International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-4)

Research Article | **April 2015**

Image Edge Detection Using Self Adaptable Bacterial Foraging Optimization Algorithm a Review

[1]Kanica, [2]Beant Kaur
[1]Dept. of ECE, Punjabi University, Patiala, Punjab, India
[2]A.P , Deptt. of ECE, Punjabi University, Patiala, Punjab, India

Abstract-

Edge detection is an important field in image processing. It is a process that detects the presence and location of edges constituted by sharp changes in intensity of the image. It can be used in many applications such as segmentation, feature extraction, and identification of objects in a scene. Introduction to bacterial foraging optimization algorithm is given in this paper which is self adaptable which would be used to improve in the edge detection parameters. Different edge detectors work under different conditions. This paper presents the review of different edge detection methods.

Keywords- Edge detection, bacterial foraging algorithm, edge detection operators.

I. INTRODUCTION

The Edge detection of digital image is the very important basic of image segmentation, area identification, feature extraction and other image analysis field. The goal of edge detection is to produce a line drawing of a scene from an image of that scene. The important features can be extracted from the edges of an image (e.g., corners, lines, curves). These features are used by higher-level computer vision algorithms (e.g., recognition). Various physical events cause intensity changes which are :

- **Geometric events**
  - Object boundary (discontinuity in depth and/or surface color and texture)
  - Surface boundary (discontinuity in surface orientation and/or surface color and texture)

- **Non-geometric events**
  - Specularity (direct reflection of light, such as a mirror)
  - Shadows (from other objects or from the same object)
  - Inter-reflections.

The edge descriptions are :

- **Edge normal: U**nit vector in the direction of maximum intensity change.
- **Edge direction:** Unit vector to perpendicular to the edge normal.
- **Edge position or center:** The image position at which the edge is located.
- **Edge strength:** Related to the local image contrast along the normal.

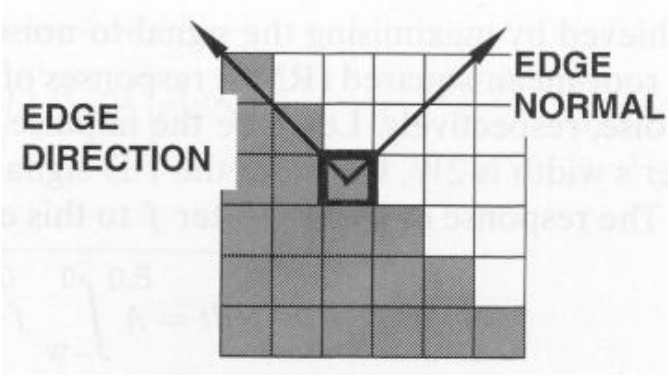

Fig 1. edge descriptions.[7]

The four steps of edge detection are:

1. **Smoothing:** The smoothing suppresses as much noise as possible, without destroying the true edges.
2. **Enhancement:** It apply a filter to enhance the quality of the edges in the image (sharpening).
3. **Detection:** It determine which edge pixels should be discarded as noise and which should be retained (usually, thresholding provides the criterion used for detection).

4. **Localization:** It determine the exact location of an edge (*sub-pixel* resolution might be required for some applications, that is, estimate the location of an edge to better than the spacing between pixels). Edge thinning and linking are usually required in this step.

The edge detection uses the derivatives which are listed below :

- Calculus describes changes of continuous functions using derivatives.
- An image is a 2D function, so operators describing edges are expressed using partial derivatives.

The points which lie on an edge can be detected by:

1. detecting local maxima or minima of the first derivative
2. detecting the zero-crossing of the second derivative.

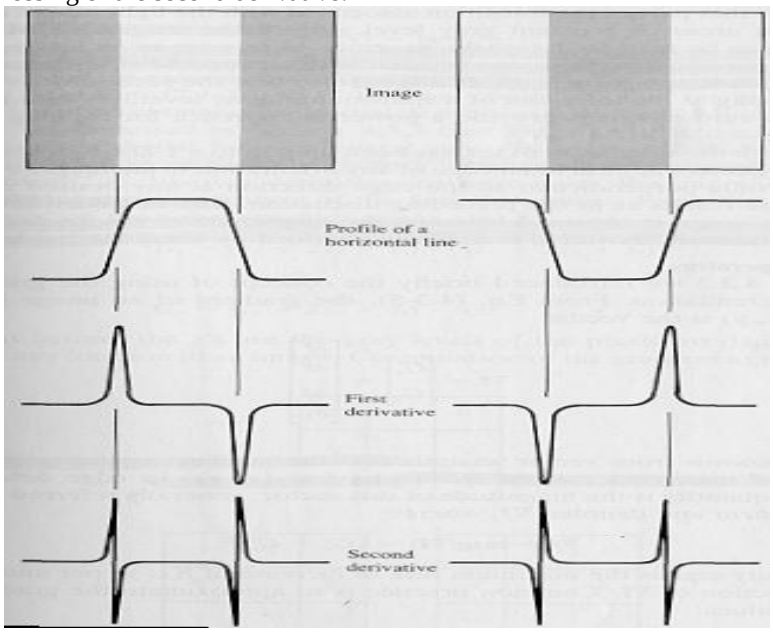

fig 2. image edge detection in first and second derivative methods[7]

- **Differencing 1D signals**

To compute the derivative of a signal, we approximate the derivative by finite differences:

Computing the 1st derivative:

$$f'(x) = \lim_{h>0} \frac{f(x+h)-f(x)}{f(x)} \approx f(x+1) - f(x) \; (h = 1)$$

mask : [-1 1]

Computing the 2nd derivative:

$$f''(x) = \lim_{h->0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) = f(x+2) - 2f(x+1) + f(x) \; (h = 1)$$

The approximation is centered about x+1 ;by replacing x+1 by x we obtain:

$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$

mask: [1 -2 1]

The different edge detection methods are:

- **The Roberts edge detector :**

$$\frac{\partial f}{\partial x} = f(i,j) - f(i+1, j+1)$$
$$\frac{\partial f}{\partial y} = f(i+1,j) - f(i, j+1)$$

This approximation can be implemented by the following masks:

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad M_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

- **The Prewitt edge detector:**

Consider the arrangement of pixels about the pixel (i, j):

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

The partial derivatives can be computed by:

$$M_x = (a2 + ca3 + a4) - (a0 + ca7 + a6)$$
$$My = (a6 + ca5 + a4) - (a0 + ca1 + a2)$$

The constant *c* implies the emphasis given to pixels closer to the center of the mask.

Setting $c = 1$, we get the Prewitt operator:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Main steps in edge detection using masks are :**
  1. Smooth the input image ($\hat{f}(x, y) = f(x, y) * G(x, y)$)
  2. $\hat{f}x = \hat{f}(x, y) * M_x(x, y)$
  3. $\hat{f}y = \hat{f}(x, y) * My(x, y)$
  4. $magn(x, y) = |\hat{f}x| + |\hat{f}y|$
  5. $dir(x, y) = tan^{-1}(\hat{f}y / \hat{f}x)$
  6. If $magn(x, y) > T$, then possible edge point

- **Criteria for optimal edge detection**
  1. Good detection: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges).
  2. Good localization: the edges detected must be as close as possible to the true edges. Single response constraint: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge (created by noise).

- **The Canny edge detector**

This is probably the most widely used edge detector in computer vision. Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise* ratio and localization. His analysis is based on "step-edges" corrupted by "additive Gaussian noise".

Algorithm:

1. Compute $f x$ and $f y$

$$f x = \frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x}G = f * G_x$$
$$f y = \frac{\partial}{\partial y}(f * G) = f * \frac{\partial}{\partial y}G = f * G_y$$

$G(x, y)$ is the Gaussian function

$G_x(x, y)$ is the derivate of $G(x, y)$ with respect to x: $G_x(x, y) = \frac{-x}{\sigma^2}G(x, y)$

$G_y(x, y)$ is the derivate of $G(x, y)$ with respect to y: $G_y(x, y) = \frac{-y}{\sigma^2}2\ G(x, y)$

2. Compute the gradient magnitude

$$magn(i, j) = \sqrt{f_x^2 + f_y^2}$$

3. Apply non-maxima suppression.
4. Apply hysteresis thresholding/edge linking.

- **Hysteresis thresholding/Edge Linking**

The output of non-maxima suppression still contains the local maxima created by noise.

- Can we get rid of them just by using a single threshold?
  - If we set a low threshold, some noisy maxima will be accepted too.
  - If we set a high threshold, true maxima might be missed (the value of true maxima will fluctuate above and below the threshold, fragmenting the edge).

A more effective scheme is to use two thresholds:

- A low threshold t_l
- A high threshold t_h
- Usually, $t_h \approx 2t_l$

Algorithm:

1. Produce two thresholded images $I_1(i, j)$ and $I_2(i, j)$.

(note: since $I_2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

2. Link the edges in $I_2(i, j)$ into contours
   2.1. Look in $I_1(i, j)$ when a gap is found.
   2.2. By examining the 8 neighbors in $I_1(i, j)$, gather edge points from $i_1(i, j)$ until the gap has been bridged to an edge in $I_2(i, j)$.

The algorithm performs edge linking as a by-product of double-thresholding.[7]

II. PREVIOUS WORK

Xiuli Zhang and Wei Liu [1] presented the different edge detection methods , wavelet transform methods and mathematical morphological methods. The grads operator algorithm is simple, but the detected edge is incomplete due to poor continuity and the detected edge line is thick. LOG operator is to use the second order differential operators to detect more edges having higher positioning accuracy. Canny operator and Jun Shen operator have the better positioning accuracy and the marginal seal with a difference that Jun Shen operator selects and used to smooth is the edge detection which is suitable for image multi-scale edge detection, suppresses noise and provide a higher accuracy in edge location. Morphological methods use the structural elements to measure and extract the corresponding shape of the image for the edge detection. The selection of structuring element is flexible, but the different forms of structuring elements must be selected for different images and the purpose of test, so the adaptability of algorithm is poor.

Zeyuan Gu[2] proposed the wavelet transform for image edge detection which is to improve the drawbacks of not being sensitive to direction properties. wavelet transform has good time-frequency localization and multi-resolution advantages for edge detection. For the better clarity, gradient sharpening process enhanced the fuzzy image. The paper shows the results by combining the two methods which is effective for an image edge detection. It is shown that it can be known that, for the same image compared with canny operator detection method which can obtain the edge detection based on new wavelet transform can abstract complete edge , take the accurate positioning and can keep better detailed information of tan image.

Mohamed A. El-Sayed [3] presented the entropic threshold technique for edge detection in image which uses both the shannon entropy and tsallis entropy. The proposed method decreased the computational time. The experimental results demonstrated that the proposed method for edge detection works efficiently for different grey level digital images. As the method is easy to implement. This detector retains the texture of the original image which can be utilized for the identification of fingerprints. With these advantages it has some disadvantages that it fails to provide all thin edges. The presence of thick edges at some locations need to be addressed by the proper choice of parameter.

Rashmi, Mukesh Kumar and Rohini Saxena [4] presented the various edge detection techniques. It is shown that the canny operator gives the better results than the others as it is less sensitive to noise, adaptive in nature, provides good localization and detects sharpen edges as compared to other operators.

Diplaxmi R. Waghule and Dr. Rohini S. Ochawar [5] presented the overview on the wide range of edge detection methods for one of the application like image segmentation. The paper presented various wavelet based methods, classical methods, FPGA based methods and so on. It is shown that wavelet methods are better than the other methods in the accuracy as it require less computations.

III. SELF ADAPTABLE BACTERIAL FORAGING OPTIMAZATION ALGORITHM[6]

Bacterial Foraging Optimization (BFO) is a nature-inspired optimization algorithm, which is based on the foraging behavior of E. coli bacteria. Experimentation with complex problems reveals that the BFO algorithm possesses a poor convergence behavior and its performance heavily decreases with the growth of the search space dimensionality and the problem complexity. In order to improve the BFO's searching performance, the Self-adaptive Bacterial Foraging Optimization (SA-BFO) is introduced. Instead of the simple description of chemotactic behaviour in original BFO, SA-BFO also incorporates the adaptive search strategy, which allows each bacterium strikes a good balance between exploration and exploitation during algorithmic execution by tuning its run-length unit self-adaptively. In what follows we briefly outline the original BFO algorithm step by step:

.

A. Step-by-step BFOA:

[Step 1] Initialize parameters n, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, $C(i)$ (i=1,2,…,S),Θ^i .

Where,

n: Dimension of the search space,
S: The number of bacterium,
N_c: chemotactic steps,
N_s: swim steps,
N_{re}: reproductive steps,
N_{ed}: elimination and dispersal steps,
P_{ed}: probability of elimination,
$C(i)$: the run-length unit during each run or tumble.

[Step 2] Elimination-dispersal loop: $l = l+1$.
[Step 3] Reproduction loop: $k = k+1$.
[Step 4] Chemotaxis loop: $j = j+1$.
[substep **a]** For i =1, 2,…, S, take a chemotactic step for bacteria i as follows.
[substep **b]** Compute fitness function, $J(i,j,k,l)$.
[substep **c]** Let $J_{last} = J(i,j,k,l)$ to save this value since we may find better value via a run.
[substep **d]** Tumble: Generate a random vector

$$\Delta(i) \epsilon R^n$$

with each element $\Delta_m(i)$, m = 1, 2, …, S, a random number on [-1, 1].
[substep **e]** Move: Let

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

This results in a step of size $C(i)$ in the direction of the tumble for bacteria i.
[substep **f]** Compute $J(i,j+1,k,l)$ with $\theta^i(j+1,k,l)$.
[substep **g]** Swim:
(**i**) Let $m = 0$ (counter for swim length).
(**ii**) While $m < N_s$ (if have not climbed down too long)

- Let $m = m+1$.
- If $J(i,j+1,k,l) < J_{last}$, let Jlast = $J(i,j+1,k,l)$. then another step of size $C(i)$ in this same direction will be taken as equation (1) and use the new generated $\theta^i(j+1,k,l)$ to compute the new $J(i,j+1,k,l)$.
- Else let m = N_s.

[substep **h]** Go to next bacterium (i+1): if $i \neq S$ go to (b) to process the next bacteria.
[Step 5] If $j < N_c$, go to step 3. In this case, continue chemotaxis since the life of the bacteria is not over.
[Step 6] Reproduction:
[substep **a]** For the given k and l, and for each $i = 1, 2, \ldots, S$, let J_{health} be the health of the bacteria. Sort bacterium in the order of ascending values.

$$J^i_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l) \quad (2)$$

[substep **b]** The S_r bacteria with the highest J_{health} values die and the other S_r bacteria with the best values split and the copies that are made are placed at the same location as their parent.
[Step 7] If $k < N_{re}$ go to step 2. In this case the number of specified reproduction steps is not reached and start the next generation in the chemotactic loop.
[Step 8] Elimination–dispersal: For $i = 1, 2, \ldots, S$, with probability p_{ed}, eliminate and disperse each bacteria, which results in keeping the number of bacteria in the population constant. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If $l < N_{ed}$, then go to step 2; otherwise end.

In order to get an insight into the behavior of the virtual bacteria in BFO model, we illustrate the bacterial trajectories in a fitness landscape by tuning the run-length unit parameter $C(i)$, which can essentially influence the bacterial behaviors. The fitness landscape is defined by the 2-D Ackley function (Equation 3), which has one narrow global optimum basin and many minor local optima. It is a widely used multimodal benchmark with the global optimum (0, 0) and the minimum is 0.

$$f(x,y) = -20\exp\left(-0.2\sqrt{\frac{1}{2}(x^2+y^2)}\right) - \exp\left(\frac{1}{2}(cos2\pi x + cos2\pi y)\right) + 20 + e \quad (3)$$

B. Self Adaptable Bacterial Foraging Algorithm[6]

In the SA-BFO algorithm, we introduce an "individual run-length unit" to the i^{th} bacterium of the colony and each bacterium can only modify the search behavior of itself by using the current status of its own. In this way, not only the position but also the runlength unit of each bacterium undergoes evolution. In SA-BFO evolution process, each bacterium displays alternatively two distinct search states:
(1) *Exploration* state, during which the bacterium employs a large run-length unit to explore the previously unscanned regions in the search space as fast as possible.
(2) *Exploitation* state, during which the bacterium uses a small run-length unit to exploit the promising regions slowly in its immediate vicinity. Each bacterium in the colony permanently maintains an appropriate balance between *Exploration* and *Exploitation* states by varying its own run-length unit adaptively. This is achieved by taking into account two decision indicators: a fitness improvement and no improvement registered lately. The criteria that determine the adjustment of individual run-length unit and the entrance into one of the states are the following:
Criterion-1: if the bacterium discovers a new promising domain, the run-length unit of this bacterium is adapted to another smaller one. Here "discovers a new promising domain" means this bacterium register a fitness improvement beyond a certain precision from the last generation to the current. Following *Criterion-1*, the bacterium's behavior will self-adapt into Exploitation state.
Criterion-2: if the bacterium's current fitness is unchanged for a number *Ku* (user-defined) of consecutive generations, then augment this bacterium's run-length unit and this bacterium enters Exploration state. This situation means that the bacterium searches on an un-promising domain or the domain where this bacterium focuses its search has nothing new to offer.
The flowchart of the ABFO1 algorithm can be illustrated by Figure , where S is the colony size, t is the chemotactic generation counter from 1 to max-generation, i is the bacterium's ID counter from 1 to S,X^i is the ith bacterium's position of the bacteria colony, Ns is the maximum number of steps for a single activity of Swim, and $flag^i$ is the number of generations the ith bacterium has not improved its own fitness. The Pseudocode of the dynamic self-adaptive strategy is written below.[8]

1. FOR (each bacterium i) IN PARELLEL

2. IF (criteria-1) then
3. C^t(t+1) =C^t (t)/α ; //exploit
4. ε^t (t+1) = ε^t/β ;
5. ELSE IF _*Criterion-2*_ then
6. C^t(t+1) = $C_{initial}$; // explore
7. ε^t (t +1)= $\varepsilon_{initial}$;
8. ELSE
9. C^t (t + 1) = C^t (t) ;
10. ε^t (t + 1) = ε^t (t) ;
11. END IF
12. END FOR IN PARELLEL

C. .Adaptive Bacterial Behaviors in ABFO Model[8]

In order to further analyze the adaptive foraging behaviours of the two proposed ABFO models, the two simulates are to be run . In both simulations, we excluded the reproduction, elimination, and dispersion events in order to illustrate the bacterial behaviours clearly. In the first simulation, the population evolution of the ABFO0 was simulated on 2-D Rosenbrock and Rastrigin functions. Specially, it shows the positions of the bacteria on certain chemotactic steps . In both cases, the evolution process proceeds 500 chemotactic steps, and S =50, n =100,$C_{initial}$ =0.1, $\varepsilon_{initial}$ =100, and α = β = 10 are chosen. Initially, it is seen that the bacteria colony is distributed randomly over the nutrient map defined by the 2-D Rosenbrock function .It is observed that, in the end of the first phase, all the bacterial producers had found the long and narrow valley of the Rosenbrock function (which contains the global optimum) and move around it. In the second and third phases , the bacterial scroungers initialized in this valley found by the producers and exploit the global optimum along it. Qualitatively, it is found a similar pattern in , where the bacteria colony pursue the valleys and avoid the peaks of the multimodal Rastrigin function. In the first phase it is starting from their random initial positions, the bacterial producers explore many regions of the nutrient map. In the second and third phases , the bacterial scroungers join the sources found by the producers and find a good deal of local optima of Rastrigin function, including the global optimum. In the second simulation, we demonstrate the self-adaptive foraging behaviour of a single bacterium in the ABFO1 model.

IV. CONCLUSION

This paper gives the review on different methods of image edge detection which is a part of image segmentation. The self adaptive bacterial foraging algorithm is introduced in this paper to search for the optimal threshold. This work would be compared to other thresholding detection methods based in terms of segmentation accuracy and segmentation time.

REFRENCES

[1] Xiuli Zhang, Wei Liu , "The Research on the Methods of Image Edge Detection, " *The International Conference on Intelligent Networks and Intelligent Systems*, pp. 100-103, 2010.

[2] Zeyuan Gu, "An Improved Method for Research on Image Edge Detection*," International Conference on Intelligent System and Knowledge Engineering*,2010.

[3] Mohamed A. El-Sayed, "A New Algorithm Based Entropic Threshold for Edge Detection in Images, *" International Journal of Computer Science Issues*, Vol.8, pp.71-78,2011.

[4] Rashmi, Mukesh Kumar and Rohini Saxena, "Algorithm and Technique on Various Edge Detection: A Survey, " *Signal and Image Processing :An International Journal* Vol.4,pp.65-75,2013.

[5] Diplaxmi R. Waghule and Dr. Rohini S. Ochawar, "Overview on Edge Detection Methods,*" International Conference on Electronic System, Signal Processing and Computing Technologies*,pp.151-155,2014.

[6] Hanning Chen1,Yunlong Zhu1, Kunyuan Hu, " Self Adaptation in Bacterial Foraging Optimization Algorithm," *International Conference on Intelligent System and Knowledge Engineering*,pp.1026-1031,2008.

[7] Trucco, Jain et al, *edge detection* Chapt 4 and, Chapt 5.

[8] Hanning Chen, Yunlong Zhu, and Kunyuan Hu **,"** *Adaptive Bacterial Foraging Optimization,"* Hindawi Publishing Corporation Abstract and Applied Analysis , 2011.

