VERSITA

OPTO-ELECTRONICS REVIEW **21**(1), 86–102

DOI: 10.2478/s11772-013-0071-0

Block-based motion estimation algorithms – a survey

M. JAKUBOWSKI* and G. PASTUSZAK

Institute of Radioelectronics, Warsaw University of Technology, 15/19 Nowowiejska Str., 00–665 Warsaw, Poland

In the multi-view video coding, both temporal and inter-view redundancies can be exploited by using standard block-based motion estimation (BBME) technique. In this paper, an extensive review of BBME algorithms proposed within the last three decades is presented. Algorithms are divided into five categories: 1) based on the search position number reduction; 2) multiresolution; 3) based on the simplification of matching criterion; 4) fast full search; 5) computation-aware. Algorithms are compared in terms of their efficiency and computational complexity.

Keywords: motion/disparity estimation, multi-view video, video compression.

1. Introduction

In the video sequence coding, motion estimation (ME) plays a vital role in a temporary redundancy reduction between frames. Due to its simplicity and efficiency [1], block-based ME (BBME) [2] has been adopted in several international video coding standards such as MPEG-x, H.26x, and VC-1 [3,4]. In BBME, the current frame is divided into $N\times N$-pixel-size macroblocks (MBs) and for each MB a certain area of the reference frame is searched to minimize a block difference measure (BDM), which is usually a sum of absolute differences (SAD) between the current and the reference MB. The displacement within the search area (SA) which gives the minimum BDM value is called a motion vector (MV). MVs together with transformed and quantized block differences (residua) are entropy-coded into the video bitstream. If maximum search range (SR) within the reference frame (RF) is set to d, $(2d + 1)^2$ search points (SPs) have to be evaluated when the full search (FS) strategy is employed. Assuming that typical MB size is 16×16 pixels and SR for CIF (352×288) resolution video is ±32 pixels, the required amount of computation to meet real time 30 frames per second (fps) processing exceeds 35 giga operations per second (GOPS). For HD720 (1280×720) resolution and ±128-pixel SR, this value increases to almost 5 tera operations per second (TOPS). When the multi-view video sequence coding is considered [5], the amount of computation related to ME increases since not only the inter-frame redundancy within a single view has to be taken into account but also the inter-view redundancy between neighbouring views. To decrease such a huge computational burden, within the last three decades many fast BBME algorithms have been proposed. In this paper, selected algorithms belonging to different categories are reviewed and compared in terms of computational complexity and efficiency.

With the development of video coding standards, basic BBME scheme was extended by several additional techniques such as sub-pixel, variable block size, and multiple reference frame ME [3]. Due to the limited space, these techniques are not addressed in this paper.

The rest of the paper is organized as follows: in Sect. 2, the selected algorithms are characterized; in Sect. 3, the comparison based on the results found in the literature is presented; Section 4 gives the conclusion.

2. Characterization of algorithms

In the paper, five types of fast BBME algorithms are distinguished: 1) based on the search position number reduction; 2) multi-resolution; 3) based on the simplification of matching criterion; 4) fast full search; 5) computation-aware. In contrast with classification proposed by Huang *et al*. [6], the usage of spatiotemporal correlations is considered only as a supporting technique not as a separate class of ME algorithms.

2.1. Algorithms based on the search position number reduction

This is the most widespread class of BBME algorithms. It is based on the assumption about the unimodality of the error surface, which means that the BDM value monotonically increases with the distance from the point corresponding to the global minimum. This assumption often does not hold strictly, especially for large SRs, however, sub-optimal results can be obtained.

Jain and Jain introducing BBME concept in Ref. 2, proposed two dimensional logarithmic (2DLOG) search procedure [(Fig. 1(a)]. At each step, four SPs are checked arran-

*e-mail: m.jakubowski@ire.edu.pl

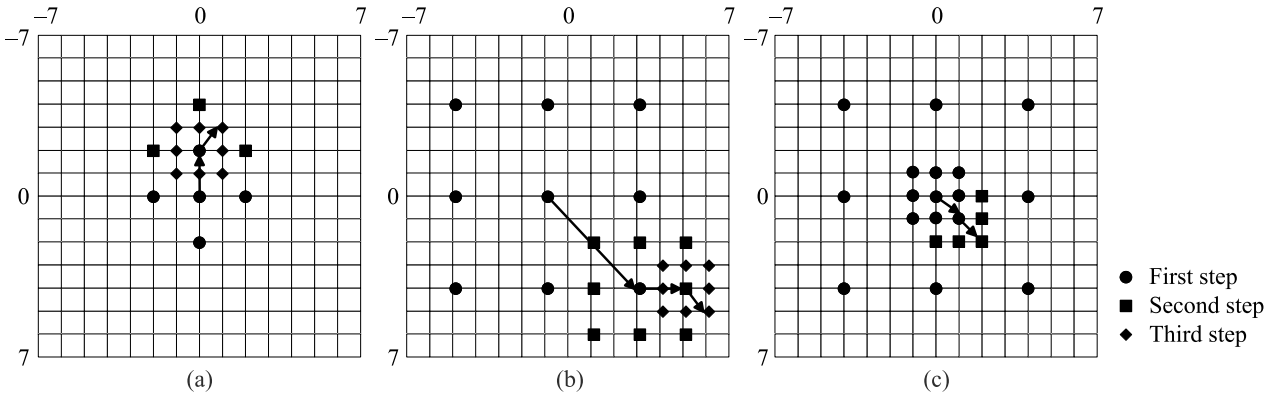

Fig. 1. Example search path of: (a) 2DLOG, (b) TSS, and (c) NTSS algorithm.

ged in the cross (+) pattern around the centre. Initial step size is $d/4$, where d is the maximum displacement. The step size is halved only when the minimum distortion is at the centre of the pattern or at the boundary of the search window. When the step size is equal to 1 – eight points around the centre are checked.

In Ref. 7, Koga *et al*. proposed simple and regular three step search (TSS) which utilizes the square search pattern with eight SPs around the centre at each step [(Fig. 1 (b)]. The initial step size is $d/2$ and is halved in the subsequent steps. When d equals 7, the number of steps is 3 and the number of used SPs is 25. For larger SRs, TSS can be easily extended to n steps with the number of SPs equal to $1 + 8[\log_2(d + 1)]$.

Several modifications of the original TSS scheme have been proposed. In Ref. 8, the set of thresholds is introduced to determine the motion activity of a block and decide whether to perform the next step of TSS or to finish.

New three step search (NTSS) [9], exploits the centre-biased characteristic of the MV distribution in most real-world video sequences. Thus, eight extra SPs around the centre are checked in the first step [(Fig. 1(c)]. Additionally, the algorithm incorporates a half-way stop technique in order to fast identify and estimate the motion of stationary and quasi-stationary blocks. If the minimum BDM value in the first step occurs at the search window centre, the search stops. If the minimum BDM point in the first step is one of the eight SPs surrounding the window centre, the search in the second step will be performed only for eight neighbouring points of the minimum and then stopped. Otherwise, the search goes exactly as for TSS.

Four step search (4SS) [10] uses more compact search pattern than TSS with the initial step size set to $d/4$ [(Fig. 2 (a)]. The step size is maintained constant through steps 1 to 3. Similarly to NTSS, the algorithm incorporates a half-way stop technique. If in the second step, the minimum BDM point is centre located, the search goes to the final step where step size is reduced to 1.

Less known TSS-based schemes include: fast three step search (FTSS) [11], adaptive centre non-linear three step search (ACNTSS) [12] and efficient three step search (E3SS) [13].

One-at-a-time search (OTS) algorithm [14] performs search first in the horizontal direction using three adjacent SPs [(Fig. 2(b)]. When the minimum BDM value is found at the central point, the search direction changes to vertical and the search is continued up to the moment when the minimum BDM value is located at the centre of the search pattern.

Orthogonal search algorithm (OSA) [15] starts with three horizontally arranged SPs: one at the centre and two in $d/2$ distance [(Fig 2(c)]. Next, two vertically arranged SPs are checked around the minimum BDM point, the step size is halved and the procedure is repeated until the step size equals 1. For $d = 7$, the required number of SPs equals 13; for larger displacements: $1 + 4[\log_2(d + 1)]$.

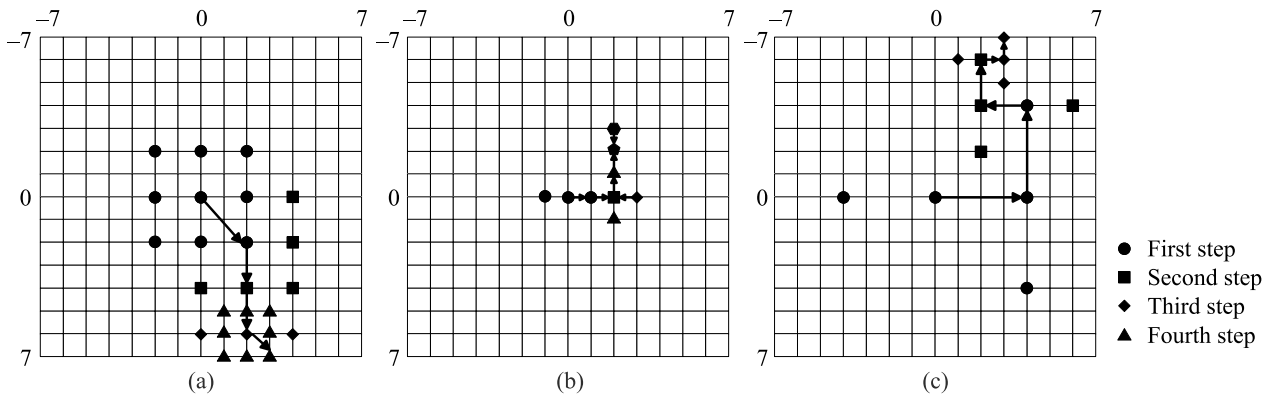

Fig. 2. Example search path of: (a) 4SS, (b) OTS, and (c) OSA algorithm.

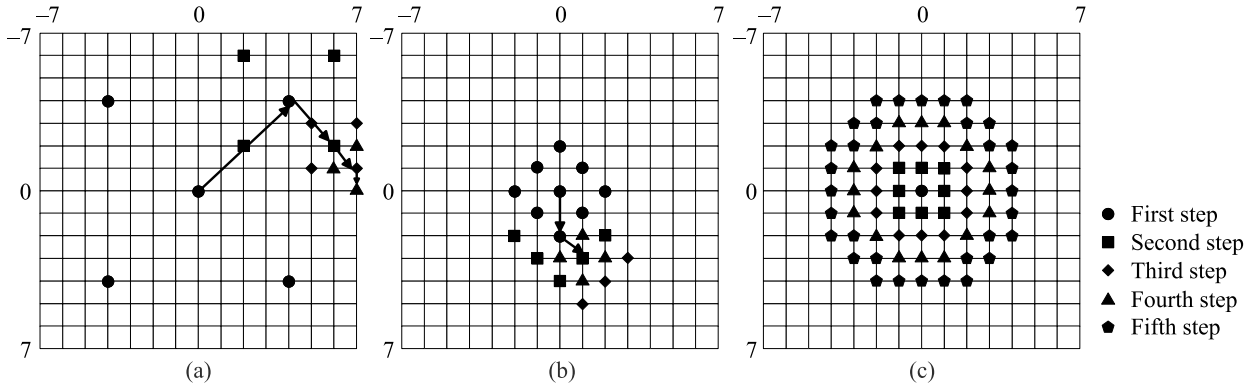

Fig. 3. Example search path of: (a) CSA, (b) DS, and (c) CZS algorithm.

Cross search algorithm (CSA) [16] [(Fig. 3(a)], first checks the zero motion point (ZMP) and compares the BDM value with a predefined threshold to decide whether the block is stationary or not. The search continues only in the second case. In the subsequent steps, four SPs arranged in a (×) cross shape are checked and the step size is halved starting form $d/2$. When the step size reaches the length of one, additional step is carried out with a (+) cross search pattern if the minimum BDM point of the previous step is either the centre, upper left or lower right SP. Otherwise, (×) cross search pattern is used. Number of required SPs (if the stop condition is not met) is $5 + 4[\log_2(d + 1)]$.

Parallel hierarchical one-dimensional search (PHODS) [17], is one of the first ME algorithms designed specifically for hardware implementation. The search along x and y axis is carried out independently and pipeline interleaving the x and y axis searches is possible when the hardware realization is taken into consideration. The algorithm uses the same search pattern as OSA and the distance between SPs is halved at each step as well. As the final solution, MV composed form displacements found along x and y axis is taken.

Dynamic search-window adjustment and interlaced search (DSWA/IS) [18], uses alternatively (×) and (+) cross shape search patterns. The step size at each step is not fixed but determined by the parameter which describes the speed of convergence towards the global minimum. The parameter is calculated as the difference between the smallest and second smallest BDM value found at the present stage of the process and compared with a predefined threshold. At the last step, 3×3 SPs are checked around the up-to-date minimum BDM point.

Block-based gradient descent search (BBGDS) [19], uses 3×3 SPs square search pattern at each step. Number of search steps is unrestricted and the search stops when the minimum BDM point is located at the centre.

Diamond search (DS) [20,21], employs two search patterns: small diamond search pattern (SDSP) and large diamond search pattern (LDSP) composed of five and nine SPs, respectively [(Fig. 3(b)]. First, the search is performed with LDSP until the minimum BDM point is found at the centre of LDSP. Then, at the last step, SDSP is applied.

Several more complex DS-based schemes have been proposed. Cross-diamond search (CDS) [22] incorporates the cross search pattern at the first step to utilize cross-centre-biased MVs distribution observed in many real-world video sequences. Small-cross-diamond search (SCDS) [23] and kite-cross-diamond search (KCDS) [24] are designed mainly for the small-motion activity video sequences such as videoconferencing and utilize the small cross and kite search patterns combined with a half-way stop technique to save more computation.

Adaptive motion tracking block matching described in Ref. 25 utilizes correlation between MVs of adjacent MBs. MVs of four neighbours are used to compute the search centre. First, the difference between each of neighbouring MVs and their mean is computed, and if the maximum value is smaller than a predefined threshold, the minimum difference between neighbouring MV and the mean is taken as the search centre. Otherwise, ZMP is used as the search centre. Then any fast block-matching algorithm, such as TSS or BBGDS, can be applied.

Circular zonal search (CZS) [26] [Fig. 3(c)] and its later modifications: modified circular zonal search (MCZS) [27] and advanced diamond zonal search (ADZS) [28, 29] exploit spatiotemporal correlations among MVs. To determine the search centre, the set of predictors is used including MVs of left, top and top right neighbours and MV of the co-located MB from the previous frame. Other predictors, like neighbouring MVs of the co-located MB are also considered [29]. The search is performed around the search centre using either circular- [26,27] or diamond-shape zones [28,29]. Fixed or adaptive threshold is used to determine whether the up-to-date minimum BDM value can be considered as optimal.

Hilbert scanning search algorithm (HSSA) [30] transfers the 2D space of the search window into a 1D space of Hilbert curve and employs the binary search to locate the global optimal MV. Both one-candidate and multiple-candidate schemes can be used.

1D gradient descent search (1DGDS) [31] first checks ZMP and MVs of four neighbouring MBs to determine the search direction and the step size. If the distance between the initial SP and ZMP is smaller than 3 – the step size is set

to 1, otherwise – to 2. Then, search is performed along a direction pointed by the vector between the origin and the initial MV. Two SPs surrounding the initial one are checked until the minimum is found between them. Then, a conjugate direction is searched in the same way.

Two-path search (2PS) [32] in contrast to the previous algorithms, which take into account only the best candidate at each step, considers the second closest candidate, as well. The 3×3 SPs window is used at each step. If the best BDM point is at the centre of the first step – the search stops. In the second step, two 3×3 sets of SPs are checked around the two best points from the previous step. Then, either TSS is performed around one selected point or the second step is repeated. If MV found after this step is considered as the optimal – the search stops. Otherwise, TSS is employed to find the final MV.

Hexagon-based search (HEXBS) algorithm [33] employs the same search scheme as DS, however, instead of LDSP the large hexagon is used. Later modifications of HEXBS introduce search centre prediction and more detailed inner search of the final large hexagon shape [34–36].

Several BBME algorithms belonging to this category applied different search patterns or basic algorithms depending on the estimated motion activity of a given MB.

Adaptive fast block-matching (A-TDB) [37] combines TSS, DS and BBGDS algorithms to track fast, moderate and slow motion MBs, respectively. The decision about character of the motion is made on the basis of the initial BDM value at ZMP. Then, all the MBs are sorted according to the initial BDM value and divided into three groups (fast, moderate and slow) using predefined thresholds. Switching between the algorithms is threshold-controlled, as well.

Adaptive hierarchical motion estimation (AHMEA) [38] tracks the history of the motion activity of blocks in the frame and classifies them to a low, medium, and high motion activity group according to a predefined threshold. The motion activity is estimated at the frame and the sequence level, as well. This information is used to determine number of predictors used as candidates for the search centre (ZMP, neighbouring MVs, their mean and median), the number of reference frames, and SR. To perform the search, SDSP, LDSP, and switching search pattern are used. Switching search pattern is the (×) cross shape with ±1 range and is used to switch between SDSP and LDSP. The threshold--controlled early-stop technique is adopted.

Motion classification-based search (MCS) [39] employs adaptively several search patterns such as SDSP, LDSP, and exhaustive logarithmic search pattern (ELSP) which is an extension of TSS. Two early-stop criteria are adopted. A linear classifier is used for a pattern selection which bases on the set of selected features such as the median of MVs of neighbouring MBs, ZMP, MV of the co-located block from the previous frame, homogeneity and dispersion of neighbouring MVs, the SAD to cost ratio for the predicted MV, and the Lagrangian multiplier.

2.2. Multiresolution (hierarchical) algorithms

Hierarchical search is based on the idea of searching the initial best match at the coarser level of a sub-sampled SA and using it as the starting point for the refinement at the finer level [40] (see Fig. 4). Since the number of SPs at each level is highly reduced, the computational complexity is substantially lower than in case of one-level search. Usually, two or three search levels are used. The size of a block can be kept constant at each level or scaled according to a sub-sampling ratio. In the first case, an initial estimate is obtained for a larger block at the coarser level. Thus, the estimate is more noise resistant than that obtained for a smaller block. However, since a larger block at the coarser level covers several blocks at the finer level, an initial estimate obtained for a larger block may be inaccurate when each of these blocks moves in a different direction. In the second case, when a block size is scaled along with SA resolution, the more accurate initial estimate can be obtained in the case of complex motion. However, the obtained MV may be less robust to noise.

In Ref. 41, a 3-level hierarchical ME algorithm is proposed. Levels are numbered from 2 to 0, where 2 represents the coarsest and 0 the finest level. At the level 2 and 1, the image is sub-sampled with 16:1 (4:1 in a horizontal and vertical direction) and 4:1 ratio, respectively. One step of TSS is performed to get an initial estimation for the next level. The size of the block is scaled according to the sub-sampling ratio. Apart from sub-sampling, low-pass filtering by simple averaging of pixels at the coarser levels is applied to reduce the noise influence.

Additionally, multiple-candidate scheme at each step can be used to improve the performance.

Fast ME algorithm based on multiresolution-spatio-temporal correlations (MRST) [42] uses MVs of spatially and temporally neighbouring MBs to determine the best candidate to start the process. Four levels of the hierarchy are used, and averaging of 2×2 blocks of pixels is employed to generate coarser levels. At the coarsest level FS is performed. At the subsequent levels fast search is applied with 3×3 search window. Predefined thresholds are used to test early-stop conditions.

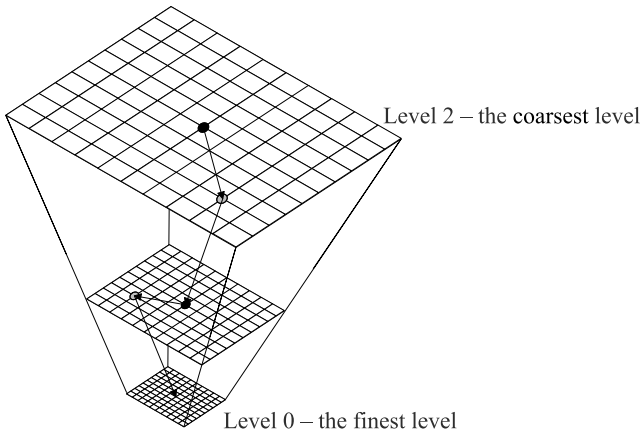

Fig. 4. General idea of multiresolution (hierarchical) search.

Multi-resolution motion search (MRMCS) [43] algorithm and its modification [44] use three levels of the hierarchy with 16:1 and 4:1 sub-sampling at level 2 and 1, respectively. At the coarsest level FS is performed and two best MVs are selected as candidates for the refinement at the level 1 together with the median predictor obtained from neighbouring MVs. At the level 1 the search is performed around these three candidates within ±2-pixel SR. The best MV form the level 1 is used at the level 0 as the search centre for search within ±2-pixel SR. The block size is scaled at each level.

In the hardware-oriented parallel multi-resolution ME (PMRME) algorithm [45] three levels of the hierarchy are searched in parallel with the 4:1 sub-sampling at level 2 and 1. Level 2 and 1 are centred at ZMP to make possible an effective data reuse. Level 0 is centred at the motion vector predictor (MVP) obtained as the median from MVs of spatially-adjacent MBs: left, top and top right. Two MVs with the lowest SAD from each level are taken for the detailed check.

The hardware-oriented multiresolution ME algorithm proposed in Ref. 46 uses three levels of hierarchy, as well. FS algorithm at the level 2 is employed and local FS around multiple candidates (four best MVs and MVP) at the level 1 within ±10-pixel horizontal and ±6-pixel vertical SR. At the level 0, local FS is performed only around the best MV from the previous level within ±14-pixel horizontal and ±8-pixel vertical SR.

2.3. Algorithms based on simplification of matching criterion

In this category of BBME algorithms, two sub-classes can be distinguished. In the first one, the simplification relies on the pixel bit precision reduction. In the second one, only a selected part of current and candidate block pixels is used for the BDM calculation.

2.3.1. Pixel bit precision reduction algorithms

Since the block matching process performed on the low-bit resolution images requires less computation and hardware resources, the original eight-bit resolution of pixels can be reduced by truncating a certain number of the least significant bits or transforming the pixels into a 1-bit representation. Particularly, binary ME algorithms use an exclusive-or (XOR) Boolean block-matching criterion instead of the integer criterion.

In Ref. 47, the binarization of each pixel of SA and the current MB are performed by comparison with the mean pixel value of SA and MB, respectively. Then, FS is performed over the bit-plane search window to select the best candidates using an adaptive threshold. Next, full-resolution BDM is calculated for these best candidates. The algorithm incorporates the starting point prediction by using MVs of neighbouring MBs and the adaptive SR adjustment.

In Ref. 48. a block matching is performed on the binary edge maps obtained from the reference and the current frame. If there are too few edge pixels inside a given MB, MV of the current, MB is interpolated by using neighbouring MVs.

In Ref. 49, the convolution kernel is applied to the current and the reference frame to obtain their filtered versions. Then, filtered and unfiltered frames are compared and if the unfiltered pixel value is larger than the filtered one, value 1 is assigned to the binary version of the frame. Next, any BBME algorithm, such as FS or TSS, can be applied.

The fast binary pyramid ME (FBPME) algorithm [50] combines the hierarchical approach to ME with binary block matching. The 4-level hierarchy is used. Levels from 4 to 2 are constructed by the binarization of the image through filtering, sub-sampling, and up-sampling by interpolation. Then the difference between original and up-sampled version of an image is compared with a threshold to get a binary version. At level 1, the sub-sampled version of the original frame is used. The search starts from the top level with blocks of different sizes (8×8, 8×4, 4×8, and 4×4). The best MVs from each block size are propagated to the next level and refined. At the level 2, four vectors from higher levels are refined but only for the 8×8 block size. The best one is projected to level 1 and refined by using 3×3 search window and 16×16 block size.

To improve the ME accuracy obtained with 1-bit representation, Erturk and Erturk proposed 2-bit transform for binary BBME in Ref. 51. The first and the second bit plane are obtained through comparison of each pixel value with the local mean and the mean plus standard deviation, respectively. The matching criterion is defined by using Boolean XOR and OR operators. It is possible to make XOR comparisons for each bit plane of the current and reference frame in parallel to speed up the computation process.

In Ref. 52, the reduced bit mean absolute difference (RBMAD) is proposed as BDM. With the bit precision reduction from one to seven bits, the results are close to those obtained with full-resolution MAD. The significant hardware resource reduction and the speed-up ratio (57% and 34% with 4-bit RBMAD, respectively) are reported.

In Ref. 53, this idea is extended to adaptive bit-reduced mean absolute difference (ABRMAD), where the number of bits for MAD calculation is selected adaptively according to the position of the first non-zero bit of the pixel with the largest intensity in the current block. The performance of ABRMAD is superior to RBMAD, especially for a low-bit resolution (1 – 3 bits) cases. In Fig. 5, a comparison of pixel bits' selection for 8-bit MAD, 4-bit RBMAD, and 4-bit ARBMAD is presented.

Lee *et al*. [54] proposed the low-resolution quantization motion estimation (LRQME) algorithm. First, the mean of the current block is subtracted from each pixel value of the search window and the block itself. The result is adaptively quantized into a 2-bit code. Then, the search is performed on the low-bit-resolution image and a set of candidate MVs is selected to calculate the full 8-bit-resolution BDM, however, with 4:1 sub-sampling to reduce the computation.

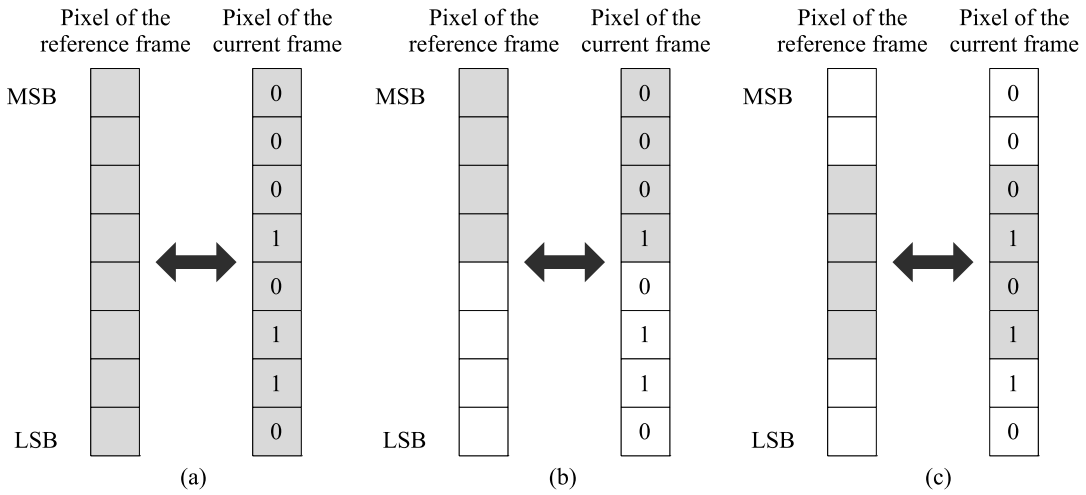

Fig. 5. Comparison of bit selection for calculation: (a) full 8-bit resolution MAD, (b) fixedly reduced 4-bit resolution MAD (RBMAD), and (c) adaptively reduced 4-bit resolution MAD (ARBMAD).

2.3.2. Block sub-sampling algorithms

In order to reduce computations associated with the BDM calculation, various types of sub-sampling can be applied to the current and reference block. In Ref. 40, every second pixel of these blocks is selected in both horizontal and vertical directions.

In Ref. 55, Liu and Zaccarin, instead of the fixed decimation pattern, proposed a set of four patterns with 4:1 sub-sampling which are altered during the search process to reduce the influence of sub-sampling on the ME accuracy (Fig. 6).

Hierarchical partial distortion search (HPDS) algorithm [56] uses hierarchical sub-sampling with three levels of hierarchy. At each level, the number of candidate MVs taken into account decreases and the number of pixels taken for the BDM calculation increases. These two parameters can be adjusted to balance the speed and the accuracy of the ME process.

Globally adaptive pixel-decimation (GAPD) [57] applies Hilbert scan to convert a 2D block into a 1D Hilbert sequence. Adaptive thresholding is employed to select pixels belonging to the edge and highly structured regions which are used for the BDM calculation.

1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4
1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4

Fig. 6. Four alternating 4:1 sub-sampling patterns [55]. Pixels belonging to each pattern are marked with appropriate number.

Pixel decimation scheme proposed in Ref. 58 is based on *N*-queen lattice which is obtained by placing *N* queens on a chessboard. Only pixels at the positions occupied by the queens are selected for the BDM calculation. The scheme guarantees that at least one pixel is selected from each row, column, and diagonal. In Fig. 7, the fixed 4:1 sub-sampling pattern is compared with a 4-queen pattern. In Ref. 59, several decimation patterns are proposed based on the boundary approach (observation that new objects

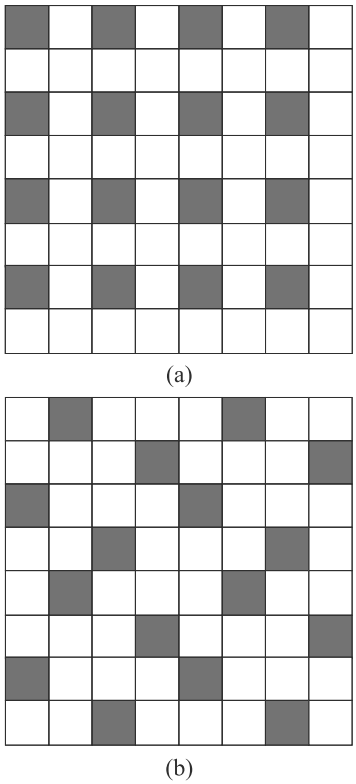

Fig. 7. Comparison of (a) fixed 4:1 pixel decimation pattern (no. 1 in Ref. 55) with (b) 4-queen pattern [58]. Pixels selected for the BDM calculation are grey-coloured.

enter an MB through its boundary regions) and *N*-queen lattices. Genetic algorithms are employed for the optimal pattern selection.

2.4. Fast full-search algorithms

BBME algorithms which belong to this category can obtain exactly the same results as FS with a significant reduction of computational complexity which is achieved by an early detection of non-optimal solution and by skipping unnecessary computations. With an appropriate selection of the search starting point, most of the unnecessary calculations can be avoided.

In the partial distortion elimination (PDE) algorithm [60], recommended for the H.263 standard, the current BDM value is gradually accumulated and the partial distortion is compared with the previously found minimum BDM value. All candidate blocks for which the partial distortion is already larger than the current minimum distortion are skipped. Successive elimination algorithm (SEA) [61] utilizes the reversed triangle inequality for the fast rejection of the most candidate blocks. Particularly, the candidate block is rejected if the absolute difference between the sums of samples within compared blocks is larger than the up-to--date smallest mean absolute difference (MAD). As the initial SP, MV of the co-located MB from the previous frame is proposed.

The fast full search block-matching algorithm described in Ref. 62, prior to calculation of the full BDM value, tests three fast matching error measures based on the reduced representation of compared blocks. The representations are calculated as massive, vertical, and horizontal integral projections of a block. If any of the fast matching criteria holds, further calculations can be skipped and a candidate block discarded.

The algorithm proposed in Ref. 63 can be considered as the combination of PDE and SEA algorithms since the SEA criterion is evaluated not only once for the whole block, but also for its subsampled versions (1×1, 2×2, 4×4, and 8×8). Thus, the rejection at an earlier stage is possible. The same approach is adopted in the multilevel successive elimination algorithm (MSEA) [64].

In Ref. 65, a PDE modification is based on the observation that BDM value is proportional to the gradient magnitude of the reference block. It allows for the early selection of more significant pixels for BDM calculation, which increases the probability of the fast rejection of inappropriate candidates. Similar approach is applied in Ref. 66; however, the pixel selection order is determined by the arrangement of pixel differences calculated at ZMP according to their contribution into the BDM value.

Fine granularity successive elimination algorithm (FGSEA) [67] successively checks for up to 85 conditions (levels) for early rejection of inappropriate candidates. Each level is formed by the incremental computation of the SADs between the parts of the candidate and the current block, from 1×1 to 16×16. To determine the order of the pixel selection, the complexity of each sub-block is estimated by using the gradient magnitude. Additionally, to save computations, the prediction of the starting level is employed, which utilizes information about the rejection level of the preceding block. Other modifications of SEA are: extended SEA (ESEA) [68], extended MSEA (EMSEA) [69] and adaptive MSEA (AdaMSEA) [70].

2.5. Computation-aware algorithms

Computation-aware BBME algorithms [71] employ adaptively different techniques and algorithms to achieve optimal or suboptimal results in a computation-limited and computation-variant environment. Previously described BBME algorithms do not take into account the available computational resources and stop only when all necessary SPs are checked. By this time, the available computational power may be exhausted, making real-time implementation impossible. When the search procedure is interrupted to meet real-time requirements, obtained results may be far from the optimum. On the other hand, when computational resources are abundant, traditional BBME algorithms are not able to utilize the excess efficiently. Computation-aware algorithms can fulfil real-time requirements without any significant performance degradation and improve the performance when additional resources are available.

Tai *et al*. [71] proposed the frame- and block-level computation distribution scheme. The frame-level computation distribution bases on the information about computation usage in the previous frames. The block-level computation allocation is proportional to the BDM value at the initial SP. Then, blocks are arranged in the descending order of BDM value and one step of the algorithm (e.g., TSS) is performed for each block. After first step, blocks are sorted again and the procedure is repeated until the resources are exhausted.

To avoid multiple allocations and sorting, Chen *et al*. [72] proposed one-pass scheme which operates in a block--by-block manner. All computational resources are divided equally between frames and distributed among MBs. Each MB receives the same basic amount of computation and some extra resources according to the ratio of SAD at the initial SP, which is the median of left, upper and right-upper MVs of neighbouring MBs, to average minimum SAD of the previously processed MBs. An adaptive search strategy selection is employed which uses variation of neighbouring MVs around the median to determine the motion activity and selects either DS or TSS algorithm (see Fig. 8). If an early-stop condition is not met and some resources are still available – FS algorithm is invoked. For further speed up, 2:1 subsampling for BDM calculation can be applied. Similar approach is adopted in the multi-path search (MPS) algorithm [73,74], where the initial SP is selected from the prediction set containing MVs of left, left-upper, upper, and right-upper neighbour, ZMP, and the co-located block in the previous frame. Search strategy is selected among TSS, DS and KCDS depending on the estimated block motion activity and the available computation.

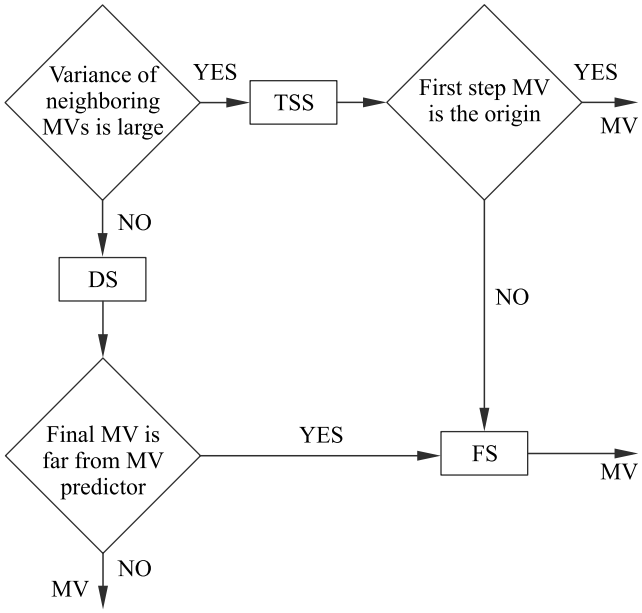

Fig. 8. Adaptive search strategy selection in Ref. 72.

The configurable complexity-bounded motion estimation (CCBME) algorithm [75] can use a different number of predictors (from two to five, including ZMP and four neighbouring MVs) and three types of subsampling for BDM calculation: 4:1, 2:1, and 1:1. When the search centre is determined, a recursive search is performed by using SDSP. Number of search steps can vary from zero to four, which is another parameter of the process. Six best configurations of the parameters are selected on the basis of experiments to trade the complexity for the performance.

Fully-adaptive distance-dependent thresholding search (FADTS) algorithm [76] uses diamond-shaped zones around the search centre to advance the search process. Either the origin or the mean of neighbouring MVs, depending on the correlation between them, is used as the search centre. Number of zones determines the quality and the usage of computation and is controlled by an adaptive threshold calculated every four frames to meet the speed and quality constrains.

2.6. Other approaches to BBME: genetic algorithms, fuzzy logic, Kalman filtering

Some of the BBME algorithms developed through the last three decades do not fit into any of the above-mentioned categories.

In Refs. 77–79, the concept of genetic algorithm is adapted to BBME. Search points are selected randomly from the initial population usually concentrated around the search centre and represented by their binary "chromosomes". The selected group of SPs undergoes the crossover and mutation procedures to produce the next generation of SPs.

In Refs. 80 and 81, the fuzzy-reasoning-based procedure is employed for the initial SP prediction.

In Ref. 82, the 1D and 2D autoregressive model for the initial MV prediction is used. The model is combined with Kalman filtering in order to obtain the final solution.

2.7. Motion and disparity estimation in multi-view video sequences

A multi-view video sequence is composed of several single-view video sequences captured at the same time from different viewpoints. To reduce the inter-frame and the inter-view redundancies present in this type of sequences, a motion and disparity estimation is performed. For this purpose, standard BBME algorithms, described in the previous subsections, can be directly applied [83]. However, to obtain further reduction of computational complexity of the process, inter-view geometrical relationships are often utilized.

In Ref. 84, the epipolar constrain is imposed on the median MVP. Two images of a single scene are related by the epipolar geometry, which can be described by a 3×3 singular matrix called the essential or fundamental matrix, depending on whether the images internal parameters (e.g., the focal length, the coordinates of the principal point) are known or not [85]. According to the epipolar geometry, projection of a a point P on an image plane is constrained to lie on a line called the epipolar line. The authors assume that the horizontal component of MVP is more reliable than the vertical one (since horizontal motion is much more intensive than vertical motion for most typical multi-view camera setups) and lies on the epipolar line. Thus, only the vertical component of MVP requires rectification. When a new MVP is derived, FS is performed within highly reduced SR (±4 pixels). The reported PSNR degradation is below 0.05 dB with about triple speed-up compared to the case when the raw MVP is used with full SR.

In Ref. 83, Lai and Ortega describe disparity than motion (DtM) and motion than disparity (MtD) schemes in which one of the processes (disparity or motion estimation) is performed first to obtain a set of predictors for the second one. At the next stage, the search is performed around the predictors within ±1-pixel SR and around mean of the predictors, within a 4×4 search window. MtD scheme gives better results than DtM. The reported degradation with respect to dual search (when both motion and disparity are estimated separately) is 0.1–0.2 dB and the improvement with reference to simulcast (when each view is coded separately) is about 1.5 dB.

The SR adjustment method proposed in Ref. 86 is based on two MVPs. The first one is the ordinary median predictor obtained from MVs of spatially adjacent blocks. The second one is derived from the geometrical relationships between the views. The search is performed within a reduced SR which is directly proportional to the difference between these two predictors. Obtained results are almost the same as with full SR with about 4-time speed-up.

In Ref. 87, the disparity is estimated first within a reduced SR predicted on the basis of geometrical relationships

between the views. Next, a relation between disparity and motion is utilized to determine SR for motion estimation. Additionally, the early stop condition is provided to limit the number of RFs. The initial RF for a given block is the same as used by the majority of neighbouring blocks. If the Lagrangian cost obtained in this RF is smaller than a certain threshold, this RF is taken as the optimal and other references are not searched. Obtained results are only tenths of dB in PSNR and less than 1% in the bitrate worse than obtained with the scheme implemented in the joint multi--view video model (JMVM) [88] software with over double speed-up.

In Ref. 89, in order to adjust SR, the motion homogeneity is measured as deviation of MVs of 4×4 blocks with respect to their mean. Blocks taken for motion homogeneity estimation come from location of the corresponding MB in the previously coded view obtained by means of a global disparity vector and from its eight neighbouring MBs. Then, obtained value of the motion homogeneity is compared with two predefined thresholds to determine whether MB is located in a homogenous, moderately homogenous, or complex motion region, and SR is adjusted accordingly. The method achieves almost the same results as JMVM with two to five speed-up ratio.

3. Comparison of algorithms

In this section, selected BBME algorithms are compared on the basis of results published in several papers. Since the experiments were conducted with different video sequences, search ranges, block sizes, numbers of frames, and spatial/temporal resolutions, the results are not always consistent. However, some general conclusions can be drawn.

The widest comparison of BBME algorithm belonging to the first category can be found in Refs. 37, 90, and 91. In Ref. 37, TSS, NTSS, 4SS, BBGDS, and DS algorithms are compared by using two 100-frame QCIF (176×144) (“Foreman” and “Mother and daughter”) and four 300-frame SIF (352×240) (“Football”, “Flower garden”, “Table tennis”, and “Mobile calendar”) video sequences. The sequences represent different types of motion activity, from low (“Mother and daughter”), through moderate (“Foreman”, “Flower garden”, “Mobile calendar”), to high (“Football”, “Table tennis”). SR is set to ±7 and ±15 for QCIF and SIF resolution, respectively. The block size is 16×16 and IPPP GOP structure is used. In Table 1, motion-compensated PSNR values and averaged numbers of SPs per MB for each algorithm are presented. It is clear that there is no single algorithm which performs equally well for all the types of sequences. However, DS and BBGDS can be distinguished as combining relatively good performance with low computational complexity. Only for the Foreman sequence these algorithms are distinctly outperformed by TSS, FSS, and NTSS. Foreman is also the only sequence for which the performance of TSS is not the poorest one.

In Ref. 90, NTSS, 4SS, DS, CDS, KCDS, and HEXBS algorithms are compared by using four 200-frame QCIF (“Akiyo”, “Car phone”, “Foreman”, “Stefan”) and three 100-frame CIF (352×288) (“Flower garden”, “Mobile calendar”, “Football”) video sequences. The block size is set to 16×16 and SR to ±7 for all sequences. The comparison in terms of motion-compensated PSNR and an averaged number of SPs per MB is presented in Table 2. In this comparison, CDS and HEXBS appear as the algorithms which maintain a relatively good performance and low computational complexity for all the types of sequences. For high--motion activity sequences (“Stefan”, “Football”), NTSS outperforms the remaining algorithms, however, at the expense of increased computational complexity. KCDS manifests the lowest computational complexity, but for most of the sequences the performance degradation is not acceptable. Only the videoconferencing materials such as “Akiyo” seem to be suitable for this algorithm.

In Ref. 91, 2DLOG, 4SS, BBGDS, DS, CDS, and HEXBS algorithms are compared using ten CIF sequences: “Football”, “Coastguard”, “Akiyo”, “Foreman”, “Sean”,

Table 1. Performance and complexity comparison of FS, TSS, 4SS, NTSS, BBGDS, and DS [37].

Sequence		Algorithm					
		FS	TSS	4SS	NTSS	BBGDS	DS
Flower garden	PSNR [dB]	23.92	21.18	23.39	21.93	23.52	23.61
	SPs	859.45	30.52	19.03	20.54	14.03	16.89
Football	PSNR [dB]	23.08	21.82	22.27	22.02	22.08	22.24
	SPs	859.45	30.52	19.19	21.66	15.15	17.11
Foreman	PSNR [dB]	30.65	30.16	29.83	30.23	29.73	29.83
	SPs	184.56	21.66	16.70	18.37	12.47	14.50
Mobile calendar	PSNR [dB]	22.61	22.31	22.51	22.58	22.58	22.54
	SPs	859.45	30.52	15.88	18.04	10.73	12.60
Mother and daughter	PSNR [dB]	39.71	39.59	39.63	39.65	39.62	39.63
	SPs	184.56	21.66	14.94	15.42	8.43	11.85
Table tennis	PSNR [dB]	27.49	25.82	26.42	26.26	26.25	26.61
	SPs	859.45	30.52	18.31	20.71	12.71	15.54

Table 2. Performance and complexity comparison of FS, NTSS, 4SS, DS, CDS, KCDS, and HEXBS [90].

Sequence		Algorithm						
		FS	NTSS	4SS	DS	CDS	KCDS	HEXBS
Akiyo	PSNR [dB]	44.94	44.94	44.94	44.94	44.94	44.79	44.92
	SPs	184.6	14.68	14.66	13.01	8.222	4.620	9.65
Car phone	PSNR [dB]	33.70	33.63	33.52	33.6	33.55	32.59	33.31
	SPs	184.6	16.75	15.61	14.63	11.00	6.441	10.36
Foreman	PSNR [dB]	32.4	32.29	32.17	32.22	32.14	30.64	31.75
	SPs	184.6	16.88	15.89	14.81	11.09	6.731	10.51
Stefan	PSNR [dB]	25.21	25.02	24.75	24.72	24.67	22.3	24.68
	SPs	184.6	19.28	16.72	16.74	14.62	7.64	11.70
Flower garden	PSNR [dB]	25.63	25.62	25.51	25.61	25.62	18.35	25.52
	SPs	204.3	18.47	17.13	15.28	12.98	6.088	11.53
Mobile calendar	PSNR [dB]	23.89	23.86	23.72	23.81	23.84	22.74	23.64
	SPs	204.3	17.99	16.20	14.07	11.40	7.28	10.70
Football	PSNR [dB]	23.36	22.99	22.79	22.76	22.67	20.9	22.57
	SPs	204.3	20.93	18.40	17.81	15.89	10.20	12.61

"News", "Silent", "Stefan", "Mobile calendar", and "Container". The block size is set to 16×16 and SR to ±16 pixels. The results are shown in Table 3. This comparison confirms efficiency of DS and CDS algorithms. Results obtained by both algorithms are very similar. However, CDS uses less SPs than DS.

For most of the sequences, 4SS exhibits similar performance as DS and CDS or even outperforms them ("Football", "Stefan"), but requires more computational resources. BBGDS is the fastest among the compared algorithms, and its performance is in the most cases comparable with DS and CDS. In general, DS and CDS can be considered as the most versatile and robust algorithms. In the case when the

Table 3. Performance and complexity comparison of FS, 2DLOG, 4SS, HEXBS, BBGDS, CDS, and DS [91].

Sequence		Algorithm						
		FS	2DLOG	4SS	HEXBS	BBGDS	CDS	DS
Football	PSNR [dB]	25.84	24.91	25.06	24.66	24.71	24.83	24.88
	SPs	858.46	33.00	27.48	16.14	20.76	21.50	21.82
Coastguard	PSNR [dB]	30.04	29.40	29.76	29.79	29.81	29.88	29.88
	SPs	868.33	33.00	26.02	13.04	13.72	15.68	16.54
Akiyo	PSNR [dB]	43.04	42.92	42.94	42.82	43.04	43.01	43.03
	SPs	868.33	33.00	23.24	10.34	8.52	8.70	11.29
Foreman	PSNR [dB]	32.84	31.92	32.00	31.42	32.14	32.01	32.10
	SPs	868.33	33.00	26.29	13.71	16.73	16.69	17.93
Sean	PSNR [dB]	39.46	39.31	39.33	39.30	39.45	39.41	39.42
	SPs	868.33	33.00	23.44	10.48	8.92	9.15	11.66
News	PSNR [dB]	37.00	36.61	36.70	36.55	36.66	36.68	36.69
	SPs	868.33	33.00	23.54	10.67	9.25	9.52	11.90
Silent	PSNR [dB]	36.24	35.85	35.88	35.56	35.56	35.57	35.66
	SPs	868.33	33.00	23.76	10.96	9.98	10.14	12.46
Stefan	PSNR [dB]	25.68	24.04	24.62	23.93	23.58	23.95	24.03
	SPs	868.33	33.00	25.94	13.87	15.52	16.96	17.61
Mobile calendar	PSNR [dB]	24.34	23.95	24.03	24.00	24.28	24.25	24.24
	SPs	868.33	33.00	23.70	10.85	10.67	10.34	12.81
Container	PSNR [dB]	38.45	38.45	38.45	38.45	38.45	38.45	38.45
	SPs	868.33	33.00	23.34	10.42	8.58	8.84	11.42

processing speed is a critical factor, BBGDS can be recommended. Additionally, the performance of each algorithm can be improved by introducing the search centre prediction based on spatially and/or temporally adjacent MVs [6].

Several multiresolution (hierarchical) algorithms are compared in Ref. 43, where MRMCS is compared with MRMC using four candidates at each level, MRST and one-level TSS algorithm. Four 300-frame CIF ("Mother and daughter", "News", "Car phone", "Foreman") and three 300-frame SIF ("Flower garden", "Football", "Table tennis") video sequences were used. SR was set to ±16 pixels. In Table 4, the motion-compensated PSNR value and computational complexity (CC), with respect to FS, for each algorithm are presented. In most cases, the computational complexity of multi-resolution algorithms is higher than TSS, however, the PSNR values obtained with TSS are even 1–2 dB lower ("Car phone", "Foreman", "Flower garden", "Table tennis"). MRST seems to be a good compromise between the speed and the performance achieving results only tenths of dB away from FS and MRMCS. Note that both MRST and MRMCS utilize the spatiotemporal correlations between MVs.

Considering the results from Table 1, where TSS is compared with other BBME algorithms based on the search position number reduction, it might be expected that BBGDS and DS can achieve results comparable with multiresolution algorithms with lower computational burden.

ME algorithms based on the reduced pixel bit precision representations of frames are dedicated for customized hardware implementations to reduce the amount of computation and the gate count. For this reason, the speed gain is highly implementation-dependent and only the efficiency in terms of motion-compensated PSNR is compared. In Ref. 53, the adaptive bit-reduced mean absolute difference (ABRMAD) criterion is compared with full 8-bit MAD, binary block matching (BBM) [48], binary block matching with 1-bit transform (1BT) [49], reduced-bit mean absolute difference (RBMAD) [52], and low-resolution quantization motion estimation algorithm (LRQME) [54]. Four CIF video sequences were used in the experiments ("Miss America", "Car phone", "Foreman", "Claire"), 100 frames each with SR set to ±16 pixels. The results in Table 5 show that for all 1- and 2-bit criteria the PSNR drop exceeds 2 dB which is unacceptable value. For RBMAD criterion, 2-bit truncation does not create significant deterioration of the search results. For ARBMAD even four bits can be safely truncated.

In Ref. 92, motion-compensated PSNR obtained with full eight-bit pixel resolution is compared with results for 1BT and 2-bit transform (2BT) [51]. Five CIF video sequences were used in the experiments ("Coastguard", "Mobile calendar", "Football", "Flower garden", and "Bowing") with SR set to ±8 pixels. The results are presented in Table 6. The performance obtained with 1BT and 2BT is similar with the exception of the Bowing sequence for which the 0.8 dB difference is observed. However, the increased computational effort related to 2BT seems to be too high regarding only a moderate improvement. It is characteristic that the biggest discrepancy between the results obtained with the 8- and reduced-bit resolution picture representation is observed for sequences with low motion activity and small amount of details such as "Claire" or "Miss America". In general, truncating more than five least significant bits causes too large performance deterioration. However, from two to five bits truncation can be considered as an efficient way to reduce the hardware complexity and the power consumption of an ME unit.

The comparison of a few pixel decimation schemes can be found in Refs. 57 and 59. In Ref. 57, Hilbert-scan-based globally adaptive pixel-decimation (HSPD) scheme using

Table 4. Performance and complexity comparison of FS, TSS, MRMC, MRST, and MRMCS [43].

Sequence		Algorithm				
		FS	TSS	MRMC	MRST	MRMCS
Mother and daughter	PSNR [dB]	40.10	39.64	40.03	40.05	40.09
	CC [%]	100	3.2	4.6	2.1	4.7
News	PSNR [dB]	36.12	35.70	36.03	35.99	36.11
	CC [%]	100	3.2	4.6	1.9	4.7
Car phone	PSNR [dB]	32.37	31.28	31.96	32.19	32.30
	CC [%]	100	3.2	4.6	4.8	4.7
Foreman	PSNR [dB]	32.53	30.78	32.10	32.35	32.50
	CC [%]	100	3.2	4.6	5.2	4.7
Flower garden	PSNR [dB]	25.26	22.49	24.65	25.11	25.25
	CC [%]	100	3.2	4.6	4.6	4.7
Football	PSNR [dB]	22.74	21.64	22.57	22.55	22.71
	CC [%]	100	3.2	4.6	3.6	4.7
Table tennis	PSNR [dB]	29.75	27.47	29.14	29.53	29.70
	CC [%]	100	3.2	4.6	3.2	4.7

Table 5. Performance comparison of bit precision reduced BDMs: BBM, 1BT, LRQME, RBMAD, and ARBMAD [53].

Algorithm			Sequence			
			Miss America	Car phone	Foreman	Claire
MAD	8b	PSNR [dB]	39.47	29.85	28.39	38.55
BBM	1b	PSNR [dB]	35.31	25.92	25.45	33.99
1BT	1b	PSNR [dB]	35.76	26.51	24.35	31.74
LRQME	2b	PSNR [dB]	35.57	27.37	26.40	35.95
RBMAD	1b	PSNR [dB]	28.37	21.33	21.92	24.36
	2b	PSNR [dB]	33.80	25.98	25.27	29.50
	3b	PSNR [dB]	35.08	28.15	27.36	32.40
	4b	PSNR [dB]	36.39	29.34	28.07	34.00
	5b	PSNR [dB]	37.69	29.59	28.27	36.90
	6b	PSNR [dB]	38.83	29.91	28.32	38.04
	7b	PSNR [dB]	39.31	29.90	28.37	38.43
	8b	PSNR [dB]	39.47	29.85	28.39	38.55
ARBMAD	1b	PSNR [dB]	37.34	27.49	25.90	35.26
	2b	PSNR [dB]	37.89	29.09	27.60	36.53
	3b	PSNR [dB]	38.75	29.58	27.98	36.68
	4b	PSNR [dB]	39.23	29.76	28.26	37.48
	5b	PSNR [dB]	39.37	29.83	28.32	38.07
	6b	PSNR [dB]	39.46	29.83	28.32	38.44
	7b	PSNR [dB]	39.46	29.89	28.37	38.44
	8b	PSNR [dB]	39.47	29.85	28.39	38.55

variable number of pixels is compared with fixed [40] and alternating [55] 4:1 sub-sampling (Table 7). Four CIF sequences were used in the experiments ("Salesman", "Claire", "Football", and "Table tennis") with block size set to 16×16 and SR to ±8. As the complexity measure, the number of pixels taken for the BDM calculation was taken. The fixed subsampling is the most regular among the compared decimation schemes and its performance is the poorest one as well, especially for low-motion activity sequences – "Claire" and "Salesman". Alternating subsampling allows to obtain similar results as HSPD by using comparable number of pixels, however, the second scheme is more flexible and able to maintain good performance for smaller numbers of pixels used for distortion calculation.

In Ref. 59, boundary-based and N-queen decimation schemes are compared with 4- and 8-queen schemes [58]. Four video sequences were used in the experiments: "Container", "Foreman" (QCIF), "Stefan" (CIF), and "Football" (SIF) with the block size and SR set to 16×16 and ±16, respectively. Motion-compensated PSNR and speed-up ratio (SUR) values are shown in Table 8. B4Q4 and BVQ8 are the schemes which combine boundary-based approach with 4- and 8-queen (4Q, 8Q) scheme, respectively. The letters BV denote that the number of horizontal and vertical layers for the boundary-based approach is not identical. This comparison shows that more sophisticated subsampling schemes can achieve similar performance as *N*-queen patterns with a higher speed-up ratio. The performance and the speed-up ratio of BVQ8 seems to be comparable with HSPD, considering the difference in PSNR for the "Football" sequence when HSPD uses 31.3 pixels per block which gives approximately 8.18-time speed-up. This type of simplification of a matching criterion is, in general, less harmful for the performance than the pixel bit precision reduction.

The comparison of several fast full search algorithms can be found in Ref. 67. In the experiments, 11 QCIF and CIF video sequences, 100-frame each were used ("Car pho-

Table 6. Performance comparison of 1BT and 2BT [92].

Algorithm		Sequence				
		Coast guard	Mobile	Football	Flower garden	Bowing
MAD 8b	PSNR [dB]	36.37	35.33	35.30	35.28	42.07
1BT 1b	PSNR [dB]	35.67	35.07	34.96	35.05	39.04
2BT 2b	PSNR [dB]	35.66	35.07	34.97	35.11	39.86

Table 7. Performance and complexity comparison of pixel decimation patterns: fixed 4:1, alternated 4:1, and HSPD with variable number of pixels [58].

Sequence		Algorithm						
		No decimation	Fixed 4:1	Alternating 4:1		HSPD		
Claire	PSNR [dB]	42.31	39.31	42.26	42.25	42.23	42.10	42.00
	Pixels/Block	256	64	64	73.0	61.3	36.7	30.8
Football	PSNR [dB]	23.66	23.52	23.61	23.57	23.56	23.42	23.34
	Pixels/Block	256	64	64	75.0	64.3	36.2	31.3
Salesman	PSNR [dB]	35.43	34.56	35.33	35.37	35.36	35.18	35.16
	Pixels/Block	256	64	64	73.2	62.6	36.5	30.4
Table tennis	PSNR [dB]	28.56	28.42	28.49	28.49	28.48	28.40	28.30
	Pixels/Block	256	64	64	73.4	65.7	36.5	28.6

Table 8. Performance and complexity comparison of pixel decimation patterns: 4-queen (4Q), 8-queen (8Q), B4Q4, and BVQ8 [57].

Sequence		Algorithm				
		No decimation	4Q	8Q	B4Q4	BVQ8
Container	PSNR [dB]	43.09	43.08	43.00	43.08	43.00
	SUR	1	4.0	8.1	6.4	9.2
Foreman	PSNR [dB]	31.84	31.76	31.57	31.65	31.52
	SUR	1	4.0	8.1	6.4	9.2
Stefan	PSNR [dB]	25.90	25.82	25.62	25.69	25.54
	SUR	1	4.0	8.1	6.4	9.2
Football	PSNR [dB]	22.88	22.72	22.41	22.47	22.26
	SUR	1	4.0	8.1	6.4	9.2

ne", "Coastguard", "Flower garden", "Foreman", "Grandmother", "Hall monitor", "Miss America", "Mother and daughter", "Stefan", "Suzie", and "Table tennis"). SEA [61], MSEA [63,64], and FGSE [67] algorithm performances were compared. The block size and SR were set to 16×16 and ±16, respectively.

Since the performance in terms of PSNR of all the algorithms is the same as FS, only the speed-up ratios are presented in Table 9. It shows that the speed gain for this type of algorithms can be even higher than for some search position reduction algorithms such as TSS. However, it is highly content-dependent. For instance, speed-up of FGSE can vary from 6.3 ("Table tennis") to 43.7 ("Car phone"). Thus, this type of fast BBME algorithms can be a good choice for quality-oriented video compression systems.

The performance of computation-aware BBME algorithms is usually visualized in the form of charts such as in Fig. 9, where changes of quality, measured by MSE or PSNR, are presented as the function of available computational resources. To simplify the comparison, the performance of selected algorithms is presented only at selected operating points which characterize an algorithm capability to converge to the optimal solution. In Ref. 73, the computation-aware multi-path search (MPS) algorithm is compared with Chen's one-pass scheme [72] (OPS) and three search position reduction algorithms: NTSS, DS, and HEXBS. Since except MPS and OPS the rest of the algorithms do not provide computational scalability feature, their work is interrupted after reaching a targeted number of SPs. The results for target number of 10 and 15 SPs per MB are presented in Table 10. Five CIF sequences were used for the experiments, 90 frames each: "Coastguard", "Football", "Foreman", "Paris", and "Stefan", the block size and SR were set to 16×16 and ±15 pixels, respectively.

Table 9. Speed-up ratios of fast full search algorithms: SEA, MSEA, and FGSE [67].

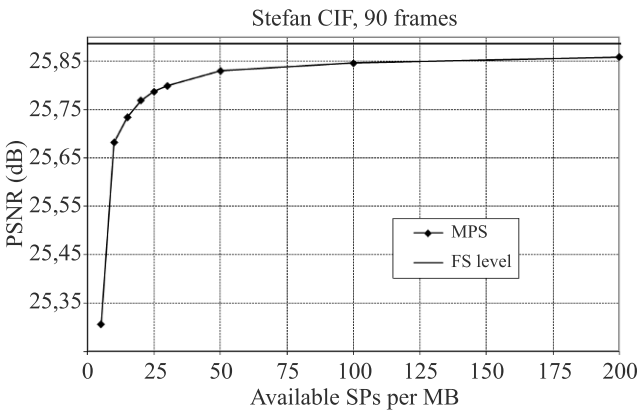

Fig. 9. Performance changes along with available computational resources for MPS algorithm [73].

Sequence		Algorithm		
		SEA	MSEA	FGSE
Car phone	Speed up	3.9	32.5	43.7
Coastguard	Speed up	2.8	24.6	34.3
Flower garden	Speed up	3.2	6.1	8.9
Foreman	Speed up	4.5	18.9	27.0
Grandmother	Speed up	4.6	20.0	22.8
Hall monitor	Speed up	4.2	12.0	19.5
Miss America	Speed up	3.9	10.4	17.4
Mother and daughter	Speed up	7.2	24.9	35.7
Stefan	Speed up	2.7	15.7	22.2
Suzie	Speed up	3.3	5.8	10.2
Table tennis	Speed up	1.8	3.6	6.3

It is clear that computation-aware algorithms perform significantly better for all types of sequences and amounts of available resources than the rest of algorithms. Particularly, for high-motion activity sequences, such as "Football" and "Stefan", the difference at 10 SPs/MB is between 3 to 4 dB. Also, the speed of convergence to the full-search limit is the highest. Even at 10 SPs/MB the difference seldom exceeds 1 dB ("Football"). Good performance and flexibility is the result of starting SP prediction using spatiotemporal correlations between MVs, adaptive search strategy selection according to the available number of SPs, and estimated motion activity within a sequence. Note that among regular algorithms, HEXBS demonstrates the highest speed of convergence. It results from the large hexagon search pattern used by HEXBS, which is slightly wider than LDSP.

In Ref. 75, configurable complexity-bounded motion estimation algorithm (CCBME) is compared with DS and HEXBS. The tests were performed using three CIF video sequences: "Coastguard", "Foreman", and "News". The block size and SR were set to 16×16 and ±16, respectively. The authors define complexity measurement unit (CCMU) to determine the amount of used computation which can be regarded as equivalent to one SP. For DS and HEXBS, only the final results are presented in Table 11. For CCBME, the first and the final operating points are selected for the comparison. As in the previous comparison, it can be observed that CCBME uses less SPs than regular algorithms and achieves better results in terms of PSNR. Computation--aware algorithms seem to be very attractive, especially for the systems operating in a computation-constrained environment.

From all above comparisons, the general observation is that BBME algorithms tend to evolve from simple and regular strategies, such as FS and TSS, to more complex and sophisticated, such as MPS and CCBME. This trend is driven by an increasing demand for performance and supported by rapid progress of silicon technology. The possibility of integration of hundreds of millions of elements in one silicon chip allows for implementations of complex ME modules with unpredictable data flow and irregular memory access. In practice, the BBME algorithm selection is highly application-dependent.

Table 10. MPS, OPS, NTSS, DS, and HEXBS performance at 10 and 15 SPs per MB [73].

Sequence		Algorithm					
		MPS	OPS	NTSS	DS	HEXBS	FS limit
Coastguard	PSNR@10 SPs/MB [dB]	29.64	29.11	26.62	28.45	28.37	29.73
	PSNR@15 SPs/MB [dB]	29.68	29.52	27.02	28.84	29.14	
Football	PSNR@10 SPs/MB [dB]	25.76	24.04	22.11	22.34	22.91	27.29
	PSNR@15 SPs/MB [dB]	26.32	25.34	23.71	23.31	24.25	
Foreman	PSNR@10 SPs/MB [dB]	33.07	32.26	31.02	31.09	31.16	33.52
	PSNR@15 SPs/MB [dB]	33.25	33.09	31.52	32.39	32.20	
Paris	PSNR@10 SPs/MB [dB]	30.36	29.81	29.54	29.29	29.01	30.70
	PSNR@15 SPs/MB [dB]	30.56	30.46	29.66	30.08	30.20	
Stefan	PSNR@10 SPs/MB [dB]	25.68	24.60	21.35	22.98	23.10	25.89
	PSNR@15 SPs/MB [dB]	25.73	25.27	21.93	23.74	23.99	

Table 11. Performance comparison of CCBME, DS, and HEXBS [75].

Sequence		Algorithm			
		CCBME		DS	HEXBS
Coastguard	PSNR [dB]	29.72@2.6 SPs/MB	30.17@10.4 SPs/MB	29.95@15.1 SPs/MB	29.84@13.0 SPs/MB
Foreman	PSNR [dB]	29.74@2.8 SPs/MB	31.01@11.2 SPs/MB	30.07@17.7 SPs/MB	29.79@13.5 SPs/MB
News	PSNR [dB]	33.48@2.2 SPs/MB	34.35@9.1 SPs/MB	34.33@12.6 SPs/MB	34.27@10.6 SPs/MB

For instance, KCDS is well suited for estimating motion in videoconferencing materials but performs poorly for high-motion activity sequences. TSS is simple and regular but its performance is mediocre in comparison with newer algorithms such as DS. Multiresolution techniques are very attractive for applications with large frame sizes and search ranges. Algorithms based on the pixel bit precision reduction are dedicated mainly to custom hardware architectures. Fast full search algorithms seem to be most useful for quality-oriented compression systems where stable processing time is not a critical factor. Computation-aware algorithms are most flexible but require more sophisticated hardware architectures.

4. Conclusions

Motion estimation usually consumes the most computation in a video coding system. In this paper, many fast BBME algorithms belonging to different categories are characterized. Performance and computational complexity of selected algorithms is compared to facilitate the choice of an appropriate algorithm to a specific application.

Acknowledgements

This work was supported in part by the Polish National Centre for Research and Development under Grant LIDER/05/8/L-2/10/NCBiR/2011.

References

1. F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV – a review and a new contribution", *Proc. IEEE* **83**, 858–876 (1995).
2. J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe image-coding" *IEEE T. Commun.* **29**, 1799–1808 (1981).
3. I.E.G. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*, edited by John Wiley & Sons, Chichester, 2002.
4. J.-B. Lee and H. Kalva, *The VC-1 and H.264 Video Compression Standards for Broadband Video Services*, Springer Science+Business Media, New York, 2008.
5. M. Flierl and B. Girod, "Multiview video compression – exploiting inter-image similarities", *IEEE Signal Proc. Mag.* **24**, 66–76 (2007).
6. Y.W. Huang, C.Y. Chen, C.H. Tsai, C.F. Shen, and L.G. Chen, "Survey on block matching motion estimation algorithms and architectures with new results", *J. VLSI Sig. Proc. Syst.* **42**, 297–320 (2006).
7. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing", *Proc. Nat. Telecom. Conf.*, pp. G.5.3.1–G.5.3.5, New Orleans, 1981.
8. S. Kwatra, L. Chow-Ming, and W. Whyte, "An adaptive algorithm for motion compensated color image coding", *IEEE T. Commun.* **35**(7), 747–754 (1987).
9. L. Reoxiang, Z. Bing, and M.L. Liou, "A new three-step search algorithm for block motion estimation", *IEEE T. Circ. Syst. Vid.* **4**, 438–442 (1994).
10. L.M. Po and W.C. Ma, "A novel four-step search algorithm for fast block motion estimation", *IEEE T. Circ. Syst. Vid.* **6,** 313–317 (1996).
11. J.N. Kim and T.S. Choi, "A fast three-step search algorithm with minimum checking points using unimodal error surface assumption", *IEEE T. Consum. Electr.* **44**, 638–648 (1998).
12. H.Y. Chung, P.Y.S. Cheung, and N.H.C. Yung, "Adaptive search centre non-linear three step search", *Proc. Int. Conf. on Image Processing*, pp.191–194, Chicago, 1998.
13. X. Jingand and L.P. Chau, "An efficient three-step search algorithm for block motion estimation", *IEEE T. Multimedia* **6**, 435–438 (2004).
14. R. Srinivasan and K.R. Rao, "Predictive coding based on efficient motion estimation", *IEEE T. Commun.* **33**, 888–896 (1985).
15. A. Puri, H.M. Hang, and D. Schilling, "An efficient block-matching algorithm for motion-compensated coding", *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing* **12**, pp. 1063–1066, Dallas, 1987.
16. M. Ghanbari, "The cross-search algorithm for motion estimation", *IEEE T. Commun.* **38**, 950–953 (1990).
17. L.G. Chen, W.T. Chen, Y.S. Jehng, and T.D. Chiueh, "An efficient parallel motion estimation algorithm for digital image processing", *IEEE T. Circ. Syst. Vid.* **1**, 378–385 (1991).
18. L.W. Lee, J.F. Wang, J.Y. Lee, and J.D. Shie, "Dynamic search-window adjustment and interlaced search for block-matching algorithm", *IEEE T. Circ. Syst. Vid.* **3**, 85–87 (1993).
19. L.K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding", *IEEE T. Circ. Syst. Vid.* **6**, 419–422 (1996).
20. S. Zhu and K.K. Ma, "A new diamond search algorithm for fast block matching motion estimation", *J. VLSI Sig. Proc. Syst.*, pp. 292–296, Singapore, 1997.
21. J.Y. Tham, S. Ranganath, M. Ranganath, and A.a. Kassim,"A novel unrestricted centre-biased diamond search algorithm for block motion estimation", *IEEE T. Circ. Syst. Vid.* **8**, 369–377 (1998).
22. C.H. Cheung and L.M. Po, "A novel cross-diamond search algorithm for fast block motion estimation" *IEEE T. Circ. Syst. Vid.* **12**, 1168–1177 (2002).
23. C.H. Cheung and L.M. Po, "A novel small-cross-diamond search algorithm for fast video coding and videoconferencing applications", *Proc. IEEE Int. Conf. Image Processing* **I**, pp. 681–684, Rochester, 2002.
24. C.W. Lam, L.M. Po, and C.H. Cheung, "A novel kite-cross-diamond search algorithm for fast block matching motion estimation", *IEEE Int. Symp. on Circuits and Systems* **3**, pp. 729–732, Vancouver, 2004.
25. J.B. Xu, L.M. Po, and C.K. Cheung, "Adaptive motion tracking block matching algorithms for video coding" *IEEE T. Circ. Syst. Vid.* **9**, 1025–1029 (1999).
26. A.M. Tourapis, O.C. Au, and M.L. Liou, "Fast motion estimation using circular zonal search", *Proc. SPIE Symp. Visual Comm. and Image Processing* **3653**, pp. 1496–1504, 1999.
27. A.M. Tourapis and O.C. Au, "Fast motion estimation using modified circular zonal search", *Proc. IEEE Int. Symp. Circuits and Systems* **4**, pp. 231–234, Orlando, 1999.

28. A.M. Tourapis, O.C. Au, M.L. Liou, and G.B. Shen, "Fast and efficient motion estimation using diamond zonal-based algorithms", *Circ. Syst. Signal Pr.* **20**, 23–251 (2001).
29. A.M. Tourapis, O.C. Au, and M.L. Liou, "Highly efficient predictive zonal algorithms for fast block-matching motion estimation", *IEEE T. Circ. Syst. Vid.* **12**, 934–947 (2002).
30. Y.K. Wang and H. Kuroda, "Hilbert scanning search algorithm for motion estimation", *IEEE T. Circ. Syst. Vid.* **9**, 683–691 (1999).
31. O.T.C. Chen, "Motion estimation using a one-dimensional gradient descent search", *IEEE T. Circ. Syst. Vid.* **10**, 608–616 (2000).
32. C.C. Chang, L.L. Chen, and T.S. Chen, "A new two-path search algorithm for block motion estimation of video data", *J. Inf. Sci. Eng.* **17**, 405–415 (2001).
33. C. Zhu, X. Lin, and L.P. Chau, "Hexagon-based search pattern for fast block motion estimation" *IEEE T. Circ. Syst. Vid.* **12**, 349–355 (2002).
34. C. Zhu, X. Lin, L.P. Chau, and L.M. Po, "Enhanced hexagonal search for fast block motion estimation", *IEEE T. Circ. Syst. Vid.* **14**, 1210–214 (2004).
35. L.M. Po, C.W. Ting, K.M. Wong, and K.H. Ng, "Novel point-oriented inner searches for fast block motion estimation", *IEEE T. Multimedia* **9**, 9–15 (2007).
36. B.J. Zou, C. Shi, C.H. Xu, and S. Chen, "Enhanced hexagonal-based search using direction-oriented inner search for motion estimation", *IEEE T. Circ. Syst. Vid.* **20**, 156–160 (2010).
37. S.Y. Huang, C.Y. Cho, and J.S. Wang, "Adaptive fast block-matching algorithm by switching search patterns for sequences with wide-range motion content", *IEEE T. Circ. Syst. Vid.* **15**, 1373–1384 (2005).
38. Y.F. Liang, I. Ahmad, J.C. Luo, Y. Sun, and V. Swaminathan, "On using hierarchical motion history for motion estimation in H.264/AVC", *IEEE T. Circ. Syst. Vid.* **15**, 1594–1603 (2005).
39. I. Gonzalez-Diaz, F. Diaz-De-Maria, "Adaptive multipattern fast block-matching algorithm based on motion classification techniques", *IEEE T. Circ. Syst. Vid.* **18**, 1369–1382 (2008).
40. M. Bierling, "Displacement estimation by hierarchical block matching", *SPIE Conf. Visual Communications and Image Processing*, pp. 942–951, Cambridge, MA, 1988.
41. K.M. Nam, J.S. Kim, R.H. Park, and Y.S. Shim, "A fast hierarchical motion vector estimation algorithm using mean pyramid", *IEEE T. Circ. Syst. Vid.* **5**, 344–351 (1995).
42. J. Chalidabhongse and C.C.J. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations" *IEEE T. Circ. Syst. Vid.* **7**, 477–488 (1997).
43. J.H. Lee, K.W. Lim, B.C. Song, and J.B. Ra, "A fast multi-resolution block matching algorithm and its LSI architecture for low bit-rate video coding", *IEEE T. Circ. Syst. Vid.* **11**, 1289–1301 (2001).
44. B.C. Song and K.W. Chun, "Multi-resolution block matching algorithm and its VLSI architecture for fast motion estimation in an MPEG-2 video encoder" *IEEE T. Circ. Syst. Vid.* **14**, 1119–137 (2004).
45. C.C. Lin, Y.K. Lin, and T.S. Chang, "PMRME: A parallel multi-resolution motion estimation algorithm and architecture for HDTV sized H.264 video coding", *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing* **II**, pp. 385–388, Honolulu, 2007.
46. H.B. Yin, H.Z. Jia, H.G. Qi, X.H. Ji, X.D. Xie, and W. Gao, "A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders", *IEEE T. Circ. Syst. Vid.* **20**, 1242–1254 (2010).
47. J. Feng, K.T. Lo, H. Mchrpor, and A.E. Karbowiak, "Adaptive block matching motion estimation algorithm using bit-plane matching", *Proc. IEEE Int. Conf. Image Processing*, pp. C496-C499, Washington, 1995.
48. M.M. Mizuki, "A binary block matching architecture with reduced power consumption and silicon area requirement", *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 3248–3251, Atlanta, 1996.
49. B. Natarajan, V. Bhaskaran, and I. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms", *IEEE T. Circ. Syst. Vid.* **7**, 702–706 (1997).
50. X.D. Song, T.H. Chiang, X. Lee, and Y.Q. Zhang, "New fast binary pyramid motion estimation for MPEG2 and HDTV encoding", *IEEE T. Circ. Syst. Vid.* **10**, 1015–1028 (2000).
51. A. Erturk and S. Erturk, "Two-bit transform for binary block motion estimation", *IEEE T. Circ. Syst. Vid.* **15**, 938–946 (2005).
52. Y.J. Baek, H.S. Oh, and H.K. Lee, "Block-matching criterion for efficient VLSI implementation of motion estimation", *Electron. Lett.* **32**, 1184–1185 (1996).
53. H.S. Oh, Y. Baek, and H.K. Lee, "Adaptive bit-reduced mean absolute difference criterion for block-matching algorithm and its VLSI design", *Opt. Eng.* **37**, 3272–3281 (1998).
54. S. Lee, J.M. Kim, and S.I. Chae, "New motion estimation algorithm using adaptively quantized low bit resolution image and its VLSI architecture for MPEG2 video encoding", *IEEE T. Circ. Syst. Vid.* **8**, 734–744 (1998).
55. B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors", *IEEE T. Circ. Syst. Vid.* **3**, 148–157 (1993).
56. C.K. Cheung and L.M. Po, "A hierarchical block matching algorithm using partial distortion measure", *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 1237–1240, Hong Kong, 1997.
57. Y.K. Wang, Y.Q. Wang, and H. Kuroda, "A globally adaptive pixel-decimation algorithm for block-motion estimation", *IEEE T. Circ. Syst. Vid.* **10**, 1006–1011 (2000).
58. C.N. Wang, S.W. Yang, C.M. Liu, and T.H. Chiang, "A hierarchical decimation lattice based on N-queen with an application for motion estimation", *IEEE Signal Proc. Let.* **10**, 228–231 (2003).
59. S. Sural, A. Saha, and J. Mukherjee, "New pixel-decimation patterns for block matching in motion estimation", *Signal Process. Image* **23**, 725–738 (2008).
60. *ITU-T recommendation H.263 software implementation,* Digital Video Coding Group, Telenor Research and Development, 1995.
61. W. Li and E. Salari, "Successive elimination algorithm for motion estimation", *IEEE T. Image Process.* **4**, 105–107 (1995).
62. Y.C. Lin and S.C. Tai, "Fast full-search block-matching algorithm for motion-compensated video compression", *IEEE T. Commun.* **45**, 527–531 (1997).
63. J.Y. Lu, K.S. Wu, and J.C. Lin, "Fast full search in motion estimation by hierarchical use of Minkowski's inequality (HUMI)", *Pattern Recogn.* **31**, 945–952 (1998).

64. X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation", *IEEE T. Image Process.* **9**, 501–504 (2000).
65. J.N. Kim and T.S. Choi, "A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan", *IEEE T. Circ. Syst. Vid.* **10**, 1040–1048 (2000).
66. W.G. Hong and T.M. Oh, "Sorting-based partial distortion search algorithm for motion estimation", *Electron. Lett.* **40**, 113–115 (2004).
67. C. Zhu, W.S. Qi, and W. Ser, "Predictive fine granularity successive elimination for fast optimal block-matching motion estimation", *IEEE T. Image Process.* **14**, 213–221 (2005).
68. M. Brunig and W. Niehsen, "Fast full-search block matching", *IEEE T. Circ. Syst. Vid.* **11**, 241–247 (2001).
69. T.G. Ahn, Y.H. Moon, and J.H. Kim, "Fast full-search motion estimation based on multilevel successive elimination algorithm", *IEEE T. Circ. Syst. Vid.* **14**, 1265–1269 (2004).
70. S.W. Liu, S.D. Wei, and S.H. Lai, "Fast optimal motion estimation based on gradient-based adaptive multilevel successive elimination", *IEEE T. Circ. Syst. Vid.* **18**, 263–267 (2008).
71. P.L. Tai, S.Y. Huang, C.T. Liu, and J.S. Wang, "Computation-aware scheme for software-based block motion estimation", *IEEE T. Circ. Syst. Vid.* **13**, 901–913 (2003).
72. C.Y. Chen Y.W. Huang, C.L. Lee, and L.G. Chen, "One-pass computation-aware motion estimation with adaptive search strategy", *IEEE T. Multimedia* **8**, 698–706 (2006).
73. M. Jakubowski and G. Pastuszak, "A new multi-path scheme for adaptive computation-aware motion estimation", *Opto--Electron. Rev.* **15**, 118–124 (2007).
74. M. Jakubowski and G. Pastuszak, "An adaptive computation-aware algorithm for multi-frame variable block-size motion estimation in H.264/AVC", *Int. Conf. on Signal Processing and Multimedia Applications*, pp. 122–125, Milan, 2009.
75. Z. Yang, J.J. Bu, C. Chen, and L.J. Mo, "Configurable complexity-bounded motion estimation for real-time video encoding", in *Advanced Concepts for Intelligent Vision Systems*, **3708**, pp. 555–562, Antwerp, 2005.
76. G. Sorwar, M. Murshed, and L.S. Dooley, "A fully adaptive distance-dependent thresholding search (FADTS) algorithm for performance-management motion estimation", *IEEE T. Circ. Syst. Vid.* **17**, 429–440 (2007).
77. K.H.K. Chow and M.L. Liou, "Genetic motion search algorithm for video compression", *IEEE T. Circ. Syst. Vid.* **3**, 440–445 (1993).
78. I.K. Kim and R.H. Park, "Block matching algorithm using a genetic algorithm", *Proc. IEEE Visual Communications and Image Processing* **2501**, pp. 1545–1552, Taipei, 1995.
79. C.H. Lin and J.L. Wu, "A lightweight genetic block-matching algorithm for video coding", *IEEE T. Circ. Syst. Vid.* **8**, 386–392 (1998).
80. Y.T. Roan and P.Y. Chen, "A fuzzy search algorithm for the estimation of motion vectors", *IEEE T. Broadcast.* **46**, 121–127 (2000).
81. P.Y. Chen and M. Jou, "An efficient blocking-matching algorithm based on fuzzy reasoning", *IEEE T. Syst. Man Cy.* **B31**, 253–259 (2001).
82. C.M. Kuo, C.H. Hsieh, Y.D. Jou, H.C. Lin, and P.C. Lu, "Motion estimation for video compression using Kalman filtering", *IEEE T. Broadcast.* **42**, 110–116 (1996).
83. P.L. Lai and A. Ortega, "Predictive fast motion/disparity search for multiview video coding", *Proc. IEEE Visual Communications and Image Processing* **6077**, pp. 7709–7709, San Jose, 2006.
84. J.B. Lu, C.T. Hua, J.G. Lou, and J. Li, "An effective epipolar geometry assisted motion estimation technique for multi--view image and video coding", *Proc. IEEE Int. Conf. Image Processing*, pp. 1089–1092, Atlanta, 2006.
85. Z.Y. Zhang, "Determining the epipolar geometry and its uncertainty: A review", *Int. J. Comput. Vision*, **27**, 161–195 (1998).
86. Y. Kim, J. Kim, and K. Sohn, "Fast disparity and motion estimation for mufti-view video coding", *IEEE T. Consum. Electr.* **53**, 712–719 (2007).
87. X.M. Li, D.B. Zhao, S.W. Ma, and W. Gao, "Fast disparity and motion estimation based on correlations for multiview video coding", *IEEE T. Consum. Electr.* **54**, 2037–2044 (2008).
88. A. Vetro, Y. Su, H. Kimata, and A. Smolic, "Joint multiview video model JMVM 2.0.", *ITU-T and ISO/IEC Joint Video Team Document JVT-U207* (http://ftp3.itu.int/av-arch/ jvt--site/2006_10_Hangzhou/JVT-U207.zip), 2006.
89. L.Q. Shen, G.R. Feng, Z. Liu, Z.Y. Zhang, and P. An, "Macroblock-level adaptive search range algorithm for motion estimation in multiview video coding", *J. Electron. Imaging* **18**, (2009).
90. X.Q. Yi and N. Ling, "Rapid block-matching motion estimation using modified diamond search algorithm", *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 5489–5492, Kobe, 2005.
91. K.H. Ng, L.M. Po, and K.M. Wong, "Search patterns switching for motion estimation using rate of error descent", *Proc. IEEE Int. Conf. Multimedia and Expo*, pp. 1583–1586, Barcelona, 2007.
92. A. Vlachos and V. Fotopoulos, and A.N. Skodras, "Low bit depth representation motion estimation algorithms: a comparative study", *J. Real-Time Image Proc.* **5**, 141–148 (2010).

