A Fully Adaptive Distance-Dependent Thresholding Search (FADTS) Algorithm for Performance-Management Motion Estimation

Golam Sorwar, *Member, IEEE*, Manzur Murshed, *Member, IEEE*, and Laurence S. Dooley, *Senior Member, IEEE*

Abstract—**Trading off computational complexity and quality is an important performance constraint for real time application of motion estimation algorithm. Previously, the novel concept of a distance-dependent thresholding search (DTS) was introduced for performance scalable motion estimation in video coding applications. This encompassed the full search as well as other fast searching techniques, such as the three-step search, with different threshold settings providing various quality-of-service levels in terms of processing speed and predicted image quality. The main drawback of the DTS was that the threshold values had to be manually defined. In this paper, the DTS algorithm has been extended to a fast and fully adaptive DTS (FADTS), a key feature of which is the automatic adaptation of the threshold using a desired target and the content from the actual video sequence, to achieve either a guaranteed level of quality or processing complexity. Experimental results confirm the performance of the FADTS algorithm in achieving this objective by demonstrating either comparable or improved search speed over existing fast algorithms including the diamond search, hexagon-based search, and enhanced hexagon-based search, while maintaining similar error performance.**

Index Terms—**Adaptive algorithms, block motion estimation, distance-dependent thresholding, scalable motion estimation, video coding.**

I. Introduction

MOTION estimation (ME) plays a vital role in video coding standards, such as MPEG-1/2 [1], [2] and H.261/3/4 [3]–[5], in exploiting temporal redundancy in video sequences. Most ME techniques use block matching algorithms (BMA) to compute motion vectors(MVs). The most straightforward method of obtaining a MV is to search all possible locations within a given search area. Since this method, known as full search (FS), uses an exhaustive search to locate the minimum block-distortion measure (BDM) for each candidate block, it provides optimal performance, but at the expense of very high computation. It is for this reason that FS is not used in real-time systems. Indeed ME is the major bottleneck in real-time video coding applications, hence the need for faster algorithms.

A number of fast block ME algorithms have been proposed to lower the computation complexity, for example, the 2-D logarithmic search (2DLOG) [6], the three-step search (TSS) [7], the new three-step search (NTSS) [8], the advanced center biased search [9], the four-step search (FSS) [10], the cross-search [11], the prediction search [12], the diamond search (DS) [13], and the hexagon-based search (HEXBS) [14], [15]. The DS algorithm has achieved a significant speed gain by considering diamond-shaped search patterns instead of the conventional square ones with a view to approximate the optimal (but unrealizable) circular shape as closely as possible. Recently, the HEXBS algorithm has surpassed the speed of DS by using a better approximation with hexagon-shaped search patterns. All of these fast algorithms assume that either the error surface is unimodal over the entire search area (i.e., there is only one global minimum) or the MV is center-biased. These assumptions essentially require that either the BDM increases monotonically as the search point moves away from the global minimum position or the MV exists in a small range. These assumptions are reasonable for certain applications e.g., in video-conferencing, where the motion is neither very fast nor complicated. However, they are generally invalid for many real video sequences because of the highly nonstationary characteristics of the video signal. Moreover, the search directions of these algorithms can be ambiguous, leading to the MV becoming entrapped in a local minimum with a resulting degradation in predictive performance.

Despite their respective differences, these fast search algorithms (as well as FS) all have one common feature that none of them has been designed to provide flexibility in controlling the performance in terms of predicted picture quality and processing time (speed). They do not allow any performance scalability in ME; they no facility to trade system parameters depending upon a particular application, or to preset a user-defined level of quality of service (QoS) in terms of predicted picture quality or computational complexity. Such a feature would be very advantageous in facilitating complexity management in video coding, especially in real-time software-only low bit rate video CODECs (Coder and Decoder) [16] or low-power video CODECs for mobile or hand-held computing platforms which particularly require a more flexible tradeoff between complexity and quality [17].

It has been observed that the distortion of an object in a video frame increases with its velocity as well as the zoom and pan factors of the camera. Thus, as the length of the MV grows, so does

Manuscript received March 3, 2006; revised August 16, 2006. This paper was recommended by Associate Editor J. Boyce.

G. Sorwar is with the School of Commerce and Management, Southern Cross University, Coffs Harbour NSW 2457, Australia (e-mail: Golam.Sorwar@scu.edu.au).

M. Murshed and L. S. Dooley are with the Gippsland School of Information Technology, Monash University, Churchill Vic 3842, Australia (e-mail: Manzur.Murshed@infotech.monash.edu.au; Laurence.Dooley@infotech.monash.edu.au).

Digital Object Identifier 10.1109/TCSVT.2006.888816

the distortion error. Based on this tenet, it can be concluded that locating a block with the minimum prediction error but with a MV of high magnitude, is not only ineffectual in the prevailing distorted search space, but will inevitably lead to many false MVs being erroneously selected. Designing a new BMA that seeks to exploit this feature can provide a variable threshold in the search process, which increases as the search expands outwards, will enable the user to restrict the search boundary so that it can be used as an effective control parameter for performance scalability and QoS in terms of predicted image quality as well as processing time in ME.

This paper addresses this matter by proposing a novel *fully adaptive distance-dependent thresholding search* (FADTS) algorithm by introducing the concept of distance-dependent thresholding search (DTS) for fast and performance management ME in video coding applications. The unique feature of this algorithm is that it can dynamically adjust the threshold to achieve any level of service required in terms of both quality and processing speed. This means for example, that a higher (lower) error or speed can be achieved by automatically adapting the threshold to a correspondingly level, depending on video content so providing the potential for performance scalable ME for real-time software only video coding application. A preliminary version of this work has been presented in [18] where threshold values are defined manually.

The paper is organized as follows. The original distance-dependent thresholding (DTS) algorithm along with its motivation is discussed in Section II. Section III details the novel fully adaptive DTS (FADTS) algorithm, including some enhancement to DTS. Computational complexity associated with FADTS is analysed in Section IV, while Section V includes both experimental results and analysis of the performance of FADTS for various levels of quality and speed. Section VI presents some conclusions.

II. Distance-Dependent Thresholding Search (DTS) Algorithm

In introduction, it was shown that many contributions in the domain of block-matching algorithms are based on the principle of reducing the checking points in a search window under unimodal error surface assumption. If a BDM is monotonic along any direction away from the optimal point, a well-designed fast algorithm can then be guaranteed to converge to the global optimal point. According to Chow and Liou [19], however, this assumption does not hold true for real world video sequences. Fig. 1 shows a typical mean absolute error (MAE) per pixel surface of *Football* sequence for a search window of ± 16 pixels, which has many local minima due to the nonstationary characteristics of the video signal. As a consequence, it is unlikely that conventional fast search algorithms, which use few directional candidates, would ever converge to the global minima. A nondirectional search, such as the FS algorithm, can always guarantee reaching the global minima on any kind of error surface at the at the expense of a large number of search points.

Definition 1 (Search Squares SS_τ): The search space with maximum displacement $\pm d$, centerd at pixel $p_{cx,cy}$, can be divided into $d+1$ mutually exclusive concentric search

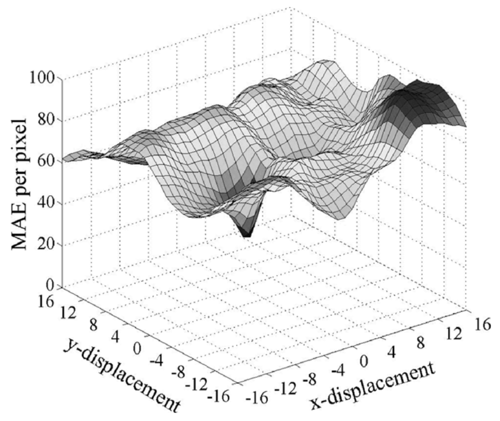


Fig. 1. MAE per pixel surface of *Football* sequence, current frame #35, reference frame #34, block coordinate (10,9), block size 16×16 pixels, maximum search displacement 16.

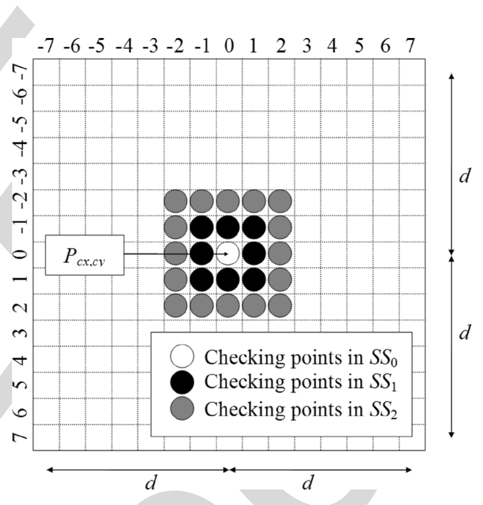


Fig. 2. DTS search squares SS_0, SS_1, and SS_2.

squares SS_τ, such that a checking point at pixel $p_{x,y}$, representing MV $(x-cx, y-cy)$, is in SS_τ if and only if $\max(|x-cx|, |y-cy|) = \tau$, for all $-d+cx \le x \le d+cx$, $-d+cy \le y \le d+cy$, and $\tau = 0, 1, \ldots, d$.

It can be readily verified that the number of checking points in search square SS_τ is

$$\begin{cases} 1, & \tau = 0; \\ 8\tau, & \tau = 1, 2, \ldots, d \end{cases} \tag{1}$$

and SS_τ represents the MVs of length in the range of $[\tau, \tau\sqrt{2}]$. The checking points used in the first three search squares are shown in Fig. 2.

Now consider the average MAE per pixel of a macroblock used as the BDM in the FS algorithm. For each macroblock, the FS algorithm looks for the minimum MAE per pixel value in the range of $[0, 2^b - 1]$ for a b-bit gray scale image. In [20] and [21], it was stated that the magnitude of a MV is proportional to the magnitude of the BDM. This observation has been explored further on a number of standard and nonstandard video sequences covering a wide range of object and camera motions. Cumulative probabilities of the minimum MAE per pixel for different search squares, using block size of 16×16 pixels, on the first 80

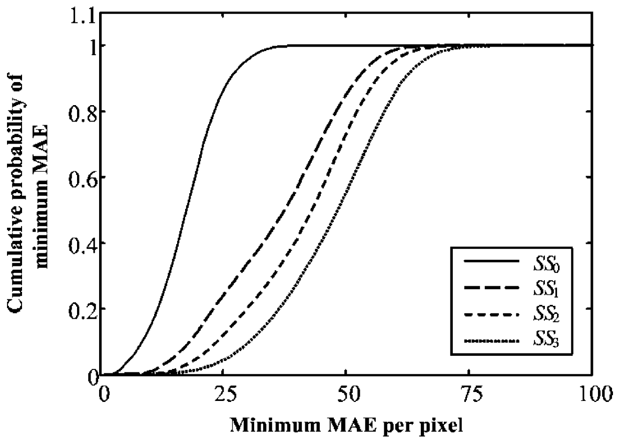


Fig. 3. Cumulative probabilities of the minimum MAE per pixel for the first four search squares on the first 80 frames of *Football* sequence using block size of 16 × 16 pixels.

frames of *Football* sequence are plotted in Fig. 3, which reveals the following:

- the cumulative probability of having a particular minimum MAE decreases as the MV length increases;
- the minimum MAE, in which the cumulative probability first reaches the value 1, increases as the MV length increases.

Both these findings were observed for all standard test sequences using three different block sizes 16×16, 8×8, and 4×4 pixels to postulate that the probability of terminating the FS algorithm at a higher MAE value increases with the length of the MV.

Based on these observations, the key finding is that the distortion of an object in a video frame increases with its velocity, as well as with the zoom and pan factors of the camera. As the length of the MV grows, so does the distortion error. It can be, therefore, concluded that locating a block with a minimum prediction error but with a MV of high length, is not only ineffectual in the prevailing distorted search space, but may lead to *false* MVs being erroneously selected.

A. Formal DTS Algorithm

Like all block-base ME search techniques, the DTS algorithm starts at the center of the search space. The search then progresses outwards by using search squares SS_τ in order while monitoring the current minimum MAE. A parametric thresholding function, $\mathrm{Threshold}(\tau, C)$, is used to determine the various thresholds to be used in the search involving each SS_τ where the parameter C is set at the start of each search and acts as a control parameter. After searching each SS_τ, the current minimum MAE is compared against the threshold value of that specific search square and the search is terminated if this MAE value is not higher than that threshold value. The DTS algorithm is formally presented in Fig. 4 where $\mathrm{MAE}_{(cx,cy)}(u,v)$ denotes the MAE per pixel of the macroblock centerd at pixel $p_{cx,cy}$ in the current frame with respect to the block centerd at pixel $p_{cx+u,cy+v}$ in the reference frame.

```
Algorithm DTS(C)
Precondition: Pixel $p_{cx,cy}$ is the centre of the search
space with maximum displacement $d$.
Postcondition: $MV$ contains the motion vector and
$MAE_{\min}$ contains the residual of the respective block.
Initialisation:
1.   $MAE_{\min} = MAE_{(cx,cy)}(0,0)$;
2.   $MV = (0,0)$;
Body:
3.   IF $MAE_{\min} > 0$ THEN
4.       FOR $\tau = 1,2,\ldots,d$
5.           FOR each checking point $p_{x,y}$ in $SS_\tau$
6.               $\varepsilon = MAE_{(cx,cy)}(x-cx, y-cy)$;
7.               IF $\varepsilon < MAE_{\min}$ THEN
8.                   $MAE_{\min} = \varepsilon$;
9.                   $MV = (x-cx, y-cy)$;
10.              ENDIF
11.          ENDFOR
12.          IF $MAE_{\min} \le Theshold(\tau, C)$ THEN
13.              RETURN;
14.          ENDIF
15.      ENDFOR
16.  ENDIF
```

Fig. 4. DTS algorithm.

B. Characteristics of the Thresholding Function

To make sure that the DTS algorithm can be transformed to an exhaustive FS algorithm, the threshold value for SS_0 is always assumed to be 0. As the maximum MAE value using a b-bit gray level intensity is $2^b - 1$, threshold values for all other search squares can, at most be $2^b - 1$. However, to ensure the algorithm includes the entire search space, all but the outermost threshold value must be less than $2^b - 1$. Moreover, to make the thresholding function distance-dependent, the function must monotonically increase. The DTS algorithm, therefore, assumes the following general properties of the thresholding function:

$$\left.\begin{array}{c} \mathrm{Threshold}\,(0,C) = 0 \\ \mathrm{Threshold}\,(1,C) \le \cdots \le \mathrm{Threshold}\,(d,C) \\ \mathrm{Threshold}\,(\tau,C) < 2^b - 1 \text{ for all } \tau = 1,2,\ldots,d-1 \\ \mathrm{Threshold}\,(d,C) \le 2^b - 1 \end{array}\right\}. \tag{2}$$

Parameter C plays a significant role in the DTS algorithm by allowing users to define different sets of monotonically increasing threshold values based on specific values of C. Obviously a set of larger threshold values terminates a search earlier than a set of smaller values. C therefore, provides a control mechanism to allow trading off between the computational complexity in terms of search points and prediction image quality.

The monotonic increasing function requirement means the DTS algorithm could use a linear, exponential, or any other complex analytic function to control the threshold with τ . In [18], the authors empirically observed linear thresholding function within the DTS algorithm outperforming and providing a wider range of flexibility compared to exponential thresholding function. This paper, therefore, has considered only linear thresholding function, which is defined as follows:

$$\mathrm{Threshold}\,(\tau, C_L) = C_L \times \tau, \text{ for all } \tau = 0,1,\ldots,d. \tag{3}$$

The subscript L in C_L specifies linear thresholding. It can be verified that the above definition satisfies all the conditions in (2) if $C_L \geq 0$ and $C_L \times d \leq 2^b - 1$. So, parameter C_L can take any value from the range given below

$$0 \leq C_L \leq \frac{(2^b - 1)}{d}. \tag{4}$$

C. Selecting the Thresholding Control Parameter

The choice of C_L involves a tradeoff between the quality of the ME and the computational complexity. When $C_L = 0$ in (3), the search terminating threshold value of any search square is zero. In this case, the DTS algorithm translates into the exhaustive FS algorithm as there is no threshold to terminate the search until all possible locations in the search space have been visited. Note that the search can still terminate earlier as soon as a search point with MSE per pixel of zero is reached. Had the search continued with the remaining search points, however, the resultant MV would remain the same as between two vectors with the same distortion, the shorter one is preferred. Conversely, when C_L is set to the maximum value, for an 8-bit gray level image and $d = 7$ the maximum integer value of C_L is 36, the DTS algorithm will be as fast as the probability of getting the minimum BDM within the search terminating threshold limit is high, especially around the search center. In the case of low motion video sequences, such as *Salesman* where MV distribution is center-biased, a high C_L performs well with low computational overhead. Conversely, for high motion video sequences like *Football* and *Flower Garden* where MV distribution is not center-biased, a high C_L may stop the search with an inaccurate MV and generate a high prediction error.

Finally, while linear thresholding means that different levels of QoS can be achieved by trading off between predicted image quality and computational complexity, in terms of search points, automatically selecting the best value of C_L is a challenging problem that will be addressed in next section.

III. Fully Adaptive DTS (FADTS) Algorithm

In performance-management ME, given a target prediction image quality in terms of average mean squared error (MSE) per pixel, the motion search algorithm tries to achieve it using as few search checking points as possible. Inversely, if a target processing speed is set in terms of average number of search point (SP) used per MV, the algorithm tries to achieve with as low MSE as possible. Performance-management ME also assumes real time constraint, which allows very limited number of passes per macroblock. Without such a constraint, trivial trial and error technique with a very high number of passes would suffice the adaptation. Without any loss of generality, this paper assumes the strictest constraint where only one ME pass is performed per macroblock. To leverage the adaptation technique, the original DTS algorithm is enhanced further.

A. Enhancing the DTS Algorithm

The concept of DTS is not linked to any specific search pattern shape. In the wake of improved speed gain by nonsquare

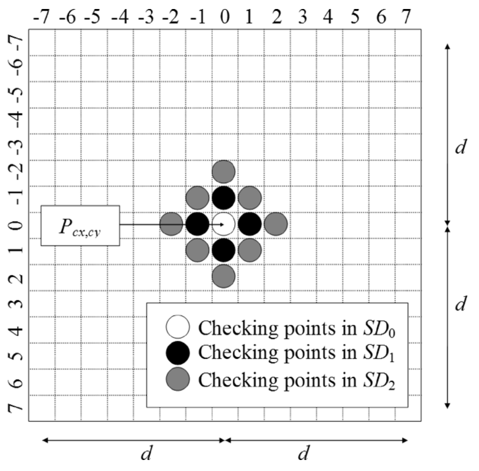


Fig. 5. DTS search diamonds SD_0, SD_1, and SD_2.

search patterns, DTS has been implemented using search diamonds SD_τ as shown in Fig. 5 where the number of checking points in SD_τ is

$$\begin{cases} 1, & \tau = 0; \\ 4\tau, & \tau = 1, 2, \ldots, 2d \end{cases} \tag{5}$$

and SD_τ represents the MV of length in the range of $[\tau/\sqrt{2}, \tau]$. Note that for $\tau = d+1, d+2 \ldots, 2d$, some of the checking points in the search diamond fall outside the search windows that are obviously ignored. Using fewer checking points for center-biased as well as horizontal and vertical MVs (prevalent in panning) makes DTS with search diamond superior to using search squares as observed with all the standard test sequences.

Well-established spatio-temporal motion correlation among the neighboring macroblocks [22]–[24] can be exploited to reduce the search point even further by using a predicted search origin rather than always using the center of the search window. Assuming row-major processing order, the search origin of macroblock at rth block row and cth block column is calculated from the mean of the MVs of already processed neighboring macroblocks at $(r-1)$th block row and $(c-1)$th block column, $(r-1)$th block row and cth block column, $(r-1)$th block row and $(c+1)$th block column, and rth block row and $(c-1)$th block column. If the magnitude of the difference between this mean vector with each of the four neighboring MV is within a predefined threshold T_{pred}, the search origin at the center is moved by that mean vector. DTS using predicted search origin has performed superior to the original DTS for all the standard test sequences. Experimental results have also confirmed that the value of T_{pred} is not very sensitive to performance, especially for prediction error. Using the threshold in the range from 3 to 7, the average MSE and the average number of search points of the first 50 frames of *Football* and *Flower Garden* sequences varied less than 1% and 5%, respectively, so to ensure average performance $T_{\mathrm{pred}} = 5$ is defined in experiments.

B. FADTS Closed-Loop Adaptation Model

The DTS algorithm works sequentially on frames of an input video sequence. Although consecutive frames are considered to be highly correlated, the input video signal can be considered

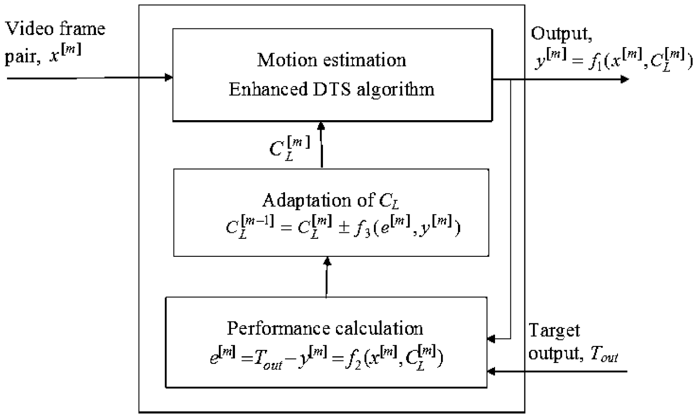

Video frame pair, $x^{[m]}$
Motion estimation
Enhanced DTS algorithm
Output, $y^{[m]} = f_1(x^{[m]}, C_L^{[m]})$
$C_L^{[m]}$
Adaptation of C_L
$C_L^{[m+1]} = C_L^{[m]} \pm f_3(e^{[m]}, y^{[m]})$
Performance calculation
$e^{[m]} = T_{out} - y^{[m]} = f_2(x^{[m]}, C_L^{[m]})$
Target output, T_{out}

Fig. 6. Closed-loop adaptation process for the FADTS algorithm.

time variable or nonstationary from the adaptation point of view. Therefore, a closed-loop adaptation model is presented for the FADTS algorithm, as shown in Fig. 6. The model has the following three modules.

- *ME*: This module calculates MVs using the enhanced DTS algorithm. The input of the module at iteration m are the video frame pair $x^{[m]}$ and the control parameter $C_L^{[m]}$. The output of the model can be either prediction image quality in terms of average MSE or speed in terms of average number of SP as selected by the user. The output at iteration m can be expressed as

$$y^{[m]} = f_1(x^{[m]}, C_L^{[m]}) \tag{6}$$

where f_1 is a monotonically increasing or decreasing function of C_L (under stationary x), if the output is MSE or SP, respectively.

- *Performance calculation*: This module calculates the performance of the adaptive system by calculating the error signal as

$$e^{[m]} = T_{\text{out}} - y^{[m]} = f_2(x^{[m]}, C_L^{[m]}) \tag{7}$$

at each iteration m where

$$f_2(x^{[m]}, C_L^{[m]}) = T_{\text{out}} - f_1(x^{[m]}, C_L^{[m]}). \tag{8}$$

The value of e must be minimized as the adaptation process progresses.

- *Adaptation of C_L*: This module updates the value of C_L for the next iteration $m+1$ as

$$C_L^{[m+1]} = C_L^{[m]} + f_3(e^{[m]}, y^{[m]}) \tag{9a}$$

if the output is MSE or as

$$C_L^{[m+1]} = C_L^{[m]} - f_3(e^{[m]}, y^{[m]}) \tag{9b}$$

if the output is SP where f_3 can be any linear or nonlinear function.

The performance of an adaptive system largely depends on how the function $f_3(e, y)$ is defined. A few gradient search algorithms [25]–[29] exist that can adapt a system in searching for the optimal parameter to minimize error signal in (7). Among these, the *least mean square* (LMS) is the most well-known and popular method for its computational simplicity, robustness, and relatively easy implementation for online estimation of time-varying system parameters. A number of variants on the LMS theme have been conceived in order to ratify potential problems of the original LMS algorithm such as the need to guess the best value of step size, slow convergence, and numerical instability. The normalized block LMS (NBLMS) [29] is considered as the best option for automatically adjusting the control parameter C_L in order to achieve a target average MSE or average SP while coding a video sequence, where this sequence can be considered as a time varying nonstationary input to the adaptation system. Based on NBLMS, the threshold control parameter is updated as

$$C_L^{[m+K]} = C_L^{[m]} + \mu e^{[m]} \frac{\frac{1}{K}\sum_{i=0}^{K-1} y^{[m+i]}}{E_y} \tag{10a}$$

if the output is average MSE or as

$$C_L^{[m+K]} = C_L^{[m]} - \mu e^{[m]} \frac{\frac{1}{K}\sum_{i=0}^{K-1} y^{[m+i]}}{E_y} \tag{10b}$$

if the output is the average SP where

$$E_y = \sum_{i=0}^{K-1} \left(y^{[m+i]}\right)^2. \tag{11}$$

C. Formal FADTS Algorithm

The FADTS algorithm utilizing the NBLMS algorithm for adapting the control parameter C_L, in order to achieve a target predicted image quality is now outlined as follows. The algorithm applies the enhanced DTS algorithm on a block of K-frames of the video sequence using the same C_L value for ME. The C_L is initialized to an initial value $C_{L_{\text{init}}}$ for the first block of K-frames and the value of C_L is then updated for the next block of K-frames by (10a) using the average output MSE and the total energy of all output MSE of the ME carried out so far on the current block of K-frames. Fig. 7 presents the complete FADTS algorithm.

The flexibility of the FADTS algorithm is illustrated by the fact that it is capable of adapting the ME in order to achieve a target prediction image quality in terms of the average MSE output by trading off search speed in terms of average number of search points. However, the same algorithm can easily be transformed for adapting the ME with a target search speed in terms of average SP while trading off prediction image quality by incorporating the following minimal changes:

- the number of average SP is used as the target T_{out};
- the output $y^{[m]}$ is the average SP for ME between iterations m and $m+1$;
- the updating factor in the adaptation of C_L is negative as in (10b) instead of positive as in (10a).

Based on the aforementioned adaptive model, the performance of the adaptive algorithms depends on the appropriate selection of the initial threshold constant $C_{L_{\text{init}}}$, the step size μ, and block length K. Each of these is now briefly explored.

```
Algorithm FADTS($T_{out}$)
Precondition: Motion is estimated for a video sequence of $N$ frames with target output $T_{out}$ in MSE, block length $K$, initial threshold constant $C_{L_{init}}$, and normalised step size $\mu$.
Postcondition: Motion vectors with as few SP as possible.
Initialisation:
1.  $C_L^{[1]} = C_L^{[2]} = \cdots = C_L^{[K]} = C_{L_{init}}$;
Body:
2.  FOR $m = 1, 1+K, \cdots, 1+(\lfloor (N-1)/K \rfloor)K$
        • K-Block Motion estimations:
3.      $S = V = 0$;
4.      FOR $i = 0, 1, \cdots, \min(K-1, N-m-1)$
5.          Calculate MV between frame pair $x^{[m+i]}$, consisting
            of frames $m+i$ and $m+i+1$, using DTS($C_L^{[m+i]}$)
            algorithm with the enhancements in Section III.A.
            Let $y^{[m+i]}$ be the average MSE of this MV
            estimation.
6.          $S = S + y^{[m+i]}$;
7.          $V = V + (y^{[m+i]})^2$;
8.      ENDFOR
        • Performance measurement:
9.      $e^{[m]} = T_{out} - \frac{S}{K}$;
        • Adaptation of $C_L$:
10.     $C_L^{[m+K]} = \cdots = C_L^{[m+K+\min(K-1,N-m-1)]} = C_L^{[m]} + \mu e^{[m]} \frac{S}{KV}$;
11. ENDFOR
```

Fig. 7. FADTS algorithm.

D. Initialization of C_L

Generally, adaptive algorithms start by setting the initial weight vector (in this case, the value of C_L) to zero. Although in the case of a large number of iteration cycles its impact may be negligible, the performance of adaptive algorithms with relatively fewer iteration cycles depends heavily on the initial value of its weight. Once shot detection is incorporated in the FADTS algorithm, the number of iterations based on an initialization of C_L depends on the number of frames in each shot, which again depends on the visual content and editorial decisions. However, after studying a large number of standard and nonstandard video sequences, it can be fairly concluded that the average number of frames in a shot is not large enough to consider it as nullifying the impact of initializing C_L to zero. Thus, the choice of the initial value of C_L impacts significantly on the performance of the FADTS algorithm for ME. Based on empirical data, the initial value of C_L, i.e., $C_{L_{\text{int}}}$, has been determined for quality and speed in the following two sections.

1) C_L Initialization for Quality Adaptation: From (4), for an 8-bit gray level image and $d = 7$, the maximum integer value of C_L is 36. Experimental results in Fig. 8 on standard test sequences have revealed that above a certain limit, $C_L > 25$, the speed variation was insignificant, so that the upper limit of the threshold control parameter $C_{L_{\max}}$ can be defined as 25 instead of 36. Similarly, though the minimum value of $C_L = 0$ (FS case) in the DTS algorithm, experimental results have also showed $C_L \leq 2$ provided almost the same prediction quality as the FS algorithm. Therefore, the lower limit of the threshold control parameter $C_{L_{\min}} = 2$ is defined.

The prediction error (quality) variation in terms of the average MSE per pixel using different values of C_L is significant for all

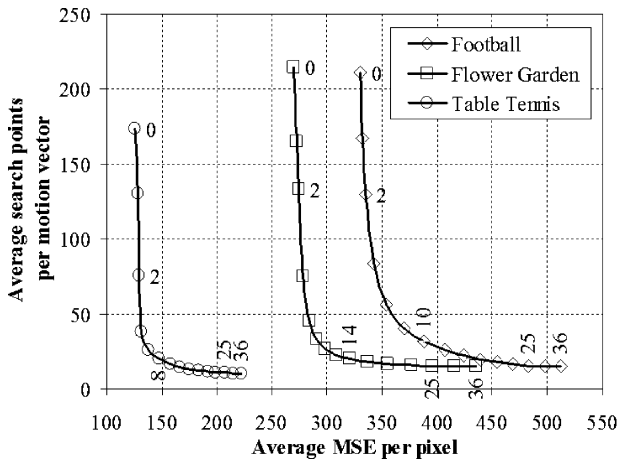


Fig. 8. Quality-speed performance of DTS for different standard video sequences with threshold control parameter C_L settings 0, 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 30, and 36.

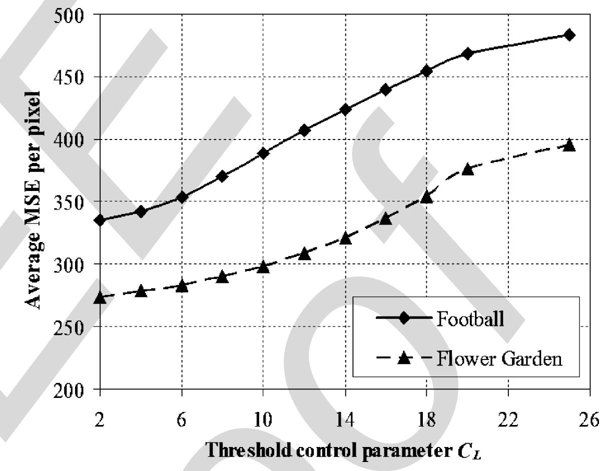


Fig. 9. Average MSE per pixel on the first 80 frames of *Football* and *Flower Garden* sequences for different values of C_L.

motion sequences as evident in Fig. 9. It is also shown that although prediction error variation with different values of C_L is not exactly linear, it can be approximated as so. Based on this premise, $C_{L_{\text{init}}}$ for a particular sequence is automatically computed from information in the first few frames of the sequence as follows.

- Compute the minimum prediction error $\text{MSE}_{C_{L_{\min}}}$ between frames #1 and #2 of the current scene using $C_{L_{\min}}$.
- Compute the maximum prediction error $\text{MSE}_{C_{L_{\max}}}$ between frames #2 and #3 of the current scene using $C_{L_{\max}}$.
- Compute $C_{L_{\text{init}}}$ for the next K-frames of that scene as

$$C_{L_{\text{init}}} = \left(\frac{T_{\text{out}} - \text{MSE}_{C_{L\,\min}}}{\text{MSE}_{C_{L_{\max}}} - \text{MSE}_{C_{L_{\min}}}} \times (C_{L_{\max}} - C_{L_{\min}}) + C_{L_{\min}} \right) \quad (12)$$

where T_{out} is the target prediction quality (in this instance, the average MSE).

2) C_L Initialization for Search Point Adaptation: Fig. 10 shows the computational cost in terms of the average number

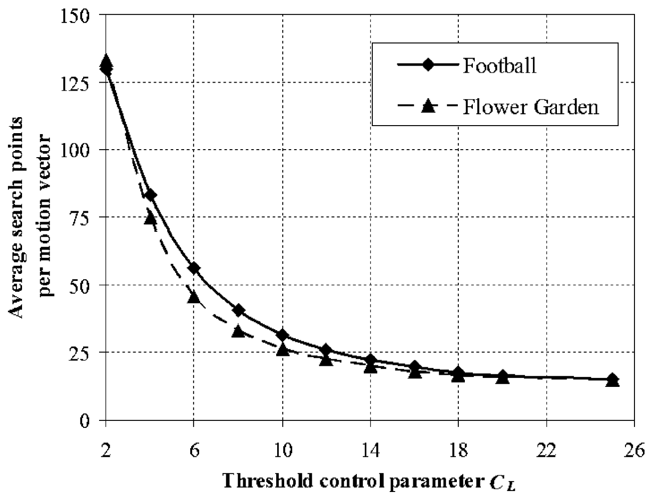


Fig. 10. Average search points per MV on the first 80 frames of *Football* and *Flower Garden* sequences for different values of C_L.

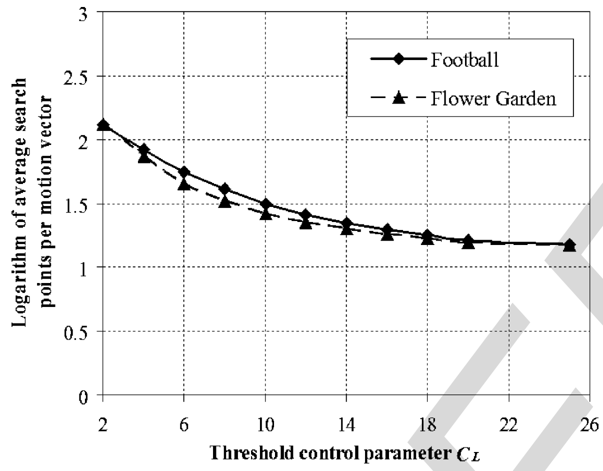


Fig. 11. Logarithm of average search points per MV on the first 80 frames of *Football* and *Flower Garden* sequences for different values of C_L.

of search points per MV for some standard high motion video sequences with different values of C_L. If a logarithmic scale is used for search points, the characteristic curve can be converted into a linear approximation as shown in Fig. 11. Using the same procedure described in the previous section, $C_{L_{\text{init}}}$ can be calculated as

$$C_{L_{\text{init}}} = \left(\frac{\log \mathrm{SP}_{C_{L_{\min}}} - T_{\text{out}}}{\log \mathrm{SP}_{C_{L_{\min}}} - \log \mathrm{SP}_{C_{L_{\max}}}} \times (C_{L_{\max}} - C_{L_{\min}}) + C_{L_{\min}} \right) \tag{13}$$

where $\mathrm{SP}_{C_{L_{\max}}}$ and $\mathrm{SP}_{C_{L_{\min}}}$ are the maximum and minimum search points obtained for $C_{L_{\max}}$ and $C_{L_{\min}}$, respectively, and T_{out} is the target speed (in this instance, the average SP).

E. Block Length K

Table I shows the performance of the FADTS algorithm with different values of K for *Football*, *Flower Garden*, and *Salesman* sequences with average 390, 280, and 16 MSE as the target errors, respectively. It can be observed that the FADTS

TABLE I
PERFORMANCE COMPARISON OF FADTS AALGORITHM WITH DIFFERENT VALUES OF K

K	*Football*		*Flower Garden*		*Salesman*	
	MSE	SP	MSE	SP	MSE	SP
2	392.0	37.5	280.99	46.71	15.42	9.37
4	391.0	37.8	281.11	44.32	15.42	9.29
6	392.8	39.1	282.86	43.87	15.42	9.32
8	391.1	40.6	283.51	43.50	15.42	9.31
10	392.2	41.7	283.89	44.32	15.42	9.31

algorithm has obtained an output average MSE closer to the target MSE with a comparatively fewer number of search points, when the block length $K = 4$ for all cases. Although a lower value of K also provided similar performance in satisfying the targets, according to (10), it increases the overhead computational cost for the adaptation process. Conversely, a higher value of K can be considered in order to reduce the overhead cost, though the block length of K in the NBLMS algorithm cannot be too high if it is assumed that the content of a video sequence may be unstable. Based on this assumption and the experimental results, a value of $K = 4$ was defined for all experiments.

F. Step Size μ

With regard to this parameter, Meghriche *et al.* [30] highlighted that there is no universal solution for finding the optimal value of μ. In [29], the NLMS algorithm considers a step size range of $(0 < \mu < 2)$ for signal processing applications. The lower the value of μ, the slower the convergence rate; while a high step size can lead to system instability. The impact of μ is not constant as the variable step size also depends on the error signal e. If e becomes large, then a greater step size is considered for the next iteration to speedily move towards the target level, while if it is low, the step size will be smaller in order to follow the target line. Moreover, the ceiling $C_{L_{\max}} \leq 25$ enforces a dampening effect, which avoids any instability even when $\mu \geq 2$ is chosen. The value of $\mu = 2$ was defined for all the various standard video sequences tested, with no instability encountered for the FADTS algorithm.

IV. COMPUTATIONAL COMPLEXITY ANALYSIS OF FADTS

Consider a ME system with the following parameters: frame size $= N_h \times N_v$ pixels, macroblock size $= M \times M$ pixels, maximum MV displacement $= \pm d$, and frame rate $= f$ fps. If there are Γ number of operations required for the BDM calculation of one search checking point, then the FS algorithm requires a maximum $\Gamma(2d+1)^2\varsigma$ operations per second using integer-pel accuracy where $\varsigma = N_h N_v f / M^2$ is the total number of macroblocks processed per second. The DTS algorithm requires extra d operations to compare the current minimum BDM with the predefined threshold of each search square, for each macrboblock, while searching the entire search window. The total number of extra operations required per second is thus $d\varsigma$ so the upper computational bound of the DTS algorithm is

$$\psi_{\max} = \Gamma(2d+1)^2\varsigma + d\varsigma = (\Gamma(2d+1)^2 + d)\varsigma \tag{14}$$

operations per second.

Conversely, by using a very high threshold value, when only the corresponding center of the search space is checked and only one operation is required to compare the BDM found at the search center with a predefined threshold for each macroblock. So, the upper computational bound of the DTS algorithm is

$$\psi_{\min} = \Gamma\varsigma + \varsigma = (\Gamma + 1)\varsigma \tag{15}$$

operations per second.

When half-pel accuracy is used for ME, eight neighboring half-pel positions around the current minimum point obtained with integer-pel accuracy are checked. In this case, the upper and lower bounds of computational complexity of the DTS algorithm are increased by $8\Gamma\varsigma$ operations per second. While enhancement of DTS due to diamond shape patterns does not alter the value of $\psi_{\max}$ and $\psi_{\min}$, enhancement due to prediction uses an extra $R = O(4)$ operations per macroblock, i.e., an extra $R\varsigma$ operations per second.

Now consider the FADTS algorithm in Fig. 7. The outer loop in step 2 and the inner loop in step 4 iterates for N/K and K times, respectively. If all basic arithmetic operations and assignment are considered equivalent in terms of processing time, steps 1, 3, 6, 7, 9, and 10 take K, 2, 2, 3, 3, and $(K+5)$ operations, respectively. So, for processing a video of N-frames, the FADTS algorithm makes $K + N(6K+10)/K$ operations, i.e.,

$$\psi_{\text{extra}} = K + f\frac{6K+10}{K} = \left(6 + \frac{10}{K}\right)f + K \tag{16}$$

operations per second, in addition to the enhanced DTS algorithm.

Now consider performance-management ME for a typical CIF video sequence with $N_h = 352$, $N_v = 240$, $M = 16$, $d = 7$, $f = 30$, and $\Gamma = 3M^2 = 768$. Assuming the BDM is measured using MAE, since $K = 4$ is defined for the FADTS algorithm, $\psi_{\max}$, $\psi_{\min}$, and ψ_{extra} can be estimated using (14)–(16) as 1.22 billion, 5.45 million, and 259 operations per second, respectively. The gap of ψ_{extra} with $\psi_{\max}$ and $\psi_{\min}$ will widen even further when half-pel accuracy and prediction enhancement are considered.

In summary, therefore, the FADTS algorithm consumes negligible computational overhead compared to the BDM calculation in DTS algorithm, while providing significant performance benefits including user-definability of key parameters by employing an adaptive thresholding process.

V. Experimental Analysis of the FADTS Algorithm

Although FADTS is the first proposed performance-management adaptive ME search algorithm, its performance is compared against the existing fast algorithms in terms of quality-speed measurement where quality and speed are measured as MSE per pixel and SP per MV, respectively. While the TSS and NTSS algorithms are well established, the principles of more recent DS and HEXBS algorithms are now briefly described. The DS algorithm uses diamond shape patterns of nine and five search checking points as shown in Fig. 12. The search starts at the origin with the larger diamond pattern and in successive steps moves the origin to the point with the lowest BDM until

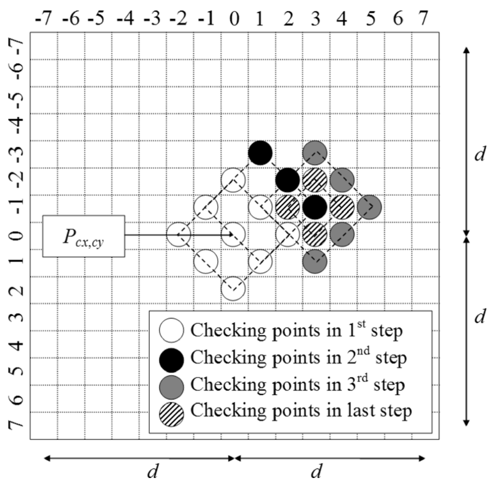

-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
$P_{cx,cy}$
d
d
d
d
Checking points in 1st step
Checking points in 2nd step
Checking points in 3rd step
Checking points in last step

Fig. 12. DS search steps.

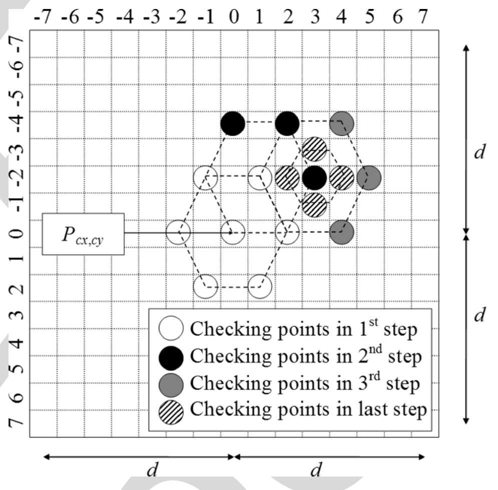

-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7
$P_{cx,cy}$
d
d
d
d
Checking points in 1st step
Checking points in 2nd step
Checking points in 3rd step
Checking points in last step

Fig. 13. HEXBS search steps.

the origin cannot be moved when the smaller diamond is used to complete the search. This algorithm uses nine and four new search points at the start and end and either three or five new search points at the intermediate steps depending on the direction of the move. The HEXBS algorithm has enhanced the DS algorithm by using hexagonal patterns of seven and five search checking points as shown in Fig. 13. A considerable speed gain is achieved as it uses seven and four new search points at the start and end, respectively, but only three search points in the intermediate steps irrespective of the direction of the move. The number of search points at the end is further reduced in the enhanced HEXBS (EHEXBS) [15] algorithm by grouping the six points comprising the last hexagon into six groups (pairs) and then using just two or three of the eight unsearched points having the minimum average BDM which are nearest to the group.

The experimental setup is as follows. For all search algorithms, MAE as the BDM, block size of 16×16 pixels, and maximum search displacement of $\pm$ 7 pixels were used. Although six standard video sequences *Football* (320×240 pixels, 345 frames), *Flower Garden* (352×240 pixels, 150 frames), *Salesman* (352×288 pixels, 150 frames), *Miss America* (176×144 pixels, 150 frames), *Tennis* (352×240 pixels,

TABLE II
AVERAGE MSE PER PIXEL AND SP PER MV OF THE FS, TSS, NTSS, DS, HEXBS, AND EHEXBS ALGORITHMS FOR *FOOTBALL* AND *FLOWER* GARDEN VIDEO SEQUENCES

BMA	*Football*			*Flower Garden*		
	MSE	PSNR [dB]	SP	MSE	PSNR [dB]	SP
FS	218.9	24.7	160.1	208.9	24.9	209.7
TSS	240.8	24.3	25.6	243.0	24.3	31.2
NTSS	239.2	24.3	26.9	213.3	24.8	29.0
DS	237.0	24.4	24.9	219.7	24.7	22.8
HEXBS	241.0	24.3	21.0	226.2	24.6	20.2
EHEXBS	246.7	24.2	19.9	229.2	24.5	18.6

150 frames), and *Foreman* (176 × 144 pixels, 298 frames) were used and FADTS performed well for all of them, results of the two most challenging sequences, *Football* and *Flower Garden* with heavy object motion and camera panning, respectively, in relation to performance-management adaptation are presented in this section. ME has been carried out on the luminance (Y) component values only with each pixel representing an 8-bit grayscale intensity.

To isolate improvement due to motion search technique only, ME was carried out differently than is done for video coding so that any influence of R-D optimization [31] and error propagation can be avoided. For each pair of successive frames, motion was estimated for the second frame using the original version of the first frame (not the motion compensated version of that frame as is used for video coding) as the reference and MSE per pixel was averaged using the first frame and the motion compensated second frame. As no entropy coding was used to compress the residual, this MSE measure was higher than what could be achieved by a video coder with residual encoding. However, this MSE measure correlates highly with residual compression and thus still represents quality of the image, if R-D tradeoff is factored in. The values of $K = 4$ and $\mu = 2$ were used with the FADTS algorithm where threshold control parameter was initialized accordingly using (12) or (13). All the search algorithms were enhanced by 1) predicting the search origin using the spatio-temporal motion correlation among neighboring blocks $T_{\text{pred}} = 5$ as explained in Section III-A and 2) refining MVs with half-pel accuracy using additional eight neighboring half-pel search points (with interpolated intensity values) around the current minimum point obtained with integer-pel accuracy.

Average MSE per pixel values and average search point numbers per MV for different nonadaptive algorithms are summarized in Table II. While FS achieves the maximum quality with the minimum average MSE per pixel for each sequence, the speed gain of DS and HEXBS over TSS is clearly evident.

The performance of the FADTS algorithm was tested and evaluated for quality and speed adaptation as follows.

A. Quality Adaptation

The performance of the FADTS algorithm for quality adaptation is presented in Tables III and IV for a number of different MSE per pixel target values for *Football* and *Flower Garden* sequences, respectively. It can be seen that the FADTS algorithm achieved all targets within ± 1% of disgreement. For example, targets were set to estimate motion with an average MSE of 230 per pixel or image quality of 24.51 dB peak signal-to-noise raio (PSNR) for *Football* sequence and on average MSE of 215 per pixel or image quality of 24.81 dB PSNR for *Flower Garden* sequence. The FADTS algorithm satisfied these demands with MSE of 232.04 and 214.41 per pixel using on average 49.18 and 24.54 search points, respectively. The disagreement of these two cases are $(232.04 - 230)/230 = 0.89\%$ and $(214.41 - 215)/215 = -0.27\%$, respectively. These settings reveal that the FADTS algorithm is able to reach any bounded target level of quality, with the implicit assumption that the minimum target error obtained by FS is the lower bound.

TABLE III
QUALITY ADAPTATION FOR *FOOTBALL* VIDEO SEQUENCE

Target Quality		Actual Quality		Actual
MSE	PSNR [dB]	MSE	PSNR [dB]	SP
230	24.51	232.04	24.48	49.18
235	24.42	234.70	24.43	32.25
240	24.32	241.00	24.31	19.87
250	24.15	252.13	24.11	16.56

TABLE IV
PREDICTION ERROR ADAPTATION FOR *FLOWER* GARDEN VIDEO SEQUENCE

Target Quality		Actual Quality		Actual
MSE	PSNR [dB]	MSE	PSNR [dB]	SP
210	24.91	212.80	24.85	34.95
215	24.81	214.41	24.82	24.58
220	24.71	218.62	24.73	16.65
225	24.61	222.79	24.65	15.21

Adaptive values of threshold control parameter C_L, for different frames of *Football* and *Flower Garden* video sequences with target set as MSE of 240 and 215 per pixel, respectively are plotted in Fig. 14. The adaptive nature of the FADTS algorithm is clearly evident for varying content between different frames. It also indicates that the FADTS algorithm automatically computed a starting value for C_L, 14 for *Football* and 7 for *Flower Garden*, based on both the content of the video sequence and the desired target.

B. Search Point Adaptation

The performance of the FADTS algorithm for computational scalability in terms of the average number of search points per MV was tested with a number of targets. Tables V and VI show some of these targets and the actual values obtained by the FADTS algorithm for *Football* and *Flower Garden* video sequences, respectively. Again, it can be seen that the FADTS algorithm achieved all targets within ± 1% of disagreement. For example, targets were set to estimate motion with on average 20 SP for *Football* sequence and 25 SP for *Flower Garden* sequence. The FADTS algorithm satisfied these demands using 20.10 and 24.69 search points with average MSE of 243.00 and 214.85 per pixel, respectively. The disagreement in these two cases is $(20 - 20.10)/20 = 0.5\%$ and $(24.69 - 25)/25 = -1.24\%$, respectively. These settings reveal that the FADTS algorithm is able to reach any bounded target

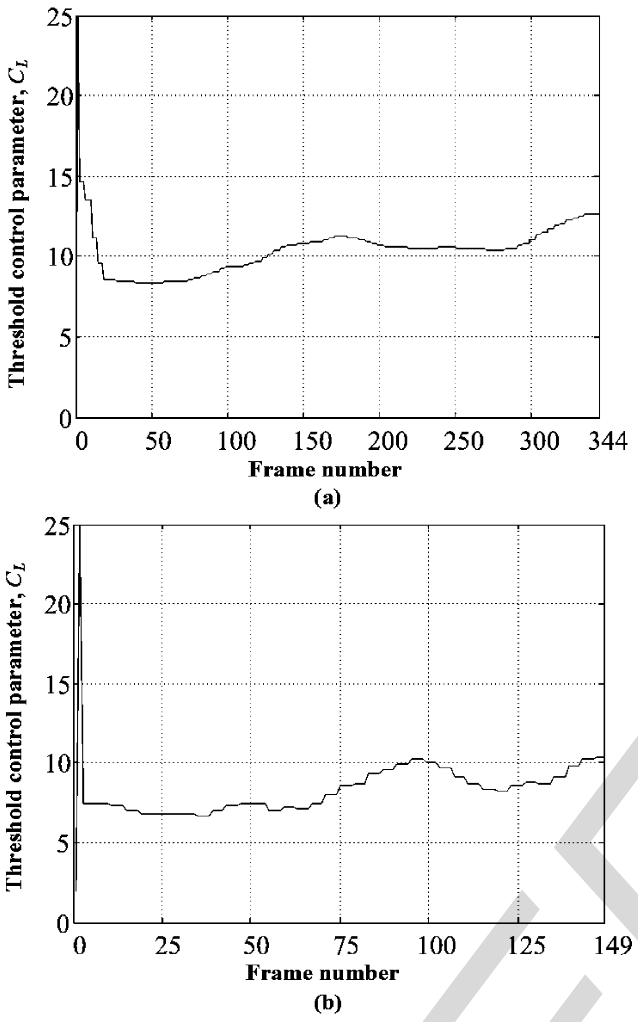


Fig. 14. Threshold control parameter adaptation for (a) *Football* sequence with quality target of 240 MSE per pixel and (b) *Flower Garden* sequence with quality target of 215 MSE per pixel.

TABLE V
SPEED ADAPTATION FOR *FOOTBALL* VIDEO SEQUENCE

Target SP	Actual SP	Actual Quality	
		MSE	PSNR [dB]
20	20.10	243.00	24.27
25	24.99	237.82	24.37
30	29.89	235.32	24.41
40	39.08	230.22	24.51

TABLE VI
SPEED ADAPTATION FOR *FLOWER* GARDEN VIDEO SEQUENCE

Target SP	Actual SP	Actual Quality	
		MSE	PSNR [dB]
15	15.52	220.54	24.70
20	19.68	216.27	24.78
25	24.69	214.85	24.81
30	29.68	214.19	24.82

level of speed, with the implicit assumption that the number of search points used by FS is the upper bound.

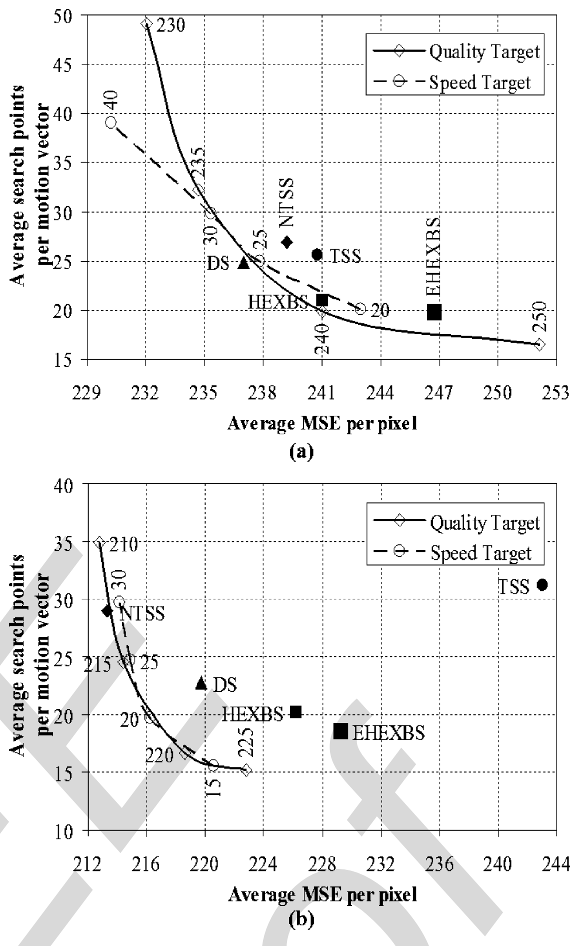


Fig. 15. Quality-speed performance of the TSS, DS, HEXBS, EHEXBS, and FADTS algorithms for (a) *Football* and (b) *Flower Garden* video sequences. The labels on the FADTS performance curves indicate the target values used.

C. *Quality-Speed Performance*

As has been clarified in Section III, the aim of a performance-management algorithm is to achieve a target image quality (speed) using as few search points (as low MSE) as possible. Whether FADTS can fulfil this objective optimally is an intractable problem, so instead the quality-speed performance of the FADTS algorithm is compared against existing fast algorithms to verify whether FADTS could match the performance of these fast algorithms in terms of quality (speed) for a prescribed target speed (quality). Fig. 15 plots the various quality-speed performance curves for the FADTS algorithm applying the data from Tables III–VI for both quality and speed adaptations together with the individual performance points for the TSS, NTSS, DS, HEXBS, and EHEXBS algorithms using the data in Table II. For the *Football* sequence Fig. 15(a), while FADTS outperformed TSS and NTSS in terms of both quality and speed adaptations, its performance is comparable to DS, HEXBS, and EHEXBS. Conversely, for the *Flower Garden* sequence Fig. 15(b), FADTS maintained the same performance as NTSS and provided superior results over all other fast algorithms for both adaptations.

In summary, therefore, the FADTS algorithm can adapt the threshold control parameter satisfactorily to achieve 1) any target quality without using any more search points per MV and 2) any target speed with no higher MSE per pixel than the existing fast algorithms. The main strength of the FADTS algorithm lies in its unique performance scalability. No other existing fast directional algorithm provides such a level of flexibility in trading off predicted image quality and computational complexity, whereas the FADTS algorithm demonstrates considerable flexibility in providing target-driven services, especially in terms of computational complexity.

If rate-distortion (RD) optimization technique [31] is embedded into a motion search algorithm, longer MVs are less likely to be selected as they incur more bits to encode. Fig. 15 reveals that FADTS used on average no more than 25 search points per MV to achieve the target MSE per pixel compared with the DS or HEXBS algorithms, so it can be reasonably concluded that majority of the MVs selected by FADTS were bounded by the search diamond SD_3 with MVs being no longer than three pixels, which is in fact less than one third of the maximum feasible length of $7\sqrt{2} \approx 10$ pixels. While DS and HEXBS used a similar number of search points, being directional no similar conclusion can be drawn on the length of their MVs, so RD optimization will, therefore, affect FADTS no more than it does affect DS and HEXBS.

VI. Conclusion

This paper has presented a novel FADTS algorithm based on NBLMS adaptive algorithm for performance-management block-based ME in real-time video coding applications. A key feature of this approach is the progressive adjustment of the required threshold control parameter via an adaptive process which uses the information from previous frames to achieve specified prediction quality and processing speed. The performance of the FADTS algorithm has been examined, and proof that it affords a unique feature in being able to tradeoff between two key model parameters, namely prediction quality and search speed, for the entire range of values of the threshold control parameter. Experimental results have shown that the FADTS algorithm has achieved guaranteed QoS demands, with performance scalability, particularly complexity scalability being inherent in the algorithm, and thereby representing an effective solution to the problem of performance scalability in real-time software-only or low power video coding applications.

The search efficiency of the FADTS algorithm has also been compared to other popular fast algorithms notably the superior diamond and hexagon-based search algorithms. Experimental results have proven that the FADTS algorithm is not only able to provide QoS but also demonstrates comparable or faster search speed for similar error performance and vice versa, thus addressing the problem of existing fast directional algorithms in providing different levels of quality of service.

Acknowledgment

The authors would like to thank the anonymous referees for their insightful comments, suggestions, and criticism that considerably improved the quality of this article.

References

[1] *Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to About 1.5 Mbit/s*, ISO/IEC 11172. MPEG-1, 1993.

[2] *Information Technology: Generic Coding of Moving Pictures and Associated Audio Information: Video*, ISO/IEC 13818. MPEG-2, 1995.

[3] *Video Codec for Audiovisual Services at* p $\times$ *64 kbit/s*, ITU-T Rec. H.261, 1993.

[4] *Video Coding for Low Bitrate Communication*, ITU-T Rec. H.263, 2000.

[5] *Joint Video Team (JVT) of ISO MPEG and ITU-T VCEG, JVT-G050*, ITU-T Rec. H.264/ISO/IEC 14496-10 AVC, 2003.

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

[7] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated inter frame coding for videoconferencing," in *Proc. Nat. Telecomm. Conf.*, New Orleans, LA, Dec. 1981, pp. G5. 3.1–G5.3.5.

[8] 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. 4, pp. .438–442, Aug. 1994.

[9] H. Nisar and T.-S. Choi, "An advanced center biased search algorithm for motion estimation," in *Proc. IEEE Int. Conf. Image Process.*, 2000, vol. 1, pp. 832–835.

[10] L. M. Po and W. C. Ma, "Novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[11] M. Ghanbari, "The cross-search algorithm for motion estimation (image coding)," *IEEE Trans. Commun.*, vol. 38, no. 7, pp. 950–953, Jul. 1990.

[12] ***sue number and month?]***L. Luo, C. Zou, X. Gao, and H. Zhenya, "A new prediction search algorithm for block motion estimation in video coding," *IEEE Trans. Consumer Electron.*, vol. 43, pp. 56–61, 1997.

[13] S. Zhu and K. -K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[14] C. Zhu, L.-P. Chau, and X. Lin, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 5, pp. 349–355, May 2002.

[15] C. Zhu, X. Lin, L. Chau, and L.-M. Po, "Enhanced hexagonal search for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1210–1214, Oct. 2004.

[16] M. Paul, M. Murshed, and L. Dooley, "A real-time pattern selection algorithm for very low bit-rate video coding using relevance and similarity metrics," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 6, pp. 753–761, Jun. 2005.

[17] I. E. G. Richardson, *Video Codec Design*. New York: Wiley, 2002.

[18] G. Sorwar, M. Murshed, and L. Dooley, "Fast block-based true motion estimation using variable distance dependent thresholds in the full-search algorithm," *J. Res. Practice Inf. Technol.*, vol. 36, no. 1, pp. 83–95, Feb. 2004.

[19] ***sue number and month?]***H. -K. Chow and M. L. Liou, "Genetic motion search algorithm for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 440–445, 1993.

[20] ***ue number and month?]***J. Feng, K. -T. Lo, H. Mehrpour, and A. E. Karbowiak, "Adaptive block matching algorithm for video compression," *Proc. IEE: Vis., Image Signal Process.*, vol. 145, pp. 173–178, 1998.

[21] D. -K. Lim and Y. -S. Ho, "A fast block matching motion estimation algorithm based on statistical properties of object displacement," in *Proc. IEEE Reg. 10 Int. Conf. Global Connectivity Energy, Comput., Commun. Contr.*, Piscataway, NJ, 1998, vol. 1, pp. 138–141.

[22] ***ue number and month?]***J. -B. Xu, L.-M. Po, and C. -K. Cheung, "Adaptive motion tracking block matching algorithms for video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 1025–1029, 1999.

[23] J. Feng, T. Y. Liu, K. T. Lo, and X. D. Zhang, "Adaptive motion tracking for fast block motion estimation," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sydney, NSW, Australia, 2001, vol. 5, pp. 219–222.

[24] ***sue number and month?]***J. -Y. Nam, J. -S. Seo, J. -S. Kwak, M. -H. Lee, and H. H. Yeong, "New fast-search algorithm for block matching motion estimation using temporal and spatial correlation of motion vector," *IEEE Trans. Consumer Electron.*, vol. 46, pp. 934–942, 2000.

[25] G. B. Thomas, *Calculus and Analytic Geometry*, 4th ed. New York: Addison-Wesley, 1968.

[26] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[27] J. Y. Stein, *Digital Signal Processing: A Computer Science Perspective*. New York: Wiley, 2000.

[28] ***sue number and month?]***E. Eweda and O. Macchi, "Convergence of the RLS and LMS adaptive filters," *IEEE Trans. Circuits Syst.*, vol. 34, pp. 799–803, 1987.

[29] ***sue number and month?]***S. C. Douglas, "A family of normalized LMS algorithms," *IEEE Signal Process. Lett.*, vol. 1, pp. 49–51, 1994.

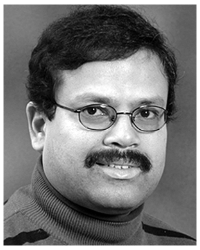
[30] K. Meghriche, S. Femmam, B. Derras, and M. Haddadi, "A non linear adaptive filter for digital data communication," in *Proc. 6th Int. Symp. Signal Process. Appl.*, Piscataway, NJ, 2001, vol. 1, pp. 304–306.

[31] ***sue number and month?]***G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Process. Mag.*, vol. 15, pp. 74–90, 1998.

Golam Sorwar (M'96) received the B.Sc. (hons.) degree in electrical and electronic engineering from Bangladesh University of Engineering and Technology (BUET), **Y?>** Bangladesh, in 1994, the M.Sc. degree in electrical, electronic and systems Engineeriung from the National University of Malaysia, **Y?>**Malaysia, in 1998, and the Ph.D. degree in information technology from Monash University, Churchill, Australia, in 2003.

He is currently a Lecturer at School of Commerce and Management, Southern Cross University, Coffs Harbour, Australia, where his major research interests are in the fields of multimedia signal processing and communication, electronic health and telemedicine applications, and artificial intelligence. He has published more than 30 peer-reviewed journal articles and conference papers.

Dr. Sorwar is a member of Australian Computer Society (ACS) and the Institute of Engineers of Bangladesh (IEB).



Manzur Murshed (M'96) received the B.Sc.Engg. (hons.) degree in computer science and engineering from Bangladesh University of Engineering and Technology (BUET), **Y?>**Bangladesh, in 1994 and Ph.D. degree in computer science from the Australian National University, Canberra, Australia, in 1999.

He is an Associate Professor and the Deputy Head of School at Gippsland School of Information Technology, Monash University, Churchill, Australia, where his major research interests are in the fields of multimedia signal processing and communications, parallel and distributed computing, simulations, and multilingual systems development. He received many research grants including a prestigious Australian Research Council (ARC) Discovery Projects grant. He has published more than 90 peer-reviewed journal articles, book chapters, and conference papers.

Dr. Murshed is the recipient of numerous academic awards including the University Gold Medal by BUET.

Laurence S. Dooley (M'82–SM'92) received the B.Sc. (hons.), M.Sc, and Ph.D. degrees in electrical engineering from the University of Wales, Swansea, U.K., in 1981, 1983, and 1987, respectively.

Since 1999, he has been Professor of Multimedia Technology in the Faculty of Information Technology, Monash University, Churchill, Australia, where his major research interests are in multimedia signal processing, mobile communications, image/video object segmentation, bioinformatics, wireless sensor networks, technology transfer and research and development commercialization strategies for regional small business. He has edited one book and published over 170 international scientific peer-reviewed journal, book chapters and conference papers. He has successfully supervised 13 PhD/Masters research students and is the co-inventor of two patents as well as being in receipt of over A$0.9M external grant funding from government and industry to support his various research projects. He has held a number of international Visiting Professorial positions including the Deutsches Zentrum für Luft- und Raumfahrt (DLR), Berlin, Germany, in 2006.

Prof. Dooley was jointly awarded the IEEE Communications Society sponsored Outstanding Paper Prize at the 1st International Conference on Next Generation Wireless Systems (ICNEWS'06) in 2006. He is a Chartered Engineer (C.Eng) and a corporate member of the BCS. **EASE SPELL OUT BCS.>**