New Adaptive Partial Distortion Search Using Clustered Pixel Matching Error Characteristic

Ko-Cheung Hui, Wan-Chi Siu, and Yui-Lam Chan

***Abstract*—In order to reduce the computation load, many conventional fast block-matching algorithms have been developed to reduce the set of possible searching points in the search window. All of these algorithms produce some quality degradation of a predicted image. Alternatively, another kind of fast block-matching algorithms which do not introduce any prediction error as compared with the full-search algorithm is to reduce the number of necessary matching evaluations for every searching point in the search window. The partial distortion search (PDS) is a well-known technique of the second kind of algorithms. In the literature, many researches tried to improve both lossy and lossless block-matching algorithms by making use of an assumption that pixels with larger gradient magnitudes have larger matching errors on average. Based on a simple analysis, it is found that, on average, pixel matching errors with similar magnitudes tend to appear in clusters for natural video sequences. By using this clustering characteristic, we propose an adaptive PDS algorithm which significantly improves the computation efficiency of the original PDS. This approach is much better than other algorithms which make use of the pixel gradients. Furthermore, the proposed algorithm is most suitable for motion estimation of both opaque and boundary macroblocks of an arbitrary-shaped object in MPEG-4 coding.**

***Index Terms*—Motion estimation, partial distortion search (PDS), video coding.**

I. Introduction

THE BLOCK-BASED motion compensation technique has been widely used in many modern video-coding standards [1], [2]. It is used to reduce the redundancy between successive frames in a video. In these video-coding schemes, a frame is divided into a number of nonoverlapping macroblocks (MBs). By the block-based motion compensation technique, the values of pixels in a MB in the current frame are predicted from another MB of pixels in a reference frame. The displacement between these two MBs is defined as a motion vector. A motion estimation process in a video encoder computes this motion vector. The motion vector and the prediction residual errors are then coded in a bitstream and transmitted to a decoder.

Manuscript received January 29, 2003; revised May 12, 2004. This work was supported in part by research studentships from the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, Hong Kong Polytechnic University, and in part by the Research Grant Council of The Hong Kong SAR Government (PolyU 5234/03E). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christoph Stiller.

The authors are with the Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: 99903700r@polyu.edu.hk; enwcsiu@polyu.edu.hk; enylchan@polyu.edu.hk).

Digital Object Identifier 10.1109/TIP.2005.846020

The motion estimation process is to obtain a motion vector for a target macroblock by using the block matching technique, which minimizes a measure of matching distortion between the target MB in the current frame and a candidate MB within a search window in a reference frame. The displacement between the candidate MB with the smallest distortion and the target MB will be selected as the resulting motion vector. One of the most frequently used criteria to measure the matching distortion is the sum of absolute difference (SAD). The SAD between a target MB at position (x, y) in the current frame, I_t, and a candidate MB at position $(x+u, y+v)$, in the reference frame, I_{t-1}, is defined as follows:

$$\mathrm{SAD}(x,y;u,v) = \sum_{j=0}^{M}\sum_{i=0}^{M} |I_t(x+i,y+j) - I_{t-1}(x+i+u,y+j+v)| \quad (1)$$

where $M \times M$ is size of a block and M is equal to 16 for our consideration; $I_t(\cdot,\cdot)$ and $I_{t-1}(\cdot,\cdot)$ represent pixels intensity in the current frame and the reference frame, respectively.

The simplest block matching motion estimation algorithm is the full search algorithm (FSA). This algorithm can give an optimal solution by exhaustively searching all possible locations within a search window. The resulting best motion vector $(\hat{u}, \hat{v})$ is

$$(\hat{u},\hat{v}) \equiv \arg \min_{(u,v)\in W} \mathrm{SAD}(x,y;u,v) \quad (2)$$

where $W = \{(u,v) | -D \le u,v \le D\}$ is a set of all possible locations in a search window and D is the maximum possible displacement of the motion vector (u,v). However, this algorithm suffers from heavy computational load. In order to resolve this difficulty, many fast search algorithms have been developed in the past.

The fast search algorithms can be classified into four categories. 1) In the first category, the fast search algorithms seek for a way to select a subset of the candidate MB in W to reduce the computational time [3]–[5]. This is most challenging part of these algorithms. Because these algorithms can easily be trapped into local minima, degradation in predicted images is an inevitable result on average. Many researchers select an initial searching point by studying the motion field to reduce the probability of being trapped into local minima. Another approach is to find a good initial point which is an hierarchical or multiresolution technique [6], [7]. 2) Algorithms in this category use a reduced complexity distortion measure to save computation, such as pixel decimation [8]–[10] and partial distortion (PDS) techniques [11], [12]. The pixel decimation technique [8] subsamples the pixels in the target MB and the candidate MBs. Hence,

the computation for each SAD can be reduced. The PDS [11] reduces the computation complexity by terminating the SAD calculation early when it finds that a partial SAD is already greater than the minimum SAD encountered so far in the searching. In general, the PDS is regarded as a fast full search algorithm because it has identical prediction quality as that of the FSA. 3) Algorithms in the third category make use of mathematical inequalities to reduce the computational load; this includes the successive elimination algorithm (SEA) [13]. By making use of the Minkowski's inequality, the SEA eliminates an impossible candidates MB without calculating the SAD. 4) The fourth category is to use a combination of the above techniques to further improve the coding efficiency, [14] and [15].

In this paper, we suggest further techniques to improve the searching efficiency of the PDS. Its efficiency also comes from an early termination of the partial SAD. Let us define a generalized form of the partial SAD as follows:

$$\mathrm{SAD}_p(x,y;u,v) = \sum_{j=0}^{p}\sum_{i=0}^{15} |I_t(x+k_n, y+l_n) - I_{t-1}(x+k_n+u, y+l_n+v)| \quad (3)$$

where $\{(k_n, I_n)|n = 0,\ldots,j\times 16+i,\ldots,256\}$ is an index set of all pixels in a MB, and $p = 0,\ldots,15$ which specifies the number of elements for producing the sum of errors for a partial SAD. For a given p, there are $16\times(p+1)$ pixels to be accumulated to SAD_p. The index set determines the coordinates and orders of the pixel matching errors to be accumulated to the SAD_p. One simple idea to improve the PDS is to design an adaptive index set such that a pixel with greater matching error is firstly computed, and this error is accumulated to the SAD_p earlier than other pixels. As a result, the SAD calculation can be terminated sooner. In the literature, many researches [16], [17] indicated that pixels with larger gradient magnitudes have larger matching errors on average. They made use of this hypothesis to develop their searching algorithms. An approach is to make use of representative pixels and adaptive matching scan PDS (AMS-PDS) [16] to determine the index set by sorting the gradient magnitude of rows or columns in the target MB of the current frame in descending order. As a result, pixels of a row or column with greater gradient magnitudes will be used to calculate the SAD_p prior to other rows or columns in the MB. However, it can be shown that pixel matching errors with similar magnitudes tend to appear in clusters in natural video sequences. This characteristic is illustrated in Fig. 3 and discussed in the next Section. In this paper, we propose an adaptive partial distortion search algorithm by using the characteristics of these clustered pixel matching errors. This approach is significantly more efficient as compared to algorithms which make use of pixel gradient properties in an adaptive partial distortion search.

In the rest of this paper, we first explain and illustrate the characteristics of the pixel errors that tend to form clusters in Section II. Section III applies these characteristics to develop a new clustered pixel matching error for adaptive partial distortion search algorithm (CPME-PDS). In Section III-A, we establish an analysis to determine an adaptive index set required for the CPME-PDS. Then, our proposed CPME-PDS is described in details in Section III-B. It is unavoidable to have a certain amount of overheads for the establishment of the adaptive index set. These overheads are described in Section III-C. Section IV gives the details of our experiments and results. In order to compare the performance of the adaptive PDS based on the cluster pixel matching error characteristic and based on the pixel gradient characteristic, we have also designed another adaptive PDS, the pixel gradient-based adaptive partial distortion search algorithm (PG-PDS). The details of the PG-PDS are described in Section IV-A. Section IV-B presents the results and analysis of the CPME-PDS comparing to other fast algorithms including the PG-PDS. Finally, concluding remarks are given in Section V.

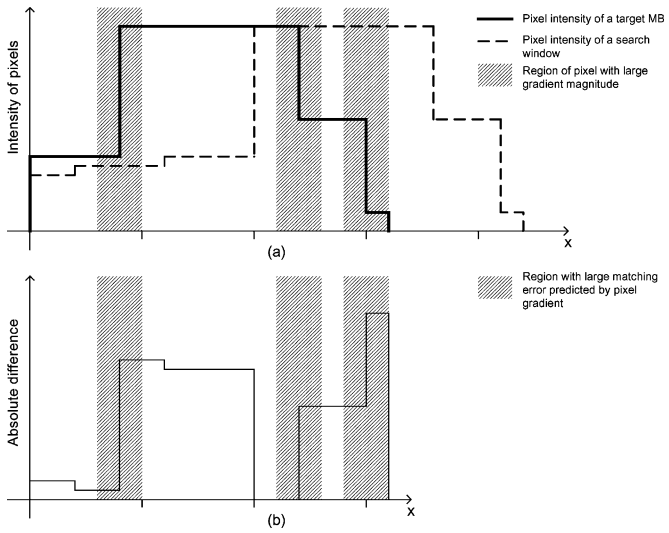

Fig. 1. (a) Matching of a 1-D target MB within a 1-D search window. (b) Corresponding pixel matching error of the target MB at the current position.

II. Characteristic of CPME

The major idea of our proposed CPME-PDS is to design an adaptive index set such that a pixel with greater matching error can be accumulated to the SAD_p sooner than other pixels according to the order indicated by the index set.

For this reason, it is necessary for us to investigate possible spatial distributions of pixel matching errors in a MB. We have found that errors with similar magnitude tend to appear together in clusters. It is because natural images are dominated by low-frequency components. The matching errors of low-frequency regions between a target MB and a candidate MB have similar magnitudes and are partitioned by edge pixels of these two MBs. This phenomenon is demonstrated in Fig. 1. Fig. 1(a) depicts the matching of a one-dimensional (1-D) target MB (thick continuous line) within a 1-D search window (thin dotted line). The corresponding pixel matching errors appear in a cluster form as shown in Fig. 1(b).

Edges are the most prominent feature in image processing. They are also frequently used to predict pixel matching errors in motion estimation. The prediction is accurate especially near a minimum distortion position. Fig. 2(b) shows that locations with large pixel matching errors (the hatched region) can be detected by using pixel gradients when the target MB is located near a good candidate MB. However, the result is not good enough in general. In Fig. 1(b), only pixel matching errors in the hatched

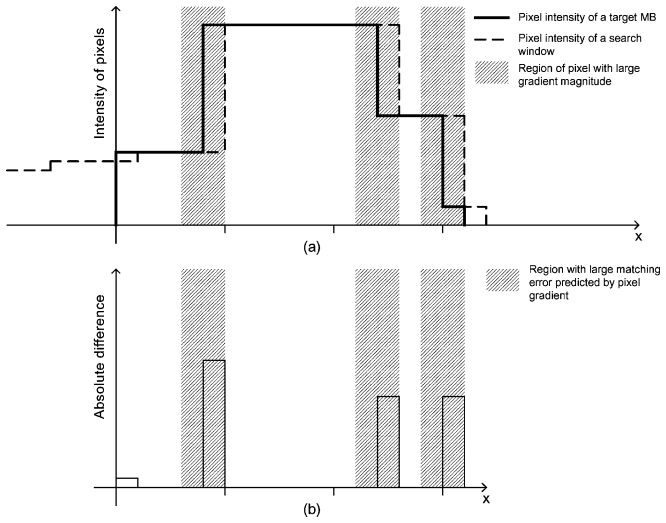

Fig. 2. (a) Matching of a 1-D target MB within a 1-D search window near a minimum distortion location. (b) Corresponding pixel matching error of the target MB at the current position.

region are found, while matching errors outside the hatched region are underestimated. Fig. 3(a) and (b) are examples of prediction errors in two motion compensated MBs, which are extracted from the sequence "Football" and video object "Goldfish," respectively. These examples demonstrate the clustered prediction errors during motion estimation.

According to the above analysis, we can predict that clustered pixel matching error characteristic can be used to achieve greater advantage in an adaptive partial distortion search.

III. Proposed Algorithm

A. Determination of an Adaptive Index Set

For a given target MB, the positions of pixels are represented by an index set $S = \{(k_n, l_n) | n = 0, \ldots, N-1\}$, where N is the number of pixels in a MB. For a single pixel at $s_n = (k_n, l_n)$, $s_n \in S$, its matching error is $e(s_n) = I_t(s_n) - R(s_n)$, where $R(s_n)$ is a random variable which represents the pixel value at s_n of a candidate MB. In the following discussion, the notation for properties relating to the pixel at s_n is indicated by the argument n, and the MB location (x, y) and motion vector (u, v) are dropped for simplicity. Hence, the pixel matching errors are represented by

$$e(n) = I_t(n) - R(n). \tag{4}$$

To improve the saving in computation of a PDS, pixel matching errors with an ideal index set must have the following relation:

$$e(0)^2 \geq \ldots \geq e(n)^2 \geq \ldots \geq e(N-1)^2.$$

To fulfill the above objective, we have to predict the pixel matching error of each $p(n)$ at location s_n. Hence, the expected values of $p(n)$ must fulfill the following criterion:

$$E\left[p(0)^2\right] \geq \ldots \geq E\left[p(n)^2\right] \geq \ldots \geq E\left[p(N-1)^2\right]. \tag{5}$$

Let us define $p(n) = I_t(n) - m$, where m is a reference value to be used to obtain the predicted pixel matching errors.

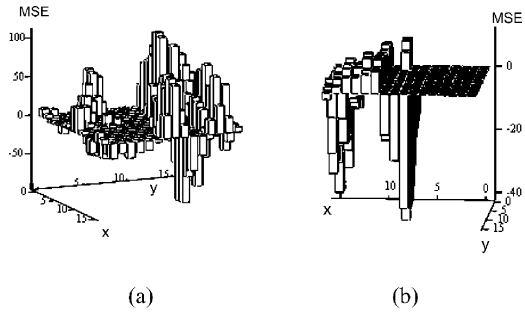

Fig. 3. Examples of clustering errors in a motion compensated prediction MB of (a) the sequences "Football" and (b) the video object "Goldfish."

One possible solution of m is to minimize the expected value of the sum of squares of the differences between $e(n)^2$ and $p(n)^2$, i.e.

$$m = \arg\min_m \left\{ E\left[\sum_{n=0}^{N-1} \left[(I_t(n) - R(n))^2 - (I_t(n) - m)^2 \right]^2 \right] \right\}.$$

In solving this equation, we have

$$\frac{d}{dm} E\left[\sum_{n=0}^{N-1} \left[(I_t(n) - R(n))^2 - (I_t(n) - m)^2 \right]^2 \right] = 0. \tag{6}$$

By substituting $R(n) = I_t(n) - e(n)$ into (6), finally, we have a cubic equation

$$m^3 - 3\overline{I_t} m^2 + \left(3\overline{I_t^2} - \overline{e^2}\right) m + \overline{I_t e^2} - \overline{I_t^3} = 0 \tag{7}$$

where $\overline{e^2} = (1/N) E[\sum_{n=0}^{N-1} e(n)^2]$, and

$$\overline{I_t e^2} = \frac{1}{N} E\left[\sum_{n=0}^{N-1} I_t(n) e(n)^2 \right].$$

The roots of the cubic equation are either all reals or one real and two complex conjugates which depend on the discriminant of the equation. We look for real roots for (7), such that m can be practically useful. Let us assume that natural images are dominated by low-frequency components. Hence, let

$$\overline{I_t}^a \approx \overline{I_t^a}$$

and

$$\overline{I_t} \cdot \overline{e^2} \approx \overline{I_t e^2}.$$

These are valid only if the image frame under question consists mainly low requencies and the standard deviation of $I_t(n)$ is small enough. As a result, it gives

$$m \approx \begin{cases} \overline{I_t} \\ \overline{I_t} \pm \sqrt{\overline{e^2}} \end{cases}. \tag{8}$$

The first approximated root is the mean of pixel values in the target MB. To use this mean as the reference value it already gives a better computational saving when comparing to the PG-PDS, for which we proposed it as a comparison. Intuitively, m is a function of pixel values in a candidate MB, i.e.

$m = m(R(n))$. The other roots $\overline{I_t} \pm \sqrt{\overline{e^2}}$ can also be obtained by the following approximation:

$$\overline{R} = \overline{I_t} - \overline{e} \approx \overline{I_t} \pm \sqrt{\overline{e^2}}.$$

It indicates that this solution is an approximation of the mean of pixel values in a candidate MB. Note that our assumption is not always true. However, a shifting of m would not affect the criterion in (5) dramatically. In fact, the solution of $m = \overline{R}$ can also be obtained directly by minimizing the equation $E[\sum_{n=0}^{N-1}[e(n) - p(n)]^2]$. Hence, this result is used to determine an adaptive index set for the CPME-PDS.

B. CPME-PDS

There is another factor which affects the ability of a PDS to reject impossible candidates. 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 two strategies as shown below.

1) The outward spiral scanning is used to exploit the center-biased motion vector distribution characteristics of the real-world video sequence [18].
2) The correlation in the motion field is exploited by using a median predictor of three adjacent blocks (left, top, and top right blocks) to the current position as the initial searching point of the spiral scanning. We have used the median predictor described in [19].

According to the above considerations and our analytical results, we suggest to use the mean of pixel values in the candidate MB of the initial searching point to compute the reference value m because we can assume that

$$\overline{I_{t-1}(i + u_{\mathrm{med}}, j + v_{\mathrm{med}})} \approx \overline{I_t(i,j)}$$

where $(u_{\mathrm{med}}, v_{\mathrm{med}})$= the median predictor. The expected pixel matching error $p_{\exp}(n)$ of each pixel in the target MB is calculated with m. The required adaptive index set S is given by sorting $|p_{\exp}(n)|$ in descending order. The partial SAD in (3) is calculated with S during the searching in an outward spiral scanning. The CPME-PDS approach can be summarized as follows.

CPME-PDS:

Note that all division operations in the following description are integer division with truncation toward zero for the sake of lower complexity.

Step 1) Determine the median predictor $(u_{\mathrm{med}}, v_{\mathrm{med}})$ of the three adjacent blocks.

Step 2) Calculate the reference value m with the median predictor $(u_{\mathrm{med}}, v_{\mathrm{med}})$

$$m = \frac{1}{256}\sum_{j=0}^{15}\sum_{i=0}^{15} I_{t-1}(x + u_{\mathrm{med}} + i, y + v_{\mathrm{med}} + j). \quad (9)$$

Step 3) Initialize an index set $S' = \{(k'_n, l'_n) | n = 0, \ldots, N-1\}$ which represents all pixels of the target MB.

Step 4) Calculate the expected absolute pixel matching error $|p_{\exp}(n)|$ of each pixel in the target MB

$$|p_{\exp}(n)| = |I_t(k'_n, l'_n) - m|. \quad (10)$$

Step 5) Rearrange the order of set S' to obtain an adaptive index set S by sorting the expected absolute pixel matching error $|p_{\exp}(n)|$ in descending order, such that

$$S = \{(k_n, l_n) | n = 0, \ldots, N-1\}.$$

The $p_{\exp}(n)$ corresponding to the order of the sorted index set S has the following feature

$$|p_{\exp}(0)| \geq \ldots \geq |p_{\exp}(n)| \geq \ldots \geq |p_{\exp}(N-1).|$$

Step 6) Apply the adaptive index set S to calculate the partial SAD in (3) during the searching in an outward spiral scanning.

Note that the adaptive index set is established on a pixel-based approach. It is straightforward to modify the above procedure for the boundary macroblocks of an arbitrary shaped video object (VO) in MPEG-4 [19]. First, the reference value m is calculated after that the repetitive padding is applied to a reference video object plane (VOP). It is only necessary to compute the expected pixel matching error $p_{\exp}(n)$ for opaque pixels in the case of a boundary MB. For an index set, N is equal to the number of pixels in a MB, such that $N = 256$ for an opaque MB, while N=number of opaque pixels in a boundary MB. Second, the partial SAD in (3) is rewritten as

$$\mathrm{SAD}_p(x, y; u, v) = \sum_{j=0}^{p}\sum_{i=0}^{q} |I_t(x + k_n, y + l_n) - I_{t-1}(x + k_n + u, y + l_n + v)| \quad (11)$$

where $p \in \{\aleph | 0, \ldots, \alpha;\ \alpha =$ integer division of $N/16$ with truncation toward zero

$$q = \begin{cases} 15, & p \neq \alpha \\ N - 16 \times \alpha, & p = \alpha \end{cases} \quad \text{and}$$

$\aleph$ is the set of Natural Numbers.

The design of our algorithm depends upon the clustering properties of the pixel matching errors. Hence, the approach is general and it is expected to be useful for both software or hardware realization. Let us consider that for example, most computer architectures tend to be in favor of regular memory pattern access and execution [20]. It is true for the modern Intel processors, say for example, which provide multimedia extensions (MMX), streaming single-instruction multiple data extensions (SSE), and SSE2 technologies. These technologies offer a set of instructions for handling a large quantity of data in parallel efficiently. The MMX and SSE instruction set can compare eight bytes or eight pixel values from each of two blocks with a single instruction, thus accelerating the program by almost a factor of four or eight. The SSE2 instruction set can operate data with 16 pixels at one time effectively.

In order to make use of the advantage of clustering characteristic, we may also arrange to sort pixels row by row in a block for SAD_p accumulation. By using an identical reference value, m, we can calculate the expected absolute pixel error of a row of 16 pixels as follows, to determine the accumulating order

$$p_{\text{exp}} = \sum_{x=0}^{15} \left| I_t\left(x, l'_n\right) - m \right|. \tag{12}$$

Because these instructions demand an increase in memory bandwidth, we have simulated three different situations to evaluate their performances, i.e.,, sorting a row of 4, 8, and 16 consecutive pixels in a block designated as CPME-PDS_4, CPME-PDS_8, and CPME-PDS_{16}, respectively.

C. Analysis of the Overhead

From the above description, it is shown that the additional computation introduced by the CPME-PDS is the process to construct the adaptive index set for each target MB in the current frame or VOP.

The calculation of the reference value m, as shown in (9), requires 255 additions and one division. For each opaque pixel, (10) shows that each expected absolute pixel matching error $|p_{\text{exp}}(n)|$ needs one absolute operation and one subtraction. Hence, N absolute operations and N subtractions are required for the calculation of $|p_{\text{exp}}(n)|$ for each target MB. In the case of a boundary MB, 256 additional checkings are needed to ensure that only the opaque pixels are involved. To obtain the final adaptive index set S, a sorting process is required in step 5) of Section III-B. Because the values of $|p_{\text{exp}}(n)|$ are integers ranging from 0 to 255, the counting sort [21] which has the complexity of $O(N)$ is the most appropriate sorting algorithm in this situation. In general, it requires $2 \times N$ increment/decrement operations and $z - 1$ additions, where z is the largest integer in the data set being sorted. For a boundary MB, (11) shows that the formulation of SAD_p is different from that of an opaque MB. As shown in (11), the computation of α involves one division, and the computation of q when $p = \alpha$ needs one multiplication and subtraction.

Note that all multiplications and divisions mentioned above can be implemented with simple bitwise shift operations. In our analysis, however, each multiplication or division is counted and assumed to be equivalent to eight additions for simplicity.

IV. Experiments

In Section III, we have proposed the CPME-PDS which makes use of the characteristics of clustered pixel matching errors to improve the searching efficiency of the conventional PDS. In this section, let us modify the adaptive PDS to become PG-PDS in order to compare saving in computation. A large amount of experimental work has been done. We describe the PG-PDS in brief in the next part. Approaches using representative pixels and AMS-PDS [16], conventional PDS, PG-PDS, and SEA [13] were also implemented for the sake of comparison.

A. PG-PDS

In the PG-PDS, an adaptive index set S_{pg} is obtained based on the magnitude of individual pixel gradient.

For each pixel in an opaque MB, let us express the magnitudes of x directional gradients, G_x, and y directional gradients G_y as

$$\begin{aligned} |G_x(x,y)| &= |I_t(x,y) - I_t(x+1,y)| \\ &\quad \text{where } x = 0, 1 \ldots, 14; y = 0, 1 \ldots, 15 \\ |G_y(x,y)| &= |I_t(x,y) - I_t(x,y+1)| \\ &\quad \text{where } y = 0, 1 \ldots, 14; x = 0, 1 \ldots, 15. \end{aligned} \tag{13}$$

There are $15 \times 16 = 240$ gradient values for each direction. Hence, totally 480 gradient magnitudes need to be found for an opaque MB. These magnitudes are sorted in descending order with a counting sort. The S_{pg} is then established by extracting the pixel's position according to the order of the sorted gradient magnitudes. Obviously, each pixel must appear only once in S_{pg}. A proper checking procedure is needed to prevent double extraction of a pixel, because each pixel involves two directional gradient magnitudes. The adaptive index set S_{pg} is applied for the calculation of the partial SAD in (3) during the search in an outward spiral scanning.

There are some differences in the implementation of a boundary MB and an opaque MB. The total number of gradient magnitudes in a boundary MB depends on the number of opaque pixels and the shape in the MB. In calculating (13), if one of the involved pixel $I_t(\cdot,\cdot)$ is a transparent pixel, the magnitude of the corresponding gradient is regarded as zero. We have also used pixel gradients in a row to perform row-based sorting for comparison. Similar to the CPME-PDS, the average gradient of a row with 4, 8, and 16 consecutive pixels in a block has been used to determine the accumulating order of different rows in a SAD_p.

B. Experimental Results and Discussion

The analytical result in Section III suggests that two mean values can be used as the reference value, m, in the CPME-PDS. These are the mean of pixel values in the target MB, m_1, and the mean of pixel values in the candidate MB of the initial searching point m_2. In addition to these two mean values, we can also choose a third candidate $m_3 = 128$ because we assume that R is a random variable which represents the pixel values in a MB and $m = \overline{R}$.

Table I compares the computational load of the CPME-PDS with these three reference values. The results show that all three values can successfully improve the efficiency of the conventional PDS. Among these three values, m_2 provides the least computational load. Hence, it confirms our suggestion that the mean of pixel values in the candidate MB of the initial searching point is used as the reference value in the CPME-PDS.

To evaluate the performance of the CPME-PDS, we implemented six algorithms: 1) the FSA, 2) the conventional PDS, 3) the representative pixels and AMS-PDS, 4) the PG-PDS, 5) the SEA, and 6) the proposed CPME-PDS. The outward spiral scan was applied to all six algorithms to exploit the center-biased motion vector distribution characteristics. In addition, a median predictor was used as an initial searching centre to exploit the correlation in the motion field for all tested algorithms. The SEA

TABLE I
COMPARISON OF THE AVERAGE NUMBERS OF OPERATIONS PER MB OF THE CPME-PDS FOR DIFFERENT REFERENCE VALUES, m

	PDS	CPME-PDS		
		m_1	m_2	m_3
Video Sequences				
Children	188931	108120	99908	110958
Bream	219654	149539	139461	197122
Arbitrary Shaped Video Objects				
Football	299882	256443	232779	268849
Tabletennis	219509	159596	149103	173683

m_1 = The mean of pixel values in the target MB.
m_2 = The mean of pixel values in the candidate MB of the initial searching point.
m_3 = 128

uses the norm of each search point to speed up the processing. The norm of each search point is calculated for frame-based sequences and opaque MBs of a video object by using a recursive method suggested in [13]. However, the recursive technique is not suitable for boundary MB of an arbitrarily shaped object. We need to calculate each norm during the search and the required operations are counted as computational load for searching in the realization. Hence, it is seen that the SEA may require much computation for the motion estimation of arbitrarily shaped objects in MPEG-4. Because AMS-PDS is an algorithm developed only suitable for block-based motion estimation, experiments which involved arbitrary shaped video objects did not include AMS-PDS. The computational efficiency of the algorithms has been assessed in terms of the number of operations required for the searching. Each addition, subtraction, absolute or checking operation mentioned above was considered as one operation. Each multiplication or division was considered to be equivalent to eight additions for simplicity. All these operations were counted in runtime during the experiments. Moreover, nonuniform memory access is the major disadvantage of these adaptive PDS algorithms. To evaluate the practical performance, we have also measured the execution time for motion estimation including the required overheads of all tested algorithms for comparison. We performed the experimental work on a standard desktop computer. The configuration of the platform was Intel P-III 600 MHz desktop PC with 256 M RAM and Windows 2000.

We used a large variety of video sequences and video objects for the evaluation. Sequences "Football," "Table Tennis," "Stefan," "Salesman," "Foreman," "Grand Mother," "Suzie," and "Trevor" were used as test sequences, while "News," "Children," "Bream," and "Goldfish" were used to test arbitrary shaped video objects. The format of the above video sequences and video objects are summarized in Table II.

TABLE II
FORMAT OF THE TESTED VIDEO SEQUENCES AND VIDEO OBJECTS

Video Sequences		
Sequence	**Image format**	**num. of Tested Frames**
Football	352×240	209
Table Tennis	352×240	299
Stefan	352×240	299
Salesman	352×288	199
Forman	176×144	299
Grand Mother	176×144	299
Suzie	176×144	149
Trevor	176×144	149
Arbitrary Shaped Video Objects		
Video Object	**Source format**	**num. of Tested VOPs**
News	352×288	299
Children	352×288	299
Bream	352×288	299
Goldfish	352×288	299

According to the analysis in Section II, we have shown that the prediction of pixel matching errors based on pixel gradients is only accurate near a minimum distortion position. Results in Figs. 4 and 5 justify our analysis. These figures show the average numbers of operations with different distances from the centre of a search window for the tested algorithms. The average number of operations at distance, d, is obtained by the equation shown at the bottom of the page. For our initial analysis, we just counted the number of operations for the realization. All overheads, such as the time for memory access, etc., were not considered, since these overheads are usually machine dependent. Two typical results of the selected sequences, including the "Table Tennis" and "Bream," are provided in Fig. 4 and Fig. 5, respectively. In these figures, algorithms with the least number of operations per search point at a specified distance, d, are shaded in grey. These results confirm our prediction that PG-PDS has a better ability to save computation when a search point is near the minimum distortion position. When the search point location is extended, the performance of CPME-PDS overrides that of PG-PDS. For the "Grand Mother" sequence, however, the SEA provides the best efficiency when the search distance is greater than six. Table III summarizes the results of all tested sequences. Entries in Table III give the search distances in which the corresponding algorithms have the least numbers of operations. On the whole, we can see that our approach (the CPME-PDS) always provides the largest range of search distance to have the best performance. This is particular true for object-based sequences and sequences with high-motion activities.

$$\text{Average number of operations at distance, } d = \frac{\sum_{u=-d}^{d}\sum_{v=-d}^{d}\begin{pmatrix}\text{Number of operations}\\ \text{to calculate}\\ \text{SAD}_p(x,y;u,v)\end{pmatrix} - \sum_{u=-(d-1)}^{d-1}\sum_{v=-(d-1)}^{d-1}\begin{pmatrix}\text{Number of operations}\\ \text{to calculate}\\ \text{SAD}_p(x,y;u,v)\end{pmatrix}}{(2d+1)^2-[2(d-1)+1]^2}$$

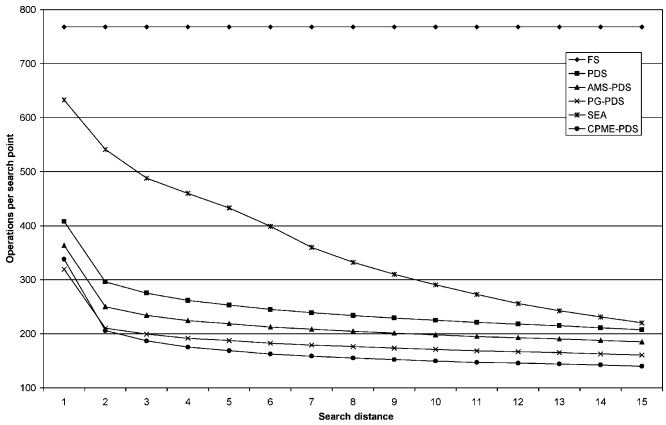

Fig. 4. Comparison between the computational saving capability of the tested algorithms at different distances from the centre of a search window for "Table Tennis."

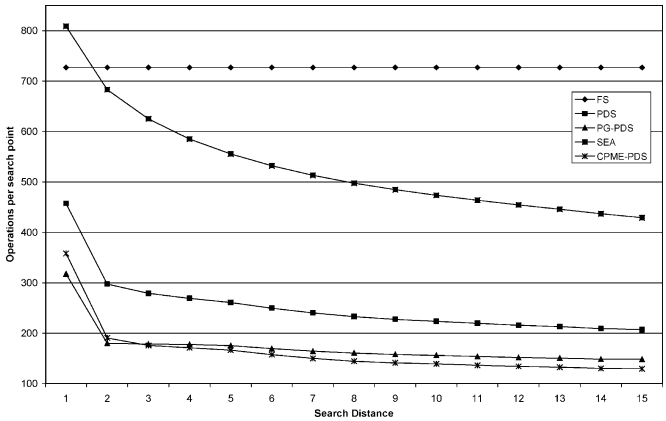

Fig. 5. Comparison of the computational saving capability of the tested algorithms at different distances from the centre of a search window for "Bream."

TABLE III
SUMMARY OF THE COMPUTATIONAL SAVING ABILITY OF THE TESTED ALGORITHMS FOR DIFFERENT SEQUENCES. THE ENTRIES INDICATE THE SEARCH DISTANCE AT WHICH THE CORRESPONDING ALGORITHMS REQUIRE THE LEAST NUMBER OF OPERATIONS

Video Sequences	FSA	SEA	PDS	AMS-PDS	PG-PDS	CPME-PDS
Football	x	x	x	x	1 - 2	3 - 15
Tabletennis	x	x	x	x	1	2 - 15
Stefan	x	x	x	x	1 - 2	3 - 15
Salesman	x	x	x	x	1 - 3	4 - 15
Foreman	x	15	x	x	1 - 5	6 - 14
Grand mother	x	6 - 15	x	x	1 - 4	5
Suzie	x	9 - 15	x	x	1 - 3	4 - 8
Trevor	x	10 - 15	x	x	1 - 4	5 - 9
Video Objects						
News	x	x	x	invalid	1 - 3	4 - 15
Children	x	x	x	invalid	1 - 2	3 - 15
Bream	x	x	x	invalid	1 - 2	3 - 15
Goldfish	x	x	x	invalid	1 - 4	5 - 15

x – Indicates that the algorithms are not the most efficient one at all search distances.
invalid – AMS-PDS was not designed for the motion estimation of arbitrary shaped video objects.

TABLE IV
COMPARISON OF THE OVERHEADS FOR DIFFERENT ALGORITHMS IN TERMS OF THE AVERAGE NUMBERS OF OPERATIONS PER MACROBLOCK

Video Sequences	FSA	SEA	PDS	*AMS-PDS	PG-PDS	CPME-PDS
Football	0	1158	0	1479	3686	1613
Tabletennis	0	1158	0	1479	3712	1618
Stefan	0	1158	0	1479	3723	1626
Salesman	0	1145	0	1479	3668	1597
Foreman	0	1272	0	1479	3678	1620
Grand mother	0	1272	0	1479	3658	1590
Suzie	0	1272	0	1479	3646	1583
Trevor	0	1272	0	1479	3668	1601
Video Objects						
News	0	1711	0	invalid	3373	1562
Children	0	2757	0	invalid	3196	1565
Bream	0	1614	0	invalid	3449	1588
Goldfish	0	1670	0	invalid	3199	1542

* The numbers of operations for sorting in the AMS-PDS were not counted in the experiments.
invalid – AMS-PDS was not designed for the motion estimation of arbitrary shaped video objects.

TABLE V
SUMMARY OF THE COMPUTATIONAL EFFICIENCIES IN TERMS OF OPERATIONS OF THE TESTED ALGORITHMS FOR DIFFERENT SEQUENCES. THE FIGURES INDICATE THE SIZES OF SEARCH WINDOW IN WHICH THE CORRESPONDING ALGORITHMS REQUIRE THE LEAST NUMBER OF OPERATIONS

Video Sequences	FSA	SEA	PDS	*AMS-PDS	PG-PDS	CPME-PDS
Football	x	x	1 - 2	x	x	3 - 15
Tabletennis	x	x	1	x	x	2 - 15
Stefan	x	x	1	x	x	2 - 15
Salesman	x	x	1 - 2	x	x	3 - 15
Foreman	x	x	1 - 2	x	4 - 7	3, 8 - 15
Grand mother	x	10 - 15	1 - 2	x	x	3 - 9
Suzie	x	15	1 - 2	3	x	4 - 14
Trevor	x	x	1 - 2	3	x	4 - 15
Video Objects						
News	x	x	1	invalid	x	2 - 15
Children	x	x	1	invalid	x	2 - 15
Bream	x	x	1	invalid	x	2 - 15
Goldfish	x	x	1	invalid	x	2 - 15

* The numbers of operations for the sorting in AMS-PDS were not counted in the experiments.
x – Indicates that the algorithms are not the most efficient one at all search distances.
invalid – AMS-PDS was not designed for the motion estimation of arbitrary shaped video objects.

On the other hand, the PG-PDS gives the largest computational saving among all tested algorithms with a short search distance, ranging from 0 to 4 on average. Furthermore, the SEA has the best performance for sequences related to video conferencing when the search distance is large enough. The number of operations of the AMS-PDS is about 4% on average smaller than that of the CPME-PDS for the sequence "Suzie" when the search range is smaller than 4. For the "Grand Mother" and "Foreman" sequences, the AMS-PDS is about 3% on average better than that of the CPME-PDS within a search range of 2. For "Football," "Salesman" and "Stefan" sequences, the number of operations of the AMS-PDS is about 2% smaller than that of the CPME-PDS within a unit search range.

TABLE VI
AVERAGE NUMBERS OF TOTAL OPERATIONS PER MB OF THE TESTED ALGORITHMS IN A SEARCH WINDOW WITH A SEARCH RANGE EQUAL TO 15 (i.e., $-15 \leq u, v \leq 15$)

Video Sequences	FSA		SEA		PDS		*AMS-PDS		PG-PDS		CPME-PDS	
	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio	No. of operation	Speed-up ratio
Football	738048	1.00	351897	2.10	299882	2.46	279341	2.64	263057	2.81	232779	3.17
Tabletennis	738048	1.00	290482	2.54	219509	3.36	194414	3.80	170269	4.33	149103	4.95
Stefan	738048	1.00	304156	2.43	241830	3.05	206729	3.57	191230	3.86	169652	4.35
Salesman	738048	1.00	172550	4.28	160614	4.60	140468	5.25	126625	5.83	111693	6.61
Foreman	738048	1.00	129522	5.70	153344	4.81	123309	5.99	110916	6.65	106310	6.94
Grand mother	738048	1.00	98119	7.52	157032	4.70	129139	5.72	126619	5.83	121658	6.07
Suzie	738048	1.00	118877	6.21	170286	4.33	137157	5.38	135321	5.45	121301	6.08
Trevor	738048	1.00	82068	8.99	110638	6.67	93004	7.94	89113	8.28	81240	9.08
Video Objects												
News	689274	1.00	321915	2.14	188931	3.65			147191	4.68	138052	4.99
Children	653006	1.00	478403	1.36	188931	3.46			116043	5.63	99908	6.54
Bream	698661	1.00	464301	1.50	219654	3.18			155870	4.48	139461	5.01
Goldfish	663847	1.00	353271	1.88	226663	2.93			151615	4.38	140129	4.74

Let us turn our attention to the overheads, such as sorting processes, etc. Table IV lists the average numbers of operations of these overheads for the tested algorithms. In Table V, we have summarized the average number of total operations per search point in a given search range D (i.e., $-D \leq u$ and $v \leq D$). In this case, overheads are also included. In our implementation, quick sort was used as the sorting approach for AMS-PDS. Its complexity is $O(n \log n)$, where n is equal to 16 in the case of the AMS-PDS algorithm. The selection of a sorting algorithm affects seriously the performance of an adaptive PDS especially if the search window is small. In order to prevent an underevaluation of the AMS-PDS, the number of operations for its sorting process was not counted and this assumption is made in all of the latter discussion.

Table V summarizes the results of algorithms which require the least number of operations with the indicated search range. Generally speaking, CPME-PDS gives the best performance in a search range within 2 to 15 for nearly all sequences. The SEA achieves the best efficiency within a search range from 10 to 15 and 15 for the "Grand mother" and "Suzie" sequences, respectively. It is interesting to point out that the PG-PDS offered the best efficiency in the middle search range, within 4 to 8, for the sequence "Foreman." In Table III, we can see that the PG-PDS gives the best computational saving for d equal to 0 to 5 in "Foreman," but it requires the largest overheads as shown in Table IV. Hence, it needs more computational saving to outperform other algorithms, the situation of which is reflected in the table. The efficiency of CPME-PDS outperforms that of PG-PDS when the search range is extended sufficiently. In terms of the total number of required operations, we have found that the performance of the PG-PDS is about 1.7% better than the CPME-PDS for only one sequence (the "Foreman" sequence) in the medium search range.

Table VI demonstrates a comparison between the computational efficiency of the tested algorithms in a search window of 15 (i.e., $-15 \leq u, v \leq 15$). The computational efficiency was compared in terms of the average number of operations per MB and speedup ratios. It shows that our algorithm, CPME-PDS, can successfully improve the computational efficiency of the conventional PDS and is the best among all other adaptive PDSs. In terms of speedup ratios, it can achieve a speedup ranging from three to nine times of that of the FSA. The SEA gives a worse performance as compared to our algorithm CPME-PDS for most sequences, but achieves better efficiency for sequences on vide conferencing, such as the "Grand mother" and "Suzie."

Let us evaluate the execution time of all algorithms. The evaluation takes into the account of both computation loads of the algorithms and their overheads. This is, to some extent, CPU dependent. Table VII and Table VIII compare the execution time per frame or per VOP of the algorithms with a search range of 15. Table VII clearly shows that the improvements of all PDS algorithms are reduced. On average, the speedup ratios in terms of the number of operations are decreased by about 33%, 40%, 52%, and 48% for the PDS, AMS-PDS, PG-PDS, and the CPME-PDS, respectively. The SEA is only degraded by about 9%. This phenomenon is caused by two factors. The first factor is that all adaptive PDS algorithms suffer from the problem of nonuniform memory access. Among these three adaptive PDS, the AMS-PDS has the minimum degradation because it makes use of pixel gradients to determine the sorting of rows or columns in a MB. When the column scanning is used in AMS-PDS, nonuniform memory access problem is occurred. On the other hand, during the process of row scanning in the AMS-PDS algorithm, it accumulates pixel errors of consecutive pixels in a row. In our experimental work, we used different implementation techniques for these two situations, such that the nonuniform access problem of AMS-PDS becomes less severe. The second factor occurs in all PDS algorithms. The pipeline structure of modern CPUs improves greatly the performance of computing consecutive data. However, the pipeline flow is interrupted when cache misses or exceptions occur. For PDS, it suffers inherently from branch misprediction penalty; say, for example, it happens in Intel CPUs. Except for the "Table Tennis" sequence and the VOs, the SEA requires less

TABLE VII
EXECUTION TIME (SECONDS) PER FRAME OR PER VOP OF THE TESTED ALGORITHMS IN A SEARCH WINDOW WITH A SEARCH RANGE EQUAL TO 15 (i.e., $-15 \leq u, v \leq 15$)

Video Sequences	FSA		SEA		PDS		AMS-PDS		PG-PDS		CPME-PDS	
	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio
Football	1.92	1.00	0.95	2.02	0.96	2.00	0.97	1.98	1.12	1.71	0.93	2.06
Tabletennis	1.92	1.00	0.89	2.16	0.79	2.43	0.77	2.49	0.86	2.23	0.73	2.63
Stefan	1.92	1.00	0.83	2.31	0.84	2.29	0.80	2.40	0.95	2.02	0.78	2.46
Salesman	2.30	1.00	0.60	3.83	0.79	2.91	0.77	2.99	0.97	2.37	0.77	2.99
Foreman	0.58	1.00	0.11	5.27	0.19	3.05	0.18	3.22	0.23	2.52	0.17	3.41
Grand mother	0.58	1.00	0.09	6.44	0.19	3.05	0.18	3.22	0.24	2.42	0.18	3.22
Suzie	0.57	1.00	0.11	5.18	0.20	2.85	0.19	3.00	0.24	2.38	0.18	3.17
Trevor	0.58	1.00	0.08	7.25	0.16	3.63	0.16	3.63	0.21	2.76	0.16	3.63
Video Objects												
News	1.05	1.00	0.51	2.05	0.45	2.32			0.43	2.43	0.38	2.75
Children	0.60	1.00	0.43	1.39	0.27	2.21			0.20	2.99	0.18	3.32
Bream	0.81	1.00	0.53	1.52	0.36	2.24			0.34	2.37	0.29	2.78
Goldfish	1.51	1.00	0.83	1.82	0.74	2.04			0.61	2.48	0.54	2.80

TABLE VIII
EXECUTION TIME (SECONDS) PER FRAME OR PER VOP OF THE ROW-BASED ADAPTIVE PDS ALGORITHMS IN A SEARCH WINDOW WITH A SEARCH RANGE EQUAL TO 15 (i.e., $-15 \leq u, v \leq 15$)

Video Sequences	PG-PDS$_4$		PG-PDS$_8$		PG-PDS$_{16}$		CPME-PDS$_4$		CPME-PDS$_8$		CPME-PDS$_{16}$	
	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio	Execution time (second)	Speed-up ratio
Football	0.88	2.18	0.91	2.11	0.86	2.23	0.80	2.40	0.83	2.31	0.80	2.40
Tabletennis	0.61	3.15	0.64	3.00	0.62	3.10	0.61	3.15	0.64	3.00	0.62	3.10
Stefan	0.71	2.70	0.74	2.59	0.70	2.74	0.64	3.00	0.68	2.82	0.65	2.95
Salesman	0.66	3.48	0.69	3.33	0.67	3.43	0.60	3.83	0.62	3.71	0.60	3.83
Foreman	0.15	3.87	0.15	3.87	0.15	3.87	0.14	4.14	0.14	4.14	0.14	4.14
Grand mother	0.16	3.63	0.16	3.63	0.16	3.63	0.15	3.87	0.16	3.63	0.15	3.87
Suzie	0.16	3.56	0.17	3.35	0.17	3.35	0.15	3.80	0.16	3.56	0.16	3.56
Trevor	0.13	4.46	0.14	4.14	0.13	4.46	0.13	4.46	0.13	4.46	0.12	4.83
Video Objects												
News	0.37	2.83	0.39	2.68	0.38	2.75	0.35	2.99	0.36	2.90	0.35	2.99
Children	0.22	2.72	0.23	2.60	0.23	2.60	0.19	3.15	0.20	2.99	0.20	2.99
Bream	0.30	2.69	0.32	2.52	0.31	2.60	0.28	2.88	0.29	2.78	0.28	2.88
Goldfish	0.59	2.56	0.62	2.44	0.61	2.48	0.54	2.80	0.56	2.70	0.55	2.75

computational time as compared to that of the PDS, while the SEA actually requires more operations. Even though the presence of these two drawbacks, our experimental results show that the CPME-PDS gives the best performance for sequences containing high-motion activities and the video objects.

In addition to the approach using pixel-based sorting, we have also tested the row-based sorting approach for the adaptive PDS algorithms by making use of PG-PDS or clustering characteristics (CPME-PDS). This is regarded as a compromise between the sorting approach and the problem of nonuniform memory access. This approach is not valid for other algorithms in our discussion. Table VIII summarizes the performances in terms of the execution time per frame or per VOP. The number of consecutive pixels in a sorted row is indicated by the subscript r, such as PG-PDS$_r$ and CPME-PDS$_r$. Note that after this modification, the PG-PDS$_r$ is very closed to the approach of AMS-PDS. Comparing Table VIII with Table VII, it shows that the row-based sorting can efficiently improve both PG-PDS and CPME-PDS. It confirms that the approach using clustering characteristics is more effective than the pixel gradient for adaptive PDS technique. It gives better performance as compared to the gradient-based approach for all tested sequences. Table VII also indicates that the SEA can attain superior results for video sequences on conferencing. Table VIII shows that CPME-PDS$_4$ is able to achieve the best performance for the remaining sequences except the "Children," for which the pixel-based CPME-PDS gives the best computational time. On average, the SEA and CPME-PDS$_4$ provide 3.42 and 3.38 times speedup when comparing to the FSA. These experimental results confirm that CPME-PDS$_4$ is most suitable for motion estimation of sequences containing high-motion activities and arbitrarily shaped video objects.

V. Conclusion

We have proposed an adaptive partial distortion search algorithm entitled as the CPME-PDS. The algorithm makes use of the phenomenon that pixel matching errors in a MB with similar magnitude tend to appear together in a cluster in natural video sequences. We have demonstrated that this is a popular phenomenon for relatively large search windows. According to this phenomenon, we have found that both mean of pixel values in a target MB and mean of pixel values in a candidate MB are good references to predict the magnitude of each pixel matching error in the target MB. Hence, the mean of pixel values in the initial candidate MB at the centre of a search window has been used to calculate a reference value and to construct an adaptive index set. As a result, the pixel matching error with larger magnitude can be accumulated to the SAD_p sooner than others and the SAD calculation can be terminated at an early stage. We have evaluated the efficiency of the CPME-PDS in two measures, the total number of operations and the execution time per frame or per VOP in motion estimation.

In terms of the number of operations, our experimental results show that for a small maximum allowable search range, such as $D = 1$ or some cases of $D = 2$, the conventional PDS is still the best algorithm due to the overheads of fast algorithms. However, in a reasonably longer maximum allowable search range, $D = 2$ to $D = 15$, the computational efficiency of the CPME-PDS outperforms other algorithms for coding sequences with high-motion activities and arbitrarily shaped objects. In the case of a large search window $D = 15$, our experimental results show that the CPME-PDS can have a speedup of three to nine as compared with FSA, depending upon the contents of the coded video sequences. Hence, the proposed CPME-PDS is generally the best among all algorithms. The major advantages of CPME-PDS are its high efficiency and conceptual simplicity. Compared to other adaptive PDS, it requires less overheads.

When motion estimation time per frame or per VOP is used for evaluation, the performance of CPME-PDS is degraded slightly due to the problem of nonuniform memory access. Nevertheless, the CPME-PDS is still able to provide the best efficiency for sequences with high-motion activities and video object encoding. We have modified the CPME-PDS into a row-based algorithm in order to remedy the nonuniform memory access problem. For example, a row of four consecutive pixels with larger prediction errors is accumulated to the SAD_p sooner than other rows. Experimental results show that the conventional SEA provides a speedup of about 3.42 times when comparing to the FSA, and it performs the best for video conferencing sequences. Meanwhile, our row-based CPME-PDS, CPME-PDS_4 can speed up the search for about 3.38 times as compared to the FSA on average, and, furthermore, the CPME-PDS_4 outperforms all other tested algorithms (including the SEA) for encoding sequences with high-motion activities and arbitrarily shaped video objects.

References

[1] *Video Coding for Low Bit Rate Communication*, ITU-T Standard H.263, Mar. 1996.

[2] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video*, ISO/IEC Standard 13818-2, 1996.

[3] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. COM-29, no. 12, pp. 1799–1808, Dec. 1981.

[4] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 369–377, Aug. 1998.

[5] *MPEG-4 Optimization Model Version 1.0*, ISO/IEC JTC1/SC29/WG11, MPEG2000/N3324, Mar. 2000.

[6] M. Bierling, "Displacement estimation by hierarchical block matching," in *Proc. VCIP*, vol. 1001, 1988, pp. 942–951.

[7] K. M. Uz, M. Vetterli, and D. J. LeGall, "Interpolative multiresolution coding of advanced television with compatible subchannels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 86–99, Mar. 1991.

[8] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommunication Conf.*, New Orleans, LA, Nov. 1981, pp. G5.3.1–G5.3.5.

[9] B. Liu and A. Zaccarin, "New fast algorithm for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 2, pp. 148–157, Apr. 1993.

[10] Y. L. Chan and W. C. Siu, "New adaptive pixel decimation for block motion vector estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 1, pp. 113–118, Feb. 1996.

[11] ITU-T Recommendation H.263 Software Implementation, Digital Video Coding Group, Telenor R&D, 1995.

[12] S. Eckart and C. Fogg, "ISO/IEC MPEG-2 software video codec," *Proc. SPIE*, vol. 2419, pp. 100–118, 1995.

[13] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Process.*, vol. 4, no. 1, pp. 105–107, Jan. 1995.

[14] Y.-S. Chen, Y.-P. Hung, and C.-S. Fuh, "Fast block matching algorithm based on the winner-update strategy," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1212–1222, Aug. 2001.

[15] X. Q. Gao, C. J. Duanmu, C. R. Zou, and Z. Y. He, "Multi-level successive elimination algorithm for motion estimation in video coding," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 4, 1999, pp. 227–230.

[16] J.-N. Kim and T.-S. Choi, "A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, pp. 1040–1048, Oct. 2000.

[17] B. Tao and M. T. Orchard, "Gradient-based residual variance modeling and its applications to motion-compensated video coding," *IEEE Trans. Image Process.*, vol. 10, no. 1, pp. 24–35, January 2001.

[18] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 5, pp. 438–442, Aug. 1994.

[19] *Information Technology—Coding of Audio-Visual Objects: Visual*, ISO/IEC Standard 14496-2, 1998.

[20] P. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*. Norwell, MA: Kluwer, 1999.

[21] H. M. Mahmoud, *Sorting: A Distribution Theory*. New York: Wiley, 2000.

Ko-Cheung Hui received the B.Sc degree (first class honors) in engineering physics from The Hong Kong Polytechnic University and the M.Sc. degree in physics from Chinese University of Hong Kong in 1997 and 1999, respectively. He is currently pursuing the Ph.D. degree at the Department of Electronic Information Engineering, The Hong Kong Polytechnic University.

His research interests are in imaging, videos, and coding techniques.

Wan-Chi Siu received the Associateship from The Hong Kong Polytechnic University, the M.Phil. degree from The Chinese University of Hong Kong, and the Ph.D. degree from the Imperial College of Science, Technology, and Medicine, London, U.K., in 1975, 1977, and 1984, respectively.

He was with The Chinese University of Hong Kong as a Tutor, initially, and later as an Engineer, from 1975 and 1980. He then joined The Hong Kong Polytechnic University as a Lecturer in 1980 and was subsequently supported by the University in October 1982 for further studies in the U.K. He was promoted to Senior Lecturer, Principle Lecturer, and Reader in 1985, 1987, and 1990, respectively, and has been Chair Professor in the Department of Electronic and Information Engineering since 1992. He was Head of the Electronic and Information Engineering Department and subsequently Dean of Engineering Faculty from 1994 and 2002. He is currently the Director of the Centre for Multimedia Signal Processing. He has published over 250 research papers, over 120 of which appeared in international journals, including IEEE publications and *IEE Proceedings*. He is also an Editor of the recent book *Multimedia Information Retrieval and Management* (Berlin, Germany: Springer, 2003) and a member of the editorial board of the *Journal of VLSI Signal Processing Systems for Signal, Image, Video Technology*, the EURASIP *Journal on Applied Signal Processing*, and a few other journals. His research interests include digital signal processing, fast computational algorithms, transforms, wavelets, image and video coding, and computational aspects of pattern recognition and neural networks.

Prof. Siu was Guest Editor and Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: ANALOG AND DIGITAL SIGNAL PROCESSING from 1995 to 1997. He received many awards, such as the Distinguished Presenter Award of the University Management Workshop of the PolyU (1997), the IEEE Third Millennium Medal for Outstanding Achievements and Contributions (2000), and, most recently, the Best Teacher Award, EIE, and the Outstanding Award in Research and Scholarly Activities, Faculty of Engineering, PolyU (2003). He has been a keynote speaker at a number of international conferences, including, most recently, the 2002 Third IEEE Pacific Rim Conference on Multimedia (PCM2002), Taiwan, R.O.C., and the 2003 IEEE International Conference on Neural Networks and Signal Processing, Nanjing, China. He has held the position of General Chair or Technical Program Chair of many international conferences. In particular, he was a Technical Program Chair of the IEEE International Symposium on Circuits and Systems (ISCAS 1997), and he is the General Chair of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003). From 1991 and 1995, he was a member of the Physical Sciences and Engineering Panel of the Research Grants Council (RGC), Hong Kong Government, and, in 1994, he chaired the first Engineering and Information Technology Panel to assess the research quality of 19 cost centers (departments) from all universities in Hong Kong.

Yui-Lam Chan received the B.Eng. (first class honors) and Ph.D. degrees from The Hong Kong Polytechnic University in 1993 and 1997, respectively.

He joined The Hong Kong Polytechnic University in 1997 and is now an Assistant Professor with the Department of Electronic and Information Engineering and a member of the Centre for Multimedia Signal Processing. He has published over 30 research papers in various international journals and conferences. His research interests include multimedia technologies, signal processing, image and video compression, video transcoding, video streaming, video conferencing, and digital TV.

Dr. Chan was the recipient of more than ten famous prizes, scholarships, and fellowships for his outstanding academic achievements, such as the Champion of the Varsity Competition in Electronic Design in 1993 and a recipient of the Sir Edward Youde Memorial Fellowship and the Croucher Foundation Scholarships.