Expert Systems with Applications 38 (2011) 15549–15564

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Optimal multilevel thresholding using bacterial foraging algorithm

P.D. Sathya [a,*], R. Kayalvizhi [b]

[a] Department of Electrical Engineering, Faculty of Engineering and Technology, Annamalai University, Chidambaram 608 002, Tamilnadu, India
[b] Department of Instrumentation Engineering, Faculty of Engineering and Technology, Annamalai University, Chidambaram 608 002, Tamilnadu, India

ARTICLE INFO

Keywords:
Multilevel thresholding
Image segmentation
Histogram
Bacterial foraging
Kapur's function
Otsu's function

ABSTRACT

The conventional multilevel thresholding methods are efficient for bi-level thresholding. However, they are computationally expensive extending to multilevel thresholding since they exhaustively search the optimal thresholds to optimize the objective functions. To overcome this drawback, a bacterial foraging (BF) algorithm based multilevel thresholding is presented in this paper. The BF algorithm is used to find the optimal threshold values for maximizing the Kapur's and Otsu's objective functions. The feasibility of the proposed BF technique has been tested on ten standard test images and benchmarked with particle swarm optimization algorithm (PSO) and genetic algorithm (GA). Experimental results of both qualitative and quantitative comparative studies for several existing methods illustrate the effectiveness and robustness of the proposed algorithm.

1. Introduction

In image understanding, image segmentation is always the first of all image processing steps and thresholding is the one of the most commonly used segmentation methods. In general, if the gray level histogram of the image is bi-model, the image objects are clearly distinguishable from the background. In this case, it is easy to choose a threshold value by taking the value that is in the valley between two peaks of the histogram. However, in the real world, the gray level histograms of the images are always multimodal. Therefore, it is not simple to determine the exact locations of distinct valleys in multimodal histograms, which can segment the image efficiently. Hence, the problem of multilevel thresholding is regarded as an important area of research interest among the research communities, worldwide.

Over these years, many thresholding techniques have been developed (Abak, Baris, & Sankur, 1997; Brink, 1995; Huang & Wang, 1995). Sahoo, Soltani, and Wong (1988) have presented a thorough survey of a variety of thresholding techniques, among which global histogram based algorithms are widely employed to determine the threshold. This global thresholding technique can be classified into parametric and nonparametric approaches (Yen, Chang, & Chang, 1995). In parametric approaches (Synder, Bilbro, Logenthiran, & Rajala, 1990; Weszka & Rosenfeld, 1979), the gray level distribution of each class has a probability density function that follows a Gaussian distribution. This approach is computationally expensive and time-consuming. Nonparametric approaches determine the threshold values in an optimal fashion based on a given criterion. The nonparametric approaches are robust and more accurate than the parametric methods.

Otsu's method (1979) chooses the optimal thresholds by maximizing the between-class variance of gray levels. Kapur, Sahoo, and Wong (1985) finds the optimal threshold values based on the maximization of the entropy of the histogram. Kittler and Illingworth (1986) assume that the gray levels of each object in an image are normally distributed. The above nonparametric approaches are simple and effective in bi-level thresholding.

The Otsu's and Kapur's method can be easily extended to multilevel thresholding problem but inefficient in determining the optimal thresholds due to the exponential growth in computation time. To improve the efficiency, many methods have been proposed for solving the multilevel thresholding problem (Tsai, 1995). Liao, Chen, and Chung, (2001) showed that the recursive algorithm greatly reduces the computational complexity of determining the multilevel thresholds by accessing a look-up table when compared with conventional Otsu and Kapur methods. But it still suffers from the problem of long durated processing time when the number of thresholds m increases.

To eliminate such problems the evolutionary techniques have been applied in solving multilevel thresholding problems. The GA, ACO and PSO, which are forms of probabilistic heuristic algorithm, have been successfully used in multilevel thresholding (Maitra & Chatterjee, 2008; Shu-Kai, Fan, & Yen, 2007; Yin, 1999;

* Corresponding author.
E-mail addresses: pd.sathya@yahoo.in (P.D. Sathya), mithuvig.knr@gmail.com (R. Kayalvizhi).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.06.004

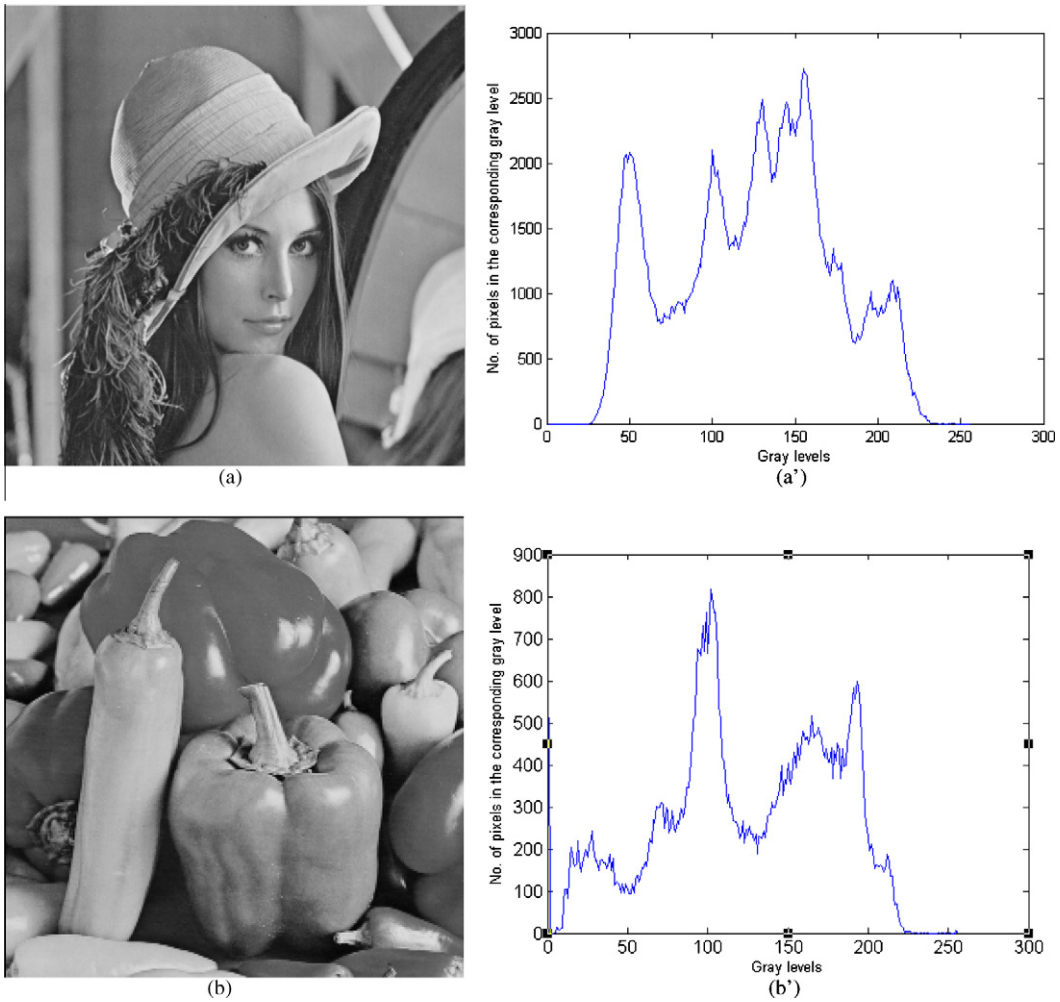

Fig. 1. Test images and their histograms (a) Lena, (b) Pepper, (c) Baboon, (d) Hunter, (e) Map, (f) Cameraman, (g) Living room, (h) House, (i) Airplane, (j) Butterfly.

Ye, Zheng, Yu, X, & Ning, 2006). The GA method is usually faster because the GA has parallel search techniques, which emulate natural genetic operations. Though the GA methods have been employed successfully to solve complex nonlinear optimization problems, recent research has identified some deficiencies in GA performance (Fogel, 2000). This degradation in efficiency is apparent when the optimized parameters are highly correlated and the premature convergence of the GA reduces its search capability. The particle swarm optimization (PSO) has been applied to the multilevel thresholding for image segmentation (Maitra & Chatterjee, 2008).

Bacterial foraging (BF), first introduced by passino, is one of the modern heuristic algorithm (Passino, 2002). The foraging behavior of *Escherichia coli* bacteria present in human intestine, which includes the method of locating, handling and ingesting food, has been successfully mimicked to propose the algorithm.

In this paper, the BF algorithm is presented for solving multilevel thresholding problem in image segmentation. The proposed algorithm is tested on ten standard test images and compared with the PSO and GA methods.

2. Problem formulation of optimum thresholding methods

The optimal thresholding methods search the thresholds such that the segmented classes on the histogram satisfy the desired property. This is performed by maximizing an objective function which uses the selected thresholds as the parameters. In this paper, two broadly used optimal thresholding methods namely entropy criterion (Kapur's) method and between-class variance (Otsu's) method are used.

2.1. Entropy criterion method (Kapur's method)

The entropy criterion method has been employed in determining whether the optimal thresholding can provide histogram-based image segmentation with satisfactory desired characteristics (Nikhil & Sankar, 1989; Kapur et al., 1985). Kapur has developed the algorithm for bi-level thresholding and this bi-level thresholding can be described as follows: let there be L gray levels in a given image and these gray levels are in a given image and these gray levels are in the range $\{0,1,2,\ldots,(L-1)\}$. Then one can define $P_i = h(i)\ N$, $(0 \leqslant i \leqslant (L-1))$ where $h(i)$ denotes number of pixels for the corresponding gray-level L and N denotes total number of pixels in the image which is equal to $\sum_{i=0}^{L-1} h(i)$.

Then the objective is to maximize the fitness function

$$f(t) = H_0 + H_1, \tag{1}$$

where

$$H_0 = -\sum_{i=0}^{t-1} \frac{P_i}{\omega_0} \ln \frac{P_i}{\omega_0}, \quad \omega_0 = \sum_{i=0}^{t-1} P_i \quad \text{and}$$

$$H_1 = -\sum_{i=t}^{L-1} \frac{P_i}{\omega_1} \ln \frac{P_i}{\omega_1}, \quad \omega_1 = \sum_{i=t}^{L-1} P_i.$$

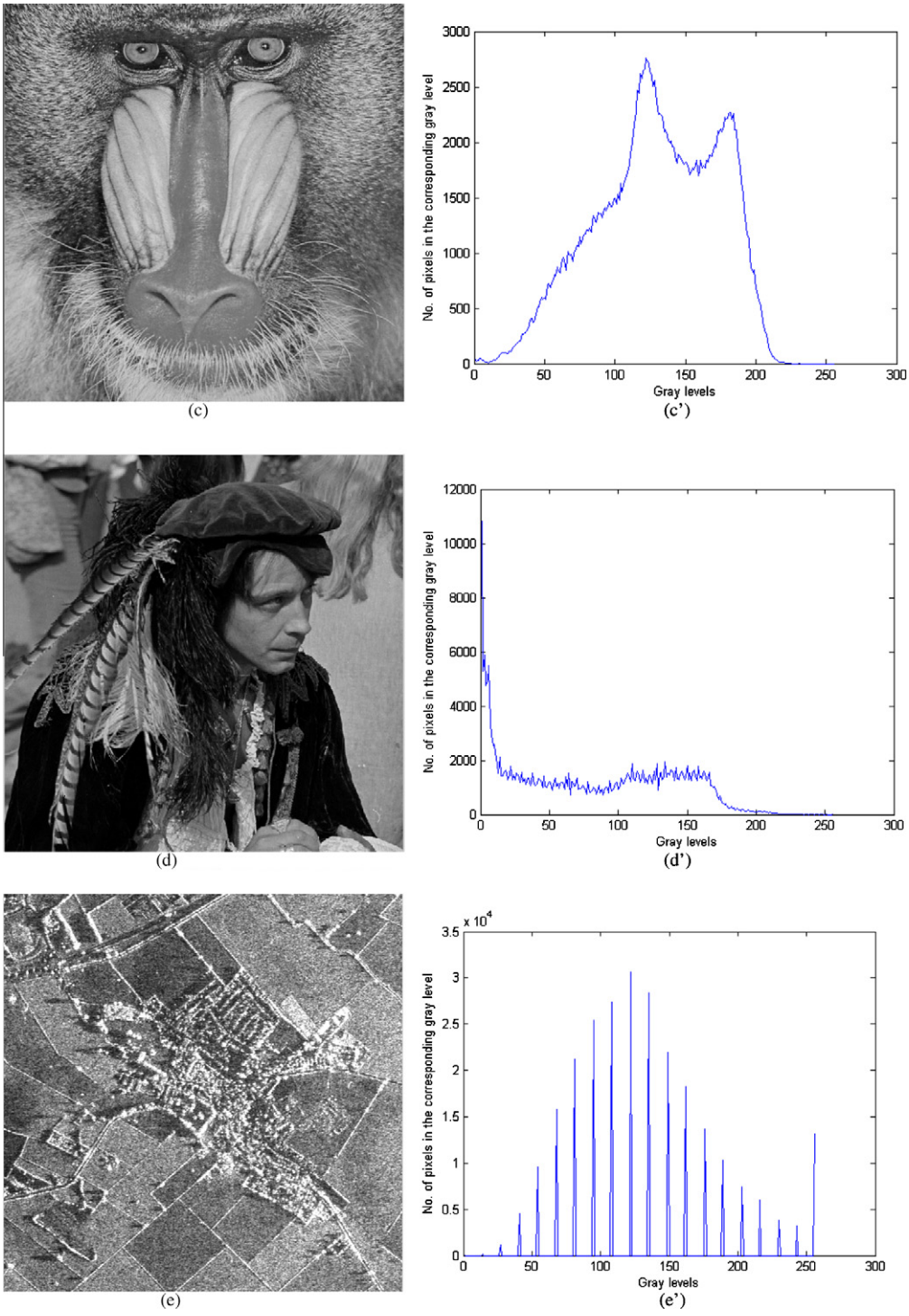

Fig. 1 (*continued*)

The optimal threshold is the gray level that maximizes (1). This Kapur's entropy criterion method tries to achieve a centralized distribution for each histogram-based segmented region of the image.

This Kapur's entropy criterion method has also been extended to multilevel thresholding and can be described as follows: The optimal multilevel thresholding problem can be configured as an m-dimensional optimization problem, for determination of m optimal thresholds for a given image $[t_1,t_2,\ldots,t_m]$, where the aim is to maximize the objective function:

$$f([t_1,t_2,\ldots t_m]) = H_0 + H_1 + H_2 + \cdots + H_m, \tag{2}$$

where

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

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

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

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

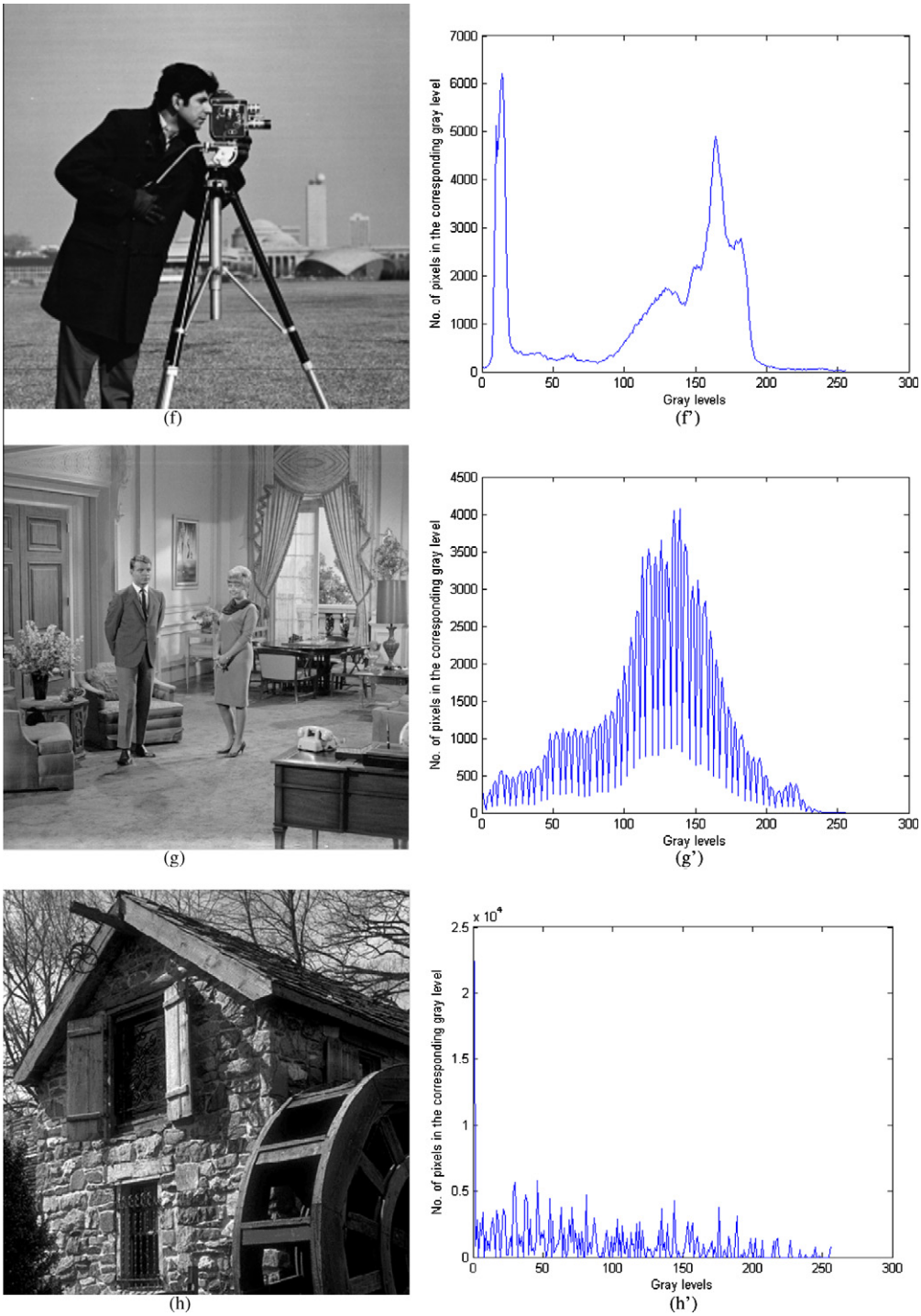

Fig. 1 (*continued*)

2.2. Between-class variance method (Otsu's method)

As Kapur based entropy criterion method, this Otsu based between-class variance method has also been employed in determining whether the optimal thresholding can provide histogram-based image segmentation with satisfactory desired characteristics (Deng-Yuan & Chia-Hung, 2009; Sahoo et al., 1988). The Otsu based between-class variance algorithm can be described as follows: Assume that an image can be represented in L gray levels $(1,2,\ldots,L)$. The number of pixels with gray level i is denoted by f_i, then total number of pixels equals N (equal to $f_1 + f_2 + \cdots + f_L$). For a given gray level image the occurrence probability of gray level i is given by

$$p_i = \frac{f_i}{N}, \quad p_i \geqslant 0, \quad \sum_{i=1}^{L} p_i = 1.$$

If an image can be divided into two classes, C_0 and C_1, by a threshold at a level t, class C_0 contains the gray levels from 0 to t and class C_1 consists of the other gray levels with $t + 1$ to L. Then, the gray level

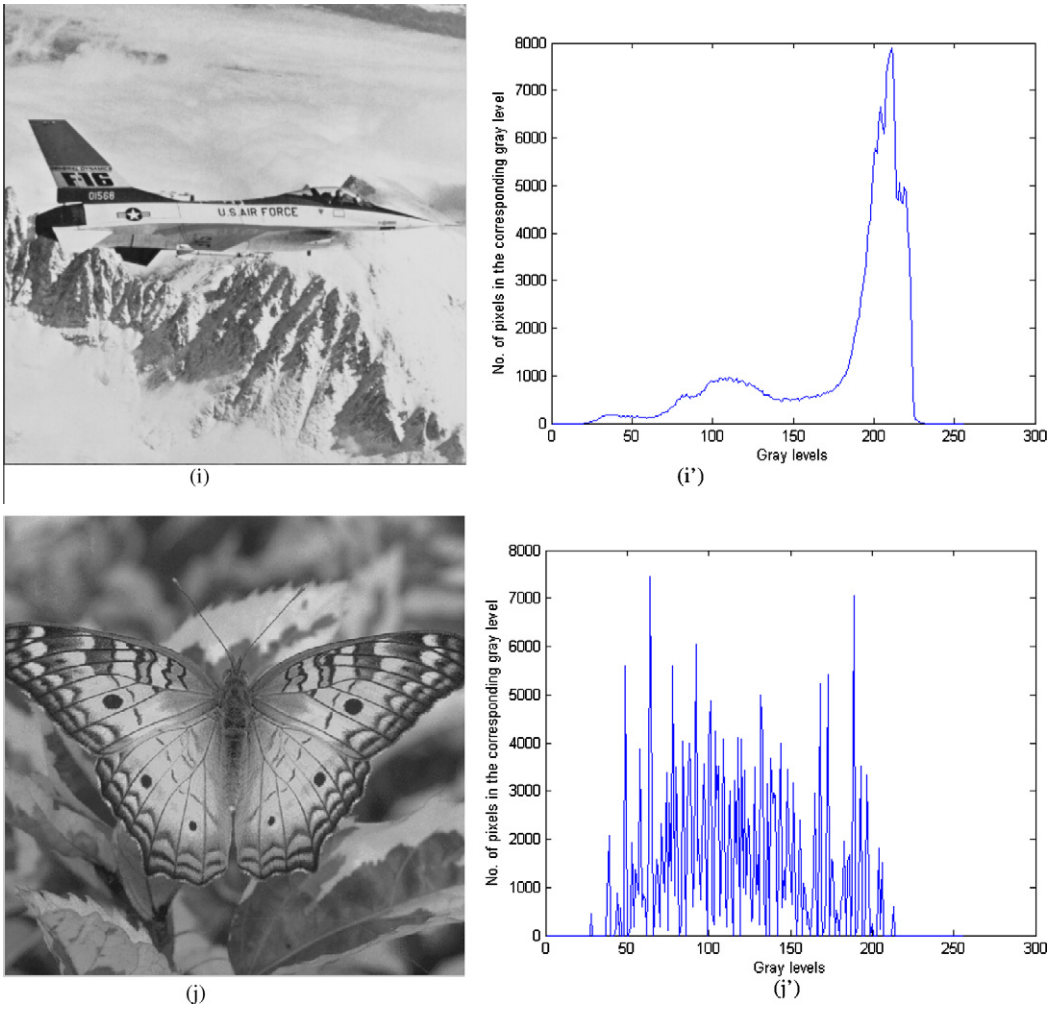

Fig. 1 (*continued*)

probabilities ($\omega_0(t)$ and $\omega_1(t)$) distributions for the two classes are as follows:

$$C_0 : \frac{p_1}{\omega_0(t)}, \ldots \frac{p_t}{\omega_0(t)} \quad \text{and}$$

$$C_1 : \frac{p_{t+1}}{\omega_1(t)}, \ldots \frac{p_L}{\omega_1(t)},$$

where, $\omega_0(t) = \sum_{i=1}^{t} p_i$ and

$$\omega_1(t) = \sum_{i=t+1}^{L} p_i.$$

Mean levels μ_0 and μ_1 for classes C_0 and C_1 are as follows:

$$\mu_0 = \sum_{i=1}^{t} \frac{ip_i}{\omega_0(t)},$$

$$\mu_1 = \sum_{i=t+1}^{L} \frac{ip_i}{\omega_1(t)}.$$

Let μ_T be the mean intensity for the whole image, it is easy to show that

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T, \quad \text{and} \quad \omega_0 + \omega_1 = 1.$$

Using discriminant analysis, Otsu based between-class variance thresholded image can be defined as follows:

$$\sigma_B^2 = \sigma_0 + \sigma_1,$$

Table 1
Parameters used for BF methods.

Parameter	Value
Number of bacterium (s)	20
Number of chemotatic steps (N_c)	10
Swimming length (N_s)	10
Number of reproduction steps (N_{re})	4
Number of elimination of dispersal events (N_{ed})	2
Depth of attractant ($d_{attract}$)	0.1
Width of attract ($\omega_{attract}$)	0.2
Height of repellent ($h_{repellent}$)	0.1
Width of repellent ($\omega_{repellent}$)	10
Probability of elimination and dispersal (P_{ed})	0.02

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

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

For bi-level thresholding, Otsu selects an optimal threshold t^* that maximizes the between-class varianceσ_B^2; that is

$$t^* = \arg_{1 \leqslant t \leqslant L} \max \{\sigma_B^2\}.$$

The above formula can be easily extended to multilevel thresholding of an image. Assuming that there are $m-1$ thresholds, (t_1,t_2,…, t_{m-1}), which divide the original image into m classes: C_0 for

Fig. 2. Thresholded images obtained by Kapur-BF method (a)–(j) represents 3-level thresholding, (a′)–(j′) represents 4-level thresholding, (a″)–(j″) represents 5-level thresholding.

$[1,\ldots,t_1]$, C_1 for $[t_1+1,\ldots,t_2]\ldots$ and C_{m-1} for $[t_{m-1}+1,\ldots,L]$, the optimal thresholds $(t_1^*, t_2^*, \ldots, t_{m-1}^*)$ are chosen by maximizing σ_B^2 as follows:

$$(t_1^*, t_2^*, \ldots, t_{m-1}^*) = \underset{1\leqslant t_1 \leqslant \ldots \leqslant t_{m-1}\leqslant L}{\arg} \max\{\sigma_B^2(t_1, t_2, \ldots, t_{m-1})\}, \tag{3}$$

where $\sigma_B^2 = \sigma_0 + \sigma_1 + \sigma_2 \ldots + \sigma_{m-1}$

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

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

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

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

$\sigma_{m-1} = \omega_{m-1}(\mu_{m-1} - \mu_T)^2$.

The Kapur and Otsu methods have been proven as an efficient method for bi-level thresholding in image segmentation. However,

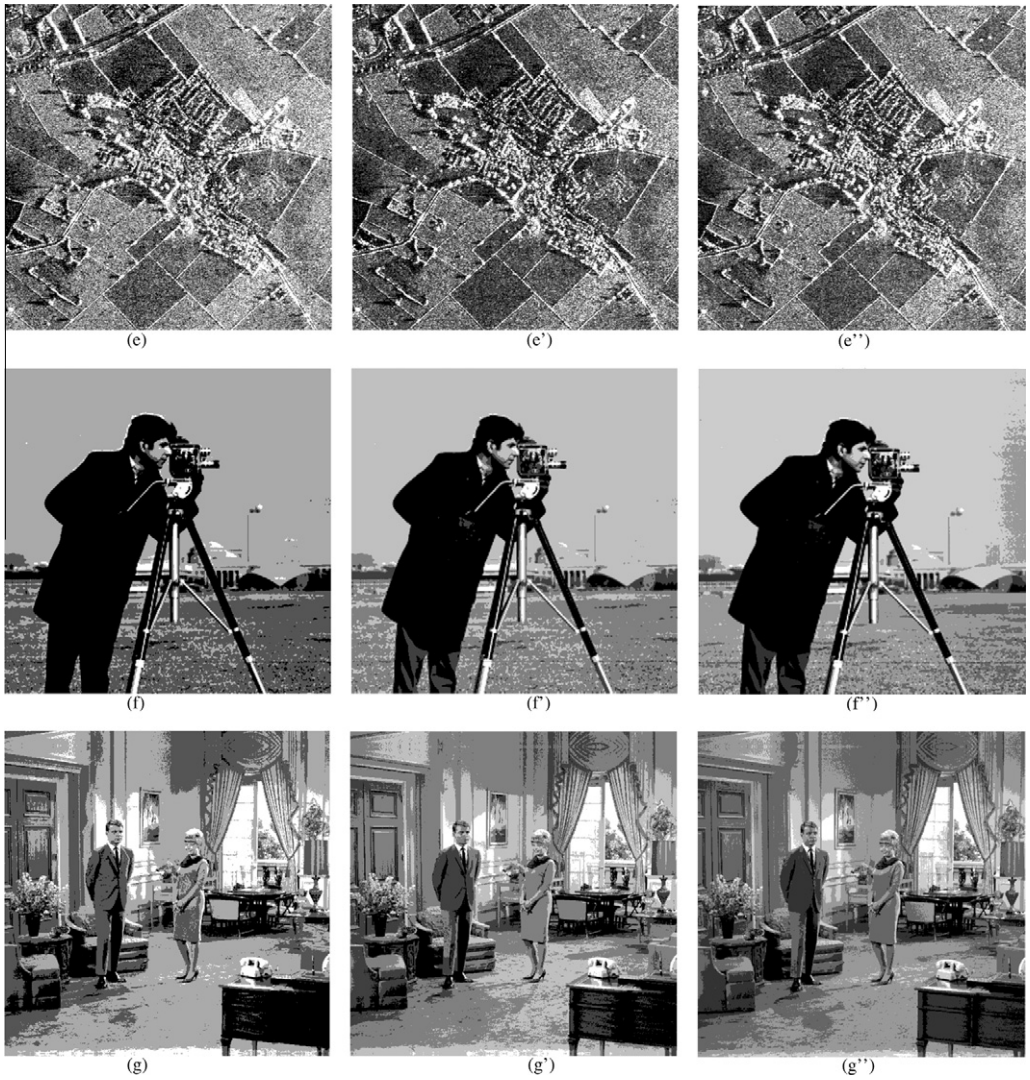

Fig. 2 (*continued*)

when these methods are extended to multilevel thresholding, the computation time grows exponentially with the number of thresholds. It would limit the multilevel thresholding applications. To overcome the above problem, this paper proposes the Kapur and Otsu based BF algorithm for solving multilevel thresholding problem. The aim of this proposed method is to maximize the Kapur's and Otsu's objective function using Eqs. (2) and (3).

3. Bacterial foraging optimization

3.1. Bacterial foraging

Natural selection has a tendency to eliminate animals having poor foraging strategies and favor the ones with successful foraging strategies to propagate their genes as these are more likely to reach a successful reproduction. Poor foraging strategies are either completely eliminated or transferred into good ones after many generations are produced. This evolutionary process of foraging inspired the researchers to utilize it as an optimization tool. The *E. coli* bacteria present in our intestines also practice a foraging strategy. The control system of these bacteria governing their foraging process can be subdivided into four actions, which are chemotaxis, swarming, reproduction and elimination-dispersal.

3.1.1. Chemotaxis

This process is achieved by swimming and tumbling via flagella. Depending upon the rotation of flagella in each bacterium, it decides whether it should move in a predefined direction (swimming) or altogether in different directions (tumbling) in the entire lifetime.

3.1.2. Swarming

During the process of reaching toward the best food location, it is always desired that the bacterium which has searched the optimum path should try to produce an attraction signal to other bacteria, so that they swarm together to reach the desired location. In this process, the bacteria congregate into groups and hence move as concentric patterns of groups with high bacterial density.

3.1.3. Reproduction

The least healthy bacteria die and the other healthiest bacteria each split into two bacteria, which are placed in the same location. This makes the population of bacteria constant.

3.1.4. Elimination and dispersal

In the local environment of the bacteria, the lives of a bacteria population may change either gradually (e.g., via consumption of nutrients) or suddenly due to some other influence. All the bacteria

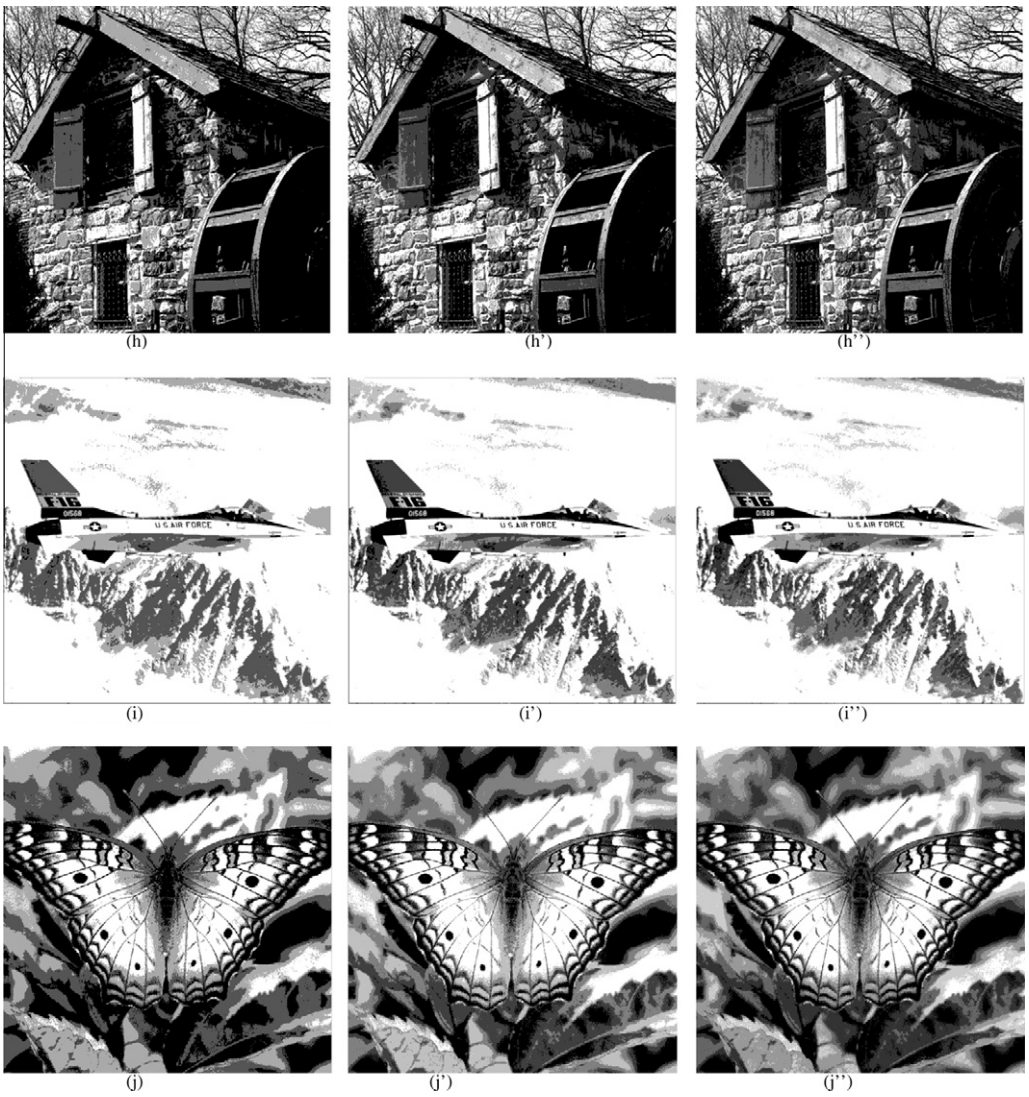

Fig. 2 (*continued*)

in a local region may be killed or a group may be dispersed into a new location in the environment. They have the effect of possibly destroying the chemotaxis progress, but they also have the effect of assisting in chemotaxis, since dispersal may place bacteria near good food sources.

3.2. Bacterial foraging algorithm

The algorithm of the proposed BF is as follows:

Step 1. *Initialize parameters P, s, N_s, N_{re}, N_{ed}, p_{ed}, $C(i)$ ($i = 1,2,\ldots,s$) and X^i. Also initialize all the counter values to zero.*

Step 2. *Elimination-dispersal loop: $l = l + 1$.*

Step 3. *Reproduction loop: $k = k + 1$.*

Step 4. *Chemotaxis loop: $j = j + 1$.*

Step 4.1. *For $i = 1,2,\ldots,s$ take a chemotatic step for bacterium i as follows:*

Step 4.2. *Compute the value fitness function $J(i,j,k,l)$. Let, $J(i,j,k,l) = J(i,j,k,l) + J_{cc}(X^i(j,k,l), P(j,k,l))$. (i.e., add on the cell-to-cell attract-repellant profile to simulate the swarming behavior) where, J_{cc} is given by*

$$J_{cc} = \sum_{i=1}^{s}\left[-d_{\text{attract}}\exp\left(-\omega_{\text{attract}}\sum_{m=1}^{p}(X_g - X^i)^2\right)\right] + \sum_{i=1}^{s}\left[h_{\text{repellant}}\exp\left(-\omega_{\text{repellant}}\sum_{m=1}^{p}(X_g - X^i)^2\right)\right].$$

Step 4.3. *Let $J_{last} = J(i,j,k,l)$ to save this value since we may find a better cost via run.*

Step 4.4. *Tumble: Generate a random vector $\Delta(i)$ with each element $\Delta_m(i)$, $m = 1,2,\ldots,P$, a random number on $[-1,1]$.*

Step 4.5. *Move: Let*

$$X^i(j+1,k,l) = X^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}.$$

This results in a step of size $C(i)$ in the direction of the tumble for bacterium i.

Step 4.6. *Compute $J(i,j+1,k,l)$ and let*

$$J(i,j+1,k,l) = J(i,j,k,l) + J_{cc}(X^i(j+1,k,l), P(j+1,k,l)).$$

Step 4.7. *Swim:*

Table 2
Comparison of objective values and their optimal threshold values obtained by Kapur based evolutionary algorithms.

Test images	m	Objective values			Optimal threshold values		
		BF	PSO	GA	BF	PSO	GA
LENA	2	12.3470	12.3459	12.3344	97,164	99,165	104,167
	3	15.2206	15.1336	14.9956	88,142,188	86,151,180	72,151,180
	4	17.9333	17.8388	17.0892	74,114,149,184	92,129,162,191	57,110,178,184
	5	20.6099	20.4427	19.5492	64,95,128,163,194	74,115,145,170,197	96,112,151,186,198
PEPPER	2	12.5191	12.5168	12.5133	79,149	79,146	82,146
	3	15.3998	15.0939	14.7122	69,100,155	104,141,180	108,127,186
	4	18.2697	18.0974	17.6959	63,109,144,178	57,110,162,199	72,102,172,204
	5	20.9999	20.7338	20.0691	54,89,131,164,197	70,116,138,166,200	77,107,124,178,209
BABOON	2	12.2164	12.2134	12.1847	81,144	76,144	93,152
	3	15.2114	15.0088	14.7457	53,112,150	72,130,181	64,151,181
	4	17.9992	17.5743	16.9356	39,90,131,168	65,121,153,180	90,106,152,188
	5	20.7200	20.2245	19.6622	38,79,113,148,180	73,110,142,166,192	96,126,150,172,197
HUNTER	2	12.3733	12.3708	12.3496	85,179	83,179	75,178
	3	15.5533	15.1286	14.8381	57,104,175	85,128,166	70,148,167
	4	18.3819	18.0401	17.3189	50,98,139,180	74,131,174,200	64,100,189,200
	5	21.2565	20.5339	19.5635	49,93,137,179,222	90,120,164,190,219	87,96,128,196,213
MAP	2	4.9789	4.9789	4.9610	95,181	97,181	84,174
	3	5.5514	5.5030	5.1351	78,125,189	74,140,181	62,94,156
	4	5.8928	5.6903	5.0740	73,132,174,206	92,128,152,207	96,113,186,218
	5	6.0666	5.9165	5.4302	64,106,135,170,210	66,109,121,150,195	85,114,159,192,211
CAMERAMAN	2	12.2646	12.2595	11.9414	116,196	115,196	76,195
	3	15.2507	15.2110	14.8278	95,139,193	96,138,191	111,165,189
	4	18.4066	18.0009	17.1665	42,96,139,200	77,116,151,202	71,80,141,192
	5	21.2111	20.9631	19.7950	42,84,115,150,198	64,95,121,156,198	66,110,169,180,209
LIVINGROOM	2	12.4057	12.4000	12.3923	89,170	86,175	84,171
	3	15.4077	15.2123	14.9700	71,124,173	73,158,187	74,138,160
	4	18.3194	18.1410	17.2063	60,104,147,189	59,124,172,202	74,137,164,175
	5	21.1198	20.6752	19.8410	47,94,134,169,200	72,97,119,158,197	60,120,148,155,200
HOUSE	2	10.8479	10.8321	10.7436	65,144	81,144	91,145
	3	13.2656	13.1006	12.8473	56,110,172	81,116,155	96,134,164
	4	15.6079	15.1027	14.6588	47,87,131,167	75,123,154,193	83,135,170,193
	5	17.6269	17.2517	16.9452	41,73,112,144,176	48,97,139,159,189	81,107,132,157,189
AIRPLANE	2	12.1759	12.1503	12.1153	76,173	80,175	90,176
	3	15.3605	15.2925	14.8059	66,124,186	72,121,191	75,110,199
	4	18.1777	18.0300	17.8923	71,113,149,185	74,129,162,188	87,124,154,187
	5	20.7515	20.3964	19.4465	68,98,131,161,187	81,118,144,167,192	95,121,141,151,196
BUTTERFLY	2	10.4749	10.4743	10.4707	97,144	95,141	93,142
	3	12.7546	12.3130	11.6280	75,109,154	63,126,172	96,103,167
	4	14.8777	14.2317	13.3144	73,97,127,157	71,113,162,184	111,149,155,173
	5	16.8282	16.3374	15.7566	74,97,120,144,167	92,116,142,157,182	75,105,140,179,198

Fig. 3. Thresholded images obtained by Otsu-BF method (a)–(j) represents 3-level thresholding, (a′)–(j′) represents 4-level thresholding, (a″)–(j″) represents 5-level thresholding.

Step 4.7.1. *Set m = 0 (counter for swim length).*

Step 4.7.2. *While $m < N_s$ (if have not climbed down too long).*

Step 4.7.2.1. *Increment m = m + 1.*

Step 4.7.2.2. *If $J(i,j+1,k,l) < J_{last}$ (if doing better), let $J_{last} = J(i,j+1,k,l)$ and let*

$$X^i(j+1,k,l) = X^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}.$$

And use this $X^i(j+1,k,l)$ to compute the new $J(i,j+1,k,l)$ as we did in Step 4.6.

Step 4.7.2.3. *Else, set $m = N_s$.*

Step 4.7.3. *Go to next bacterium (i + 1) if $i \neq s$. That is go to Step 4.2 to process the next bacterium.*

Step 5. *If $j < N_c$, go to Step 4. In this case, continue chemotaxis since the life of the bacteria is not over.*

Step 6. *Reproduction:*

Step 6.1. *For the given k and l, and for each $i = 1,2,\ldots,s$, let*

$$J^i_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l)$$

be the health of bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria in order of ascending cost J_{health} (higher cost means lower health).

Step 6.2. *The s_r bacteria with the highest J_{health}values die and the remaining s_r bacteria with the best values split (this process is performed by the copies that are made already, now placed at the same location as their parent).*

Step 7. *If $k < N_{re}$, go to Step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.*

Step 8. *Elimination-dispersal: For $i = 1,2,\ldots,s$ with probability p_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain. If $l < N_{ed}$, then go to Step 2; otherwise, print the results and stop.*

Fig. 3 (*continued*)

4. Application of BF algorithm for multilevel thresholding problem

BF algorithm is employed to search the optimal threshold values for image segmentation. The procedure of the proposed multilevel thresholding method is as follows:

Step 1.
Input the BF parameters and specify the lower and upper boundaries of the threshold values.

Step 2.
Generate the positions of the threshold values randomly for a population of bacteria.

Step 3.
Evaluate the objective value of each bacterium in the population using Eq. (2) or (3).

Step 4.
Modify the positions of the threshold values for all the bacteria using tumbling/swimming process.

Step 5.
Perform reproduction and elimination-dispersal operation.

Step 6.
If the maximum number of chemotatic, reproduction and elimination-dispersal steps are reached, then go to Step 7. Otherwise, go to Step 4.

Step 7.
Output the optimal threshold values corresponding to the overall best bacterium.

5. Results and discussions

The applicability and viability of the aforementioned technique for multilevel thresholding in image segmentation has been tested on ten standard test images. The obtained results are compared with the PSO and GA methods. Ten well-known images namely lena, pepper, baboon, hunter, map, cameraman, living room, house, airplane and butterfly are gathered with their histograms in Fig. 1. All the images are 512×512 size except the pepper image which is of size 256×256. The parameters of the BF algorithm are given in Table 1.

Fig. 3 (*continued*)

5.1. Case studies

In order to verify the efficiency and effectiveness of the proposed BF algorithm, the two case studies, where the Kapur based and Otsu based objective functions are examined.

5.1.1. Case study 1

In this section, the proposed algorithm is applied on the entropy based Kapur's objective function and compared with the PSO and GA algorithms. Table 2 shows the number of thresholds, optimal threshold values and their corresponding objective values obtained by BF, PSO and GA methods. As seen in Table 2, the proposed Kapur-BF always provides better results than the PSO and GA methods. A detailed qualitative validation for the standard test images are presented in Fig. 2, followed by quantitative comparisons in Table 2. From the Fig. 2, it can be seen that the quality of the segmentation for all the test images is better when the number of thresholds m is 5 than when it is assumed as 3 and 4.

Table 3
Comparison of objective values and their optimal threshold values obtained by Otsu based evolutionary algorithms.

Test images	m	Objective values			Optimal threshold values		
		BF	PSO	GA	BF	PSO	GA
LENA	2	1961.5556	1961.4149	1960.9603	92,151	94,152	91,149
	3	2128.0706	2127.7771	2126.4107	79,125,170	79,127,170	80,124,173
	4	2189.0267	2180.6868	2173.7148	76,117,151,182	78,112,134,175	80,126,159,185
	5	2215.6092	2212.5555	2196.2745	66,92,122,149,183	79,110,140,167,188	80,116,146,179,213
PEPPER	2	2474.8090	2469.5788	2457.1517	73,137	76,144	84,144
	3	2625.3627	2623.2739	2614.0841	63,125,174	72,124,171	65,116,175
	4	2697.7838	2695.8867	2682.8391	54,89,128,171	57,92,130,172	62,108,142,177
	5	2735.6447	2733.5097	2725.8750	47,86,123,158,183	56,84,115,150,179	52,90,128,166,191
BABOON	2	1548.0125	1547.9977	1547.6588	98,150	96,149	98,151
	3	1637.0079	1635.3623	1633.5220	84,126,159	85,126,166	86,125,155
	4	1690.7220	1684.3363	1677.7052	77,109,139,169	79,105,140,174	82,122,146,173
	5	1716.7283	1712.9582	1699.3909	70,99,127,154,177	74,104,134,161,180	73,106,140,167,199
HUNTER	2	3064.1188	3064.0688	3064.0156	51,117	52,116	51,115
	3	3213.4460	3212.0585	3211.7947	36,86,135	39,86,135	36,89,133
	4	3266.3504	3257.1767	3231.1313	31,80,120,152	36,84,130,157	39,93,142,163
	5	3291.1339	3276.3173	3244.7387	31,73,109,141,178	37,85,125,154,177	39,94,130,169,204
MAP	2	2340.3950	2340.3950	2252.3864	109,176	113,177	81,173
	3	2529.9348	2526.3034	2503.7932	98,146,189	81,145,197	83,132,181
	4	2621.1476	2618.4894	2617.9534	88,134,173,222	92,133,162,206	90,110,158,204
	5	2668.0699	2665.4116	2660.8599	80,109,135,165,224	79,116,139,162,204	68,106,138,170,214
CAMERAMAN	2	3609.4995	3609.3703	3609.0761	70,143	71,143	72,145
	3	3682.5693	3677.1783	3643.2153	61,118,155	71,134,166	71,143,196
	4	3737.1201	3722.6447	3710.7311	48,104,142,170	65,121,147,172	59,119,155,203
	5	3769.2239	3764.9571	3755.5529	40,86,125,151,174	45,78,121,146,172	51,106,141,167,194
LIVINGROOM	2	1627.8240	1627.7966	1627.0537	87,146	88,145	89,155
	3	1759.8457	1757.4664	1748.6885	75,124,164	81,127,165	83,132,174
	4	1826.6284	1822.1136	1816.0692	64,102,134,172	69,110,143,178	71,116,150,182
	5	1869.9966	1865.4766	1858.0959	56,94,125,148,180	56,98,128,156,190	65,104,133,160,189
HOUSE	2	3421.2828	3420.9868	3418.4387	56,129	57,127	56,124
	3	3622.3050	3617.9836	3592.1268	43,102,165	48,104,165	50,119,182
	4	3711.7000	3702.2895	3686.1240	34,82,135,182	40,88,140,194	41,98,149,184
	5	3759.0158	3752.1468	3700.3010	33,80,124,170,212	32,74,129,158,188	48,106,136,169,199
AIRPLANE	2	1837.7517	1837.7222	1837.7144	117,175	117,174	116,175
	3	1910.7434	1905.7664	1844.5642	91,147,190	99,158,193	86,133,204
	4	1954.2480	1953.8872	1950.5919	84,127,169,202	84,125,168,201	71,119,164,200
	5	1978.4335	1977.9742	1973.0894	71,110,138,175,203	60,101,138,177,204	84,124,164,188,204
BUTTERFLY	2	1553.0734	1553.0687	1552.4129	99,151	99,150	100,151
	3	1667.2801	1665.7589	1662.6963	78,117,162	79,119,164	74,115,155
	4	1707.0994	1702.9069	1696.6940	75,105,135,165	80,113,145,177	82,119,154,184
	5	1733.0317	1730.7879	1716.0428	76,104,129,154,180	75,106,129,157,180	77,107,134,171,185

Table 4
Comparison of standard deviation for Kapur based and Otsu based algorithms.

Test images	m	Standard deviation					
		Kapur's method			Otsu's method		
		BF	PSO	GA	BF	PSO	GA
LENA	2	2.9929e−004	0.0033	0.0049	0.0390	0.1423	0.2077
	3	0.0061	0.0390	0.1100	0.0750	0.4155	0.5555
	4	0.0081	0.1810	0.2594	0.7305	2.3601	3.0640
	5	0.0502	0.2181	0.3043	2.7184	4.5341	5.7362
PEPPER	2	4.0856e−004	0.0012	0.0031	0.0111	0.0956	0.1455
	3	0.0324	0.0764	0.1750	0.0605	0.1629	0.2891
	4	0.0465	0.1080	0.2707	0.2468	2.1102	3.9721
	5	0.0181	0.1758	0.3048	0.2711	3.2057	4.9999
BABOON	2	8.8872e−004	0.0077	0.0567	0.0470	0.1040	0.2224
	3	0.0287	0.0816	0.1580	0.3293	0.5720	1.5317
	4	0.0336	0.0853	0.1765	1.0655	2.1501	3.0653
	5	0.1065	0.1899	0.2775	1.6105	3.4447	4.6721
HUNTER	2	0.0033	0.0068	0.0148	0.0266	0.2282	0.3283
	3	0.1155	0.0936	0.1741	0.3257	0.8203	1.8080
	4	0.0055	0.1560	0.2192	2.2894	2.9836	6.3644
	5	0.00287	0.2720	0.3466	3.6102	7.3030	11.1247
MAP	2	5.8135e−005	0.0023	0.0030	0.5059	1.2241	1.8856
	3	0.0143	0.1153	0.1226	0.6284	1.2298	2.1368
	4	0.0604	0.1366	0.1849	1.0409	2.2333	4.5790
	5	0.0447	0.1521	0.1901	1.2536	3.4511	6.3580
CAMERAMAN	2	0.0041	0.1001	0.1270	0.0363	0.0908	0.3812
	3	0.0075	0.1107	0.2136	1.4250	6.3502	9.4711
	4	0.0081	0.2005	0.2857	1.5756	2.4498	4.5059
	5	0.0741	0.2734	0.3528	2.1916	8.9650	11.0079
LIVINGROOM	2	0.0018	0.0022	0.0039	0.075	0.2637	0.5425
	3	0.0089	0.0718	0.1364	0.3475	1.0446	2.4428
	4	0.0871	0.2286	0.3220	0.9089	2.0787	3.0313
	5	0.082	0.2619	0.3805	1.0430	2.2655	4.3189
HOUSE	2	0.0050	0.0224	0.0637	0.5965	0.8001	1.7181
	3	0.0254	0.0805	0.1549	1.7712	3.1018	6.2939
	4	0.087	0.1324	0.2555	2.1780	3.7038	8.2156
	5	0.1079	0.1824	0.2696	2.2820	6.5478	9.9390
AIRPLANE	2	0.0076	0.0106	0.0305	0.8861	1.1731	2.7001
	3	0.0062	0.1248	0.1958	1.6725	2.5107	5.0948
	4	0.0484	0.1424	0.3011	1.9995	3.4728	7.0157
	5	0.0188	0.2760	0.3369	2.0057	4.7571	8.6500
BUTTERFLY	2	0.0014	0.0025	0.0872	0.7817	1.6744	2.3493
	3	0.0118	0.1880	0.2021	1.2321	2.2356	3.4016
	4	0.0166	0.2473	0.2596	2.2632	4.2227	5.2383
	5	0.0887	0.2821	0.3977	2.6021	5.1212	6.2719

5.1.2. Case study 2

The second case study includes the between-class variance based Otsu's method. To utilize the BF algorithm for Otsu's objective function, ten standard test images are considered. Table 3 illustrates the number of thresholds, the optimal threshold values and their corresponding objective values of the Otsu based evolutionary algorithms. From Table 3, it is clear that the objective values of the proposed Otsu-BF algorithm are higher than those obtained by the other algorithms. To visually compare the segmented results of the Otsu-BF method, the image is segmented with various thresholds ($m = 3$, 4 and 5), which is shown in Fig. 3. It is observed that the segmentation is the best when $m = 5$ for all the images.

5.2. Comparison of BF with other algorithms

5.2.1. Solution quality

The objective values obtained out of 50 trials for BF, PSO and GA methods are given in Tables 2 and 3. It can be seen that the objective value produced by BF is higher than the other methods which emphasizes the best solution quality. The dynamic behavior of the three methods is also studied by calculating the mean and standard deviation. The mean μ and standard deviation s are defined as

$$\mu = \frac{\sum_{i=1}^{k} \sigma_i}{k},$$

$$s = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(\sigma_i - \mu)^2},$$

where k is the number of runs for each algorithm ($k = 50$), σ_i is the best objective value obtained by the ith run of the algorithm. The standard deviation values obtained by the various algorithms are given in Table 4. The higher standard deviation shows that the results of the experiment are unstable. It is observed that the BF method records a lower value over the other methods, showing the stability of the proposed BF algorithm. From Table 4, it is also observed that the Kapur-BF method gives lower standard deviation value than the Otsu-BF method.

5.2.2. Computation efficiency

Computation efficiencies of all the methods are compared, based on the average CPU time (in seconds) taken to converge

Table 5
Comparison of CPU time (in seconds) for various methods.

Test images	m	Computation time					
		BF		PSO		GA	
		Kapur	Otsu	Kapur	Otsu	Kapur	Otsu
LENA	2	7.2063	3.2969	7.8594	3.5781	8.5469	3.9688
	3	7.6060	3.8281	8.3594	4.4031	8.8594	5.2969
	4	8.5000	4.2188	9.1719	4.7500	9.5156	5.6094
	5	8.8125	4.7813	9.4063	5.2031	10.1250	5.8938
PEPPER	2	6.5001	2.9889	7.1358	3.4010	8.6492	3.8569
	3	6.8625	3.2157	7.6250	4.3125	9.1056	4.9787
	4	7.4706	3.5875	8.1254	4.6719	9.6406	5.5156
	5	7.8177	3.6094	8.4844	4.8125	9.9688	5.9844
BABOON	2	7.6250	3.2813	8.0016	3.8469	8.3563	4.3969
	3	8.2824	3.7969	8.7188	4.3125	9.3750	4.7969
	4	8.7188	4.2813	9.1084	4.9063	9.6750	5.6094
	5	9.1875	4.8206	9.7813	5.3281	10.1875	6.0109
HUNTER	2	7.3594	3.2344	8.000	3.8438	8.6406	4.4063
	3	8.2813	3.9063	8.7031	4.4844	9.9844	4.8625
	4	8.7344	4.1875	9.0313	4.8125	9.6219	5.3906
	5	9.6250	4.8128	10.1406	5.3031	10.6094	6.1563
MAP	2	6.4375	3.0000	6.8906	3.6094	7.4625	4.2000
	3	6.6750	3.8125	7.1563	4.4219	7.6563	4.9688
	4	7.4313	4.1250	8.1250	4.8750	8.9094	5.5156
	5	7.9688	5.0156	8.3594	5.7500	9.7969	6.4188
CAMERAMAN	2	7.7813	3.0625	8.4844	3.4844	9.2500	3.9531
	3	8.2720	3.6875	9.0625	4.1250	9.7000	4.8125
	4	8.5938	4.2344	9.1250	4.7406	9.9844	5.2500
	5	9.2969	4.6719	10.1094	5.2656	10.9688	6.0025
LIVINGROOM	2	6.7656	2.8688	7.5844	3.3281	8.2156	3.7656
	3	7.4143	3.6094	8.7188	4.0469	9.6250	4.9531
	4	8.6250	4.0469	9.1001	4.5000	9.7656	5.1056
	5	9.3494	5.0030	10.1719	5.7969	10.5631	6.6094
HOUSE	2	7.3531	3.1250	7.9063	3.6252	8.3656	4.4313
	3	7.7558	3.7656	8.2626	4.2969	9.2500	4.9844
	4	8.2500	4.0025	8.8438	4.6094	9.5938	5.3750
	5	8.7191	4.8438	9.6406	5.7344	10.0938	6.6963
AIRPLANE	2	7.3094	3.0250	7.9844	3.4688	8.7188	4.0000
	3	8.2500	3.8531	8.9688	4.5938	10.4844	5.1875
	4	8.5625	4.0125	9.2031	4.7969	9.9531	5.3594
	5	9.0156	4.7500	9.9688	5.0781	10.4031	5.8125
BUTTERFLY	2	7.1719	3.2500	7.7188	3.5313	8.4906	3.9219
	3	7.8906	3.7188	8.5469	4.1875	9.4656	4.9531
	4	8.4688	4.2031	9.0000	4.8281	9.8659	5.5156
	5	8.6563	5.0625	9.3813	5.4594	10.2469	6.1313

the solution which is given in Table 5. From Table 5, it is clear that the proposed BF algorithm converges quickly than the other algorithms, and also the average convergence time increases significantly with the threshold levels. It is also observed that the Otsu-BF algorithm converges faster than the Kapur-BF algorithm.

6. Conclusion

In this paper, the bacterial foraging (BF) algorithm is presented for solving multilevel thresholding problem in image segmentation. In order to verify the efficiency and effectiveness of the proposed BF algorithm, two case studies are investigated in which the Kapur's and Otsu's objective functions are considered. The proposed algorithm has been tested on the various standard test images. The results obtained by this method are verified with those obtained by the PSO and GA methods. The experimental results show that the proposed method outperforms the other methods in terms of solution quality, stability and computation efficiency. The Kapur-BF algorithm is best for its lowest standard deviation, whereas the Otsu-BF algorithm converges quickly. Further research is to be carried out to test the feasibility of the proposed algorithm for various types of image processing applications.

References

Abak, A. T., Baris, U., & Sankur, B. (1997). The performance evaluation of thresholding algorithms for optimal character recognition. In *IEEE proceedings international conference document analysis and recognition, Germany* (pp. 697–700).

Brink, A. D. (1995). Minimum spatial entropy threshold selection. *IEE Proceedings on Vision Image and Signal Processing, 142*, 128–132.

Deng-Yuan, Huang., & Chia-Hung, Wang. (2009). Optimal multi-level thresholding using two-stage Otsu optimization approach. *Pattern Recognition Letters, 30*, 275–284.

Fogel, D. B. (2000). Evolutionary computation: Toward a new philosophy of machine intelligence. *2nd ed.*. Piscataway, NJ: IEEE Press.

Huang, L. K., & Wang, M. J. (1995). Image thresholding by minimizing the measure of fuzziness. *Pattern Recognition, 28*, 41–51.

Kapur, J. N., Sahoo, P. K., & Wong, A. K. C. (1985). A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics and Image Processing, 29*(3), 273–285.

Kittler, J., & Illingworth, J. (1986). Minimum error thresholding. *Pattern Recognition, 19*, 41–47.

Liao, P. S., Chen, T. S., & Chung, P. C. (2001). A fast algorithm for multilevel thresholding. *Journal of Information Sciences and Engineering, 17*(5), 713–727.

Maitra, M., & Chatterjee, A. (2008). A hybrid cooperative-comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. *Expert Systems with Applications, 34*, 1341–1350.

Nikhil, R. Pal., & Sankar, K. Pal. (1989). Entropic thresholding. *IEEE Transactions on Signal Processing, 16*, 97–108.

Otsu, N. (1979). A threshold selection method from gray level histograms. IEEE 481 Transactions on Systems, Man and Cybernetics SMC-9, 62–66.
Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Transactions on Control Systems Magazine, 22*(3), 52–67.
Sahoo, P. K., Soltani, S., & Wong, A. K. C. (1988). A survey of thresholding techniques. *IEEE Transactions on Computer Vision, Graphics and Image Processing, 41*(2), 233–260.
Shu-Kai, S., Fan & Yen, Lin (2007). A multilevel thresholding approach using a hybrid optimal estimation algorithm. *Pattern Recognition, 28*, 662–669.
Synder, W., Bilbro, G., Logenthiran, A., & Rajala, S. (1990). Optimal thresholding – A new approach. *Pattern Recognition Letters, 11*, 803–810.
Tsai, Du-Ming (1995). A fast thresholding selection procedure for multimodel and unimodel histograms. *Pattern Recognition, 16*, 653–666.
Weszka, J., & Rosenfeld, A. (1979). Histogram modification for threshold selection. *IEEE Transactions on Systems, Man and Cybernetics, 9*, 38–52.
Yen, J. C., Chang, F. J., & Chang, S. (1995). A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Process, 4*(3), 370–378.
Yin, Peng-Yeng (1999). A fast scheme for multilevel thresholding using genetic algorithms. *Signal Processing, 72*, 85–95.
Ye, Z., Zheng, Z., Yu, X, & Ning, X. (2006). Automatic threshold selection based on ant colony optimization algorithm. In *International conference on neural networks and brain, Beijing* (pp. 728–732).