Multimedia Tools and Applications (2020) 79:34993–35016
https://doi.org/10.1007/s11042-020-08931-5

# An improved cuckoo search algorithm for multi-level gray-scale image thresholding

**Min Sun**[1] · **Hui Wei**[1]

Received: 5 April 2019 / Revised: 24 March 2020 / Accepted: 13 April 2020 /
Published online: 26 April 2020


**Abstract**
In decades, Yang's cuckoo search algorithm has been widely developed to select the optimal threshold of bi-level image thresholding, but the amount of computation of which increases exponentially with multi-level thresholding. To reduce the computation quantity, the iterative step size is adaptively decided by its fitness values of the current iteration without using the *Lévy* distribution in this study. The modification may cause the solution drops into the local optima during the later period. Therefore, the constant discovery probability $p_a$ is automatically changed relating to the current and total iterations. And then, to verify segmentation accuracy and efficiency of the proposed method, an adaptive cuckoo search algorithm proposed by Naik and Yang's cuckoo search algorithm are included to test on several gray-scale images. The results show that the proposed algorithm is expert in selecting optimal thresholds for segmenting gray-scale image.

**Keywords** Cuckoo search algorithm · Multi-level thresholding · Adaptive · Gray-scale image

## 1 Introduction

Image segmentation is to separate an image into some distinct regions with different characteristics and extract the objective of interests from their background. In these years, lots of techniques have been proposed and applied in image segmentation [6, 15, 17, 19, 33]. Among all the existing techniques, the thresholding is thought as a prevalent technique whereas its perspicuity and precision [18]. The key of thresholding is to quickly find an optimal threshold with certain criteria to achieve segmentation accurately.
However, for multi-level thresholding, owing to the exhaustive search, the traditional thresholding methods can't select the optimal thresholds efficiently and the amount of calculation increases. To solve such limitations, researchers formulated the existing criteria of conventional thresholding methods as objective functions, and incorporated meta-heuristic

✉ Hui Wei
2196892863@qq.com

[1] School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, Anhui, 232001, China

algorithms to enhance the computational speed, such as GA [8], PSO [7], cuckoo search algorithm (CS) [28], etc., Hammouche [8] used GA to find multi-level thresholds. Zhang [32] developed artificial bee colony to optimize the Tsallis entropy. Ghamisi [7] proposed a fractional-order DPSO for determining thresholds. Wei [28] presented Yang's CS algorithm for solving multi-level Otsu's problem. Sharma [20] designed a firefly algorithm based on the *Lévy* flight to maximize Kapur's entropy. Among these meta-heuristic algorithms, the Yang's CS algorithm has drawn the attention of researchers.

CS algorithm [31] is a technique for optimization problems proposed by Yang in 2009. Many researchers have proved the efficiency of Yang's CS algorithm in different applications, such as face recognition [22], engineering design [29] and neural network training [24]. Although CS algorithm is simple and has few parameters as well as high efficiency, it's easy to fall into local optima during the later period which causes premature problem and time-consuming. Therefore, to enhance the performance of CS algorithm, many scholars have proposed improved CS algorithm [10, 13, 21, 25, 26], such as an adaptive cuckoo search algorithm (ACS) was proposed by Naik [12], the adaptive strategy is used to adjust the step size and eventually leads to faster convergence. But the adaptive strategy isn't applied to the discovery probability $p_a$, this may influence the algorithm accuracy.

Therefore, based on ACS algorithm, an improved ACS algorithm (IACS) is introduced to for segmenting. For the proposed method, the initial step size randomly yields without being designed a prior. Furthermore, to avoid local extremum point and enhance the variety of cuckoos, the $p_a$ value changes nonlinearly with iterations. To explain the efficiency of the IACS algorithm, two methods, Yang's [31] and Naik's [12] are involved to searching multi-level thresholds.

The next parts are structured as: Section 2 briefly leads to three types objective functions usually applied for the multi-level image thresholding, namely, Otsu's method, Kapur's entropy and Tsallis entropy. Section 3 provides the methodologies of Yang's CS and ACS algorithms and presents the improved method(IACS). And then in Section 4, some experiments are expressed to verify the effect of the proposed method and analyze the results. Finally, Section 5 offers some conclusions.

## 2 Multi-level thresholding

Recently, multi-level thresholding methods have been widely applied to separate multiple objectives from background for an image [2–5, 16, 21]. In this section, the key thought for multi-level thresholding is briefly introduced. Take a image $I$ with $L$ distinct gray levels into account, where $L$= 256, then the multi-level image thresholding is defined as:

$$\begin{aligned} C_0 &= \{m(x,y) \in I \mid 0 \le m(x,y) \le t_1 - 1\} \\ C_1 &= \{m(x,y) \in I \mid t_1 \le m(x,y) \le t_2 - 1\} \\ &\vdots \\ C_n &= \{m(x,y) \in I \mid t_n \le m(x,y) \le L - 1\} \end{aligned} \tag{1}$$

where $m(x, y)$ denotes a gray level value of pixel, $t_1, t_2, \ldots, t_n$ are the different thresholds.

### 2.1 Otsu's method

The Otsu's criterion is to maximize the between-class variance [14], assume that the probability of pixels at level $i$ is $p_i$ ($p_i \geq 0$ ), the cumulative probability of each class $\omega_i$ can be calculated as:

$$\omega_0 = \sum_{i=0}^{t_1-1} p_i \ , \ \omega_1 = \sum_{i=t_1}^{t_2-1} p_i \ , \cdots , \ \omega_n = \sum_{i=t_n}^{L-1} p_i \tag{2}$$

Then the optimal thresholds are obtained as follows:

$$\left(t_1^*, t_2^*, ..., t_n^*\right) = \text{argmax} \left\{\sigma_0^2 + \sigma_1^2 + \cdots + \sigma_n^2\right\} \tag{3}$$

where

$$\begin{aligned} \sigma_0^2 &= \omega_0(\mu_0 - \mu_T)^2, \ \mu_0 = \frac{\sum_{i=0}^{t_1-1} i\, p_i}{\omega_0} \\ \sigma_1^2 &= \omega_1(\mu_1 - \mu_T)^2, \ \mu_1 = \frac{\sum_{i=t_1}^{t_2-1} i\, p_i}{\omega_1} \\ &\vdots \\ \sigma_n^2 &= \omega_n(\mu_n - \mu_T)^2, \ \mu_n = \frac{\sum_{i=t_n}^{L-1} i\, p_i}{\omega_n} \end{aligned} \tag{4}$$

where $\mu_T = \sum_{i=0}^{L-1} i\, p_i$ is the mean value of input image, $\mu_i$ ($i = 0, 1, \ldots, n$) is the mean value of the pixels of the corresponding region.

### 2.2 Kapur's entropy method

Kapur's entropy was proposed to maximize the sum of the entropy for segmenting images [9]. The entropies $H_i$ are defined as:

$$\begin{aligned} H_0 &= -\sum_{i=0}^{t_1-1} \left(\frac{p_i}{\omega_0}\right) \ln\left(\frac{p_i}{\omega_0}\right), \quad \omega_0 = \sum_{i=0}^{t_1-1} p_i \\ H_1 &= -\sum_{i=t_1}^{t_2-1} \left(\frac{p_i}{\omega_1}\right) \ln\left(\frac{p_i}{\omega_1}\right), \quad \omega_1 = \sum_{i=t_1}^{t_2-1} p_i \\ &\vdots \\ H_n &= -\sum_{i=t_n}^{L-1} \left(\frac{p_i}{\omega_n}\right) \ln\left(\frac{p_i}{\omega_n}\right), \quad \omega_n = \sum_{i=t_n}^{L-1} p_i \end{aligned} \tag{5}$$

The optimal thresholds are gained as:

$$\left(t_1^*, t_2^*, \ldots, t_n^*\right) = \text{argmax} \left\{\sum_{i=0}^{n} H_i\right\} \tag{6}$$

### 2.3 Tsallis entropy method

Tsallis entropy is come from Shannon entropy [23], it's defined as:

$$S_q = \frac{1 - \sum_{i=1}^{k} p_i^q}{q - 1} \tag{7}$$

where $q$ is Tsallis parameter. The relationship between the entropy of each subsystem is as follows:

$$S_q(O + B) = S_q(O) + S_q(B) + (1 - q) . S_q(O) . S_q(B) \tag{8}$$

For multi-level image thresholding problem [2], the optimal thresholds are obtained using (9):

$$\begin{aligned} \left(t_1^*, t_2^*, ..., t_n^*\right) = \text{argmax} \Big\{ & S_q^{C_0}(t) + S_q^{C_1}(t) + \cdots + S_q^{C_n}(t) \\ & + (1 - q) . S_q^{C_0}(t) . S_q^{C_1}(t) \cdots S_q^{C_n}(t) \Big\} \end{aligned} \tag{9}$$

where

$$\begin{aligned} S_q^{C_0}(t) &= \frac{1 - \sum_{i=0}^{t_1 - 1} \left(p_i / p^{C_0}\right)^q}{q - 1}, \quad p^{C_0} = \sum_{i=0}^{t_1 - 1} p_i \\ S_q^{C_1}(t) &= \frac{1 - \sum_{i=t_1}^{t_2 - 1} \left(p_i / p^{C_1}\right)^q}{q - 1}, \quad p^{C_1} = \sum_{i=t_1}^{t_2 - 1} p_i \\ &\vdots \\ S_q^{C_n}(t) &= \frac{1 - \sum_{i=t_n}^{L - 1} \left(p_i / p^{C_n}\right)^q}{q - 1}, \quad p^{C_n} = \sum_{i=t_n}^{L - 1} p_i \end{aligned} \tag{10}$$

subject to the following constraints:

$$\begin{aligned} \left|p^{C_0} + p^{C_1}\right| - 1 < S^{C_0} < 1 - \left|p^{C_0} + p^{C_1}\right| \\ \left|p^{C_1} + p^{C_2}\right| - 1 < S^{C_1} < 1 - \left|p^{C_1} + p^{C_2}\right| \\ \vdots \\ \left|p^{C_{(n-1)}} + p^{C_n}\right| - 1 < S^{C_{(n-1)}} < 1 - \left|p^{C_{(n-1)}} + p^{C_n}\right| \end{aligned} \tag{11}$$

In (11), $p^{C_0}, p^{C_1}, ..., p^{C_n}$ corresponding to $S^{C_0}, S^{C_1}, \ldots, S^{C_{n-1}}$ are computed with $t_1^*, t_2^*, ..., t_n^*$, respectively.

## 3 Methodology

### 3.1 Yang's CS algorithm

Yang's CS algorithm is inspired by the obligate brood parasitism of some cuckoos, for simulating the whole process, Yang uses three assumptions [30]:

(1) one egg is hatched and put in a nest randomly by each cuckoo;

(2) the excellent nests will be preserved for next iteration;
(3) the possibility of host bird finding cuckoo's egg is represented by $p_a$.

Then a solution $x_i^{(t+1)}$ is produced by *Lévy* flight:

$$x_i^{(t+1)}=x_i^{(t)}+\alpha\oplus L\acute{e}vy\,(\beta)\,,\quad i=1,2,\ldots,N \tag{12}$$

where $\alpha$ represents the step size and $\oplus$ is entry-wise multiplications.
The other part of CS algorithm is biased random walk [25]:

$$x_i^{(t+1)}=\begin{cases} x_i^{(t)}+r\left(x_p^{(t)}-x_g^{(t)}\right), & if\ r_a>p_a \\ x_i^{(t)}, & otherwise \end{cases} \tag{13}$$

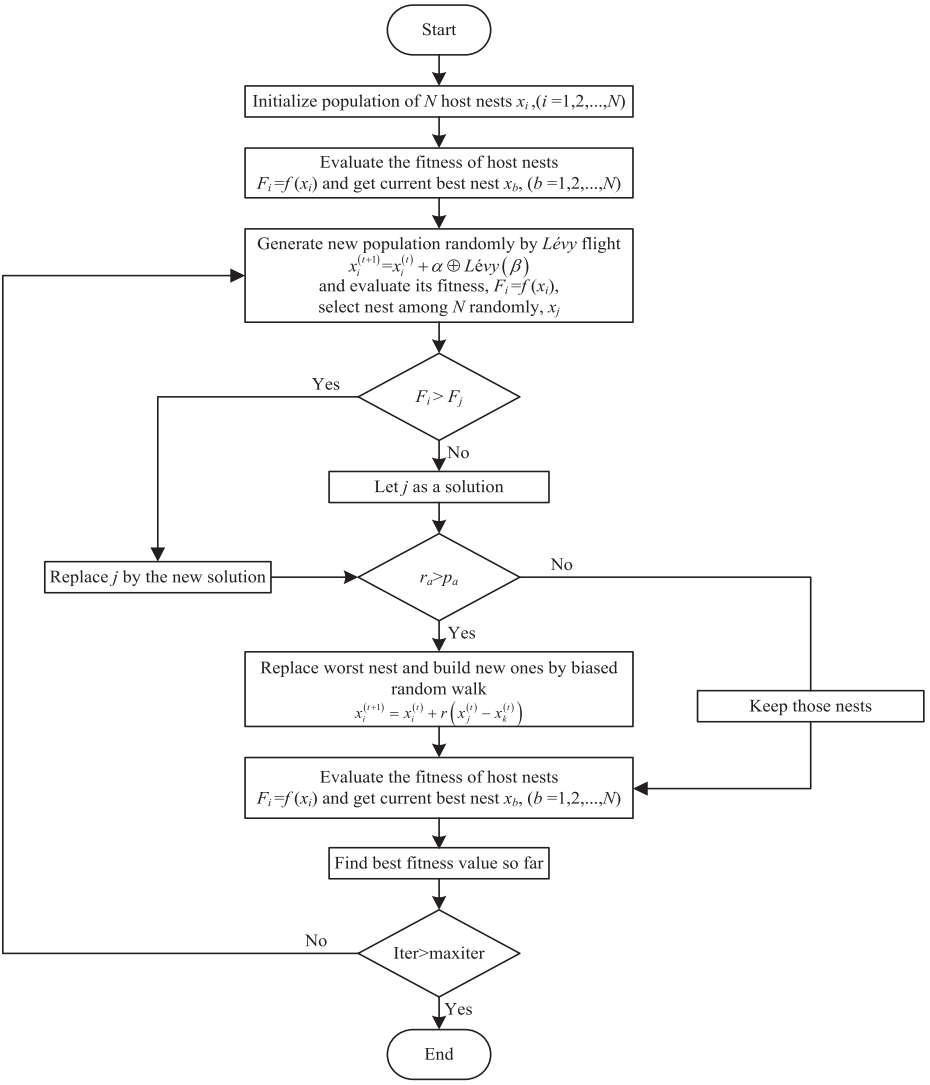



**Fig. 1** Flowchart of Yang's CS algorithm

where $p$ , $g$ is the $p$th and $g$th random solutions in the population, respectively, $r \in [0, 1]$ and $r_a \in [0, 1]$. The flowchart of Yang's CS is given by Fig. 1.

### 3.2 Adaptive CS algorithm

The Yang's CS algorithm uses the *Lévy* step to search the optimal solution, which generally satisfies the *Lévy* distribution [11], but the *Lévy* step in the iteration process of Yang's CS algorithm is not controlled by any measures, the process of optimization may be time-consuming. To promote the efficiency of the Yang's CS, Naik [12] modified the *Lévy* step adaptively based on the individual fitness value and the current iterative number

$$stepsize_i^{(t+1)} = \left(\frac{1}{t}\right)^{\left|\frac{f_{best}^t - f_i^t}{f_{best}^t - f_{worst}^t}\right|} \tag{14}$$

where $t$ represents current iterations, $f_i^t$ denotes the fitness of $i$th nest in $t^{th}$ iteration, $f_{best}^t$ , $f_{worst}^t$ is the best, worst fitness in $t^{th}$ iteration, respectively.
From the (14), the step size is long at the beginning, but as the iterations increase, it decreases. Hence, the step size will be very short when the algorithm achieves the global best solution. Equation (14) also reveals that the step size changes with fitness adaptively. Then (12) is rewritten as:

$$x_i^{(t+1)} = x_i^{(t)} + randn \times stepsize_i^{(t+1)} \tag{15}$$

The superiority of ACS algorithm is that, it does not set any initial parameter, Therefore, the modified method is less parameter and faster than Yang's CS algorithm.

### 3.3 Improved ACS algorithm

The discovery probability $p_a$ introduced in the Yang's CS represents the possibility whether the nest will be discarded or be renewed [10]. It's used to control the coordination of global and local search. If $p_a$ value is small, a large number of dimensions of a solution are changed in each generation, conversely, the poor individual will remain for the next generation, it may be unable to obtain the best solutions [25]. The parameter $p_a$ is fixed to 0.25 in the ACS algorithm [12], but for different problems, they need different values. In this work, the value of $p_a$ is changed with the number of iterations adaptively and showed as follows:

$$\begin{aligned} p_a &= p_{a\_\max} - (p_{a\_\max} - p_{a\_\min}) \cdot \exp(\eta \cdot t) \\ \eta &= \frac{1}{T} ln\left(\frac{p_{a\_\min}}{p_{a\_\max}}\right) \end{aligned} \tag{16}$$

where $T$ is total iterations, $p_{a\_\max}$ and $p_{a\_\min}$ are the predefined maximum, minimum discovery probability.
It can be observed from the (16), in the early iterations, the $p_a$ value is relatively small, it will increase the diversity of solutions. In the final iterations, the value of $p_a$ will generate faster convergence to the optimum solutions. In short, $p_a$ is nonlinearly altered from a small value to a large value relatively in the whole process, it will improve the accuracy of the algorithm as a whole.

Further, (15) is reformulated as [13]:

$$x_i^{(t+1)} = x_i^{(t)} + randn \times stepsize_i^{(t+1)} \times \left(x_i^{(t)} - x_{gbest}\right) \tag{17}$$

(a) ACS

**begin**
*Generate initial population of N host nests* $x_i$ ;
**Set the discovery probability** $p_a$.
**For** all $x_i$ **do**
*Evaluate the fitness* $F_i=f(x_i)$.
**end for**
**While**(*t <Maxiteration)or (stop criterion)*
*Find the bestfitness and worstfitness of the current generation among the host nests.*
***Calculate the step size by using***
$stepsize_i^{(t+1)} = \left(\frac{1}{t}\right)^{\left|\frac{f_{best}-f_i}{f_{best}-f_{worst}}\right|}$
***and the new position of cuckoo nests using***
$x_i^{(t+1)} = x_i^{(t)} + randn \times stepsize_i^{(t+1)}$
*evaluate its fitness* $F_i=f(x_i)$
*Choose a nest among N* ($x_j$) *randomly.*
**If** ($F_i>F_j$)
*For maximization problem*
*Replace* $x_j$ *with* $x_i$
*Replace* $F_j$ *with* $F_i$
**end**
*A fraction* ($p_a$) *of worse nests are abandoned and new ones are built;*
*Evaluate the fitness of new nests;*
*Keep the best solutions;*
*Sort the solutions and find the current best.*
**end while**
**end**

(b) IACS

**begin**
*Generate initial population of N host nests* $x_i$ ;
***Define minimum discovery probability value*** $p_{a\_min}$
***Define maximum discovery probability value*** $p_{a\_max}$.
**For** all $x_i$ **do**
*Evaluate the fitness* $F_i=f(x_i)$.
**end for**
**While**(*t <Maxiteration) or (stop criterion)*
*Find the bestfitness and worstfitness of the current generation among the host nests.*
***Calculate the step size by using***
$stepsize_j^{(t+1)} = \left(\frac{1}{t}\right)^{\left|\frac{f_{best}-f_i}{f_{best}-f_{worst}}\right|}$
***and the new position of cuckoo nests by using***
$x_i^{(t+1)} = x_j^{(t)} + randn \times stepsize_j^{(t+1)} \times \left(x_i^{(t)} - x_{gbest}\right)$
*evaluate its fitness* $F_i=f(x_i)$
*Choose a nest among N* ($x_j$) *randomly.*
**If** ($F_i>F_j$),
*For maximization problem*
*Replace* $x_j$ *with* $x_i$
*Replace* $F_j$ *with* $F_i$.
**end**
*A fraction* ($p_a$) *of worse nests are abandoned and new ones are built;*
*Evaluate the fitness of new nests;*
*Keep the best solutions;*
*Sort the solutions and find the current best.*
**end while**
**end**

**Fig. 2** Pseudo-code of **a** ACS and **b** IACS

where $x_{gbest}$ is global best solution. In the early search phase, the optimization efficiency of CS and ACS algorithms is faster, but in later period, the two algorithms are easy to sink into local extremum, which leads to low segmentation accuracy. The IACS algorithm can adaptively adjust $p_a$ value, hence it jumps out of the local extremum point in time, improves the nest's fitness continuously. And then, the optimal threshold is obtained to extract the required target accurately. The basic steps of IACS and ACS algorithms are shown in Fig. 2 and the flowchart of IACS is given by Fig. 3.

## 4 Experiments and discussions

Table 1 is parametric settings of optimization algorithms. The number of nests and maximum iterations of all the three algorithms are fixed to 50 and 150, respectively. They are identical for all experiments due to having a great influence on convergence rate and computational time of the algorithm. The discovery probability $p_a$ decides whether to give up the nest,when $p_a = 1$, host bird throws out the cuckoo's egg or simply abandons its nest, when $p_a = 0$, host bird can't recognize the cuckoo's egg and may hatch it. Hence, the range of $p_a$ is set between 0 and 1. For IACS algorithm, the maximum $p_a$ is 0.95 and the minimum $p_a$ is 0.15 [26].

### 4.1 Image dataset

To explore the effectiveness of the three optimization algorithms for segmentation, gray-scale test images namely Lena, Baboon, and Cameraman are taken from Image Databases and given in Fig. 4.

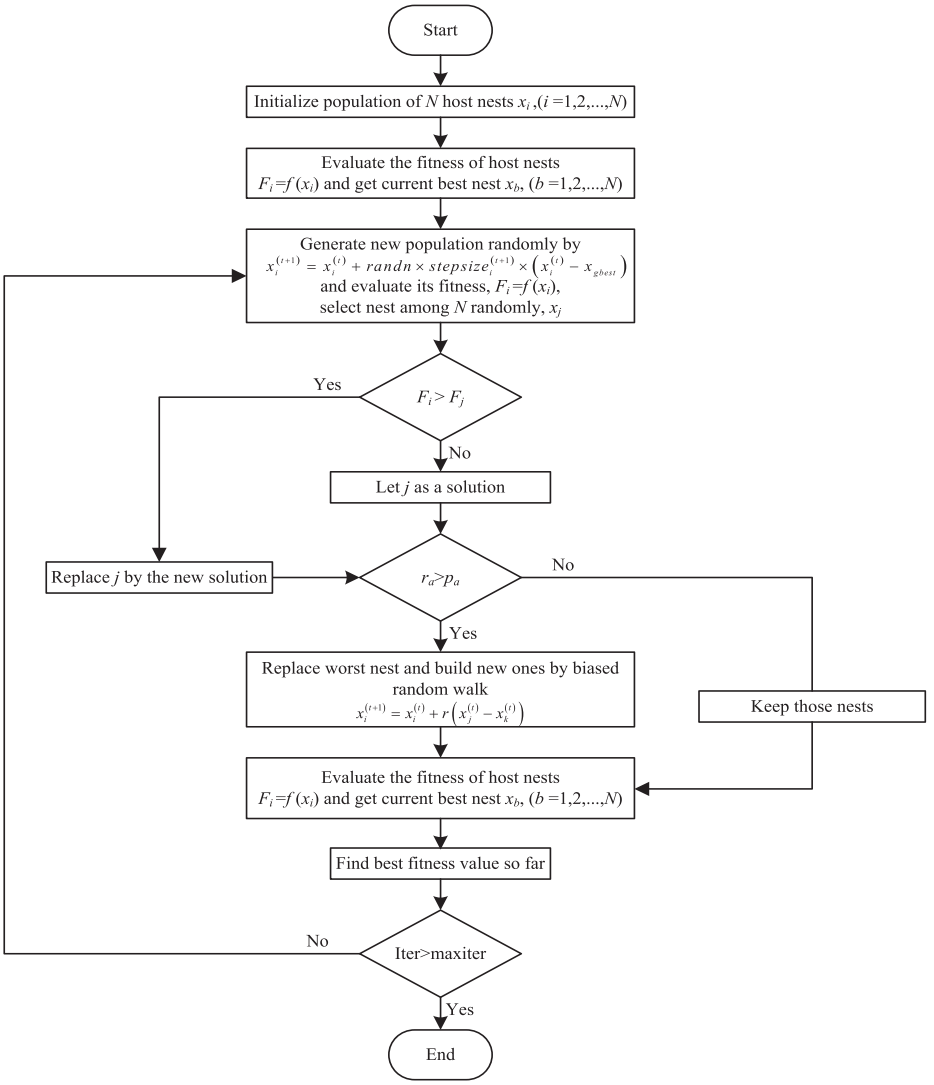
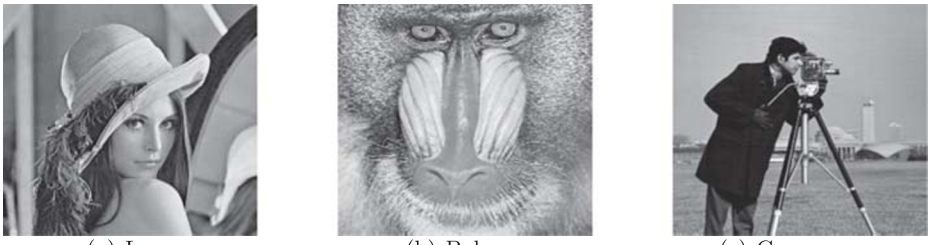



**Fig. 3** Flowchart of IACS algorithm

**Table 1** Parameters values used in CS, ACS and IACS algorithms

| Parameters | CS | ACS | IACS |
|---|---|---|---|
| No. of nests($N$) | 50 | 50 | 50 |
| Discovery probability($p_a$) | 0.25 | 0.25 | $p_{a\_\max} = 0.95$<br>$p_{a\_\min} = 0.15$ |
| No. of total iterations($T$) | 150 | 150 | 150 |
| Lower limit for egg | 1 | 1 | 1 |
| Upper limit for egg | 256 | 256 | 256 |



(a) Lena (b) Baboon (c) Cameraman

**Fig. 4** Original test images

The size of Baboon and Cameraman are 512 × 512, Lena has size 256 × 256. As it can be seen from their respective histograms in Fig. 5, they are multimodal in nature, all the images own distinct histogram features with irregular distributions depicting objective and background. Hence, to identify the target region exactly, multi-level thresholding is a more efficient technique.

### 4.2 Performance metrics

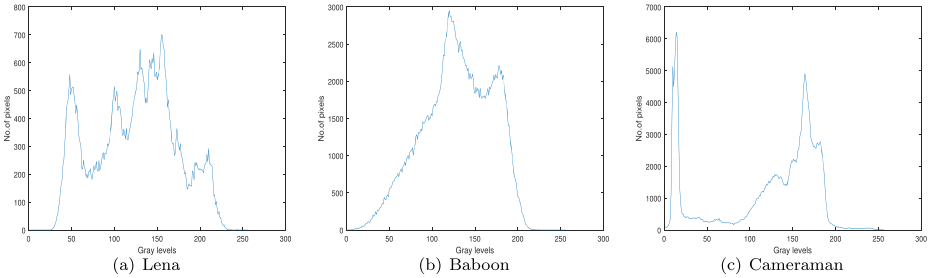
The accuracy of the segmentation is assessed by Peak Signal to Noise Ratio(PSNR) [1]. The larger the PSNR, the better quality of thresholding, the mathematical expression is given below:

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{18}$$

where

$$MSE = \frac{1}{EF}\sum_{j=1}^{E}\sum_{k=1}^{F}[I(j,k) - I'(j,k)]^2 \tag{19}$$

where $E \times F$ denotes the image size, $I$ and $I'$ are original, segmented images, respectively. Mean Square Error(MSE) is computed between the original image and its segmented image for different thresholds and for each algorithm. Lower MSE value indicates lower segmentation errors.




(a) Lena (b) Baboon (c) Cameraman

**Fig. 5** Histograms of those images

**Table 2** Comparison of mean PSNR, MSE, CPU time and SSIM values between different algorithms using Otsu's method

| Images | $n$ | CS | ACS | IACS | | | |
|---|---|---|---|---|---|---|---|
| (a) PSNR value comparison | | | | | (b) PSNR value comparison | | |
| Lena | 5 | 19.2517 | 19.2517 | 19.2517 | 772.5149 | 772.5149 | 772.5149 |
| | 7 | 21.0423 | 21.0828 | 21.1350 | 511.5217 | 506.7987 | 500.7290 |
| | 9 | 22.8682 | 23.2432 | 23.5056 | 336.2906 | 308.8179 | 290.1680 |
| | 11 | 24.6340 | 24.9702 | 25.5256 | 224.0162 | 207.6243 | 182.3058 |
| Baboon | 5 | 21.7105 | 21.7105 | 21.7105 | 438.5573 | 438.5573 | 438.5573 |
| | 7 | 24.5214 | 24.7249 | 24.8280 | 229.6240 | 219.1696 | 213.9425 |
| | 9 | 26.9813 | 27.1650 | 27.2705 | 130.3424 | 125.0180 | 121.9749 |
| | 11 | 28.5088 | 28.8025 | 29.1335 | 91.9003 | 85.8146 | 79.4963 |
| Cameraman | 5 | 23.3822 | 23.3822 | 23.3822 | 195.2716 | 192.8844 | 192.1902 |
| | 7 | 25.2165 | 25.2779 | 25.2929 195.2716 | 192.8844 | 192.1902 | |
| | 9 | 27.2537 | 27.2656 | 27.3021 | 122.4333 | 122.0750 | 121.0388 |
| | 11 | 28.4528 | 28.4971 | 28.5555 | 92.8670 | 91.9143 | 90.6848 |
| (c) CPU time comparison (s) | | | | | (d) SSIM comparison | | |
| Lena | 5 | 5.9652 | 5.2498 | 4.7422 | 0.7059 | 0.7059 | 0.7059 |
| | 7 | 5.9965 | 5.3803 | 4.9724 | 0.7724 | 0.7734 | 0.7743 |
| | 9 | 6.2880 | 5.5402 | 5.1602 | 0.8153 | 0.8207 | 0.8251 |
| | 11 | 6.3601 | 5.7902 | 5.3669 | 0.8541 | 0.8576 | 0.8644 |
| Baboon | 5 | 5.7913 | 5.1817 | 4.6616 | 0.8323 | 0.8323 | 0.8323 |
| | 7 | 5.9226 | 5.3514 | 4.8850 | 0.8927 | 0.8945 | 0.8960 |
| | 9 | 6.1382 | 5.5554 | 5.1057 | 0.9287 | 0.9295 | 0.9313 |
| | 11 | 6.2692 | 5.6497 | 5.2786 | 0.9469 | 0.9495 | 0.9521 |
| Cameraman | 5 | 5.8312 | 5.2474 | 4.6621 | 0.7140 | 0.7140 | 0.7140 |
| | 7 | 6.0631 | 5.3534 | 4.8951 | 0.7465 | 0.7473 | 0.7478 |
| | 9 | 6.2251 | 5.5687 | 5.1033 | 0.7771 | 0.7782 | 0.7797 |
| | 11 | 6.3492 | 5.7280 | 5.2368 | 0.7897 | 0.7913 | 0.7925 |

Structural similarity index(SSIM) [27] is to measure the structure of segmented and original image. Higher SSIM value shows that the segmented image contains more information, it's defined as:

$$SSIM\left(I, I'\right) = \frac{\left(2\mu_I\mu_{I'} + D_1\right)\left(2\sigma_{II'} + D_2\right)}{\left(\mu_I^2 + \mu_{I'}^2 + D_1\right)\left(\sigma_I^2 + \sigma_{I'}^2 + D_2\right)} \tag{20}$$

where $\mu_I$ and $\mu_{I'}$ are the average values and $\sigma_{II'}$ is the covariance, $\sigma_I^2$ and $\sigma_{I'}^2$ are the variance, $D_1 = D_2 = 0.065$.

### 4.3 Experimental results evaluation

All the images were independently executed 20 times using each of presented multi-level segmentation methods. Tables 2, 3, 4, 5, 6 and 7 validate the contrast of different quality metric values received by Yang's CS, ACS and the proposed IACS algorithm, where $n$ represents the thresholds number. Best image segmentation results and the rate of convergence

**Table 3** Comparison of optimal thresholds and corresponding fitness values between CS, ACS and IACS using Otsu's method

| Images | *n* | Mean optimal thresholds | | | Mean fitness values | | |
|---|---|---|---|---|---|---|---|
| | | CS | ACS | IACS | CS | ACS | IACS |
| Lena | 5 | 73,109,136,160,188 | 73,109,136,160,188 | 73,109,136,160,188 | 2.2161e+03 | 2.2161e+03 | 2.2161e+03 |
| | 7 | 63,89,112,132,149,168,193 | 63,88,112,132,149,168,192 | 62,88,119,132,149,168,192 | 2.2475e+03 | 2.2475e+03 | 2.2475e+03 |
| | 9 | 55,74,94,113,132,148, 164,183,202 | 53,73,93,115,132,148, 163,183,202 | 53,73,94,114,132,148, 164,183,202 | 2.2610e+03 | 2.2610e+03 | 2.2610e+03 |
| | 11 | 50,65,83,98,112,126,138,151, 166,184,202 | 49,65,81,96,111,125,137,151, 166,184,202 | 50,64,81,96,111,124,137,151, 166,184,202 | 2.2692e+03 | 2.2692e+03 | 2.2693e+03 |
| Baboon | 5 | 71,101,125,147,171 | 71,101,125,147,171 | 71,101,125,147,171 | 1.4062e+03 | 1.4062e+03 | 1.4062e+03 |
| | 7 | 60,85,106,124,140,158,177 | 60,85,106,123,140,159,177 | 60,84,105,123,140,158,177 | 1.4317e+03 | 1.4317e+03 | 1.4317e+03 |
| | 9 | 52,73,91,108,122,136, 151,166,182 | 52,73,92,108,123,137, 151,167,182 | 52,73,92,108,122,136, 151,166,182 | 1.4439e+03 | 1.4439e+03 | 1.4440e+03 |
| | 11 | 46,65,81,96,110,122,134, 146,159,171,184 | 45,65,82,96,110,122, 133,146,159,172,185 | 47,67,83,97,110,123,134, 147,160,172,186 | 1.4506e+03 | 1.4506e+03 | 1.4510e+03 |
| Cameraman | 5 | 36,83,122,148,172 | 36,83,122,148,172 | 36,83,122,148,172 | 3.7324e+03 | 3.7324e+03 | 3.7324e+03 |
| | 7 | 33,76,111,134,154,172,200 | 33,76,111,134,154,172,200 | 33,76,111,134,154,172,201 | 3.7627e+03 | 3.7627e+03 | 3.7627e+03 |
| | 9 | 25,52,82,108,126,142, 158,173,201 | 25,52,82,108,126,142, 157,173,200 | 25,52,82,108,126,142, 158,173,201 | 3.7776e+03 | 3.7776e+03 | 3.7776e+03 |
| | 11 | 23,47,74,98,115,129,143, 156,167,178,204 | 24,47,73,97,115,129,143, 156,167,178,205 | 23,47,72,97,115,129,143, 156,167,178 ,205 | 3.7848e+03 | 3.7848e+03 | 3.7850e+03 |

**Table 4** Comparison of mean PSNR, MSE, CPU time and SSIM values between different algorithms using Kapur's method

| Images | *n* | CS | ACS | IACS | CS | ACS | IACS |
|---|---|---|---|---|---|---|---|
| (a) PSNR value comparison | | | | | (b) PSNR value comparison | | |
| Lena | 5 | 21.0383 | 21.0383 | 21.0383 | 511.9813 | 511.9813 | 511.9813 |
| | 7 | 22.2627 | 22.5240 | 23.6316 | 386.3805 | 364.1523 | 283.3549 |
| | 9 | 24.2503 | 24.4584 | 24.9585 | 245.5116 | 233.6096 | 208.1279 |
| | 11 | 25.6715 | 26.3240 | 26.8165 | 177.3152 | 152.0990 | 135.5399 |
| Baboon | 5 | 21.2950 | 21.2950 | 21.2950 | 482.5294 | 482.5924 | 482.5924 |
| | 7 | 23.7380 | 23.8566 | 23.9485 | 274.9776 | 267.5955 | 261.9814 |
| | 9 | 25.5913 | 25.7084 | 25.9289 | 179.5357 | 174.7045 | 166.0659 |
| | 11 | 26.3955 | 26.5876 | 27.1958 | 149.3852 | 142.4394 | 124.5233 |
| Cameraman | 5 | 20.9418 | 20.9418 | 20.9418 | 523.4753 | 523.4753 | 523.4753 |
| | 7 | 23.5016 | 23.5356 | 23.5662 | 290.3220 | 288.0938 | 286.0681 |
| | 9 | 25.5867 | 25.4985 | 25.6646 | 185.8428 | 179.6982 | 176.4676 |
| | 11 | 26.3194 | 26.3745 | 26.7061 | 152.1317 | 150.0434 | 138.8779 |
| (c) CPU time comparison (s) | | | | | (d) SSIM comparison | | |
| Lena | 5 | 8.4326 | 7.5221 | 6.9415 | 0.7715 | 0.7715 | 0.7715 |
| | 7 | 8.9781 | 8.3447 | 7.6382 | 0.7883 | 0.7980 | 0.8220 |
| | 9 | 9.9341 | 9.2058 | 8.3867 | 0.8325 | 0.8359 | 0.8431 |
| | 11 | 10.8084 | 10.0695 | 9.1876 | 0.8549 | 0.8668 | 0.8735 |
| Baboon | 5 | 8.1783 | 7.4452 | 6.7894 | 0.7957 | 0.7957 | 0.7957 |
| | 7 | 8.8817 | 8.2604 | 7.4652 | 0.8639 | 0.8672 | 0.8687 |
| | 9 | 9.8245 | 9.0177 | 8.3255 | 0.9008 | 0.9033 | 0.9063 |
| | 11 | 10.8056 | 9.8812 | 9.1314 | 0.9157 | 0.9183 | 0.9266 |
| Cameraman | 5 | 8.3212 | 7.3096 | 6.7519 | 0.7095 | 0.7095 | 0.7095 |
| | 7 | 8.9885 | 8.1043 | 7.5578 | 0.7406 | 0.7411 | 0.7417 |
| | 9 | 9.6794 | 9.1231 | 8.3332 | 0.7582 | 0.7595 | 0.7604 |
| | 11 | 10.6016 | 9.9064 | 9.1643 | 0.7637 | 0.7690 | 0.7739 |

curve between different algorithms are shown visually in Figs. 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 and 17.

#### 4.3.1 Analysis of the consequences employing Otsu's method

In Table 2, all the algorithms perform equally for $n$=5, however, as the number of thresholds increases, it's clear that IACS algorithm has got higher values of PSNR which leads to a reduced MSE, SSIM values are also the highest, ACS algorithm has obtained acceptable results than CS. Hence the IACS algorithm exhibits more accuracy and robustness. Furthermore, Table 2c presents the information of average CPU time, compared to other algorithms, IACS has faster computational speed with minimum CPU time, ACS closely follows IACS. It can also be summarized that the proposed IACS algorithm can keep the lowest computational time along with the threshold levels increase.

For the visual analysis, the segmented results are shown in Figs. 6, 7 and 8. The three algorithms have produced better quality of the segmented images when $n$=11. However, the segmented results obtained by IACS algorithm are more pleasant visually, ACS is slightly

**Table 5** Comparison of optimal thresholds and corresponding fitness values between CS, ACS and IACS using Kapur's entropy method

| Images | $n$ | Mean optimal thresholds | | | Mean fitness values | | |
|---|---|---|---|---|---|---|---|
| | | CS | ACS | IACS | CS | ACS | IACS |
| Lena | 5 | 26,64,97,138,179 | 26,64,97,138,179 | 26,64,97,138,179 | 21.2649 | 21.2649 | 21.2649 |
| | 7 | 26,63,96,129,162,193,243 | 26,64,94,127,161,192,243 | 26,64,95,126,160,191,243 | 26.3552 | 26.3777 | 26.3822 |
| | 9 | 25,59,84,110,132,157,179, 204,243 | 26,59,83,107,131,155,179, 203,243 | 26,59,83,108,133,158,183, 207,243 | 30.8525 | 30.8992 | 30.9065 |
| | 11 | 21,47,69,91,111,132,152, 172,192,212,243 | 24,47,67,86,106,126,149, 171,191,213,243 | 25,52,72,91,111,130,150, 172,193,214,243 | 34.8719 | 34.9058 | 34.9348 |
| Baboon | 5 | 47,81,117,158,220 | 47,81,117,158,220 | 47,81,117,158,220 | 21.3673 | 21.3673 | 21.3673 |
| | 7 | 28,52,81,111,142,172,220 | 28,53,82,111,141,172,220 | 28,53,81,111,141,172,220 | 26.3270 | 26.3274 | 26.3274 |
| | 9 | 26,49,72,95,119,143,168, 193,222 | 27,49,73,97,120,144,168, 192,221 | 27,49,72,94,117,141,165, 190,220 | 30.8630 | 30.8740 | 30.8748 |
| | 11 | 26,46,66,87,107,129,150, 171,192,220,236 | 25,46,67,87,108,129,149,170, 191,220,237 | 26,46,68,88,109,129,148, 170,192,220,237 | 35.0336 | 35.0732 | 35.0878 |
| Cameraman | 5 | 24,62,101,146,195 | 24,62,101,146,195 | 24,62,101,146,195 | 21.2488 | 21.2488 | 21.2488 |
| | 7 | 23,59,95,124,156,191,216 | 24,59,95,125,156,191,217 | 23,59,95,124,156,191,217 | 26.5844 | 26.5855 | 26.5855 |
| | 9 | 19,44,70,95,119,145,169, 191,216 | 20,45,69,95,120,144,168, 191,217 | 20,45,70,96,120,145,168, 191,217 | 31.2372 | 31.2402 | 31.2453 |
| | 11 | 19,41,63,85,104,126,148, 169,191,211,232 | 19,40,63,84,105,126,148, 169,191,211,232 | 19,42,65,88,108,129,150, 170,191,211,232 | 35.4765 | 35.4839 | 35.4970 |

**Table 6** Comparison of mean PSNR, MSE, CPU time and SSIM values between different algorithms using Tsallis method

| Images | $n$ | CS | ACS | IACS | CS | ACS | IACS |
|---|---|---|---|---|---|---|---|
| (a) PSNR value comparison | | | | | (b) MSE comparison | | |
| Lena | 5 | 21.0383 | 21.0383 | 21.0383 | 511.9813 | 511.9813 | 511.9813 |
| | 7 | 22.1921 | 22.2271 | 23.6187 | 392.7075 | 389.7416 | 284.7962 |
| | 9 | 24.0711 | 24.3698 | 24.9259 | 255.0979 | 238.0847 | 209.3731 |
| | 11 | 25.6010 | 26.0544 | 26.7735 | 182.3216 | 161.5543 | 137.0723 |
| Baboon | 5 | 21.2950 | 21.2950 | 21.2950 | 482.3216 | 482.3216 | 482.3216 |
| | 7 | 23.7236 | 23.8240 | 23.9151 | 275.8934 | 269.6454 | 263.9901 |
| | 9 | 25.4600 | 25.6887 | 25.8588 | 185.6943 | 175.5128 | 168.7902 |
| | 11 | 26.6043 | 26.8219 | 27.2100 | 142.3692 | 135.2800 | 123.9134 |
| Cameraman | 5 | 20.9418 | 20.9418 | 20.9418 | 523.4753 | 523.4753 | 523.4753 |
| | 7 | 23.5111 | 23.5311 | 23.5556 | 289.7218 | 288.5328 | 286.7626 |
| | 9 | 25.5442 | 25.6053 | 25.6804 | 181.4624 | 178.9001 | 175.8752 |
| | 11 | 26.2291 | 26.3649 | 26.7181 | 155.0875 | 150.4582 | 138.4693 |
| (c) CPU time comparison (s) | | | | | (d) SSIM comparison | | |
| Lena | 5 | 9.5379 | 8.5060 | 7.9931 | 0.7715 | 0.7715 | 0.7715 |
| | 7 | 10.5661 | 9.3823 | 8.7194 | 0.7872 | 0.7876 | 0.8220 |
| | 9 | 11.4806 | 10.2308 | 9.5662 | 0.8294 | 0.8328 | 0.8413 |
| | 11 | 12.5901 | 11.3930 | 10.4029 | 0.8536 | 0.8608 | 0.8727 |
| Baboon | 5 | 9.4943 | 8.6282 | 7.8874 | 0.7957 | 0.7957 | 0.7957 |
| | 7 | 10.5696 | 9.4960 | 8.7405 | 0.8632 | 0.8667 | 0.8680 |
| | 9 | 11.3745 | 10.4586 | 9.6199 | 0.8991 | 0.9032 | 0.9055 |
| | 11 | 12.3720 | 11.3412 | 10.4581 | 0.9191 | 0.9224 | 0.9270 |
| Cameraman | 5 | 9.2219 | 8.6487 | 7.8934 | 0.7095 | 0.7095 | 0.7095 |
| | 7 | 10.2626 | 9.4833 | 8.7314 | 0.7415 | 0.7418 | 0.7430 |
| | 9 | 11.0546 | 10.4926 | 9.7636 | 0.7569 | 0.7579 | 0.7595 |
| | 11 | 12.1457 | 11.4214 | 10.6497 | 0.7680 | 0.7699 | 0.7740 |

better than CS. In Fig. 6, at $n$=11, the background in the Lena image is the clearest using IACS algorithm, to some extent, the hat and hair are identifiable clearly. But the segmentation quality of CS algorithm is not accurate, especially for hat. This phenomenon of segmenting is also observed in other two images.

Figure 9 presents the convergence curve of the three algorithms for $n$=11. In Fig. 9a, after about 60 iterations, the maximum fitness value is acquired only in the case of IACS algorithm. ACS and CS algorithms require at least 150 iterations. In Fig. 9b, at 20 iterations, the fitness value is 1.445e+03 by using IACS algorithm. However, the fitness value is 1.440e+03 for ACS and CS algorithms at 20 iterations. In Fig. 9c, when the fitness value is 3.782e+03, there are not more than 20 iterations for the IACS algorithm, while the ACS algorithm requires 40 iterations and the CS algorithm needs 60 iterations at least. Therefore, the proposed IACS algorithm is much faster to get the global optimum.

**Table 7** Comparison of optimal thresholds and corresponding fitness values between CS, ACS and IACS using Tsallis entropy method

| Images | *n* | Mean optimal thresholds | | | Mean fitness values | | |
|---|---|---|---|---|---|---|---|
| | | CS | ACS | IACS | CS | ACS | IACS |
| Lena | 5 | 26,64,97,138,179 | 26,64,97,138,179 | 26,64,97,138,179 | 23.2902 | 23.2902 | 23.2902 |
| | 7 | 26,64,97,130,162,193,241 | 26,63,95,128,161,191,243 | 26,64,94,128,162,192,243 | 28.6672 | 28.7193 | 28.7265 |
| | 9 | 25,60,85,109,133,156,179, 206,243 | 25,60,84,109,134,157,180, 204,243 | 26,59,82,107,131,155,177, 201,243 | 33.4121 | 33.4521 | 33.4876 |
| | 11 | 24,49,70,91,111,131,149, 169,191,212,243 | 24,49,70,92,112,133,153, 173,192,213,243 | 25,54,74,95,1151,134,154, 175,195,214,243 | 37.6106 | 37.6767 | 37.7262 |
| Baboon | 5 | 47,81,117,158,220 | 47,81,117,158,220 | 47,81,117,158,220 | 23.4172 | 23.4172 | 23.4172 |
| | 7 | 28,54,84,113,142,173,220 | 28,53,82,111,141,172,220 | 28,53,82,112,142,172,220 | 28.6607 | 28.6607 | 28.6610 |
| | 9 | 26,48,73,96,120,144,169, 195,223 | 27,50,73,97,120,143,166, 190,220 | 27,49,72,95,119,142,166, 190,220 | 33.4186 | 33.4250 | 33.4435 |
| | 11 | 25,46,68,90,110,130,150, 171,192,220,236 | 25,46,67,87,108,129,150, 171,192,220,237 | 26,47,69,89,110,130,150, 171,192,220,238 | 37.7478 | 37.8081 | 37.8670 |
| Cameraman | 5 | 24,61,100,146,196 | 24,61,100,146,196 | 24,61,100,146,196 | 23.3191 | 23.3191 | 23.3191 |
| | 7 | 23,58,95,124,156,191,216 | 24,59,95,125,156,191,216 | 24,59,95,125,156,191,216 | 28.9853 | 28.9855 | 28.9859 |
| | 9 | 19,45,70,96,120,145,168, 191,216 | 19,44,69,95,119,145,168, 191,217 | 20,45,70,95,120,146,169, 191,217 | 33.8664 | 33.8692 | 33.8771 |
| | 11 | 19,40,62,84,104,125,147, 169,191,212,232 | 19,41,64,85,105,126,147, 169,191,212,233 | 19,42,65,88,108,130,151, 170,191,212,232 | 38.2819 | 38.2957 | 38.3260 |


**Fig. 6** Results of segmenting Lena using CS, ACS and IACS for 4 different threshold levels maximizing Otsu's method

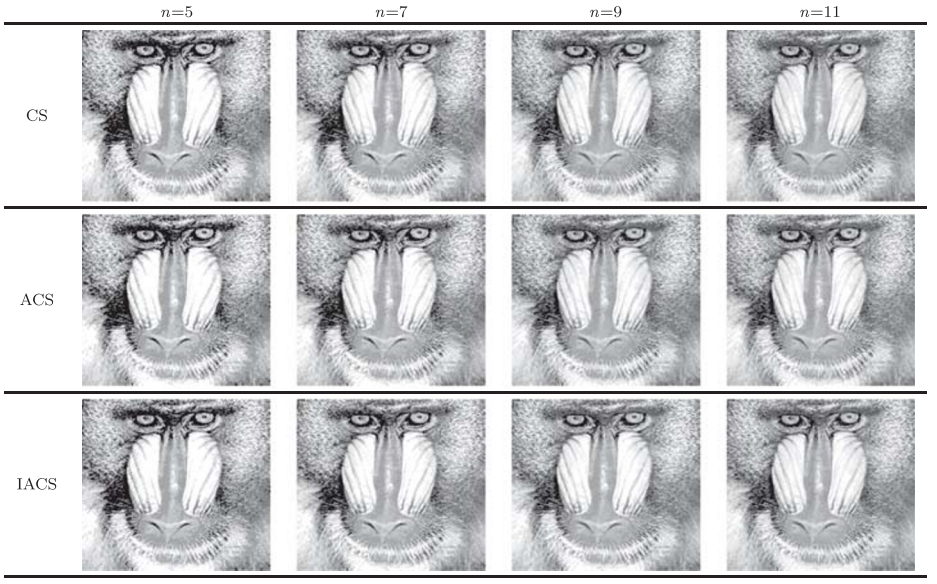



**Fig. 7** Results of segmenting Baboon using CS, ACS and IACS for 4 different threshold levels maximizing Otsu's method

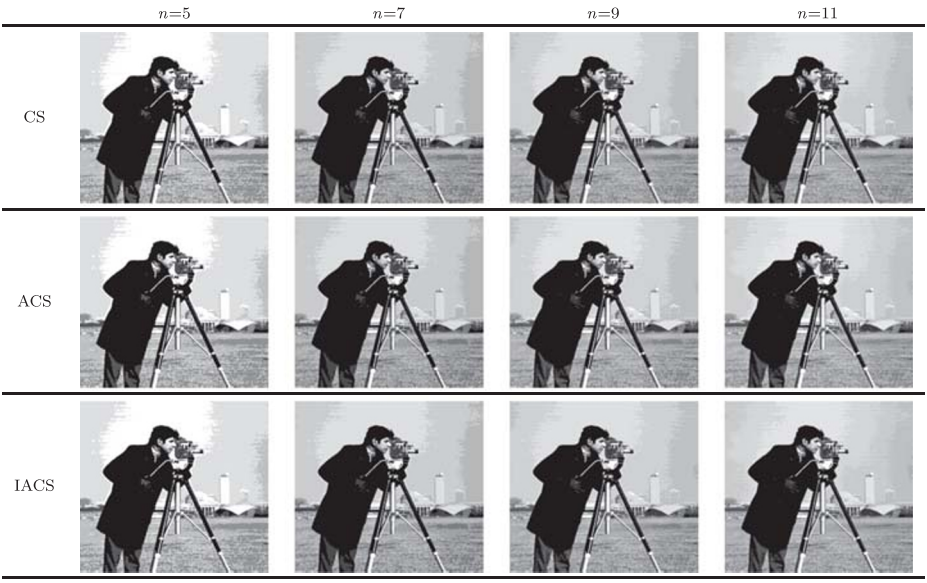
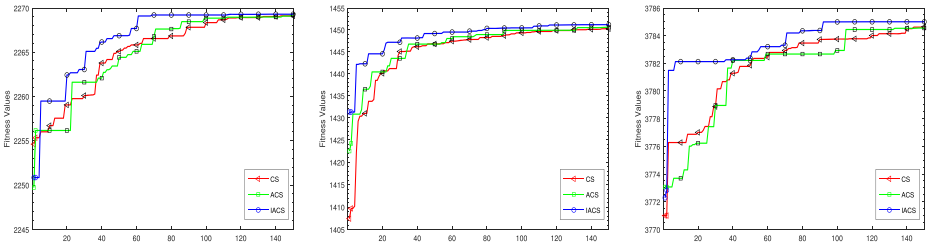



**Fig. 8** Results of segmenting Cameraman using CS, ACS and IACS for 4 different threshold levels maximizing Otsu's method

### 4.3.2 Analysis of the consequences employing Kapur's method

In Table 4, the number of thresholds exceeds 5, for all the test images, IACS algorithm performs superior to the other algorithms quantitatively, it obtains the highest values of PSNR, SSIM and minimum values of MSE, ACS has gained acceptable results than CS. This indicates the IACS is more accurate and robust in image segmentation. Table 4c illustrates the information of average CPU time. IACS algorithm is always around 1 second faster than ACS and CS algorithms. Additionally, at higher threshold levels, IACS algorithm remains the lowest computational time that makes it efficient for segmenting.
In Figs. 10, 11 and 12, the results depict that the segmented outputs based on IACS algorithm are much clearer visually for all the test images, ACS follows the IACS and CS is the




(a) Lena (b) Baboon (c) Cameraman

**Fig. 9** Convergence curves of segmenting using Otsu's method with 11 threshold levels using CS, ACS and IACS algorithms


**Fig. 10** Results of segmenting Lena using CS, ACS and IACS for 4 different threshold levels maximizing Kapur's method

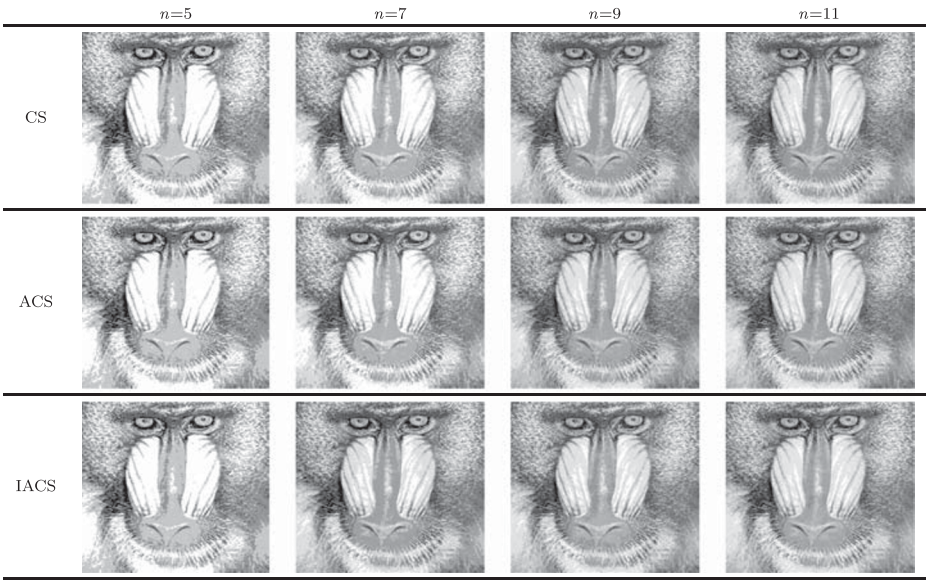



**Fig. 11** Results of segmenting Baboon using CS, ACS and IACS for 4 different threshold levels maximizing Kapur's method

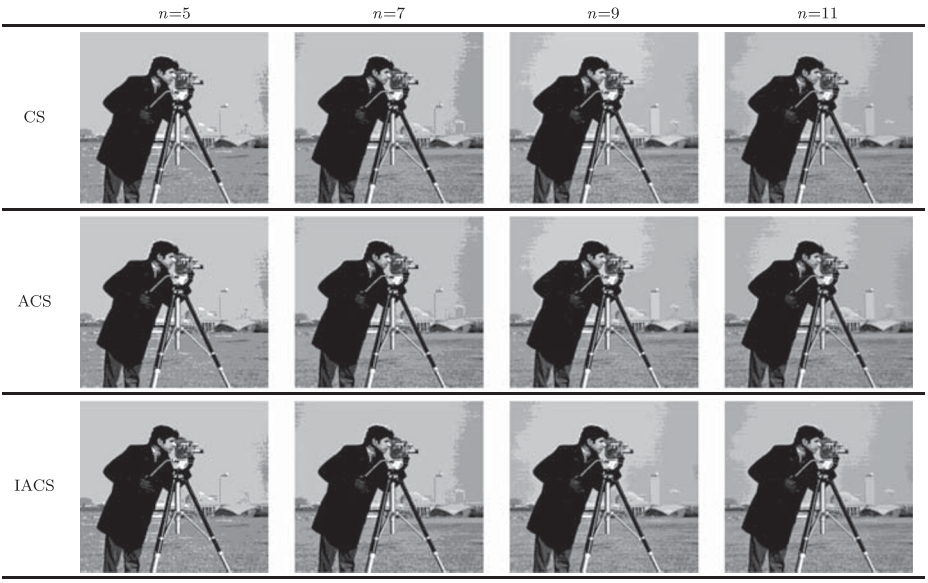
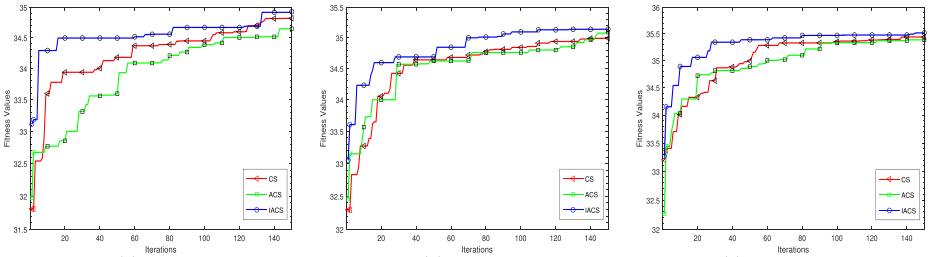



**Fig. 12** Results of segmenting Cameraman using CS, ACS and IACS for 4 different threshold levels maximizing Kapur's method

last. In Fig. 12, for the Cameraman image when $n$=11, the sky and buildings are clear and become recognizable by using IACS algorithm, but the thresholded images gained by the CS and ACS algorithms lose some information to some degrees. Similarly, Figs. 10 and 11 also demonstrate that the results of IACS algorithm seem better qualitatively.

Fig. 13 displays the convergence curve of the three algorithms at $n$=11. In Fig. 13a, when the fitness value is 34.5, just 20 iterations for IACS algorithm, whereas CS requires about 120 iterations and ACS needs 150 iterations at least. In Fig. 13b, it's not competitive for IACS algorithm, the maximum fitness value is obtained at around 100 iterations. For CS and ACS algorithms, the maximum fitness value is gained after about 150 iterations. In Fig. 13c, the convergence curve of CS and ACS are not obviously different. For IACS algorithm, it only takes about 40 iterations to reach the fitness value, while the ACS and CS algorithms require




(a) Lena (b) Baboon (c) Cameraman

**Fig. 13** Convergence curves of segmenting using Kapur's method with 11 threshold levels using CS, ACS and IACS algorithms


**Fig. 14** Results of segmenting Lena using CS, ACS and IACS for 4 different threshold levels maximizing Tsallis method

150 iterations at least. Therefore, the IACS algorithm displays a faster convergence to the best solutions.

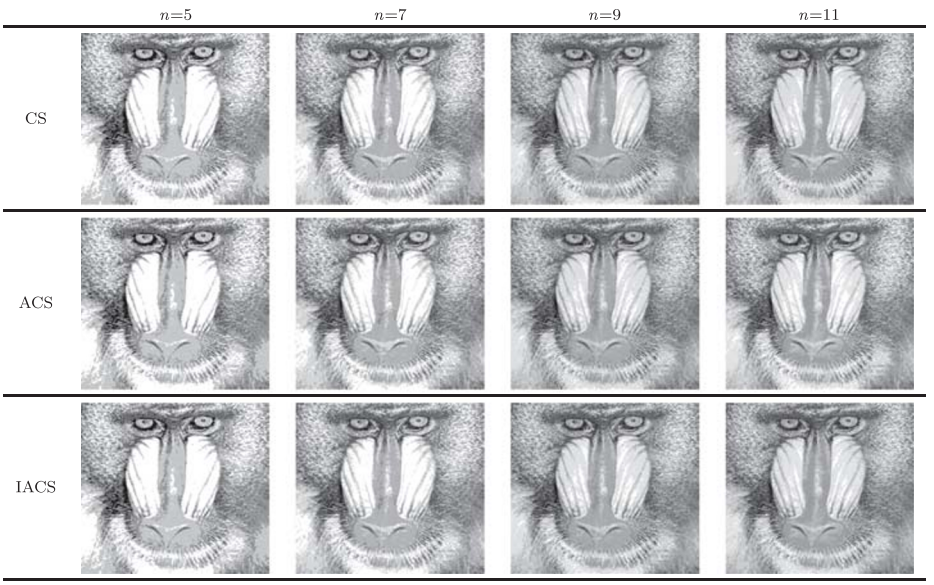



**Fig. 15** Results of segmenting Baboon using CS, ACS and IACS for 4 different threshold levels maximizing Tsallis method

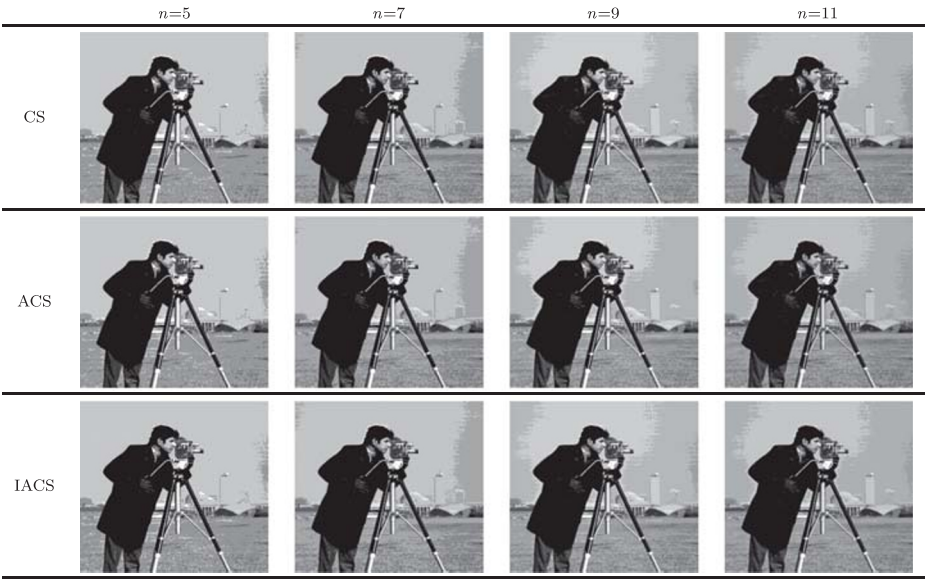



**Fig. 16** Results of segmenting Cameraman using CS, ACS and IACS for 4 different threshold levels maximizing Tsallis method

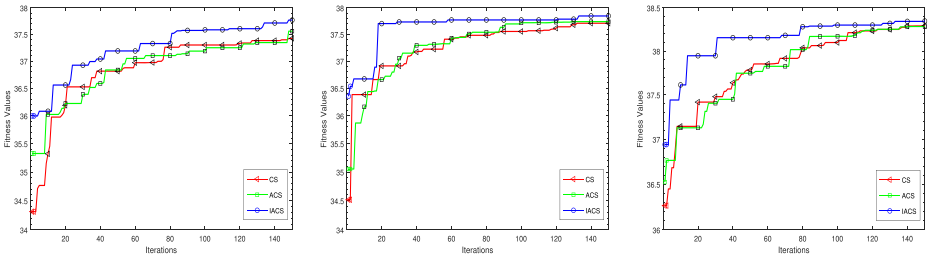
### 4.3.3 Analysis of the consequences employing Tsallis method

In Table 6, with the number of thresholds increasing, it's clear that IACS algorithm has better values of PSNR, SSIM and minimum values of MSE, and ACS algorithm has slightly better results than CS algorithm. In summary, the IACS algorithm can segment images more accurately than other algorithms. In Table 6c, the IACS has taken the least time compared to CS and IACS algorithms, CS algorithm costs the most. At different threshold levels, IACS algorithm is always around 2 seconds faster than CS algorithm and about 1 second faster than ACS algorithm.

Figures. 14, 15 and 16 give the segmented results visually. From these figures, It can be concluded that higher-threshold images contain more details. When $n$=11, for the Baboon image, the results of IACS algorithm make the beard and nose object clearer, but in CS and ACS algorithms, the nose is not clearly distinct.

Further, Fig. 17 shows the convergence curve of the three algorithms at $n$=11. It is clearly shown from Fig. 17a the convergence rate of the three algorithms have produced similar results in the early period, but the fitness value of IACS is always higher than the others. Only IACS obtains the maximum fitness value after about 100 iterations. ACS and CS algorithms require at least 150 iterations. In Fig. 17b, at 20 iterations, the fitness value is about 37.5 for IACS algorithm, whereas at least 80 iterations for ACS and CS algorithms to gain the value. In Fig. 17c, convergence rate of the proposed IACS algorithm is much faster. Only IACS algorithm obtains the maximum fitness value after about 80 iterations. ACS and CS algorithms require at least 150 iterations. Consequently, the IACS outperforms CS and ACS algorithms. In summary, from different

assessment measures, the proposed IACS algorithm gives higher segmentation performance using less CPU time and is well suited for segmenting,




(a) Lena (b) Baboon (c) Cameraman

**Fig. 17** Convergence curves of segmenting using Tsallis method with 11 threshold levels using CS, ACS and IACS algorithms

## 5 Conclusions

Due to multi-level thresholding based on Yang's CS algorithm has larger calculated amount, the IACS algorithm is introduced. Without employing the *Lévy* distribution, the iterative step size based on its fitness adaptively, it enables step size variable in the process of searching. Moreover, the dynamic adjustment strategy is also applied to the discovery probability $p_a$, which can enrich the population variety and avoid premature. The IACS algorithm is then applied to searching optimal thresholds. To validate its performance, it is compared to Yang's CS and ACS algorithms. The results reveal that the IACS algorithm can efficiently select the multi-level thresholds and has better segmented quality than other two algorithms. According to the rate of convergence curve, the IACS algorithm attains its optimal value in less iterations without falling into local optima. Next step is to check the effectiveness of IACS algorithm for all kinds of issues in image processing.

**Acknowledgements** This work was supported by the National Natural Science Foundation of China under Grant No.11601007 and No.11701007.

## References

1. Abhinaya B, Sri Madhava Raja N (2015) Solving Multi-level Image Thresholding Problem-An Analysis with Cuckoo Search Algorithm. Adv Intell Syst Comput 339:177–186
2. Agrawal S, Panda R, Bhuyan S, Panigrahi BK (2013) Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm. Swarm Evol Comput 11:16–30
3. Alihodzic A, Tuba M (2014) Improved bat algorithm applied to multilevel image thresholding. Sci World J 2014:1–16
4. Bhandari AK, Kumar A, Singh GK (2015) Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur's, Otsu and Tsallis functions. Expert Syst Appl 42(3):1573–1601
5. Bhandari AK, Singh VK, Singh GK, Singh GK (2014) Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. Expert Syst Appl 41(7):3538–3560
6. Feng YC, Shen XJ, Chen HP, Zhang XL (2017) Segmentation fusion based on neighboring information for MR brain images. Multi Tools Appli 76(22):23139–23161
7. Ghamisi P, Couceiro MS, Benediktsson JA, Ferreira NMF (2012) An efficient method for segmentation of images based on fractional calculus and natural selection. Expert Syst Appl 39(16):12407–12417
8. Hammouche K, Diaf M, Siarry P (2008) A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation. Comput Vision Image Understan 109(2):163–175

9. Kapur JN, Sahoo PK, Wong AKC (1985) A new method for gray-level picture thresholding using the entropy of the histogram. Comput Vision Graphics Image Process 29(3):273–285
10. Li XT, Yin MH (2015) Modified cuckoo search algorithm with self adaptive parameter method. Inf Sci 298(20):80–97
11. Mantegna RN (1994) Fast, accurate algorithm for numerical simulation of *lévy* stable stochastic processes. Phys Rev E 49(5):4677–4683
12. Naik MK, Nath MR, Wunnava A, Sahany S, Panda R (2015) A new adaptive Cuckoo search algorithm. In: Proceeding of international conference on recent trends in information systems, pp 1–5
13. Naik MK, Panda R (2016) A novel adaptive cuckoo search algorithm for intrinsic discriminant analysis based face recognition. Appl Soft Comput 38:661–675
14. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 9(1):62–66
15. Pal NR, Pal SK (1993) A review on image segmentation techniques. Pattern Recogn 26(9):1277–1294
16. Panda R, Agrawal S, Bhuyan S (2013) Edge magnitude based multilevel thresholding using Cuckoo search technique. Expert Syst Appl 40(18):7617–7628
17. Portes de Albuquerque M, Esquef IA, Gesualdi Mello AR (2004) Image thresholding using Tsallis entropy. Pattern Recogn Lett 25(9):1059–1065
18. Sahoo PK, Soltani S, Wong AKC (1988) A survey of thresholding techniques. Comput Vis Graph Image Process 41(2):233–260
19. Sezgin M, Sankur B (2004) Survey over image thresholding techniques and quantitative performance evaluation. J Electron Imaging 13(1):146–168
20. Sharma A, Chaturvedi R, Dwivedi U, Kumar S, Reddy S (2018) Firefly algorithm based Effective gray scale image segmentation using multilevel thresholding and Entropy function. Int J Pure Appl Math 118(5):437–443
21. Suresh S, Lal S (2016) An efficient cuckoo search algorithm based multilevel thresholding for segmentation of satellite images using different objective functions. Expert Syst Appl 58:184–209
22. Tiwari V (2012) Face recognition based on cuckoo search algorithm. Ind J Comput Sci Eng 3(3):401–405
23. Tsallis C (1988) Possible generalization of Boltzmann-Gibbs statistics. J Stat Phys 52(1):479–487
24. Valian E, Mohanna S, Tavakoli S (2011) Improved cuckoo search algorithm for feed forward neural network training. Int J Artif Intell Appl 2(3):36–43
25. Wang LJ, Zhong YW (2015) Cuckoo search algorithm with chaotic maps. Math Probl Eng 2015:1–14
26. Wang W, Xie C (2018) A cuckoo search algorithm based on self-adjustment strategy. J Phys Conference Series 1087(2):1–7
27. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
28. Wei HT, Yang Q (2017) A multilevel threshold segmentation technique using self-adaptive Cuckoo search algorithm. In: Advanced Information Technology, Electronic and Automation Control Conference, pp 2292–2295
29. Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1(4):330–343
30. Yang XS, Deb S (2013) Multiobjective cuckoo search for design optimization. Comput Oper Res 40(6):1616–1624
31. Yang XS, Suash D (2009) Cuckoo search via *lévy* flights, NaBIC, USA
32. Zhang YD, Wu LN (2011) Optimal Multi-Level thresholding based on maximum tsallis entropy via an artificial bee colony approach. Entropy 13(4):841–859
33. Zhou YQ, Yang X, Ling Y, Zhang JZ (2018) Meta-heuristic moth swarm algorithm for multilevel thresholding image segmentation. Multimed Tools Appl 77(18):23699–23727

**Min Sun** Her current research interests are computational intelligence, image processing.