BF ALGORITHM BASED IMAGE SEGMENTATION USING MINIMUM CROSS ENTROPY

P.D. Sathya
Assistant Professor
Department of Electronics and Communication Engineering
Annamalai University, Chidambaram – 608002, India
pd.sathya@yahoo.in

Abstract : **Image segmentation is a essential step for many image analysis and preprocessing tasks. In segmentation, minimum cross entropy (MCE) based multilevel thresholding is regarded as an effective improvement over the bi-level method. However, it is very time consuming for real-time applications. In this paper, a fast threshold selection method based on bacterial foraging optimization (BFO) algorithm is proposed to speed up the original MCE threshold method in image segmentation. BFO algorithm is a newly developed memetic meta-heuristic evolutionary algorithm with good global search ability. Experimental results compared with particle swarm optimization (PSO) and genetic algorithm (GA) show that the BFO based thresholding can exactly obtain the global optimal threshold values with significant decrease in the computational time and provide better peak to signal noise ratio (PSNR) value and stability.**

IndexTerms - **multilevel thresholding, image segmentation, minimum cross entropy, bacterial foraging algorithm.**

I. INTRODUCTION

Image segmentation is widely used in a variety of applications such as robot vision, object recognition, geographical imaging and medical imaging. Classically, image segmentation is defined as the partitioning of an image into non-overlapped consistent regions which are homogeneous with respect to some characteristics such as gray value or texture.

Thresholding is one of the most used methods for image segmentation. Many methods define the optimal threshold as the one which maximizes or minimizes an objective function. Its basic objective is to classify the pixels of a given image into two classes: those partitioning to an object and those partitioning to the background.

During the past decade, many research studies have been devoted to the problem of selecting the appropriate threshold value. For the image with clear objects in the background, the bi-level thresholding method can easily divide the object from the background. Sahoo et al. [1] have presented a thorough survey of a variety of thresholding techniques. Among those techniques, global, histogram based algorithms [2] are widely used to determine the threshold, and they can be classified as parametric and non-parametric approaches.

In the parametric approaches [3], the gray level distribution of each class is assumed to have a probability density function. It is usually assumed to be a Gaussian distribution. One attempts to find an estimate of the parameters of the distribution that will best fit the given histogram data in the least squares sense. The result is typically a nonlinear optimization problem that is computationally expensive and time-consuming to find the solution.

In the non-parametric approaches, one is to find the thresholds that separate the gray-level regions of an image in an optimum manner according to some discriminate criteria such as the between-class variance [4], entropy [5], and cross entropy [6]. The non-parametric approaches are computationally efficient and simple to implement, compared to the parametric approaches.

In bi-level thresholding the existing non-parametric methods are robust and computationally fast for time-critical applications. However, the computational complexity of those methods is exponentially increased and the selected thresholds generally become less credible as the number of classes to be separated increases. Moreover, to segment complex images, multilevel thresholding method is required. In multilevel image thresholding, pixels can be classified into many classes, not just foreground and background. To mitigate this problem, many methods have been proposed for multilevel thresholding [7-10].

In [8], the Otsu’s function is modified by a fast recursive algorithm along with a look-up-table for multilevel thresholding. In [9], Lin has proposed a fast thresholding computation using Otsu’s function. Another fast multilevel thresholding technique has been proposed by Yin [10].

Various deterministic methods have been applied to solve multilevel thresholding problem in image segmentation. Several techniques using genetic algorithms (GAs) have also been proposed to solve the multilevel thresholding problem [11], [12]. The particle swarm optimization (PSO) has been applied to the multilevel thresholding for image segmentation [13].

In this study, in order to solve the multilevel thresholding problem in image segmentation more efficiently, BFO algorithm is proposed. The algorithm is based on the foraging (methods for locating, handling and ingesting food) behavior of *E. Coli* bacteria present in the human intestine. It was successfully used to solve various kinds of engineering problems [14-16]. It has been shown that the BFO algorithm offers superior performance in terms of solution quality and convergence speed than the PSO and GA. The effectiveness of the proposed BFO method is demonstrated for various benchmark test images, and is compared with the PSO and GA algorithm in terms of solution quality and evolutionary computing efficiency.

II. MINIMUM CROSS ENTROPY THRESHOLDING PROBLEM FORMULATION

The cross entropy was proposed by Kullback [14]. Let $P = \{p_1, p_2, p_3 ...p_N\}$ and $Q = \{q_1, q_2, q_3 ...q_N\}$ be the two probability distributions on the same set. The cross entropy P and Q is information theoretic distance between the two distributions and it is defined by

$$D(p,q) = \sum_{i=1}^{N} p_i ln \frac{p_i}{q_i}$$

The minimum cross entropy thresholding algorithm selects the thresholds by minimizing the cross entropy between the original image and its thresholded version. Let there be L gray levels in a given image and these gray levels are in the range {0, 1, 2,………,(L-1)}, I be the original image and $h(i) = 0, 1, 2 ... L$ be the corresponding histogram. Then the resulting image, denoted by I_t using t as the thresholded value that is constructed by

$$I_t(x,y) = \begin{cases} \mu(1,t), & I(x, y) < t \\ \mu(t, L+1), & I(x, y) \geq t \end{cases}$$

where,

$$\mu(a,b) = \sum_{i=a}^{b-1} ih(i) \Big/ \sum_{i=a}^{b-1} h(i)$$

The cross entropy for bi-level thresholding is then calculated by:

$$min\{D(t)\} = D_0 + D_1$$

where,

$$D_0 = -\sum_{i=0}^{t-1} ih(i)log\left(\sum_{i=0}^{t-1} ih(i) \Big/ \sum_{i=0}^{t-1} h(i)\right)$$

$$D_1 = -\sum_{i=t}^{L} ih(i)log\left(\sum_{i=t}^{L} ih(i) \Big/ \sum_{i=t}^{L} h(i)\right)$$

This MCE thresholding method has also been extended to multilevel thresholding and can be described as follows: The optimal multilevel thresholding problem can be configured as a m-dimensional optimization problem, for determination of m optimal thresholds for a given image [$t_1, t_2 ...t_m$], where the aim is to minimize the objective function:

$$min\{D(t_0 + t_1 + t_2 .... + t_m)\} = D_0 + D_1 + D_2 .... + D_m \quad (1)$$

where,

$$D_0 = -\sum_{i=0}^{t_1-1} ih(i)log\left(\sum_{i=0}^{t_1-1} ih(i) \Big/ \sum_{i=0}^{t_1-1} h(i)\right)$$

$$D_1 = -\sum_{i=t_1}^{t_2-1} ih(i)log\left(\sum_{i=t_1}^{t_2-1} ih(i) \Big/ \sum_{i=t_1}^{t_2-1} h(i)\right)$$

$$D_2 = -\sum_{i=t_2}^{t_3-1} ih(i)log\left(\sum_{i=t_2}^{t_3-1} ih(i) \Big/ \sum_{i=t_2}^{t_3-1} h(i)\right).....$$

and

$$D_m = -\sum_{i=m}^{L} ih(i)log\left(\sum_{i=m}^{L} ih(i) \Big/ \sum_{i=m}^{L} h(i)\right)$$

The minimum cross entropy thresholding method is very efficient in bi-level thresholding cases. However, its computational time becomes aggravated in the case of multilevel thresholding. To make the multilevel MCE thresholding method more practical in image segmentation, this paper proposes MCE threshold selection based on BF algorithm. The aim of this proposed method is to minimize the MCE thresholding objective function using Eq. (1).

III. BACTERIAL FORAGING OPTIMIZATION ALGORITHM

BFO is an evolutionary optimization technique motivated by the foraging behavior of the *E. Coli* bacteria. The biological aspects of the bacterial foraging strategies and their motile behavior as well as their decision-making mechanisms can be found in [17]. As a heuristic method, BFO is designed to tackle non-gradient optimization problems and to handle complex and non-differentiable objective functions. Searching the hyperspace is performed through three main operations, namely chemotaxis, reproduction and elimination dispersal activities [17].

The chemotaxis process is performed through swimming and tumbling. The bacterium spends its lifetime alternating between these two modes of motion. In the BFO, a tumble is represented by a unit length in a random direction, $\varphi(i)$, which specifies the direction of movement after a tumble. The size of the step taken in the random direction is represented by the constant run-length unit, $C(i)$.

For a population of bacteria, the location of the *i*th bacterium at the *j*th chemotactic step, *k*th reproduction step and the *l*th elimination/dispersal event is represented by $X^i(j,k,l) \in \Re^p$. At this location, the cost function is denoted by *J (i, j, k, l)*, which is also known as the nutrient function. After a tumble, the location of the *i*th bacterium is represented as

$$X^i(j+1,k,l) = X^i(j,k,l) + C(i)\varphi(i) \qquad (2)$$

When at $X^i(j+1,k,l)$ the cost function *J (i, j+1, k, l)* is better (lower) that *J (i, j, k, l*), another step of size *C (i)* in the same direction is taken. This swimming operation is repeated as long as a lower cost is obtained until a maximum preset number of steps, N_s is reached.

The cost function of each bacterium in the population is affected by a kind of swarming that is performed by the cell-to-cell signaling released by the bacteria group to form swarm patterns. This swarming is expressed as follows:

$$J_{cc} = \sum_{i=1}^{S} [-d_{attract} \exp(-\omega_{attract} \sum_{m=1}^{p} (X_g - X_m^i)^2)] + \sum_{i=1}^{S} [h_{repellant} \exp(-\omega_{repellant} \sum_{m=1}^{p} (X_g - X_m^i)^2)]$$

where $d_{attract}$, $\omega_{attract}$, $h_{repellant}$ and $\omega_{repellant}$ are coefficients that represent the characteristics of the arrtactant and repellant signals released by the cell and X_m^i is the *m*th component of *i*th bacterium position X^i. *P(j, k, l)* is the position of each member of the position of the *S* bacteria and is defined as

$$P(j,k,l) = \{X^i(j,k,l) \quad \text{for } i = 1,2...S\}$$

where, *S* is the size of the bacteria population.

The cell-to-cell signaling effect is added to the cost function as follows:

$$J(i,j,k,l) + J_{cc}(X,P) \qquad (3)$$

A reproduction process is performed after taking a maximum number of chemotactic steps, N_c. The population is halved so that the least healthy half dies and each bacterium in the other healthiest one splits into two bacteria that take the same position.

$$S_r = \frac{S}{2}$$

After N_{re} reproduction steps, an elimination/dispersal event takes place for N_{ed} number of executions. In this operation, each bacterium could be moved to explore other parts of the search space. The probability for each bacterium to experience the elimination/dispersal event is determined by a predefined fraction p_{ed}.

The algorithm of the proposed BFO technique is as follows:

Step 1: Initialization of the following parameters:

P: dimension of the search space;
S: the number of bacteria in the population;
N_c: number of chemotactic steps;
N_s: the length of the swim when it is on a gradient;
N_{re}: the number of reproduction steps
N_{ed}: the number of elimination/dispersal events;
p_{ed}: the probability that each bacterium will be eliminated/dispersed;
C(i): initial run-length unit;
X^i : the initial random location of each bacterium;

Step 2: Elimination/dispersal loop, $l = l + 1$
Step 3: Reproduction loop, $k = k + 1$
Step 4: Chemotaxis loop, $j = j + 1$

For $i = 1, 2,..... S$, execute the chemotactic step for each bacterium as follows:

- Evaluate the cost function *J (i, j, k, l)* using (3).
- Let $J_{last} = J(i, j, k, l)$ so that a lower cost could be found.
- Tumble: generate a random vector $\Delta(i) R^p$ and $\Delta_m(i)$, $m = 1, 2, ... p$ is a random number in the range [-1, 1].
- Compute *φ(i)*
- Move using (2)
- Compute *J (i, j+1, k, l)* and use (3) to compute $J_{cc}(X, P(j+1, k, l))$ then use to find the new *J (i, j+1, k, l)*.
- Swim: let $m = 0$ (counter for swim length)
  While $m < N_s$ (no climbing down too long)
  Let $m = m + 1$
  If $J(i, j+1, k, l) < J_{last}$ let $J_{last} = J(i, j+1, k, l)$ then take another step in the same direction and compute the new *J (i, j+1, k, l)*.
- Go to next bacterium ($i + 1$) if $i \neq S$.

Step 5: If $j < N_c$ go to step 4 ($j = j + 1$).
Step 6: Reproduction: For the given *k* and *l*, evaluate the

health of each bacterium *i* as follows

$$J^{i}_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \qquad (5)$$

The health of the bacterium *i* measures how many nutrient it got over its lifetime.

- Sort bacteria according to their J^{i}_{health} in ascending order.
- The bacteria with the highest J^{i}_{health} values, computed by Eq. (5) die while the other S_r with the lowest values split and take same location of their parents.

Step 7: If $k < N_{re}$, go to step 3 ($k = k + 1$)
Step 8: Elimination/dispersal: With probability p_{ed}, randomly eliminate and dispersal each bacterium *i*, keeping the size of the population constant.

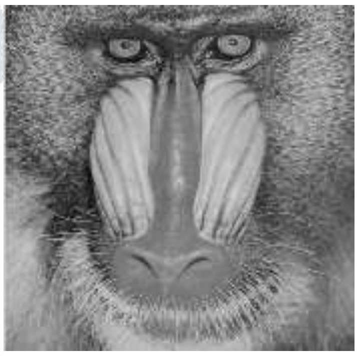
Step 9: If $l < N_{ed}$, go to step 2 ($l = l + 1$), otherwise end.

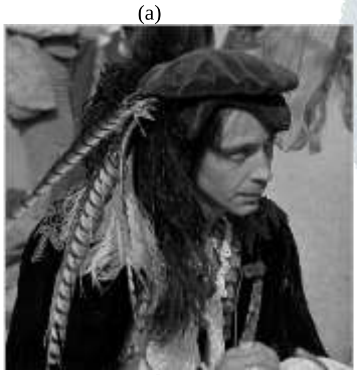
IV. EXPERIMENTAL RESULTS AND EVALUATION

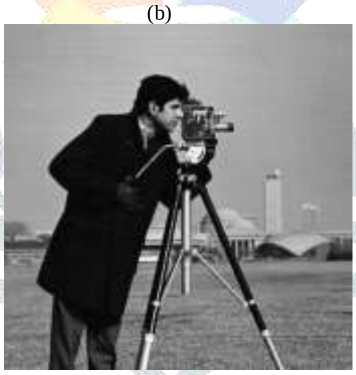
In order to evaluate the effectiveness of the proposed BFO algorithm, several images shown in Fig. 1 are tested. These images are of 512×512 in size, with gray levels L = 256. Their corresponding gray level histograms are shown in Fig. 2. In addition to the proposed algorithm, two other methods, which are PSO and GA, are used for comparison. The BFO parameters used for the simulation are given in Appendix.

(a)

(b)


(c)


(d)


(e)

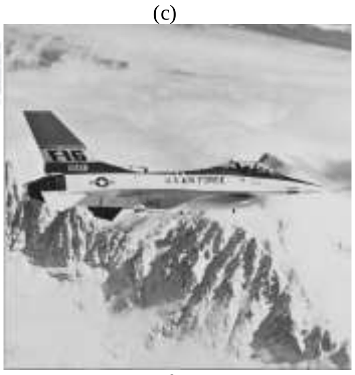
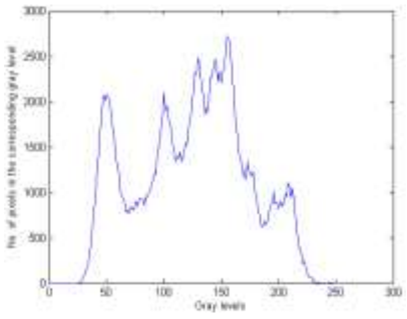
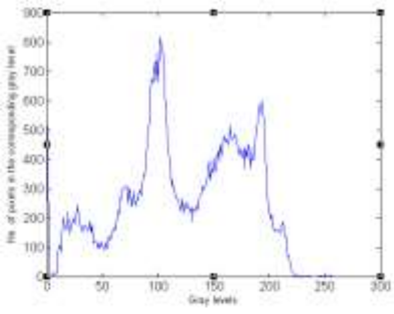

(f)

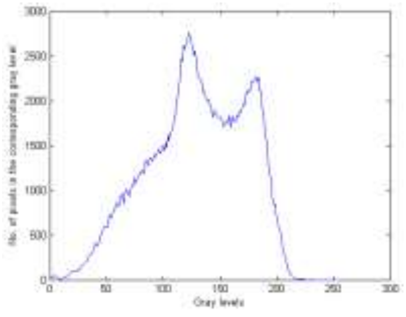
Fig. 1. Test Images [(a) Lena, (b) Pepper, (c) Baboon, (d) Hunter, (e) Cameraman, (f) Airplane]



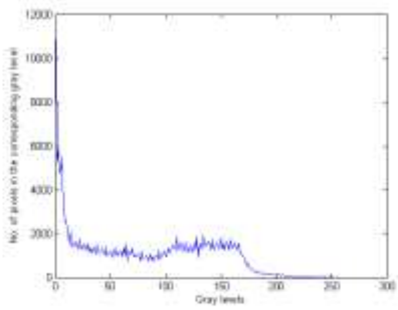
(a')



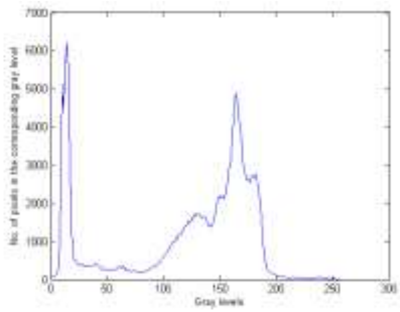
(b')



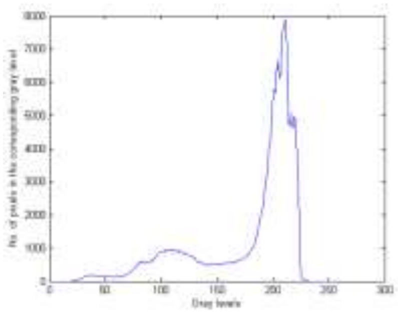
(c')



(d')

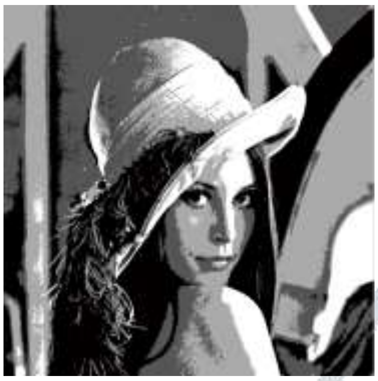


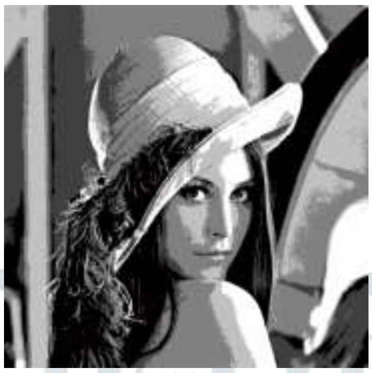
(e')

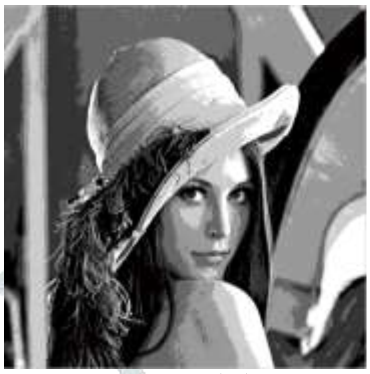


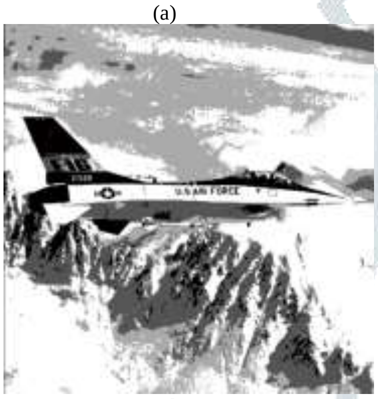
(f')

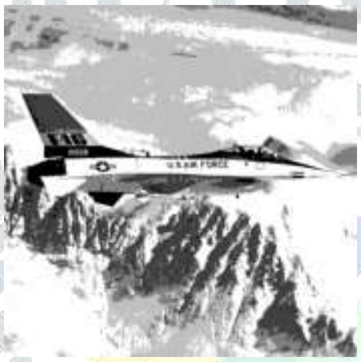
Fig. 2. Histogram of test images [(a') Lena, (b') Pepper, (c') Baboon, (d') Hunter, (e') Cameraman, (f') Airplane]


(a)


(a')


(a'')


(b)


(b')

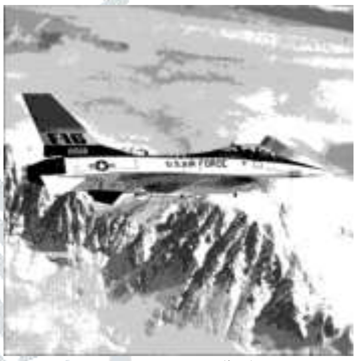

(b'')

Fig. 3. Segmented test images of lena and cameraman [(a) & (b) for 3-level thresholding, (a') & (b') for 4-level thresholding and (a'') & (b'') for 5-level thresholding]

TABLE I. COMPARISON OF OBJECTIVE VALUES AND THEIR OPTIMAL THRESHOLD VALUES OBTAINED BY MCE BASED EVOLUTIONARY ALGORITHMS

Test Images	**m**	**Objective values × (10^9)**			**Optimal threshold values**		
		BF	PSO	GA	BF	PSO	GA
LENA	2	10.7132	10.7132	10.7132	110,165	110,165	110,165
	3	7.9376	8.1331	8.4895	94,138,178	87,134,174	94,138,178
	4	6.4852	6.7606	7.1689	76,116,152,187	64,113,145,181	67,107,145,183
	5	5.4113	5.8280	6.2674	63,107,136,162,197	65,93,135,168,195	61,103,124,169,194
PEPPER	2	2.8420	2.8420	2.8420	110,174	110,174	110,174
	3	2.0504	2.0518	2.1164	95,144,186	92,144,184	87,131,179
	4	1.6265	1.7093	1.8113	77,116,161,191	74,115,151,189	75,101,148,196
	5	1.3693	1.4226	1.5008	60,108,147,175,197	63,96,131,169,202	60,93,129,168,206
BABOON	2	10.7194	10.7194	10.7194	113,166	113,166	113,166
	3	7.8400	7.9080	8.0249	99,140,180	96,135,175	90,132,173
	4	6.2505	6.8674	7.2793	79,120,150,182	71,109,142,180	78,106,146,194
	5	5.2524	6.3128	6.7091	63,103,132,163,190	62,98,121,161,202	61,90,124,153,203
HUNTER	2	6.9948	6.9948	6.9948	106,163	106,163	106,163
	3	4.9507	4.9598	4.9675	88,133,171	85,132,170	95,137,171
	4	3.9094	4.1335	4.3752	62,109,142,174	64,99,135,171	77,105,157,182
	5	3.3338	3.5576	3.8779	56,98,130,157,190	62,106,136,166,210	66,94,116,158,201
CAMERAMAN	2	10.2203	10.2203	10.2203	134,181	134,181	134,181
	3	6.9625	7.5112	7.6798	100,154,185	85,147,185	81,146,182
	4	5.9721	6.3600	7.0503	48,118,159,187	73,105,155,187	70,91,151,185
	5	4.8906	5.1134	5.9511	58,96,145,167,190	55,98,127,160,190	61,91,130,166,193
AIRPLANE	2	11.6252	11.6252	11.6252	140,198	140,198	140,198
	3	9.5151	9.9986	1.0273	89,160,202	91,143,197	79,142,198
	4	8.1856	8.5374	8.7379	67,118,170,204	71,102,170,204	61,99,169,204
	5	6.3994	6.6450	7.2280	64,104,143,190,209	60,102,134,188,209	63,86,133,181,207

TABLE II. THE PSNR MEASURE BY FOUR MULTILEVEL THRESHOLDING METHODS

Test Images	m	PSNR (db)			Standard Deviation		
		BF	PSO	GA	BF	PSO	GA
LENNA	2	15.2352	15.2352	15.2352	0.0000	0.0000	0.0000
	3	17.5483	17.4893	17.3556	2.4828e+007	1.4028e+008	2.5106e+008
	4	19.3910	19.0003	18.6737	2.9046e+007	1.5260e+008	2.8948e+008
	5	21.5078	21.1539	20.5928	3.1495e+007	2.0828e+008	4.7978e+008
PEPPER	2	14.5835	14.5835	14.5835	0.0000	0.0000	0.0000
	3	16.2467	16.1968	16.1593	1.4300e+006	1.4146e+007	1.0806e+007
	4	18.5793	18.4858	18.1006	6.6532e+006	2.8183e+007	4.3821e+007
	5	20.5222	20.0197	19.7713	8.6984e+006	3.2069e+007	1.0156e+008
BABOON	2	14.5746	14.5746	14.5746	0.0000	0.0000	0.0000
	3	17.2967	16.8421	16.4741	4.4147e+006	2.8496e+007	8.2234e+007
	4	19.0635	18.3975	18.0022	1.0406e+007	5.0558e+007	1.1282e+008
	5	21.3085	21.2791	21.1005	3.3228e+007	9.1601e+007	3.2760e+008
HUNTER	2	11.8996	11.8996	11.8996	0.0000	0.0000	0.0000
	3	14.3842	14.1020	13.9110	6.4358e+005	4.2094e+006	8.6320e+006
	4	17.1689	16.8850	16.0069	6.5486e+006	2.9516e+007	5.4299e+007
	5	18.3772	17.5551	16.9753	1.2305e+007	5.2389e+007	8.7779e+007
CAMERAMAN	2	12.1338	12.1338	12.1338	0.0000	0.0000	0.0000
	3	14.8679	14.7588	14.4201	4.4697e+005	6.4922e+006	3.1927e+007
	4	18.8094	17.4608	17.3764	2.8474e+007	5.3860e+007	8.7752e+007
	5	20.0712	19.5395	18.9669	3.3605e+007	5.2067e+008	8.5638e+008
AIRPLANE	2	15.6646	15.6646	15.6646	0.0000	0.0000	0.0000
	3	16.7233	16.2762	16.0567	3.9914e+007	6.7746e+007	1.3900e+008
	4	17.9986	17.4232	17.3138	4.2097e+007	3.8022e+008	7.0890e+008
	5	19.8918	19.2434	18.8074	5.7113e+007	8.9570e+008	1.8200e+009

The threshold values determined by the above methods are presented in Table I. The thresholding results of the testing images obtained by the proposed method are shown in Fig. 3. The experimental results indicate that the proposed method based on BFO algorithm seems to have satisfactory thresholding performance.

One important concern in image thresholding is the effectiveness in segmentation. According to the thresholding results, the proposed method has demonstrated satisfactory results. However, it is somewhat difficult to compare quantitatively the performance of global thresholding results. Two common performance evaluation criteria, the Peak to Signal Noise Ratio (PSNR) and standard deviation measure, are employed to evaluate the thresholding methods.

Table II shows the PSNR and standard deviation values obtained by the BFO, PSO and GA methods. The higher value of PSNR means that the quality of the thresholded image is better. For all the images, the performance of the proposed method is better than the PSO and GA, since their PSNR measure is higher.

As all the optimization algorithms are stochastic and random searching one, the results of experiments are not absolutely the same in each run of the algorithm. Hence, it is necessary to analyze the stability of all the algorithms. This comparison is utilized to find which algorithm is more stable than others. Table II also summarizes the standard deviation value obtained by all the algorithms using the testing images shown in Fig. 1. From the results, the standard deviation value of BFO algorithm is lesser than the PSO and GA which illustrates the stability of the proposed BFO algorithm.

TABLE IV. THE CPU TIME TAKEN BY FOUR MULTILEVEL THRESHOLDING METHODS

Test Images	m	CPU time (Seconds)		
		BF	PSO	GA
LENNA	2	8.1875	9.5781	10.2031
	3	8.3438	9.9219	10.8281
	4	9.4688	10.3438	11.1406
	5	10.2500	10.9401	11.8871
PEPPER	2	7.7581	8.7969	9.4375
	3	8.4375	9.6094	10.0000
	4	9.7581	10.4375	10.9435
	5	10.7085	11.0625	12.0469
BABOON	2	9.2031	10.0781	10.9688
	3	10.0781	11.1563	11.9375
	4	10.9844	11.9563	12.7344
	5	11.4063	12.4688	13.0137
HUNTER	2	8.9803	9.8906	10.5431
	3	9.2500	10.9063	11.7344
	4	10.5938	11.7031	12.5781
	5	11.1406	12.8125	13.9354
CAMERAMAN	2	8.9531	10.0625	10.9375
	3	9.6719	11.1344	11.9603
	4	11.2381	12.6769	13.1250
	5	11.8438	13.0000	13.8657
AIRPLANE	2	8.8444	9.4781	10.2969
	3	9.7500	10.9699	11.7865
	4	10.6094	11.8751	12.6094
	5	11.6875	12.8594	14.2813

In the view point of the computation time, the proposed method is faster than the PSO and GA. It is shown in Table IV. Further, the CPU time increases with the number of thresholds.

V. CONCLUSION

In this paper, bacterial foraging optimization (BFO) algorithm is presented for multilevel thresholding in image segmentation. The proposed method uses minimum cross entropy (MCE) as objective function, which is minimized by the BFO algorithm. The effectiveness of the new method is illustrated by using the test images of having various histograms. The Experiments demonstrate that this method provides superior thresholding results to existing thresholding methods such as PSO and GA for various images. Moreover, the segmentation results of the proposed BFO method have demonstrated the good performance in five level thresholding than the other levels. The obtained results also show the robustness of the method, and its non independence towards the kind of the image to be segmented.

APPENDIX PARAMETERS USED FOR BF ALGORITHM

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

REFERENCES

[1] P. K. Sahoo, S. Soltani, and A. K. C. Wong, "A survey of thresholding techniques," *Computer Vision, Graphics and Image Processing*, vol. 41(2), pp. 233-260, 1988.

[2] C.A. Glasbey, "An analysis of histogram based thresholding algorithms," *CVGIP: Graphical Models and Image Processing*, Vol. 55, pp. 532-537, 1993.

[3] J.S. Weszka, "A survey of threshold selection techniques," *Computer Vision Graphics Image Processing*, Vol. 7, 259-265, 1979.

[4] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transaction on Systems, Man and Cybernetics*, SMC-9(1), pp. 62-66, 1979.

[5] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision, Graphics and Image Processing*, vol. 29, pp. 273-285, 1985.

[6] C.H. Li and C.K. Lee, "Minimum cross entropy thresholding," *Pattern Recognition*, Vol. 26(4), pp. 617-625, 1993.

[7] P. Y. Yin and L. H. Chen, "A fast iterative scheme for multilevel thresholding methods", *Signal Processing*, vol.60, pp. 305-313, 1997.

[8] P. S. T. Liao, S. Chen, and P. C. Chung, "A fast algorithm for multilevel thresholding", *Journal of Information science and Engineering*, vol. 17, pp. 713-727, 2001.

[9] K. C. Lin, "Fast image thresholding by finding zero(s) of the first derivative of between class variance", *Machine Vision and Applications*, vol. 13, pp. 254-262, 2003.

[10] Peng-Yeng Yin and Ling-Hwei Chen, "A fast iterative scheme for multilevel thresholding methods," *Signal Processing*, vol. 60(3), pp. 305-313, 1997.

[11] P. Y. Yin, "A fast scheme for optimal thresholding using genetic algorithms", *Signal processing*, vol. 72, pp 85−95, 1999.

[12] C. C. Lai, D. C. Tseng, "A hybrid approach using Gaussian smoothing and genetic algorithm for multilevel thresholding", *International Journal of Hybrid Intelligent Systems*, vol. 1(3), pp. 143-152, 2004.

[13] M. Maitra, and A. Chatterjee, "A hybrid cooperative-comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding," *Expert Systems with Applications*, Vol. 34, 1341-1350, 2008.

[14] S. Mishra, "A hybrid least square-fuzzy bacteria foraging strategy for harmonic estimation," *IEEE Trans. Evol. Comput.*, 9(1), pp. 61-73, 2005.

[15] M. Tripathy, and S. Mishra, "Bacterial foraging based solution to optimize both real power and voltage stability limit," *IEEE Trans. Power Syst.*, 22(1), pp. 240-248, 2007.

[16] W. Lin, and P.X. Liu, "Hammerstein model identification based on bacterial foraging," *IEE Electronics Letters*, 42(23), pp. 1332-1334, 2006.

[17] K.M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Transactions on Control Systems Magazine*, Vol. 22(3), 52–67, 2002.