Yu et al. EURASIP Journal on Image and Video Processing (2017) 2017:66
DOI 10.1186/s13640-017-0214-1

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

A novel search method based on artificial bee colony algorithm for block motion estimation

Weiyu Yu[1*], Dan Hu[1], Na Tian[1] and Zhili Zhou[2]

Abstract

The large amount of bandwidth that is required for the transmission or storage of digital videos is the main incentive for researchers to develop algorithms that aim at compressing video data while keeping their quality as high as possible. Block matching has been extensively utilized in compression algorithms for motion estimation as they reduce the memory requirements of any video file. A novel method by using modified artificial bee colony algorithm (MABCA) is proposed, which is the combination of basic ABCA and several techniques—initialization based on temporal-spatial correlation, duplicate searching avoidance technique, adaptive search criteria for iteration, and early termination of block matching. Experimental results show that the proposed method has achieved significant improvement over the existing popular block-matching methods in terms of estimation accuracy and computational complexity, especially when dealing with sequences with violent motion. This method can also adaptively adjust the average search point number according to the motion intensity of the different sequences in order to get the best search result at the lowest possible cost.

Keywords: Artificial bee colony algorithm, Motion estimation, Fast block-matching method

1 Introduction

Motion compensation has been adopted by all of the existing video-coding standards, such as the MPEG-1, MPEG-2, and MPEG-4 and ITU-T H.261, H.264, and H.265, owing to its high compression efficiency. Video redundancy is mainly from spatial (intra-frame) and temporal (inter-frame) correlations. Motion estimation and compensation are used to remove the data redundancy and the unimportant visual details. It is apparent that the changes in a scene are due to object motion. Even minute movement can still lead to a great difference between successive frames, especially when movement is near the edge of the moving object. However, prediction residual can be reduced by using a region with a certain displacement in the previous frame to predict the current region in the current frame. Such method taking advantage of the motion vector of the offset to predict the new scene is called motion compensation (MC) [1]. And the process for the encoder to search for the best motion vector is called motion estimation (ME) [2]. There are generally two kinds of motion estimation: block-matching algorithm (BMA) and pixel-based algorithm (PBA). BMA is simple and efficient. A large number of researchers employed and improved BMA in their lossless video compression schemes. The motion estimation prediction difference is encoded by a method called motion compensation prediction residual coding (MCPRC). Temporal redundancy between successive frames allows the ME and MC techniques to play an active role in video compression coding [3, 4]. Motion estimation is the process of determining motion vectors that describe the transformation from one image to another, while motion compensation [5] describes a picture in terms of the transformation of a reference frame to the current frame. Therefore, the more accurate the motion vectors are yielded by ME, the less forecast error there will be. And finally, the amount of transmission information will be largely reduced. For the foregoing reasons, motion estimation has been a critical technique in promoting the video compression ratio.

* Correspondence: yuweiyu@scut.edu.cn
[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China
Full list of author information is available at the end of the article

There are three basic types of motion estimation: the BMA, the pel-recursive method, and the model-based method [6]. Among them, the block-matching approach is more widely employed in motion estimation, because it is more practical and adaptable. The purpose of a BMA is to find the displacements of the blocks in the current frame according to the reference frame [7, 8]. It can be used to remove temporal redundancy in the video sequence, increasing the video compression effectiveness.

The BMA is described as follows: Each luma frame is divided into non-overlapping blocks of size $N \times M$, and each block in the current frame is matched with the candidate blocks of size $N \times M$ within the search window in the reference frame, as shown in Fig. 1. The best matched block has the lowest distortion among all of the candidate blocks. The displacement of the best matched block or namely the motion vector of current block will be transmitted with prediction residues to the decoder.

In the whole process, several criteria are used to determine whether a given block in current frame matches the search block in reference frame, such as mean absolute error (MAE), mean square error (MSE), and sum absolute difference (SAD).

The computational complexity of a motion estimation technique can be determined by three factors: (1.) search algorithm, (2.) cost function, and (3.) search range parameter p. Actually, we can reduce the complexity of the motion estimation algorithms by reducing the complexity of the applied search algorithm and/or the complexity of the selected cost function.

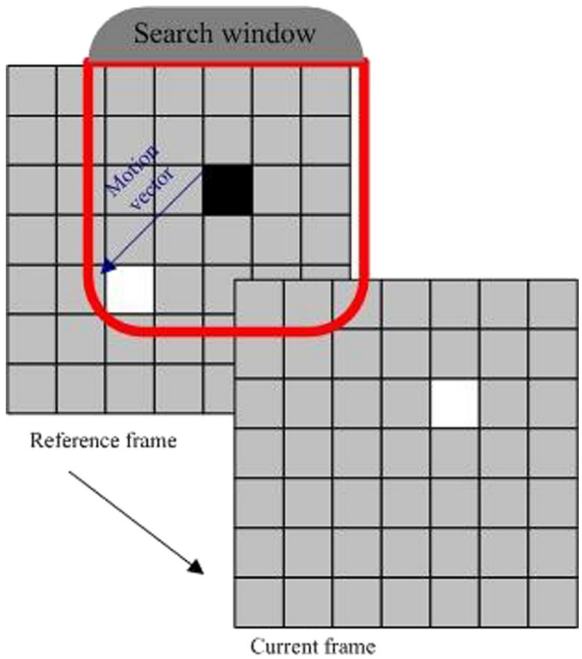

Fig. 1 Block-matching motion estimation

For the past decade, many fast BMA motion estimation methods have been proposed, such as full search (FS) [9], three-step search (TSS) [10], new three-step search (NTSS) [11], simple and efficient search (SES) [12], four-step search (FSS) [13], diamond search (DS) [14], dynamic search window adjustment and interlaced search (DSWA/IS) algorithm [15], and nearest neighbors search algorithm [16]. The main concepts of fast algorithms can be classified into six categories: reduction in search positions, simplification of matching criterion, bit width reduction, predictive search, hierarchical search, and fast full search. Full search method can guarantee the best motion vector and be implemented in hardware easily. However, the full search method just simply searches all the possible points within a search window, which also makes it not very practical for its enormous computational complexity. The other methods can achieve adequate reconstructed image quality with less average search point number for most of the images with smooth motion. The TSS is the most popular algorithm. While the FSA requires, for a standard window size of ±7 pixels, 225 comparisons to find the best match, the TSS algorithm requires only 25 comparisons. The TSS algorithm, however, uses a uniformly allocated checking point pattern, making it inefficient for searching small motion video sequences. The NTSS algorithm is an altered version of the existing TSS algorithm. While the TSS algorithm requires predefined checking point pattern, the NTSS algorithm uses center-biased checking point pattern, by making the search adaptive to the motion vector distribution. In NTSS, a halfway-stop technique is used to reduce the computation cost. SES aims at further reducing the computational complexity to speed up the TSS by a factor of two and maintain its regularity and good performance comparable with the TSS. The FSS produces better performance than the TSS and has similar performance as compared with the NTSS, while reducing the worst-case computational search point number from 33 to 27 and the average search point number from 21 to 19 as compared with NTSS. The DS algorithm can deal with both small and large motion image sequences with smaller motion estimation error and search point number than other fast search algorithms like TSS, NTSS, SES, and FSS which might only work well for the image sequences with certain degree of motion. The DSWA algorithm adaptively adjusts the size of the search window based on the best match position, while in the IS algorithm, only five positions are alternatively tested. The DSWA/IS algorithm presents a more intelligent logarithmic step search with variable patterns. The nearest neighbors search algorithm [16] presents a new motion estimation

algorithm for low bit rate video coding, where each motion vector is predicted from its neighboring (already coded) motion vectors. It uses the (+) cross as a search pattern. The algorithm terminates only when the best match is found at the center or when the window boundary is reached. A modified one-at-a-time search algorithm [17] modifies the existing one-at-a-time search algorithm [18]. The modification is based on using the inter-block matching between the motion vectors and the adjacent blocks. The algorithm improves the quality and reduces the number of comparisons required to obtain the best match. A Polynomial Approximation Motion Estimation Model for motion-compensated frame interpolation is proposed [19, 20]. Adaptive low-complexity motion estimation algorithm for high efficiency video coding encoder is presented [21] in order to speed up integer ME in high efficiency video coding. This study presents a fast center search algorithm (FCSA) and an adaptive search window algorithm (ASWA) for integer pixel ME. In addition, center adaptive search algorithm, a combination of the two proposed algorithms FCSA and ASWA, is proposed in order to achieve the best performance. Nevertheless, all these fast search methods typically suffer from different degrees of loss in the image quality. This can be attributed to the demand for less time complexity which prelimits the candidate points to be only a very small fraction of the points in the search window, consequently leaving the search with a great possibility of being trapped in local optima when processing violent motion. Motion estimation is facing a dilemma of reducing the computational complexity at the cost of worsening the estimation accuracy. Therefore, it is very necessary to find a balance between the accuracy and computational complexity.

In recent years, optimization algorithms such as the genetic algorithm (GA) and the particle swarm optimization (PSO) have been successfully applied to motion estimation [22–24]. However, these algorithms still suffer from the problem of local optima sticking. Meanwhile, artificial bee colony algorithm (ABCA) has emerged in the field of swarm intelligence [25–27], with the ability to effectively avoid falling into local optimum. It has already been applied to various problem domains, such as image segmentation [28–30], data clustering [31], and digital filters [32] and has made significant performances. Hence, in this paper, a new block-matching algorithm for motion estimation using modified ABCA is proposed. The modified ABCA is the combination of basic ABCA and several techniques—initialization based on temporal-spatial correlation, duplicate searching avoidance technique, adaptive search criteria for iteration, and early termination of block matching. Compared with other six motion estimation algorithms on 10 different video sequences, experimental results show that the proposed method could achieve significant improvements over existing fast block search methods in estimation accuracy and computational complexity.

This paper proposes a new ABC-based block motion estimation approach. The proposed approach has the following two contributions, which are significantly different with conventional ABC-based block motion estimation approaches:

- First, a new multiple objective fitness function is proposed in this paper with the incorporation of a new motion estimation. The spatial neighborhood information of frame is critical to image quality. However, such spatial information is neglected in conventional ABC-based motion estimation approaches. In view of this, a new measure is incorporated into the cost function of the proposed approach.
- Second, a parametric motion estimation method is utilized in the proposed approach, rather than searching for optimal pixel values in the whole image space. A novel method by using modified artificial bee colony algorithm (MABCA) is proposed, which is the combination of basic ABCA and several techniques—initialization based on temporal-spatial correlation, duplicate searching avoidance technique, adaptive search criteria for iteration, and early termination of block matching.

The remainder of the paper is organized as follows: Section 2 mainly introduces the basic ABCA. In Section 3, the block-matching algorithm for motion estimation using basic ABCA is firstly discussed and then the modified ABCA (MABCA) based on several improvement techniques is proposed. Section 4 gives the parameter setting and experimental results, and the conclusion is drawn in Section 5.

2 Artificial bee colony algorithm

Artificial bee colony algorithm [33–35] is a population-based minimal bee foraging model, consisting of three groups of bees: employed bees, onlookers, and scouts. The search space represents the solution, within which all the points are considered as food sources the bees can exploit. The quality of the food source is measured by the function to be optimized [15, 36].

In the bee colony, the total number of bees is NP. Half of the bee colony is employed bees and the other half contains the onlookers. The employed bees $\{EB|\, EB_1, EB_2, \ldots, EB_{FN}\}$ are randomly assigned to different food sources $\{FS|\, FS_1, FS_2, \ldots, FS_{FN}\}$ in the search window. (The ABCA assumes that for every D dimensional food

source $\mathrm{FS}_i = \left(x_1^i, x_2^i, \ldots, x_D^i\right)$, there is only one employed bee EB_i. Thus, the number of the food sources FN is equal to that of the employed bees.) Then, the employed bees show the information about their food sources using a waggle dance in the dance area to the onlookers who are waiting in the hive. Each onlooker will watch numerous dances and select a food source. The food source with higher quality will consequently attract more onlookers. The probability $P(i)$ of selecting a food source i is determined by fitness $f(i)$:

$$P(i) = \frac{f(i)}{\sum_1^{\mathrm{FN}} f(i)}, i = 0, 1, \ldots, \mathrm{FN} \tag{1}$$

$$f(i) = \begin{cases} \dfrac{1}{\mathrm{Obj}(i)+1} & , \mathrm{Obj}(i) \geq 0 \\ 1 + |\mathrm{Obj}(i)| & , \mathrm{Obj}(i) \leq 0 \end{cases} \tag{2}$$

where $f(i)$ is the fitness of the solution of the food source FS_i. FN is the number of food sources, and $\mathrm{Obj}(i)$ is the solution value of the function to be optimized. After an onlooker has selected its food source, it will randomly search the neighborhood FS_i^* of the chosen food source. The location of the food source is produced by the following equation:

$$\mathrm{FS}_i^* = \left(x_1^i, \ldots, x_j^i + \left(x_j^i - x_j^k\right) \times (\,\mathrm{rand}(0,1) - 0.5) \times 2, \ldots, x_D^i\right) \tag{3}$$

where $j (j \leq D)$ and $k (k \leq \mathrm{FN} \cap k \neq i)$ are chosen randomly.

Then, the onlooker will compute the fitness value of the new food source FS_i^*. If this value is better than the original one corresponding to EB_i, then the EB_i will change its position to the position of FS_i^*. The equation is shown as follows:

$$\mathrm{FS}_i = \begin{cases} \mathrm{FS}_i^*, & f\left(\mathrm{FS}_i^*\right) > f(\mathrm{FS}_i) \\ \mathrm{FS}_i, & \text{else} \end{cases} \tag{4}$$

However, if the fitness value has not been improved by the new fitness after the preordained maximum iterations (limit), the associated employed bee will abandon this food source and will become a scout to randomly search another food source in the search window. After the new locations of the employed bees (or the food sources) are determined, a next iteration will be started until the termination condition is satisfied.

The pseudocode is as follows:

Artificial Bee Colony algorithm for Constrained Optimization

Input:
$Obj(z), z = [FS_1, FS_2, \ldots, FS_{FN}]$ *{cost function}*
$S = [L_k, H_k], \forall k = 1,2, \ldots, FN$ *{given constrains}*
NP, limit *{algorithm's parameters}*
Output:
FS_{min} *{minimum location}*
Begin
initialize the parameters of food sources
while termination condition not met **do**
for each employed bee **do** *{all employed bees}*
try to find a better food source FS_i^* *(equation (3))*
calculate the $Obj(i)$ and $f(i)$ *(equation(2))*
if better food source found **then**
Move to the better food source $FS_i \Leftarrow FS_i^*$
end if
end for
for each onlooker **do** *{all onlookers}*
choose a food source FS_i *(equation (1) and (2))*
try to find a better food source FS_i^* *(equation (3))*
if better food source found **then**
Move to the better food source $FS_i \Leftarrow FS_i^*$
end if
end for
if FS_i is not improved for limit times **then** *{scouts}*
randomly select a new food source FS_i^*
calculate the $\mathrm{Obj(i)}^*$ and $f(i)^*$ *(equation(2))*
end if
record global optimal location
end while

3 Method

As mentioned above, in block-matching algorithm, each frame is divided into a matrix of non-overlapping macro blocks with the size of 8×8 typically. Each block in current frame is compared with the candidate blocks of equal size within the search window in reference frame to find the best matching block and its displacement from the current block. And this displacement represents the motion vector [4]. The best matching block is the one that minimizes the criterion MSE (mean square error). The definition of MSE is shown as below:

$$\mathrm{MSE}(i,j) = \frac{1}{N \times N} \sum_{m=1}^{N} \sum_{n=1}^{N} |L_k(m,n) - L_{k-1}(m+i, n+j)|^2 \tag{5}$$

where size of the block is $N \times N$, L_k is the luminance of current block, and L_{k-1} is the luminance of reference block.

In block-matching algorithm using ABCA, a swarm of bees will exploit within a search window which range from $-p$ to p in the reference frame. Each food source in the search window represents the central pixel of a

candidate block, which indicates that parameters x_1^i, $x_2^i, \ldots, x_D^i$ of the food source $\mathrm{FS}_i = \left(x_1^i, x_2^i, \ldots, x_D^i\right)$ record the location of the central pixel of the candidate block. So, in this paper, we choose the dimension $D = 2$ and use x_1^i, x_2^i to describe the vertical and horizontal coordinates of the central pixel of the two-dimensional block. In the following, we firstly proposed a block-matching algorithm based on ABCA. Since the result is not good enough, we then proposed a novel block-matching algorithm based on MABCA with several techniques to improve the quality of the reconstructed image and the computational speed.

3.1 Motion estimation using ABCA

In the ABCA, the search window ($[-p, p]$) is the environment for the bee colony to search for honey, and each pixel in the search window is a candidate food source that a bee could exploit. The object function $\mathrm{Obj}(\mathrm{FS}_i)$ to be optimized is $\mathrm{MSE}\left(x_1^i, x_2^i\right)$:

$$\mathrm{Obj}(\mathrm{FS}_i) = \mathrm{MSE}\left(x_1^i, x_2^i\right) = \frac{1}{\mathrm{ms} \times \mathrm{ms}} \sum_{x=1}^{\mathrm{ms}} \sum_{y=1}^{\mathrm{ms}} \left| L_i(x, y) - L_{i-1}\left(x + x_1^i, y + x_2^i\right) \right|^2 \tag{6}$$

where the block size is $\mathrm{ms} \times \mathrm{ms}$, $L_i(x, y)$ is the luminance of the i^{th} block at coordinate (x, y), and $x_1^i, x_2^i \in [-p, p]$ are the central pixel coordinates of the candidate.

The rule for the onlookers and the employed bees to choose the neighborhood pixel is as follows:

$$\mathrm{FS}_i^* = \left(x_{\mathrm{pn}}^i, x_{\mathrm{pc}}^i + \left(x_{\mathrm{pc}}^i - x_{\mathrm{pc}}^k\right) \times (\mathrm{rand}(0, 1) - 0.5) \times 2\right) \tag{7}$$

where the randomly chosen parameter x_{pc}^i is one of the two parameters of pixel FS_i; the other parameter x_{pn}^i which is not chosen is left unchanged. x_{pc}^i is the $\mathrm{pc}^{\mathrm{th}}$ parameter of the k^{th} pixel FS_k which is also chosen randomly among $\{\mathrm{FS} | \mathrm{FS}_1, \mathrm{FS}_2, \ldots, \mathrm{FS}_{\mathrm{FN}}\}$.

3.2 Improvement techniques

The earlier experimental results show that the efficiency of ABCA is not good enough. We then find that there are several following problems existing in the basic ABCA that are influencing its estimation efficiency:

- Firstly, it fails to consider the temporal-spatial correlation of the motion vectors. In motion estimation, the vectors of the neighbor blocks and previous frame block are interrelated [3, 6]. We can make use of the regularity to estimate the current motion vector and improve the search result.
- Secondly, one candidate food source can be searched for several times due to the fact that the new food sources are chosen randomly in the basic ABCA. The computational complexity will decrease a lot, if choosing the point previously searched is avoided.
- In addition, the flexibility of stop condition is poor. The computational complexity will be reduced if the stop condition is appropriately adjusted when dealing with smooth motion sequences because it is easier to find the optimal solution for these sequences.
- Lastly, the judgment on stationary block is not enough. The basic ABCA is still running to find the motion vector even when dealing with stationary block, which will greatly increase the time cost. In fact, there is no necessity to run ABCA for each block of the sequence, since most of the blocks from the small motion sequences are stationary.

In view of the above observations, a modified ABCA (MABCA) model is proposed. Below is the further explanation of the improvement process.

3.2.1 Food source initialization

The motion estimation using basic ABCA lacks prejudgment of the initial location of the food sources. In fact, in motion estimation, the distribution of the motion vectors has two main characters: enter-bias property and temporal-spatial correlation. Hence, we can make use of these characters and choose several points as the fixed initial points. However, if all the initial points are fixed, there will be a possibility for the algorithm to lose the randomness for severe motion videos. Therefore, we adopt a new technique of fixed points and random points combined to improve the search result.

As is shown in Fig. 2, experimental statistics show that the motion vector of current block 0# has the strongest correlation with those of the current block's adjacent blocks 1#, 2#, 3#, 4#, and the corresponding block 5# of the previous one frame. Combined with the center-bias property, the fixed initial points are $(0, 0)$ and the motion vectors $\left(x_1^1, x_2^1\right)$, $\left(x_1^2, x_2^2\right)$, $\left(x_1^3, x_2^3\right)$, $\left(x_1^4, x_2^4\right)$, and $\left(x_1^5, x_2^5\right)$, corresponding to the blocks 1#, 2#, 3#, 4#, and 5#. So, finally, the initial points are composed of the six fixed points and several random points. This technique gives consideration to both the distribution characters of motion estimation and the randomness of the ABCA.

3.2.2 Duplicate searching avoidance technique

During the process of employed bees and onlookers improving the current food source, since random numbers are involved in the rule of selecting new neighborhood, shown in Eq. (7), there is a large possibility of researching the same point, especially when the search

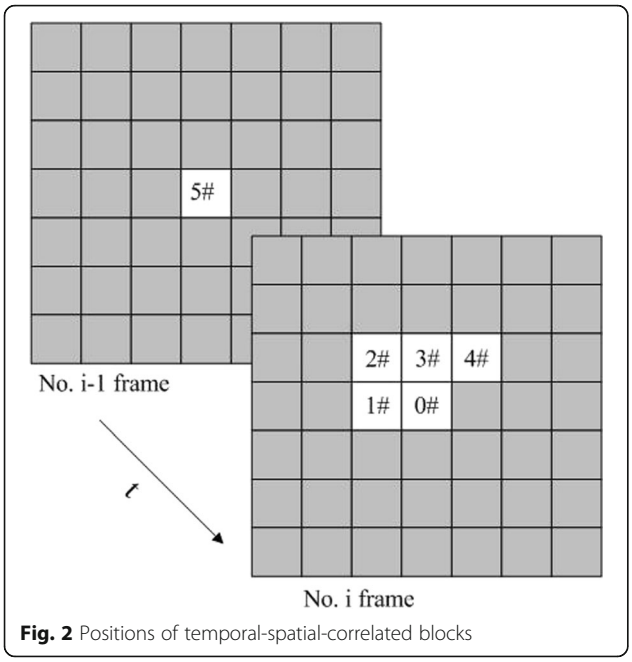

Fig. 2 Positions of temporal-spatial-correlated blocks

window is restricted in a small range. Accordingly, the computation complexity will increase unnecessarily as the same point is researched again and again.

To solve this problem, we proposed a duplicate searching avoidance technique with a two-dimensional history flag matrix recording all the previous research points. Each element in the matrix represents a candidate point in the search window. The initial value of the matrix is zero at the beginning of the algorithm for each block. If a point is visited during the search, then the value of the matrix's element corresponding to that point is set to "1." Before calculating any point within the block, the value of the element corresponding to that point is checked. If the value is zero, the calculating operation is permitted or the operation is forbidden, and the algorithm has to produce a new candidate point to check.

3.2.3 Adaptive search criteria for iteration

A small number of iteration maxCycle contributes to a fast search process at the cost of losing estimation accuracy. On the other hand, a big maxCycle guarantees the good quality of the reconstructed image at the expense of high computational complexity which is not suitable for real-time video coding. Therefore, we need a good balance between the result and the iteration number. We find that as the iteration number increases, the result for each more time of iteration usually becomes to yield smaller change compared with its previous iteration. Thus, for the proposed ABCA to get well-qualified reconstructed image with acceptable computational complexity, several adaptive search criteria are shown as follows:

- Stop searching the current block when the total number of iterations reaches a predetermined number (maxCycle).
- Stop searching the current block even if the current iteration number does not reach maxCycle when the result of each iteration remains the same value for N times of iteration.

The estimation accuracy and computational complexity are both proportional to the maxCycle and N. So, in this paper, we select maxCycle to be 10 and N to be 7 to provide a good trade-off between the image quality and time cost.

3.2.4 Early termination of block matching

For the videos of which the movement is relatively flat, most of the blocks are stationary. It is a big waste of resources to search the motion vector which is very close to the origin point with the basic ABCA. Therefore, we proposed a new technique (early termination of block matching) to further reduce the computational complexity under the premise of ensuring the estimation accuracy. Early termination of block-matching technique decides whether to abandon the search of the current block before starting the ABCA or not with a certain criterion SAD. SAD is called sum of absolute difference shown below:

$$\mathrm{SAD}(i,j) = \sum_{m=1}^{N}\sum_{n=1}^{N}|L_k(m,n) - L_{k-1}(m+i,n+j)| \tag{8}$$

where size of the block is $N \times N$, L_k is the luminance of current block, and $L_{k-1}f_{k-1}$ is the luminance of reference block.

In early termination of block-matching estimation, we first assume the current block to be a stationary block and let the motion vector be (0,0) and then calculate the $\mathrm{SAD}_0 = \mathrm{SAD}(0,0)$ between the current block and its reference block.

$$\mathrm{SAD}_0 = \sum_{m=1}^{N}\sum_{n=1}^{N}|L_k(m,n) - L_{k-1}(m,n)| \tag{9}$$

If the values of SAD_0 is below a pre-set threshold SAD_T, the current block is considered to be a stationary block, its corresponding motion vector is recorded as zero, and the block-matching algorithm will start searching the next block of the current image. Based on a series of test value of SAD_T, we find that $\mathrm{SAD}_T = 128\mathrm{SAD}_T$ gives a good performance in balancing the estimation accuracy and computational cost. Hence, in this paper, we choose SAD_T to be 128.

3.3 Block-matching algorithm for motion estimation using MABCA

The pseudocode is shown as follows:

Block-matching based motion estimation using MABCA

Input:
mbsize
{block size}
$Obj(z) = MSE(z),\ z = [FS_1, FS_2, \ldots, FS_{FN}]$
{cost function}
$S_k = [-p, p], \forall k = 1,2, \ldots, FN$ *{given constrains}*
NP, limit, maxCycle, N, SAD_T *{algorithm's parameters}*
Output:
FS_{min} *{minimum locations}*
Begin
for each frame **do**
divide each frame into $(row \times col/mbsize^2)$ matching blocks
for each block **do**
calculate its SAD_0 *(equation (9))*
if $SAD_0 < SAD_T$ **then**
abandon the current block and search the next block
end if
initialize the history flag matrix
initialize the fixed and random points *(shown in Figure 2)*
while termination condition not satisfied **do**
for each employed bee **do** *{all employed bee}*
try to find a better food source FS_i^*
check its history flag
if flag=0 **then**
calculate the $Obj(i)$ and $f(i)$ *(equation(2))*
end if
if better food source found **then**
Move to the better food source $FS_i \Leftarrow FS_i^*$
end if
end for
for each onlooker **do** *{all onlookers}*
choose a food source FS_i *(equation (1) (2) (6))*
try to find a better food source FS_i^* *(equation (7))*
check its flag
if flag=0 **then**
calculate the $Obj(i)$ and $f(i)$ *(equation(2))*
end if
if better food source found **then**
Move to the better food source $FS_i \Leftarrow FS_i^*$
end if
end for
if FS_i is not improved for limit times **then** *{scouts}*
while flag=1 **do**
randomly select a new food source FS_i^*
check its flag
end while
calculate the $Obj(i)^*$ and $f(i)^*$ *(equation(2))*
end if
record global optimal location
end while
end for
end for

4 Experimental results and discussions

4.1 Experimental setup

The result of motion estimation based on MABCA was compared with six other famous fast search methods, full search [9], three-step search [10], new three-step search [11], simple and efficient search [12], four-step search [13], and diamond search [14], on both accuracy and computational complexity. MSE is used as the criterion for block matching and the peak signal-to-noise ratio (PSNR) is used to evaluate the quality of the reconstructed image. PSNR is defined as follows:

Table 1 Parameters of block-matching algorithm

Parameter	Explanation	Value
mbsize	Search block size	8
p	Maximum displacement	7

$$\mathrm{PSNR} = 10\log_{10}\frac{L^2}{\mathrm{MSE}} \tag{10}$$

where MSE is the mean squared error between the current image and its reconstructed image, averaged over all the images of a test sequence. L is the dynamic range of pixel values (since we use only the luminance component in experiment, L = 255 is chosen in this paper). For color images, the *Y*, *U*, *V* components respectively have a MSE value. In order to simplify the computational complexity, only *Y* component is calculated in the experiment because *Y* component is the luminance value and can represent all the three RGB components of a color image. Based on practical experience, for *Y* component, a PSNR value of > 40 dB means the reconstructed image is very close to the current image. If the PSNR value is between 30 to 40 dB, the distortion can be detected but accepted. A PSNR value between 20 to 30 dB represents poor image quality. While if the PSNR value is below 20 dB, the reconstructed image is unacceptable. Both the PSNR and averaged search points (computational complexity) are calculated. The source code is written in Matlab R2008a.

The seven methods are tested on 10 classic test video sequences, including five CIF videos "Bus," "Stefan," "Hall_motion," "Bridge_close," and "News" and five QCIF videos "Hall_motion," "Highway," "News," "Salesman," and "News," which are compressed formats of their corresponding CIF videos with a rate of 1/4. And 45 frames are tested from each video sequence. The search window is 15×15 pixels, with a search range of ±7 and the size of matching block being 8×8 pixels, shown in Table 1.

Since the MABCA is based on a certain degree of randomness, the result of each test video sequence is averaged over the 45 tested frames, and the MSE result of each frame is in addition averaged over a mass of

Table 2 Parameters of ABC algorithm

Parameter	Explanation	Value
D	Dimension of the algorithm	2
NP	Swarm size	16
FoodNumber (FN)	The total number of the food source	8
maxCycle	The maximum number of cycles for foraging	10
Limit	The maximum number of trials	10
N	Adaptive search criteria for iteration	7
SAD_T	Early termination criterion	128

Table 3 Motion intensity of the five CIF and five QCIF sequences (the means and standard deviations of the magnitude of the motion vectors)

Sequence	Mean	Standard deviation
CIF		
Bus	4.9924	2.0486
Stefan	2.2919	2.7147
Hall_motion	1.7279	2.4868
Bridge_close	1.3882	3.2778
News	0.7015	1.8178
QCIF		
Highway	1.5741	2.3646
Hall_motion	0.4588	1.2365
Silent	0.3217	1.1873
News	0.1044	0.5914
Salesman	0.0776	0.3802

blocks. For example, for a CIf sequence "Bus," there are in total $1584\left(=\frac{288\times352}{8^2}\right)$ blocks for each frame. And the final result of each video sequence is further averaged over multiple times of running the MABCA until the averaged results of the same video sequence become to yield very little changes.

4.2 MABCA parameter setup

For the MABCA, the swarm size NP is chosen to be NP = 16, including 8 onlookers waiting in the colony and the other 8 employed bees respectively assigned to the fixed and random points.

For the FoodNumber, as discussed in Section 2, FoodNumber is equal to the number of employed bees, and since the initial number of employed bees is 8, we have FoodNumber = 8.

For the limit and maxCycle, small values of limit and maxCycle can lead to a fast search progress, while big values help in gaining better reconstructed image accuracy. In order to get a balance between the speed and accuracy, we finally choose limit = 10 and maxCycle = 10.

As discussed in Sections 3.2.3 and 3.2.4, for the adaptive search criteria for iteration (N) and early termination criterion SAD_T, we choose N = 7, SAD_T=128.

All the parameters are listed in Table 2.

4.3 Experimental results

The proposed MABCA can provide significant improvement of the estimation accuracy especially for the sequences with violent motion, such as "Bus.cif" (4.3 dB better than SES), and adaptively adjust its computational cost according to different sequence features especially when dealing with smooth sequences, such as most of the QCIF sequences. The evaluation of the sequences' motion intensity is shown in Table 3. The means and standard deviations of the magnitudes of the best matching motion vectors, acquired by full search of all the blocks within the 45 frames, are calculated. We assume that the motion intensity of a sequence is proportional to the mean and standard deviation values.

4.3.1 PSNR

The average PSNR of various methods (FS, TSS, NTSS, SES, FSS, DS, and MABCA) and the difference of PSNR values (Δ) between the FS and the other methods for CIF and QCIF sequences are shown in Table 4 and Table 5. The average PSNR values of the CIF and QCIF sequences based on the seven methods are shown in Figs. 3 and 4.

Table 4 and Fig. 3 denote that for all the CIF sequences, the proposed MABCA gives significant improvement in estimation accuracy and performs the best among the five fast search methods, only very slightly lower than FS. For example, the PSNR differences of sequence "Bus" yielded by TSS, NTSS, SES, FSS, and DS are respectively 27.0, 43.6, 27.6., 39.3, and 39.6 times of the difference by MABCA, which indicates that the

Table 4 The average PSNR values of the five CIF sequences using the seven methods

CIF sequence		FS	TSS	SES	NTSS	FSS	DS	MABCA
Bus	PSNR	25.6952	22.9695	21.2909	22.816	21.722	21.6976	25.5943
	Δ	0	− 2.7258	− 4.4043	− 2.8792	− 3.9732	− 3.9976	− 0.1010
Stefan	PSNR	26.7676	24.6908	24.3372	24.7661	24.5414	24.3416	26.6243
	Δ	0	− 2.0767	− 2.4304	− 2.0015	− 2.2262	− 2.4260	− 0.1433
Hall	PSNR	35.9483	35.5824	35.2018	35.6083	35.5667	35.5654	35.8232
	Δ	0	− 0.3659	− 0.7465	− 0.3401	− 0.3817	− 0.3830	− 0.1252
Bridge	PSNR	35.0582	35.0542	35.0534	35.0546	35.0545	35.05443	35.0563
	Δ	0	− 0.0040	− 0.0048	− 0.0036	− 0.0037	− 0.0038	− 0.0020
News	PSNR	37.2027	36.6369	36.0007	36.7163	36.3581	36.3203	38.94554
	Δ	0	− 0.5658	− 1.2019	− 0.4864	− 0.8445	− 0.8824	− 0.2075

Table 5 The average PSNR values of the five QCIF sequences using the seven methods

QCIF sequence		FS	TSS	SES	NTSS	FSS	DS	MABCA
Highway	PSNR	37.4059	36.2228	35.6754	36.3604	36.3254	36.3353	36.8790
	Δ	0	− 1.1831	− 1.7305	− 1.0455	− 1.0805	− 1.0706	− 0.5270
Hall	PSNR	36.0298	35.7070	35.4881	35.9241	35.8282	35.8998	35.9385
	Δ	0	− 0.3228	− 0.5417	− 0.1057	− 0.2016	− 0.1300	− 0.0913
Silent	PSNR	37.1058	36.6899	36.2318	36.7073	36.5686	36.5434	36.9554
	Δ	0	− 0.4159	− 0.8739	− 0.3985	− 0.5372	− 0.5624	− 0.1504
News	PSNR	37.7176	37.4014	37.2655	37.5466	37.4927	37.5625	37.6087
	Δ	0	− 0.3162	− 0.4521	− 0.1710	− 0.2249	− 0.1551	− 0.1090
Salesman	PSNR	39.2987	39.0400	38.8450	39.2411	39.0850	39.2029	39.2376
	Δ	0	− 0.2586	− 0.4537	− 0.0576	− 0.2137	− 0.0958	− 0.0611

result based on MABCA is on average 35.4 times better than the other five fast search methods. The PSNR values of the 45 frames of "Bus" are shown in Fig. 5. From Fig. 5, we also find that the high estimation accuracy of MABCA is extremely stable, which can prevent the image-quality fluctuation that can be usually seen when using other methods.

Further observation of the comparison of the seven block-matching estimation methods (Fig. 3) tells that the superiority of MABCA is increasingly apparently reflected as the motion intensity increases. For sequences with smooth motion, "News," "Bridge_close," and "Hall_motion," the PSNR values of the seven methods are very close to each other and yield very small Δ values. The reason is that the motion vectors of the sequences with unimodal error surface and smooth motion are mostly in the vicinity of the origin, which makes it more easy to find the global optimum. While for the sequences with violent motion, "Bus" and "Stefan," the gaps of the difference Δ values between MABCA and the other five fast search methods increase sharply, and MABCA starts to be distinctly superior to the other five fast search methods. This is because the motion vector of the sequences with violent motion is far from the origin, and the number of extreme points increases, which indicates that the unimodal error surface assumption is not satisfied. In this case, the conventional fast search methods are easily trapped in local optimum. However, the MABCA, different from the conventional fast search methods, which does not rely on the unimodal error surface assumption, can jump out of local optimum and find the global optimum.

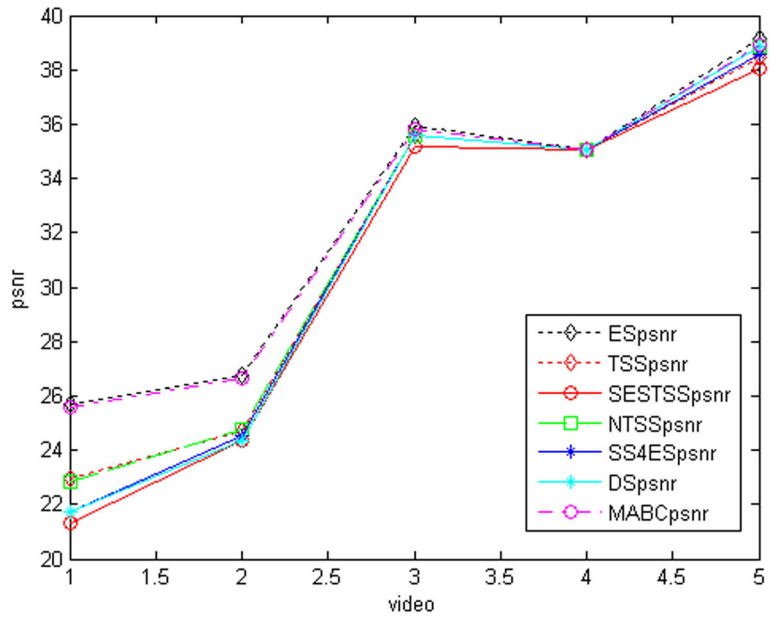

Fig. 3 Comparison of the average PSNR values of the five CIF sequences. Video 1 to video 5 respectively represent sequences "Bus," "Stefan," "Hall_motion," "Bridge_close," and "News"

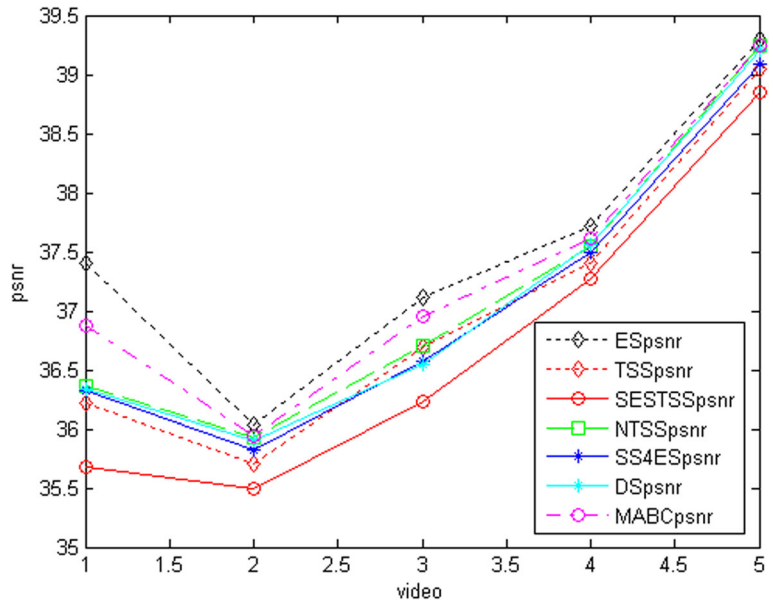

Fig. 4 Comparison of the average PSNR values of the five QCIF sequences. Video 1 to video 5 respectively represent sequences "Highway," "Hall_motion," "Silent," "News," and "Salesman"

So, in this way, the result of MABCA is always very close to FS.

The above analysis is also tenable when judging from the 45 frames of one single sequence. Take the sequence "Stefan" as an example; the PSNR values of the 45 frames of "Stefan" are shown in Fig. 6. The PSNR values of the first 20 frames of the sequence "Stefan" are generally lower than the later frames because we can see from the luminance images that the first 20 images mainly record the scene of the ball flying to the tennis ball player and a series of the player's swing which are the vigorous exercise part of the sequence with violent motion vector. The estimation accuracy of these motion vectors are reasonably lower than the motion vectors of the later frames that record the scene after hitting the ball. Consequently, the PSNR values of the first 20 frames are generally lower than the later 25 frames. Besides, for the first 20 frames, the unimodal error

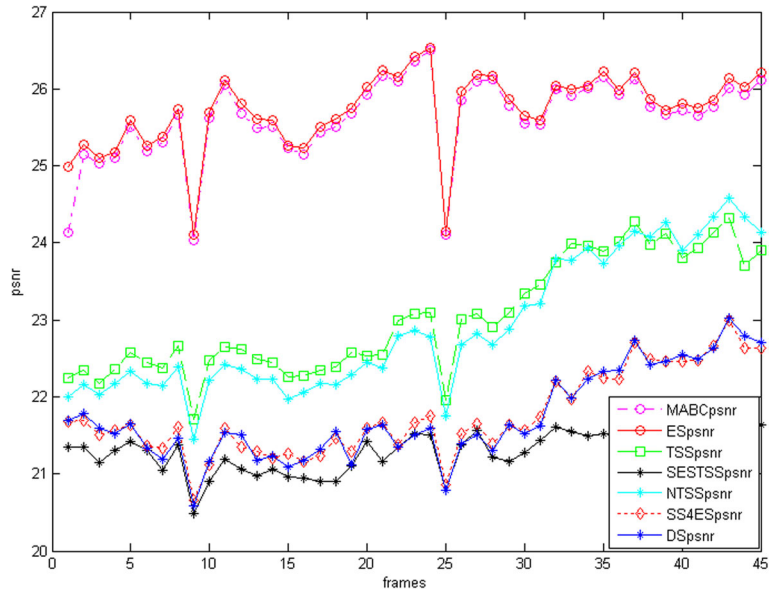

Fig. 5 Comparison of the PSNR values of the 45 frames of "Bus.cif"

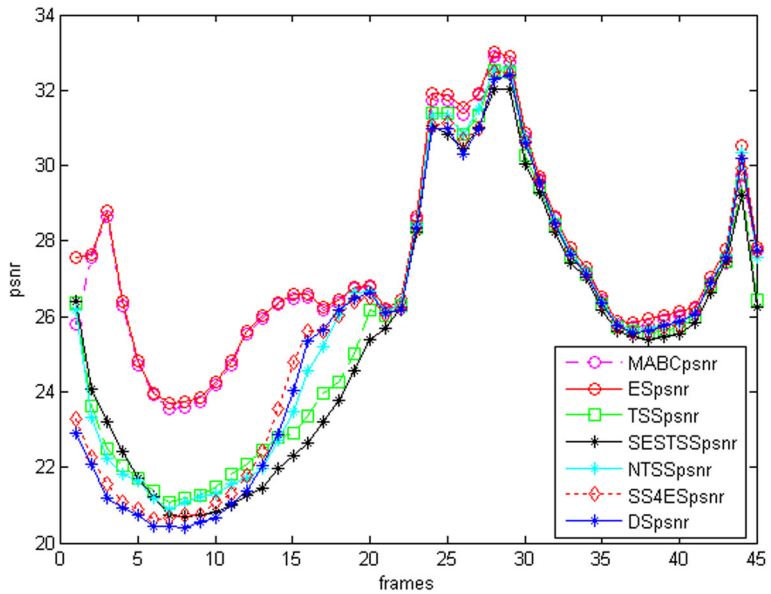

Fig. 6 Comparison of the PSNR values of the 45 frames of "Stefan.cif"

surface assumption is not satisfied, and the number of extreme point increases. Therefore, we can see from Fig. 6 that for the first 20 frames, the PSNR values of the MABCA are obviously better than the other five fast search methods.

Table 5 and Fig. 4 show that for the QCIF sequences, the proposed MABCA still maintains quite a good performance among the seven methods. For four of the five QCIF sequences, "Highway," "Hall," "Silent," and "News," the MABCA is evidently better than all the other five fast search methods. Take the sequence "Highway" as an example; as is shown in Fig. 7, the Δ value acquired by MABCA is on average 2.3 times better than all the other five methods. For the remaining sequence "Salesman", the accuracy of MABCA is still higher than most of the five methods (TSS, SES, FSS, and DS) and only slightly lower (0.0035 dB) than the NTSS because of the consideration of computational complexity, which as we will discuss in the next section, the average search point number of "Salesman" using

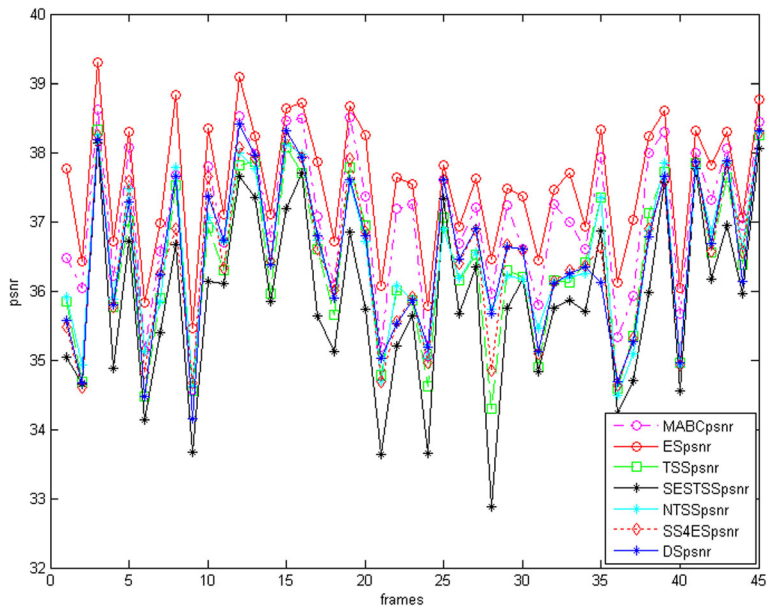

Fig. 7 Comparison of the PSNR values of the 45 frames of "Highway.qcif"

Table 6 The average search point numbers of the 10 video sequences using the seven methods

Sequence	FS	TSS	SES	NTSS	FSS	DS	MABCA
CIF							
Bus	214.5176	24.3574	16.2369	28.0036	21.701	20.9074	48.2545
Stefan	214.5176	24.1371	16.7680	21.2419	18.5633	16.1671	37.0971
Hall_motion	214.5177	24.1346	17.0952	19.2263	17.6345	14.4339	33.9602
Bridge_close	214.5176	24.1191	16.8881	17.1419	17.6922	15.1559	46.4251
News	214.5176	24.1000	17.2851	17.8282	17.0949	13.7218	6.6690
QCIF							
Highway	204.2828	23.3167	16.6563	18.1492	17.2760	14.4792	21.4177
Hall_motion	204.2828	23.2501	16.9569	16.6065	16.1936	12.7627	9.9697
Silent	204.2828	23.2149	16.9587	16.5957	16.1836	12.8024	10.5074
News	204.2828	23.2129	17.0460	16.0354	15.8814	12.3547	6.2020
Salesman	204.2828	23.2141	17.0595	16.0131	15.8884	12.3349	8.1457

MABCA is nearly reduced to half of those numbers when using other algorithms.

4.3.2 Average search point number

For the searching process, the whole searching time is mostly cost by the process of calculating the MSE value between the reference block and the new candidate block. Therefore, we have good reason to believe that, under the condition of the same sequence frames, the same search range, and the same search block, the average search point number can represent the computational complexity.

In terms of computational complexity, the average search point numbers of the 10 video sequences using the seven methods are listed in Table 6.

We can see from Table 6 that the proposed MABCA can adaptively adjust its average search point number according to the motion intensity of the sequences to capture the best motion vectors at the lowest possible computational cost. This is because, as is shown in Table 7, when dealing with sequences with smooth motion, most of the blocks whose SAD_0 is below SAD_T are regarded as stationary blocks, and the total search point number for this block is only 1 point. Besides, for these smooth sequences, the motion vectors are in the vicinity of the origin, which provides a great possibility of finding the best matching motion vector and satisfying the stop criteria very quickly.

Table 7 The total block, average stationary block, and percentage parameters of the five QCIF sequences

Sequence	Total block	Average stationary block	Percentage (%)
Highway	396	251.0227	63.39
Hall_motion	396	335.1778	84.64
Silent	396	332.4889	83.96
News	396	361.2222	91.22
Salesman	396	346.5333	87.51

Total block is the total block number of one frame. Average stationary block is the number of stationary block ($SAD_0 < SAD_T$), averaged over 45 frames. Percentage represents the proportion of the stationary blocks.

From the experimental result, for the sequences with large motion, the average search point number of MABCA is correspondingly higher than the other five methods but is lower than FS. Take the most violent sequence "Bus.cif" as an example; the proposed MABCA has a speedup ratio about 4.4 (=214.5176/48.2545) over the FS method. For the sequences with slight motion, the average search point number of MABCA is also the lowest than all the other methods. See the example of "News.cif"; MABCA has a speedup ratio of 32(=214.5176/6.6690) and 3.6(=24.1000/6.6690) over the FS and TSS. And for both of the two kinds of sequences, violent and smooth, the estimation accuracy of the MABCA is always the best for most of them.

5 Conclusions

Block-matching algorithm is widely used in motion estimation and has made a great contribution to improve the transmission efficiency. In this paper, a combination of artificial bee colony algorithm and a series of improved approaches—the modified ABCA (MABCA)—is proposed and applied to block-matching algorithm for motion estimation. The experiment results show that the proposed method can effectively improve the estimation accuracy and avoid local optima problem. It can improve the estimation accuracy of violent motion sequences with a little sacrifice of computational complexity, while reducing the computational cost greatly

when dealing with smooth sequences at the premise of high estimation accuracy.

Though the proposed MABCA has achieved significant performances, the work can be extended by using dynamic search window adjustment, switching search patterns, and block classification to further improve the estimation accuracy for sequences with smooth motion and reduce the computational complexity for the sequences with violent motion.

Acknowledgements
Not applicable.

Funding
This work is partially supported by the National Natural Science Foundation of China (No.61302120), National Engineering Center for Mobile Ultrasonic Detection (No.2013FU125X02), and the supports of the Priority Academic Program Development of Jiangsu Higher Education Institutions, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (No.KJR16237).

Availability of data and materials
Not applicable.

Authors' contributions
Yu is the first and corresponding author. The other authors made the algorithm and experiments. All authors read and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China. [2]Nanjing University of Information Science and Technology, Nanjing, China.

Received: 9 April 2017 Accepted: 4 September 2017
Published online: 19 September 2017

References

1. S. Kappagantula, K.R. Rao, Motion compensated interframe image prediction. IEEE Trans. Commun. **COM-33**, 1011–1015 (1985)
2. C. STILLER, Estimating motion in image sequences: a tutorial on modeling and computation of 2d motion. IEEE Signal Process. Mag. **16**(4), 70–91 (1999)
3. D.W. Kim, J.H. Choi, Y.S. Choi, C.H. Jeon, N.Y. Ko, in *1996 IEEE TENCON-digital signal processing applications*. Block motion estimation based on spatio-temporal correlation **2**, 955–960 (1996)
4. Y.-J. Koh, S.-B. Yang, An adaptive search algorithm for finding motion vectors, in *Proceedings of the IEEE region 10 conference on TENCON99*. **1**, 186–189 (1999)
5. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, in *Proceedings of NTC8*. Motion compensated interframe image coding for video conference (1981)
6. L. Hsieh, W.-S. Chen, C.-H. Liu, Motion estimation using two-stage predictive search algorithms based on joint spatio-temporal correlation information. Expert Syst. Appl. **38**, 11608–11623 (2011)
7. M.C. Chang, J.S. Chein, in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'06)*. An adaptive search algorithm based on block classification for fast block motion estimation, 3982–3985 (2006)
8. F. Yu, M. Hui, W. Han, P. Wang, L.-q. Dong, Y.-j. Zhao, The application of improved block-matching method and block search method for the image motion estimation. Opt. Commun. **283**, 4619–4625 (2010)
9. T.G. Ahn, Y.H. Moon, J.H. Kim, Fast full-search motion estimation based on multilevel successive elimination algorithm. IEEE Transactions on Circuits and Systems for Video Technology **14**(11), 1265–1269 (2004)
10. J.N. Kim, T.S. Choi, A fast three-step search algorithm with minimum checking points using unimodal error surface assumption. IEEE Trans. Consum. Electron. **44**(3), 638–648 (1998)
11. R. Li, B. Zeng, M.L. Liou, A new three-step search algorithm for block motion estimation. IEEE Trans.on Circ. Syst. Video Technol **4**(4), 438–442 (1994)
12. J. Lu, M.L. Liou, A simple and efficent search algorithm for block matching motion estimation. IEEE Trans.on Circ. Syst. Video Technol **7**(2), 429–433 (1997)
13. L.-M. Po, W.-C. Ma, A novel four-step search algorithm for fast block motion estimation. IEEE Trans. Image Processing. **9**(2), 287–290 (2000)
14. S. Zhu, K.-K. Ma, A new diamond search algorithm for fast block-matching motion estimation. IEEE Trans. Image Processing. **9**(2), 287–290 (2000)
15. Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. [S. l.]: Erciyes University, Technical Report: TR06, (2005)
16. L.W. Lee, J.F. Wang, J.Y. Lee, J.D. Shie, Dynamic search-window adjustment and interlaced search for block-matching algorithm. IEEE Trans. Circ. Syst. Video Technol. **3**(1), 85–87 (1993)
17. M. Gallant, F. Kossentini, An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding. IEEE Trans. Image Process. **8**, 1816–1823 (1998)
18. A. Mohammad, L. George, in *World Comput. Int. Conf. on IPCV*. Fast predictive coding method based on an enhanced blocks motion estimation. 472–476 (2008)
19. R. Srinivasan, K.R. Rao, Predictive coding based on efficient motion estimation. IEEE Trans. Commun. COM **33**(8), 888–896 (1985)
20. Y.B. Zhang, L. Xu, X.Y. Ji, Q. Dai, Polynomial approximation motion estimation model for motion-compensated frame interpolation IEEE trans. ON Circuits and systems-for video technology **26**(8), 1421–1432 (2016)
21. 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 Process. **10**(6), 438–447 (2016)
22. S. Li, W. Xu, N. Zheng, H. Wang, A novel fast motion estimation method based on genetic algorithm. Acta Electron. Sin. **28**(6), 114–117 (2000)
23. Z.H.A.N.G. Ping, W.E.I. Ping, Y.U. Hong-yang, F.E.I. Chun, A novel search algorithm based on particle swarm optimization and simplex method for block motion estimation. International Journal of Digital Content Technology and its Applications **5**(Number 1) (2011)
24. R.C. Eberhart, Y. Shi, in *Proceedings of the 7th Interna. Conference on Evolutionary Programming VII*. Comparison between genetic algorithm and particle swarm optimization, vol 1447 (1998)
25. D. Karaboga, B. Akay, A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**(1), 108–132 (2009)
26. Zhaoqing Pan, Yun Zhang, and Sam Kwong, "Efficient motion and disparity estimation optimization for low complexity multiview video coding," IEEE Trans. Broadcasting. **61**(2), 166–176 (2015)
27. Yu Xue, Jiongming Jiang, Binping Zhao and Tinghuai Ma, "A self-adaptive articial bee colony algorithm based on global best for global optimization", Soft Computing. (2017). doi:10.1007/s00500-017-2547-1
28. Karaboga D. A New Design Method Based on Artificial Bee Colony Algorithm for Digital IIR Filters, Journal of the Franklin Institute, **346**(4), 328–348 (2009)
29. Y.-f. Cao, W.-y. Yu, Y.-h. Xiao, et al., Image segmentation using artificial bee colony and fast FCM algorithms. Adv. Sci. Lett. **4**, 400–407 (2011)
30. P.K. Sahooa, G. Arorab, A thresholding method based on 2-D Renyi's entropy. Pattern Recogn. **37**, 1149–1161 (2004)
31. D. Karaboga, C. Ozturk, A novel clustering approach: artificial bee colony (ABC) algorithm. Appl. Soft Comput. **11**, 652–657 (2011)
32. D. Karaboga, A new design method based on artificial bee colony algorithm for digital IIR filters. Journal of the Franklin Institute **346**(4), 328–348 (2009)
33. D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. **8**(1), 687–697 (2008)
34. A. Baykasoùlu, L. Özbakır, P. Tapkan, Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem. Swarm Intelligence: Focus

on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, 532, Itech Education and Publishing, Vienna, (2007)
35. B. Akay, D. Karaboga, Parameter tuning for the artificial bee colony algorithm. In: Nguyen N, Kowalczyk R, Chen SM (eds) Proceedings of the ICCCI 2009, LNCS, vol 5796. Springer, Berlin/Heidelberg, 608-619 (2009)
36. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. **39**(3), 171–459 (2007)