Multimedia Systems (2021) 27:125–142
https://doi.org/10.1007/s00530-020-00716-y

REGULAR PAPER

A hybrid firefly and particle swarm optimization algorithm applied to multilevel image thresholding

Taymaz Rahkar Farshi[1] · **Ahad K. Ardabili**[2]

Received: 1 October 2020 / Accepted: 29 October 2020 / Published online: 19 November 2020

Abstract
There are many techniques for conducting image analysis and pattern recognition. This papers explores a way to optimize one of these techniques—image segmentation—with the help of a novel hybrid optimization algorithm. Image segmentation is mostly used for a semantic segmentation of images, and thresholding is one the most common techniques for performing this segmentation. Otsu's and Kapur's thresholding methods are two well-known approaches, both of which maximize the between-class variance and the entropy measure, respectively, in a gray image histogram. Both techniques were developed for bi-level thresholding. However, these techniques can be extended to multilevel image thresholding. For this to occur, a large number of iterations are required to account for exact threshold values. However, various optimization techniques have been used to overcome this drawback. In this study, a hybrid firefly and particle swarm optimization algorithm has been applied to yield optimum threshold values in multilevel image thresholding. The proposed method has been assessed by comparing it with four well-known optimization algorithms. The comprehensive experiments reveal that the proposed method achieves better results in term of fitness value, PSNR, SSIM, FSIM, and SD.

Keywords Image segmentation · Multilevel thresholding · Kapur's function · Otsu's function · Hybrid optimization

1 Introduction

There are many ways to conduct an image analysis or run pattern recognition. The use of image segmentation is growing, particularly because digital cameras are becoming more accessible for implementations. The process of image segmentation divides an image into homogenous regions and classifies the pixels. Some of the practical applications of image segmentation are satellite images [1], agriculture automation [2], medical images [3, 4], traffic control systems [5] and optical character recognition (OCR) [6].

An RGB image consists of three primary color channels [7]: red (R), green (G), and blue (B). In most cases RGB images are converted to grayscale because intensity can still be measured, and it is considered sufficient to showcase the possible performance gain in image segmentation.

There is a wide variety of image segmentation techniques, and thresholding methods based on image histograms are some of the most widely used segmentation methods. These histogram-based methods are very fast and effective compared to other existing segmentation methods because they only need to pass through the pixel once [8, 9]. Moreover, bi-level thresholding divides image pixels into two different homogenous regions, whereas multilevel thresholding divides image pixels into many homogenous regions. The between-class variance method (Otsu's method) [10] and the entropy criterion method (Kapur's method) [11] are two of the most well-known thresholding algorithms. Although these algorithms have been developed for bi-level thresholding, these algorithms can be extended for multilevel image thresholding. Because of iterative processes of these methods, which need to increase due to the thresholding values, the computational complexity of the algorithm increases exponentially; there are also problems regarding the length of time it takes these models to find the optimal solution. To overcome this obstacle we have turned to evolutionary

Communicated by Y. Zhang.

✉ Ahad K. Ardabili
ahad.ardabili@altinbas.edu.tr

[1] Software Engineering Department, Ayvansaray University, 34020 Istanbul, Turkey

[2] Department of Basic Sciences, Altınbaş University, 34217 Istanbul, Turkey

algorithm models [12–19]. Many of these models have specifically tackled the multilevel thresholding problem.

Research on multilevel image thresholding based on optimization methods has expanded over the past few years, with researchers building off of Otsu's and Kapur's works and integrating new techniques. Sathya and Kayalvizhi [20–22] developed a multilevel image thresholding method based on the minimum variance and maximum entropy criterions using an original bacterial foraging algorithm, an adaptive bacterial foraging algorithm, an amended bacterial foraging algorithm, and a modified bacterial foraging algorithm. Yin [23] created a particle swarm optimization (PSO) algorithm for the multilevel minimum cross entropy criterion, while Horng and Jiang [24] applied the ABC optimization algorithm to optimize the maximum cross entropy criterion. The multilevel thresholding method of Oliva et al. [25], based on Otsu's and Kapur's works, uses the harmony search optimization algorithm; Ayala and et al. [26] also employed an improved beta differential evolution algorithm to optimize Otsu's minimum variance criteria. Both Muppidi [27] Chao and et al. [15] suggested multilevel image thresholding techniques based on fuzzy entropy: Muppidi's utilizes a genetic algorithm (GA), while Chao et al.'s uses a modified gravitational search algorithm. Pal and et al. [28] worked with the spider monkey optimization algorithm to generate a multilevel thresholding technique based on Otsu's and Kapur's works, and Khairuzzaman and Chaudhury [29] brought together Kapur's entropy and Otsu's between-class variance for image segmentation using the grey wolf optimizer (GWO). More recently, Rahkar-Farshi [19] employed an animal migration optimization (AMO) algorithm for a multilevel thresholding method that maximizes Otsu's and Kapur's objective functions. These are all novel solutions, but there are still issues that need to be solved and optimized, such as computational complexity and the amount of time it takes to run the segmentation.

No optimization scheme can satisfy all aspect of all optimization and search problems. The no-free-lunch (NFL) theorem [30] states that the advantages gained by a optimization algorithm are offset by other problems and that no one algorithm works the best for all situations. Therefore, we need to develop new algorithms and try to enhance the performance of existing algorithms to tackle new class of problems. In this paper, a hybrid firefly and PSO (HFPSO) [31] algorithm is utilized to address the multilevel image thresholding problem.

The rest of the paper is organized as follows: In Sect. 2, Otsu's and Kapur's methods are described; in Sect. 3, we give a detailed overview of the particle swarm optimization (PSO) and the firefly optimization algorithms and introduce our hybrid PSO and firefly optimization algorithm. In Sect. 4, we describe our hybrid method in detail, in Sect. 5, we provide the comprehensive experimental results and discussion, and in Sect. 6 we give a summary and some concluding remarks.

2 Otsu's and Kapur's methods

The image segmentation process uses thresholding methods that are specifically created to maximize an objective function(s); in this case, the objective functions set forth by Otsu and by Kapur were used to determine the optimal thresholding values. The algorithms work to maximize these objective functions, and the thresholding results in a histogram that displays segmented groups that reflect the desired objectives of the thresholding process. $\{0,1,\cdots(L-1)\}$ is a range given to show the probability of ith gray level that is mathematically calculated per $p_i = h_i/(m \times n)$ where h_i indicates the number of pixels that corresponds gray level $i, 0 \le i \le (L-1)$ and where M and N correspond to the height and width of the image, respectively. According to Otsu's method, the objective function for bi-level thresholding is shown as:

$$J(t_1) = \sigma_0 + \sigma_1, \tag{1}$$

where,

$$\sigma_0 = \omega_0(\mu_0 - \mu_T)^2 \text{ and } \sigma_1 = \omega_1(\mu_1 - \mu_T)^2,$$

with,

$$\omega_0 = \sum_{i=0}^{t_1-1} p_i,\ \mu_0 = \sum_{i=0}^{t_1-1} \frac{i\,p_i}{\omega_0} \text{ and } \omega_1 = \sum_{i=t_1}^{L-1} p_i,\ \mu_1 = \sum_{i=t_1}^{L-1} \frac{i\,p_i}{\omega_1},$$

and,

$\mu_T = \sum_{i=0}^{L-1} i\,p_i,$

Therefore, it is easy to show that:

$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T$ and $\omega_0 + \omega_1 = 1$.

Otsu's objective function can be extended to multilevel model as follows:

$$J(t_1, t_2, \cdots, t_m) = \sigma_0 + \sigma_2 + \cdots \sigma_m, \tag{2}$$

where

$$\sigma_0 = \omega_0(\mu_0 - \mu_T)^2,$$

$$\sigma_1 = \omega_1(\mu_1 - \mu_T)^2,$$

$$\sigma_2 = \omega_2(\mu_2 - \mu_T)^2,$$

$$\vdots$$

$$\sigma_m = \omega_m(\mu_m - \mu_T)^2,$$

with

$$\omega_0 = \sum_{i=0}^{t_1-1} p_i,\ \mu_0 = \sum_{i=0}^{t_1-1} \frac{i p_i}{\omega_0},$$

$$\omega_1 = \sum_{i=t_1}^{t_2-1} p_i,\ \mu_1 = \sum_{i=t_1}^{t_2-1} \frac{i p_i}{\omega_1},$$

$$\vdots$$

$$\omega_m = \sum_{i=t_m}^{L-1} p_i,\ \mu_m = \sum_{i=t_m}^{L-1} \frac{i p_i}{\omega_m},$$

$[t_1, t_2, \cdots, t_m]$ states the m optimal thresholds for a gray scale image. There are some constraints, which are defined as:$t_1 < t_2 < \cdots < t_m$.

The objective function according to Kapur's method for bi-level thresholding is shown as:

$$J(t_1) = H_0 + H_2, \tag{3}$$

where

$$H_0 = -\sum_{i=0}^{t_1-1} \frac{p_i}{\omega_0} \ln \frac{p_i}{\omega_0},\ \omega_0 = \sum_{i=0}^{t_1-1} p_i,$$

$$H_1 = -\sum_{i=t_1}^{L-1} \frac{p_i}{\omega_1} \ln \frac{p_i}{\omega_1},\ \omega_1 = \sum_{i=t_1}^{L-1} p_i,$$

where H_0 and H_1 are partial entropies of histograms. The threshold value t_1 is the gray level that maximizes the objective function given in Eq. (3). Moreover, Kapur's entropy criterion can be extended for multilevel thresholding through the following equation:

$$J(t_1, t_2, \cdots, t_m) = H_0 + H_2 + \cdots H_m, \tag{4}$$

where,

$$H_0 = -\sum_{i=0}^{t_1-1} \frac{p_i}{\omega_0} \ln \frac{p_i}{\omega_0},\ \omega_0 = \sum_{i=0}^{t_1-1} p_i,$$

$$H_1 = -\sum_{i=t_1}^{t_2-1} \frac{p_i}{\omega_1} \ln \frac{p_i}{\omega_1},\ \omega_1 = \sum_{i=t_1}^{t_2-1} p_i,$$

$$H_2 = -\sum_{i=t_2}^{t_3-1} \frac{p_i}{\omega_2} \ln \frac{p_i}{\omega_2},\ \omega_2 = \sum_{i=t_2}^{t_3-1} p_i,$$

$$\vdots$$

$$H_m = -\sum_{i=t_m}^{L-1} \frac{p_i}{\omega_m} \ln \frac{p_i}{\omega_m},\ \omega_m = \sum_{i=t_m}^{L-1} p_i,$$

3 Hybrid firefly and particle swarm optimization

Since each optimization algorithm has its own advantages and disadvantages, a hybrid that combines these algorithms would allow for an increased robustness and more flexibility to tackle complex problems [32]. This section is devoted to a review of a hybrid algorithm (HFPSO) that combines the firefly and the particle swarm optimization algorithms, originally proposed by Aydilek [31]. In the following, we first review some basic principles of the PSO and firefly algorithms, then discuss the combination these two algorithms in a hybrid model.

3.1 Particle swarm optimization

Particle swarm optimization (PSO) is a population-based optimization algorithm that was introduced by Russell Eberhart and James Kenned in [6]. The model was originally inspired by the collective movements exhibited by bird and fish swarms while searching for food or escaping from perceived threats.

The PSO algorithm has shown better performance in comparison to many other search algorithms because it is quick to find the results, uses less parameters, and has a lower chance of getting stuck at the local optima. The PSO algorithm starts with a bunch of random solutions, which are called particles, and a community of particles is called a crowd or flock. Initialization of ith particle is mathematically calculated per:$x_i = lb + r\text{and} \times (ub - lb)$. Here ub and lb are lower and upper bund of the problem space. The search process starts by giving each particle a random value in the solution space. At every iteration each individual particle position x_i will get updated based on its best position p_1 and the best position of the other particles $g$best in its topological neighborhood. The new position will be found by adding velocity vector v_i^t. PSO is an iterative algorithm; the position components are updated at each iteration by calculating the particle's velocity components using Eq. (5a).

$$v_i^t = w v_i^{t-1} + R_1 C_1 (p\text{best}_i - x_i) + R_2 C_2 (g\text{best} - x_i). \tag{5a}$$

$$x_i = x_i + v_i^t. \tag{5b}$$

Here v_i^t and x_i represent the velocity and position of ith particle at iteration t respectively, w is the inertia weight, which plays an important role in the exploration/exploitation

aspects of the PSO algorithm, and $pbest_i$ and g_{best} represent the position of the best solution found so far by ith particle and the best overall solution, respectively.R_1 and R_2 are two random numbers generated from a uniformly distributed range [0,1]. C_1 and C_2 are the cognition and social learning factors, respectively, since C_1 is multiplied by the distance to the best position of an individual particle and C_2 is multiplied by the distance to the best position among all the particles. The main steps of the PSO algorithm can be expressed in the form of the pseudo-code covered in Algorithm 1.

Algorithm 1. PSO Algorithm

Initialize the first population, $\boldsymbol{x_i}, (i = 1,2,\ldots n)$ *and velocities* $\mathbf{V_i}$ *of population vector*
t=0;
while *the end criterion is not satisfied*
    t++;
    if $f(x_i) > f(p_{best,i})$
        $pbest_i = x_i$
        update $gbest$
    end
    update x_i (Eq. 5)
end

3.2 The firefly optimization algorithm

The firefly algorithm (FA) is a well-known optimization algorithm based on swarm intelligence and was first introduced in 2008 by Yang [33], who was inspired by the natural behavior of fireflies. Fireflies use their flashing lights for several reasons: the light can help them find a mating partner based on the pattern of flashing, it can be a means to attract prey, or it can be a way to deter predators. The bioluminescence light is produced when oxygen combines with calcium, the luciferin enzyme, and adenosine triphosphate [34]. The communication of fireflies through flashing lights and their subsequent behaviors inspired Xin-She Yang to propose a metaheuristic optimization algorithm. The firefly algorithm is not a detailed simulation of the behavior of fireflies but rather an idealized version of it. For this purpose, one need to assume that (1) all the fireflies are unisex (so there is no difference between females and males in terms of attraction); (2) mutual attraction between two fireflies is proportional to the intensity of the light emitted and observed by each of them; and (3) the brightness depends on the objective function, which is the subject of optimization. Therefore, the intensity of the light produced by each firefly depends on how optimal its position is and is proportional to its fitness value. In each itaration the intensity of the light of each firefly will be compared to the intensity of the light of the other fireflies, and those which are less bright will move towards brighter ones. The amount of light perceived by other fireflies depends on the distance; fireflies receive lights with varying intensities. The firefly with best fitness will search the space in a random manner to find the optimum solution. The mathematical formula that describes the motion of a less bright firefly moving toward the brighter one is given as follows:

$$x_i = x_i + \beta_0 e^{-\gamma r_{i,j}^2}\left(x_j - x_i\right) + \alpha(\text{rand} - 0.5). \tag{6}$$

Here β_0 is the coefficient of the maximum attraction between fireflies at positions x_i and x_j, with Euclidean distance r_{ij}. γ is the light absorption coefficient, and α is the coefficient of random displacement vector. The pseudo-code of the firefly algorithm is given in Algorithm 2.

Algorithm 2. Firefly Algorithm

Initialize the first population of fireflies, $\boldsymbol{x_i}, (i = 1,2,\ldots n)$
Calculate the fitness of fireflies
Define light absorption coefficient γ
while $(t < NumberofIteration)$
  for $i = 1: n$
    for $j = 1 : i$
      if $(I_j > I_i)$
        Move firefly i towards j in d-dimension;
      end
      Attractiveness varies with distance r*via* $exp[-\gamma r]$
      Evaluate new solutions and update light intensity
    end
  end
  Rank the fireflies and find best
end

3.3 Hybrid firefly and PSO

It has been demonstrated that for some problems the PSO algorithm has faster convergence in comparison to other algorithms. However, it has the potential to slow down as it approaches the global optimal point. Since the algorithm adjusts itself at each iteration with the global best position, this increases the risk of getting stuck in the region of local optima. The other drawback to the PSO algorithm is its dependency on search parameters. Different parameter settings can lead to different convergence speeds [35]. Many variants of the PSO algorithm have been proposed to enhance its performance, with the main goal being to balance the exploratory and exploitatory aspects. These proposals include changes in the strategy of the evolution, tuning the parameters, changing the updating rules, and incorporating better evolving strategies. For the method proposed in this paper, the velocity, calculated using Eq. (5a), will be substituted in the formula of Eq. (5b) to update the position of the particles in the next stage. If this velocity is too slow or too fast, it can lead to difficulties such as oscillation around the solution, which can cause the speed of convergence to decrease. To tackle this problem, we need to adjust the inertia weight w and acceleration coefficients (c_1, c_2) and then calculate velocity according to Eq. (5a). These adjustments are challenging problem: the FA does not include velocity in its algorithm, but it has also been shown that the FA is more efficient and has higher success rate in multimodal problems in comparison to PSO [33]. The intent behind the hybrid firefly and particle swarm optimization algorithm (HFPSO) [31] algorithm proposed by Aydilek is to benefit from the strengths of each of these algorithms. The PSO algorithm will help with exploration, while the FA will take care of local search. Meanwhile the inertia weight will be updated dynamically.

The first step in the HFSPO algorithm is to initialize all of the parameters. Then the positions and velocities of particles will be initialized to random values (in the predetermined ranges). The fitness, global best ($g$best), and personal best ($p\text{best}_i$) will be calculated as the next step. Then the particle's fitness in the current stage and its last iteration according to Eq. (8) will be compared. If the fitness value of particle is the same or improved, the FA will take over and local search will start; otherwise, the PSO algorithm continues according to Eq. 5. IF the FA algorithm takes over, then the position and velocity will be calculated according to Eqs. ((9 and ((10. In the next step the position and velocity ranges are checked for all the fireflies and particles. If the fitness value reaches the desired value, the algorithm will be terminated, and the final result will be given as output.

$$w = w_i - \left(\frac{w_i - w_i}{\text{iteration}_{\max}}\right) \times \text{iteration}. \tag{7}$$

$$f(i,t) = \begin{cases} true, if fitness(\text{particle}_i^t) \le g\text{best}^{t-1} \\ false, if fitness\left(\text{particle}_i^t\right) > g\text{best}^{t-1}. \end{cases} \cdot \tag{8}$$

$$x_i(t+1) = x_i(t) + B_0 e^{-\gamma r_{ij}^2}\left(x_i(t) - g\text{best}^{t-1}\right) + \alpha\left(\text{r and} - \frac{1}{2}\right). \tag{9}$$

$$V_i(t+1) = x_i(t+1) - x_{i-\text{temp}} \tag{10}$$

Here, indices i and t refer to the number of the step and the number of the particle, respectively. Choosing the right strategy to control the inertial weight w parameter is helpful in balancing the exploratory and exploitary aspects of the PSO algorithm. Here a linearly decreasing inertia weight is implemented, which will help the PSO to converge better [36]. The pseudo-code of HFPSO is given below:

Fig. 1 Images used for performance evaluation

Table 1 Parameter settings for the optimization algorithms

Algorithm	Initial values of control parameters
HFPSO	(C_1 and C_2) = 1.49445, maximum velocity (V_{max}) = 0.1 × search range ($X_{max} - X_{min}$), where minimum velocity (V_{min}) = $-V_{max}$. Inertia weight (w), w_i = 0.9, w_f = 0.5 are used according to Eq. (7). β_0 = 1, γ = 5, α_0 = 0.5
AMO	Neighborhood length = 5, δ is a random number controlled by a Gaussian distribution
PSO	Cognitive coefficient (C_1 and C_2) = 0.88, inertia weight (W) = 0.91,
GA	Roulette Selection, CR = 0.8, Gaussian Mutation scale = 2 shrink = 1
FA	β_0 = 1, γ = 5, α_0 = 0.5

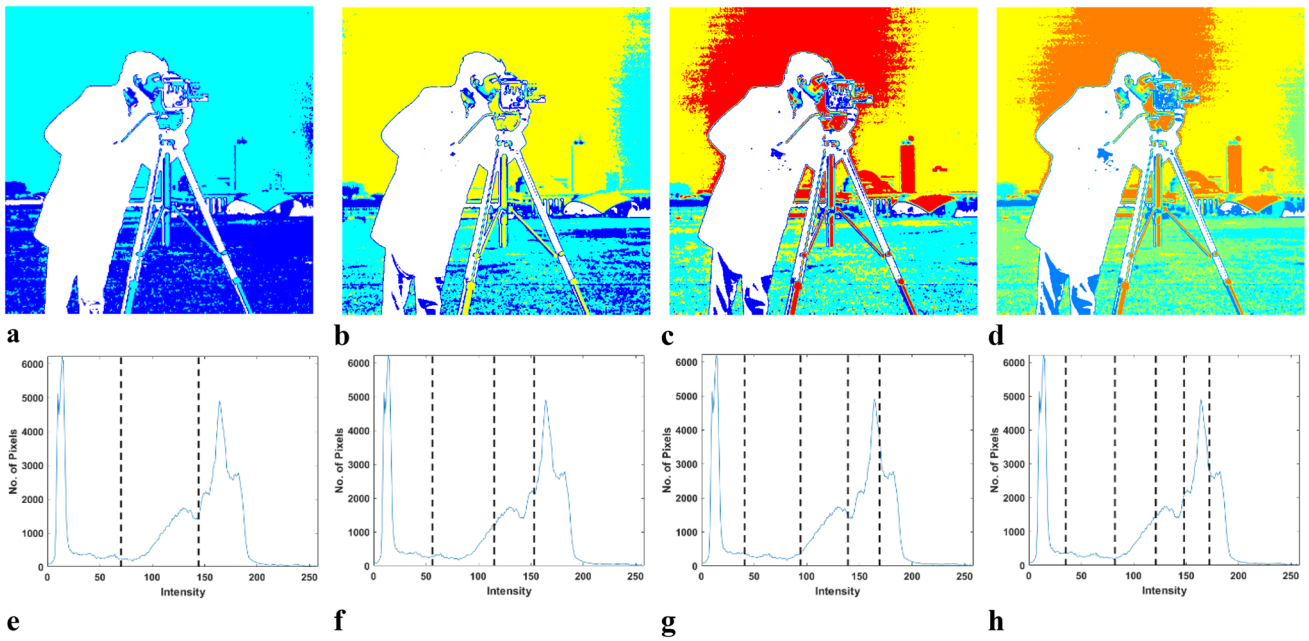

Fig. 2 Segmented Cameraman with located thresholds

Algorithm 3. HFPSO Algorithm

Initialize the first population of particles, $\boldsymbol{x_i}, (i = 1,2,\ldots n)$ *and velocities* $\boldsymbol{V_i}$ *of population vector*
Initialize all original parameters of PSO, minimum and maximum velocity range limits (V_{min}, V_{max})
Calculate the fitness of fireflies
t=0;
while *the end criterion is not satisfied*
t++;
if $f(x_i) > pbest_i$ *Eq*(8)
$x_{i-temp} = x_i$
update x_i and V_i Eqs. (9) and (10)
else
update w Eq. (7)
update x_i and V_i Eq.(5a)
end
update gbest
end

4 Multilevel thresholding using HFPSO

In this section we will use the HFPSO algorithm to search in the threshold's space. The aim of the optimization is to find the thresholds that will give the desired segmentation of the histogram. This can be done by maximizing the Otsu objective function or Kapur's entropy criterion given in Eqs. (2) and (4), respectively. The dimension (D) of the optimization algorithm is equal to the number of thresholds ($x = [t_1, t_2, ..., t_D]$). The gray level q in an image is bounded from below by 0, and its upper bound is 255 ($0 \le q < L - 1 = 255$). Thresholds are then subjected to the constraint $1 < t_1 < t_2 < ... < t_D < L$.

In the HFPSO algorithm the first step is to initialize the thresholds randomly as follows:

$$x_{j,i,0} = x_{j,\min} + r\text{and}_{j,i}[0,1] \times (x_{j,\max} + x_{j,\min)}. \quad (11)$$

The randomizer r and$_{i,j}$ gives a number that is randomly generated from a uniform distribution in the interval [0,1]. Here i is the number of the particles ($i = 1, \ldots, N$). The number of dimensions is shown by $j = 1,2, \ldots, D$.

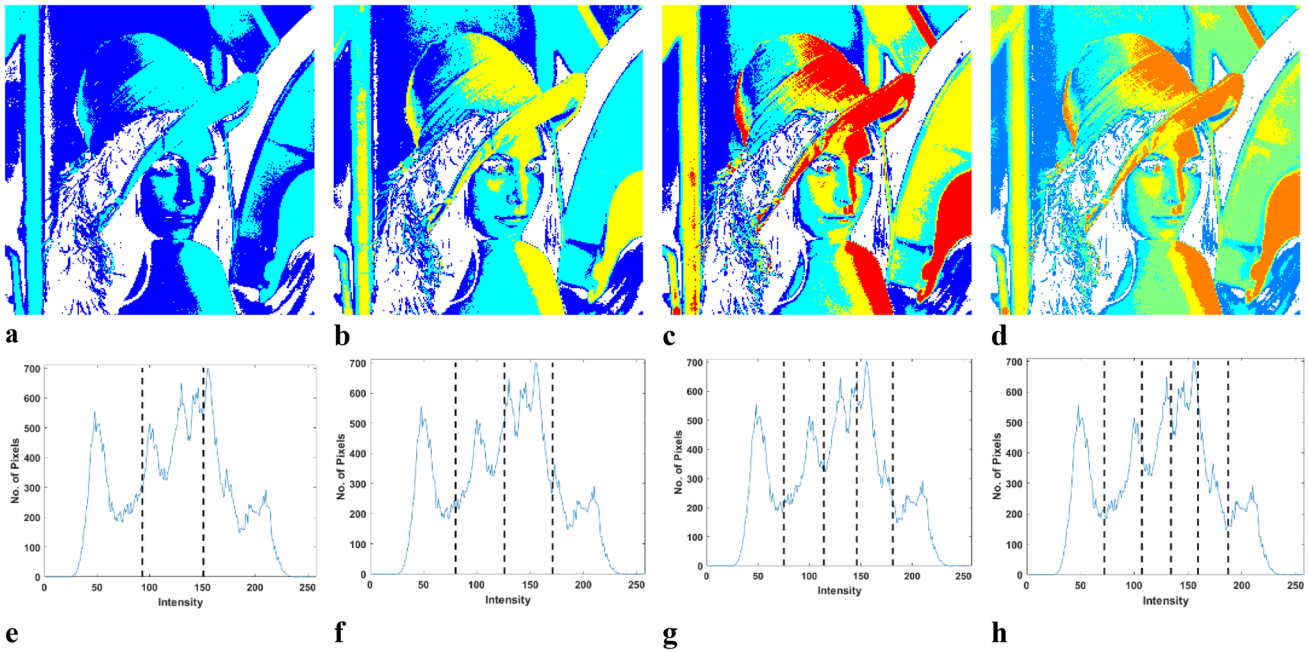

Fig. 3 Segmented Lena with located thresholds

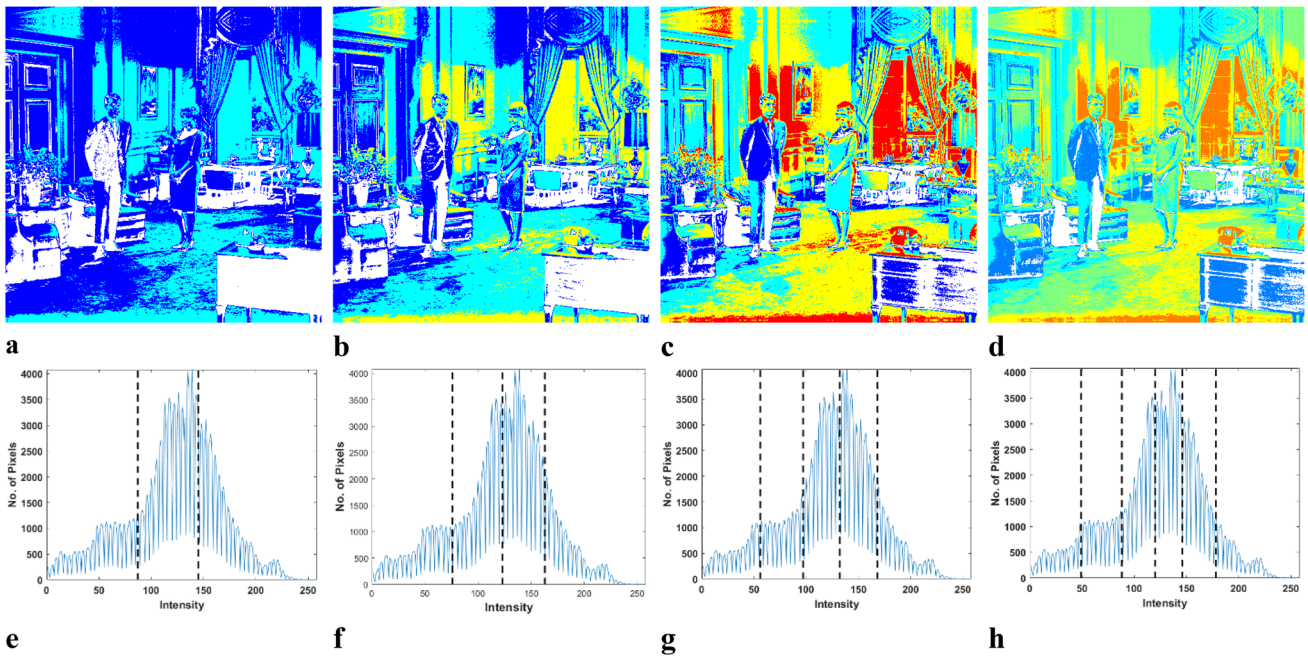

Fig. 4 Segmented Livingroom with located thresholds

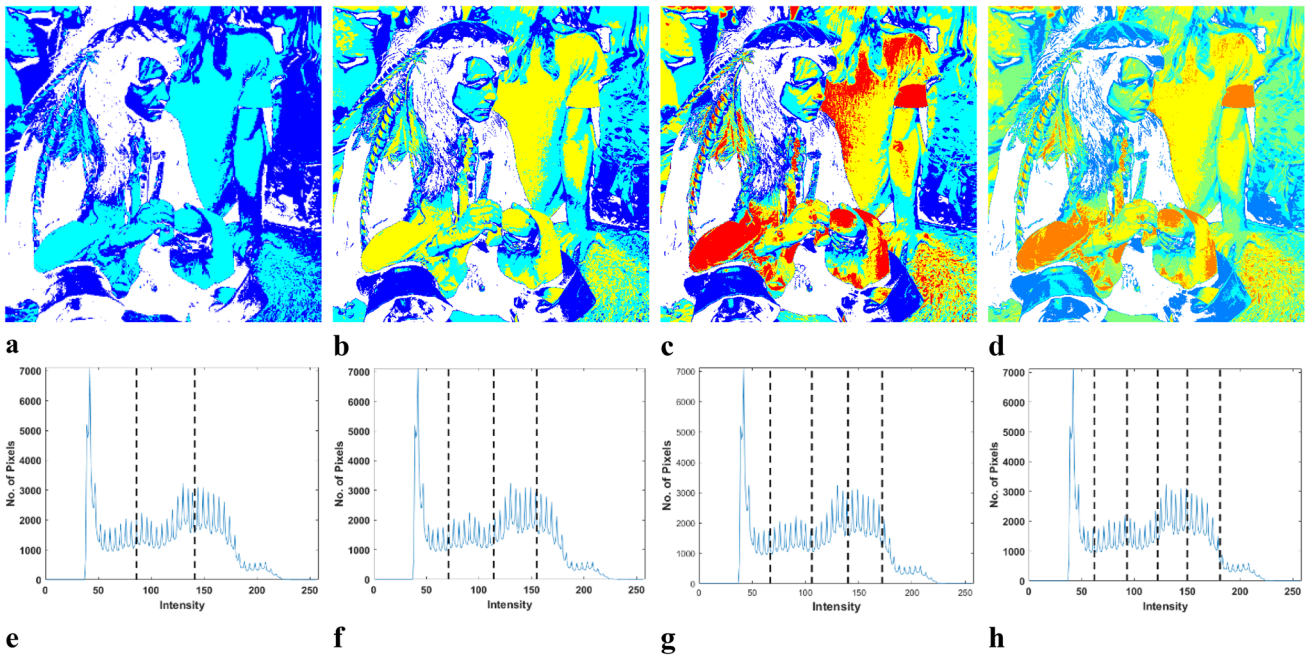

Fig. 5 Segmented Hunter with located thresholds

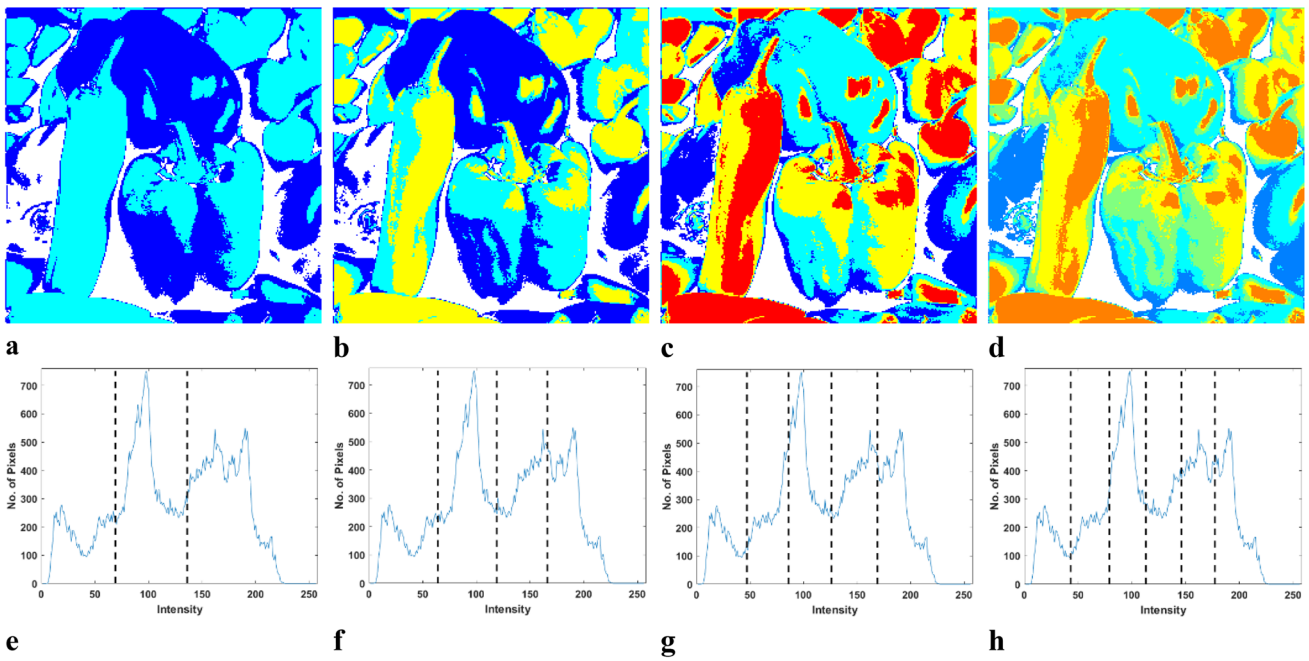

Fig. 6 Segmented pepper with located thresholds

5 Experimental results and discussion

To compare different algorithms with the proposed algorithm, we carried out computations using MATLAB R2018 on PC with Intel Core (TM) i7- 3632QM 2.2 GH processor and 8 gigabytes of RAM. We evaluated the efficiency of these algorithms by applying them to the benchmark images: "Cameraman," "Lenna," "Livingroom," "Hunter," "Pepper," and "Mandrill." The images are shown in Fig. 1. These

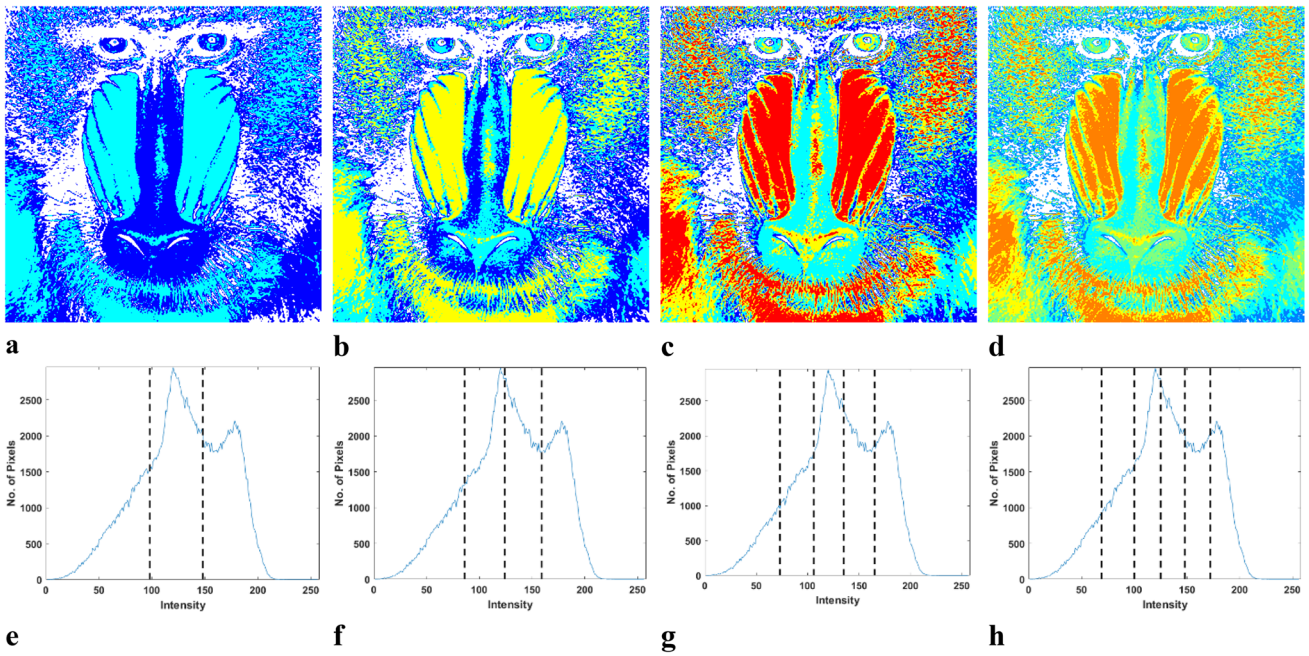

Fig. 7 Segmented Mandrill with located thresholds

images are difficult to segment using the more traditional, bi-level thresholding methods because their gray scale image histograms are multimodal. The multimodal histograms of "Lenna" and "Pepper" exhibit many valleys and peaks, while "Livingroom" and "Hunter" are characterized by extreme changes in the number of pixels. "Mandrill's" histogram has a smooth distribution in the gray level in comparison to "Cameraman."

The performance of the algorithm that we implemented here depends on the parameters. To perform a fair comparison among the proposed algorithm and the other algorithms—AMO, PSO, GA, and FA—we used the following parameters: the number of agents (animals, particles, chromosomes, and fireflies in AMO, PSO, GA, and FA, respectively) and number of maximum generations for all algorithms is taken to be 200. Table 1 contains the parameters of the different optimization algorithms we used in the experiments.

To compare the outcome of different algorithms, we evaluated them both visually and numerically. Figure 2 depicts the segmented version for different levels of thresholding ($m = 2$, 3, 4, and 5).

Tables 2, 3, 4, 5, 6, 7 give the numerical evaluation of algorithms. The performance of the algorithms was evaluated using well-known indicators: peak signal to noise ratio (PSNR), structural similarity index measure (SSIM), Feature SIMilarity (FSIM), standard deviation (SD), and fitness value. The mathematical representation of PSNR is as follows:

$\mathrm{PSNR} = 10\log_{10}(\frac{255}{\mathrm{MSE}})$, with mean squared error

$$\mathrm{MSE} = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x,y) - \tilde{I}(x,y)]]^2. \tag{12}$$

where $I(x, y)$ is the value of a pixel of original image located at (x, y), and $\tilde{I}(x, y)$ is the segmented image value at (x, y). The size of the images is M × N. Also, SSIM is calculated using the following equation:

$$\mathrm{SSIM}(I, \mathrm{Seg}) = \frac{(2\mu_I \mu_{seg} + c_1)(2\sigma_{I,seg} + c_2)}{(\mu_I^2 + \mu_{seg}^2 + c_1)(\sigma_I^1 + \sigma_{seg}^2 + c_2)} \tag{13}$$

where μ_{seg} and μ_I indicate the mean gray value of the segmented and original image, respectively.σ_{seg} and σ_I are the mean standard deviation of the segmented and original image, respectively. $\sigma_{I,\mathrm{seg}}$ is the covariance of I and Seg. Here, c_1 and c_2 are replaced by 6.5025 and 58.52252, respectively. Additionally, FSIM is computed as follows:

$$\mathrm{FSIM} = \frac{\sum_{w \in \Omega} S_L(w) PC_m(w)}{\sum_{w \in \Omega} PC_m(w)}, \tag{14}$$

where Ω indicates the entire domain of the image and their value is calculated as foloows:

Table 2 Comparison of best obtained objective and optimal threshold values based on Kapur

Test images	No. of thresholds	Average fitness value					Average optimal threshold values				
		HFPSO	AMO	PSO	GA	FA	HFPSO	AMO	PSO	GA	FA
Cameraman	2	12.28018	12.28018	12.28018	12.28018	12.28000	125, 196	125, 196	125, 196	125, 196	125, 196
	3	15.38796	15.38795	15.38796	15.38795	15.38485	44, 102, 196	44, 102, 196	44, 102, 196	44, 102, 196	43, 102, 196
	4	18.54972	18.54972	18.54972	18.54912	18.52541	42, 96, 145, 198	42, 96, 145, 198	42, 96, 145, 198	43, 96, 145, 197	41, 95, 144, 197
	5	21.26134	21.25441	21.26134	21.26666	21.24315	42, 96, 145, 191, 222	41, 95, 144, 191, 222	42, 96, 145, 191, 222	38, 83, 126, 172, 209	28, 69, 110, 155, 202
Lenna	2	12.34404	12.34404	12.34404	12.34404	12.34394	97, 165	97, 165	97, 165	97, 165	97, 165
	3	15.31581	15.31581	15.31581	15.31581	15.31363	82, 126, 175	82, 126, 175	82, 126, 175	82, 126, 175	83, 127, 176
	4	18.00613	17.99322	18.00167	18.00285	17.98513	64, 98, 138, 179	71, 108, 148, 184	67, 102, 142, 182	66, 100, 139, 179	69, 105, 143, 181
	5	20.60132	20.60047	20.60132	20.60088	20.53378	64, 95, 128, 163, 194	64, 95, 128, 163, 194	64, 95, 128, 163, 194	64, 95, 128, 163, 194	63, 93, 127, 163, 195
Livingroom	2	12.40587	12.40586	12.40587	12.40587	12.40577	94, 175	94, 175	94, 175	94, 175	93, 174
	3	15.55251	15.51251	15.31740	15.55208	15.54024	47, 103, 175	50, 110, 180	67, 135, 204	47, 104, 175	48, 105, 175
	4	18.47075	18.29950	18.22663	18.46680	18.44283	47, 98, 149, 197	47, 101, 165, 222	47, 103, 173, 233	47, 99, 149, 196	47, 97, 150, 195
	5	21.12884	21.07808	21.09465	21.13683	21.08844	44, 91, 133, 172, 212	44, 92, 137, 183, 235	47, 98, 145, 188, 236	45, 88, 127, 165, 202	41, 84, 123, 165, 202
Hunter	2	12.00334	12.00334	12.00334	12.00334	12.00331	112, 179	112, 179	112, 179	112, 179	112, 179
	3	14.97478	14.97454	14.97481	14.97453	14.97051	86, 129, 180	86, 130, 180	86, 129, 180	86, 131, 180	87, 133, 181
	4	17.65848	17.65820	17.62709	17.65806	17.62924	76, 114, 149, 183	76, 114, 149, 183	77, 115, 151, 186	76, 114, 149, 183	76, 113, 148, 183
	5	20.10976	19.96223	20.00623	20.09958	20.02444	66, 95, 124, 154, 183	73, 107, 142, 178, 211	72, 107, 140, 172, 208	68, 97, 125, 154, 183	67, 97, 126, 157, 187
pepper	2	12.58520	12.5852	12.58520	12.58520	12.58514	77, 147	77, 147	77, 147	77, 147	77, 147
	3	15.62384	15.62384	15.62384	15.62384	15.62273	61, 112, 164	62, 112, 164	61, 112, 164	61, 112, 164	62, 113, 165
	4	18.45446	18.44723	18.43819	18.45141	18.44130	57, 101, 145, 191	55, 98, 142, 188	52, 91, 136, 181	57, 102, 145, 190	58, 105, 150, 195
	5	21.19279	21.19196	21.19279	21.19255	21.15207	45, 78, 114, 153, 195	44, 78, 114, 154, 195	45, 78, 114, 153, 195	45, 78, 114, 154, 195	42, 77, 114, 153, 195
Mandrill	2	12.12446	12.12446	12.12446	12.12446	16.17889	80, 143	80, 143	80, 143	80, 143	80, 143
	3	15.13098	15.13097	15.13098	15.13098	19.18426	56, 104, 153	56, 104, 153	56, 104, 153	56, 104, 153	57, 105, 154
	4	17.89466	17.89369	17.89613	17.89355	21.07016	46, 84, 121, 162	46, 84, 123, 163	44, 81, 118, 160	47, 85, 124, 163	46, 84, 122, 163
	5	20.50841	20.50273	20.50841	20.50377	22.50820	40, 73, 106, 139, 172	40, 74, 108, 141, 174	40, 73, 106, 139, 172	42, 75, 108, 140, 172	39, 73, 106, 140, 175
Average rank		**1,333,333**	3,166,667	2,166,667	2,166,667	3,916,667					

Table 3 Comparison of standard deviation for based on Kapur

Test images	m	Standard deviation				
		HFPSO	AMO	PSO	GA	FA
Cameraman	2	0	0	0	0	0.000445
	3	0	0.000058	0	0.000058	0.001732
	4	0	0	0	0.000442	0.017537
	5	0	0.017974	0	0.026591	0.030404
Lenna	2	0	0	0	0	0.000284
	3	0	0.000029	0	0	0.001294
	4	0.003119	0.010953	0.010367	0.006775	0.008164
	5	0	0.001825	0	0.001235	0.030825
Livingroom	2	0	0.000062	0	0	0.000209
	3	0	0.135441	0.249758	0.001005	0.006737
	4	0	0.130444	0.073166	0.004961	0.009016
	5	0.026311	0.018067	0	0.017017	0.024339
Hunter	2	0	0	0	0	0.000065
	3	0.000139	0.000944	0	0.000346	0.002245
	4	0	0.001416	0.108648	0.000914	0.012181
	5	0	0.047041	0.079500	0.009460	0.038341
pepper	2	0	0	0	0	0.000148
	3	0	0.000005	0	0.000004	0.001034
	4	0.015222	0.018778	0.020611	0.016801	0.009540
	5	0	0.001212	0	0.000348	0.017309
Mandrill	2	0	0	0	0	0.000086
	3	0	0.000052	0	0	0.001111
	4	0.002304	0.002267	0.001512	0.002195	0.004870
	5	0	0.008513	0	0.004116	0.017606
Average rank		**1,375**	3,125	1,916,667	2,125	4,416,667

$$
\begin{aligned}
S_L(w) &= S_{PC}(w)S_G(w)S_{PC}(w) \\
&= \frac{2PC_1(w)PC_2(w)+T_1}{PC_1^2(w)+PC_2^2(w)+T_1}S_G(w) \\
&= \frac{2G_1(w)G_2(w)+T_2}{G_1^2(w)+G_2^2(w)+T_2}.
\end{aligned} \tag{15}
$$

The gradient magnitude (GM) of a digital image and is defined by G, and the value of PC that is the phase congruence is computed per: $G = \sqrt{G_x^2 + G_y^2}$ and $PC(w) = E(w)/(\varepsilon + \sum_n A_n(w))$, here $A_n(w)$ indicates the local amplitude on scale n and $E(w)$ is the magnitude of the response vector in w on n, ε is a small positive number and finally, $PC_m(w) = \max(PC_1(w), PC_1(w))$. After all to calculate the standard deviation (SD), we use the following formula:

$$
\sigma = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \mu\right)^2, \tag{16}
$$

where N is the number of runs, x_i is the fitness value obtained in ith run, and μ is mean value obtained over all runs of the algorithm.

We used the AMO, PSO, GA, and FA algorithms to optimize the corresponding objective functions. These results were obtained by applying the algorithms to the 6 test images; we ran each algorithm 30 times over the test images. In Table 2, we report the thresholds and the corresponding fitness values, which were obtained and averaged from the 30 runs for each algorithm on each test image, for Kapur's objective function. Additionally, Table 5 shows the corresponding thresholds and average fitness values for Otsu's objective function.

Figures (2, 3, 4, 5, 6, 7) show the segmented "Cameraman", "Lenna", "Livingroom", "Hunter", "Pepper" and "Mandrill" images using Otsu's method with various "m" levels. These images highlight the segments identified using a different color palette than the original image. The figures to the right of each image depict the histograms, which are superimposed (shown by dashed vertical lines) with multilevel thresholds. These images show that increasing the number of the levels (m) helps identify more details in the picture. However, this also increases the risks of splitting a segment in the image into many more segments. For each image, we first ranked the fitness values for each number of thresholds of each algorithm. After taking the average

Table 4 Comparison of PSNR for based on Kapur

Test images	m	PSNR					SSIM					FSIM				
		HF-PSO	AMO	PSO	GA	FA	HF-PSO	AMO	PSO	GA	FA	HF-PSO	AMO	PSO	GA	FA
Cameraman	2	13.775	13.775	13.775	13.775	13.775	0.5146	0.5146	0.5146	0.5146	0.5146	0.7121	0.7121	0.7121	0.7121	0.7121
	3	14.426	14.426	14.426	14.426	14.422	0.6046	0.6046	0.6046	0.6046	0.6047	0.8319	0.8319	0.8319	0.8319	0.8322
	4	20.155	20.155	20.155	20.168	20.054	0.6637	0.6637	0.6637	0.6633	0.6671	0.8768	0.8768	0.8768	0.8763	0.8810
	5	20.279	20.171	20.279	20.275	21.532	0.6628	0.6662	0.6628	0.6582	0.6806	0.8795	0.8835	0.8795	0.8864	0.8862
Lenna	2	14.553	14.553	14.553	14.553	14.553	0.5245	0.5245	0.5245	0.5245	0.5245	0.6723	0.6723	0.6723	0.6723	0.6723
	3	17.224	17.224	17.224	17.224	17.175	0.6251	0.6251	0.6251	0.6251	0.6227	0.7443	0.7443	0.7443	0.7443	0.7439
	4	19.025	18.645	18.873	18.950	18.821	0.6893	0.6769	0.6764	0.6833	0.6826	0.7934	0.8024	0.7975	0.7945	0.7549
	5	19.761	19.761	19.761	19.761	19.786	0.7041	0.7041	0.7041	0.7041	0.7579	0.8222	0.8222	0.8222	0.8222	0.8044
Livingroom	2	14.596	14.596	14.596	14.596	14.568	0.4831	0.4831	0.4831	0.4831	0.4843	0.7160	0.7160	0.7160	0.7160	0.7189
	3	17.146	17.290	16.238	17.198	17.253	0.6148	0.6148	0.5428	0.6141	0.6131	0.7993	0.7993	0.7746	0.8009	0.8035
	4	19.144	17.681	17.307	19.223	19.067	0.6694	0.6694	0.6191	0.6700	0.6674	0.8545	0.8545	0.8057	0.8565	0.8551
	5	20.750	19.968	19.618	21.037	20.818	0.7073	0.4831	0.6816	0.7225	0.7201	0.8933	0.7160	0.8661	0.9030	0.9007
Hunter	2	13.233	13.233	13.233	13.233	13.233	0.3529	0.3529	0.3529	0.3529	0.3529	0.7263	0.7263	0.7263	0.7263	0.7263
	3	16.214	17.893	16.214	16.197	16.122	0.4867	0.4847	0.4867	0.4837	0.4801	0.8186	0.8193	0.8186	0.8192	0.8168
	4	17.893	17.893	17.777	17.893	17.916	0.5699	0.5699	0.5652	0.5699	0.5719	0.8694	0.8694	0.8657	0.8694	0.8702
	5	19.437	18.302	18.485	19.263	19.316	0.6357	0.5832	0.5892	0.6318	0.6303	0.8999	0.8771	0.8837	0.8989	0.8994
pepper	2	16.443	16.443	16.443	16.443	16.443	0.5813	0.5813	0.5813	0.5813	0.5813	0.6855	0.6855	0.6855	0.6855	0.6855
	3	18.357	18.373	18.974	18.357	18.357	0.6589	0.6583	0.6589	0.6589	0.6583	0.7507	0.7502	0.7507	0.7507	0.7502
	4	19.390	19.727	19.905	9.4359	19.119	0.6720	0.6696	0.6965	0.6763	0.6826	0.7714	0.7706	0.7800	0.7742	0.7753
	5	21.340	21.342	21.340	21.349	21.281	0.7477	0.7475	0.7477	0.7472	0.7492	0.8143	0.8148	0.8143	0.8145	0.8153
Mandrill	2	16.178	16.178	16.178	16.178	16.178	0.6138	0.6138	0.6138	0.6138	0.6138	0.8336	0.8336	0.8336	0.8336	0.8336
	3	19.158	19.158	19.158	19.158	19.184	0.7241	0.7241	0.7241	0.7241	0.7220	0.8861	0.8861	0.8861	0.8861	0.8860
	4	21.121	21.041	21.084	21.066	21.070	0.7729	0.7720	0.7720	0.7743	0.7714	0.9174	0.9188	0.9142	0.9195	0.9182
	5	22.731	22.677	22.731	22.855	22.508	0.8230	0.8194	0.8230	0.8250	0.8153	0.9450	0.9457	0.9450	0.9460	0.9438
Average rank		**1,3333**	3,1666	2,1666	2,1666	3,9166	**1,6666**	2,4583	2,3333	2,0416	2,6666	2,2083	2,1666	2,5	**1,7083**	2,6190

Table 5 Comparison of best obtained objective and optimal threshold values based on Otsu

Test images	No. of thresholds	Average Fitness value					Average Optimal threshold values				
		HFPSO	AMO	PSO	GA	FA	HFPSO	AMO	PSO	GA	FA
Cameraman	2	3.6039e+03	3.6039e+03	3.6039e+03	3.6039e+03	3.6039e+03	70, 144	70, 144	70, 144	70, 144	70, 144
	3	3.6774e+03	3.6729e+03	3.6774e+03	3.6774e+03	3.6767e+03	56, 115, 153	64, 126, 161	56, 115, 153	56, 115, 153	56, 115, 153
	4	3.7325e+03	3.7293e+03	3.7325e+03	3.7324e+03	3.7558e+03	41, 94, 139, 169	41, 95, 141, 172	41, 94, 139, 169	43, 95, 139, 169	41, 92, 140, 170
	5	3.7631e+03	3.7542e+03	3.7631e+03	3.7625e+03	3.7300e+03	35, 82, 121, 148, 172	37, 87, 132, 159, 190	35, 82, 121, 148, 172	39, 85, 123, 149, 173	36, 83, 123, 150, 176
Lenna	2	1.9599e+03	15.282251	1.9599e+03	1.9599e+03	1.9599e+03	93, 151	93, 151	93, 151	93, 151	93, 151
	3	2.1268e+03	17.355030	2.1268e+03	2.1268e+03	2.1262e+03	80, 126, 171	80, 126, 171	80, 126, 171	80, 126, 171	81, 127, 171
	4	2.1904e+03	8.606074	2.1904e+03	2.1904e+03	2.1881e+03	75, 114, 146, 181	75, 114, 146, 181	75, 114, 146, 181	75, 114, 146, 181	74, 113, 145, 181
	5	2.2160e+03	19.311527	2.2160e+03	2.2158e+03	2.2118e+03	72, 107, 134, 159, 187	72, 107, 135, 160, 188	72, 107, 134, 159, 187	72, 105, 132, 157, 186	70, 102, 130, 157, 187
Livingroom	2	1.6273e+03	1.6273e+03	1.6273e+03	1.6273e+03	1.6273e+03	87, 145	87, 145	87, 145	87, 145	87, 145
	3	1.7594e+03	1.7594e+03	1.7594e+03	1.7594e+03	1.7581e+03	76, 123, 163	76, 123, 163	76, 123, 163	76, 123, 163	76, 123, 163
	4	1.8281e+03	1.8280e+03	1.8281e+03	1.8281e+03	1.8240e+03	56, 97, 132, 168	97, 132, 168	56, 97, 132, 168	56, 97, 132, 168	55, 96, 132, 169
	5	1.8712e+03	1.8707e+03	1.8712e+03	1.8708e+03	1.8652e+03	49, 88, 120, 146, 178	49, 89, 122, 148, 180	49, 88, 120, 146, 178	51, 90, 121, 147, 179	46, 87, 119, 147, 179
Hunter	2	1.9945e+03	1.9945e+03	1.9945e+03	1.9945e+03	1.9945e+03	86, 141	86, 141	86, 141	86, 141	86, 141
	3	2.1153e+03	2.1153e+03	2.1153e+03	2.1153e+03	2.1146e+03	71, 114, 155	71, 114, 155	71, 114, 155	71, 114, 155	71, 114, 155
	4	2.1697e+03	2.1694e+03	2.1697e+03	2.1697e+03	2.1677e+03	67, 106, 140, 172	68, 107, 141, 174	67, 106, 140, 172	67, 106, 140, 172	68, 106, 141, 173
	5	2.2006e+03	2.1972e+03	2.2006e+03	2.2006e+03	2.1955e+03	62, 93, 122, 150, 181	65, 100, 130, 156, 185	62, 93, 122, 150, 181	62, 93, 122, 150, 181	62, 96, 124, 151, 181
pepper	2	2.4695e+03	2.4695e+03	2.4695e+03	2.4695e+03	2.4694e+03	69, 136	69, 136	69, 136	69, 136	69, 136
	3	2.6367e+03	2.6367e+03	2.6367e+03	2.6367e+03	2.6361e+03	64, 119, 166	64, 119, 166	64, 119, 166	64, 119, 166	64, 119, 167
	4	2.6984e+03	2.6979e+03	2.6984e+03	2.6984e+03	2.6966e+03	47, 86, 126, 169	48, 88, 128, 170	47, 86, 126, 169	47, 86, 126, 169	49, 88, 127, 170
	5	2.7425e+03	2.7420e+03	2.7425e+03	2.7425e+03	2.7393e+03	43, 79, 113, 146, 177	43, 80, 115, 149, 179	43, 79, 113, 146, 177	43, 79, 113, 146, 177	43, 78, 114, 147, 178
Mandrill	2	1.3673e+03	1.3673e+03	1.3673e+03	1.3673e+03	1.3673e+03	98, 148	98, 148	98, 148	98, 148	98, 148
	3	1.4546e+03	1.4546e+03	1.4546e+03	1.4546e+03	1.4539e+03	86, 124, 159	86, 124, 159	86, 124, 159	86, 124, 159	86, 124, 158
	4	1.5039e+03	1.5039e+03	1.5039e+03	1.5039e+03	1.4995e+03	73, 106, 135, 165	73, 106, 135, 165	73, 106, 135, 165	73, 106, 135, 165	73, 106, 136, 166
	5	1.5286e+03	1.5269e+03	1.5286e+03	1.5285e+03	1.5234e+03	69, 100, 125, 148, 172	65, 95, 120, 144, 170	69, 100, 125, 148, 172	69, 99, 124, 147, 172	66, 96, 122, 147, 172
Average rank		**1,041,667**	3	**1,041,667**	1,458,333	3,833,333					

Table 6 Comparison of standard deviation based on Otsu

Test images	Number of thresholds	Standard deviation				
		HFPSO	AMO	PSO	GA	FA
Cameraman	2	0	0	0	0	0.019945
	3	0	8.441050	0	0	0.674793
	4	0	8.884541	0	0.185656	1.799805
	5	0	4.670287	0	0.710497	3.154229
Lenna	2	0	0	0	0	0.034656
	3	0	0.019827	0	0	0.435547
	4	0	0.113073	0	0.028327	1.735644
	5	0	0.828885	0.374665	0.483083	2.737694
Livingroom	2	0	0	0	0	0.044281
	3	0	0.001544	0	0	0.686807
	4	0	0.104463	0	0.012190	2.768423
	5	0	0.934447	0	0.468806	2.599497
Hunter	2	0	0	0	0	0.051701
	3	0	0.040290	0	0	0.552600
	4	0	0.335195	0	0	1.053928
	5	0	3.487077	0	0	2.208677
pepper	2	0	0	0	2.4695e+03	0.077890
	3	0	0	0	2.6367e+03	0.367532
	4	0	1.223259	0	2.6984e+03	1.191859
	5	0	0.701220	0	2.7425e+03	2.197863
Mandrill	2	0	0	0	0	0.035860
	3	0	0.055361	0	0	0.539065
	4	0	0.144469	0	0.004483	3.460228
	5	0	1.990693	0	0.093835	2.463172
Average rank		**1**	3,25	1,041,667	2,333,333	4,625

over the ranks of each algorithm, the highest averaged rank is bolded.

5.1 Kapur's function

It is clear from Table 2 that PSO produced very competitive results in comparison to HFPSO; however, HFPSO overall surpasses PSO in performance evaluation. From Tables 2, 3, 4, it is seen that the HFPSO ranks first in fitness value, SD, PSNR, and SSIM. Likewise, GA, AMO, HFPSO, PSO, and FA rank first to last respectively in terms of FSIM. Also, Table 3 reveals that in most of the cases the obtained SD values of the HFPSO are zero. This shows that HFPSO is a more stable algorithm in comparison to other algorithms in this study. On the other hand, FA ranks worst in fitness value, SD, and PSNR. Moreover, AMO ranks fourth in all criteria, whereas GA ranks second, alongside PSO, in fitness value. PSO ranks second in SD, and GA's rank is third. GA ranks second in PSNR, whereas PSO ranks third.

5.2 Otsu's function

Table 5 shows that HFPSO and PSO outperform the other algorithms in fitness value and that FA performs the worst. GA and AMO rank second and fourth, respectively. Furthermore, based on Table 6, HFPSO is the most stable algorithm because its SD is 0 in all levels of segmentation in all the images. PSO ranks second in SD value, whereas GA, AMO, and FA rank third to fifth, respectively. For PSNR, GA performs the best and HFPSO ranks second, where PSO. AMO and FA rank fourth and fifth, respectively. Moreover, HFPSO and PSO rank first according to SSIM. Likewise, HFPSO ranks First alongside PSO in terms of FSIM (Table 7).

The experimental results show that our proposed method of an HFPSO algorithm achieves better fitness value and PSNR. It also shows better stability in comparison to the other optimization algorithms for multilevel image segmentation.

Table 7 Comparison of PSNR based on Otsu

Test images	m	PSNR					SSIM					FSIM				
		HF-PSO	AMO	PSO	GA	FA	HF-PSO	AMO	PSO	GA	FA	HF-PSO	AMO	PSO	GA	FA
Cameraman	2	17.247	17.247	17.247	17.247	17.247	0.5964	0.5964	0.5964	0.5964	0.5964	0.8076	0.8076	0.8076	0.8076	0.8076
	3	20.322	19.794	20.322	20.322	20.287	0.6441	0.6034	0.6441	0.6441	0.6441	0.8483	0.8335	0.8483	0.8483	0.8483
	4	21.545	21.444	21.545	21.617	21.330	0.6504	0.6553	0.6504	0.6509	0.6501	0.8783	0.8806	0.8783	0.8783	0.8760
	5	23.376	22.244	23.376	23.425	23.203	0.6890	0.6690	0.6890	0.6867	0.6825	0.9146	0.8986	0.9146	0.9143	0.9140
Lenna	2	15.282	15.282	15.282	15.282	15.282	0.5465	0.5465	0.5465	0.5465	0.5465	0.6990	0.6990	0.6990	0.6990	0.6990
	3	17.355	17.355	17.355	17.355	17.314	0.6324	0.6324	0.6324	0.6324	0.6289	0.7549	0.7549	0.7549	0.7549	0.7537
	4	18.606	18.606	18.606	18.606	18.661	0.6775	0.6775	0.6775	0.6775	0.6790	0.8041	0.8041	0.8041	0.8041	0.8043
	5	19.317	19.311	19.317	19.366	19.540	0.7006	0.7015	0.7006	0.6946	0.6931	0.8320	0.8326	0.8320	0.8300	0.8294
Livingroom	2	15.999	15.999	15.999	15.999	15.999	0.5368	0.5368	0.5368	0.5368	0.5368	0.7573	0.7573	0.7573	0.7573	0.7573
	3	18.197	18.197	18.197	18.197	18.197	0.6163	0.6163	0.6163	0.6163	0.6163	0.8288	0.8288	0.8288	0.8288	0.8288
	4	20.673	20.673	20.673	20.673	20.660	0.7043	0.5689	0.7043	0.7043	0.7046	0.8832	0.7864	0.8832	0.8832	0.8834
	5	22.225	22.130	22.225	22.207	22.194	0.7561	0.7537	0.7561	0.7546	0.7577	0.9109	0.9121	0.9109	0.9112	0.9147
Hunter	2	15.591	15.591	15.591	15.591	15.591	0.4559	0.4559	0.4559	0.4559	0.4559	0.7866	0.7866	0.7866	0.7866	0.7866
	3	17.788	17.788	17.788	17.788	17.788	0.5622	0.5622	0.5622	0.5622	0.5622	0.8527	0.8527	0.8527	0.8527	0.8527
	4	18.765	18.695	18.765	18.765	18.708	0.6021	0.5991	0.6021	0.6021	0.5988	0.8845	0.8825	0.8845	0.8845	0.8830
	5	19.815	19.499	19.815	19.815	19.796	0.6512	0.6373	0.6512	0.6512	0.6502	0.9042	0.9053	0.9042	0.9042	0.9054
pepper	2	16.323	16.323	16.323	16.323	16.323	0.6096	0.6096	0.6096	0.6096	0.6096	0.6964	0.6964	0.6964	0.6964	0.6964
	3	18.469	18.469	18.469	18.469	18.442	0.6573	0.6573	0.6573	0.6573	0.6573	0.7499	0.7499	0.7499	0.7499	0.7503
	4	20.679	20.708	20.679	20.679	20.674	0.7168	0.7137	0.7168	0.7168	0.7128	0.7883	0.7876	0.7883	0.7883	0.7863
	5	22.393	22.372	22.393	22.393	22.351	0.7613	0.7576	0.7613	0.7613	0.7621	0.8251	0.8238	0.8251	0.8251	0.8249
Mandrill	2	15.433	15.433	15.433	15.433	15.433	0.5867	0.5867	0.5867	0.5867	0.5867	0.8082	0.8082	0.8082	0.8082	0.8082
	3	17.851	17.851	17.851	17.851	17.842	0.6786	0.6786	0.6786	0.6786	0.6790	0.8605	0.8605	0.8605	0.8605	0.8589
	4	20.415	20.415	20.415	20.415	20.410	0.7708	0.7708	0.7708	0.7708	0.7703	0.9010	0.9010	0.9010	0.9010	0.9024
	5	21.695	22.202	21.695	21.709	22.054	0.8097	0.8156	0.8097	0.8087	0.8131	0.9222	0.9249	0.9222	0.9217	0.9263
Average rank		1,375	2,29,166	1,375	**1,29,166**	2,95,238	**1,4166**	2,3333	**1,4166**	1,7083	2,2857	**1,5416**	2,2083	**1,5416**	1,75	2,1904

6 Conclusion

In this study, a multilevel image thresholding method has been proposed using a hybrid firefly and particle swarm optimization algorithm that maximizes Otsu's and Kapur's objective functions. The proposed method has been assessed by comparing it with four well-known optimization algorithms (AMO, PSO, GA, and FA) for six commonly used benchmark images. The visual experiments show that the thresholds are mostly located in valleys between two peaks.

The proposed HFPSO algorithm is extremely stable. The SD results are zero in Otsu's objective function, and it was ranked first according to Kapur's method. Moreover, HFPSO ranked first in fitness value for both Otsu's and Kapur's methods. In terms of the PSNR performance criterion, for Otsu's objective function, HFPSO came in second, behind GA and for Kapur's objective function, HFPSO ranks first. HFSPO performs well in terms of SSIM: it ranks first in Kapur's method and first alongside PSO in Otsu's method. On the other hand, for FSIM, HFPSO is tied with PSO for first under Otsu and is ranked third under Kapur. The comprehensive experiments reveal that this hybrid method achieves overall better results in term of fitness value and PSNR and that its SD is the best out of all the algorithms.

Acknowledgements The authors declare no conflict of interest in this study. We want to thank Dr. Lauren Davis for editing the article.

Funding We confirm that we do not have a funding source.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests.

References

1. Rahimzadeganasl, A., Alganci, U., Goksel, C.: An approach for the pan sharpening of very high resolution satellite images using a CIELab color based component substitution algorithm. Appl Sci **9**(23), 5234 (2019)
2. Ruiz-Ruiz, G., Gómez-Gil, J., Navas-Gracia, L.: Testing different color spaces based on hue for the environmentally adaptive segmentation algorithm (EASA). Comput Electr Agric **68**(1), 88–96 (2009)
3. Zhao, F., Chen, Y., Hou, Y., He, X.: Segmentation of blood vessels using rule-based and machine-learning-based methods: a review. Multimed Syst **25**(2), 109–118 (2019). https://doi.org/10.1007/s00530-017-0580-7
4. Jamal SB, Bilgin G (2019) Use of spatial information via markov and conditional random fields in histopathological images. In: 2019 42nd international conference on telecommunications and signal processing (TSP), 2019. IEEE, pp 71–75
5. Yamada, K., Mizuno, M.: A vehicle parking detection method using image segmentation. Electr Commun Jpn (Part III Fundam Electr Sci) **84**(10), 25–34 (2001)
6. Lienhart, R., Effelsberg, W.: Automatic text segmentation and text recognition for video indexing. Multimed Syst **8**(1), 69–81 (2000). https://doi.org/10.1007/s005300050006
7. Yan, C., Shao, B., Zhao, H., Ning, R., Zhang, Y., Xu, F.: 3D room layout estimation from a single RGB image. IEEE Trans Multimed 22(11):3014–3024. https://doi.org/10.1109/TMM.2020.2967645
8. Shapiro, L., Stockman, G.: Computer vision. Prentice-Hall, New Jersey (2001)
9. Farshi, T.R., Drake, J.H., Özcan, E.: A multimodal particle swarm optimization-based approach for image segmentation. Expert Syst Appl **149**, 113233 (2020)
10. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern **9**(1), 62–66 (1979)
11. Kapur, J.N., Sahoo, P.K., Wong, A.K.C.: A new method for gray-level picture thresholding using the entropy of the histogram. Comput Vis Graph Image Process **29**(3), 273–285 (1985). https://doi.org/10.1016/0734-189X(85)90125-2
12. Akay, B.: A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. Appl Soft Comput **13**(6), 3066–3091 (2013). https://doi.org/10.1016/j.asoc.2012.03.072
13. Roy S, Kumar U, Chakraborty D, Nag S, Mallick A, Dutta S (2015) Comparative analysis of cuckoo search optimization-based multilevel image thresholding. In, New Delhi, 2015. Intelligent computing, communication and devices. Springer, India, pp 327–342
14. Ouadfel, S., Taleb-Ahmed, A.: Social spiders optimization and flower pollination algorithm for multilevel image thresholding: a performance study. Expert Syst Appl **55**, 566–584 (2016). https://doi.org/10.1016/j.eswa.2016.02.024
15. Chao Y, Dai M, Chen K, Chen P, Zhang Z (2016) Fuzzy entropy based multilevel image thresholding using modified gravitational search algorithm. In: 2016 IEEE international conference on industrial technology (ICIT), 14–17 March 2016. pp 752–757. doi:https://doi.org/10.1109/ICIT.2016.7474845
16. Rahkar Farshi, T., Demirci, R., Feizi-Derakhshi, M.-R.: Image clustering with optimization algorithms and color space. Entropy **20**(4), 296 (2018)
17. Kahraman, A.S., Farshi, T.R., Demirci, R.: Renkli Görüntülerin Çok Seviyeli Eşiklenmesi ve Sınıflandırılması. Düzce Üniversitesi Bilim ve Teknoloji Dergisi **6**(4), 846–859 (2018)
18. Rahkar Farshi, T., Orujpour, M.: Multi-level image thresholding based on social spider algorithm for global optimization. Int J Inf Technol **11**(4), 713–718 (2019). https://doi.org/10.1007/s41870-019-00328-4
19. Rahkar Farshi, T.: A multilevel image thresholding using the animal migration optimization algorithm. Iran J Comput Sci **2**(1), 9–22 (2019). https://doi.org/10.1007/s42044-018-0022-5
20. Sathya, P.D., Kayalvizhi, R.: Optimal multilevel thresholding using bacterial foraging algorithm. Expert Syst Appl **38**(12), 15549–15564 (2011a). https://doi.org/10.1016/j.eswa.2011.06.004
21. Sathya, P.D., Kayalvizhi, R.: Amended bacterial foraging algorithm for multilevel thresholding of magnetic resonance brain images. Measurement **44**(10), 1828–1848 (2011b). https://doi.org/10.1016/j.measurement.2011.09.005
22. Sathya, P.D., Kayalvizhi, R.: Modified bacterial foraging algorithm based multilevel thresholding for image segmentation. Eng Appl Artif Intell **24**(4), 595–615 (2011c). https://doi.org/10.1016/j.engappai.2010.12.001
23. Yin, P.-Y.: Multilevel minimum cross entropy threshold selection based on particle swarm optimization. Appl Math Comput **184**(2), 503–513 (2007). https://doi.org/10.1016/j.amc.2006.06.057
24. Horng, M.-H.: Multilevel thresholding selection based on the artificial bee colony algorithm for image segmentation. Expert Syst Appl **38**(11), 13785–13791 (2011). https://doi.org/10.1016/j.eswa.2011.04.180

25. Oliva D, Cuevas E, Pajares G, Zaldivar D, Perez-Cisneros M (2013) Multilevel thresholding segmentation based on harmony search optimization. J Appl Math
26. Ayala, H.V.H., Santos, FMd., Mariani, V.C., Coelho, Ld.S.: Image thresholding segmentation based on a novel beta differential evolution approach. Expert Syst Appl **42**(4), 2136–2142 (2015). https://doi.org/10.1016/j.eswa.2014.09.043
27. Muppidi M, Rad P, Agaian SS, Jamshidi M (2015) Image segmentation by multi-level thresholding using genetic algorithm with fuzzy entropy cost functions. In: 2015 International conference on image processing theory, tools and applications (IPTA), 10–13 Nov. 2015. pp 143–148. doi:https://doi.org/10.1109/IPTA.2015.7367114
28. Pal SS, Kumar S, Kashyap M, Choudhary Y, Bhattacharya M (2016) Multi-level Thresholding Segmentation Approach Based on Spider Monkey Optimization Algorithm. In: Satapathy SC, Raju KS, Mandal JK, Bhateja V (Eds.) Proceedings of the second international conference on computer and communication technologies, New Delhi. Springer India, pp 273–287
29. Khairuzzaman, A.K.M., Chaudhury, S.: Multilevel thresholding using grey wolf optimizer for image segmentation. Expert Syst Appl **86**, 64–76 (2017). https://doi.org/10.1016/j.eswa.2017.04.029
30. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans Evol Comput **1**(1), 67–82 (1997). https://doi.org/10.1109/4235.585893
31. Aydilek, İB.: A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. Appl Soft Comput **66**, 232–249 (2018). https://doi.org/10.1016/j.asoc.2018.02.025
32. Blum, C., Roli, A., Sampels, M.: Hybrid metaheuristics: an emerging approach to optimization, vol. 114. Springer, Berlin (2008)
33. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, pp. 169–178. Springer, Berlin (2009)
34. Branham, M.: How and Why do Fireflies Light Up? Scientific America, http://www.scientificamerican.com/article.cfm?id=howand-why-do-fireflies (2005). Accessed 11 Nov 2012
35. Wang, F., Zhang, H., Li, K., Lin, Z., Yang, J., Shen, X.-L.: A hybrid particle swarm optimization algorithm using adaptive learning strategy. Inf Sci **436–437**, 162–177 (2018). https://doi.org/10.1016/j.ins.2018.01.027
36. Wenhua H, Ping Y, Haixia R, Jianpeng S (2010) Comparison study of several kinds of inertia weights for PSO. In: 2010 IEEE international conference on progress in informatics and computing, 10–12 Dec. pp 280–284. doi:https://doi.org/10.1109/PIC.2010.5687447
37. Rahkar Farshi, T.: Battle royale optimization algorithm. Neural Comput. Appl. 1–19 (2020)