Fast partial difference elimination algorithm based on block matching error prediction

Se-Ill Shin,[a] Sangkeun Lee,[b] and Jeong-Su Oh[a]
[a]Pukyong National University, Division of Image and Information Engineering, San 100, Yongdang-Dong, Nam-Gu, Pusan, 608-739 Korea
[b]Samsung Information and Systems America, Digital Media Solutions Lab, 3345 Michelson Drive #250, Irvine, California 92612
E-mail: ojs@pknu.ac.kr

Abstract. We introduce a fast partial difference elimination (PDE) algorithm for motion estimation (ME). The basic idea of the proposed approach is to eliminate invalid candidates earlier by predicting a total matching error between matching and candidate blocks. The matching error prediction is performed by using a partially computed matching error between the blocks. Experimental results show that the proposed algorithm can significantly save about 40% of the averaged computational costs versus the conventional PDE algorithm for ME at the cost of ignorable image quality degradation, whose average value is 0.0012 dB. *© 2007 Society of Photo-Optical Instrumentation Engineers.*
[DOI: 10.1117/1.2721453]

Subject terms: motion estimation (ME); block matching; partial difference elimination (PDE).

Paper 060870LR received Nov. 4, 2006; revised manuscript received Jan. 13, 2007; accepted for publication Jan. 29, 2007; published online Apr. 5, 2007.

1 Introduction

A block-based full search algorithm (FSA) has been widely used for motion estimation (ME) in video encoding, but it has the serious problem of significant computation requirements.[1] In order to solve this problem, many fast algorithms have been developed, and partial difference elimination (PDE) algorithms[1–4] are good examples. PDE algorithms reduce computations by eliminating impossible candidates before the complete matching error calculation between matching and candidate blocks. PDE algorithms have also been improved by selecting a low initial block matching error[1,2] or arranging the line/sub-block matching order in a candidate block.[3,4] In this letter, we propose a fast PDE algorithm for reducing additional computational costs based on the property of block matching error. The proposed algorithm predicts a total block matching error from a partially computed matching error between the blocks and removes impossible candidates earlier than conventional algorithms. Simulation results show that the proposed algorithm can save a large amount of computations with ignorable image degradation in comparison with conventional PDE algorithms.

0091-3286/2007/$25.00 © 2007 SPIE

2 Conventional PDE Algorithms

The FSA finds the most similar block to a matching block within a given search range of the reference frame. The similarity between blocks is measured by the block matching error and is often computed by the sum of absolute difference (SAD) between matching and candidate blocks. In the PDE algorithm, the partial SAD ($pSAD$), which is the matching error accumulated for every period, such as a block's line[3] or a sub-block,[4] is computed and compared with the minimum block SAD (min_bSAD). Once the $pSAD$ is larger than the min_bSAD at each period, the candidate block cannot be the most similar block regardless of the rest of the incomplete matching computations. Therefore, the PDE algorithm can find and remove impossible candidates before the complete matching error calculation of the candidate block. In this letter, we follow the line-based SAD comparison. The accumulated matching error at the k'th line of a candidate block ($pSAD^k$) in the (x,y) position of the given search range can be expressed by

$$pSAD^k(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{N-1} \left| f_t(i,j) - f_{t-1}(i+x,j+y) \right|, \quad k = 0,1,\ldots,N-1, \tag{1}$$

where N is the matching block size, and f_t and f_{t-1} indicate current and reference frames, respectively. To summarize the procedures of the conventional PDE algorithm, the approach computes the $pSAD^k$ at every line of a candidate block, compares it with the min_bSAD, and then moves to the next line if the $pSAD^k$ is less than the min_bSAD. It is noted that the min_bSAD is updated only when a complete matching error computation ($k=N-1$) is finished if necessary.

The efficiency of the PDE algorithm can be improved when a small initial block matching error is selected or large line/sub-block matching errors occur early. A spiral PDE algorithm[3] and a sorting-based PDE algorithm[4] are typical examples, respectively, and they are compared with the proposed PDE algorithm for performance evaluation.

3 Proposed Fast PDE Algorithm

The disadvantage of the PDE scheme is that the algorithm has to compute the $pSAD^k$ of a candidate block for $k=0,\ldots,N-1$, which will be eliminated last, until the $pSAD^k$ reaches min_bSAD. It is obvious that predetermination of the elimination will help in reducing the computational costs and increasing the process speed for the ME.

A fast PDE algorithm is proposed by predicting a total block SAD ($bSAD$), which is equivalent to the $pSAD^{N-1}$. The proposed algorithm is based on the property that a $pSAD^k$ is an intermediate value and is gradually increased to reach the $bSAD$ because the $pSAD^k$ is continuously accumulated as k goes from 0 to $N-1$. The prediction at the k'th line of a block (p_bSAD^k) is performed, whenever the $pSAD^k$ for $k<N-1$ is less than the min_bSAD by using the $pSAD^k$, as follows:

$$p\_bSAD^k = pSAD^k + w(g\_SAD^k)(N-k),$$

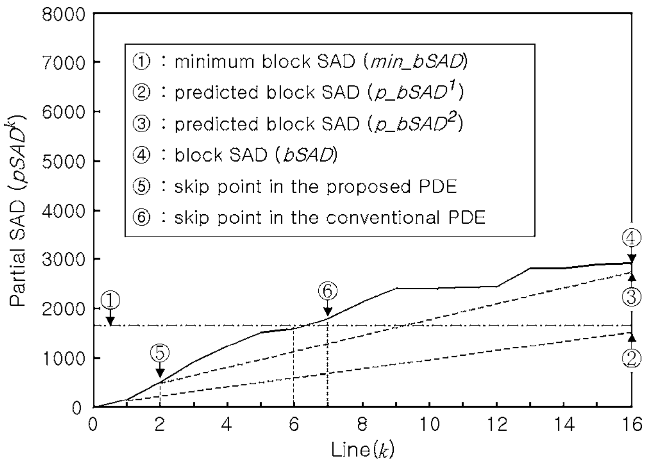

Fig. 1 Procedures for reducing the computational cost based on the block SAD prediction.

$$g\_SAD^k = pSAD^k/k, \quad k = 1, \ldots, N-1, \tag{2}$$

where g_SAD^k is an averaged line-matching error of $pSAD^k$, and w is a weighted value determined by the complexity of a matching block to avoid an incorrect prediction. The incorrect prediction may be caused by a large g_SAD^k, which is caused by large line-matching errors at the candidate blocks containing the complex image such as edges. So w is selected in inverse proportion to the image complexities and defined by using an average of available $bSAD$s (avg_bSAD) in the neighboring blocks including the matching block, as follows:

$$w = \begin{cases} 0.8, & \text{if } avg\_bSAD \leqslant 300, \\ 0.8 - \dfrac{0.7}{600}(avg\_bSAD - 300) & \text{if } 300 < avg\_bSAD < 900, \\ 0.1 & \text{if } avg\_bSAD \geqslant 900. \end{cases} \tag{3}$$

Here, numerical values are empirically selected for all of the tested sequences.

The proposed approach computes a p_bSAD^k to predict the $bSAD$ from the $pSAD^k$ if the $pSAD^k$ for $k < N-1$ is less than the min_bSAD and compares the predicted block SAD (p_bSAD^k) with the min_bSAD before the next line computation. Therefore, the proposed algorithm has a high possibility to eliminate impossible candidates earlier than the conventional approach because the p_bSAD^k always exceeds the $pSAD^k$. It is worth noting that the p_bSAD^k is adjusted adaptively according to the contents of each matching block because the accuracy of the prediction is affected by complexity of the block. We provide a practical example to show the procedures of the proposed algorithm in Fig. 1. First, a $pSAD^1$ is computed and compared with a min_bSAD (see ① in Fig. 1). A p_bSAD^1 is estimated because the $pSAD^1$ is less than the min_bSAD and is compared with the min_bSAD. The p_bSAD^1 (see ② in Fig. 1) is still smaller than the min_bSAD, and the algorithm goes to the next line. Similarly, a $pSAD^2$ and a p_bSAD^2 are calculated at the second line because the $pSAD^2$ is less than the min_bSAD. However, the p_bSAD^2 (see ③) is larger than the min_bSAD. Therefore, the proposed scheme skips the rest of the matching procedures for the current candidate block and goes to the next candidate, while the conventional PDE keeps performing the same procedures until the $pSAD^k$ exceeds the min_bSAD at the seventh line (see ⑥). Consequently, the proposed algorithm requires only two line matching events, while the conventional algorithm requires seven line matching events.

Table 1 Comparisons with respect to image quality and prediction ability.

	s_PDE, S_PDE	P_PDE		
	PSNR	PSNR	Difference	Mismatched MV
Foreman	32.2338	32.2304	−0.0051	1.311
Stefan	24.6757	24.6733	−0.0032	2.599
Akiyo	45.8185	45.8175	0.0006	0.043
Mobile	26.0088	26.0088	−0.0001	0.027
Container	43.0933	43.0944	0.0000	0.395
Silent voice	35.0830	35.0809	−0.0031	0.221
News	36.2600	36.2596	0.0014	0.171
Table tennis	31.2572	31.2560	−0.0024	0.472
Average	34.3038	34.3026	−0.0012	0.6547

4 Simulation Results

The proposed PDE algorithm (P_PDE) is simulated with various video sequences—Foreman, Stefan, Akiyo, Mobile, Container, Silent voice, News, and Table tennis—and they consist of 300 frames at 30 Hz in the format of QCIF. The matching block size is 16×16, and the search range is ±16. The simulation results are compared with the spiral PDE algorithm (s_PDE)[3] and the sorting-based PDE algorithm (S_PDE).[4] It is noted that both the P_PDE and the S_PDE also employ a spiral scanning searching scheme, as the s_PDE does. Table 1 shows the performance of the proposed algorithm with respect to PSNR and matching ability. It can be seen that the degradations in PSNR and matching ability are only 0.0012 dB and 0.66% (0.65 block in 99 matching blocks) average values, respectively. In Table 1, mismatched MV indicates the number of mismatched motion vectors in the proposed algorithm versus the corresponding motion vectors of the conventional PDE algorithm.

In order to evaluate the computational performance, the average checked line numbers per candidate block to determine its validity for three algorithms are summarized in Table 2. It is noted that the proposed scheme requires 1 addition, 1 division, and 2 multiplications more to estimate a total block matching error if necessary. The results in Table 2 show that the proposed method reduces

Table 2 Average checked line numbers per candidate block.

	Average checked lines		
	s_PDE	*S_PDE*	*P_PDE*
Foreman	3.309	2.688	1.874
Stefan	4.236	3.827	2.529
Akiyo	1.236	1.107	1.032
Mobile	3.429	2.964	1.676
Container	2.169	1.691	1.094
Silent voice	1.914	1.804	1.314
News	1.723	1.420	1.238
Table tennis	3.306	2.709	2.015
Average	2.665	2.276	1.596
Efficiency (%)	40.110	29.880	0.000

computational cost on average 40% and 30% compared with the s_PDE[3] and S_PDE,[4] respectively. The efficiency value of the proposed approach versus the conventional PDE algorithm is defined by

$$efficiency = \frac{PDE - P\_PDE}{PDE} \times 100, \tag{4}$$

where PDE and P_PDE are the conventional PDE and the proposed PDE algorithms, respectively.

5 Conclusion

In this letter, we proposed a fast PDE algorithm to reduce the computational cost of the conventional PDE algorithms. We predict a block matching error from a partial block matching error and remove the impossible candidates earlier using the predicted block matching error. The simulation results show that this can considerably reduce the computational complexity at the cost of negligible image quality degradation. We believe that the proposed approach can be applied to the conventional PDE algorithms without significant modifications and can play an important role as a significant tool for fast motion estimation.

References

1. S. Eckart and C. Fogg, “Iso/iec mpeg-2 software video codec,” *Proc. SPIE* **2419**, 100–118 (1995).
2. ITU-T Recommendation H.263 software implementation, Digital Video Coding Group at Telenor R&D (1995).
3. 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. Signal Process.* **50**(9), 2355–2365 (2002).
4. B. Montrucchio and D. Quaglia, “New sorting-based lossless motion estimation algorithm and a partial distortion elimination performance analysis,” *IEEE Trans. Circuits Syst. Video Technol.* **15**(2), 210–220 (2005).