Efficient tree structured motion estimation using successive elimination

M. Yang, H. Cui and K. Tang

Abstract: In H.264/AVC, tree structured motion estimation enhances the coding efficiency significantly while dramatically increasing the computational complexity of block matching. In the paper, a successive elimination algorithm (SEA) is implemented in tree structured motion estimation with a simple and effective method to determine the initial motion vector, which exploits the strong correlation among the partially overlapped variable-size blocks. With identical performance to a full search algorithm, computations for block matching can be reduced to 1%–20%. Further, the SEA can be improved by incorporating two early termination conditions, then named 'Quick SEA'. Finally, a novel fast motion estimation algorithm, successive elimination diamond search (SEDS), is proposed by efficiently integrating the Quick SEA and a modified diamond search pattern. Simulation results show that the proposed Quick SEA can reduce the computational complexity of block matching by 3–5 times compared to the basic SEA. SEDS further reduces by about one-half the computations of Quick SEA. With similar rate distortion performance, 0.2%–1% block matching distortion is calculated for SEDS with corresponding speed-up factors of 100 to 500 in comparison with the full search algorithm.

1 Introduction

Block matching based motion estimation and compensation is a fundamental process in international video compression standards [1–4], which can efficiently remove interframe redundancy. A full search algorithm (FS) exhaustively examines every candidate position in the search window to find the global optimal matched block in the reference frame. However, its highly intensive computations prevent practical implementation in real time. Many fast motion estimation algorithms have been proposed to alleviate the problem. Most algorithms are based on the well known *a priori* knowledge 'the block motion field of a real world image sequence is usually gentle, smooth and varies slowly'. Fast motion estimation algorithms can be mainly categorised into three families as described below.

1.1 Searching over a subset of possible candidate points with certain search patterns

Based on the assumption of convexity of the unimodal error surface, i.e. block matching distortion will increase monotonically away from the global minimum point, many gradient search methods with carefully designed search patterns have been developed to limit the search points to a small subset of all possible candidates. This category includes the well-known three-step search (3SS) [5], the new three-step search (N3SS) [6], the cross search (CS) [7], the one-dimensional gradient descent search (1DGDS) [8], the block-based gradient descent search (BBGDS) [9], the four-step search (4SS) [10], the diamond search (DS) [11], the cross-diamond search (CDS) [12] and the hexagon-based search (HEXBS) [13]. Although this category of algorithms may be trapped into a local minimum point and hence the efficiency of the motion compensation may drop, they can considerably reduce the number of block matching computations.

1.2 Prediction of the motion vector based on correlations among motion vectors

Motion in most natural image sequences involves a few blocks and lasts a few frames. So spatially or temporally adjacent blocks may have similar motion information. Taking advantage of the correlation among motion vectors, the search window can be constrained to a small clique of the predicted vector. Many prediction algorithms have been developed with quite different complexities. The prediction search algorithm (PSA) [14] simply predicts the current block motion vector as the mean value of its neighbouring blocks' motion vectors; fuzzy search [15] applies fuzzy logic to predict the motion vector; in [16] motion vectors are predicted by integral projections; in [17] a spatial-temporal AR model of motion vectors is established and an adaptive Kalman filter is employed; multiresolution search [18] downsamples the picture to obtain raw motion vectors at different resolution levels, then estimates finer motion vectors from the coarser ones; the multiresolution-spatio-temporal (MRST) scheme [18] adjusts the normal raster scan order to enable some blocks to obtain more motion information about neighbouring blocks from more directions, then combines a multiresolution scheme and spatio-temporal correlation to predict motion vectors. For some burst motions and blocks at the top-left corner without much correlation information, the performance of this category of algorithms may deteriorate because the refinement of prediction is restricted to a small search area. Moreover, too much prediction overhead may counteract the speed gain to some extent.

IEE Proceedings online no. 20040720

doi: 10.1049/ip-vis:20040720

Paper first received 11th July and in revised form 15th December 2003. Originally published online 4th October 2004

The authors are with State Key Lab on Microwave & Digital Communications, Department of Electronic Engineering, Tsinghua University, Beijing, 100084, People's Republic of China

1.3 Low complexity block matching criteria

The majority of the computations in motion estimation originate from computations of block matching distortion. In general, block matching metrics, such as mean absolute difference (MAD) and mean square error (MSE), involve pixel-wise operations, which are highly computationally intensive. Some methods try to simplify distortion computation by substituting the distortion of a subset of pixels for the whole block distortion. For instance, the MAD of 128 pixels is used as the matching distortion for a 16×16 macroblock in [18]; the computations can be reduced by one half with little performance loss. However, this method is not suitable for small blocks such as 4×4 blocks. Partial distortion elimination (PDE) [19] is employed in matching processes that compare every line's distortion in the block to avoid computing the distortion of the entire block. In [20] hypothesis testing is used to estimate the MAD from the partial mean absolute difference (PMAD), and the estimated MAD value is used to measure the matching effect.

When fast algorithms in the above three categories are incorporated together, the effectiveness of low complexity block matching criteria will suffer obvious degradation, and too much prediction computational overhead may counteract the speed gain of combination. All these fast motion estimation algorithms greatly reduce computational complexity by trading off decreasing reconstruction image quality or increasing coding bit rates. The speed-up factors for different image sequences range from tens to hundreds, relative to FS.

1.4 Overview of this work

Current work is based on the successive elimination algorithm (SEA) proposed by Li and Salari [21], which pre-excludes some impossible candidate points before calculating the matching distortion. SEA is a fast full search algorithm having performance identical to FS while speeding up the search process approximately by 10 times for 16×16 macroblock based motion estimation, as shown in [21]. Some further improvements have been made in subsequent research [19, 22–24].

The H.264/AVC standard enhances motion estimation accuracy by introducing the tree structured macroblock partition. The fundamental difference from motion estimation with fixed-size block in traditional video coding standards is that every image pixel has to be matched in several blocks with different sizes. Therefore tree structured motion estimation exhibits a particular motion vector correlation owing to the partial overlapping of different size blocks. In this paper, the SEA is applied in tree structured motion estimation and incorporated with a novel initial motion vector determination method exploiting the partially overlapped correlation. This algorithm reduces block matching distortion computations dramatically without performance loss and achieves much greater search speed gain factors compared with 16×16 macroblock based motion estimation. Based on an accurate initial motion vector and the unimodal assumption, we improve SEA to Quick SEA by introducing two early termination conditions, which further reduce block matching computations with negligible performance loss. Finally a novel fast motion estimation algorithm named successive elimination diamond search (SEDS) is proposed, which efficiently integrates Quick SEA and the modified DS. Extensive experimental results demonstrate that SEDS requires far fewer block matching distortion computations while maintaining a similar rate distortion performance to other fast block matching algorithms. Simulation results also reflect the performance of various popular fast motion estimation algorithms in the new tree structured motion estimation.

2 Application of SEA in H.264/AVC

2.1 Enhanced motion estimation in H.264/AVC

In H.264/AVC, novel techniques are incorporated to enhance the efficiency of motion estimation including multiple reference picture prediction, tree structured 1/4 pixel accuracy motion estimation and the generalised B-picture [25]. Each of these techniques contributes to more accurate motion estimation, but simultaneously increases the computational complexity significantly.

In tree structured motion estimation, the luminance component of a macroblock can be partitioned into four smaller blocks and each of these sub-macroblocks, i.e. 8×8 blocks, can be further split into four types of blocks, as shown in Fig. 1. Up to seven different size blocks ranging from 16×16 to 4×4 are used in motion estimation. Partitioning and sub-partitioning give rise to a large number of possible combinations within each macroblock.

The seven blocks with variable sizes are denoted $m_t \times n_t$; let t denote the block type, let m_t, with $m_t \in \{16, 16, 8, 8, 8, 4, 4\}$, denote the number of luminance pixels in the horizontal direction, and let n_t, with $n_t \in \{16, 8, 16, 8, 4, 8, 4\}$, denote the number of luminance pixels in the vertical direction.

2.2 Basic concept of SEA

The basic concept of the SEA is to remove impossible candidate points using geometric inequality before calculating distortions over these points.

The type t target block in the kth frame is denoted as $B_t^k(x, y)$ with top-left corner coordinates (x, y), and let $f^k(x, y)$ denote the intensity of the pixel at coordinates (x, y) in the kth frame. The motion estimation process tries to find the best matching block $B_t^{k-1}(x, y)$ in the $(k-1)$th reference frame that yields minimum matching distortion. The matching metric is the sum of the absolute difference (SAD) between the target block and the reference block.

First the sum norm of a block is defined as the sum of all pixel intensities within the block:

$$\|B_t^k(x, y)\| = \sum_{i=1}^{m_t} \sum_{j=1}^{n_t} |f^k(x+i, y+j)|$$

Then the SAD between the target block $B_t^k(x, y)$ and the reference block $B_t^{k-1}(x', y')$ can be formulated as

$$SAD\left(B_t^k(x, y), B_t^{k-1}(x', y')\right) = \left\|B_t^k(x, y) - B_t^{k-1}(x', y')\right\|$$

The sum norm difference (SND) between $B_t^k(x, y)$ and $B_t^{k-1}(x', y')$ can be expressed as

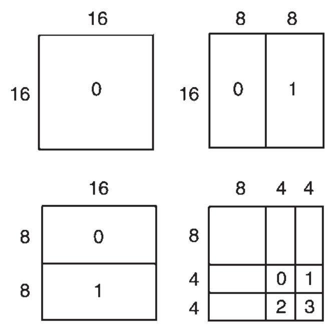


Fig. 1 *Macroblock and sub-macroblock partition in tree structured motion estimation*

$$SND\left(B_t^k(x,y), B_t^{k-1}(x',y')\right) = |\|B_t^k(x,y)\| - \|B_t^{k-1}(x',y')\||$$

From the triangle inequality [21]:

$$|\|B_t^k(x,y)\| - \|B_t^{k-1}(x',y')\|| \leq \|B_t^k(x,y) - B_t^{k-1}(x',y')\|$$

From the above derivation, the SND value of $B_t^{k-1}(x',y')$ is a lower bound of its SAD value. If the SAD of the current reference block is less than that of the current best matching reference block $B_t^{k-1}(x'',y'')$, it must satisfy the necessary condition:

$$SND\left(B_t^k(x,y), B_t^{k-1}(x',y')\right) \leq SAD\left(B_t^k(x,y), B_t^{k-1}(x'',y'')\right) \tag{1}$$

By evaluating candidate points using (1), some impossible candidates can be eliminated. When a better matching point is found, the current minimum SAD is updated to gradually diminish the possible points set, just as the name 'successive elimination' implies. When splitting the blocks into four sub-blocks or even more, the sum of the sub-blocks' SND is a tighter lower bound on the original block's SAD [22].

The SEA does not rely on any assumption regarding image sequences and has the same performance as FS. The reasons why the SEA can speed up the search process are: (i) the sum norms of the blocks in the reference frame can be utilised by all blocks in the current frame; and (ii) the SND value is a good lower bound for SAD, thus many candidates in the search window can be effectively eliminated. In 16×16 macroblock based motion estimation, the SAD of only about 6–14% of the blocks are actually calculated [21].

2.3 Applying SEA in H.264/AVC

2.3.1 Block matching metric:

In H.264/AVC, different numbers of motion vectors will be coded in different macroblock partition modes. When evaluating the rate distortion slope, the number of bits for the block matching error and the motion vectors should both be taken into account. Herewith, the rate distortion optimised sum of the absolute difference is used as the block matching distortion metric [26], which is defined as

$$RD\_SAD = SAD + \lambda rate(MV)$$

$rate(MV)$ is the number of bits for coded motion vectors, λ is the Lagrange multiplier. Under the measurement of RD_SAD, the distribution of final motion vectors becomes more centralised, as shown in Fig. 2.

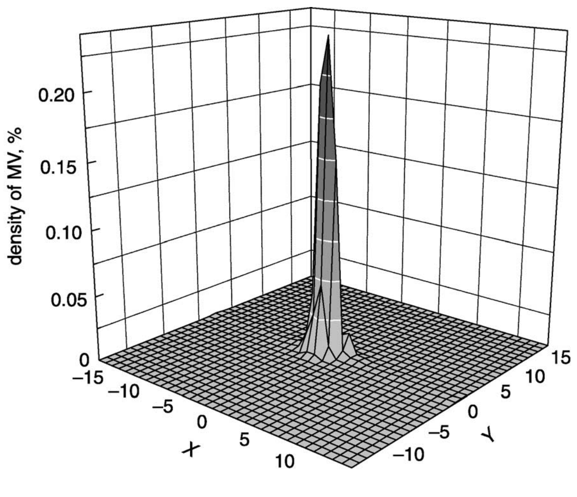


Fig. 2 *Probability distribution of motion vectors of 4×4 blocks over search window ±16 (100 frames in CIF sequence 'Mobile')*

2.3.2 The SEA in H.264/AVC

Stage (i) – Frame: For each reference frame, sum norms of all 4×4 blocks are calculated with the fast algorithm in [21].
Stage (ii) – Macroblock: For each macroblock in the current frame, calculate the SND values of sixteen 4×4 blocks on each search point. The SND values of larger blocks are derived from these of the 4×4 blocks.
Stage (iii) – Block: For each size block in the current macroblock.
Step 1. The initial motion vector, namely the predict motion vector (PMV), is determined by the method introduced in Section 3. *RD_SAD* of PMV is reserved in *RD_SAD_minimum*, and PMV is kept in Best_MV.
Step 2. Spiral search on every point in the search window, if SND value of the current point is less than *RD_SAD_minimum* and if the sum of SND value and $\lambda rate(\mathrm{MV})$ is also less than *RD_SAD_minimum*, the *RD_SAD* values of the current point are calculated using the PDE technique [19], otherwise this point is skipped. If the *RD_SAD* value of the current point is less than *RD_SAD_minimum*, *RD_SAD_minimum* and Best_MV are updated simultaneously.

The ultimate Best_MV is the final motion vector (FMV) of the block. Macroblock mode selection is performed after motion estimation to choose the best rate distortion partition. To avoid recalculating, calculated 4×4 block SAD values are stored for future use.

2.3.3 Analysis of SEA in H.264/AVC:

SEA is more efficient in H.264/AVC tree structured motion estimation than in traditional motion estimation with a fixed-size block. It benefits from the following factors: (i) The sum norms of large size blocks are derived from sum norms of 4×4 blocks obtained by a fast algorithm, so the ratio of computational overhead is much lower. (ii) The SND values of large size blocks are added from their 4×4 sub partitions, which are tighter lower bounds of SAD [22]. (iii) Strong correlations among partially overlapped variable-size blocks can offer a fairly accurate initial motion vector with a small initial *RD_SAD_minimum*. (iv) The computational overhead of SEA, sum norms of all 4×4 blocks in the reference frame, can be reused in bidirection motion estimation and multiple reference picture prediction.

It's worth noting that the parallelism of sum norm and SAD calculations is suited to software optimisation using the SIMD technique, and the regular data flow and search process of the SEA are suitable for hardware implementation.

3 Determination of initial motion vector

The initial motion vector of the motion estimation, namely the predict motion vector (PMV), has an important impact on the overall speed of motion estimation. The closer PMV is to the final motion vector (FMV), the more points are pre-excluded, the fewer SAD values are calculated and the more straightforward it is to apply the unimodal error surface assumption to further reduce the computations.

In tree structured motion estimation, motion vectors of partially overlapped blocks have strong correlations. For instance, each 4×4 block belongs to six larger size blocks and will be matched as part of them, so the motion vectors of 8×4, 4×8, 8×8 blocks including this 4×4 block can provide a fairly precise prediction. The correlation of motion vectors among blocks that have partially overlapped image pixels or even include one another is named 'partial overlapped correlation of motion vectors'. In our prediction method, four candidates for the initial motion vector are selected by employing spatial correlation and partially overlapped correlation. The one giving minimum matching distortion is chosen as the PMV. Owing to considerations of storage space and complexity, temporal correlation among motion vectors is not utilised in our method, which makes it compatible with encoding schemes using multiple reference pictures.

The search order of variable-size blocks is as follows: 8×8, 8×4, 4×8, 4×4, 8×16, 16×8, 16×16, to ensure that every size block has some similar size blocks previously searched. The same size blocks are searched in raster scan order. Four candidates for the initial motion vector are chosen by exploiting partially overlapped correlation among similar size blocks. The one with the minimum SAD value will be selected as the PMV. Except for 8×8 blocks, candidates for the initial motion vector are chosen from the FMV of searched blocks in the same macroblock. Consequently, storage memory required for motion vector prediction is close to that required for fixed-size block motion estimation.

The current target block is denoted $B_t^k(x, y)$ as in Section 2. Then $B_t^k(x, y - n_t)$ is the same type of block above it, $B_t^k(x - m_t, y)$ is the same type of block to the left of it, and $B_{t'}^k(x/m_{t'} \times m_{t'}, y/n_{t'} \times n_{t'})$ is the t' type block partially overlapped with $B_t^k(x, y)$. Selection of initial motion vector candidates is listed in Table 1.

The rationality of the selection of the initial motion vector candidates and the accuracy of this prediction method can be evaluated in two aspects:

(i) The accumulated probability of FMV in the $r \times r$ neighbourhood of PMV, which reflects the accuracy of the prediction. For example, when $r = 1$, FMV equals PMV; we call it 'hit'.
(ii) The probability that FMV equals PMV when the four candidates for initial motion vector are identical. This probability reveals the effectiveness of the selection of initial motion vector candidates.

Table 2 shows the ratio when four candidate positions are identical and in parentheses the corresponding 'hit' probability. Ten test sequences with 100 frames are used in the test. For instance, in the QCIF sequence 'Foreman', 32.9% of the PMV have four identical candidates for 8×8 blocks, among which 90.5% hit the FMV; in the CIF sequence 'Mobile', 56.0% of 16×16 blocks' four candidates are identical, and 99.9% of them hit the FMV. With more partially overlapped correlation, 'hit' probability rises from about 90% for 8×8 blocks to about 99.9% for 16×16 blocks in every sequence.

In the H.264/ AVC standard, the median of neighbouring blocks' motion vectors (left, top, right blocks) is used as the PMV. The differences between this PMV and FMV are actually encoded and transferred. This prediction method is denoted as *Orig Pred*. The accumulated probability that the FMV is located in the $r \times r$ central area of the PMV is compared in Table 3 between *Orig Pred* and the proposed prediction method *Our Pred*. The results of two high motion content sequences 'Mobile' and 'Stefan' are listed here. The accumulated probability of *Our Pred* is always higher than that of *Orig Pred* for every block size and has an obvious

Table 1: Selection of initial motion vector candidates

Block type	Candidate 1	Candidate 2	Candidate 3	Candidate 4
8×8	$B_4(x-8, y)$	$B_4(x, y-8)$	$B_4(x+8, y-8)$	last FMV
8×4	$B_5(x-8, y)$	$B_5(x, y-4)$	$B_4(x, y/8\times8)$	last FMV
4×8	$B_6(x-4, y)$	$B_6(x, y-8)$	$B_4(x/8\times8, y)$	last FMV
4×4	$B_4(x/8\times8, y/8\times8)$	$B_5(x/8\times8, y)$	$B_6(x, y/8\times8)$	last FMV
8×16	$B_4(x, y)$	$B_4(x, y+8)$	$B_4(x-8, y)$	last FMV
16×8	$B_4(x, y)$	$B_4(x+8, y)$	$B_4(x, y-8)$	last FMV
16×16	$B_2(x, y)$	$B_2(x, y+8)$	$B_3(x, y)$	$B_3(x+8, y)$

Table 2: Ratio when four candidates for initial motion vector are identical, and the corresponding 'hit' probability

Sequence	Foreman(QCF)	Car Phone(QCF)	Sales(QCF)	Silent(QCF)	Stefan(QCF)
8×8	32.9%(90.5%)	42.3%(92.7%)	90.5%(99.1%)	78.6%(98.3%)	39.2%(94.9%)
8×4	53.2%(93.1%)	57.9%(94.9%)	93.0%(99.3%)	83.2%(98.8%)	61.7%(94.3%)
4×8	46.7%(94.2%)	56.1%(96.1%)	92.6%(99.4%)	82.5%(98.8%)	55.8%(94.8%)
4×4	60.6%(94.2%)	65.3%(97.7%)	93.8%(99.5%)	85.4%(99.3%)	66.2%(96.6%)
8×16	43.2%(96.1%)	49.4%(93.0%)	90.7%(99.9%)	78.8%(99.2%)	53.1%(97.1%)
16×8	44.4%(94.7%)	50.3%(95.2%)	91.4%(99.9%)	79.4%(99.3%)	54.7%(97.3%)
16×16	47.6%(99.1%)	49.3%(98.5%)	92.7%(99.8%)	80.4%(99.9%)	56.8%(99.8%)
Sequence	Table(QCF)	Stefan(CIF)	Paris(CIF)	Tempete(CIF)	Mobile(CIF)
8×8	30.6%(92.6%)	32.8%(93.0%)	68.0%(97.6%)	43.4%(92.3%)	31.1%(90.5%)
8×4	47.5%(93.4%)	56.3%(93.0%)	76.4%(97.9%)	60.8%(93.1%)	53.8%(90.9%)
4×8	40.9%(95.3%)	47.8%(93.9%)	75.4%(98.1%)	56.9%(94.0%)	44.2%(91.0%)
4×4	57.0%(96.6%)	60.8%(95.7%)	80.4%(98.8%)	64.2%(95.4%)	57.2%(94.0%)
8×16	38.2%(91.4%)	46.2%(94.9%)	71.4%(97.4%)	51.1%(98.0%)	44.4%(98.4%)
16×8	39.6%(93.5%)	47.8%(96.2%)	71.9%(98.1%)	53.2%(98.2%)	47.8%(98.8%)
16×16	38.9%(98.9%)	49.2%(99.7%)	71.2%(99.7%)	58.7%(99.7%)	56.0%(99.9%)

Table 3: Accumulated probability that FMV is located in the $r \times r$ neighbourhood of PMV for 100 frames of CIF sequences 'Mobile' and 'Stefan'

Mobile(CIF)	$r \times r$	8×8	8×4	4×8	4×4	8×16	16×8	16×16
Orig Pred	1×1	0.42515	0.42586	0.42702	0.43170	0.42042	0.41937	0.42707
	3×3	0.92150	0.91804	0.91252	0.91196	0.93141	0.93164	0.94546
	5×5	0.94266	0.94067	0.93623	0.93639	0.95199	0.95178	0.96340
	7×7	0.95232	0.95146	0.94797	0.94848	0.95970	0.95992	0.96956
Our Pred	1×1	0.83093	0.86304	0.86554	0.90574	0.95186	0.95642	0.98684
	3×3	0.94745	0.95828	0.95580	0.96964	0.98375	0.98450	0.99524
	5×5	0.95907	0.96813	0.96625	0.97647	0.98758	0.98867	0.99635
	7×7	0.96690	0.97452	0.97316	0.98132	0.98995	0.99084	0.99710
Stefan (CIF)	$r \times r$	8×8	8×4	4×8	4×4	8×16	16×8	16×16
Orig Pred	1×1	0.48171	0.48887	0.48709	0.49566	0.45395	0.46419	0.46241
	3×3	0.81611	0.81992	0.82621	0.83390	0.78764	0.79293	0.79697
	5×5	0.84219	0.84642	0.85290	0.86074	0.81891	0.82294	0.82574
	7×7	0.86444	0.86779	0.87435	0.88166	0.84528	0.84796	0.85096
Our Pred	1×1	0.80626	0.85323	0.86243	0.90495	0.88734	0.89287	0.94562
	3×3	0.90451	0.93464	0.94107	0.96105	0.93785	0.94226	0.97562
	5×5	0.92015	0.94475	0.95118	0.96713	0.95039	0.95333	0.98131
	7×7	0.93221	0.9524	0.95877	0.97191	0.95876	0.96130	0.98485

increasing trend from 8×8 to 16×16 blocks because more partially overlapped correlations are obtained.

4 Improvement of SEA

4.1 Quick SEA

The SEA has some drawbacks for some low motion sequences with little detail because the sum norms of many search points are so close that they cannot be eliminated by triangle inequality. In the worst case, for two frames with the same DC coefficient only, motion estimation cannot benefit from SEA. On the other hand, the PMV obtained by exploiting partially overlapped correlation is very close to the FMV and a large proportion even hit the FMV. Therefore with the unimodal error surface assumption, two early termination conditions can be employed to simplify the FMV determination process to further reduce the number of computations. We call this improvement of SEA 'Quick SEA', which has similar performance to FS.

Early termination condition 1. In the initial motion vector determination process, when four candidate positions are identical, the PMV is determined to be FMV, stop searching.
Early termination condition 2. If *RD_SAD* of the PMV is the minimum point in its 3×3 neighbourhood, the PMV is determined to be FMV, stop searching.

Conditions 1 and 2 are inserted in the basic SEA stage (iii) ahead of step 2. If the PMV does not meet conditions 1 and 2, continue a spiral search on all candidate points in the search window.

4.2 Successive elimination diamond search (SEDS)

The relatively low overhead of SEA in tree structured motion estimation and the accurate prediction of the initial motion vector suggest integrating SEA with some efficient search patterns to further reduce the number of block matching computations. Therefore, we proposed a novel and efficient fast motion estimation algorithm, named 'successive elimination diamond search' (SEDS), which incorporates SEA, the proposed determination method for the initial motion vector, the two early termination conditions and a modified diamond search pattern. An enlarged diamond search pattern (EDSP) including 13 points, as shown in Fig. 3, are first checked instead of the nine points LDSP in the DS [11] to employ early termination techniques. The successive elimination technique ensures that the four additional search points do not involve many more calculations.

Proposed SEDS Algorithm
Stage (i) – Frame: For each reference frame, sum norms of all 4×4 blocks are calculated with the fast algorithm in [21].
Stage (ii) – Block: For each size block in the current macroblock.

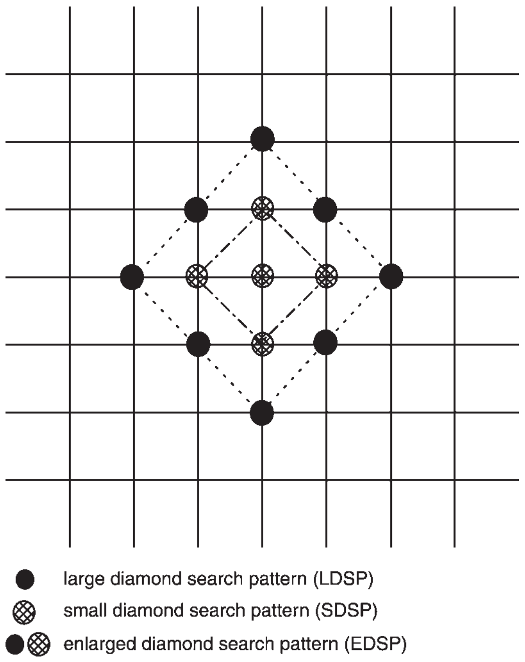

● large diamond search pattern (LDSP)
▨ small diamond search pattern (SDSP)
●▨ enlarged diamond search pattern (EDSP)

Fig. 3 *Search patterns used in SEDS*

Step 1. Initial motion vector is determined by the proposed method in Section 3. *RD_SAD* of PMV is reserved in *RD_SAD_minimum*. PMV is kept in Best_MV. If the four candidates for the initial motion vector are identical, the search stops.
Step 2. The minimum point is found from the 13 points of EDSP with the central point PMV. If the minimum point is located in the five points of the inner small diamond search pattern (SDSP), the search stops. Otherwise the regular diamond search process in [11] is performed until the minimum point is found at the centre of SDSP.
Step 3. For each point, if the SND value is less than *RD_SAD_minimum* and if the sum of SND value and $\lambda rate(MV)$ is also less than *RD_SAD_minimum*, the *RD_SAD* of the current point is calculated using the PDE technique [19], otherwise this point is skipped. Calculated 4×4 block SAD values are stored for future use.

Quick SEA reduces the computational complexity of block matching by 3–5 times compared to the basic SEA described in Section 2, with little performance degradation. The SEDS further reduces the number of matching computations in Quick SEA by about two times. The overall rate distortion performance of Quick SEA and SEDS are compared with FS, SEA, BBGDS, HEXBS, DS and 4SS in Section 6. The DS pattern is replaced by the BBGDS pattern to form a SE-BBGDS algorithm to demonstrate the effectiveness of the modified DS search pattern.

5 Computational complexity analysis

In this Section, the computational complexity of FS is first presented, then the computation overhead of SEA is analysed. The conclusion is that the ratio of actually calculated 4×4 block SAD is a proper indicator of the computational complexity in tree structured motion estimation.

To facilitate the analysis, assume each add, subtract or if decision operation needs 1 calculation; absolute value needs 1.5 calculations. $C(X)$ represents the computational amount of X. Under these assumptions, the SAD of a 4×4 block requires 55 calculations and is regarded as the basic computation unit to measure other computations. Operations that involve many fewer calculations than the 4×4 block are ignored.

Suppose that the coding image sequence has $F \times (K+1)$ frames, the number of B-frames between two successive P-frames is K, and L multiple reference frames are employed. Image frame size is $W \times H$, search window is $[-S, +S]$, and the maximum search position is $(2S+1)^2$, seven different size blocks in a $M \times M$ macroblock ($M = 16$) are involved in motion estimation. The minimum block size is $N \times N$ pixels ($N = 4$).

In FS, for every macroblock, SAD values of 16 4×4 blocks at each search position have to be calculated, while the SAD values of other size blocks are derived from these 4×4 SAD values. The computation amount of motion estimation for the whole sequence is

$$C(FS) = \frac{FWH(KL+K+L)}{N^2}(2S+1)^2 C(4 \times 4SAD)$$

In SEA, sum norms of every $N \times N$ block in the reference images are calculated using the fast algorithm [21]. Sum norm computations of the whole sequence are denoted $C(sum)$:

$$C(sum) = \frac{F+1}{55}[4WH - (H-N)(N+3) - 3W(N+1)]C(4 \times 4SAD)$$

For one macroblock, the computation amount related to determination of the initial motion vector is denoted as $C(MB\_PRED)$, which is the product of prediction time and one prediction computation amount $C(MB\_PRED)$:

$$C(MB\_PRED) = \sum_{t=1}^{7} \frac{M^2}{m_t n_t} C(MV\_PRED) = 41C(MV\_PRED)$$

$C(MV\_PRED)$ involves three look-up table operations and three comparisons; the possible four point SAD calculations are counted later.

The computation amount of SEA can be expressed as

$$C(SEA) = C(sum) + \eta C(FS)$$

where η is the ratio of actually calculated 4×4 block SAD values in SEA over that of FS, including the SAD calculations in the initial motion vector prediction and in the search process.

The ratio of $C(sum)$ to $C(FS)$ is

$$\mu = \frac{C(sum)}{C(FS)} \le \frac{4N^2}{55(KL+K+L)(2S+1)^2}$$

where μ is unrelated to frame size. For $N = 4$, suppose $K = 1$, $L = 1$, $S = 16$, and $\mu < 0.0003$. When L and K are larger than 1, μ will become even smaller. So η, the ratio of actually calculated 4×4 block SAD, mainly reflects the computation complexity in SEA. Similarly, η also indicates the complexity in Quick SEA and the SEDS. In the above analysis, some operations are ignored, such as the $\lambda rate(MV)$ computation when SND is less then *RD_SAD_minimum*; they are also proportional to η.

Table 4: Ratio η of actually calculated 4×4 block SAD in the SEA, Quick SEA and SEDS for 10 test sequences

η	Foreman(QCIF)	Car Phone(QCIF)	Sales(QCIF)	Silent(QCIF)	Stefan(QCIF)
SEA	0.02366(42.26)	0.02006(49.85)	0.01038(96.37)	0.01587(62.99)	0.07795(12.83)
Quick SEA	0.00779(128.4)	0.00638(156.6)	0.00269(371.3)	0.00697(143.4)	0.01678(59.58)
SEDS	0.00536(186.6)	0.00423(236.4)	0.00170(587.9)	0.00321(311.4)	0.00726(137.7)
η	Table(QCIF)	Stefan(CIF)	Paris(CIF)	Tempete(CIF)	Mobile(CIF)
SEA	0.19367(5.163)	0.08160(12.25)	0.03395(29.45)	0.05293(18.89)	0.10456(9.564)
Quick SEA	0.03522(28.39)	0.02404(41.60)	0.00909(110.0)	0.01266(78.99)	0.01845(54.19)
SEDS	0.00971(103.0)	0.00839(119.2)	0.00389(257.2)	0.00566(176.8)	0.00783(127.7)

6 Experiment results

CIF (352×288) and QCIF (176×144) format video sequences with different motion content were employed in simulations within the H.264/AVC JM61e framework. One CIF format and one QCIF format 'Stefan' were encoded to show the frame size influence.

For each sequence, 101 frames were encoded with the H.264/AVC Main Profile except the multiple reference pictures. The encoding parameters were as follows: QP equals 28 for one I-frame and 50 P-frames, and 30 for 50 B-frames, the frame rate was 30 frames per second (fps), search window $[-16, +16]$, rate-distortion optimisation (RDO) was used to select the best macroblock partition mode.

Table 4 compares the ratio η of actually calculated 4×4 block SAD of the SEA, Quick SEA and SEDS for the 10 test sequences. The reciprocal of η is given in parentheses as a straightforward indication of speed-up ratio compared to

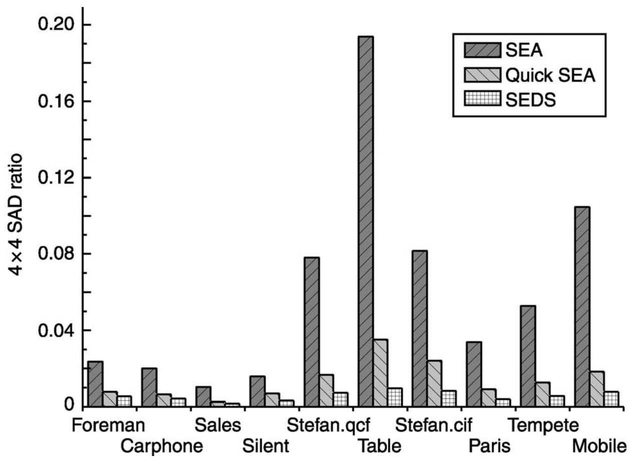


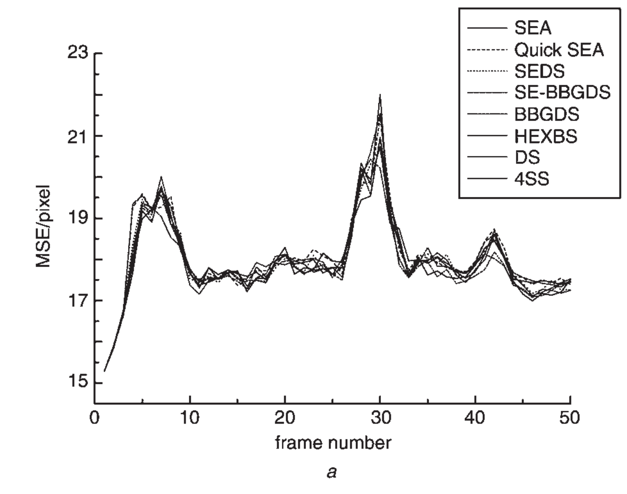
Fig. 4 *Column graph of the ratio η of actually calculated 4×4 block SAD in SEA, Quick SEA and SEDS for 10 test sequences*

Table 5: Comparison of overall rate distortion performance for 101 frames of different sequences

Sales	PSNRY (dB)	MSE	Bitrate (kbit/s)	Points	η	Speedup
FS	35.75	19.94	48.43	1089	1	1
SEA	35.75(0.00)	19.94(+0.00%)	48.43(+0.00%)	4.68	0.01038(100%)	96.87
Quick SEA	35.75(0.00)	20.69(+3.76%)	49.02(+1.22%)	0.565	0.00269(25.9%)	371.3
SEDS	35.73(−0.02)	20.64(+3.54%)	48.80(+0.76%)	0.207	0.00170(16.4%)	587.9
SE-BBGDS	35.76(+0.01)	20.77(+4.13%)	49.22(+1.63%)	0.498	0.00229(22.1%)	436.7
BBGDS	35.76(+0.01)	20.30(+1.80%)	48.60(+0.35%)	8.188	0.00867(83.5%)	115.4
HEXBS	35.74(−0.01)	20.30(+1.81%)	48.62(+0.39%)	12.02	0.01067(103%)	93.75
DS	35.75(0.00)	20.25(+1.55%)	48.69(+0.54%)	13.05	0.01247(120%)	80.20
4SS	35.74(−0.01)	20.26(+1.57%)	48.83(+0.83%)	16.02	0.01626(157%)	61.49
Foreman	**PSNRY (dB)**	**MSE**	**Bitrate (kbit/s)**	**Points**	**η**	**Speedup**
FS	35.58	22.44	88.31	1089	1	1
SEA	35.58(0.00)	22.44(+0.00%)	88.31(+0.00%)	8.865	0.02366(100%)	42.27
Quick SEA	35.53(−0.05)	23.18(+3.33%)	89.12(+0.92%)	2.047	0.00779(32.9%)	128.6
SEDS	35.55(−0.03)	23.23(+3.53%)	88.72(+0.46%)	1.330	0.00536(22.7%)	186.6
SE-BBGDS	35.58(0.00)	23.10(+2.95%)	88.84(+0.60%)	3.692	0.00789(33.3%)	126.8
BBGDS	35.56(−0.02)	23.11(+2.98%)	88.22(−0.10%)	9.152	0.01105(46.7%)	90.50
HEXBS	35.53(−0.05)	23.55(+4.93%)	89.44(+1.28%)	12.08	0.01403(59.3%)	71.26
DS	35.58(0.00)	22.86(+1.89%)	89.07(+0.86%)	13.34	0.01576(66.6%)	63.44
4SS	35.58(0.00)	23.00(+2.50%)	88.59(+0.32%)	16.12	0.02056(86.9%)	48.65
Tempete	**PSNRY (dB)**	**MSE**	**Bitrate (kbit/s)**	**Points**	**η**	**Speedup**
FS	34.07	56.06	803.9	1089	1	1
SEA	34.07(0.00)	56.06(+0.00%)	803.9(+0.00%)	24.42	0.05293(100%)	18.97
Quick SEA	34.08(+0.01)	59.09(+5.40%)	804.6(+0.09%)	3.456	0.01266(23.9%)	78.99
SEDS	34.07(0.00)	62.44(+11.4%)	805.6(+0.21%)	1.503	0.00566(10.7%)	176.8
SE-BBGDS	34.07(0.00)	64.57(+15.2%)	806.8(+0.36%)	2.700	0.00649(12.3%)	154.1
BBGDS	34.07(0.00)	62.96(+12.3%)	807.4(+0.44%)	9.244	0.01110(20.9%)	90.42
HEXBS	34.07(0.00)	61.58(+9.84%)	805.5(+0.20%)	12.11	0.01408(26.6%)	71.03
DS	34.07(0.00)	61.70(+10.1%)	805.2(+0.16%)	13.32	0.01580(29.8%)	63.30
4SS	34.07(0.00)	61.20(+9.16%)	804.8(+0.11%)	16.12	0.02045(38.6%)	48.91
Mobile	**PSNRY (dB)**	**MSE**	**Bitrate (kbit/s)**	**Points**	**η**	**Speedup**
FS	33.33	66.44	1243.7	1089	1	1
SEA	33.33(0.00)	66.44(+0.00%)	1243.7(+0.00%)	52.62	0.10455(100%)	9.565
Quick SEA	33.33(0.00)	68.25(+2.72%)	1246.8(+0.25%)	5.769	0.01845(17.6%)	54.19
SEDS	33.32(−0.01)	69.06(+3.95%)	1244.1(+0.03%)	2.886	0.00783(7.50%)	127.7
SE-BBGDS	33.32(−0.01)	69.95(+5.29%)	1246.7(+0.24%)	4.544	0.00835(8.00%)	119.8
BBGDS	33.33(0.00)	68.93(+3.75%)	1243.5(−0.02%)	9.458	0.01097(10.5%)	91.13
HEXBS	33.32(−0.01)	69.19(+4.15%)	1242.8(−0.07%)	12.10	0.01472(14.1%)	67.93
DS	33.32(−0.01)	68.48(+3.07%)	1242.3(−0.11%)	13.37	0.01594(15.2%)	62.74
4SS	33.32(−0.01)	68.16(+2.59%)	1241.3(−0.19%)	16.13	0.02099(20.1%)	47.65

FS. The corresponding column graph is shown in Fig. 4. In the SEDS, for the video conference sequences 'Sales' and 'Silent' the speed-up factors reach 587.9 and 311.43 respectively and for other sequences with a higher degree of motion the ratios are all below 1% over the FS.

Table 5 compares the overall rate distortion performance of FS, SEA, Quick SEA, SEDS, SE-BBGDS, BBGDS, HEXBS, DS and 4SS. All eight fast algorithms employ our initial motion vector determination method. Owing to space limitation, the results of only four representative sequences are tabulated here. Peak signal noise ratio (PSNR) of luminance component and average MSE per pixel reflect the reconstructed image quality. The differences relative to FS are listed in parentheses. Encoding bit rates and the changed amount over FS are listed in column 'Bitrate'. Computational complexity declinations are measured with the ratio η of actually calculated 4×4 block SADs, and the ratio of η relative to the basic SEA is given in parentheses. The 'Speedup' column shows the reciprocal of η. 'Average search points' is not an adequate indicator of search speed in tree structured motion estimation. It is listed here for comparison with previous work. The possible four test points related to initial motion vector determination are not included in average search points but are counted in the ratio η. The proposed SEDS calculates 7.50% of 4×4 block SAD values over the SEA for sequence 'Mobile' with 0.0 3% increment in bit rates and 3.95% in average MSE. With similar rate distortion performance, the SEDS accomplishes a 2–6 times search speed improvement over the SEA, BBGDS, HEXBS, DS and 4SS.



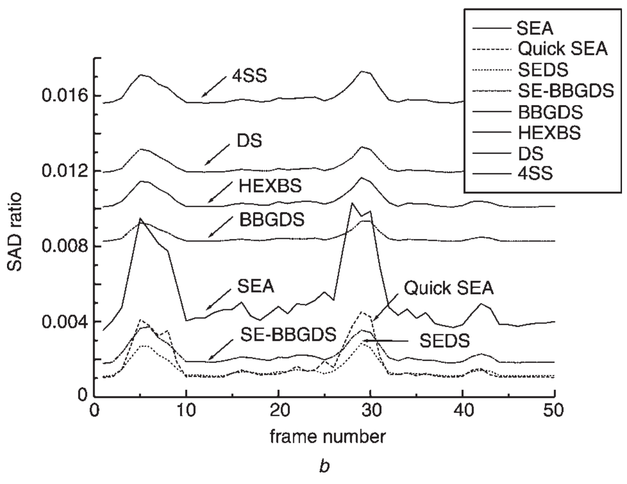
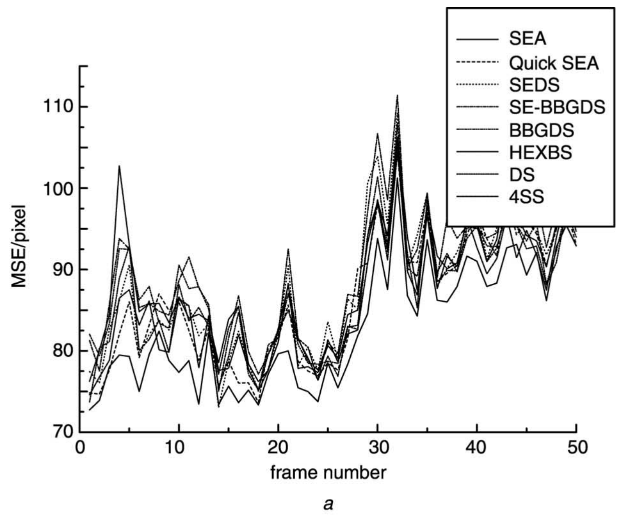


Fig. 5 *Frame-wise comparison of different block matching algorithms on QCIF sequence 'Sales'*

a Average MSE per pixel for 50 B-frames
b Ratio of actually calculated 4×4 block SAD for 50 B-frames



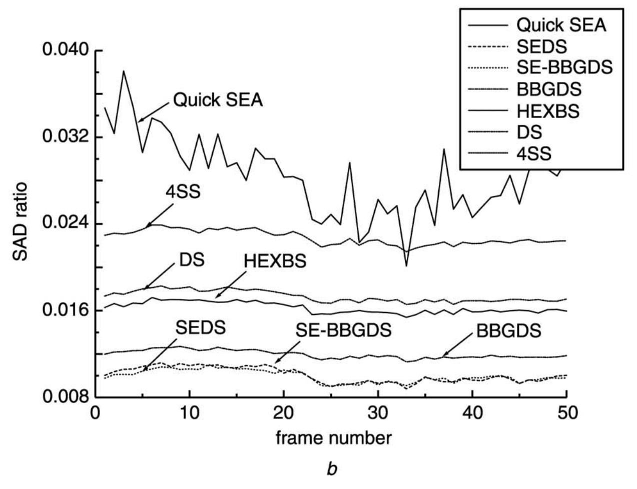


Fig. 6 *Frame-wise comparison of different block matching algorithms on CIF sequence 'Mobile'*

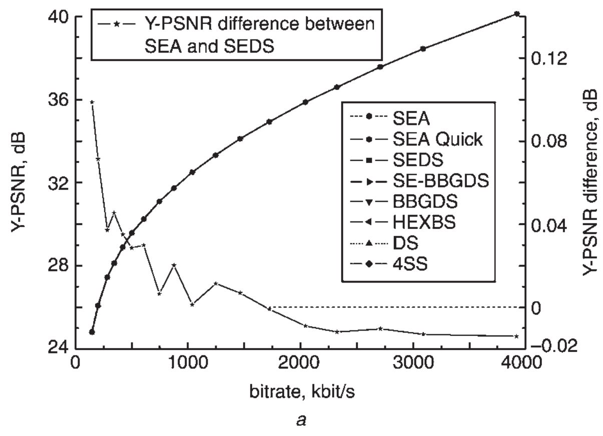
a Average MSE per pixel for 50 P-frames
b Ratio of actually calculated 4×4 block SAD for 50 P-frames

Frame by frame comparison of MSE per pixel and the ratio of actually calculated SAD of 50 B-frames in 'Sales' are illustrated in Figs 5*a* and 5*b*. Simulation results for 50 P-frames in 'Mobile' are illustrated in Figs 6*a* and 6*b*. These Figures show that the SEDS algorithm always has the smallest computational complexity while exhibiting comparable distortion in terms of MSE per pixel.

The rate distortion curve and the calculated ratio of 4×4 block SAD for 'Mobile' are shown in Fig. 7. The PSNR difference of luminance component between the SEA and SEDS is magnified in Fig. 7*a*. The PSNR loss of SEDS is below 0.1 dB and in the high bit rate situation the PSNR of SEDS is even a little higher than that of the SEA. Figure 7*b* illustrates that the calculated SAD ratio of the SEDS is always less than the other fast motion estimation algorithms over a wide range of bit rates.

7 Conclusions

In this paper, SEA has been applied to H.264/AVC tree structured motion estimation with a proposed simple and effective method to determinate the initial motion vector, which exploits the partially overlapped correlations among motion vectors. A novel fast motion estimation algorithm SEDS is proposed by incorporating the concept of SEA,



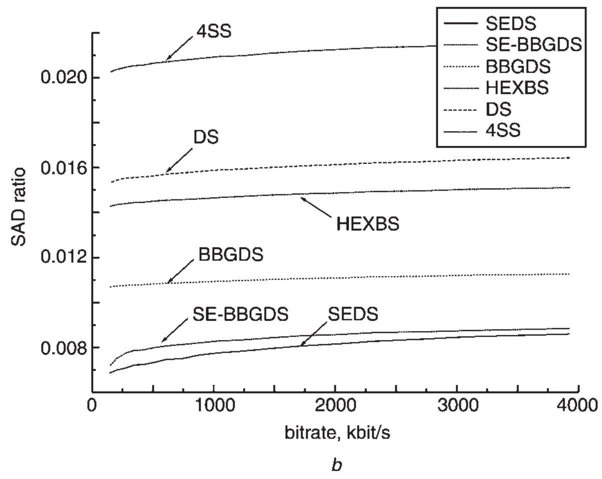


Fig. 7 *Performance comparison for CIF sequence 'Mobile' at 30 frames/s for a wide range of bit rates*

a Rate distortion curves
b Ratio of actually calculated 4 × 4 block SAD

partially overlapped correlations, early termination techniques and a modified diamond search pattern. Extensive experiments illustrate that the SEDS can dramatically reduce the number of block matching computations, ranging from 0.2% to 1% in comparison with FS, while maintaining similar rate distortion performance. This algorithm outperforms SEA, BBGDS, HEX, DS and 4SS in search speed and is very efficient in real-time applications of H.264/AVC video encoding.

8 Acknowledgments

This work was supported by the National Natural Science Foundation of China (60272020.) in part by the National Fundamental Research Program (G1998030406.). The authors would like to thank the anonymous reviewers for their constructive and helpful comments.

9 References

1 ITU-T Rec. H.263 'Video coding for low bit rate communication', Feb. 1998
2 ITU-T Rec.H.264, ISO/IEC 14496-10 AVC 'Advanced video coding', Final Draft International Standard, JVT-G050r1, Geneva, Switzerland, May 2003
3 ITU-T Rec.H.262, /ISO/IEC 13818-2 'Generic coding of moving pictures and associated audio information: Video (MPEG-II video)' 2000
4 ISO/IEC 14469-2(MPEG-IV visual). 'Coding of audio visual objects-Part 2: Visual'. 1999
5 Jain, J., and Jain, A.: 'Displacement measurement and its application in interframe image coding', *IEEE Trans. Commun.*, 1981, **29**, (12), pp. 1799–1808
6 Li, R., Zeng, B., and Liou, M.L.: 'A new three-step search algorithm for block motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 1994, **4**, (4), pp. 438–443
7 Ghanbari, M.: 'The cross-search algorithm for motion estimation', *IEEE Trans. Commun.*, 1990, **38**, (7), pp. 950–953
8 Chen, O.T.-C.: 'Motion estimation using a one-dimensional gradient descent search', *IEEE Trans. Circuits Syst. Video Technol.*, 2000, **10**, (4), pp. 608–616
9 Liu, L.-K., and Feig, E.: 'A block-based gradient descent search algorithm for block motion estimation in video coding', *IEEE Trans. Circuits Syst. Video Technol.*, 1996, **6**, (4), pp. 419–422
10 Po, L.-M., and Ma, W.-C.: 'A novel four-step search algorithm for fast block motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 1996, **6**, (2), pp. 313–317
11 Zhu, S., and Ma, K.-K.: 'A new diamond search algorithm for fast block matching motion estimation', *IEEE Trans. Image Process.*, 2000, **9**, (2), pp. 287–290
12 Cheung, C.-H., and Po, L.-M.: 'A novel cross-diamond search algorithm for fast block motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 2002, **12**, (12), pp. 1168–1177
13 Zhu, C., Lin, X., and Chau, L.-P.: 'Hexagon-based search pattern for fast block motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 2002, **12**, (5), pp. 349–355
14 Luo, L., Zou, C., Gao, X., and He, Z.: 'A new prediction search algorithm for block motion estimation in video coding', *IEEE Trans. Consumer Electron.*, 1997, **43**, (1), pp. 56–61
15 Roan, Y.-T., and Chen, P.-Y.: 'A fuzzy search algorithm for the estimation of motion vectors', *IEEE Trans. Broadcast.*, 2000, **46**, (2), pp. 121–127
16 Lee, J.H., and Ra, J.B.: 'Block motion estimation based on selective integral projections'. Int. Conf. on Image Processing ICIP'2002, 22–25 Sept. 2002, vol. 1, pp. 689–692
17 Kuo, C-M., Chao, C-P., and Hsieh, C-H.: 'A new motion estimation algorithm for video coding using adaptive kalman filter', *Real-Time Imaging*, 2002, **8**, pp. 387–398
18 Chalidabhongse, J., and Kuo, C.-C.J.: 'Fast motion vector estimation using multiresolution-spatio-temporal correlations', *IEEE Trans. Circuits Syst. Video Technol.*, 1997, **7**, (3), pp. 477–488
19 Wang, H.-S., and Mersereau, R.M.: 'Fast algorithms for the estimation of motion vectors', *IEEE Trans. Image Process.*, 1999, **8**, (3), pp. 435–438
20 Lengwehasatit, K., and Ortega, A.: 'Probabilistic partial-distance fast matching algorithms for motion estimation', *IEEE Trans. Circuits Syst. Video Technol.*, 2001, **11**, (2), pp. 139–152
21 Li, W., and Salari, E.: 'Successive elimination algorithm for motion estimation', *IEEE Trans. Image Process.*, 1995, **4**, (1), pp. 105–107
22 Jung, S.-M., Shin, S.-C., Baik, H., and Park, M.-S.: 'New fast successive elimination algorithm'. Proc. 43rd IEEE Midwest Symp. on Circuits and Systems, 8–11 Aug. 2000, vol. 2, pp. 616–619
23 Gao, X.Q., Duanmu, C.J., and Zou, C.R.: 'A multilevel successive elimination algorithm for block matching motion estimation', *IEEE Trans. Image Process.*, 2000, **9**, (3), pp. 501–504
24 Jung, S.-M., Shin, S.-C., Baik, H., and Park, M.-S.: 'Efficient multilevel successive elimination algorithms for block matching motion estimation', *IEE Proc., Vis., Image Signal Process.*, 2002, **149**, (2), pp. 73–84
25 Flierl, M., and Girod, B.: 'Generalized B pictures and the draft H.264/AVC video-compression standard', *IEEE Trans. Circuits Syst. Video Technol.*, 2003, **13**, pp. 587–597
26 Wiegand, T., Schwarz, H., and Joch, A.: 'Rate-constrained coder control and comparison of video coding standards', *IEEE Trans. Circuits Syst. Video Technol.*, 2003, **13**, pp. 688–703