Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 610230, 10 pages
http://dx.doi.org/10.1155/2013/610230

Research Article

Single-Dimension Perturbation Glowworm Swarm Optimization Algorithm for Block Motion Estimation

Xiangpin Liu,[1] Shibin Xuan,[1,2] and Feng Liu[1]

[1] *College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, China*

Correspondence should be addressed to Shibin Xuan; xshibin1997@126.com

Received 17 July 2013; Revised 10 October 2013; Accepted 24 October 2013

Academic Editor: Swagatam Das

In view of the fact that the classical fast motion estimation methods are easy to fall into local optimum and suffer the high computational cost, the convergence of the motion estimation method based on the swarm intelligence algorithm is very slow. A new block motion estimation method based on single-dimension perturbation glowworm swarm optimization algorithm is proposed. Single-dimension perturbation is a local search strategy which can improve the ability of local optimization. The proposed method not only has overcome the defect of falling into local optimum easily by taking use of both the global search ability of glowworm swarm optimization algorithm and the local optimization ability of single-dimension perturbation but also has reduced the computation complexity by using motion vector predictor and terminating strategies in view of the characteristic of video images. The experimental results show that the performance of the proposed method is better than that of other motion estimation methods for most video sequences, specifically for those video sequences with violent motion, and the searching precision has been improved obviously. Although the computational complexity of the proposed method is slightly higher than that of the classical methods, it is still far lower than that of full search method.

1. Introduction

Motion estimation (ME) is the prediction of the motion vector (MV) between the current frame and the reference frame. As an important part of video processing, it has been widely used in compression, computer vision, target tracking, and monitoring of industrial and other areas. Among many methods of ME, block matching algorithm (BMA) has been accepted by many video compression coding standards due to its simple structure and ease of implementation, for example, MPEG-1, MPED-4, H.261, H.263, and H.264. Full search (FS) [1] algorithm is the simplest and the most accurate BMA, but its high computational complexity makes it not suitable for real-time implementation, so a number of fast BMAs have been proposed. The most widely used of them is the BMA based on template, the classic representatives including three-step search (TSS) [2], new three-step search (NTSS) [3], four-step search (FSS) [4], diamond search (DS) [5], and the improvement based on these methods like hexagon search (HEXBS) [6] and directional diamond search (DDS) [7]. In addition, adaptive rood pattern search (ARPS) [8] which is the classic representative of the BMA based on motion vector predictor has a superior performance. However, all above methods rely on the unimodal error surface assumption (UESA), which is usually not established for the real video sequences. Therefore, all these algorithms will get trapped into local optimum easily, especially under the condition of a complex movement. In order to solve this problem, some BMAs based on global optimal are proposed in recent years, such as fast ME method based on genetic algorithm (GA) [9], block motion estimation based on immune clonal selection (BMEICS) [10], BMA based on particle swarm optimization for ME [11, 12], and so forth. These methods have a good global optimization ability which can overcome the defect of falling into local optimum easily to a certain extent, but it also loses the ability of local optimization, resulting in slow convergence speed and high computational complexity.

Glowworm swarm optimization (GSO) [13] algorithm presented by Krishnanand and Ghose is a new kind of swarm intelligence algorithm. It is inspired by social behavior of

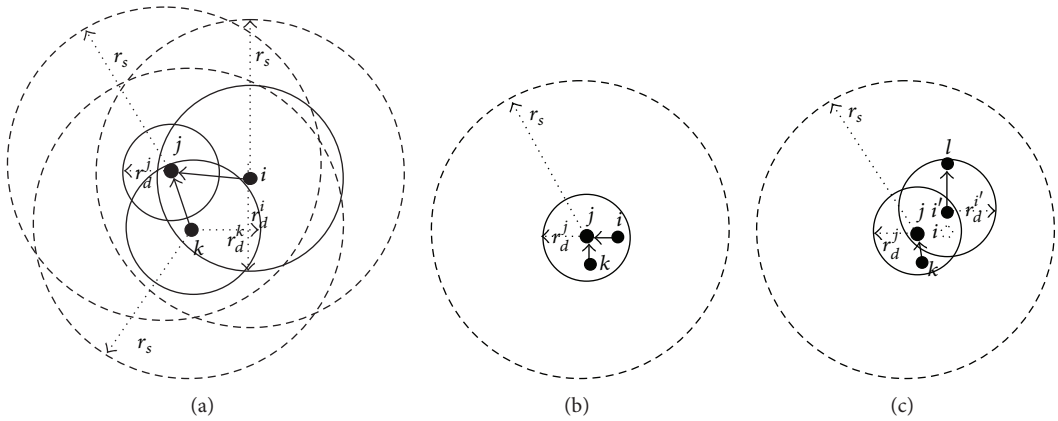

Figure 1: The gathering process diagram of basic GSO algorithm.

the glowworm and the phenomenon of bioluminescent communication. GSO algorithm has the characteristics of ant colony optimization (ACO) algorithm and artificial bee colony (ABC) algorithm, but compared to ACO, GSO algorithm has better generality and a wider range of application. Although the roulette wheel method is applied to choose the moving object in ABC algorithm as well as GSO algorithm, all scout bees are candidate moving objects of the onlooker Bee, which makes it easy to fall into local optimum. In GSO algorithm, each glowworm selects its moving object only within its neighborhood (a collection of the neighbor glowworms who have higher luciferin than the current glowworm). Since different glowworms have different neighborhoods, the choice of moving object is different too, which can avoid falling into local optimum. Therefore, GSO algorithm has a good performance in solving multiobjective optimization problems. It has been successfully applied to the multisource issue tracking and location problem, harmful gas leak location problem, solving the nonlinear equations [14], clustering analysis [15], traveling salesman problem [16], 0-1 knapsack problem [17], constrained engineering design problem [18], the noise of the sensor test, and so forth. So far, it still has the problems of slow convergence speed and weak local search ability as well as other swarm intelligent algorithms. For video processing, on the premise of high precision strong ability of local search, fewer iteration times and faster convergence speed are very important for reason of huge image data. According to these defects of original GSO algorithm and the characteristics of ME, single-dimension perturbation glowworm swarm optimization (SDGSO) algorithm is proposed by introducing the local search strategy called single-dimension disturbance and applied to ME. In simulation experiments, the proposed method has been compared to other ME methods, including classic fast BMAs and the BMAs based on swarm intelligent algorithms.

2. Single-Dimension Perturbation Glowworm Swarm Optimization Algorithm

In GSO, the agents are thought of as glowworms that carry a luminescent quantity called luciferin along with them. The glowworms encode the fitness of their current locations, evaluated using the objective function, into a luciferin value that they broadcast to their neighbors. The glowworm identifies its neighbors and computes its movements by exploiting an adaptive neighborhood, which is bounded above by a radial sensor range. Each glowworm selects, using a probabilistic mechanism, a neighbor that has a luciferin value higher than its own and moves toward it. These movements, based only on local information and selective neighbor interactions, enable the swarm of glowworms to partition into disjoint subgroups that converge on optima, but it also means that most glowworms will get together in the late iteration, which will lead to a slow optimum speed or even fall into local optimum. There is a typical example to illustrate the gathering process of GSO algorithm as showed in Figure 1(a). There are two neighbors j and k within the local-decision domain of glowworm i, and we suppose that glowworm i has chosen glowworm j as its neighbor in the probability p_{ij}, so it moves to j at a step size. There is only one neighbor j within the local-decision domain of glowworm k, so it moves to j at a step size. After many times of iteration, glowworms i and k will always move along the direction of glowworm j, and converge to the location of glowworm j finally as shown in Figure 1(b). However, this location may be a local optimum, rather than the global optimal. We suppose that the luciferin value of glowworm l is bigger than that of glowworm j as shown in Figure 1(c), but because glowworm i cannot find glowworm l within its local-decision domain, it will always move along the direction of glowworm j and fall into local optimum in the end. Therefore, we consider disturbance after each move, glowworm i is taken for instance, and it can be found that the location of glowworm i' is better than that of i by means of the disturbance, and the glowworm l is within the local-decision domain of i', and then glowworm i will move to l at a probability, which will change the direction of the movement and avoid falling into local optimum.

In this section, the basic glowworm swarm optimization algorithm and single-dimension perturbation strategy are introduced first, and then a novel glowworm swarm optimization algorithm based on single-dimension perturbation is proposed.

2.1. Basic Glowworm Swarm Optimization Algorithm. Here is the description of basic GSO algorithm.

In Luciferin-update phase, the luciferin updating of each glowworm depends on its current fitness value. That is to say, current luciferin value needs to add up a proportion of its current fitness value on the basis of its value at the previous moment. Besides, the luciferin value which is volatilized as the passage of time needs to be subtracted. The specific luciferin update formula can be stated as

$$l_i(t+1) = (1-\rho) \times l_i(t) + \gamma \times J(x_i(t)), \quad (1)$$

where $l_i(t)$ is the luciferin value of glowworm i, ρ is the volatile coefficient of luciferin, γ is the update coefficient, and $J(x_i(t))$ is the objective function value of glowworm i.

In movement phase, neighbor glowworms which have higher luciferin value than its own are selected by each glowworm in its local-decision domain and form a neighborhood, and the moving object will be chosen from the neighborhood according to the probabilistic mechanism. For the glowworm i, probabilistic equation of moving to a neighbor j is given as follows:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}, \quad (2)$$

where $j \in N_i(t)$, $N_i(t)$ is the neighborhood of glowworm i.

Movement equation is

$$x_i(t+1) = x_i(t) + s\left[\frac{x_j(t) - x_i(t)}{\left\|x_j(t) - x_i(t)\right\|}\right], \quad (3)$$

where $x_i(t)$ is the location of glowworm i at tth iteration, s represents the step size, and $\|\cdot\|$ is the Euclidean operator.

In local-decision domain update rule, during the course of iteration, the number of neighbors is dynamic for each agent, so the local decision range must be adjusted to ensure that the swarm of glowworms can capture multiple peaks. The update rule is given by

$$r_d^i(t+1) = \min\left\{r_s, \max\left\{0, r_d^i(t) + \beta\left(n_t - \left|N_i(t)\right|\right)\right\}\right\}, \quad (4)$$

where r_s is the perceive domain, β is a constant factor, n_t is a modulus used to control the number of neighbors, and $|N_i(t)|$ is the number of neighbor glowworms.

2.2. Local Search Strategy. Though GSO algorithm has many advantages such as strong universality, good overall and good performance at solving multi-objective optimization, it also has some shortcomings as well as other swarm intelligent algorithms, such as slow convergence speed and poor local search ability. According to these problems, a novel local search strategy called single-dimension perturbation search (SDPS) [19] is introduced in this paper. SDPS is a local search strategy which is applied after the movement phase according to (3). With the good global optimization ability of GSO algorithm, SDPS will enhance the ability of local optimization, speed up the convergence, and improve the precision.

In 2D space, for instance, the four points which distance the glowworm a steps will be selected randomly after the location updating, and the new location will be calculated at a certain probability. If the objective function value of the new location is bigger than that of the current location, move or else, keep still.

2.3. Main Steps of the SDGSO Algorithm. The computational procedure of the proposed algorithm can be summarized as follows.

Step 1. Initialize the population. A swarm of n glowworms is generated randomly within the search space: dimensions of the search space—D, initial location—$x_i(0)$, initial luciferin—l_0, perceive domain—r_s, maximum iterations—$\max t$, step size—s, step size of local search—a, volatile coefficient of luciferin—ρ, and update coefficient of luciferin—γ.

Step 2. Update each glowworm i.

(2.1) Calculate the objective function value of the glowworm's location—$J(x_i(t))$, and then it is turned into the luciferin—$l_i(t)$ according to (1).

(2.2) Select the neighbor glowworms in the local-decision domain—$r_d^i(t)$ and form a neighborhood—$N_i(t)$.

(2.3) Calculate the probability—$p_{i\bullet}(t)$ of glowworm i according to (2).

(2.4) Select glowworm j as the neighbor of glowworm i by using the roulette method, and update the location according to (3).

(2.5) Let $\vec{c} = \vec{x}_i$, and a single-dimension—m $(1 \le m \le D)$ is selected randomly, then a rand number rand $\in [0, 1]$ is generated. If rand < 0.5, then $\vec{c}[m] = \vec{c}[m] - a$; else $\vec{c}[m] = \vec{c}[m] + a$.

(2.6) Calculate $J(c)$ and $J(x_i(t))$, and if $J(c) > J(x_i(t))$, the location should be updated $x_i(t) = c$ or else, keep still.

(2.7) Update local-decision domain according to (4).

Step 3. In iteration accumulation, if it reaches $\max t$, turn to Step 4 or else, turn to Step 2.

Step 4. Output the result.

2.4. Convergence Analysis of Algorithm. The convergence of GSO algorithm is proved mainly through a large number of simulation experiments, and its mathematical theory basis is relatively weak. Krishnanand and Ghose analyzed the convergence of luciferin updating equation in 2006 [20], and they had proved that the maximum luciferin level was bounded due to luciferin decay and the luciferin of glowworms which is located at a peak would converge to the same value according to the deduction of the two theorems.

Theorem 1. *Assuming that the luciferin update rule in* (1) *is used, the luciferin level $l_j(t)$ for any glowworm j is bounded above asymptotically as follows:*

$$\lim_{t \to \infty} l_i(t) \le \lim_{t \to \infty} l^{\max}(t) = \left(\frac{\gamma}{\rho}\right) J_{\max}, \quad (5)$$

Table 1: Formulas of the functions, searching space, and theoretical optimal solution.

Function	Formula	Searching space	Solution
Rastrigin	$f_1(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$[5.12, 5.12]^{30}$	0
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^{30}$	0
Rosenbrock	$f_4(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2\right]$	$[-30, 30]^{30}$	0

Table 2: Comparison of DE, ABC, and GSO algorithm with SDGSO algorithm.

Function	Algorithm	Best	Median	Worst	Mean	Std
Rastrigin	DE	$1.26E+02$	$1.97E+02$	$2.26E+02$	$1.90E+02$	26.70915
	ABC	79.52209	95.50064	$1.12E+02$	95.38415	9.477102
	GSO	$4.51E-06$	0.001316	0.012433	0.002444	0.003363
	SDGSO	$2.21E-06$	$1.00E-04$	$6.57E-04$	$2.21E-04$	0.000231
Griewank	DE	0.160226	0.270926	0.354931	0.266130	0.048583
	ABC	9.701455	24.90274	40.40466	25.53636	8.467937
	GSO	$3.67E-05$	0.001396	0.043757	0.007885	0.013473
	SDGSO	$1.93E-10$	$7.83E-05$	$9.78E-04$	$2.44E-04$	0.000293
Rosenbrock	DE	$2.17E+02$	$2.71E+02$	$3.67E+02$	$2.79E+02$	42.24615
	ABC	$2.39E+05$	$1.59E+06$	$6.59E+06$	$2.01E+06$	$1.65E+06$
	GSO	$2.53E-05$	0.004393	0.222633	0.017098	0.048862
	SDGSO	$5.05E-07$	$8.15E-05$	$6.77E-04$	$1.78E-04$	0.000232

where $J_{\max}(>0)$ is the global maximum value of the objective function.

Theorem 2. *For all glowworms j located at peak location X_i^* associated with objective function values $J_i^* \le J_{\max}(>0)$ ($i = 1, 2, \ldots, n_p$, with n_p the number of peaks), if the luciferin update rule* (1) *is used, then $l_j(t)$ increases or decreases monotonically and asymptotically converges to $l_i^* = (\gamma/\rho)J_i^*$.*

In order to show that the convergence of SDGSO algorithm has been improved compared against the standard GSO, and also better than differential evolution (DE) and ABC algorithm, three classical test functions are selected to do the convergence test. The formula of the functions, searching space, and theoretical optimal solution are given in Table 1. In parameters setting, population size is 50, dimension of space is 30, number of iterations is 200, and step size of perturbation is 10^{-6}. SDGSO algorithm and GSO algorithm are tested with the three functions, respectively, for 20 times, and the best value, the median value, the worst value, the mean value, and the standard deviation of the 20 time test are recorded with *Best*, *Median*, *Worst*, *Mean*, and *Std*. Among them, *Best*, *Median*, and *Worst* reflect the quality of the solution; *Mean* shows the precision of the algorithm and reflects the convergence speed of the algorithm; *Std* reflects the stability and robustness of the algorithm. The results are shown in Table 2.

It can be seen from Table 2 that the precision and stability of the proposed algorithm have been improved obviously compared to DE and ABC algorithm and also better than those of standard GSO algorithm.

In order to illustrate the convergence speed of the improved algorithm more intuitively, convergence curves of GSO and SDGSO algorithms during the solving process of function Griewank and function Rosenbrock are shown in Figure 2. It can be seen that the convergence speed of SDGSO algorithm is faster than that of GSO algorithm.

2.5. Statistical Analyses. In order to statistically analyze the results in Table 2, a nonparametric significance proof known as the Wilcoxon's rank test [21] has been conducted which allows assessing result differences among two related methods. The analysis is performed considering a 5% significance level. Table 3 reports the P value produced by Wilcoxon's test for pairwise comparison of three groups. Such groups are formed by SDGSO versus DE, SDGSO versus ABC, and SDGSO versus GSO. As a null hypothesis, it is assumed that there is no significant difference between mean values of the two algorithms. The alternative hypothesis considers a significant difference between the two algorithms. All P-values reported in the table are less than 0.05 (5% significance level) which is strong evidence against the null hypothesis, indicating that the results of SDGSO algorithm are statistically significant and that it has not occurred by coincidence.

3. Application of SDGSO Algorithm in Block Motion Estimation

In block matching motion estimation, each frame image of video sequences is divided into macro block, and the blocks within the search window of the reference frame will be

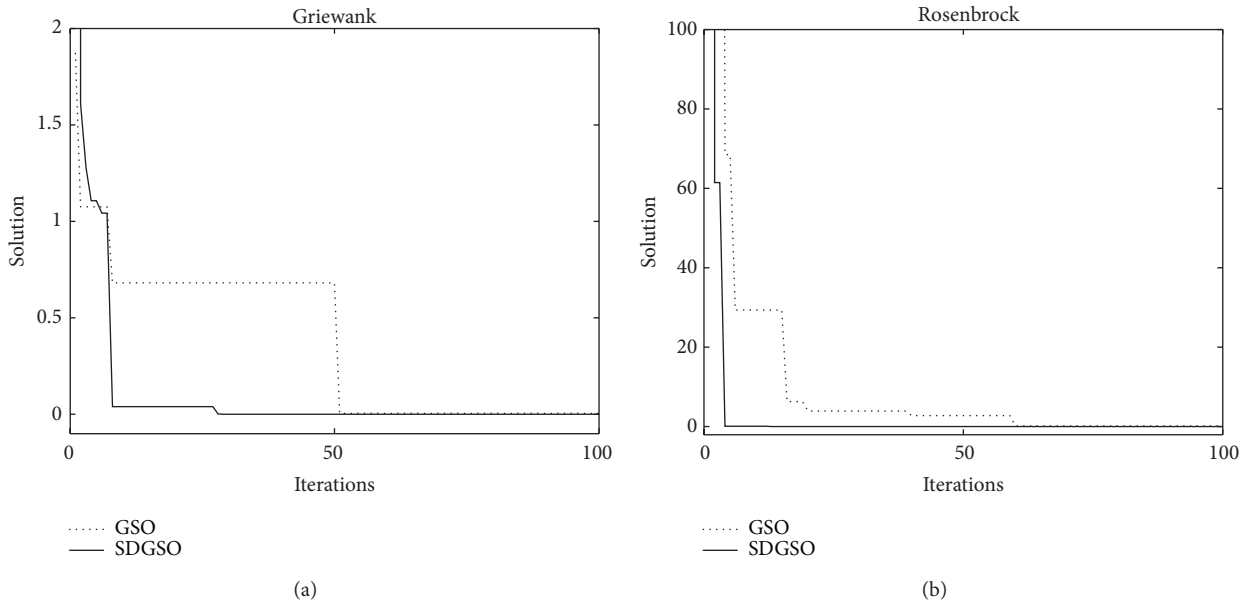

Figure 2: Convergence curves of GSO algorithm and SDGSO algorithm.

Table 3: P values produced by Wilcoxon's test.

SDGSO versus	DE	ABC	GSO
Rastrigin	$6.77E-08$	$6.79E-08$	0.025639
Griewank	$6.79E-08$	$6.79E-08$	$3.55E-04$
Rosenbrock	$6.79E-08$	$6.79E-08$	$1.41E-05$

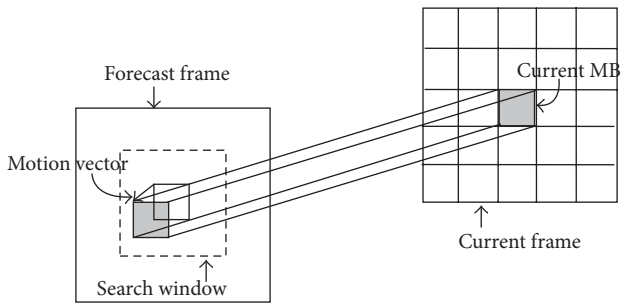

Figure 3: Principle diagram of ME based on BMA.

compared in a certain search pattern. Consequently the best matching block will be found, and the motion vector (MV) can be calculated as shown in Figure 3.

GSO algorithm has strong commonality and a good performance of the low dimensional multimodal function optimization, especially suitable for block matching optimization of 2D images. In this paper, a novel block motion estimation method based on GSO algorithm which combines the good global searching characteristic of GSO algorithm with the single-dimension perturbation strategy and features of the video sequence is proposed. The proposed method not only has improved the accuracy of ME but also has decreased the average number of searching points. Specific implementation process is given as follows.

3.1. Predict the Motion Vector and Initialize the Population. In BMA based on SDGSO algorithm, each macro block is regarded as glowworm, and the center location of the macro block is the location of glowworm. Initial population should be distributed within the search window of the reference frame after selecting the current macro block.

According to the statistics, video sequences have a characteristic called center biased distribution (CBD), and it means that most of the MVs will be concentrated in a central place. Besides, the MV of the current macro block has a similar size, and the same direction with its adjacent MVs, and the MV of the same block in the previous frame; that is to say, there is a strong space-time correlation among motion vectors. Therefore, we use the predictive MV to determine the search center. The average of the left block's MV, the top block's MV, the right-top block's MV and the same block's MV in the previous frame is regarded as the predictive MV, as shown in Figure 4(a), and then the initial population is distributed around the search center, as shown in Figure 4(b) (MV = (3, 1)).

3.2. Object Function and Luciferin Updating. ME based on BMA is looking for the MV which can get the minimum (or maximum) value of the matching function. Matching function is an important factor of algorithm's precision accuracy, and the mean average difference (MAD) is chosen as the matching function in this paper which has a small computation and convenient for hardware implementation, and it is defined as follows:

$$\mathrm{MAD}_i(t) = \frac{1}{N_1 N_2} \sum_{x \in \mathfrak{I}} \left| \varphi_2(x+d) - \varphi_1(x) \right|, \quad (6)$$

where $\varphi_1(x)$ is the pixel value of the current macro block x, $\varphi_2(x+d)$ is the pixel value of the reference block, and N_1 and N_2 are the horizontal and vertical size of macro blocks.

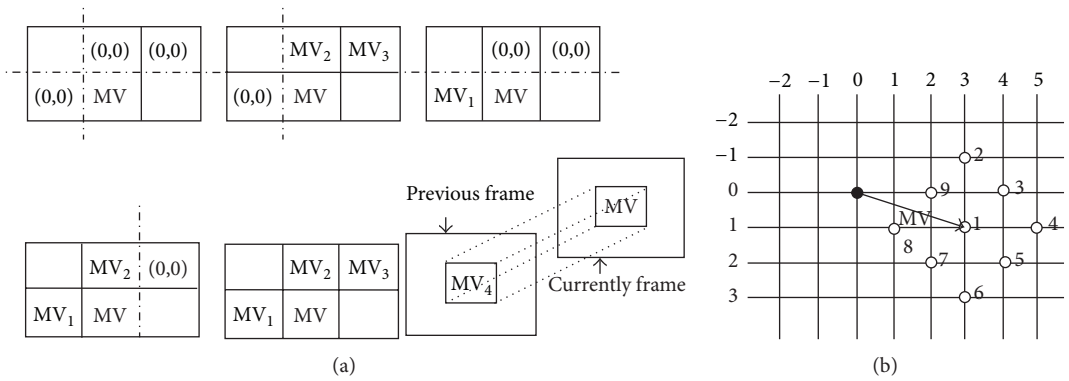

Figure 4: Motion vector predictor and the distribution of the initial population.

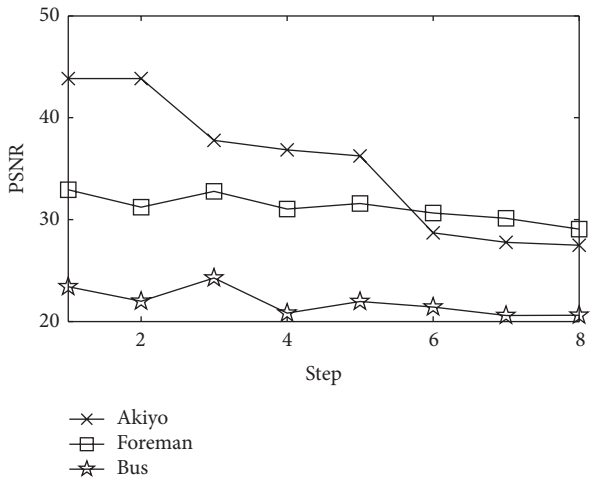

Figure 5: Diagram of step size and search accuracy.

Luciferin-update equation is associated with matching function, defined as follows:

$$l_i(t+1) = (1-ro) * l_i(t) - \gamma * \mathrm{MAD}_i(t). \quad (7)$$

3.3. Location and Local-Decision Domain Updating. In the process of location updating, size of the step has a great effect on the convergence speed, especially for different motion intensity of video sequences. For example, when the movement is smooth like Akiyo sequence, a small step size like $s = 1$ can get the highest search accuracy as shown in Figure 5, and when the movement is intense like Bus sequence, a small step size may slow down the convergence speed, and reduces accuracy. For example, the accuracy is lower when step size of $s = 1$ than that of $s = 3$ as shown in Figure 5.

Aiming at this problem, the adaptive step size strategy [22] is adopted in this paper, defined as follows:

$$s_i(t) = s_{\min} + (s_{\max} - s_{\min}) * H_i(t), \quad (8)$$

where $H_i(t) = \|x_i(t) - x_{\mathrm{best}}(t)\|/d_{\max}$ is called Luciferin factor, $x_i(t)$ is the position of glowworm i, $x_{\mathrm{best}}(t)$ is the position of the best glowworm which has the maximum luciferin value, $d_{\max}$ is the maximum distance between the best glowworm and other glowworms, and $s_{\max}$ and $s_{\min}$ are the maximum and minimum step size.

After figuring out the step size, the location of the glowworm should be updated according to (3), and then perform local search of SDPS to determine the final location at the current iteration. Finally, the local-decision domain should be updated according to (4).

3.4. Terminating Strategy. Due to the introduction of SDPS, the calculation of the objective function might be increased, but the speed of finding the optimized solutions is also accelerated. That is to say, the algorithm may find the optimal solution before it achieves the maximum iterations. Therefore, the following strategies are taken to terminate the iteration.

(1) Check whether the global optimal location has been changed within the prescribed continuous iterations and if it has not, the iteration should be terminated early.

(2) Check whether the maximum iteration is reached and if it is, terminate the iteration.

3.5. Algorithm Steps. For more details, see Algorithm 2.

4. Simulation Results and Performance Evaluation

In the experiments, 5 typical YUV video sequences are chosen according to the different motion intensity: Akiyo (very slow), Foreman (slow), Soccer (fast), Football (fast), and Bus (very fast). The first 100 frames of them are used as the test object, CIF format (352 × 188), 30 frames/second. Under the same test environment, full search (FS), three-step search (TSS), diamond search (DS), adaptive rood pattern search (ARPS), mutation particle swarm search optimization (MPSO) [23] algorithm, glowworm swarm optimization (GSO) algorithm, artificial bee colony (ABC) algorithm, and single-dimension perturbation glowworm swarm optimization (SDGSO) are tested respectively from two aspects of search precision and computation complexity.

Table 4: Average PSNR of each algorithm.

Algorithm	PSNR (dB)					
	Akiyo	Foreman	Soccer	Football	Bus	Average
FS	39.98	32.22	28.76	24.93	24.32	30.04
DS	39.98	31.94	27.64	23.54	21.46	28.91
TSS	39.98	31.55	27.60	24.00	22.12	29.05
ARPS	39.98	31.84	28.04	23.91	22.57	29.27
MPSO	39.89	31.74	28.04	24.06	23.06	29.36
SDGSO	39.91	31.79	28.27	24.10	23.32	29.48

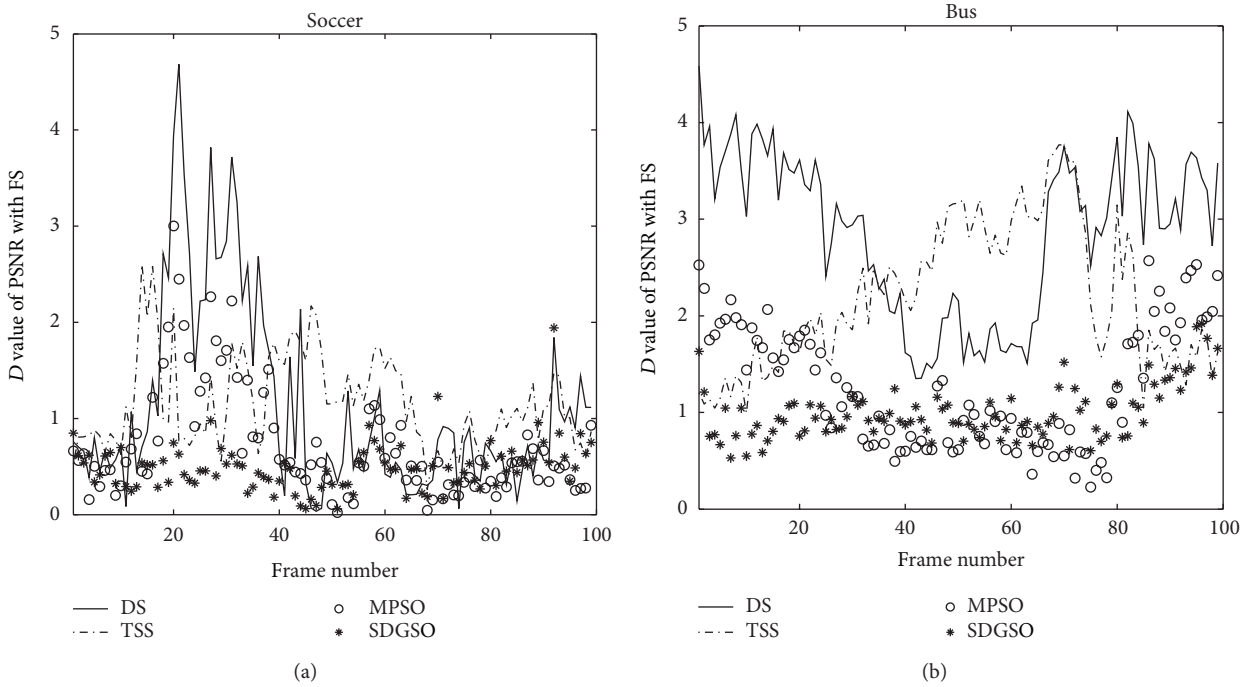

Figure 6: D value of PSNR with FS of each algorithm.

Table 5: Precision of ABC algorithm and GSO algorithm.

Algorithm	PSNR (dB)					
	Akiyo	Foreman	Soccer	Football	Bus	Average
ABC	38.38	31.39	27.96	23.42	23.07	28.84
GSO	39.90	31.75	28.19	23.93	23.23	29.40

Table 6: Average search points of each algorithm.

Algorithm	Average search points (round off)					
	Akiyo	Foreman	Soccer	Football	Bus	Average
FS	869	869	869	869	869	869
DS	11	15	25	34	20	21
TSS	28	31	31	31	31	30
ARPS	5	8	11	21	10	11
MPSO	44	46	51	40	49	46
ABC	38	42	43	47	42	42
GSO	49	51	50	52	50	50
SDGSO	30	34	33	43	35	35

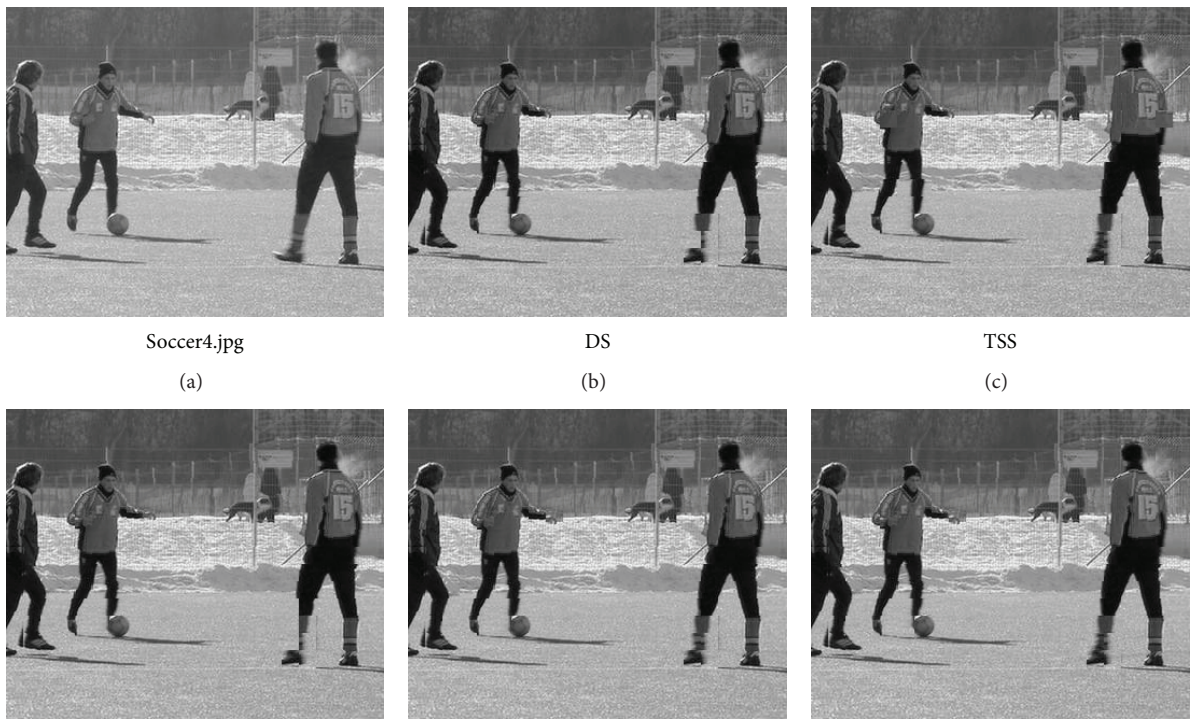

Figure 7: The restructuring images of the fourth frame of Soccer by using different algorithms.

Parameters setting: size of macro block is 16×16, scope of search window is $(-15, 15)$, $p = 15$. The previous frame is the reference frame. The population $n = 9$, and the maximum iterations $\max t = 5$. According to the reference value of GSO algorithm and the results of the experiment, set volatile coefficient of luciferin $\rho = 0.98$, update coefficient of luciferin $\gamma = 0.98$, minimum step size $s_{\min} = 1$, maximum step size $s_{\max} = 3$, constant factor $\beta = 0.8$, modulus $n_t = 6$, perceive domain $r_s = p$, and step size of local search $a = 1$.

4.1. Search Precision. In this paper, the peak signal-to-noise ratio (PSNR) is adopted as the evaluation criteria of search precision. The average PSNR value of the five test video sequences is listed in Table 4. Figure 6 illustrates the D value of PSNR with FS of Soccer and Bus sequences, respectively.

It can be seen from Table 4, that the average PSNR value of FS is the largest one, and it means that the search precision of FS is the highest one; DS has a high search precision for the slow motion like Akiyo and Foreman sequences, but for the fast motion sequence, its precision is significantly lower, that is to say, DS only applies to the slow motion; contrary to DS, TSS just applies to the fast motion; ARPS has superior performance compared to other fast motion estimation methods, and it has high search precision and strong applicability; although the average PSNR values of SDGSO and MPSO for slow motion like Akiyo sequence are slightly lower than those of DS and TSS, their overall performances are still higher than those of any other methods because of their high search precision for fast motion sequences as shown in Figure 6; SDGSO has higher search precision compared to MPSO no matter for slow motion or fast motion sequences; in other words, the image restructured by compensating with the motion vectors which are obtained by using SDGSO algorithm will be the most ideal one.

In order to show the good performance of SDGSO algorithm more intuitively, the restructuring images of the fourth frame of Soccer sequence are shown in Figure 7. It can be seen from Figure 7, that the fourth frame images of Soccer restructured by DS, TSS, and ARPS all miss some details, and the images restructured by MPSO and SDGSO algorithm have retained the details of the original image, especially for the latter one, the edge of its objects is almost smooth.

In order to prove that the application of GSO algorithm is better than that of ABC algorithm, the average PSNR value of the two algorithms which take the same population size, the same number of iterations, and the same prediction method of motion vector is listed in Table 5.

It can be seen from the results that the precision of ABC algorithm is worse than that of GSO algorithm no matter for video sequence with slow motion or that with fast motion. That is because the convergence speed of ABC algorithm is slower than that of GSO algorithm, and it cannot converge to the optimal value within the same iteration times. It is also worth noting that the precision of GSO algorithm in Table 5 has not been decreased compared to that of SDGSO algorithm in Table 4. The reason is that ABC and GSO algorithm in Table 5 did not adopt the early termination strategy like SDGSO algorithm, so their searching precisions

Step 1. Set parameters: population size n, dimension of space D, step size of local search a
Step 2. Update the location of all glowworms
Step 3. for each $i = 1 \ldots n$ doing
(3.1) $\vec{c} = \overrightarrow{x_i}$ {Candidate Solution}
(3.2) Randomly choose a dimension m to perturb $\{1 \le m \le D\}$, and generate a *rand* number, *rand*.
(3.3) if *rand* < 0.5 then let $step = -a$ else let $step = a$
(3.4) Let $\vec{c}\,[m] = \vec{c}\,[m] + step$
(3.5) if $f(\vec{c}) > f\left(\overrightarrow{x_i}\right)$ then $\overrightarrow{x_i} = \vec{c}$

Algorithm 1: Single-dimension perturbation search (SDPS).

Step 1. Read the data.
Step 2. Set parameters: ρ, γ, β, n_t, $s_{\min}$, $s_{\max}$, a.
Step 3. Set the size of block *mbSize* and divide the current frame image into z rules of block.
Step 4. for $k = 1$ to z do
(4.1) Predict the MV of the current macro block and the center of search window.
(4.2) Generate initial population of glowworms x_i $(i = 1, 2, \ldots, n)$
(4.3) for each $i = 1$ to n initializing the luciferin value $l(i)$;
(4.4) Set $t = 1$;
(4.5) for each $(t \le \max t)$ doing
(4.5.1) for each glowworm i doing
(4.5.1.1) Form the neighborhood $N_i(t)$;
(4.5.1.2) for each glowworm $j \in N_i(t)$, computing probability $P_{ij}(t)$ according to the formula (3);
(4.5.1.3) Select glowworm j using $P_{ij}(t)$;
(4.5.1.4) Update glowworm step s with the formula (8);
(4.5.1.5) Update glowworm position x_i with the formula (2);
(4.5.1.6) Search the new location by using SDPS;
(4.5.1.7) Update the luciferin value $l(i)$ according to the formula (8);
(4.5.1.8) Update local-decision domain $r_d^i(t+1)$ according to the formula (4);
(4.5.1.9) if x_i keep unchanged for lim$t$ generation then terminating the iteration;
Step 5. Output MV.

Algorithm 2: Single-dimension perturbation glowworm swarm optimization algorithm (SDGSO).

are relatively high, but it also means that more points need to be calculated; in other words, their high precision is at the cost of high computation complexity, as shown in Table 6.

4.2. Computation Complexity. In block matching motion estimation, the average number of search points for matching a block is used as the evaluation criterion of computation complexity. The average search point of each algorithm is listed in Table 6.

It can be seen in Table 6 that FS method searches every macroblock within the search window, so it needs to search 869 points for one matching block, DS and TSS are based on fixed template, the search point is relatively less, they need to search 20–30 points on average, ARPS is a kind of predictive adaptive algorithm, it has reduced the search points effectively, and the average search points for one matching block are less than 20. MPSO, ABC, GSO, and SDGSO are based on the swarm intelligence algorithm, so their search points are more than those of other fast MEs, but due to the introduction of SDPS, time-space correlation of video sequence and the effective termination strategies, the average search point of SDGSO has been reduced to a certain extent. Besides, the proposed algorithm has a good stability, and its average search point changes little for different test video sequences, so it is supposed to be a robust algorithm.

4.3. SDPS Local Search Strategy. In order to prove that the introduction of SDPS has improved the performance, the application of SDGSO algorithm in ME is compared with that of the adaptive step glowworm swarm optimization (AGSO) algorithm from two aspects of search precision and computation complexity as shown in Tables 7 and 8. The two algorithms have taken the same population size, same times of iterations, same prediction method of motion vector, and the same termination strategies.

Table 7: Average PSNR of AGSO and SDGSO.

	Akiyo	Foreman	Soccer	Football	Bus
AGSO	39.68	31.25	27.94	22.89	23.17
SDGSO	39.71	31.79	28.27	24.10	23.32

Table 8: Average search points of AGSO and SDGSO.

	Akiyo	Foreman	Soccer	Football	Bus
AGSO	31.10	30.24	33.28	36.91	36.66
SDGSO	30.00	34.20	33.31	43.21	34.90

It can be seen from Tables 7 and 8 that the search precision of SDGSO which has introduced SPDS is higher than that of AGSO, and the computation complexity of SDGSO is similar to that of AGSO, or even less.

5. Conclusions

In this paper, a novel glowworm swarm optimization algorithm based on single-dimension perturbation search strategy is proposed and applied to block motion estimation (Algorithm 1). The proposed algorithm has overcome the defect of being easy to fall into local optimum by taking advantage of the global optimization ability of GSO and the local optimization ability of SDPS. Besides, it also has reduced the computation complexity by taking the time-space correlation of video sequences and effective terminating strategies. All the experimental results show that SDGSO algorithm has improved both the search accuracy and convergence speed to a certain extant, especially for the block motion estimation of video sequences with fast motion, and good performance is reflected.

Acknowledgment

This research is supported by the National Science Foundation Council of Guangxi (2012GX NSFAA053227).

References

[1] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE Transactions on Image Processing*, vol. 5, no. 8, pp. 1266–1271, 1996.

[2] T. Koga and K. Iinuma, "Motion compensated inter-frame coding for video conferencing," in *Proceedings of the National Telecommunication Conference*, vol. C9, number 6, pp. 1–5, New Orleans, La, USA, 1981.

[3] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[4] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 313–317, 1996.

[5] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[6] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349–355, 2002.

[7] H. Jia and L. Zhang, "Directional diamond search pattern for fast block motion estimation," *Electronics Letters*, vol. 39, no. 22, pp. 1581–1583, 2003.

[8] X.-Y. Wang and J.-H. Zheng, "Adaptive rood pattern search for fast block-matching motion estimation," *Journal of Electronics and Information Technology*, vol. 27, no. 1, pp. 104–107, 2005.

[9] S. Li, W. Xu, N. Zheng et al., "A Novel fast motion estimation method based on genetic algorithm," *Acta Electronica Sinica*, vol. 6, no. 28, pp. 114–117, 2000.

[10] F. Liu and X.-Y. Pan, "Block motion estimation based on immune clonal selection," *Journal of Software*, vol. 18, no. 4, pp. 850–860, 2007.

[11] G. Y. Du, T. S. Huang, L. X. Song, and B. J. Zhao, "A novel fast motion estimation method based on particle swarm optimization," in *Proceedings of the 4th International Conference on Machine Learning and Cybernetics (ICMLC '05)*, pp. 5038–5042, Guangzhou, China, August 2005.

[12] X. Yuan and X. Shen, "Block matching algorithm based on particle swarm optimization for motion estimation," in *Proceedings of the International Conference on Embedded Software and Systems (ICESS '08)*, pp. 191–195, Sichuan, China, July 2008.

[13] K. N. Krishnanand and D. Ghose, "Detection of multiple source locations using a glowworm metaphor with applications to collective robotics," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 87–94, Pasadena, Calif, USA, June 2005.

[14] Y. Yang, Y. Zhou, and Q. Gong, "Hybrid artificial glowworm swarm optimization algorithm for solving system of nonlinear equations," *Journal of Computational Information Systems*, vol. 6, no. 10, pp. 3431–3438, 2010.

[15] Z. Huang and Y. Zhou, "Using glowworm swarm optimization algorithm for clustering analysis," *Journal of Convergence Information Technology*, vol. 6, no. 2, pp. 78–85, 2011.

[16] Y. Q. Zhou and Z. X. Huang, "Artificial glowworm swarm optimization algorithm for TSP," *Control and Decision*, vol. 27, no. 12, pp. 1816–1821, 2012.

[17] K. Cheng and L. Ma, "Artificial glowworm swarm optimization algorithm for 0-1 knapsack problem," *Application Research of Computers*, vol. 30, no. 4, pp. 996–998, 2013.

[18] Y. Zhou, G. Zhou, and J. Zhang, "A hybrid glowworm swarm optimization algorithm for constrained engineering design problems.," *Applied Mathematics and Information Sciences*, vol. 7, no. 1, pp. 379–388, 2013.

[19] D. R. de Oliveira, S. R. Parpinelli, and S. H. Lopes, "Bioluminescent swarm optimization algorithm," in *Evolutionary Algorithms*, E. Kita, Ed., pp. 69–84, InTech, Hampshire, UK, 2011.

[20] K. N. Krishnanand and D. Ghose, "Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications," *Multiagent and Grid Systems*, vol. 2, no. 3, pp. 209–222, 2006.

[21] F. Wilcoxon, "Individual comparisons by ranking method," *The International Biometric Society*, vol. 1, no. 6, pp. 80–83, 1945.

[22] Y. Ou and Y. Zhou, "Self-adaptive step glowworm swarm optimization algorithm," *Journal of Computer Applications*, vol. 31, no. 7, pp. 1804–1807, 2011.

[23] P. Zhang and P. Wei, "Fast motion estimation algorithm base on particle swarm optimization with mutation," *Journal of Electronic Measurement and Instrument*, vol. 25, no. 1, pp. 23–28, 2011.