EFFICIENT PARTIAL DISTORTION SEARCH ALGORITHM FOR BLOCK BASED MOTION ESTIMATION

Mohammed Golam Sarwer and Q.M. Jonathan Wu

Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada
sarwer@uwindsor.ca, jwu@uwindsor.ca

ABSTRACT

The block based full search algorithm has been widely used for motion estimation in video coding, but it has the serious problem of significant computation requirements. In order to reduce the computation, this paper proposes a novel partial distortion search algorithm which reduces the computation of each distortion measure by using partial distortion search. In this algorithm, the entire macroblock is divided into different sub-blocks and the calculation order of partial distortion is determined based on the complexity of sub-blocks. A lossy algorithm is also presented by adaptively changing the early termination threshold for the current accumulated partial sum of absolute difference value. Experimental results show that the proposed lossless and lossy algorithm can significantly save about 60% and 70% of the total computational costs, respectively. PSNR degradation of lossy algorithm is very negligible and 0.016 dB on average.

Index Terms— Video coding, motion estimation, motion compensation, partial distortion

1. INTRODUCTION

Block based motion estimation is a key component of many video coding standards due to its high efficiency in reducing temporal redundancy between successive frames. However, motion estimation (ME) is also the most computationally intensive part in a typical video encoder. The partial distortion search (PDS) [1] is one of the excellent fast ME methods and removes unnecessary computations efficiently and can be easily realized in very large scale integration. The PDS algorithm reduces the computational complexity by terminating the measuring distortion (sum of absolute difference: SAD) calculation early when it finds that a partial SAD is already greater than the minimum SAD encountered so far in the searching. Even though PDS is a lossless fast matching approach, it can be also used in a lossy way. The well known PDS-based version for lossy ME is the normalized partial distortion search (NPDS) [2]. In the NPDS, partial distortion and the current minimum distortion are normalized on the number of checked pixels before comparison. However, NPDS has a limitation on the maximum computational reduction and the performance of visual quality [7].

In order to improve the efficiency of lossless and lossy PDS algorithms, several approaches are presented in the literature. The enhanced normalized partial distortion search based on the block similarity is presented in [3]. An adaptive matching scan strategy to quickly reject the unnecessary candidates is introduced in [4]. A sorting based lossless partial distortion based motion estimation in presented in [5]. Pixel positions are sorted according to the gradient magnitude between adjacent pixels. However, gradient is performed by pixel by pixel operation, it takes extra computations. A clustered pixel matching error for adaptive PDS is proposed by Hui et al. [6] in which matching order is determined by using the characteristics of clustered pixels matching error. A two stage sorting based PDS algorithm by using the characteristic of pattern similarity matching error is described in [7, 8]. The basic idea in [9] is to eliminate invalid candidates earlier by predicting a total matching error between matching and candidate block. However this prediction is based on the linear model which results significant degradation of image quality. A ME algorithm by adaptively changing the early termination threshold for current accumulated SAD value is develop in [10] in order to reduce the complexity of H.264/AVC. Recently, a Hadamard transform based partial distortion search algorithm (HPDS) uses sum of DC and AC Hadamard transform coefficients in order to determine the calculation order of partial differences [11]. But Hadamard transform results large extra computation which make this algorithm inefficient to real time implementation.

This paper proposes an efficient sorting based PDS algorithm. The entire macroblock (MB) is divided into 16 4x4 sub-blocks. The calculation orders of sub-blocks are sorted based on their complexity. The rest of the paper is organized as follows. In section 2, review of PDS is briefly explained. Section 3 describes the algorithm of proposed lossless and lossy PDS methods. Simulation results are presented in Section 4. Finally, Section 5 concludes the paper.

2. PARTIAL DISTORTION SEARCH (PDS)

Partial distortion search is a technique used to introduce early termination in the calculation of SAD. In block-matching ME, a block of *MxM* pixels can be divided into a number of small groups, k, such that the distortion of the k-th group, d_k, can be measured. The k-th partial SAD to check during the matching is

$$d_k = \sum_{i=1}^{k}\sum_{j=1}^{M}\left|f_t(x+i,y+j) - f_{t-1}(x+i+mx,y+j+my)\right| \quad (1)$$

where $M \times M$ is the size of the block, M is equal to 16 for a macroblock (MB), (x,y) is the position of the MB being coded, mx and my are the horizontal and vertical component of candidate motion vector (MV) and $f_t(.,.)$ and $f_{t-1}(.,.)$ represent the luminance pixel intensity of the current frame and reference frame, respectively. The partial SAD , which is the matching error accumulated for every period is computed and compared with the minimum SAD already found (with another candidate vector). Once this is larger than the minimum SAD at each period, the candidate block cannot be the most similar block regardless of the rest of the incomplete matching computations. Therefore, the PDS algorithm can find and remove impossible candidates before complete matching error calculation of candidate block. In (1), the matching is performed by row by row and the test is performed after every row.

3. PROPOSED PDS ALGORITHM

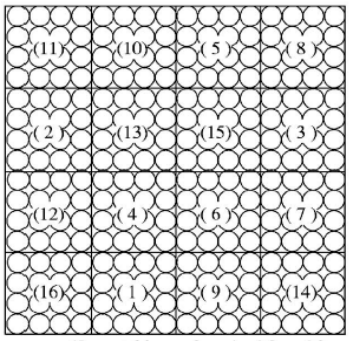
3.1. Proposed lossless PDS algorithm

The ability of a PDS to reject the impossible candidates is also affected by the searching strategy, that is, the order in which blocks are tested during the searching phase. The earlier the global minimum is met in a search, the earlier the PDS can terminate a partial SAD to reject the candidates. To achieve this purpose, we use the following searching strategy described in Table 1.

Table 1: Searching order

Order	Candidate	Description
1	Origin	center of search window (0,0)
2	MV1	motion vector of left block
3	MV2	motion vector of upper block
4	MV3	motion vector of upper left block
5	MV4	motion vector of upper right block
then	Spiral search of all of the remaining search point within the search window	

The matching strategy, that is, to say, the order in which pixels within a block are picked up to compute the SAD, affects the speed of the ME; in fact; if the highest contribution to SAD are found early, then the distortion bound may be reached after small number of differences and partial sum can be stopped. The proposed algorithm divides the entire target MB into different sub-blocks as shown in Fig. 1. In Fig. 1, the 16x16 MB is divided into 16 4x4 sub blocks.



(*i*): matching order, $i = 1,2,\ldots,16$

Fig. 1: Partition of a MB

The main idea is based on the assumption that a higher complex block results large distortion. In order to classify the complexity of block, we have used absolute deviation from its mean value. The absolute deviation of k-th block in the MB is defined as

$$AD_k = \sum_{i=1}^{4}\sum_{j=1}^{4}\left|f_k(i,j) - \mu_k\right| \quad (2)$$

where f_k is the pixel intensity of (i,j) th position of the k-th sub-block and μ_k is the mean pixel intensity of k-th sub-block. After calculation of all AD_k , the partial SAD is calculated in the order of block with greatest AD_k to smallest AD_k. This enables ME more quickly in the SAD calculation of a candidate position. The P'th accumulated partial distortion is defines as

$$D_p = \sum_{k=1}^{p} SAD_k \quad (3)$$

where SAD_k is the sum of the absolute distortion of k-th 4x4 block. Firstly, AD_k of all 16 sub-block is calculated by (2) and calculation order is determined based on the absolute deviation. During the SAD computation if partial SAD (D_p) is greater than the SAD_{min}, it terminates the remaining partial SAD computation and jumps to the next search position.

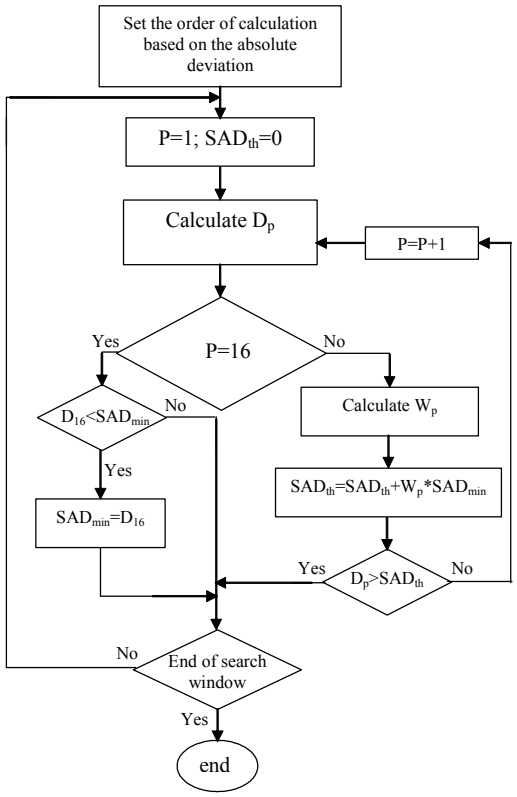


Fig. 2 Flow diagram of lossy PDS algorithm

3.2. Proposed lossy PDS algorithm

Fig. 2 shows the flow diagram of the proposed lossy PDS algorithm. The lossy algorithm adopts the similar approach of lossless PDS. However, during the accumulation of SAD, we use a threshold value SAD_{th} instead of minimum SAD. The comparison starts from p=1 and proceeds towards p=16, and comparison is stopped if partial distortion is greater than the threshold $SAD_{th.}$ At the end of comparison (i.e., p=16), if D_{16} is smaller than the SAD_{min}, then this candidate motion vector becomes a new current minimum point. By comparing with the SAD_{th}, computational complexity is reduced by the high rejection of impossible candidates an early stage. The adaptive threshold for p-th partial distortion is defined as

$$SAD_{th(p)} = SAD_{th(p-1)} + W_p * SAD_{\min} \quad (4)$$

$$\text{for p=1,2,3…16 and } SAD_{th(0)} = 0$$

Here SAD_{min} is the current minimum distortion and W_p is the weighting factor for p-th partial distortion. W_p represent the contribution of the p-th sub-block to SAD calculation. It is reasonable to say that if the deviation of a block is higher, then the contribution of this block in SAD computation is also higher. Therefore, we can conclude that deviation of the k th block AD_k, is proportional to the distortion. From this observation, W_p is defined as

$$W_p = \frac{AD_p}{\sum_{p=1}^{16} AD_p} \quad (5)$$

3.3. Overhead Computation

It is shown that the proposed method introduces some additional computations in order to construct the calculation order. The calculation of the mean pixel intensity μ_k of k-th sub-block requires 15 additions and one division operation. This division operation is implemented by shifting right by four bits. For each pixel, (2) shows that each absolute deviation needs one absolute operation and one subtraction. Hence, a total of 62 operation (16 absolute operations, 16 subtractions, 15 additions, 15 additions for μ_k and one shift right) are required for the calculation of AD_k in (2) for each sub-block. To obtain the final calculation order, a sorting process is required. Sorting of 16 numbers is not too complex unit because it need only comparator operation. In case of lossy algorithm, the denominator of weighting factors in (5) is same for all p and it needs 15 additions. Hence, 15 additions and 16 divisions (one for each p) are required in order to calculate the weighting factors of entire MB. Although it is shown that a lot of overhead computations are introduced in the proposed algorithm, the simulation results verified that the resulting motion estimation complexity is lower than conventional PDS algorithm.

4. SIMULATION RESULTS

In order to compare the performance of proposed algorithm five different types of video sequences (Akiyo, Foreman, Stefan, Flower and Bus) with CIF format are tested. We use 50 frames of all image sequences. Matching block size is 16x16 pixels and the search window is ± 16 pixels.

Table 2 shows the computational reduction in terms of the average number of checking partial SAD per MB and percentage of partial SAD saving by the tested algorithms. As we predicted, we can find that the proposed method can successfully improve the computational efficiency of the conventional PDS. Since the proposed method introduces the overhead computations, number of checking partial SAD is not suitable criteria to compare the complexity of the algorithm. That's why we have used motion estimation time reduction as an alternate measure. The execution time reduction ($\Delta T\%$) is calculated as follows:

$$\Delta T\% = \frac{T_{FS} - T_{M(i)}}{T_{FS}} \times 100 \quad (6)$$

where T_{FS} is the motion estimation time of the FS method and $T_{M(i)}$ is the motion estimation time of method i. The results are tabulated in Table 3. It is shown that the proposed lossless method saves about 5% of computation of PDS algorithm. The results for PSNR performance are same for lossless algorithm.

Table 2: Average number of computed partial SAD per MB of different methods

Method	Average number of partial SAD/MB					pSAD save (%)
	Stefan	Fore-man	Akiyo	Flower	Bus	
FS	16	16	16	16	16	
PDS	4.38	4.09	1.68	5.51	5.52	73.1
HPDS[11]	3.57	3.47	1.26	4.56	4.59	78.1
Proposed	3.25	3.17	1.25	4.35	4.50	79.3
NPDS[9] (lossy)	1.09	1.07	1.01	1.11	1.12	93.2
Proposed (lossy)	1.18	1.17	1.02	1.27	1.19	92.7

Table 3: % of execution time (ΔT%) reduction

Method	Stefan	Fore-man	Akiyo	Flower	Bus	Average
PDS	55.1	57.5	68.7	50.3	49.5	56.2
HPDS[11]	56.7	59.4	70.4	52.4	52.5	58.2
Proposed	60.2	62.2	72.6	54.8	55.1	60.9
NPDS [9] (lossy)	71.2	72.7	73.3	71.7	73.2	72.4
Proposed (lossy)	69.4	70.8	72.3	70.7	71.0	70.8

The lossy method achieves about 70% of computation of full search algorithm and more similar computation with NPDS algorithm. The PSNR reduction of lossy PDS algorithm is tabulated in Table 4 which confirmed that PSNR degradation is very low and is 0.016 dB whereas NPDS reduces the PSNR of is 0.17 db.

Table 4: PSNR comparison of proposed lossy PDS

Method	PSNR in dB					PSNR reduction in dB
	Stefan	Fore-man	Akiyo	Flower	Bus	
FS	26.36	33.93	42.92	26.35	24.39	
NPDS[9] (lossy)	26.20	33.71	42.77	26.24	24.17	0.172
Proposed (lossy)	26.33	33.94	42.90	26.34	24.36	0.016

5. CONCLUSIONS

In this paper, we proposed a new block matching algorithm by sorting square sub-blocks. By using a calculation order according to block complexity, we can obtain faster elimination of impossible candidate vectors than PDS algorithms. In order to increase the computational saving, a lossy algorithm is also successfully developed. The lossy PDS algorithm adaptively changes the weighting factors of early termination threshold. Experimental results show that, this adaptive thresholding technique reduces about 70% of computation without significant degradation of video quality.

6. ACKNOWLEDGEMENTS

This work has been supported in part by the Canada Research Chair Program, AUTO21 NCE, and the NSERC Discovery grant.

7. REFERENCES

[1] S. Eckart and C. Fogg, ‘ ISO/IEC MPEG-2 software video codec,” *Proc. SPIE 2419*, 100-118 (1995).

[2] C. K. Cheung and L. M. Po, “ Normalized partial distortion algorithm for block motion estimation,” *IEEE Trans. on. Circuit. Syst. Video Tech.*, 10(3), 417-422 (2000).

[3] Won- Gi Hong, and Tae-Myoung Oh, “Enhanced partial distortion search algorithm for block motion estimation,” *Electronic Letters*, 39(15), 1112-1113, 2003.

[4] J.N. Kim, S.C. Byun, Y. H. Kim, and B. H. Ahn, “ Fast full search motion estimation algorithm using early detection of impossible candidate vectors,” *IEEE Trans. on signal Processing*, 50(9), 2355-2365, (2002).

[5] B. Montrucchio, and D. Quaglia, “ New sorting based lossless motion estimation algorithms and a partial distortion elimination performance analysis,” *IEEE Trans. on. Circuit. Syst. Video Tech*., 15(2), 210-220 (2005).

[6] K. C. Hui, W. C. Siu, and Y.L.Chan, “ New adaptive partial distortion search using clustered pixel matching error characteristics”, *IEEE Transaction on image processing*, 14(5), 597-607 (2005)

[7] C. C. Wang, and C. J. Lo, “ Using two stage sorting based partial distortion search for motion estimation in H.264/AVC,” *Optical Engineering*, 46(9), 097002, September, 2007.

[8] C. C. Wang, C. J. Lo, C. W. Yu, “ Efficient motion estimation using sorting based partial distortion search,” *ICME 2006*, 153-156.

[9] S.L.Shin, S. Lee, and J.S.Oh, “Fast partial difference elimination algorithm based on block matching error prediction,” *OE letters*, 46(4), 040503, April 2007

[10] E. A. Qaralleh, and T.S.Cheng, “ Fast variable block size motion estimation by adaptive early termination,” *IEEE Trans. on. Circuit. Syst. Video Tech*., 16(8), 1021-1026 (2006).

[11] S. Jin, H. Lee, and J. Jeong, “ Hadamard transform based fast partial distortion elimination algorithm for lossless and lossy motion estimation,” *2008 Congress of Image and Signal Processing*, 201-205.