Circuits, Systems, and Signal Processing
https://doi.org/10.1007/s00034-024-02790-3

Efficient Block Matching Motion Estimation Using Variable-Size Blocks and Predictive Tools

Milad Mirjalili[1] · **Amir Mousavinia**[2]

Received: 13 July 2023 / Revised: 8 July 2024 / Accepted: 8 July 2024

Abstract
In this research paper, we introduce an adaptive block-matching motion estimation algorithm to improve the accuracy and efficiency of motion estimation (ME). First, we present a block generation system that creates blocks of varying sizes based on the detected motion location. Second, we incorporate predictive tools such as early termination and variable window size to optimize our block-matching algorithm. Furthermore, we propose two distinct search patterns to achieve maximum quality and efficiency. We evaluated the proposed algorithms on 20 videos and compared the results with known algorithms, including the full search algorithm (FSA), which is a benchmark for ME accuracy. Our proposed quality-based algorithm shows an improvement of 0.27 dB in peak signal-to-noise ratio (PSNR) on average for reconstructed frames compared to FSA, along with a reduction of 71.66% in searched blocks. Similarly, our proposed efficiency-based method results in a 0.07 dB increase in PSNR and a 97.93% reduction in searched blocks compared to FSA. These findings suggest that our proposed method has the potential to improve the performance of ME in video coding.

Keywords Variable-size block matching · Adaptive block matching motion estimation · Video compression · Motion estimation · Peak signal-to-noise ratio (PSNR)

✉ Milad Mirjalili
mirjalili.m@email.kntu.ac.ir

Amir Mousavinia
moosavie@kntu.ac.ir

[1] Department of Electrical Engineering, K. N. Toosi University of Technology, Tehran, Iran

[2] Department of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran

Published online: 01 August 2024

1 Introduction

Video compression is a crucial process that aims to reduce the amount of data required to store and transmit videos. When sending video data, it is more efficient to send frame differences or residuals rather than sending the entire frames. Therefore, reducing temporal redundancy between frames is critical in optimizing video coding efficiency [28, 33]. Exploiting temporal redundancies between frames can be accomplished by leveraging the information that similar pictures often appear in close time intervals, particularly in frames with limited motion. By estimating the motion between two consecutive frames, it is possible to create a motion-compensated image by shifting the second frame by an amount equivalent to the detected displacement. This process enhances the resemblance between the frames, resulting in a reduction in the amount of data required to encode them. With the increasing trend towards high-definition (HD) videos, the need for an efficient ME technique has become critical in order to reduce the excessive bandwidth required to transfer such videos. Two main methods for ME are pixel-based and block-based. In pixel-based ME, motion vectors (MVs) are computed for each pixel using brightness and smoothness constraints [7, 35]. This method provides detailed pixel-level motion information, making it ideal for scenarios where precise motion tracking is required, such as in object tracking. However, pixel-based methods can be computationally intensive, especially for HD videos [5]. On the other hand, the block-matching algorithm (BMA) is a popular ME technique known for its ability to provide high-quality motion estimation while keeping computational complexity low [8, 9]. The conventional approach for BMA involves several steps: (1) dividing the current frame into non-overlapping blocks; (2) searching for the most similar block in the reference frame for each block within a defined search window; and (3) calculating the MV that points from the corresponding block to the best match in the reference frame (as illustrated in Fig. 1). The sum of absolute differences (SAD)

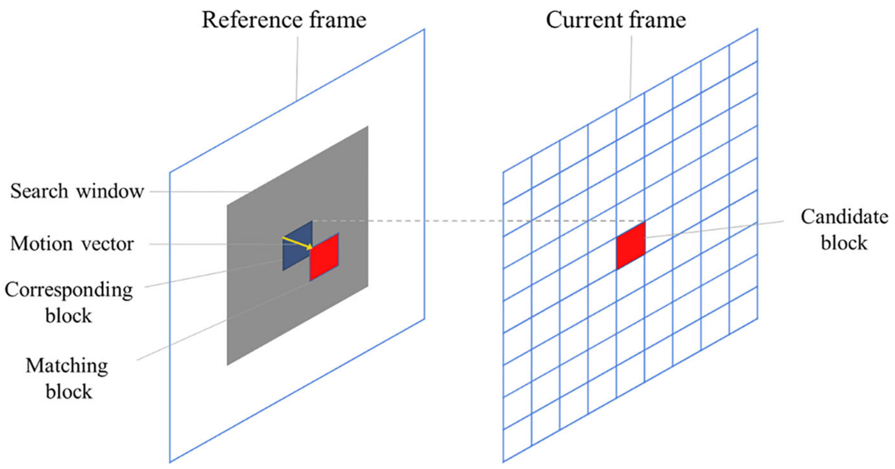


Fig. 1 Block-matching algorithm (BMA). In BMA, the final goal is to report a motion vector (MV) showing the block displacement in two consecutive frames

is a commonly used metric for finding the best match between blocks in ME due to its simplicity and efficiency in computation. The SAD is calculated by summing the absolute differences between corresponding pixels in the block in the current frame and its displaced counterpart in the reference frame. Specifically, for a symmetrical block with a size of N * N pixels in the (i, j) coordinates of the current frame, denoted by $C(i, j)$, and its corresponding block in the reference frame with a displacement of (v_x, v_y), indicated by $R(i + v_x, j + v_y)$, the SAD is defined as:

$$SAD = \sum_{i=1}^{N}\sum_{j=1}^{N}\left|C(i, j) - R(i + v_x, j + v_y)\right| \tag{1}$$

Other commonly used block distortion measures in block-matching motion estimation include mean absolute difference (MAD) and mean square error (MSE), which aim to quantify the average and the squared difference between the pixels in the blocks under analysis. However, these metrics require additional computational resources compared to SAD and may be more challenging to implement in hardware due to their more complex calculations.

FSA is an exhaustive block search method that examines all the blocks in the search window, providing the most accurate results. However, it also incurs a high computational burden. To overcome this limitation, various techniques have been proposed to accelerate the search and reach the best block sooner while maintaining acceptable levels of ME quality. These techniques fall into several categories, including fixed-pattern, adaptive, and hierarchical search algorithms, which will be described in the following section.

In this paper, we present an adaptive method for selecting block sizes in ME. First, we identify stationary blocks in the video frame to reduce computational complexity, group moving pixels into variable-sized blocks with three priorities for selecting the shape of blocks based on the identified motion locations, and investigate the relationship between motion direction and block structure. Second, we utilize predictive tools such as early stopping and varying window sizes to create a dynamic block-matching algorithm. Finally, we introduce two scanning strategies for blocks in the search window to form our final proposed algorithm.

The remainder of this paper is structured as follows. In Sect. 2, we provide an overview of related works and clarify our motivation. In Sect. 3, we describe our proposed algorithm in detail. In Sect. 4, we present and discuss the results of our approach, and in Sect. 5, we draw conclusions based on our findings.

2 Related Works and Motivation

2.1 Review

This section provides a review of algorithms that researchers use to accelerate the speed of ME. Based on the assumption of unimodal surface error [21], researchers have developed a series of fixed patterns, such as diamond, hexagon, square, cross, or a combination of these shapes, to limit the number of search points and reduce computation in ME [10, 17, 22, 30, 49, 51, 52]. The idea behind these methods is that by following a fixed pattern towards the best block in the search window, the error decreases continuously. Thus, a directional search towards the block with the lowest cost function will minimize the number of search points [12]. However, because the error surface has many local minimums, especially in complex and large patterns, these methods can get stuck in them and fail to track motion accurately [22, 50]. Another approach is multi-resolution search algorithms, which involve the creation of an image pyramid with varying sizes [25, 36, 45]. ME is performed first on the lowest-resolution image, and the resulting MVs are then rescaled and used to estimate motion on higher-resolution images. This approach can help to accelerate convergence. However, starting from low-resolution images means that any mistakes made will carry over to higher-resolution ones, leading to lower performance. Also, resizing the motion vectors when moving between resolutions may reduce the level of detail in motion and decrease the accuracy of ME. Some algorithms use a partial distortion search (PDS) to speed up the FSA [15, 43, 47]. PDS gradually calculates the distortion measure, such as SAD, pixel by pixel and stops the search when a partial SAD exceeds the current minimum SAD found during the search. Various modifications to PDS have been proposed to reduce the number of pixels needed to reach the minimum SAD sooner. For instance, sorting the absolute pixel differences in descending order is suggested since the difference between two blocks mainly depends on a small number of pixels with higher matching errors [6]. Additionally, selecting the first block to examine either at the center of the search window or based on a predicted MV can help find a better candidate block with the minimum SAD [15]. To address non-linear motion like zooming and rotation, some algorithms generate multiple transformed frames based on affine parameters and utilize them for ME [41]. However, a drawback of this approach is the high memory requirement for storing these transformed frames. Different approaches use interpolated versions of the frame instead of storing transformed frames with affine parameters to overcome the need for extensive memory storage [42]. Other methods focus on scaling candidate blocks rather than the entire frame. For example, in [14], authors introduce a zoom vector to adjust the block size and interpolate the zoomed block to match the size of the current block, enabling matching based on intensity. These algorithms face challenges due to the increased complexity of generating image versions and the increased number of search points required to identify the best match block. These factors contribute to the development of resource-intensive algorithms that may not be suitable for practical applications, given their high computational demands and memory requirements.

Other methods make use of the correlation between adjacent blocks in the spatial and corresponding blocks in the temporal domain. The concept is based on the observation that blocks near each other in time and space often share similar motion characteristics [13, 22]. Because of this, an MV for a block can be predicted before the search process even begins [12], and it can be used to adaptively modify the search parameters. For example, when we observe many blocks with large reported MVs, we can assume that this pattern will repeat for unexamined blocks. Therefore, we can use a bigger search window size to increase the scope of the search [16, 20, 32], modify search patterns to track such motion [1, 2, 18, 19, 24], and start searching from a more distant place than the origin of the search [29]. Additionally, correlated blocks can provide further insights, such as the cost function range. This can help to identify stationary blocks without searching for the remaining blocks [3, 4], a technique referred to as zero-motion prejudgment (ZMP). It also allows for the early selection of blocks with acceptable distortion, thereby leading to the early termination of the search [11]. Notably, all the studies mentioned herein have utilized fixed-sized blocks during BMA.

The utilization of fixed-size blocks in video compression has several drawbacks. When using large fixed block sizes, multiple moving objects can be located within a single block, leading to increased block residual and reduced ME accuracy due to the reporting of only one MV for the candidate block. Conversely, selecting a smaller block size leads to higher-quality ME due to an increased number of MVs that better capture motion information in a particular frame. However, this approach results in more MVs requiring additional bandwidth, which can degrade the compression ratio in both scenarios. Variable-size block-matching (VSBM) can be utilized to overcome these issues by selecting different block sizes. For instance, in High Efficiency Video Coding (HEVC), the size of a luma coding tree block (CTB) can be 64 * 64, 32 * 32, or 16 * 16 samples, and each CTB can be further partitioned into smaller blocks [34]. In Advanced Video Coding (AVC), the macroblock, the analogous structure to CTB, has a maximum size of 16 * 16 luma samples [23, 33]. This approach allows for enhanced coding efficiency by selecting larger blocks for stationary parts of frames and smaller blocks for areas with intricate motion, such as edges, thereby improving overall quality while minimizing motion information. The decision for selecting block sizes can vary, and in video coding standards, encoders employ a technique known as rate-distortion optimization to determine the optimal settings. This technique aims to minimize distortion while simultaneously satisfying a constraint on the number of bits required to encode the target frame by minimizing the cost function represented by Eq. (2). The Lagrange multiplier λ determines the trade-off between rate and distortion [34, 44]. However, solving this equation for every possible decision mode is computationally demanding [46].

$$J = D + \lambda R \tag{2}$$

Two popular approaches to block size selection are bottom-up and top-down VSBM algorithms [27]. The primary concept behind these techniques is to merge or split blocks based on their dissimilarities, resulting in variable block sizes [48]. For example, the top-down VSBM method conducts matching on large blocks and calculates the SAD between each block and its corresponding block in the reference frame. If the resulting cost exceeds a predetermined threshold, it subdivides the block into smaller sub-blocks and repeats the ME process for the generated sub-blocks. The remaining blocks that cannot improve the ME accuracy are subsequently merged [38]. Researchers in [30] introduce a recent implementation of this method. Initially, they resize each frame into a 256 * 256-pixel format. Subsequently, they divide the frame into 64 * 64 blocks and calculate the SAD between each block and its corresponding block in the reference frame. If the resulting cost exceeds a predefined threshold, they partition the block into smaller sub-blocks and perform the ME again for the generated sub-blocks. These methods require an adaptive threshold determined by the frame attributes because each region of every frame can display unique motion characteristics. This leads to an increase in computational complexity [40]. Additionally, resizing MVs to match the original frame resolution can reduce motion precision.

Another approach to selecting block sizes involves considering the texture of the frame. For example, [31] employs the Sobel operator to compute the image derivative in both horizontal and vertical directions and subsequently selects block sizes based on the edge value and direction. However, the high computational cost associated with calculating the edges can limit the practicality of this method. Furthermore, this algorithm does not account for motion in the temporal domain. As a result, the approach may report a smooth texture for a given region, but in the subsequent frame, a rupture may occur in that uniform area due to pixel movement, leading to a decrease in the quality of ME.

Another notable effort mentioned in [37] creates a statistical model for the BMA process, aiming to explain the relationship between various parameters, including the probability distribution function (PDF) of the MVs. Initially, it applies FSA and extracts MVs from several video test sequences. The study evaluates three distribution functions, Gaussian, Laplacian, and Cauchy, for modeling MVs and suggests that the Cauchy distribution, with some modifications, better models the PDF of the MVS. Secondly, using an unimodal error surface assumption, it links the number of search points and search patterns via a weighting function (WF) that sets the minimum search points needed to find the best match based on using a specific search pattern. It finds that broader search scopes result in fewer search points for higher motion sequences, while narrower patterns need fewer points for limited motion sequences. Finally, it models the average number of search points for a particular search algorithm as a linear function of the MV distribution and the WF of a specific search pattern.

When evaluating the quality of the proposed methods, it is common to calculate the mean square error (MSE) between the current and reconstructed frames, denoted as f and $\hat{\mathrm{f}}$, respectively, as shown in Eq. (3), where M and N are the dimensions of the image, and f(i,j) and $\hat{\mathrm{f}}$(i,j) represent pixel values at position (i,j). Subsequently, PSNR is used, as shown in Eq. (4), where $\mathrm{f}_{\mathrm{max}}$ represents the maximum possible pixel value, typically 255 for an 8-bit system. Alternatively, we can calculate MAD to find the dissimilarity between our reconstructed frame and the original one, as indicated in Eq. (5). Other common measure for evaluating the accuracy of motion estimation is the Structural Similarity Index (SSIM). It is based on the structural similarity between the original and reconstructed frames, taking into account the differences in luminance, contrast, and structural information. The SSIM index ranges from − 1 to 1, where 1 indicates perfect similarity between the two frames, 0 indicates no similarity, and negative values indicate dissimilarity. The formula for SSIM is given by Eq. (6), where f and $\hat{\mathrm{f}}$ are original and reconstructed frames,μ_f and $\mu_{\hat{f}}$ are the mean pixel values of f and $\hat{\mathrm{f}}$, $\sigma_f{}^2$ and $\sigma_{\hat{f}}{}^2$ are the variance of pixels, $\sigma_{f\hat{f}}$ is the covariance of pixels between f and $\hat{\mathrm{f}}$, and C_1 and C_2 are constants added to the denominator to avoid division by zero. To assess the efficiency of an algorithm, we quantified the number of blocks searched during BMA as a measure of algorithmic performance. A suitable algorithm should establish a trade-off between the quality and speed of ME.

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (f(i,j) - \widehat{f}(i,j))^2 \tag{3}$$

$$PSNR = 10\log_{10}(\frac{(f_{max})^2}{MSE})dB \tag{4}$$

$$MAD = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left| f(i,j) - \widehat{f}(i,j) \right| \tag{5}$$

$$SSIM(f, \widehat{f}) = \frac{(2\mu_f \mu_{\widehat{f}} + C_1)(2\sigma_{f\widehat{f}} + C_2)}{(\mu_f{}^2 + \mu_{\widehat{f}}{}^2 + C_1)(\sigma_f{}^2 + \sigma_{\widehat{f}}{}^2 + C_2)} \tag{6}$$

2.2 Motivation

As discussed earlier, although recent BMAs offer good performance in ME, their use of fixed-sized blocks ignores a powerful tool for further algorithm enhancement. Moreover, these methods fall behind the FSA due to their incorrect assumption of unimodal surface error and limited search points. On the other hand, FSA's high computational cost limits its practical application. Therefore, our primary objective in this study was to reduce ME's computational complexity while maintaining or improving its quality. To achieve this goal, first, we proposed a system that automatically detects moving pixels and groups them into blocks of varying forms and sizes. In this approach, we only perform ME on moving blocks and skip stationary regions. This block generation mechanism can be integrated into any BMA, making it a flexible tool for video processing applications. Second, we developed a predictive search algorithm that uses predictive tools with optimized implementation. This results in a reduction in computational requirements while maintaining the quality of ME.

3 Proposed Algorithm

In this section, we provide a comprehensive explanation of our proposed algorithm. To showcase the effectiveness of our approach, we conducted a comparative analysis with various existing methods, namely FSA, diamond search (DS) [51], new three-step search (NTSS) [17], four-step search (FSS) [26], adaptive rood pattern search (ARPS) [22], fast predictive search (FPS) [18], and an adaptive pattern selection (APS) [24] algorithm. The FPS method classifies motion types and incorporates different search patterns, and APS utilizes a modified version of the DS pattern and leverages predictive tools such as ZMP and adaptive pattern selection. It is important to note that APS and FPS employ fixed-size blocks for their computations.

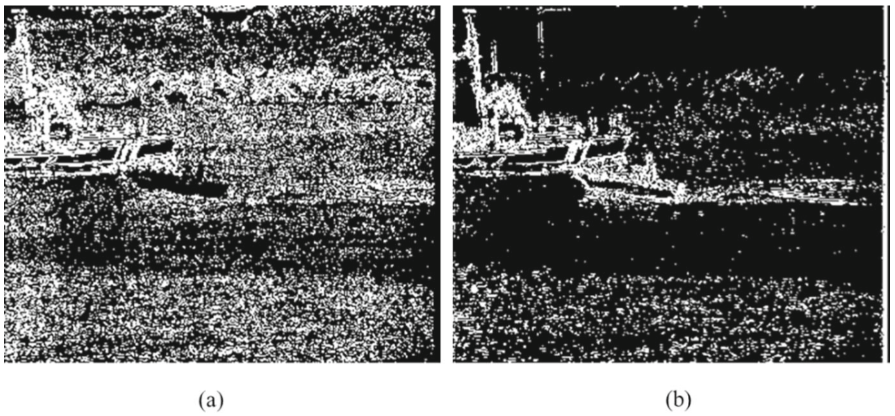


Fig. 2 Binary mask, without shifting the frame (**a**), with shifting the frame (**b**). We see a considerable reduction in moving pixels and, hence decreasing computational cost in the second setup

3.1 Finding Motion Parts

A widely used method to distinguish between moving and stationary pixels in video frames involves generating a difference image (DI) by subtracting the reference frame from the current frame, followed by applying a threshold to produce a binary mask (BM). In this approach, stationary areas correspond to black pixels (0), while moving parts correspond to white pixels (1). To improve the accuracy of this technique, we applied global motion compensation (GMC) to account for camera movements. In cases where many blocks within a frame have the same MV, we can assume that this motion corresponds to global motion. By applying this shift before subtraction, the resulting BM will have fewer white pixels and areas with different movements will be highlighted, as shown in Fig. 2. This approach is effective in identifying static blocks, reducing computational overhead, and accurately capturing areas of movement.

3.2 Extracting Static and Moving Blocks

In the first step of our proposed method, we divide our block-matching (BM) algorithm into large blocks with dimensions of 64 * 64 pixels. We calculate the sum of pixel values in each block and compare it to a predefined threshold to determine whether the block is static or not. Identifying stationary blocks allows us to replace them with corresponding blocks from the reference frame, thereby reducing computational requirements, especially for videos with slow motion. The larger block sizes used in this step facilitate better utilization of parallel processing and efficient handling of higher-resolution frames. In the second step, we further partition the moving blocks into smaller blocks of size 8 * 8 pixels, and similar to the previous step, we identify the static blocks (Fig. 3). We also can use an integral image to calculate the sum of rectangular regions in an image, which can enhance the computational efficiency of our method [39].

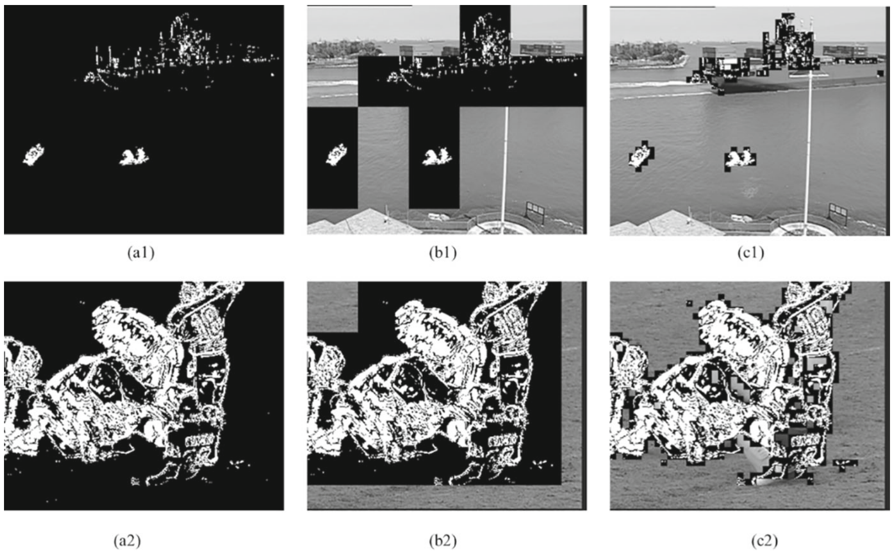


Fig. 3 Two examples of the proposed method for extracting static and moving blocks. Binary masks (**a1**, **a2**), placing corresponding large static blocks from the reference frame into the reconstructed frame (**b1**, **b2**), all the static blocks have been replaced with corresponding blocks in the reference frame (**c1**, **c2**)

3.3 Merging Moving blocks

After identifying all 8 * 8 moving blocks, we propose a method for merging them into larger blocks with varying sizes. Figure 4 illustrates the specific structures we have selected for the joined blocks. We choose these arrangements based on their potential to capture different motion directions; thus, the proposed method can potentially provide a more comprehensive representation of motion patterns in video frames, leading to improved accuracy of ME. The proposed method offers three options for prioritizing the shape of merged blocks: 1-square, 2-vertical, and 3-horizontal. For instance, if we select the square shape priority, we extract all possible 16 * 16 blocks first. For the remaining blocks, we choose the block shape with the largest possible size in any orientation. Choosing block shapes, from the smallest to largest sizes, depends on several factors. Two significant factors are the resolution of the original frames and the trade-off between accuracy and efficiency of ME. As the frame size increases, we can also increase the block sizes to improve algorithm efficiency, but this may result in some loss of accuracy because we're sampling less motion in the image. We choose the described setup specifically for frames with a resolution of 352 * 288 pixels, and we'll adjust it accordingly for higher-resolution images. Furthermore, In the results section, we will analyze the benefits and drawbacks of each structure and present a solution for selecting the most appropriate setup for ME.

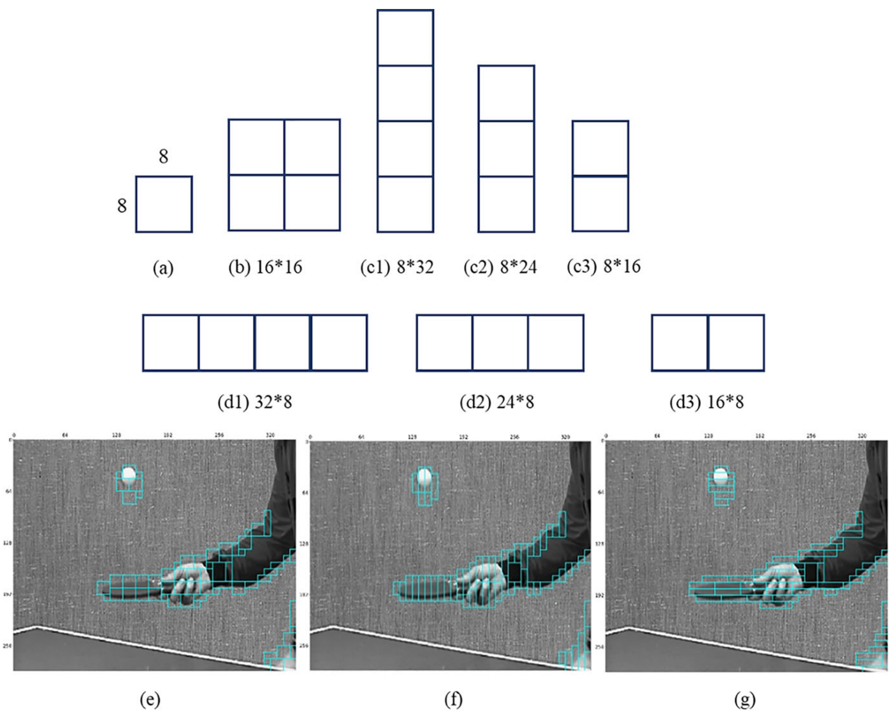


Fig. 4 Proposed structures and sizes in pixels for blocks, smallest block (**a**), square shape block (**b**), vertical blocks (**c1**, **c2**, **c3**), and horizontal blocks (**d1**, **d2**, **d3**). Three preferences for selecting the shape of blocks: square (**e**), vertical (**f**), and horizontal (**g**)

3.4 Using Predictive Tools

In the previous section, we constructed a base model for generating blocks with variable shapes and sizes that can be employed in various block-matching algorithms (BMAs). In the next step, we will utilize predictive tools to develop an efficient algorithm for ME.

3.4.1 Variable Window Size

Conventional BMAs typically use a fixed window size. However, we can adapt the window size based on the type of motion in the video sequence to optimize the performance of the algorithm. We considered two essential factors in determining the appropriate window size. Firstly, we used the most frequently occurring MV in the previously coded frame to approximate the direction and type of motion in the subsequent frame. Secondly, we compared the PSNR of the previously reconstructed frame to the mean PSNR of all reconstructed frames up to that point. This comparison helps estimate the degree of change in motion and determines the need for an increase in window size accordingly. An illustration of the proposed algorithm's operation is presented in Fig. 5 and compared to a BMA with a fixed window size. In this test series,

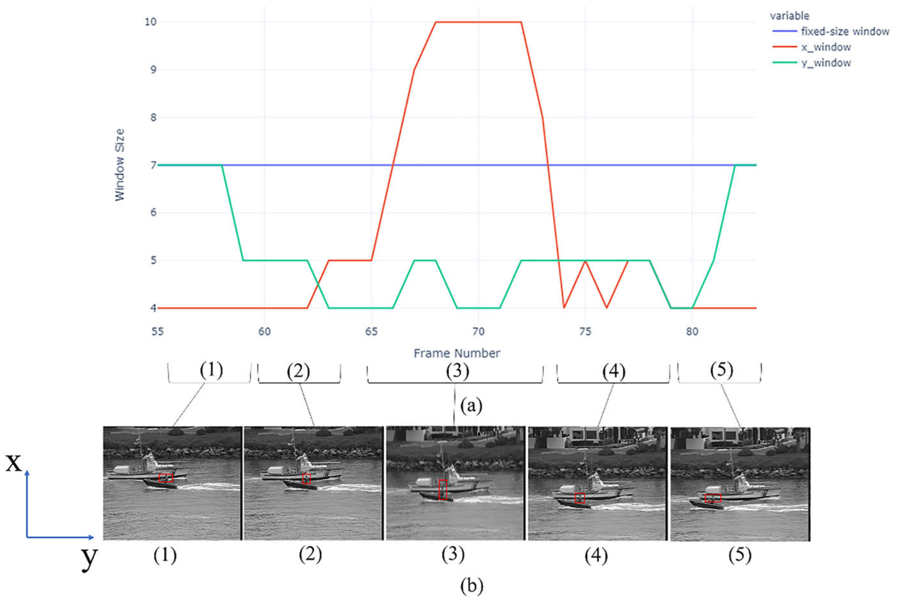


Fig. 5 Adaptive window size selection based on motion direction, selected window sizes (**a**), corresponding sample frames from five regions of motion, red rectangle shows the enlarged search window (**b**)

we examine five segments with unique motion patterns and display five frames from each part. We enlarge it to improve visibility. Regions one and five have significant horizontal motion, leading us to focus on this direction and enlarge the window accordingly (y_window). Regions two and four exhibit no clear dominant motion direction; therefore, we utilize a symmetrical window. Lastly, region 3 experiences a sudden shift to vertical motion, so we increase the window size in that direction (x_window) to enhance motion tracking.

3.4.2 Search Termination and Starting Point

Prior to initiating the search process for a block (B) in the current frame (CF) within the reference frame (RF), we evaluate three potential candidate blocks that may provide a good match for B. These candidate blocks include: (1) the corresponding block to B in the RF, (2) the block indicated by the MV reported for the identical block to B in the RF, and (3) the block corresponding to the global motion (GM) vector as shown Fig. 6. During this evaluation process, we compare the calculated cost function with the threshold defined in Eq. (7), where B_SADs represent the reported cost functions for the best matching blocks in the previously coded frame. If the cost is lower than the threshold, we terminate the search. However, if the search continues, we reposition the center of the search window to the block with the lowest cost function among the three mentioned blocks to expedite reaching the minima.

$$T = Max(Min(B\_SADs) + 256, 256) \tag{7}$$

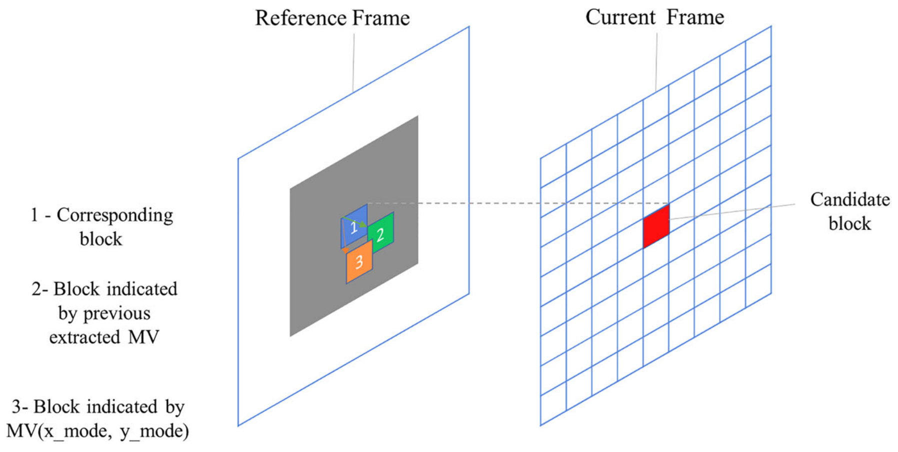

Reference Frame
Current Frame
1 - Corresponding block
2- Block indicated by previous extracted MV
3- Block indicated by MV(x_mode, y_mode)
Candidate block
1
2
3

Fig. 6 Examining three possible candidate blocks before starting the search

3.4.3 Threshold for BM

The generation of a BM through the subtraction of two consecutive frames and subsequent thresholding is a crucial step in our block generation algorithm. We must select a suitable threshold value based on the characteristics of the video sequences under analysis to ensure optimal performance. To choose the optimal configuration for generalization, we analyze different test sequences with varying levels of motion, ranging from low to high. Through this examination, we identify a recurring pattern in the accuracy of ME for videos with similar motion characteristics. Upon examination of the reconstructed frame PSNR across a range of thresholds, we have observed that, in instances of slow motion, a high threshold value leads to the exclusion of pertinent information, thereby resulting in quality degradation. Conversely, for more complex frames, selecting a low threshold value captures a greater degree of pixel intensity variation, including noise, and consequently reduces PSNR, as demonstrated in Fig. 7. To determine the most appropriate threshold value for each video, we analyze our dataset and categorize videos into five groups based on the sum of pixels in DIs, ranging from low to high variation. We then select an appropriate threshold value for each group.

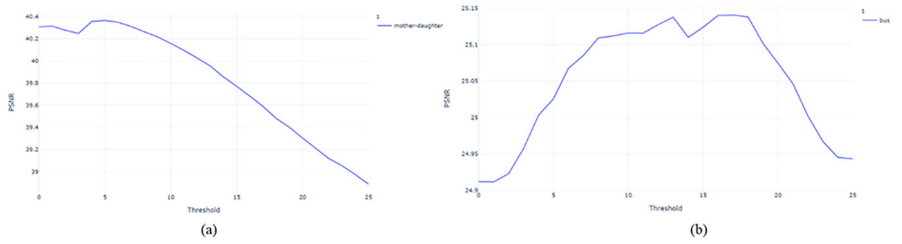


Fig. 7 PSNR versus threshold value, slow motion (**a**), fast motion (**b**)

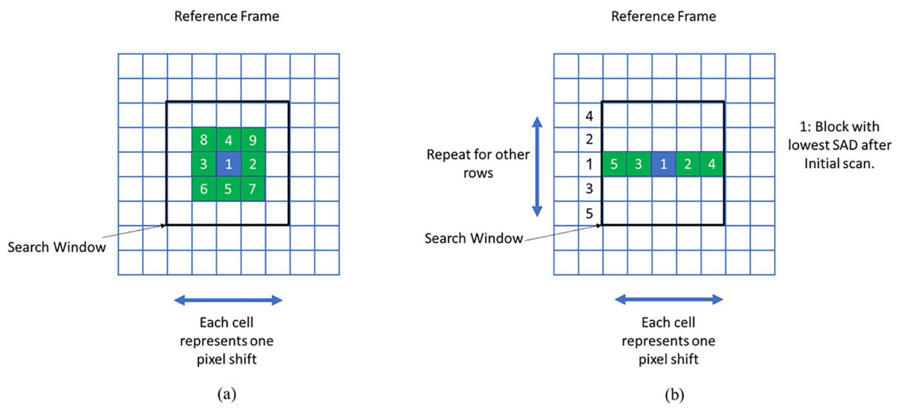

Reference Frame
8 4 9
3 1 2
6 5 7
Search Window
Each cell represents one pixel shift
(a)
Reference Frame
4
2
1 5 3 1 2 4
3
5
Repeat for other rows
Search Window
1: Block with lowest SAD after Initial scan.
Each cell represents one pixel shift
(b)

Fig. 8 Using a square pattern for the initial phase in our proposed quality-based BMA (**a**), scanning the rest of the blocks after the initial phase (**b**)

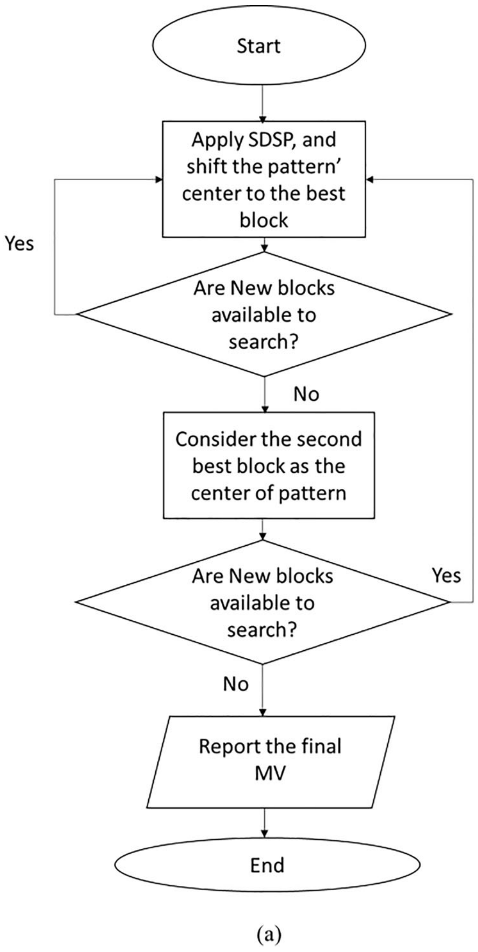
3.4.4 Scanning Patterns

In this section, we introduce two scanning strategies for developing our quality-based and efficient-based algorithms. Both methods involve examining all blocks within the respective search pattern until we find a block that satisfies the condition expressed in Eq. (7), upon which we terminate the search. The quality-based strategy employs a two-stage search. In the initial stage, we start from a small square pattern and examine nine blocks to determine the three blocks with the lowest cost function. We then move the center of the square pattern to the best block from the previous step and repeat the search. If the best match is in the center of the current square pattern or located on the border of the search window with no additional blocks to search, we consider the second or third-best block. If no new block is available after the first stage of searching and early termination hasn't stopped the search, we continue scanning the remaining blocks, as shown in Fig. 8.

In the efficiency-based method, we employ a small diamond search pattern (SDSP) and consider only the first two blocks with the highest similarity to the candidate block. Once the initial phase of searching is complete, we terminate the operation and report the best block found in that stage, as illustrated in Fig. 9.

4 Results and Discussion

In this section, we evaluate the performance of our algorithm by comparing it to other mentioned algorithms. The BMAs were configured with a window size of $\pm$ 7 and a block size of 16 * 16 with the previous frame as the reference frame. We conducted evaluations based on PSNR and the total number of searched blocks per frame across 16 CIF videos with a resolution of 352 * 288 pixels. To demonstrate the scalability of our algorithm, we further tested it on four high-definition (HD) videos with a resolution of 1280 * 720 pixels, with some necessary adjustments, such as selecting


Start
Apply SDSP, and shift the pattern' center to the best block
Yes
Are New blocks available to search?
No
Consider the second best block as the center of pattern
Are New blocks available to search?
Yes
No
Report the final MV
End
(a)

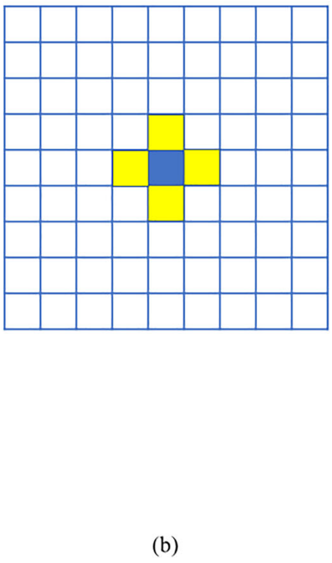


Fig. 9 The scanning pattern flowchart in our efficiency-based BMA (**a**), the SDSP (**b**)

32 * 32 pixels as the dimensions of the square-shaped block and 16 × 16 pixels as the smallest block size. To enhance the robustness of our comparisons, we have included MAD and SSIM metrics to further evaluate the performance of our algorithm. We performed simulations using Python version 3.8.6 on the Windows 10 OS platform, utilizing an AMD Ryzen 5 1600 at 3.20 GHz CPU with 8 GB RAM.

4.1 Proposed Block Generation Algorithm

To evaluate the impact of different block structures on ME, we conducted experiments by comparing the results of ME for three distinct block shape priorities, as shown in Table 1. For comparison, we used both the traditional block generation method and

Table 1 Comparison between the proposed block generation algorithm and FSA with respect to PSNR (dB) and reduction ratio of searched blocks

Videos	# Frames	FSA	Proposed block generation algorithm					
			Square		Horizontal		Vertical	
		PSNR	PSNR	Blocks searched reduction ratio (%)	PSNR	Blocks searched reduction ratio (%)	PSNR	Blocks searched reduction ratio (%)
1- news	300	36.73	**36.78 (+ 0.04)**	**87.43**	36.46 (– 0.28)	**87.0**	36.69 (– 0.05)	**87.64**
2- mobile	300	24.55	**24.58 (+ 0.03)**	– 4.2	24.12 (– 0.43)	**1.39**	**24.65 (+ 0.09)**	**1.9**
3- bus	150	24.67	**24.85 (+ 0.18)**	– 3.35	24.38 (– 0.29)	**3.27**	24.64 (– 0.03)	**1.63**
4- container	300	38.26	**38.28 (+ 0.01)**	**88.21**	38.26 (– 0.0)	**88.43**	**38.29 (+ 0.02)**	**88.14**
5- coastguard	300	30.43	**30.52 (+ 0.09)**	– 2.47	29.98 (– 0.46)	**3.05**	**30.44 (+ 0.01)**	**3.7**
6- foreman	300	31.21	**31.23 (+ 0.02)**	**21.3**	31.02 (– 0.19)	**25.44**	31.07 (– 0.14)	**25.02**
7- tennis	150	30.15	**30.41 (+ 0.25)**	**47.2**	29.92 (– 0.24)	**49.38**	**30.26 (+ 0.1)**	**48.81**
8- football	260	23.96	**24.03 (+ 0.07)**	**1.94**	23.68 (– 0.28)	**7.74**	23.76 (– 0.2)	**8.44**
9- stefan	90	25.46	**25.52 (+ 0.06)**	**24.92**	24.99 (– 0.48)	**26.99**	**25.69 (+ 0.23)**	**22.14**
10- mother-daughter	300	40.39	**40.24 (– 0.15)**	**84.31**	40.15 (– 0.24)	**84.12**	40.15 (– 0.24)	**84.61**
11- akiyo	300	42.94	42.91 (– 0.03)	**93.41**	42.94 (– 0.01)	**93.31**	42.8 (– 0.14)	**93.44**
12- harbour	300	28.53	**28.61 (+ 0.07)**	**7.13**	28.26 (– 0.28)	**8.99**	**28.88 (+ 0.34)**	**11.07**
13- silent	300	35.44	**35.84 (+ 0.39)**	**77.45**	**35.57 (+ 0.13)**	**77.35**	**35.64 (+ 0.2)**	**78.3**
14- waterfall	260	34.57	**34.59 (+ 0.02)**	**53.87**	**34.6 (+ 0.04)**	**53.4**	34.52 (– 0.04)	**55.05**
15- hall	300	34.81	**34.91 (+ 0.09)**	**80.33**	34.72 (– 0.09)	**79.47**	**34.93 (+ 0.12)**	**80.43**
16- flower	250	25.75	**26.1 (+ 0.35)**	**23.16**	24.7 (– 1.05)	**26.2**	**26.36 (+ 0.61)**	**28.12**

The bold numbers show better performance for each algorithm compared to FSA our proposed generating block system and examined all blocks similar to FSA without any predictive tools.

Based on the results shown in Table 1, our proposed algorithm demonstrates a significant reduction in computation cost for slow-movement videos (1, 4, 10, 11, 13, 15) due to the efficient selection of moving regions and performing ME only on those regions. On average, we observe a reduction of 42.54%, 44.90%, and 44.72% in the number of searched blocks compared to the traditional model for square, vertical, and horizontal priorities. These results demonstrate the ability of our algorithm to decrease the computation load. Furthermore, for square and vertical block priorities, we observe a slight increase in PSNR by 0.09 dB and 0.055 dB, respectively, while we observe a reduction of 0.26 dB for the horizontal preference.

To investigate the effect of block size on the accuracy of ME, we analyzed frames with horizontal and vertical movement, respectively, as depicted in Fig. 10. For the horizontal motion, selecting vertical blocks can sample more MVs along the flat path, thus improving the quality of ME. Conversely, choosing horizontal blocks captures fewer MVs along the dominant motion direction, resulting in a decrease in the quality of the reconstructed frame. Our investigation of the MVs in our dataset revealed that the principal motion direction in video frames is left and right. Therefore, the reduction in the accuracy of ME observed in the horizontal setup can be associated with this reason. For images with vertical movement, selecting horizontal blocks yields a better realization of motion in that direction. In conclusion, we have selected the square block pattern for our algorithm due to its ability to perform well in both horizontal and vertical directions in terms of accuracy and efficiency.

4.2 Proposed Algorithms Results

In this section, we present a comparative analysis between our proposed quality-based and efficiency-based algorithms and various block matching algorithms, including non-adaptive methods such as FSA, DS, NTSS, and FSS, as well as adaptive methods like ARPS, FPS, and APS. According to Table 2, we observe a noticeable improvement in video quality for most of the test sequences for our quality-based algorithm. It consistently achieves the highest PSNR compared to the other algorithms. For instance, it demonstrates an average improvement of 0.274 dB, 0.6 dB, and 0.59 dB in PSNR compared to FSA, DS, and APS, respectively. Furthermore, our efficiency-based algorithm also exhibits improvements in PSNR when compared to FSA, DS, and APS, with an average improvement of 0.07 dB, 0.39 dB, and 0.38 dB, respectively. We also compare the MAD and SSIM between reconstructed and original frames using our proposed methods against four high-accuracy algorithms: FSA, DS, ARPS, and APS. Our results demonstrate consistent reductions in MAD across all comparisons. Specifically, we observed reductions of 0.15 and 0.03 in MAD compared to FSA, reductions of 0.47 and 0.35 compared to DS, decreases of 0.32 and 0.19 compared to ARPS, and reductions of 0.4 and 0.28 compared to APS. Similarly, the SSIM improvements were notable, with increases ranging from 0.5 to 2.2% compared to the aforementioned algorithms. These findings suggest that our proposed algorithms could be potential candidates for video coding applications that require high accuracy.

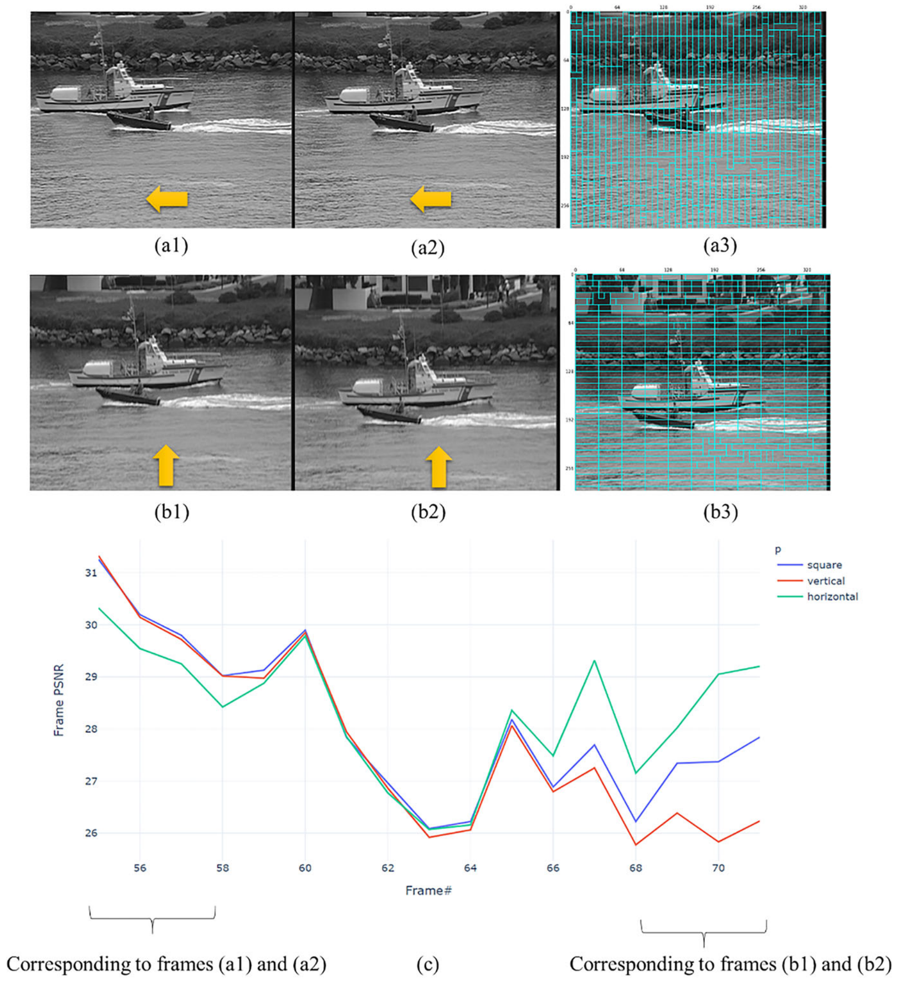


Fig. 10 Choosing block structures based on motion direction. Selecting vertical blocks (**a3**) for horizontal motion (**a1**, **a2**) and horizontal blocks (**b3**) for vertical motion (**b1**, **b2**), yellow arrows indicate motion direction. PSNR for reconstructed frames (**c**), investigating the quality of ME with consideration to both the motion direction and block shapes

To compare the bitrate of our proposed block matching motion estimation algorithm with the FSA, we entropy encoded key components involved in video compression: frame residuals, MVs, and keyframes. Our analysis revealed that the proposed algorithm results in a mean bitrate increase of 0.7% compared to FSA. This marginal increase is attributed to the higher number of MVs generated by our algorithm, which are crucial for refining the quality of ME. Despite the slight rise in bitrate, this increase is justified by the significant enhancement in ME quality, leading to better visual fidelity and more accurate frame predictions. Thus, the improved quality ME offered by our algorithm makes the small increase in bitrate a worthwhile trade-off.

Table 2 Performance comparisons between the proposed algorithms and seven block matching algorithms in terms of PSNR (dB)

Videos	#Frames	FSA	DS [51]	NTSS [17]	ARPS [22]	FSS [26]	FPS [18]	APS [24]	Proposed_Q	Proposed_E
1- news	300	36.73	36.53	36.5	36.46	36.5	36.04	36.5	**36.95**	36.7
2- mobile	300	24.55	24.49	24.36	24.5	24.36	24.5	24.45	**24.74**	24.7
3- bus	150	24.67	22.15	22.1	23.1	22.1	23.19	22.43	**25.14**	24.2
4- container	300	38.26	38.22	38.22	38.22	38.22	38.22	38.21	**38.31**	38.24
5- coastguard	300	30.43	30.36	30.34	30.41	30.34	30.27	30.38	**30.75**	30.72
6- foreman	300	31.21	30.83	30.77	30.92	30.77	31.06	30.89	**31.68**	31.45
7- tennis	150	30.15	29.57	29.51	29.4	29.51	29.35	29.36	**30.55**	29.8
8- football	260	23.96	23.33	23.4	23.48	23.4	20.22	23.38	**24.7**	24.01
9- stefan	90	25.46	24.46	24.5	25.14	24.5	25.47	24.84	**26.01**	25.81
10- mother-daughter	300	40.39	40.29	40.28	40.27	40.28	40.01	40.22	40.35	40.25
11- akiyo	300	**42.94**	42.93	42.86	42.91	42.86	42.85	42.91	**42.94**	42.92
12- harbour	300	28.53	28.52	28.44	28.49	28.44	28.38	28.48	**28.65**	28.63
13- silent	300	35.44	35.12	35.15	35.04	35.15	35.02	35.03	**35.99**	35.63
14- waterfall	260	**34.57**	**34.57**	34.56	34.56	34.56	34.55	34.56	**34.57**	**34.57**
15- hall	300	34.81	34.78	34.76	34.75	34.76	34.71	34.74	**34.89**	34.85
16- flower	250	25.75	25.66	25.48	25.65	25.48	25.63	25.68	**26.18**	26.12
17-fourpeople_HD	100	38.63	38.58	38.59	38.54	38.57	38.53	38.55	**38.82**	38.78
18-parkrun_HD	100	24.5	24.5	24.43	24.49	24.49	24.47	24.47	**24.69**	24.68
19-shields_HD	100	31.32	31.15	31.3	31.18	31.09	31.19	31.2	**31.52**	31.27
20-stockholm_HD	100	31.08	30.98	30.94	30.97	30.98	30.98	30.94	**31.52**	31.4
Average		31.67	31.35	31.32	31.42	31.32	31.23	31.36	**31.95**	31.74

The bold numbers show the best performance compared to other algorithms

To evaluate the efficiency of our algorithm, we measured the number of searched blocks per frame. According to Table 3, comparing our quality-based algorithm to FSA, we observed a substantial reduction in the examined blocks, ranging from 41.36 to 91.6%, with an average of 71.66%. This indicates that our method achieves comparable video quality to FSA while significantly reducing the computational cost. Moreover, our efficiency-based algorithm demonstrated a significant reduction of 97.93% in the number of searched blocks compared to FSA. Additionally, when compared to DS, APS, and FPS algorithms, we observed reductions of 74.22%, 20.07%, and 21.74%, respectively, in the searched blocks. These findings suggest that our proposed BMA outperforms other algorithms in terms of both video quality and computation costs. It achieves comparable quality to FSA while considerably reducing the computational burden. The substantial reduction in searched blocks in our efficiency-based algorithm highlights its efficiency and suitability for real-time processing applications.

By analyzing the implementation of algorithms such as FSA and fast FSA, we can find that this improvement comes from various strategies absent in those approaches. The increased accuracy comes from adjusting the search window based on the motion type, starting the search from a location closer to the global minimum, and selecting smaller block sizes for areas with complex motion to sample more MVs and achieve better accuracy. Improved efficiency arises from disregarding static blocks before initiating the search process, compensating global motion within frames before starting ME, starting the search from a position closer to the global minimum through predicted MVs to accelerate search termination, and choosing larger blocks in low-motion areas to speed up ME by reducing the overall number of search points required. Subsequently, we provide a numerical analysis of each predictive tool in the following section.

4.3 Analyzing Predictive Tools

We conducted an analysis of the results obtained by applying each tool to our CIF test sequences to determine the impact of each predictive tool on the performance of the quality-based algorithm. As mentioned before, by selecting variable block sizes, we notice an improvement in PSNR of nearly 0.1 dB and a reduction of 43% in the number of blocks compared to FSA. We proceeded by integrating multiple predictive tools and evaluated the resulting performance against the previous state. For example, choosing the window size based on the type and direction of predicted motion contributed to a 0.13 dB improvement in PSNR, while eliminating redundant search points resulted in a 13% reduction in the count of searched blocks. Combining early stopping with the scanning pattern and starting search point led to a significant decrease of almost 10% in the examined blocks. Furthermore, by shifting the search window closer to the minimum location, we achieved a nearly 0.3 dB increase in PSNR compared to the previous state. We also considered the computational burden imposed by each of these tools to determine their usefulness, and since they are independent of one another, we can customize our algorithm (Fig. 11).

Table 3 Performance comparisons between the proposed algorithms and seven block matching algorithms in terms of the number of searched blocks per frame / 1000

Videos	#Frames	FSA	DS [51]	NTSS [17]	ARPS [22]	FSS [26]	FPS [18]	APS [24]	Proposed_Q	Proposed_E
1- news	300	80.9	5.07	6.55	6.41	2.25	0.9	0.88	12.79	**0.51**
2- mobile	300	80.9	5.27	7.21	6.45	2.87	1.74	2.06	25.76	2.39
3- bus	150	80.9	7.77	10.54	8.11	3.79	3.6	3.51	44.07	**2.99**
4- container	300	80.9	4.91	6.38	6.31	2.04	0.53	0.58	12.21	**0.36**
5- coastguard	300	80.9	6.54	7.84	7.33	3.4	1.85	2.34	21.43	**1.62**
6- foreman	300	80.9	6.83	8.51	7.51	3.52	2.81	2.55	26.67	**2.37**
7- tennis	150	80.9	6.08	7.78	7.12	3.12	2.4	2.25	20.71	**1.97**
8- football	260	80.9	8.59	10.41	8.53	4.91	3.37	4.94	47.44	4.68
9- stefan	90	80.9	6.63	8.85	7.32	3.23	3.14	2.31	28.33	**2.24**
10- mother-daughter	300	80.9	5.37	7.12	6.67	2.57	0.91	0.95	16.35	**0.66**
11- akiyo	300	80.9	4.86	6.31	6.27	2.0	0.47	0.56	11.94	**0.37**
12- harbour	300	80.9	5.09	6.61	6.39	2.34	2.15	1.93	17.82	1.94
13- silent	300	80.9	5.25	6.77	6.52	2.43	1.13	1.23	19.27	**0.99**
14- waterfall	260	80.9	4.86	6.39	6.27	2.09	1.14	1.49	15.86	**0.97**
15- hall	300	80.9	5.09	6.69	6.43	2.29	0.83	1.48	12.75	**0.46**
16- flower	250	80.9	6.31	8.01	7.27	3.43	2.59	1.85	23.5	2.1
17-fourpeople_HD	100	193.97	12.23	16.07	15.44	5.47	5.45	4.75	16.27	**0.69**
18-parkrun_HD	100	193.97	17.15	23.05	18.25	8.39	7.84	4.9	84.09	5.54
19-shields_HD	100	193.97	19.53	27.83	20.05	8.4	8.35	5.58	80.3	5.93
20-stockholm_HD	100	193.97	15.8	21.76	17.54	8.37	7.79	5.52	62.98	**4.27**
Average		103.514	7.9615	10.534	9.1095	3.8455	2.9495	2.583	30.027	**2.1525**

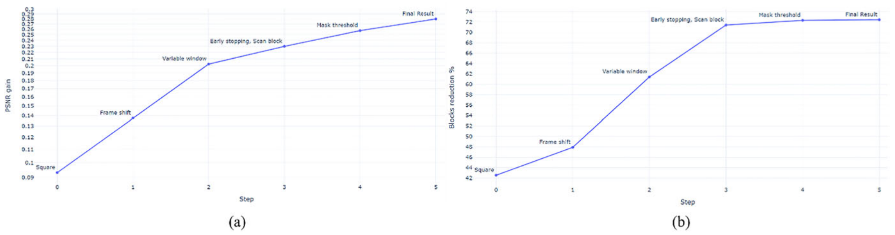


Fig. 11 PSNR gain in comparison with FSA after each step in the CIF test sequences (**a**), Blocks reduction percentage in comparison with FSA after each step in the CIF test sequences (**b**)

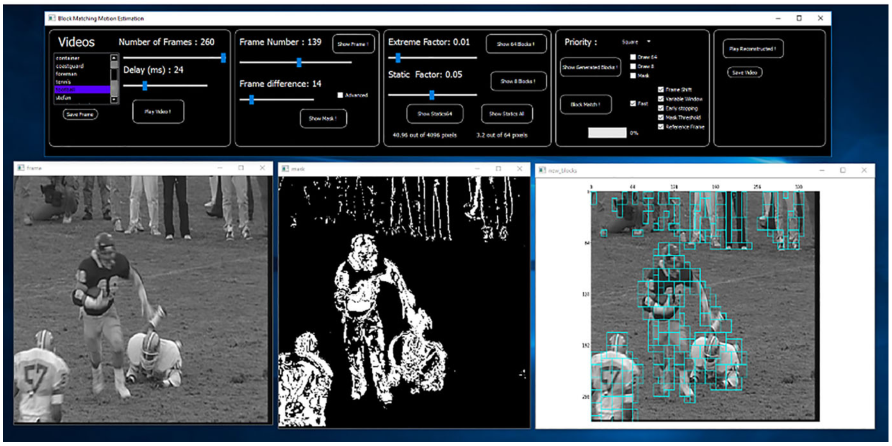


Fig. 12 Software implementation of the final algorithm

4.4 Software Implementation of the Final Algorithm

The final outcome of this project is the design and development of a Windows operating system software. We gather all the capabilities of the proposed algorithm in this software, and it can be personalized according to specific needs. This software encompasses features such as running experimental videos, displaying binary images, constructing and displaying created blocks with the ability to prioritize block shapes, executing the ME algorithm in terms of speed and quality, and ultimately generating and storing the reconstructed video after ME (Fig. 12).

4.5 Future Works

At the beginning of this algorithm, we mentioned three priorities for selecting the shape of our blocks. If we plot each constructed frame's PSNR with respect to the block shapes, we can see the performance of these shapes changes in every frame (Fig. 13). Although we can predict the best structure for movements in some directions, we cannot find a general pattern of how different motions and block shapes affect the

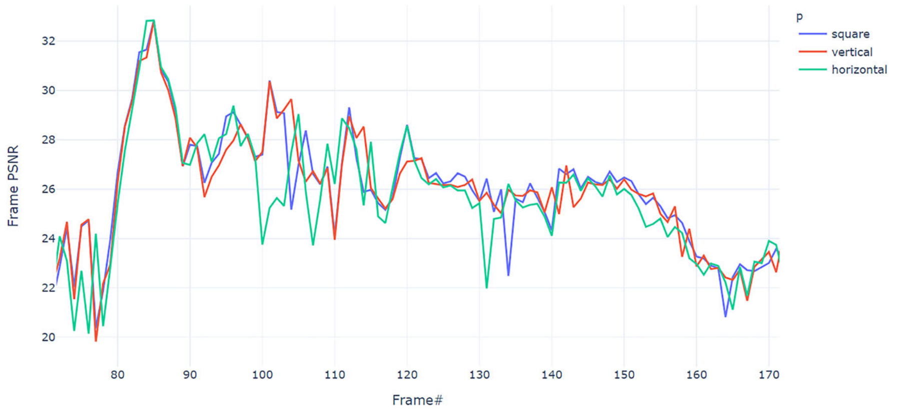


Fig. 13 How to find the best block structure to yield the best accuracy

resulting PSNR. One way we can solve this is to use deep learning; by using one of the well-known pre-trained convolutional neural networks (CNN) for feature extraction and building a classification on top of that model, we can construct a system that accepts DI as its input, and it will predict the proper shape for our blocks as its output. We look forward to tackling this problem in our future work.

5 Conclusion

Our goal in this paper was to introduce an adaptive block-matching motion estimation that utilizes predictive tools to search more efficiently. Therefore, first, we created variable-size blocks based on the motion. Second, we focused on designing the search process. We developed a dynamic window size selection system based on the predicted MVs for the next frame. We chose early stopping criteria and starting search points to converge to the best match sooner. We used two search strategies with different patterns to achieve the best accuracy and speed. Finally, we showed how each step affects the overall performance of our system. Our experimental results demonstrate that our proposed approach successfully meets the objectives of this study. Specifically, our approach achieves the precision of FSA while significantly improving the speed of ME.

Authors' Contributions The authors confirm contribution to the paper as follows: study conception and design: Milad Mirjalili, Amir Mousavinia; data collection and implementation: Milad Mirjalili; analysis and interpretation of results: Milad Mirjalili, Amir Mousavinia; draft manuscript preparation: Milad Mirjalili, Amir Mousavinia; all authors reviewed the results and approved the final version of the manuscript.

Funding This work was supported by the Department of Electrical Engineering, K.N. Toosi University of Technology.

Data Availability The datasets analyzed during the current study are available at http://trace.eas.asu.edu/yuv/ and https://media.xiph.org/video/derf/.

Code Availability Code is available upon request.

Declarations

Conflict of interest The authors declare that they have no potential known competing financial interests or personal relationships that could have affected the work reported in this paper.

References

1. H. Amirpour, A. Mousavinia, A dynamic search pattern motion estimation algorithm using prioritized motion vectors. SIViP **10**(8), 1393–1400 (2016). https://doi.org/10.1007/s11760-016-0906-5
2. H. Amirpour, M. Ghanbari, A. Pinheiro, M. Pereira, Motion estimation with chessboard pattern prediction strategy. Multimed. Tools Appl. **78**, 21785–21804 (2019). https://doi.org/10.1007/s11042-019-7432-8
3. S. M. Arora, N. Rajpal, R. Purwar, Dynamic pattern search algorithm with zero motion prejudgment for fast motion estimation, in *Fifth International Conference on Advanced Computing & Communication Technologies*. 138–142 (2015). https://doi.org/10.1109/ACCT.2015.133
4. S.M. Arora, N. Rajpal, K. Khanna, A new approach with enhanced accuracy in zero motion prejudgment for motion estimation in real-time applications. J. Real-Time Image Proc. **16**, 989–1005 (2019). https://doi.org/10.1007/s11554-016-0593-z
5. D. Fortun, P. Bouthemy, C. Kervrann, Optical flow modeling and computation: a survey. Comput. Vis. Image Underst. **134**, 1–21 (2015). https://doi.org/10.1016/j.cviu.2015.02.008
6. W.-G. Hong, T.M. Oh, Sorting-based partial distortion search algorithm for motion estimation. Electron. Lett. **40**(2), 113–115 (2004). https://doi.org/10.1049/el:20040091
7. B.K.P. Horn, B.G. Schunck, Determining optical flow. Artif. Intell. **17**(1–3), 185–203 (1981). https://doi.org/10.1016/0004-3702(81)90024-2
8. Y.-W. Huang, C.-Y. Chen, C.-H. Tsai, C.-F. Shen, L.-G. Chen, Survey on block matching motion estimation algorithms and architectures with new results. J. VLSI Signal Process. **42**(3), 297–320 (2006). https://doi.org/10.1007/s11265-006-4190-4
9. A.J. Hussain, Z. Ahmed, A survey on video compression fast block matching algorithms. Elsevier Neurocomput. **335**, 215–237 (2019). https://doi.org/10.1016/j.neucom.2018.10.060
10. D. Kerfa, M.F. Belbachir, Star diamond: an efficient algorithm for fast block matching motion estimation in H.264/AVC video codec. Multimed. Tools Appl. **75**, 3161–3175 (2016). https://doi.org/10.1007/s11042-014-2428-x
11. D. Kerfa, A. Saidane, An efficient algorithm for fast block matching motion estimation using an adaptive threshold scheme. Multimed. Tools Appl. **79**, 24173–24184 (2020). https://doi.org/10.1007/s11042-020-09040-z
12. B.-G. Kim, S.-K. Song, P.-S. Mah, Enhanced block motion estimation based on distortion-directional search patterns. Pattern Recogn. Lett. **27**(12), 1325–1335 (2006). https://doi.org/10.1016/j.patrec.2006.01.004
13. D.-W. Kim, J.-S. Choi, J.-T. Kim, Adaptive motion estimation based on spatio–temporal correlation. Signal Process. Image Commun. **13**(2), 161–170 (1998). https://doi.org/10.1016/S0923-5965(98)80013-1
14. H.-S. Kim, J.-H. Lee, C.-K. Kim, B.-G. Kim, Zoom motion estimation using block-based fast local area scaling. IEEE Trans. Circuits Syst. Video Technol. **22**(9), 1280–1291 (2012). https://doi.org/10.1109/TCSVT.2012.2198137
15. J.-H. Kim, B.-G. Kim, C.-S. Cho, Distortion-based partial distortion search for fast motion estimation, in *IEEE International Conference on Multimedia and Expo*., pp. 1599–1602 (2007). https://doi.org/10.1109/ICME.2007.4284971
16. Y.-H. Ko, H.-S. Kang, S.-W. Lee, Adaptive search range motion estimation using neighboring motion vector differences. IEEE Trans. Consum. Electron. **57**(2), 726–730 (2011). https://doi.org/10.1109/TCE.2011.5955214
17. R. Li, B. Zeng, M.L. Liou, A new three-step search algorithm for block motion estimation. IEEE Trans. Circuits Syst. Video Technol. **4**(4), 438–442 (1994). https://doi.org/10.1109/76.313138

18. L. Lin, I.-C. Wey, J.-H. Ding, Fast predictive motion estimation algorithm with adaptive search mode based on motion type classification. SIViP **10**(1), 171–180 (2016). https://doi.org/10.1007/s11760-014-0723-7
19. J. Luo, X. Yang, L. Liu, A fast motion estimation algorithm based on adaptive pattern and search priority. Multimed. Tools Appl. **74**, 11821–11836 (2015). https://doi.org/10.1007/s11042-014-2280-z
20. A. Medhat, A. Shalaby, M.S. Sayed, M. Elsabrouty, F. Mehdipour, Adaptive low-complexity motion estimation algorithm for high efficiency video coding encoder. IET Image Proc. **10**(6), 438–447 (2016). https://doi.org/10.1049/iet-ipr.2015.0666
21. O. Ndili, T. Ogunfunmi, Algorithm and architecture co-design of hardware-oriented, modified diamond search for fast motion estimation in H.264/AVC. IEEE Trans. Circuits Syst. Video Technol. **21**(9), 1214–1227 (2011). https://doi.org/10.1109/TCSVT.2011.2133990
22. Y. Nie, K.-K. Ma, Adaptive rood pattern search for fast block-matching motion estimation. IEEE Trans. Image Process. **11**(12), 1442–1449 (2002). https://doi.org/10.1109/TIP.2002.806251
23. J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, T. Wedi, Video coding with H.264/AVC: tools, performance, and complexity. IEEE Circuits Syst. Mag. **4**(1), 7–28 (2004). https://doi.org/10.1109/MCAS.2004.1286980
24. Z. Pan, R. Zhang, W. Ku, Y. Wang, Adaptive pattern selection strategy for diamond search algorithm in fast motion estimation. Multimed. Tools Appl. **78**(2), 2447–2464 (2019). https://doi.org/10.1007/s11042-018-6353-2
25. A.V. Paramkusam, V.S.K. Reddy, A novel fast search motion estimation boosted by multilayer concept. Multimed. Tools Appl. **75**, 2169–2188 (2016). https://doi.org/10.1007/s11042-014-2400-9
26. L.-M. Po, W.-C. Ma, A novel four-step search algorithm for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol. **6**(3), 313–317 (1996)
27. I. Rhee, G. Martin, S. Muthukrishnan, R.A. Packwood, Quadtree-structured variable-size block-matching motion estimation with minimal error. IEEE Trans. Circuits Syst. Video Technol. **10**(1), 42–50 (2000). https://doi.org/10.1109/76.825857
28. I.E. Richardson, *The H.264 Advanced Video Compression Standard* (Wiley Publishing, New York, 2010)
29. Y.A. Salih, L.E. George, Improved hybrid block-based motion estimation for inter-frame coding. Circuits Syst. Signal Process. **40**(7), 3500–3522 (2021). https://doi.org/10.1007/s00034-020-01637-x
30. G. Senbagavalli, R. Manjunath, Motion estimation using variable size block matching with cross square search pattern. SN Appl. Sci. (2020). https://doi.org/10.1007/s42452-020-03248-2
31. S. Shaik, P. Muralidhar, C. B. R. Rao, A new approach to variable block size motion estimation with fast and effective searching technique, in *International Conference on Communication and Signal Processing (ICCSP)*, pp. 676–680 (2016). https://doi.org/10.1109/ICCSP.2016.7754228
32. S.M.R. Soroushmehr, S. Samavi, S. Shirani, Simple and efficient motion estimation algorithm by continuum search. Multimed. Tools Appl. **71**(3), 1615–1633 (2014). https://doi.org/10.1007/s11042-012-1298-3
33. G.J. Sullivan, T. Wiegand, Video compression—from concepts to the H.264/AVC standard. Proc. IEEE **93**(1), 18–31 (2005). https://doi.org/10.1109/JPROC.2004.839617
34. G.J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012). https://doi.org/10.1109/TCSVT.2012.2221191
35. D. Sun, S. Roth, M. J. Black, Secrets of optical flow estimation and their principles, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2432–2439 (2010). https://doi.org/10.1109/CVPR.2010.5539939
36. N.V. Thang, J. Choi, J.-H. Hong, J.-S. Kim, H.-J. Lee, Hierarchical motion estimation for small objects in frame-rate up-conversion. IEEE Access **6**, 60353–60360 (2018). https://doi.org/10.1109/ACCESS.2018.2875688
37. J.-J. Tsai, H.-M. Hang, Modeling of pattern-based block motion estimation and its application. IEEE Trans. Circuits Syst. Video Technol. **19**(1), 108–113 (2009). https://doi.org/10.1109/TCSVT.2008.2009248
38. Y.-K. Tu, J.-F. Yang, Y.-N. Shen, M.-T. Sun, Fast variable-size block motion estimation using merging procedure with an adaptive threshold, in *International Conference on Multimedia and Expo.*, vol. 2, II-789 (2003). https://doi.org/10.1109/ICME.2003.1221735

39. P. Viola, M. J. Jones, Rapid object detection using a boosted cascade of simple features, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2001). https://doi.org/10.1109/CVPR.2001.990517
40. H. Wang, Q. Liu, H. Lu, An improved variable-size block-matching algorithm. Multimed. Tools Appl. **34**(2), 1573–7721 (2007). https://doi.org/10.1007/s11042-006-0091-6
41. T. Wiegand, E. Steinbach, B. Girod, Affine multipicture motion-compensated prediction. IEEE Trans. Circuits Syst. Video Technol. **15**(2), 197–209 (2005). https://doi.org/10.1109/TCSVT.2004.841690
42. K.-M. Wong, L.-M. Po, K.-W. Cheung, K.-H. Ng, Block-matching translation and zoom motion-compensated prediction by sub-sampling, in *16th IEEE International Conference on Image Processing (ICIP)*, pp. 1597–1600 (2009). https://doi.org/10.1109/ICIP.2009.5413383
43. X.-P. Xia, E.-H. Liu, J.-J. Qin, A fast partial distortion search algorithm for motion estimation based on the multi-traps assumption. Signal Process. Image Commun. **31**, 25–33 (2015). https://doi.org/10.1016/j.image.2014.11.007
44. E.-H. Yang, X. Yu, Rate distortion optimization for H.264 interframe coding: a general framework and algorithms. IEEE Trans. Image Process. **16**(7), 1774–1784 (2007). https://doi.org/10.1109/TIP.2007.896685
45. H. Yin, H. Jia, H. Qi, X. Ji, X. Xie, W. Gao, A hardware-efficient multi-resolution block matching algorithm and its VLSI architecture for high definition MPEG-like video encoders. IEEE Trans. Circuits Syst. Video Technol. **20**(9), 1242–1254 (2010). https://doi.org/10.1109/TCSVT.2010.2058476
46. J. Zhang, X. Yi, N. Ling, W. Shang, Bit rate distribution for motion estimation in H.264 coding. IEEE Trans. Consum. Electron. **52**(2), 606–610 (2006). https://doi.org/10.1109/TCE.2006.1649686
47. R. Zhang, Z. Pan, W. Ku, A novel parallel implementation of partial distortion search algorithm based on template search. Multimed. Tools Appl. **77**, 20615–20628 (2018)
48. Z. Zhou, M.-T. Sun, Y.-F. Hsu, Fast variable block-size motion estimation algorithm based on merge and slit procedures for H.264/MPEG-4 AVC, in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, III-725 (2004). https://doi.org/10.1109/ISCAS.2004.1328849
49. C. Zhu, X. Lin, L.-P. Chau, Hexagon-based search pattern for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol. **12**(5), 349–355 (2002). https://doi.org/10.1109/TCSVT.2002.1003474
50. C. Zhu, X. Lin, L. Chau, L.-M. Po, Enhanced hexagonal search for fast block motion estimation. IEEE Trans. Circuits Syst. Video Technol. **14**(10), 1210–1214 (2004). https://doi.org/10.1109/TCSVT.2004.833166
51. S. Zhu, K.-K. Ma, A new diamond search algorithm for fast block-matching motion estimation. IEEE Trans. Image Process. **9**(2), 287–290 (2000). https://doi.org/10.1109/83.821744
52. B.-J. Zou, C. Shi, C.-H. Xu, S. Chen, Enhanced hexagonal-based search using direction-oriented inner search for motion estimation. IEEE Trans. Circuits Syst. Video Technol. **20**(1), 156–160 (2010). https://doi.org/10.1109/TCSVT.2009.2031461