Optimization of edge detection algorithm for digital images through dynamic programming

Anju Mishra, Kushagra Goel, Pankaj Sharma

Abstract— Most of the edge detection algorithms are optimized with classic techniques and I proposed the “optimization of edge detection algorithm for digital images through dynamic programming.”

A dynamic programming technique is proposed for locally finding edges in gray level digital images. A class of characteristic functions is proposed from which the best “edge function” (according to an optimality criterion) is chosen for each window by the dynamic programming technique. The windows are selected by first decomposing the entire image into equal square regions and then breaking into four equal subsquares any region where the quality of fit of the best edge function (as measured by the optimality criterion) falls below a fixed tolerance, and so on. The procedure is summarized in a conceptual algorithm, and the technique is first illustrated by application to a digital image of an artificial design. Second, for comparison, the dynamic programming method and a split-and-merge method are used for image enhancement on a noisy version of a muscle cell culture image. Then the dynamic programming method and a thresholded gradient operator technique are used for edge detection on both the original and the noisy version of this image.

Keywords— Edge detection, Chemotaxis, Elimination, Swarming, Reproduction, Sobel, Bacterial Foraging, Canny.

1 Introduction

An edge may be defined as a set of connected pixels that forms a boundary between two disarrange regions. Edge detection is a method of segmenting an image into regions of conclusion. Edge detection plays a very important role in digital image processing and practical aspects of our life. Most of the edge detection algorithms are optimized with classic techniques i.e., (1) Bacterial Foraging Algorithm commonly known as BFA ,(2) Sobel Edge Detection and (3) SUSAN Edge Detection. Bacterial Foraging works on swarm intelligence and Sobel uses the concept of Image Gradient for Edge detection. SUSAN method works by associating a small area of neighboring pixels with similar brightness to each center pixel. This Paper uses dynamic programming technique is proposed for locally finding edges in gray level digital images.

Swarm word means the group of flying insects. Swarm intelligence works on the intelligence and behavior of that group. Bacterial Foraging Algorithm is based on Swarm Intelligence. In Bacterial Foraging Algorithm foraging means search of food. In this search bacteria seeks to maximize its nutrition value as it is optimization algorithm. In our case nutrition means tracing of pixel of edges. Movement of bacteria is swimming or tumbling. Direction of tumbling movement is in all directions. Canny edge detector is an edge detection operator that uses a multistage algorithm to detect a wide range of edges in images directions. Sobel operator creates image emphasizing edges. It uses Image gradient factor for Edge detection.

- *Anju Mishra is currently pursuing masters degree in computer science engineering inMaharshi Dyanand University, Rohtak.*
- *Kushgra Goel, Assistant Professor in IEC-College of Engineering,Greater Noida, U.P.*
- *Pankaj Sharma, Associate Professor in DITMR, Fardibad,Haryana.*

2 Edge Detection

Edge detection is very important terminology in image processing and for computer vision. Edge detection is in the forefront of image processing for object detection, so it is crucial to have a good understanding of edge detection operators. Edge detection is the name for a set of mathematical methods which aim at identifying points in a digital image at which the image brightness can changes sharply or more formally, has discontinuities. Any points where image brightness changes sharply are typically organized into a set of curved line segments into termed edges. The same problem of finding breaks in 1D signal is known as step detection and the problem of finding signal discontinuities over time is known as change detection in image processing. Edge detection is a important basic tool in image processing where machine vision and computer vision particularly in the areas of feature detection and feature extraction. Digital image is composed of a finite number of components, each of which has a special place or position and value. These components are cited to as picture elements, image elements, and pixels .Image processing is any form of signal processing for which image is the input, such as a photograph and the image processing output may be whether an image or, a set of characteristics or parameters associated to the image Edge can also be defined as in binary images as the black pixels with one nearest white neighbor. Edges include large amount of important information about the image. The changes in pixel intensity describe the boundaries of objects in a picture. Feature detection and Feature extraction are the main areas of image processing, where Edge detection is a important tool. Image edge detection trades with drawing out of edges in an image by recognizing high intensity variations in the pixels. This action discovers outlines of an object and boundaries between objects and the back part of the image.

Detection of edges for an image may help in image segmentation, data compression, and for image reconstruction and so on. Variables involved in selection of an edge detection operator include edge orientation, noise environment and edge structure. Edge detection is difficult in noisy images, since both the noise and the edges include high-frequency essence. Attempts to reduce the noise consequence are blurred and distorted edges.
Dynamic programming is a technique for solving optimization problems when not all the evaluation functions are interrelated simultaneously.
Edge detection is used mainly to extract the information about the image e.g. image sharpening and enhancement, location of object present in the image, their shape, size. Depending upon variation of intensity / grey level, various types of edges are shown in Figure 1.

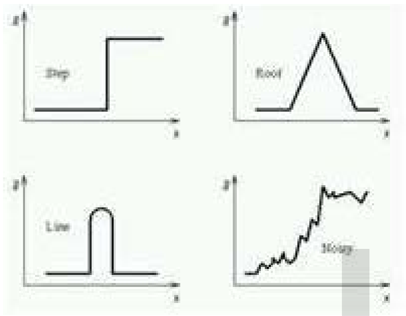


Fig 1 Typical Edge Profile

3 Bacterial Foraging Algorithm

Bacterial Foraging Scheme appeared in Passino (2002), Liu And Passino (2002). Foraging can be modeled as an Optimization Process where Bacteria Seek to Maximize the Energy Obtained per Unit Time Spent during Foraging. In this scheme, an objective function is posed as the effort or a cost incurred by the bacteria in search of food.

Search Space: Search space is a 2- dimensional area with x-y plane which are discrete pixel value representation of any image. Search space is finite as it is confined pixels' vertical and horizontal location.
BFA consist of mainly four steps:
3.1 Chemotaxis
3.2 Swarming
3.3 Reproduction
3.4 Elimination

3.1 Chemotaxis

This step is the important step for BF. As in it bacteria decides whether it will swim or tumble in predefined direction or in another different direction respectively. In this process we led bacteria to search a pixel and make him away from noisy pixels.The direction of bacteria's movement will depend upon its flagella. As our search space is 2- dimensional so there
will be 8 neighboring pixels in directions E, W, N, S, NE, SE, NW,SW. out of these direction pixel has to pick one direction to reach to a pixel and avoiding a noisy pixel. The bacterial movement is:

$$\theta^i(j+1,k,l) = \phi[m', n', i, j+1, k, l], \qquad (1)$$

Where function represent the position of bacteria, m', n' are coordinates for movement.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + \frac{C(i)\cdot\Delta(i)}{\sqrt{\Delta(i)^T\Delta(i)}} \qquad (2)$$

C(i) is length of step size, is random number in R2 .

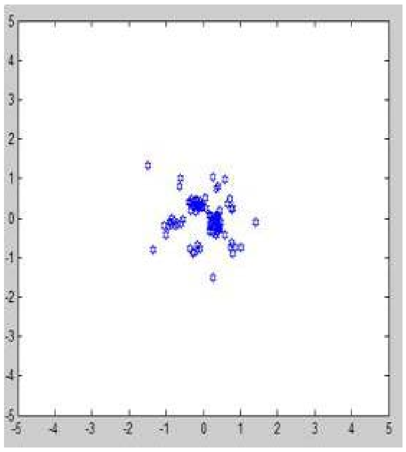
3.2 Swarming

For As in swarming bacteria follows follow but in this a bacteria who found optimum path to reach pixel signals another bacteria to follow him. Each bacteria releases a fluid repellent which signals another bacteria to have a safer distant. The cell to cell movement of bacteria with signaling combines both attraction and repelling effect. So we have Θi is the location of ith bacteria. The eq(3) is basically a function for distancez between the bacteria

$$\begin{aligned} J_{cc}(\theta^i(j,k,l),\theta(j,k,l)) &= \sum_{t=1}^{S} J^t_{cc}(\theta^i,\theta^t) \\ &= \sum_{t=1}^{S}\left[-d_{att}\exp\left(-w_{att}\sum_{m=1}^{P}\left(\theta^i_m-\theta^t_m\right)^2\right)\right] \\ &\quad+\sum_{t=1}^{S}\left[h_{rep}\exp\left(-w_{rep}\sum_{m=1}^{P}\left(\theta^i_m-\theta^t_m\right)^2\right)\right] \end{aligned} \qquad (3)$$

So according to the clique of the position (pixel) of bacteria the distance from bacteria will be

$$J_{cc}(\theta^i(j,k,l),\theta(j,k,l)) = f(distance) + f(clique)$$



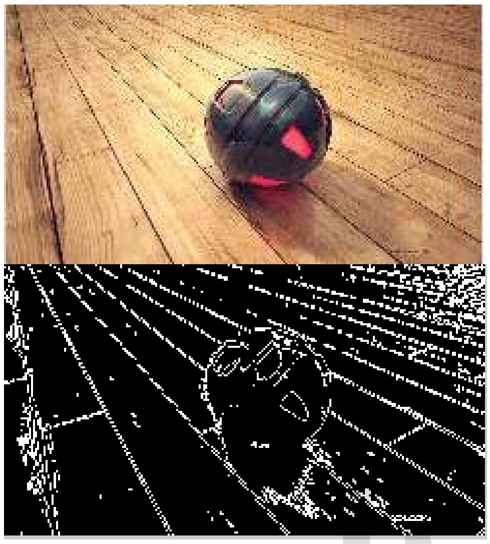
1.3 Reproduction Step

It means the healthy bacteria will be divided into two bacteria in other words reproduces two bacteria

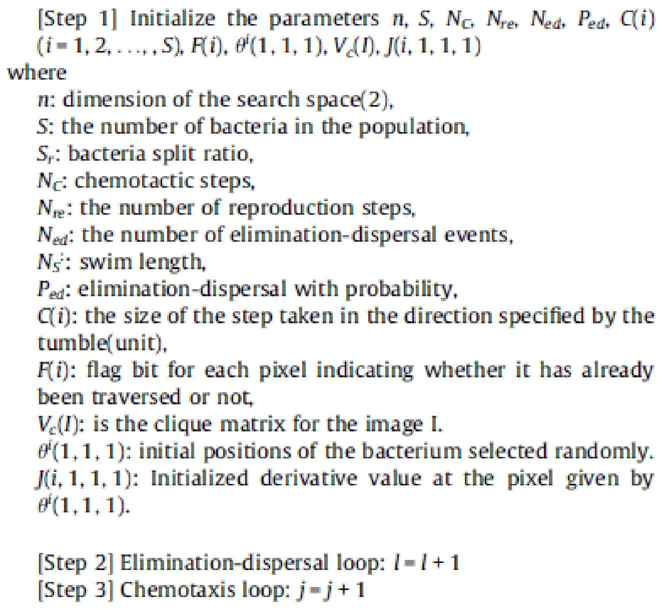
Sr = S/2

3.4 Elimination Step

The divided bacteria will have certain health state less healthy bacteria will eliminate. There will be two states one will be healthy and other is toxic state. The bacteria having toxic state will eliminate.



Algorithm



[Step 1] Initialize the parameters n, S, N_C, N_{re}, N_{ed}, P_{ed}, $C(i)$ $(i = 1, 2, \ldots, S)$, $F(i)$, $\theta^i(1, 1, 1)$, $V_c(I)$, $J(i, 1, 1, 1)$

where

n: dimension of the search space(2),

S: the number of bacteria in the population,

S_r: bacteria split ratio,

N_C: chemotactic steps,

N_{re}: the number of reproduction steps,

N_{ed}: the number of elimination-dispersal events,

N_S: swim length,

P_{ed}: elimination-dispersal with probability,

$C(i)$: the size of the step taken in the direction specified by the tumble(unit),

$F(i)$: flag bit for each pixel indicating whether it has already been traversed or not,

$V_c(I)$: is the clique matrix for the image I.

$\theta^i(1, 1, 1)$: initial positions of the bacterium selected randomly.

$J(i, 1, 1, 1)$: Initialized derivative value at the pixel given by $\theta^i(1, 1, 1)$.

[Step 2] Elimination-dispersal loop: $l = l + 1$

[Step 3] Chemotaxis loop: $j = j + 1$

4 SOBEL

IJSER style A convolution mask is used is usually much smaller than the actual image. As a result, the mask is slid over an area of the input image, changes that pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of a row. It then starts at the beginning of the next row. The example below shows the mask being slid over the top left portion of the input image represented by the green outline. The formula shows how a particular pixel in the output image would be calculated. The center of the mask is placed over the pixel you are manipulating in the image. And the I & J values are used to move the file pointer so you can mulitply, for example, pixel (A22) by the corresponding mask value (M22). It is important to notice that pixels in the first and last rows, as well as the first and last columns cannot be manipulated by a 3x3 mask. This is because when placing the center of the mask over a pixel in the first row (for example), the mask will be outside the image boundaries.

Input Image + Pseudo- Convolution Mask

A11	A12	A13	…	A1k
A21	A22	A23	…	A2k
A31	A32	A33	…	A3k
:	:	:	:	:

M11	M12	M13
M21	M22	M23
M31	M32	M33

= Output Image

B11	B12	B13	…	B1k
B21	B22	B23	…	B2k
B31	B32	B33	…	B3k
:	:	:	:	:

The above figure illustrates the working of sobel pseudo convolution masks when it is applied to the input given input image. The sobel pseudo mask performs the following operation to calculate the point B22 of the image:

B22 = (A11*M11) + (A12*M12) + (A13*M13) + (A21*M21) + (A22*M22) + (A23*M23) + (A31*M31) + (A32*M32) + (A33*M33)

The masks can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx2 + Gy2}$$

Although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\Theta = \arctan(Gx/Gy) - 3\pi/4$$

In this case, orientation 1 is taken to mean that the direction of maximum contrast from black to white runs from left to right on the image, and other angles are measured anticlockwise from this. Often, this absolute magnitude is the only output the user sees --- the two components of the gradient are conveniently computed and added in a single pass over the input image using the pseudo-convolution operator as shown in Figure.

PSEUDO-CODES FOR SOBEL EDGE DETECTION

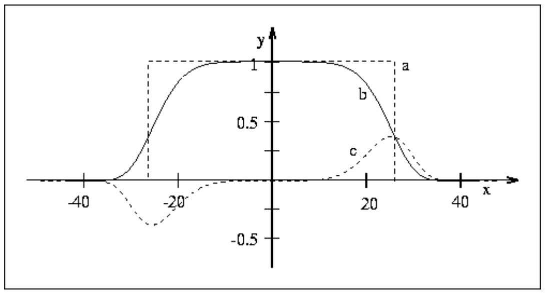
```
Input: A Sample Image.
Output: Detected Edges.
Step 1: Accept the input image.
Step 2: Apply mask Gx, Gy to the input image.
Step 3: Apply Sobel edge detection algorithm and the gradient.
Step 4: Masks manipulation of Gx, Gy separately on the input image.
Step 5: Results combined to find the absolute magnitude of the gradient.
Step 6: The absolute magnitude is the output edges.
```

5 SUSAN

The SUSAN edge finder has been implemented using circular masks (sometimes known as windows or kernels) to give isotropic responses. Digital approximations to circles have been used, either with constant weighting within them or with Gaussian weighting -- this is discussed further later. The usual radius is 3.4 pixels (giving a mask of 37 pixels), and the smallest mask considered is the traditional three by three mask. The 37 pixel circular mask is used in all feature detection experiments unless otherwise stated.

The mask is placed at each point in the image and, for each point; the brightness of each pixel within the mask is compared with that of the nucleus (the centre point). Originally a simple equation determined this comparison.

$$c(\vec{r},\vec{r_0}) = \begin{cases} 1 & \text{if } |I(\vec{r}) - I(\vec{r_0})| \le t \\ 0 & \text{if } |I(\vec{r}) - I(\vec{r_0})| > t, \end{cases} \qquad (1)$$



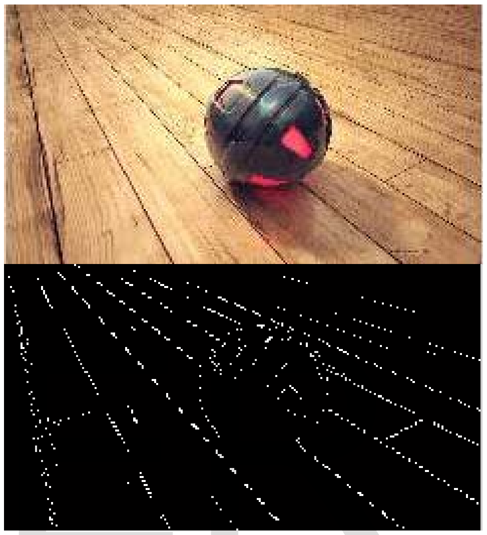
a) a.) The original similarity function (y axis, no units) versus pixel brightness difference (x axis, in grey levels). For this example the pixel brightness difference ``threshold'' is set at ±27 grey levels.

b) b.) The more stable function now used.

c) c.) The boundary detector B (see later text).

6 CANNY

Canny edge detector is an edge detection operator that uses a multistage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced
a computational theory of edge detection explaining why the technique works.



7 DYNAMIC PROGRAMMING FOR IMAGES

To formulate the edge following procedure as dynamic programming, one must define an evaluation function that embodies the notion of the "best edges" [Montanari 1971; Ballard 1976].

Suppose that local edge detection operator is applied to gray level picture to produce magnitude and direction information. Then one possible criterion for a "good edge" is a weighted sum of high cumulative edge strength and low cumulative curvature; that is, for an n-segment curve,

$$h(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{k=1}^{n} s(\mathbf{x}_k) + \alpha \sum_{k=1}^{n-1} q(\mathbf{x}_k, \mathbf{x}_{k+1})$$

Where the implicit constraint is that consecutive xk must be grid neighbor:

$$\|\mathbf{x}_k - \mathbf{x}_{k+1}\| \leqslant \sqrt{2}$$

$$q(\mathbf{x}_k, \mathbf{x}_{k+1}) = diff\,[\phi(\mathbf{x}_k), \phi(\mathbf{x}_{k+1})]$$

Where α is negative. The function g we take to be edge strength, i.e., g(x)=s(x). Notice that this evaluation function is in the form of

$$h(\cdot) = h_1(x_1, x_2) + h_2(x_2, x_3) + h_3(x_3, x_4)$$

and can be optimized in the stages:

$$f_0(\mathbf{x}_1) \equiv 0$$
$$f_1(\mathbf{x}_2) = \max_{x_1}[s(\mathbf{x}_1) + \alpha q(\mathbf{x}_1, \mathbf{x}_2) + f_0(\mathbf{x}_1)]$$
$$f_k(\mathbf{x}_{k+1}) = \max[s(\mathbf{x}_k) + \alpha q(\mathbf{x}_k, \mathbf{x}_{k+1}) + f_{k-1}(\mathbf{x}_k)]$$

These equations can be put into the steps:

Algorithm: Dynamic Programming for edge detection

1. Set k=1
2. Consider only x such that s(x) ≥ T. For each of these x, define low curvature pixel "in front of " the contour direction.
3. Each of these pixels may have a curve emanating from it. For k =1, the curve is one pixel in length. Join the curve to x that optimizes the left hand side of the recursion equation.
4. If k = N , pick the best fN-1 and stop. Otherwise, set k = k + 1 and go to step 2.

This algorithm can be generalized to the case of picking a curve emanating from x (that we have already generated: Find the end of that curve, and join the best of three curves emanating from the end of that curve. Figure shows this process. The equations for general case are

$$f_0(\mathbf{x}_1) \equiv 0$$
$$f_l(\mathbf{x}_{k+1}) = \max_{x_k}[s(\mathbf{x}_k) + \alpha q(\mathbf{x}_k, t(\mathbf{x}_{k+1}))$$
$$+ f_{l-1}(\mathbf{x}_k)]$$

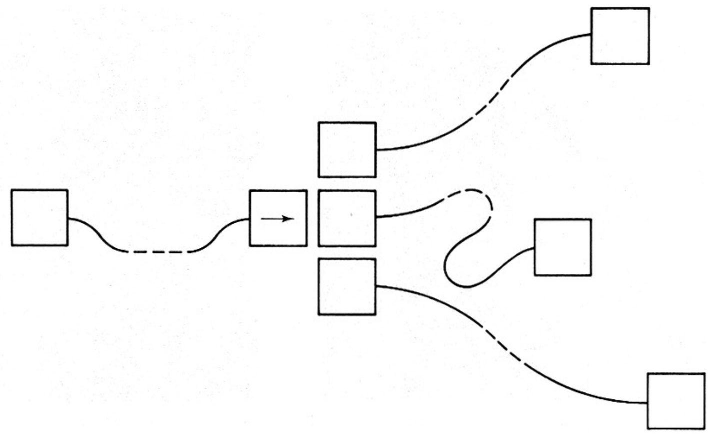


Figure: Dynamic Programming optimization for edge selection

Where the curve length n is related to α by building sequence n(l) such that n(1) =1, n(L) =N, and n(l) – n(l-1) is a member of {n(k) | k = 1,……..,l-1}. Also, t(xK) is a function that extracts the tail pixel of the curve headed by xK .

8 Lower Resolution Evaluation Function

In the Dynamic Programming formulation, the components g(xK) and q(xK,, xk+1) in the evaluation function is very localized; the variable x for successive s and q are in fact constrained to be grid neighbors. This need to be the case : The x can be very distant from each other without altering the basic technique. Furthermore, the function g and q need not be local gradient and absolute curvature, respectively, but can be any functions defined on permissible x.

The Fischler and Elschlager 1973, formulation models an object as a set of parts and relations between parts. Template functions, denoted by g (x), measure how well a part of model matches a part of image at the point x. Relational functions, denoted by qkj(x,y) , measure how well the position of the matches of the k th part at (x) agrees with the position of the match of the jth part at (y).

T(x) ≡ template centered at x computed as an aggregate of a set chest radiographs

$g(x_k) = \Sigma\ (T(x-x_k) f(x)) / |T|\,|f|$

$\Theta(x_k, x_j)$ = expected angular orientaton of x_k from x_j

$q(x_k\, x_j) = [\Theta(x_k, x_j) - \arctan(y_k - y_j)/(x_k - x_j)]$

This method was formalized using lower resolution objective function.

9 Generalization to Higher – Dimensional Image Data

The generalization to higher-dimensional spaces is straight forward [Liu 1997; Herman and Liu 1978]. Higher dimensional image spaces arise when the image has more than two spatial dimensions is time-varying, or both. In these images the notion of the gradient is the same (a vector describing the maximum gray level change and its corresponding direction), but the intuitive interpretation of the corresponding edge element may be difficult. In three dimensions, edge elements are primitive surface elements, separating volumes of differing gray scales. The objective of contour following is to link together neighboring surface elements with high gradient modulus values and similar orientations into larger boundaries. In four dimensions, edge elements are primitive volumes; contour following links neighboring volumes with similar gradients.

10 Conclusion

The adaptability of the proposed edge detector was demonstrated in a dynamically changing environment made of a set of digital grayscale images. The windows are selected by first decomposing the entire image into equal square regions and then breaking into four equal sub squares any region where the quality of fit of the best edge function. The algorithm responded to the changes by generating sub square patterns according to the distribution of the newly-created edges. It also proved to be robust since even a BFA, Sobel, SUSAN and canny of a smaller size could detect the edges, even though the number of detected edge pixels was reduced and expectedly faster than these edge detection technique.

11 References

(1) Comparative Analysis of Digital Image for Edge Detection by Using Bacterial Foraging, Amit Agarwal, Kushagra Goel, CICT IEEE Conference, Febrauary 2016

(2) A Survey on Edge Detection Using Different Techniques, Kiranjeet Kaur, Sheenam Malhotra, International Journal of Application or Innovation in Engineering & Management(IJAIEMI) Volume 2, Issue 4, April 2013

(3) A novel bacterial foraging technique for edge detection-Om Prakash Verma(a), Madasu Hanmandlu(b), Puneet Kumar©, Sidharth Chhabra(a), Akhil Jindal(a). (a) Delhi Technological University (Formerly Delhi College of Engineering), Delhi, India (b) Department of Electrical Engineering, IIT Delhi, Delhi, India (c) Advanced Systems Laboratory, Hyderabad, India.

(4) Abdallah, A.A., Ayman, A.A., 2009. Edge detection in digital images using fuzzy Logic techniques. World Academy of Sci. Eng. Technol.

(5) Canny, J.F., 1986. A computational approach to edge detection. IEEE Trans. Pattern Anal. Machine Intell.

(6) Cohen, J., 1960. A coeffici Verma, O.P., Hanmandlu, M., Kumar, P., Srivastava, S., 2009. A novel approach for edge detection using ant colony optimization and fuzzy derivative technique Proc. IEEE, IACC ent of agreement for nominal scales. Educ. Psychol Mea

(7) Optimal Segmentation of Signals Based on Dynamic Programming and Its Application to Image Denoising and Edge Detection, Steven Kay and Xu Han,Citeseex

(8) R. J. Beattie, "Edge detection for semantically based early visual processing," Ph.D. dissertation, Univ. Edinburgh, 1984.

(9) C. Bingham, M. D. Godfrey, and J. W. Tukey, "Modem techniques of power spectrum estimation," IEEE Trans. Audio Electroacoust., vol. AU-15, no. 2, pp. 56-66, 1967.

(10) R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," Dep. Comput. Sci., Stanford Univ., Stanford, CA, Rep. AIM- 343, 1981.

(11) J. F. Canny, "Finding edges and lines in images," M.I.T. Artificial Intell. Lab., Cambridge, MA, Rep. Al-TR-720, 1983.

(12) F. S. Cohen, D. B. Cooper, J. F. Silverman, and E. B. Hinkle, "Simple parallel hierarchical and relaxation algorithms for segmenting textured images based on noncasual Markovian random field models," in Proc. 7th Int. Conf. Pattern Recognition and Image Processing, Canada, 1984.

(13) R. Courant and D. Hilbert, Methods of Mathematical Physics, vol. 1. New York: Wiley-Interscience, 1953.

(14) J. R. Fram and E. S. Deutsch, "On the quantitative evaluation of edge detection schemes and their comparison with human performance," IEEE Trans. Comput., vol. C-24, no. 6, pp. 616-628, 1975.

(15) M. Gennert, "Detecting half-edges and vertices in images," in IEEE Conf. Comput. Vision and Pattern Recognition, Miami Beach, FL, June 24-26, 1986.