Applied Intelligence
https://doi.org/10.1007/s10489-018-1312-1

A new block matching algorithm based on stochastic fractal search

Abir Betka[1] · Nadjiba Terki[1] · Abida Toumi[1] · Madina Hamiane[2] · Amina Ourchani[1]

Abstract

Block matching algorithm is the most popular motion estimation technique, due to its simplicity of implementation and effectiveness. However, the algorithm suffers from a long computation time which affects its general performance. In order to achieve faster motion estimation, a new block matching algorithm based on stochastic fractal search, SFS, is proposed in this paper. SFS is a metaheuristic technique used to solve hard optimization problems in minimal time. In this work, two main contributions are presented. The first one consists of computing the motion vectors in a parallel structure as opposed to the other hierarchical metaheuristic block matching algorithms. When the video sequence frame is divided into blocks, a multi-population model of SFS is used to estimate the motion vectors of all blocks simultaneously. As a second contribution, the proposed algorithm is modified in order to enhance the results. In this modified version, four ideas are investigated. The random initialization, usually used in metaheuristics, is replaced by a fixed pattern. The initialized solutions are evaluated using a new fitness function that combines two matching criteria. The considered search space is controlled by a new adaptive window size strategy. A modified version of the fitness approximation method, which is known to reduce computation time but causes some degradation in the estimation accuracy, is proposed to balance between computation time and estimation accuracy. These ideas are evaluated in nine video sequences and the percentage improvement of each idea, in terms of estimation accuracy and computational complexity, is reported. The presented algorithms are then compared with other well-known block matching algorithms. The experimental results indicate that the proposed ideas improve the block matching performance, and show that the proposed algorithm outperforms many state-of-the-art methods.

Keywords Block matching algorithm · Motion estimation · Stochastic fractal search · Metaheuristics

1 Introduction

Motion estimation (ME) is an important topic in computer vision. It consists of computing the motion vectors of pixel displacements in the frames. ME techniques are divided into two basic approaches: Pixel-based motion estimation (PBME) and block-based motion estimation (BBME). While, the first approach, PBME, known as optical flow technique, computes a motion vector for each pixel in the frame by using the brightness constancy constraint equation and an additional smoothness constraint [1–5], the second approach, BBME, called also block matching (BM) algorithm, divides the frame into blocks and then computes a motion vector for each block; the same motion vector is associated with all the pixels in the block [6, 7]. BM algorithms are more suitable for real-time applications, and are commonly used in video compression systems for temporal redundancy exploitation [8].

For each block in the current frame, BM algorithms compute the motion vectors by searching for the block of closer intensity (or color) within a search window in the reference frame. The motion vector is the difference between the positions of the current block and the best-matched block found in the reference frame. The global best-matched block is attained when all possible candidate

✉ Abir Betka
abir_betka@hotmail.fr

Nadjiba Terki
t_nadjiba@yahoo.fr

Abida Toumi
abida_ba@yahoo.fr

Madina Hamiane
mhamiane@ahlia.edu.bh

Amina Ourchani
minanim7@yahoo.fr

[1] Department of Electrical Engineering, University of Biskra Algeria, Biskra, Algeria

[2] Department of Telecommunication Engineering, Ahlia University, Manama, Bahrain

Published online: 25 October 2018

blocks in the search window are evaluated. This searching strategy is called full search algorithm (FSA). The main problems of FSA are the need for a high computation time and the right choice of the size of the search window.

To reduce the computational complexity of FSA, several fast and efficient block matching algorithms have been developed. Two comprehensive surveys of these algorithms have been reported by Barjatya [9] and Choudhury et al. [10]. The searching strategies of such algorithms could reduce the computation time but they failed in dynamic motion and in many cases they can be trapped into local optimums.

Since the BM problem can be seen as an optimization problem that can be solved by any metaheuristic technique, several BM algorithms based on metaheuristics have been proposed in the literature. The work presented by Li et al. [11] is one of the first researches that used genetic algorithm (GA) in BM. Particle swarm optimization (PSO) algorithm was also used to solve the BM problem in [12–14]. Cuevas et al. have developed a block matching algorithm using differential evolution (DE) [15]. To reduce the number of fitness evaluations, a fitness approximation method based on the nearest-neighbor-interpolation concept is proposed. In this method the fitness values of some candidate blocks are estimated based on previous evaluated neighboring candidate blocks. Recently, a BM algorithm, based on the Harmony Search method was proposed by Daz-Corts et al. [16]. Damerchilu et al. [17] proposed a different approach in which motion estimation is based on learning automata. In their contribution, the authors considered the current block as a learning automaton and the candidate blocks as actions and at each instance, the learning automaton selects an action according to its probability vector. In order to achieve faster convergence, they utilized the fast discretized pursuit learning automata method [18].

Stochastic fractal search (SFS) is a recent metaheuristic technique developed by Salimi [19]. The technique has been applied in different optimization problems with promising results [20–22].

In this paper, a new BM algorithm based on stochastic fractal search is proposed. The presented algorithm, SFS-BM, uses a multi-population model in order to simultaneously compute the motion vectors of all the blocks in the frame. While the sequential implementation computes the motion vectors one block at a time, its parallel counterpart concurrently computes all blocks' motion vectors which makes the motion estimation significantly faster. To further improve the results of SFS-BM, a modified version of the presented algorithm, MSFS-BM, which incorporates four new ideas, is developed:

1. The random initialization of metaheuristics is replaced by a fixed pattern, in order to accelerate the convergence.
2. The initialized candidate blocks are evaluated through a fitness function. While BM algorithms usually employ the mean square error (MSE) and the sum of absolute differences (SAD) as typical fitness functions, we have combined the two matching criteria to fully benefit from the strengths of both criteria.
3. With fast movement, a large search window is usually required, whereas for slower motion, a small window may be sufficient. To find the most appropriate window size, we propose to adaptively adjust the window size for each block in a given frame using the motion vectors obtained in preceding frames.
4. The fitness approximation method proposed by Cuevas et al. [15] speed ups the algorithm by decreasing the number of fitness evaluations, but causes significant degradation in the estimation accuracy. To prevent such poor performance, we propose to replace the condition that decides which blocks are evaluated or estimated by the successive elimination algorithm's condition suggested by Li and Salari [30].

The rest of the paper is organized as follows: In the next section, we describe the basic concept of block matching algorithms. In the third section, the principle of SFS algorithm is explained. Section 4 presents SFS-BM and MSFS-BM algorithms. In Section 5 the experimental results are presented. Finally, we summarize this work with a conclusion and some perspectives.

2 Block matching algorithm

Similarly to all motion estimation techniques, BM algorithms are based on the assumption that the pixel's intensity or color stays constant during motion, and the transformations which change one frame to the next frame result solely from the motion in the frame. The basic concept of BM algorithms is illustrated in Fig. 1 and can be described as follows: First, the current and reference frames, of size $M \times N$ pixels, are subdivided into non-overlapping square blocks B_{ij} of size $L \times L$ with $i = 1, 2, 3, \cdots \frac{M}{L}$ and $j = 1, 2, 3, \cdots \frac{N}{L}$. Then, for each block in the current frame, the algorithm searches for the most similar block in the reference frame within a search window of size $(2w + 1) \times (2w+1)$. The best-matched block is the candidate block that optimizes a certain matching criterion. For this evaluation, almost all existing block matching algorithms use the sum of absolute differences (SAD), the mean square error (MSE) or peak signal to noise ratio (PSNR). The motion vector (MV) is the difference between the position of the current block in the current frame and that of the best-matched block in the reference frame. The full search algorithm (FSA) is the BM algorithm that provides an exhaustive searching strategy. It evaluates all possible candidate blocks in the search window, so it verifies $(2w + 1)^2$ search locations. With a

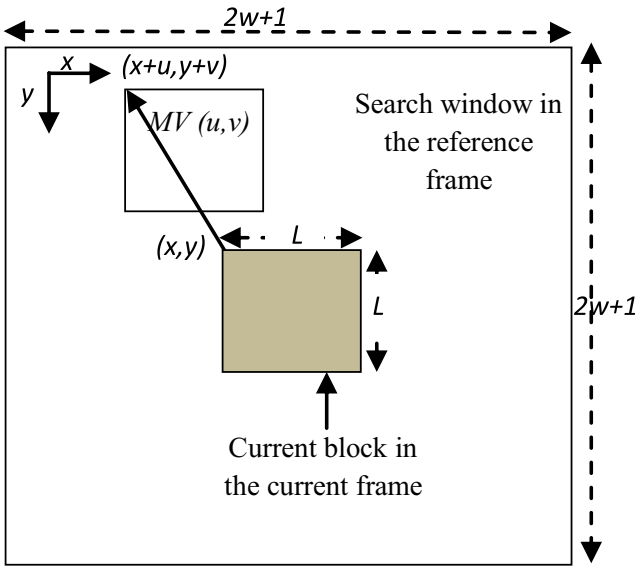

Fig. 1 BM principle MV(u,v) represents the horizontal and vertical displacements

window size $w = 8$, FSA evaluates $17 \times 17 = 289$ blocks. The high computational complexity of FSA represents the main drawback of this technique that needs to be addressed [6, 7].

Algorithm 1 depicts the pseudo code of FSA.

Algorithm 1 BM algorithm

```
Initialize the parameters (window size and block size)
Read the sequence video I
for f = 1: the maximum number of frames do
    Current frame = I(f)
    Reference frame = I(f − 1) or I(f + 1)
    Subdivide the current and reference frames into blocks
    for Each current block in the current frame do
        Minimum cost=Inf
        for Each candidate block in the reference frame do
            Cost=∑∑ | current block−candidate block|
            if Cost < Minimum cost then
                The candidate block is the best-matched block
                Minimum cost = Cost
            end if
        end for
    end for
end for
```

3 Stochastic fractal search

Metaheuristic techniques have been developed to approximately solve hard optimization problems. They are able to procure optimal solutions for complex problems with limited a prior knowledge and merely require a fitness function to guide the search [23, 24]. These techniques are classified into three categories, evolutionary algorithms, swarm-based techniques and physics-based techniques.

Stochastic fractal search (SFS) is a recent evolutionary algorithm proposed by Hamid Salimi [19] in 2015. It was inspired by the natural phenomenon of growth that uses the fractal mathematical concept. SFS algorithm starts with a random initialization of Np candidate solutions in the search space, each candidate solution is evaluated through a fitness function to determine the best solution. SFS subsequently uses three main processes to converge iteratively to the global best solution.

SFS algorithm consists of the following sequential processes:

Diffusion process. In which each candidate solution diffuses and produces some other solutions. To realize the diffusion process, SFS uses two Gaussian walks, defined by (1) and (2) and swaps them randomly. Figure 2 illustrates the diffusion process concept;

$$p_i^g = Gaussian(\mu, \sigma) + (r\,BS - r' p_i) \tag{1}$$

$$p_i^g = Gaussian(\mu, \sigma) \tag{2}$$

$$\sigma = \left| \frac{\log(iter)}{iter} (p_i - BS) \right| \tag{3}$$

where μ and σ represent the mean and standard deviation of the Gaussian process, p_i is the seed point, and r and r' are two uniform random number within the range [0,1]. μ takes the best solution (BS) value in the first Gaussian walk, but in the second μ takes the seed point value, p_i.

1st updating process: In the first updating process, a probability value is computed for each candidate solution using (4), in which $rank(p_i)$ represents the rank of the point p_i. Then for each candidate solution with probability less than a uniform random number $Pa_i < r$, its corresponding position is updated with (5).

$$Pa_i = \frac{rank(p_i)}{Np} \tag{4}$$

$$p_i'(j) = p_r(j) - r \times (p_t(j) - p_i(j)) \tag{5}$$

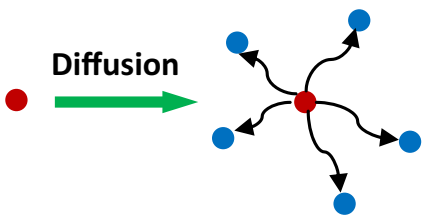

Fig. 2 Diffusion process concept

Where p_r and p_t are two candidate solutions selected from the population. The aim of this process is to change the positions of the bad candidate solutions while keeping the best solutions unchanged, in order to pass the best solutions to the next iteration.

2nd updating process: In this second updating process, the probability values are computed as in the first process, then every solution that satisfies the condition $Pa_i < r$ has its position updated via the inequalities (6) and (7) and the algorithm randomly swaps the solutions.

$$p_i'' = p_i' - r \times (p_t' - BS) \quad if\ r' \leq 0.5 \tag{6}$$

$$p_i'' = p_i' - r \times (p_t' - p_r') \quad if\ r' > 0.5 \tag{7}$$

Where p_r' and p_t' are two candidate solutions selected from the population. The above diffusion and updating processes are repeated until the maximum iteration number is achieved. Algorithm 2 represents the SFS algorithm.

Algorithm 2 SFS algorithm

```
Initialize the maximum number of candidate solutions,
Initialize the maximum number of iterations
Random initialization of candidate solutions
for iter = 1: the maximum number of iterations do
    Fitness evaluation of candidate solutions
    Diffusion process using (1) and (2)
    1st updating process using (5)
    2nd updating process using (6) and (7)
end for
```

4 Block matching algorithms based on stochastic fractal search

The block matching problem can be defined as an optimization problem solved by any metaheuristic technique. Unlike the exhaustive search algorithm that evaluates all possible candidate blocks in the search window, BM algorithm based on metaheuristics randomly initializes some candidates blocks then uses various mathematical tools to converge iteratively to the best matched-block. In this section, a novel BM algorithm based on SFS is described in detail, then a modified version, MSFS-BM, that integrates four new ideas is proposed in order to enhance the SFS-BM results.

4.1 SFS-BM

Our proposed algorithm follows the same steps of SFS. First, a number of candidate blocks are initialized in the search window, and are evaluated with a fitness function, then the SFS's processes are executed through many iterations to converge iteratively to the best-matched block. SFS-BM steps can be detailed as follows:

a) Initialization: Since our algorithm simultaneously computes the motion vectors of all blocks in the current frame, the SFS-BM's initialization has a multi-population model. Equation (8) describes how the initialized multi-population scheme is defined.

$$\begin{bmatrix} CB_{k,s} & CB_{k,s+1} & CB_{k,s+2} & \cdots & CB_{k,S} \\ CB_{k+1,s} & CB_{k+1,s+1} & CB_{k+1,s+2} & \cdots & CB_{k+1,S} \\ CB_{k+2,s} & CB_{k+2,s+1} & CB_{k+2,s+2} & \cdots & CB_{k+2,S} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ CB_{K,s} & CB_{K,s+1} & CB_{K,s+2} & \cdots & CB_{K,S} \end{bmatrix} \tag{8}$$

$CB_{k,s}$ indicates the candidate block number s, $s = 1, 2, \cdots Np$, for the block number k, $k = 1, 2, \cdots K$. Where Np is the maximum number of candidate solutions specified for the resolution of the block matching problem and $K = \frac{M}{L} \times \frac{N}{L}$ represents the maximum number of blocks. In our algorithm, we have initialized twenty candidate blocks, i.e., $Np = 20$, all chosen randomly in the search window.

b) Fitness function: To evaluate the candidate blocks, we have used MSE criterion as a fitness function.

$$fitness = \frac{1}{L \times L} \sum_{i=1}^{L} \sum_{j=1}^{L} (B_{i,j} - CB_{i,j})^2 \tag{9}$$

B represents the current block taken from the current frame t and CB represents the candidate block taken from the reference frame t + 1. In the proposed multi-population scheme, there is no interaction between blocks and the fitness function of each block is defined independently of others.

c) SFS's processes: After the initialization and fitness evaluation, the SFS's processes are executed in order to change the candidate blocks' positions and converge iteratively to the best-matched block.

d) Reducing the computational complexity: To further reduce the computational complexity of SFS-BM, three popular strategies are used:

1. Fitness function values exploitation: Almost all metaheuristics face the same re-evaluation problem. They tend to evaluate some points more than once, which increases the computation time. To avoid this problem, we have assigned a flag to each possible candidate block, this flag is set to 0 until this block is evaluated, then it is set to 1 and

its corresponding fitness value is memorized in a matrix. The algorithm computes the fitness value of a candidate block if its flag is set to 0, otherwise, it takes the value directly from the matrix [13].

2. Zero-motion prejudgment: To reduce the computation time even more, we have exploited the zero-motion prejudgment idea [12]. In the approach, if the matching error measured between the current block and the candidate block located in the same position of the current block is less than a given threshold, the current block is considered static and its motion vector is (0,0), hence no searching is needed.
   The threshold value should be selected from a range of error matching values. We have noticed that the error matching values are within the following ranges:
   - [0.43 − 16] for low motion video sequences,
   - [0.6 − 76] for median motion video sequences,
   - [0.75 − 1358] for high motion video sequences.

   The chosen threshold value is very important as it is used by the algorithm to decide on whether the blocks are static or not. The value of this threshold should not be too large because the algorithm may be mislead into considering the block as static while it is in fact moving. The threshold value should not be very small either as the algorithm is this time mislead into considering the block in motion while it is in fact static; and computation time will unnecessarily be spent in the search for non-existing motion vectors. To avoid this problem, we have chosen a median value suitable for all motion types, which is 1.5.
3. Elimination of worst solutions: At the end of each iteration, the three worst solutions are discarded, in order to further reduce the computational complexity.

e) Termination criterion: To converge to the global best-matched block, the processes are repeated until a termination criterion is achieved. Since our algorithm computes the motion vectors of all blocks simultaneously but independently, there is no interaction between them during the search, we have thus defined two termination criteria;

- Primary criterion: The processes are stopped if all blocks found the best-matched blocks or the maximum number of iterations is attained.
- Secondary criterion: For each block, the search for the best-matched blocks is stopped if the global best fitness value is less than a threshold value. The used threshold is the same as for the Zero-motion prejudgment and is set at 1.5 for similar reasons.

Algorithm 3 depicts the proposed SFS-BM algorithm.

Algorithm 3 SFS-BM algorithm

```
Initialize the maximum number of candidate solutions,
Initialize the maximum number of iterations
Read the sequence video I
Initialize the parameters (window size and block size)
for f = 1: the maximum number of frames do
    Current frame = I(f)
    Reference frame = I(f − 1)or I(f + 1)
    Subdivide the current and reference frames into blocks
    Determine the static blocks
    Random initialization of candidate blocks
    for iter = 1: the maximum number of iterations do
        Fitness evaluation of candidate blocks using (9)
        Diffusion process using (1) and (2)
        1st updating process using (5)
        2nd updating process using (6) and (7)
    end for
    Motion vectors are computed using the best-matched blocks
end for
```

4.2 MSFS-BM

The performances of any block matching algorithms are related to the searching strategy and are also related to the initialized candidate blocks, the used fitness function and the search window size. In this regard, a modified SFS-BM algorithm, MSFS-BM, is presented, in which four ideas are proposed to ameliorate the initialization, the fitness function, the search window size and the fitness approximation method [15].

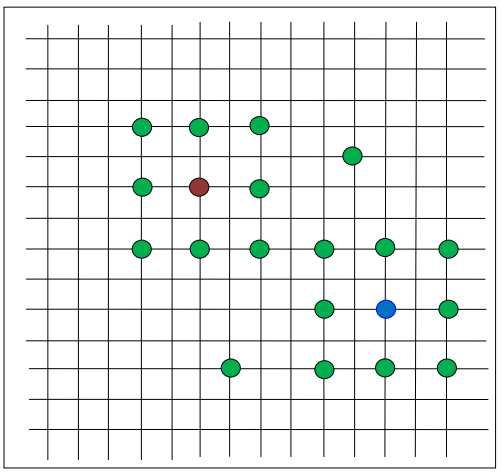

Fig. 3 The initialization fixed pattern

Initialization: Unlike the random initialization used in SFS-BM algorithm, the initialized candidate blocks in MSFS-BM are selected carefully, in order to accelerate the convergence. Twenty candidate blocks are selected as follows:

- Nine candidate blocks, the red dot and the eight green dots surrounding it and shown in Fig. 3, are located in the same positions as the current block and its neighbors, and correspond to the following motion vectors {(0,0); (0,-1);(0,1); (-1,-1); (-1,0); (-1,1); (1,-1); (1,0); (1,1)}. This selection is based on the assumption that the current block moves slowly.
- Nine blocks, the blue dot and the green dots surrounding it, are located in the same positions as that of the predicted block and its neighbors. Equation (10) gives the coordinates of the predicted block using the motion vectors found in the preceding frame.

$$B_{predicted} = B + MV \tag{10}$$

  Figure 4 shows that the predicted block can be very close to the best-matched block. The car headlight surrounded with a blue rectangle in frame t-1 is found in the yellow rectangle in frame t. Using the same motion vector, we predict that the car headlight will be found in a green rectangle in frame t + 1, and it is indeed in the green rectangle. It is noticed that the prediction is more accurate if the moving velocity of the block is constant. To obtain a robust prediction, our modified SFS-BM algorithm starts from the second frame and the motion vectors in the first frame are computed using full search algorithm.
- Two candidate blocks are selected far from the others in order to explore different areas and avoid the local optimums.

Fitness function: To measure the similarity or dissimilarity between two images, various criteria have been formulated throughout the years, with each one having its own strengths and weaknesses [25, 26]. Among these criteria, there is:

- L1 norm, Manhattan norm or Sum of absolute difference (SAD) defined as follows:
  $SAD = \sum_{i=1}^{L}\sum_{j=1}^{L}|B_{i,j} - CB_{i,j}|$
  Where L represent the square block size.
  SAD gives very accurate results in case of a high signal-to-noise ratio (presence of noise).
- Mean Square Error (MSE) measured as follows:
  $MSE = \frac{1}{(L\times L)}\sum_{i=1}^{L}\sum_{j=1}^{L}(B_{i,j} - CB_{i,j})^2$
  MSE is more useful when large errors are particularly undesirable (illumination changes).

We have proposed a new modified fitness function to handle the case of illumination changes or pixels intensities noise which are the only available information. Noise affecting pixels intensities can lead to bad motion estimation results. To get good motion estimation results in all cases, even in presence of noise or illumination changes, the fitness function should be able to evaluate correctly the candidate blocks. The proposed fitness function combines two matching criteria, SAD and MSE, in order to benefit from the strengths of both criteria. The proposed fitness function can be written as:

$$Fitness = \frac{1}{(L\times L)}\sum_{i=1}^{L}\sum_{j=1}^{L}(1-a)\times D^2 + a\times|D| \tag{11}$$

Where

$$D = B_{i,j} - CB_{i,j} \tag{12}$$

a is a weighting parameter fixed at 0.5.

Adaptive window size: The search window size has a direct effect on the motion estimation quality and computational complexity. With a large window size, the motion estimation quality is better but the computational complexity is excessive. Conversely, with a small window size, we can speed up the algorithm but the motion estimation quality will be decreased, especially in the sequences with a high motion. To find the appropriate search window size, many adaptive window size algorithms have been proposed [28,

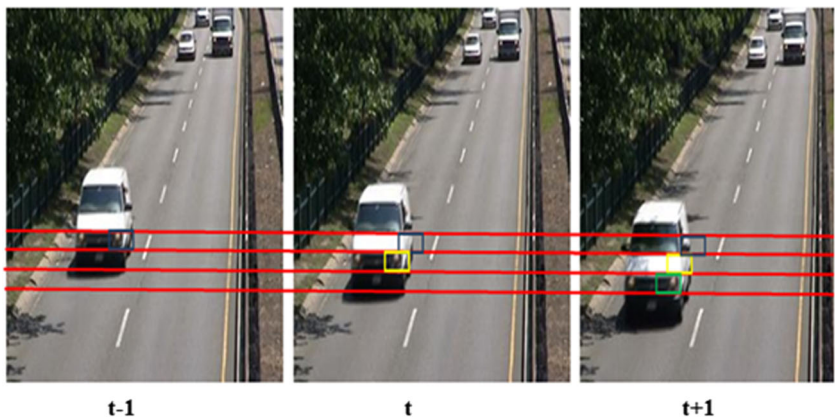

Fig. 4 An example of the movement prediction

29]. In the same context, we propose an algorithm that determines, at each frame, different sizes of the search window for each block. The new horizontal and vertical window sizes of each block are updated using (13) and (14); they are based on the maximum motion vector found in preceding frames and the predicted motion vector of the block.

$$W_{i,f}^{h} = \max(|MV_{f}^{h}|) + |MVP_{i,f}^{h}| \tag{13}$$

$$W_{i,f}^{v} = \max(|MV_{f}^{v}|) + |MVP_{i,f}^{v}| \tag{14}$$

Where f represents the current frame index, MV_{f}^{h} and MV_{f}^{v} are the horizontal and vertical motion vectors found in the preceding frames. $MVP_{i,f}^{h}$ and $MVP_{i,f}^{v}$ are the predicted motion vectors of block i. The predicted motion vectors of block i could be sufficient to determine the new window sizes, but as mentioned before, the predicted motion vectors of a block are accurate only if its velocity is constant. To take into account the cases of a sudden increase of velocity, we have added the maximum motion vector found in the sequence. To ensure that the window sizes are not larger than a maximum window size, already defined as $w_{\max}$, and not less than a minimum window size, $w_{\min} = 1$, we used the following equations:

$$W_{i,f}^{h} = \max(\min(W_{i,f}^{h}, w_{\max}), w\_min) \tag{15}$$

$$W_{i,f}^{v} = \max(\min(W_{i,f}^{v}, w_{\max}), w_{\min}) \tag{16}$$

Modified fitness approximation method: To reduce the number of fitness evaluation, a fitness approximation method based on the nearest-neighbor-interpolation method has been proposed in the literature [15]. In this method, some candidate blocks are evaluated through the real fitness function while the other candidate blocks are only estimated based on previously evaluated neighboring candidate blocks. Algorithm 4 is used to decide on the candidate blocks to be evaluated. For each new candidate block P_n, the algorithm first computes the distance between this candidate block and the nearest block P_v, P_v is a candidate block already evaluated and its fitness value is F_v. If the computed distance is less than a predefined value d, the algorithm checks the F_v value. If F_v corresponds to the best fitness value found up to the current iteration, the new candidate block P_n is evaluated through the real fitness function, otherwise it is estimated by assigning to F_n the same fitness value of F_v. If P_n is longer than a distance d to the nearest block P_v, this block is directly evaluated.

Algorithm 4 Fitness approximation method

```
if A new candidate block P_n is closer than a distance d to the nearest block P_v then
    if F_v correspond to the best fitness value then
        P_n is evaluated though the real fitness function
    else
        F_n = F_v
    end if
else
    P_n is evaluated though the real fitness function
end if
```

To reduce the computational complexity of block matching algorithms another method called the successive elimination algorithm (SEA) has already been proposed [30]. In this method the search is performed only on the blocks which satisfy the following condition;

$$S_{current} - SAD(u,v) \leq S_{candidate} \leq S_{current} + SAD(u,v) \tag{17}$$

Where $S_{current}$ and $S_{candidate}$ represent the sum of all pixel intensities in current and candidate blocks respectively.

$$S_{current} = \sum_{i=1}^{L}\sum_{j=1}^{L} current\ Block(i,j)$$

$$S_{candidate} = \sum_{i=1}^{L}\sum_{j=1}^{L} candidate\ Block(i,j)$$

SAD(u,v) expresses the sum of absolute differences between the current block and the best candidate block found up to the current iteration.

Returning to the fitness approximation method [15], the candidate blocks which are closer than a distance d to the nearest block are evaluated if the nearest block is the best block, and estimated otherwise. However, if the global best block is not close to the best block found up to the current iteration, the algorithm could be trapped into a local optimum. Therefore we have proposed to replace this condition by the successive elimination algorithm condition. The modified fitness approximation is presented in Algorithm 5, and Algorithm 6 represents the MSFS-BM algorithm.

Algorithm 5 Modified Fitness approximation method

```
if A new candidate block P_n is closer than a distance d to the nearest block P_v then
    if S_current − SAD(u,v) ≤ S_candidate ≤ S_current + SAD(u,v) then
        P_n is evaluated though the real fitness function
    else
        F_n = F_v
    end if
else
    P_n is evaluated though the real fitness function
end if
```

Algorithm 6 MSFS-BM algorithm

```
Initialize the maximum number of candidate solutions,
Initialize the maximum number of iterations
Read the sequence video I
Initialize the parameters (window size and block size)
for f = 1: the maximum number of frames do
    Current frame = I(f)
    Reference frame = I(f − 1) or I(f + 1)
    Subdivide the current and reference frames into blocks
    Determine the static blocks
    if f=1 then
        Motion vectors are computed using FSA
    else
        Initialization of candidate blocks using the fixed pattern (Fig. 3)
        for iter = 1: the maximum number of iterations do
            Fitness evaluation using (11) or Fitness approximation
            Diffusion process using (1) and (2)
            1st updating process using (5)
            2nd updating process using (6) and (7)
        end for
        Motion vectors are computed using the best-matched blocks
    end if
    Determine the predicted movement for each block using (10)
    Update the window sizes for each block using (13–16)
end for
```

5 Results and discussion

In our simulations, we have computed the motion vectors in nine video sequences of different formats and motion types. The used video sequences and their characteristics are listed in Table 1 and illustrated in Fig. 5. The links from which the used video sequences are downloaded are given in the Appendix.

This section is divided into two parts. In the first part, we analyse the impacts of each proposed idea on MSFS-BM performances. In the second part, we compare the results of our algorithms to those obtained with different state-of-the-art BM algorithms.

In our algorithm, the total number of candidate blocks is 20 blocks. After several executions, we noticed that the global optimum solution can be obtained from the 3rd iteration. For this reason, we have fixed the maximum number of iterations at 3 iterations. To test the algorithm's performances, the following metrics are used:

1. *Motion estimation accuracy*: It consists of two measures
   - PSNR that indicates the quality of the compensated frame, which is constructed by the best-matched blocks found with BM algorithm;

$$PSNR = 10 \times \log\left(\frac{I_{\max}}{Error}\right) \tag{18}$$

$$Error = \frac{1}{M \times N}\left(\sum_{i=1}^{M}\sum_{j=1}^{N} Current\ frame_{i,j} - Compensated\ frame_{i,j}\right)$$

   Where M and N represent the frame's horizontal and vertical lengths and $I_{\max} = 255$.
   - D_{PSNR} which measures the degradation in PSNR compared to the results of FSA,

$$D_{PSNR} = -\left(\frac{PSNR_{FSA} - PSNR_{BMA}}{PSNR_{FSA}}\right) \times 100\% \tag{19}$$

2. *The Average number of searched points*: It represents the average number of evaluated candidate blocks per block.

Table 1 The used video sequences and their characteristics

Sequence	Format	Frame size	Motion type	Total frames	Maximum window size
Akiyo	Qcif	(144 × 176)	Low	300	8
Grandma	Qcif	(144 × 176)	Low	870	8
Miss	Qcif	(144 × 176)	Median	150	8
Salesman	Qcif	(144 × 176)	Median	449	8
Hall	Cif	(288 × 352)	High	300	8
Container	Qcif	(144 × 176)	Low	300	8
Carphone	Qcif	(144 × 176)	High	382	8
Foreman	Qcif	(144 × 176)	High	300	8
Stefan	Cif	(288 × 352)	Very High	90	16

Fig. 5 The video sequences used in the simulation

Table 2 Improvement percentages of the proposed ideas

Algorithms		Sequences					Improvement %
		Akiyo	Grandma	Miss-america	Salesman	Hall	
SFS-BM	PSNR	32.01	32.66	32.18	29.50	25.56	
	NSP	5.02	8.69	12.26	11.25	31.30	
Initialization	PSNR	44.28	43.42	41.33	40.32	34.81	34.51
fixed pattern	NSP	2.42	4.12	6.28	5.21	15.72	51.32
Proposed	PSNR	32.11	32.70	32.22	29.55	25.57	0.15
fitness function	NSP	5.01	8.67	12.28	11.24	31.28	0.08
Proposed adaptive	PSNR	42.86	32.70	32.73	30.50	25.60	7.85
window size	NSP	0.63	8.65	11.90	9.97	31.30	20.44
Fitness	PSNR	30.49	31.15	30.73	28.03	23.71	− 5.21
approximation [15]	NSP	2.22	3.83	5.27	4.82	13.39	56.61
Modified fitness	PSNR	31.99	32.66	32.11	29.54	25.53	− 0.05
approximation	NSP	4.96	5.53	12.04	11.06	30.90	8.46
MSFS-BM	PSNR	44.28	43.42	41.33	40.32	34.80	34.50
	NSP	1.41	3.82	5.70	4.69	15.45	58.08

Table 3 The PSNR values obtained with different fast BM algorithms

Algorithms		Sequences					Average D_{psnr}	Rank
		Akiyo	Grandma	Miss-america	Salesman	Hall		
FSA	PSNR	44.29	43.43	41.33	40.32	34.81	0	1
	D_{psnr}	0	0	0	0	0		
TSS	PSNR	44.29	43.42	41.27	40.30	34.76	− 0.07	6
	D_{psnr}	0	− 0.02	− 0.14	− 0.04	− 0.14		
NTSS	PSNR	44.29	43.43	41.33	40.31	34.776	− 0.02	3
	D_{psnr}	0	0	0	− 0.02	− 0.11		
SES	PSNR	44.29	43.42	41.15	40.26	34.72	− 0.17	7
	D_{psnr}	0	− 0.02	− 0.43	− 0.14	− 0.25		
4SS	PSNR	44.29	43.42	41.27	40.30	34.76	− 0.07	5
	D_{psnr}	0	− 0.02	− 0.14	− 0.04	− 0.14		
DS	PSNR	44.29	43.42	41.33	40.31	34.77	− 0.03	4
	D_{psnr}	0	− 0.02	0	− 0.02	− 0.11		
MSFS-BM	PSNR	44.28	43.42	41.33	40.32	34.80	− 0.01	2
	D_{psnr}	− 0.02	− 0.02	0	0	− 0.02		

Our goal is to obtain a higher PSNR value, hence a lower D_{PSNR} and to speed up the algorithm by decreasing the average number of searched points.

5.1 Analysis of MSFS-BM algorithm

In MSFS-BM four ideas are proposed to enhance the results of SFS-BM algorithm. To show the effectiveness of each idea alone, we have reported in Table 2 the results of SFS-BM, the results of SFS-BM when a new idea is added and the results of MSFS-BM that combines all the proposed ideas. Table 2 also provides the percentage improvement in PSNR and average number of searched points. The percentage improvement of SFS-BM with new idea integrated (ALG2) over the original SFS-BM (ALG1) is measured as follows:

The improvement in PSNR is measured by:

$$Improvement_{PSNR} = -\left(\frac{PSNR_{ALG2} - PSNR_{ALG1}}{PSNR_{ALG2}}\right) \times 100\% \quad (20)$$

The improvement in the number of searched points is measured by:

$$Improvement_{NSP} = -\left(\frac{NSP_{ALG2} - NSP_{ALG1}}{NSP_{ALG2}}\right) \times 100\% \quad (21)$$

Where NSP is the average number of searched points.

Starting from the results reported in Table 2, the proposed initialization pattern enhanced the results of SFS-BM with significant percentage improvements as high as 34% in PSNR and 51% in the number of searched points. Which proves the effectiveness of the proposed initialization pattern.

Table 4 The number of searched points obtained with different BM algorithms

Algorithms	Sequences					Average search points	Rank
	Akiyo	Grandma	Miss-america	Salesman	Hall		
FSA	236.63	236.63	236.63	236.63	262.62	241.82	7
TSS	21.48	21.51	21.50	21.48	23.25	21.84	6
NTSS	14.68	15.05	15.66	14.71	16.92	15.40	4
SES	16.20	16.17	16.04	16.19	16.92	16.30	5
4SS	14.65	14.84	14.95	14.67	16.25	15.07	3
DS	11.43	11.69	12.01	11.45	12.89	11.89	2
MSFS-BM	1.41	3.82	5.70	4.69	15.45	6.21	1

Table 5 The PSNR values obtained with different metaheuristics and learning automata BM algorithms

Algorithms		Sequences				Average D_{psnr}	Rank
		Container	Carphone	Foreman	Stefan		
FSA	PSNR	43.18	31.51	31.69	25.95	0	1
	D_{psnr}	0	0	0	0		
PSO-BM	PSNR	43.15	31.39	31.27	25.39	− 0.98	4
	D_{psnr}	− 0.07	− 0.38	− 1.34	− 2.15		
DE-BM	PSNR	43.17	31.47	31.51	25.15	− 0.96	3
	D_{psnr}	− 0.04	− 0.13	− 0.58	− 3.09		
HS-BM	PSNR	43.16	31.40	31.55	25.055	− 1.07	7
	D_{psnr}	− 0.05	− 0.34	− 0.45	− 3.46		
PLA-BM	PSNR	43.18	31.42	31.43	25.12	− 1.07	6
	D_{psnr}	0	− 0.28	− 0.82	− 3.19		
TPLA-BM	PSNR	43.18	31.42	31.42	25.22	− 1.01	5
	D_{psnr}	0	− 0.28	− 0.85	− 2.92		
MSFS-BM	PSNR	43.18	31.51	31.69	25.42	− 0.51	2
	D_{psnr}	0	0	0	− 2.04		

The results also reveal the capacity of the proposed fitness function to ameliorate the SFS-BM results, in estimation accuracy and computation time with percentage improvements of 0.15% and 0.08% respectively. So the combination of two matching criteria can effectively enhance the results.

The proposed adaptive window size strategy gives a percentage improvement in the number of searched points that exceeds 20%. Despite the fact that the proposed strategy decreased the number of searched points, it has nevertheless improved the PSNR values.

A comparison is then drawn between the results of the proposed modified fitness approximation method and the fitness approximation method. It is clear from the results of the comparison that the proposed modified fitness approximation method can reduce the number of searched points with a lower degradation in PSNR values.

Finally, the overall MSFS-BM results are reported and indicate significant improvement of MSFS-BM over SFS-BM with percentages of 34.50% in PSNR and 58.08% in the number of searched points.

From this analysis it can be concluded that the proposed ideas have the capability to improve the estimation accuracy as well as the computational complexity of the SFS-BM algorithm.

5.2 Comparison of MSFS-BM with several state-of-the-art algorithms

The performances of our proposed algorithm MSFS-BM are compared with different state-of-the-art BM algorithms.

Table 6 The number of searched points obtained with different metaheuristics and learning automata BM algorithms

Algorithms	Sequences				Average searched points	Rank
	Container	Carphone	Foreman	Stefan		
FSA	236.63	236.63	236.63	984.91	423.7	7
PSO-BM	32.5	48.5	48.1	52.2	45.32	6
DE-BM	9.2	12.2	12.5	16.1	12.50	4
HS-BM	8	12.5	12.2	17.1	12.45	3
PLA-BM	75.27	71.54	77.5	131.7	89.00	5
TPLA-BM	10.85	9.17	10.09	13.9	11.00	2
MSFS-BM	1.41	7.52	14.62	19.65	10.8	1

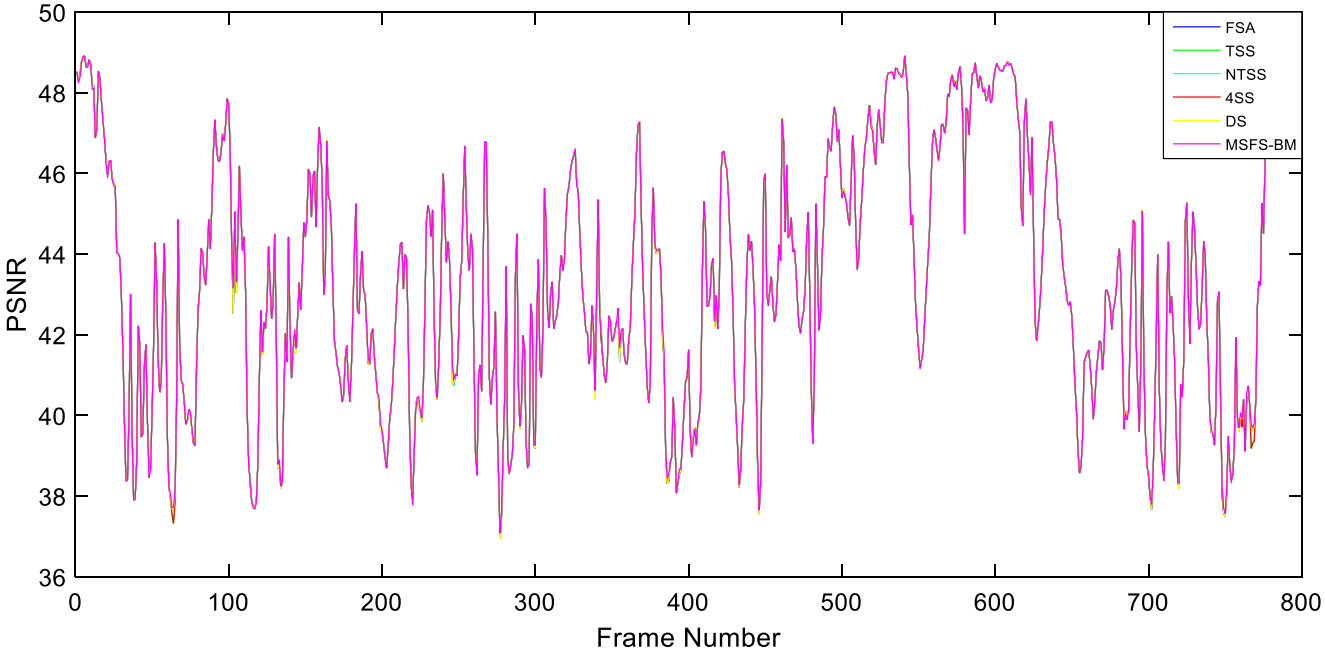

Fig. 6 PSNR values measured in Grandma sequence

They are first compared with FSA and different fast BM algorithms such as the three step search (TSS) [31], the new TSS (NTSS) [32], the simple and efficient TSS (SES) [33], the four step search (4SS) [34] and the diamond search (DS) [35]. MSFS-BM is then compared with three metaheuristics BM algorithms, PSO-BM [14], DE-BM [15] and HS-BM [16], and two learning automata BM algorithms, PLA-BM and TPLA-BM [17].

Tables 3 and 4 present the results of PSNR and the number of searched points obtained using MSFS-BM and different fast BM algorithms.

Tables 5 and 6 present the results of PSNR and the number of searched points obtained using MSFS-BM and different metaheuristics and learning automata BM algorithms.

The results reported in Tables 3 to 6, indicate that the MSFS-BM algorithm can obtain higher PSNR values

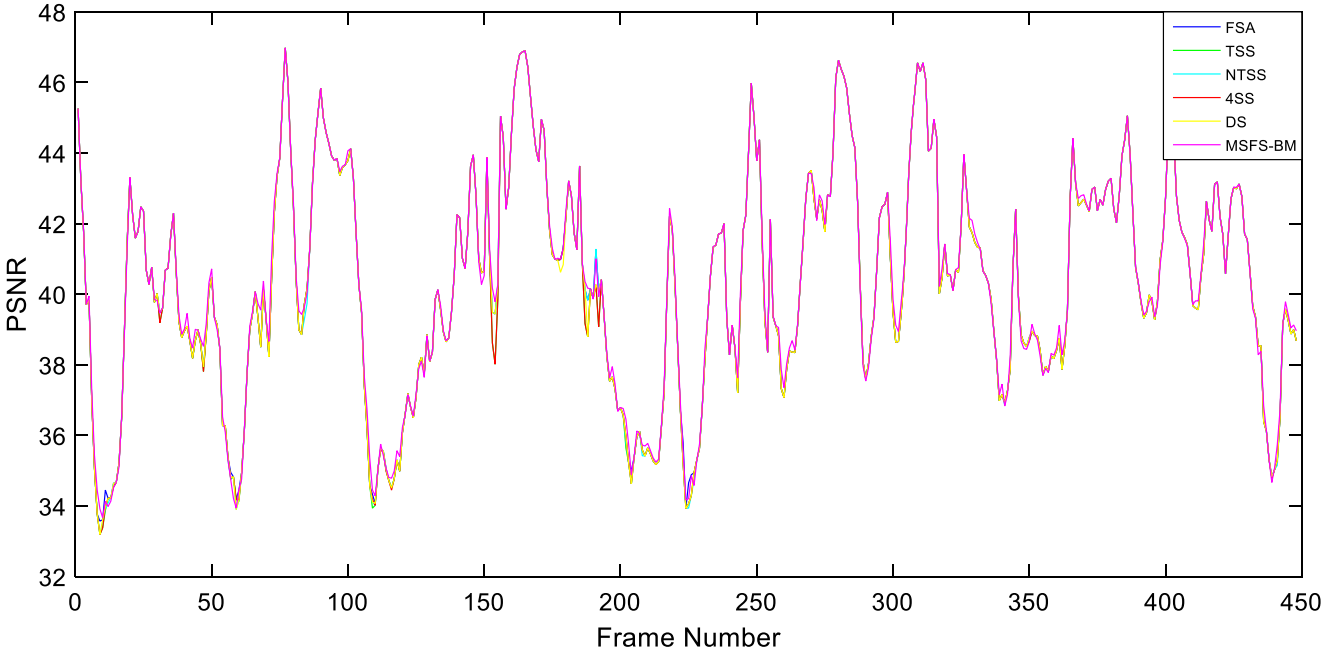

Fig. 7 PSNR values measured in Salesman sequence

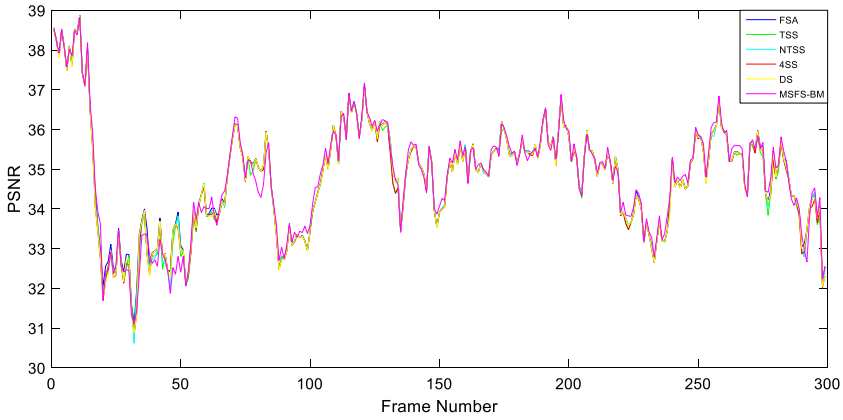

Fig. 8 PSNR values measured in Hall sequence

with a lower number of searched points; its results lie in the intervals $[-0.01\ -0.51]$ and [6.21 10.8] for the D_{PSNR} and the number of searched points respectively. It is therefore ranked in the first place outperforming all the cited algorithms.

In Figs. 6, 7 and 8, we have plotted, frame-by-frame, the PSNR values obtained with FSA, TSS, NTSS, 4SS, DS and MSFS-BM algorithms for three different sequences (Grandma, Salesman, and Hall). It can be seen that the magenta curve, that represents the MSFS-BM's PSNR values is above the other curves for most frames and close to the blue curve that represents FSA's PSNR values.

From the comparison, it is clear that the proposed algorithm MSFS-BM is a powerful BM algorithm that is capable of producing higher PSNR values, very close to those given by FSA, while requiring a minimal number of searched points.

6 Conclusion

In this paper, we have presented a new BM algorithm based on SFS algorithm. The proposed algorithm, SFS-BM, uses a multi-population model in order to compute the motion vectors of all blocks simultaneously. Furthermore, in order to enhance SFS-BM's results, a modified version of SFS-BM, MSFS-BM that integrates four new ideas, is proposed. The random initialization is replaced by a fixed pattern and a combination of two matching criteria is used as a new fitness function. The considered search space is controlled by a new adaptive window size strategy, and a modified fitness approximation method is proposed. In the experimental results section, these ideas were examined carefully, then MSFS-BM was compared with several state-of-the-art algorithms such as TSS, FSS, DS, DE-BM and HS-BM, etc. The results have shown that the proposed ideas can very well improve the SFS-BM results. The initialization fixed pattern provided a significant percentage improvement of up to 34.51% in the PSNR and 51.32% in the NSP respectively. The proposed fitness function enhanced the results with percentage improvements of 0.51% in PSNR and 0.08% in NSP. The adaptive widow size has also ameliorated the results with up to 7.58% in PSNR and 20.44% in NSP. Finally, the modified fitness approximation method has given an improvement of 8.46% in NSP but lead to a somewhat insignificant degradation of the PSNR not exceeding -0.05%. The results show also that MSFS-BM outperforms all the cited algorithms with higher PSNR values close to those of FSA, requiring a minimal number of searched points and ranging from 6.21 to 10.8. The proposed work can be further enhanced with the use of other metaheuristics to solve the BM problem. Another interesting perspective for further research would be the use of the proposed new ideas in other BM algorithms.

Appendix

The video sequences used in this paper were downloaded from these addresses:

http://trace.kom.aau.dk/yuv/index.html
https://media.xiph.org/video/derf/

The code sources used in this paper were downloaded from these addresses:

https://www.mathworks.com/matlabcentral/fileexchange/8761-block-matching-algorithms-for-motion-estimation

References

1. Fortun D, Bouthemy P, Kervrann C (2015) Optical flow modeling and computation: a survey. Comput Vis Image Underst 134:1–21
2. Ilg E, Mayer N, Saikia T et al (2016) Flownet 2.0: Evolution of optical flow estimation with deep networks. arXiv:1612.01925
3. Chen Q, Koltun V (2016) Full flow: Optical flow estimation by global optimization over regular grids. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4706–4714

4. Palomares RP, Meinhardt-Llopis E, Ballester C, ohers (2017) FALDOI: A new minimization strategy for large displacement variational optical flow. J Math Imaging Vision 58(1):27–46
5. Horn BKP, Schunck BG (1981) Determining optical flow. Artif Intell 17(1–3):185–203
6. Metkar S, Talbar S (2013) Performance evaluation of block matching algorithms for video coding. In: Motion estimation techniques for digital video coding. Springer, India, pp 13–31
7. Furht B, Greenberg J, Westwater R (2012) Motion estimation algorithms for video compression. Springer Science, Business Media
8. Terki N, Saigaa D, Cheriet L et al (2013) Fast motion estimation algorithm based on complex wavelet transform. Journal of Signal Processing Systems 72(2):99–105
9. Barjatya A (2004) Block matching algorithms for motion estimation. IEEE Trans Evol Comput 8(3):225–239
10. Choudhury HA, Saikia M (2014) Survey on block matching algorithms for motion estimation. In: 2014 international conference on communications and signal processing (ICCSP). IEEE, pp 036–040
11. Li S, Xu W-P, Wang H et al (1999) A novel fast motion estimation method based on genetic algorithm. In: 1999 international conference on image processing, 1999. ICIP 99. Proceedings. IEEE, pp 66–69
12. Ren R, Shi Y, Zheng B et al (2006) A fast block matching algorithm for video motion estimation based on particle swarm optimization and motion prejudgment. arXiv:cs/0609131
13. Cai J, Pan WD (2012) On fast and accurate block-based motion estimation algorithms using particle swarm optimization. Inf Sci 197:53–64
14. Yuan X, Shen X (2008) Block matching algorithm based on particle swarm optimization for motion estimation. In: International conference on embedded software and systems, 2008. ICESS'08. IEEE, pp 191–195
15. Cuevas E, Zaldivar D, Pérez-Cisneros M et al (2013) Block-matching algorithm based on differential evolution for motion estimation. Eng Appl Artif Intell 26(1):488–498
16. Díaz-Cortés M-A, Cuevas E, Rojas R (2017) Motion estimation algorithm using block-matching and harmony search optimization. In: Engineering applications of soft computing. Springer International Publishing, pp 13–44
17. Damerchilu B, Norouzzadeh MS, Meybodi MR (2016) Motion estimation using learning automata. Mach Vis Appl 27(7):1047–1061
18. Zhang J, Wang C, Zhou M (2015) Fast and epsilon-optimal discretized pursuit learning automata. IEEE Transactions on Cybernetics 45(10):2089–2099
19. Salimi H (2015) Stochastic fractal search: a powerful metaheuristic algorithm. Knowl-Based Syst 75:1–18
20. Chuan SUN, Wei Z-Q, Zhou C-J et al (2016) Stochastic fractal search algorithm for 3d protein structure prediction. DEStech Transactions on Computer Science and Engineering, no aics
21. Rahman TAZ (2016) Parameters optimization of an SVM-classifier using stochastic fractal search algorithm for monitoring an aerospace structure
22. Sivalingam R, Chinnamuthu S, Dash SS (2017) A hybrid stochastic fractal search and local unimodal sampling based multistage PDF plus (1 + PI) controller for automatic generation control of power systems. Journal of the Franklin Institute
23. Parejo Maestre JA, Ruiz Cortés A, Lozano Segura S et al (2012) Metaheuristic optimization frameworks: a survey and benchmarking. Soft Comput 16(3):1–35
24. Boussaïd I, Lepagnot J, Siarry P (2013) A survey on optimization metaheuristics. Inf Sci 237:82–117
25. Deviant S. (2011) The practically cheating statistics handbook–. https://Lulu.com
26. Goshtasby AA (2012) Image registration: principles, tools and methods. Springer Science, Business Media
27. Smith SW et al (1997) The scientist and engineer's guide to digital signal processing
28. Feng J, Lo K-T, Mehrpour H et al (1995) Adaptive block matching motion estimation algorithm for video coding. Electron Lett 31(18):1542–1543
29. Oh H-S, Park G, Lee H-K (1997) Block-matching algorithm based on dynamic search window adjustment. Dept. of CS, KAIST
30. Li W, Salari E (1995) Successive elimination algorithm for motion estimation. IEEE Trans Image Process 4(1):105–107
31. Jong H-M, Chen L-G, Chiueh T-D (1994) Accuracy improvement and cost reduction of 3-step search block matching algorithm for video coding. IEEE Trans Circuits Syst Video Technol 4(1):88–90
32. Li R, Zeng B, Liou ML (1994) A new three-step search algorithm for block motion estimation. IEEE Trans Circuits Syst Video Technol 4(4):438–442
33. Lu J, Liou ML (1997) A simple and efficient search algorithm for block-matching motion estimation. IEEE Trans Circuits Syst Video Technol 7(2):429–433
34. Po L-M, Ma W-C (1996) A novel four-step search algorithm for fast block motion estimation. IEEE Trans Circuits Syst Video Technol 6(3):313–317
35. Zhu S, Ma K-K (1997) A new diamond search algorithm for fast block matching motion estimation. In: Proceedings of 1997 international conference on information, communications and signal processing, 1997. ICICS. IEEE, pp 292–296

Abir Betka was born in Biskra, Algeria. She received her Master's degree in Signal and Communication from the University of Biskra, Algeria, in 2015. She is currently pursuing her PhD in Signal and Communication at the same university. Her research interests include Image processing, motion estimation, Optimization, Metaheuristics, nature-inspired techniques.

Nadjiba Terki was born in June 12, 1971 Algeria. She received her Engineering degree in Automatics, a Master's degree in Non-Destructive Control and a PhD degree in Signal Processing from Badji Mokhtar University of Annaba, Algeria, in 1994, 2000 and 2009, respectively. In 2001, she joined Mohamed Khider University of Biskra, Algeria, where she currently works as an Associate Professor in the department of Electrical Engineering. Her research interests include Digital signal and Image processing, artificial intelligence, wavelet transform and robust control.

Abida Toumi MCA, received her BEng. in Electrical and Electronic Engineering from Mohamed Khider University of Biskra, Algeria, in 1995, her Master's degree in Electrical and Electronic Engineering from Farhat ABASS University of Sétif , Algeria, in 2002 and the grade of Doctor of Science in Electrical and Electronic Engineering from Mohamed Khider University of Biskra, Algeria. She is now a research professor at LESIA Laboratory in Methaheuristics, Systems, Signal and Image Processing.

Madina Hamiane received her DES in Electronics from the University of Science and Technology Houari Boumediene (USTHB), Algeria; her MSc degree in Cybernetics and Instrument Technology from the University of Reading, UK, and her PhD degree in Control Engineering from the University of Sheffield, UK. She is now with the College of Engineering at Ahlia University in the Kingdom of Bahrain. Dr. Hamiane's current research interests span applications of signal processing and pattern recognition, biomedical signal and image processing, and applications of intelligent control.

Amina Ourchani was born in Biskra, Algeria, in 07 December 1987. In 2011 she received a master degree in communication signal and network from the university Mohemed Khider Biskra. She is currently a PhD student in university of mohamed khider, she is a member in LESIA research laboratory. Her research interests include image processing, image segmentation and motion detection.