology*, vol. 12, No. 6, pp. 648-652, June 2003.

[13] S. Zhu, K. MA, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", *IEEE Transactions on Image Processing*, vol. 9, No. 2, pp. 287-290, February, 2000.

[14] L. Kuo, E. Feig, "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding", *IEEE Trans. Circuits Syst. Video technology*, vol. 6, No. 4, pp. 419-422, August 1996.

[15] S. Akramullah, G. Rajan, "MPEG-4 adavanced simple profile video encoding on an embedded multimedia system", Proc. Of VCIP, Electronic Imaging, San Jose, California, USA, January 2004.

TABLE I
AVERAGE PSNR OF THE TEST IMAGE SEQUENCES

	Full Search	Original Method	Original method plus PSAD with DF=4	Proposed method	Performance degradation against full search, dB	Performance degradation against original method, dB
Calendar	33.49	33.37	32.88	32.77	0.72	0.6
Bike	35.2	34.52	34.26	34.27	0.93	0.25
Lizard	35.75	35.76	35.72	35.67	0.08	0.09
Basketball	30.75	31.02	31.02	30.98	0.23	0.04
Carrousel	35.96	35.96	35.92	35.86	0.1	0.1
Sub	41.21	41.21	41.18	41.15	0.06	0.06
Free	46.83	46.84	46.80	46.78	0.05	0.06

TABLE II
AVERAGE NUMBER OF 256-PIXEL SAD-COMPUTATIONAS

	Full Search	Original Method	Original method plus PSAD with DF=4	Proposed method	Speed-up against full search	Speed-up against original method
Calendar	2160000	12419	3001	2131	1013	5.8
Bike	2160000	24280	4902	2707	798	8.9
Lizard	2160000	8536	2143	1273	1698	6.7
Basketball	2160000	39058	6372	3302	654	11.8
Carrousel	2160000	28752	5335	2942	734	9.7
Sub	2160000	9655	2381	1354	1595	6
Free	2160000	8568	2339	1285	1681	5