Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 167 (2020) 1421–1436

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

Edge Detection Technique using Binary Particle Swarm Optimization

Naveen Singh Dagar[a,*], Pawan Kumar Dahiya[b]

[a]*Research Scholar, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Haryana, India-131039*
[b]*Assistant Professor, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Haryana, India-131039*

Abstract

Edge detection is long established in computer eyesight applications such as article identification, shape matching, medical image classification, etc. For this reason, many edge detectors like LOG, Prewitt, Canny, etc. have been developed in the past in order to boost the grouping correctness of edge pixels. All these approaches work fine on images having minimum variation in intensity, however, their performance is not consistent on images having high-intensity variation. Therefore, in this paper "Binary Particle Swarm Optimization (BPSO)" based edge detection methodology minimizing multi-objective fitness function is proposed. Multi-objective fitness function is formulated by considering the weighted sum of five cost factors and all these cost factors are associated with four techniques of edge validation. The proposed approach is examined on 500 "BSD" images and results are compared with classical edge detectors (Canny, Prewitt) as well as computational intelligent techniques (ACO, GA) using the F score performance parameter. Performance of the proposed approach are consistent on all testing images and outperform all classical edge detectors, ACO and GA having average F score 0.2901 and have little standard deviation (0.0401).

Peer-review under responsibility of the scientific committee of the International Conference on Computational Intelligence and Data Science (ICCIDS 2019).

Keywords: Image Processing; Edge Detection; BPSO; Canny; Prewitt; ACO; GA; PSO; BSD

* Corresponding author. Tel.: +918700099478.
*E-mail address:*naveendagar87@gmail.com

Peer-review under responsibility of the scientific committee of the International Conference on Computational Intelligence and Data Science (ICCIDS 2019).
10.1016/j.procs.2020.03.353

1. Introduction

Edge detection is a critical assignment in computer eyesight. It is frequently used as a first step in getting information from images, that's why it is still an active research area. Edge detection is used in object recognition, image segmentation as well as to understand the structure of an image. In computer eyesight application, the edges in an image give very helpful information about the frontier/boundary of the articles. The edges in an image are formed by changes in the object's properties such as geometry, shape, size, light intensity and reflection. In an image, an edge in a gray-level image is the boundary between two regions of different grey-level [1]. In other words, an edge is the discontinuities either in image intensity or its first derivative. Discontinuities in the intensity of the image are of two types one is step discontinuities and the second one is line discontinuities. Step discontinuities are due to abrupt changes in image intensity. Whereas, line discontinuities are also abrupt changes in image intensity and return back to original intensity after a certain distance. However, these discontinuities rarely exist in real images because of the smoothening introduced by most sensing devices, low-frequency components in images and external noise. Thus, step intensity change edges formed the ramp edges and line intensity change edges formed roof edges.

Depending upon the image quality varieties of edge cross-sections are produced. Some of the universal factors that contribute to edge degradation are:

1. Quantum effects or photon noise
2. Defocusing or blurring
3. Irregularities of the surface of the objects

Since, the edge is a key attribute in computer eyesight applications such as article identification, the examination of the scene, etc. Therefore, edge degradation must be dealt with any edge detection scheme [1]. Most of the edge detection operators are Laplacian-of-Gaussian (LOG), gradient operator (GO), and Laplacian operator (LO). These edge detectors are very simple and find edge pixels when the "first-order derivative" intensity of the pixel is greater than some threshold or "second-order derivative" intensity of the pixel sustain a zero crossing. The drawbacks of these operators are that it is effective in detecting finite types of edges and is highly responsive to noise/commotion which many times often causes splinter\fragmented edges [2]. Upon analyzing the problem of differentiating on sampled images, the researcher suggests that differentiation is an ill-posed problem [3]. LOG uses Laplacian filter and because of this, it reduces the accuracy in finding out the orientation of edges and often fails at the corners and at the bends where the intensity of the gray level changes rapidly [4]. Gaussian filter is very helpful in regularizing ill-posed problem of differentiation. Robert's edge detectors also fail on the curved edge in an image.

The two most common edge detection techniques used in computer vision applications are Thresholding and the edge connection approach. In the thresholding approach, the discontinuities in the gray tone of the image can be enhanced using neighborhood operators [5]. "Robert's" operator is the simplest type of differential operator which was proposed nearly fifty years ago. This operator made up of 2 * 2 convolution kernels. "Prewitt and Sobel" operators are an improvement on "Robert's" operator, where the mask has made up of 3 * 3 pixels [5]. Generally, all gradient-based operators produce broken and thick edges [6]. However, on sharp edges, good results are provided by all the operators. Whereas, the working capability of gradient-based algorithms significantly reduces the presence of noise. The "Sobel" operator calculates the intensity of the local gradients and replies to non-ideal step edges which are not good. "Laplacian" edge operator is obtained by making small changes in Sobel operator which is also called "second-order derivative operator" [5]. "RUKF (reduced update Kalman filter)" operation gives enough repairing empower to 3 * 3 derivative operators to be successful. Derivative operators do not need images having a noise for edge detection [7]. Diagonal edges in an image are not detected by the Prewitt operators efficiently. A gradient operator detects the edge as well as the isolated point [8]. "Canny" operator is time consuming due to complex computation [9]. Local gradient-based operators such as Roberts, Prewitt, Sobel canny, Laplace, LOG are limited to unit pixel gradient [10]. Some drawbacks of gradient-based operators are shown in table 1 [11].

Table 1 some drawbacks of gradient-based operator

OPERATORS	DRAWBACKS
Classical(Sobel, Prewitt)	Thick and incorrect edges. Affectability towards commotion/noise.
Zero-Crossing, Boolean. (Laplacian, Second vector derivative)	Specked and separated edges. Affectability towards commotion/noise
Marr-Hildreth (Laplacian of Gaussian(LOG))	Breaking down at the corners, bends and where the dim level force work differs. Not detecting the introduction of the edge as a result of utilizing the Laplacian channel Specked and wide edges.
Canny (Gaussian)	Complicated Calculations Fake zero intersection Tedious Genuinely befuddled at corners because of the Gaussian smoothing of the picture Failure to manage fine grain locales
Coloured Edge Detectors	Complicated Complex Computations

One of the reasons for the poor performance of the majority of the "edge detection techniques" is their assumption about what creates the edge. Step intensity changes in the images are assumed edges by most of the techniques whereas in real images step edges become ramp edges as discussed above. This generalization of these techniques makes it difficult to work on texture edge and intensity edge with arbitrary profile [2]. Moreover, these edge detectors work on the response at a single-pixel location as an edge detection problem. However, these approaches largely ignore the edge structure around a given pixel.

"Tan et al.[12] and Acton and Bovik [13] have formulated the edge detection problem as an optimization problem whose objective is to minimize the cost." The paper solves the drawback of prior "edge detection techniques" by producing a common clarity of edge which involves almost all edge types. The paper clearly examines the local edge structure in the locality of the assumed edge pixels in order to refine the existing edge detection techniques. An edge in an image is a boundary that separates two areas that have notably different features. For edge detection, Tan et al. [12] have applied a comparative cost function approach. An edge image a cost measure function that explains in terms of edge thickness, continuity, the region of variation and splinter\fragmentation, etc. Later, authors in [2] proposed an "edge detection technique using a genetic algorithm (GA)" considering the same cost components as taken by Tan et. al [12]. The paper also formulated the edge detection problem as a multi-objective cost minimization problem.

In this paper "Binary Particle Swarm Optimization (BPSO)" based edge detection methodology minimizing multi-objective fitness function is proposed. BPSO is computationally faster, having very less number of parameters

to optimize and converge at global optima. Multi-objective fitness function is formulated by considering the weighted sum of five cost factors and all these cost factors are associated with four techniques of edge validation. Rest of the paper is organized as follows: Section 2 gives the brief idea of problem formulation; Section 3 explains the cost factor linked with cost function; Section 4 the describe the PSO technique; Section 5 gives idea of BPSO technique considered for the edge detection; Section 6 gives the experimental result and discussion; Section 7 gives the conclusion and future scope of the paper.

2. Problem Formulation

Every gray-scale image can also be projected as a bi-dimensional matrix formed up of pixels H(p, q), $1 \leq p \leq p_{max}$, $1 \leq n \leq q_{max}$, where every one of those pixels has a certain GSV(i.e. Gray Scale Value) which falls in between 0 and 255 (i.e. 8 bits every pixel). This considered scenario has assumed $p_{max} = q_{max} = 256$ is the size of the grayscale image. The vertex image i.e. configuration of the edge, identify itself as a Boolean or binary matrix of pixels J(p, q), $1 \leq p \leq p_{max}$, $1 \leq q \leq q_{max}$ where every pixel values at either 0 or at 1.Whenever the value of J(p, q) = 1 the pixel is identified with edge/vertex pixel and if the value of J(p, q) isn't equal to zero it isn't identified as edge/vertex pixel. The cost function is also formulated as described in [2, 12]. Edge structure validation within a primary neighborhood of the pixel identified as the edge pixel is described in the following subsections.

2.1 Local edge structure validity

The validity of a structure defined as edge structure compromising of the 3by3 matrix can be grouped in one of the below-mentioned groups. So, if an edge pixel compromises of:
(1) Absent or Singular neighboring pixel of the edge.
(2) Bi-neighbouring pixels of the edge added to that our edge structure does not make any turn of an angle $\theta \geq 45^0$as appeared in Fig. 1.
(3) Three neighboring pixels/pixels of the edge and result in an edge structure which isn't similar to any of the mentioned structures in Fig. 2.
(4) Four neighboring pixels of the edge and results in any structures as explained in Fig. 3.
Hereby any structure that doesn't fulfill at least one of the conditions and can't be grouped into any of these categories is not identified as a valid structure.

2.2 Calculation of the cost of the edge image

Edge image J (p, q) valuation at the individual segment (point) on the selected location k conspires of a calculated valuation of cost factors:

$$F(\mathrm{J}, k) = \sum_i w_i C_i(\mathrm{J}, k) \quad (1)$$

Accumulated value F(J) of the provided edge-image J equals to the total costs of points at every given pixel site of the examined image:

$$F(J) = \sum_i F(J, k) = \sum_i \sum_i w_i C_i(J, k) \quad (2)$$

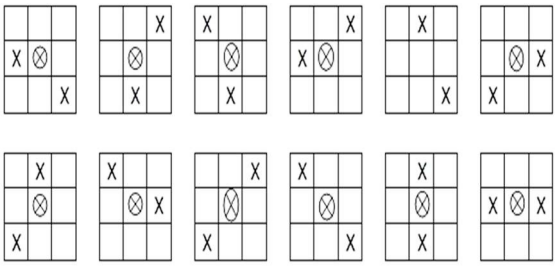


Fig 1. Acceptable bi-neighbor structures

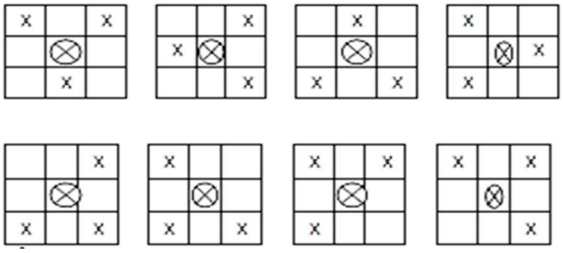


Fig 2. Acceptable tri-neighbor structures

Fig 3. Acceptable quad-neighbor structures

After considering these binary group present in edge images J_i and J_j whose structure is hard to distinguish despite the window *W(k)* which surrounds our site of interest *k*, conclude that it will result and give comparative cost/valuation function:-

$$\Delta F(J_i, J_j, k) = \sum_{w(k)} \sum_{g} w_g [\, C_g(J_i, k) - C_g(J_j, k)$$

$$= \sum_{w(k)} \sum_{g} \Delta C_g(J_i, J_j, k) \quad (3)$$

Where $0 \le C_g \le 1$ & $w_g \ge 0$.The $C_g J$ identifies itself as a factor affecting the value, $w_g J$ the weights related to our examined pixel. This window W*(k)* varies b/w sole pixel or a 9-pixel square matrix (i.e. a 3 by 3 matrix/array of pixels) around our pixel of interest *k*. When $\Delta F(J_i, J_j, k) < 0$, J_i conspires to be a preferable configuration despite when $\Delta F(J_i, J_j, k) > 0$, J_j is a preferable layout in our case. When the value of $\Delta F(J_i, J_j, k) = 0$ all the layouts tend to have an equal cost or show identical cost.

3. Cost Factors Linked with the Cost Function

The five cost factors to be considered are:

(1) $C_d(J, k)$ is the cost factor associated with region dissimilarity.
(2) $C_t(J, k)$ is the cost factor associated with edge thickness.
(3) $C_e(J, k)$ is the cost factor associated with edge curvature.
(4) $C_f(J, k)$ is the cost factor associated with edge fragmentation.
(5) $C_q(J, k)$ is the cost factor associated with the number of pixels detected on edges.

3.1 Cost factor associated with region dissimilarity

This cost factor is used to group the edge pixels on the boundary of the dissimilar regions and penalize the non-edge pixels. Dissimilarity enhancement is performed before computing this cost factor and the enhanced image is stored in $B = B(p, q)$ such that $0 \le B(p, q) \le 1$. Where the enhanced image B contains the probability of a pixel being edge pixel. The higher the value$B$, the more likely the pixel is an edge pixel. The region of interest is defined with respect to the basic edge structure. For each edge structure, a pair of dissimilar region R1, R2 can be defined using Fig 4.

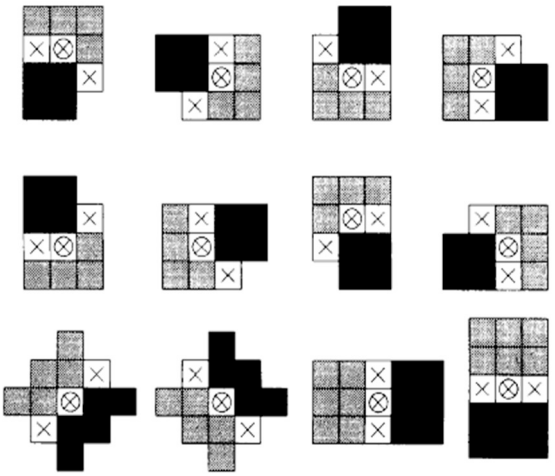


Fig 4. Region of interest for validation of two-neighbor edge structure

The enhanced image $B(k)$ can be obtained using the following steps:
(i) All pixels in $B(k)$ are initialized to zero
(ii) For each pixel k, operations (a) and (b) are performed

(a) For each centered pixel k, the difference in gray-level value between region R1 and R2 as shown in Fig 4 is calculated and degree of dissimilarity f(R1, R2) between these regions R1 and R2 are obtained. This degree of dissimilarity could be simply the average gray level difference between R1 and R2 or could be more complex measures depending upon the requirements. In this case, a simple average gray-level difference is considered. The edge structure having maximum objective f(R1, R2) value is considered as the best fitting edge structure.
(b) In this phase, the best-fitted edge structure is shifted to the new position identified by edge structure to perform non-maxima suppression. For horizontal, vertical and crossways structures, the edge is dislocated by one pixel in these directions to perform this operation known as shift operation. This operation is done by dislocating the location of edges in each of the 4(up, down, right, left) directions as shown in Fig 5 in case of other edge structures. New regions R1 and R2 are calculated at shifted edge structures and their fitness f(R1, R2) are obtained.

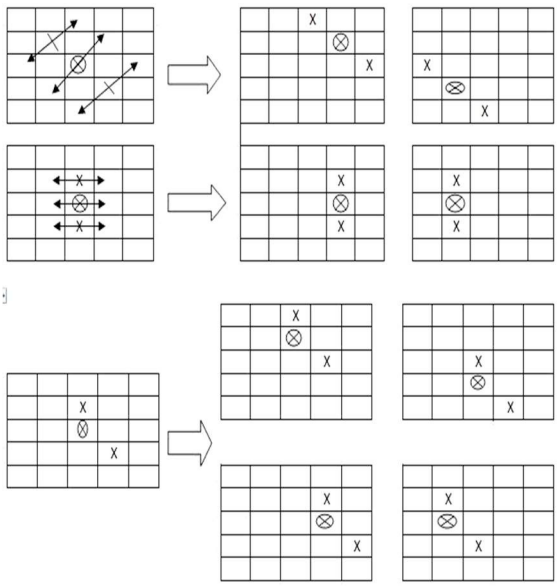
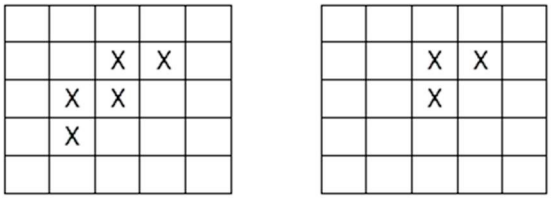


Fig 5. Examples of non-maximum suppression

3.2 Cost factor associated with an edge thickness

A pixel having multiple connections between neighboring pixels as shown in Fig. 6 is said to be a thick edge pixel.



(a) (b)

Fig. 6. (a) Examples of thick edge with 5 pixels; (b) with 3 pixels

3.3 Cost factor associated with edge curvature

The cost factor $C_c(J, \mathrm{k})$ related with edge bend bolsters straight edges and punished strange edges. A cost $C_c(J, \mathrm{k})$ is designated to a non-endpoint edge pixel at site k in the edge picture J (an endpoint is an edge pixel that has at most one neighboring edge pixel). On the off chance that there are several neighboring edge pixels in J related by methods for the pixel at site k with the true objective that the resulting edge structure turns by more than 45°, set $C_c(J, \mathrm{k}) = 1$. If there is no match of neighboring edge pixels that causes the consequent edge structure to turn by more than 45°, yet there exist several edge pixels that cause the resulting edge structure to turn by 45°, set $C_c(J, \mathrm{k}) = 0.5$. If there is no joining of neighboring edge pixels that causes the resulting edge structure to turn by more than 0° set $C_c(J, \mathrm{k}) = 0$.

3.4 Cost factor associated with edge fragmentation

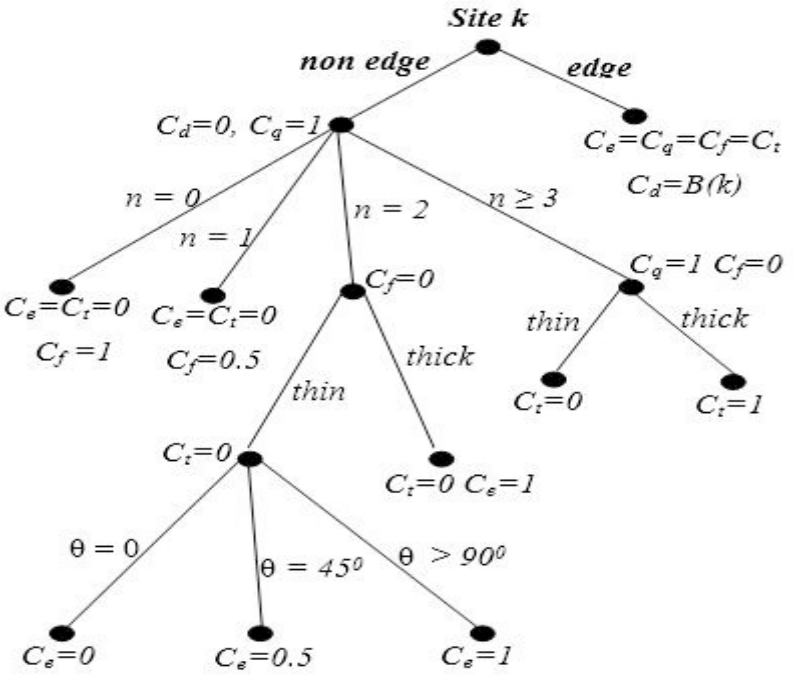
The cost factor $C_f(J, \mathrm{k})$, is intended for lessening the discontinuity of the subsequent edge structure. In the event that the edge pixel at site k in the edge picture J does not have any neighboring edge pixel, set $C_f(J, \mathrm{k}) = 1$. On the other hand in case the pixel of the edge at site k has just a single neighboring edge pixel, set $C_t(J, \mathrm{k}) = 0.5$. For the rest of the observations, fix $C_t(J, \mathrm{k}) = 0$.

3.5 Cost factor linked with no. of edge pixels identified

The cost factor $C_q(J, \mathrm{k})$ favors the discovery of edge pixels when $f(R_1, R_2)$ is non-zero, in this way promoting the identification of an extreme quantity of edge pixels. A cost factor $C_q(J, \mathrm{k})$ suppress this tendency. In the event that pixel k is an edge pixel, set $C_q(J, \mathrm{k}) = 1$ otherwise $C_q(J, \mathrm{k}) = 0$.

The last edge image has disparate and frequently opposing cost factors. Since the issue is an obliged optimization issue and the last edge image must be the one that can best satisfy all imperatives forced by cost variables. A decision tree is performed to register every one of the cost components. Fig 7 demonstrates a decision tree for deciding the cost variables where C_v v∈{d,t,e,f,q}at each pixel site k. In Fig 7, n connotes the number of neighboring edge pixels for a pixel site k and θ the edge related with a two neighbor edge structure as delineated in Section 2.1. The total expense of an edge picture J can be resolved using equation (1) where w_kJ are the weights related to comparing cost factors. The prescribed estimations of these weights are:

$w_d(\mathrm{J}, k)$=2.00, $w_q(J, k)$=1.00, $w_e(J, k)$={0.25,0.50,0.75}, $w_f(J, k)$={2.00,3.00,4.00},and $w_t(J, k)$= $2w_f(J, k)$ - $w_e(J, k)$+$w_d(J, k)$-$w_q(J, k)$ + 0.01.



Where θ = Angle linked to edge structure, *B(k)* = Area of variation count, n = No. of adjacent Pixels

Fig. 7 Decision tree for computing the cost function F(J, k)

4. Particle Swarm Optimization

4.1 *Particle Swarm Optimization*

Particle swarm optimization (PSO) was introduced by J. Kennedy and R. Eberhart in 1995 [14]. PSO shows the system that mimics the behavior of birds flocking. The birds fly in a solution space and the optimum solution is determined by their flocking behavior. Birds chase some path in order to find the optimum solution. The local or particle best solution is the shortest path followed by a bird. Particles are likely to move towards its personal best position ($pbest$) found by them so far. Particles also maintain track of global best position ($gbest$) that is the best shortest path found by any particle at a particular illustration. Each particle is linked with a velocity by which it gets accelerated towards local and global best path and the current position of the particle in d dimension space with respect to $gbest$ and $pbest$. Birds make communication with each other to achieve the most optimum (best) path to achieve an optimum solution. Therefore, birds discover from the experience of their personal best solutions and global best solutions. To detect global most favorable, the velocity and position of each particle are upgraded repeatedly with help of succeeding equations [15, 16]:

$$vel_{id}^{t+1} = w * vel_{id}^{t} + c_1 * rand * (pbest_{id}^{t} - p_{id}^{t}) + c_2 * rand * \left(gbest_{gd}^{t} - pop_{id}^{t}\right) \quad (4)$$

$$p_{id}^{t+1} = p_{id}^{t} + vel_{id}^{t+1} \quad (5)$$

$$w = (w_{min} - w_{max}) * \frac{Iter^{Max} - t}{Iter^{Max}} + w_{max} \quad (6)$$

Where t is the repetition (generation) figure, vel_{id}^{t} and p_{id}^{t} are the velocity and population of i^{th} particle in d dimension. w is the inertia constant, c1 and c2 are the coefficients of acceleration, w_{min} and w_{max} are the minimum

and maximum value of inertia constants. $rand \in [0,1]$ is the uniformly distributed random number. $pbest_{id}^{t}$ and $gbest_{id}^{t}$ are the local and global best of i^{th} particle in d dimension. Kennedy has referred $c_1 * rand * (pbest_{id}^{t} - pop_{id}^{t})$ as the cognitive component and $c_2 * rand * \left(gbest_{gd}^{t} - pop_{id}^{t}\right)$ as the social component respectively.

4.2 Binary Particle Swarm Optimization (BPSO)

The concept of binary particle swarm optimization (BPSO) is also given by Kennedy and Eberhart which allows BPSO to operate in binary space. In BPSO, a new approach is suggested to update the position of particle which takes either 0 or 1 in d^{th} dimension as:

$$p_{id}^{t+1} = \begin{cases} 0 & if\ rand() \geq Sig(vel_{id}^{t+1}) \\ 1 & if\ rand() < Sig(vel_{id}^{t+1}) \end{cases} \quad (7)$$

Where, $Sig(.)$ is the sigmoidal function which is used to transform the velocity into probability between [0, 1]. The sigmoidal function can be expressed as:

$$Sig(vel_{id}^{t+1}) = \frac{1}{1+e^{-vel_{id}^{t+1}}} \quad (8)$$

Algorithm 1 shows the pseudo-code of BPSO . It must be taken care of that the BPSO is responsive to sigmoid function congestion, which occurs in case values of velocity are either too huge or too small. When the velocity of the particle approaches the lower bound, the probability in the change in value comes near to zero, thereby limit exploration. On the other hand, when the velocity of the particle approaches the upper bound, the probability in the change in value comes near to one, thereby limit exploitation. A probability of 0.5 returns by the sigmoidal function when the velocity of the particle comes near to zero, it means there is 50% chances for the bit to flip. However, velocity clamping will delay the occurrence of the sigmoid function saturation. Hence, the optimal selection of velocity is important for faster convergence [15, 17].

f	Fitness function given in equation ()
$Iter^{Max}$	Maximum number of iteration
$c1, c2$	Acceleration coefficients
$pbest$	Local best solution of particle p
$gbest$	Global best solution
d	Dimension of search space
$rand$	Uniformly distributed random numbers between (0 and 1)

1. Input image
2. Input optimization parameters
3. vel= Velocity_Initialization ();
4. pop= Particle_Initialization ();

```
5. for i = 1 to Iter^Max
        for each particle p in pop
            obj = f(p)
            if obj is better than f(pbest)
                pbest = p
            End if
            if obj is better than f(gbest)
                gbest = p
            End if
        update velocity using equation (4)
        update position using equation (7)
        end for
end for
```

Fig. 8 Algorithm for Edge Detection using Binary Particle Swarm Optimization (BPSO)

5. Edge Detection Using Binary Particle Swarm Optimization

The population size (no. of particles) is taken as 200 and initially, all particles are randomly assigned the velocities between -6 to 6 in 256x256 dimensions (same as the size of image). The velocities are then transformed into a position using the BPSO updating mechanism. Bit value 1 represents edge while the bit value 0 represents non-edge. Therefore, 200 binary images or initial edge images of size (256,256) are generated. All cost factors are calculated for each particle and then final fitness value is obtained using equation (1). Since it is a cost minimization problem, therefore particle having minimum fitness value is considered as *gbest*. Velocities of each particle that following the *pbest* and *gbest* are updated in each iteration and then a new edge image is created. Again, the fitness value at the new edge image is calculated and it is compared against *pbest* and *gbest*. If current fitness is better than previous then update the previous solution. In this way, the optimum edge map can be obtained at the end of this iterative process. The whole process is iterated for 500 times.

6. Results and Discussion

"The proposed algorithm is tested on the Berkeley Segmentation Dataset [18], which is popular for edge detection and image segmentation work. BSD consists of natural images (of size 481×321 pixels) with ground truth provided." All the ground truth images are resized into 256x256 pixels for the experiment. The ground truth images of this dataset are combined from five to ten human observations so that edges in these images have fair judgment and is the strong reason to select this dataset for the experiment. The aim of the proposed approach is to minimize the error between edge pixels of ground truth image and that of obtained through the proposed approach. The subjective output of an image is difficult to approximate by an edge detector due to differences in human observations. Therefore, the BSD image dataset is mainly used for boundary and contour detection [21, 22]. For this reason, only those edge pixels obtained through the proposed approach are considered as a boundary which is matching with the ground truth image. The rest of the edge boundary is not considered for the evaluation of parameters like Precision, Recall, and F score.

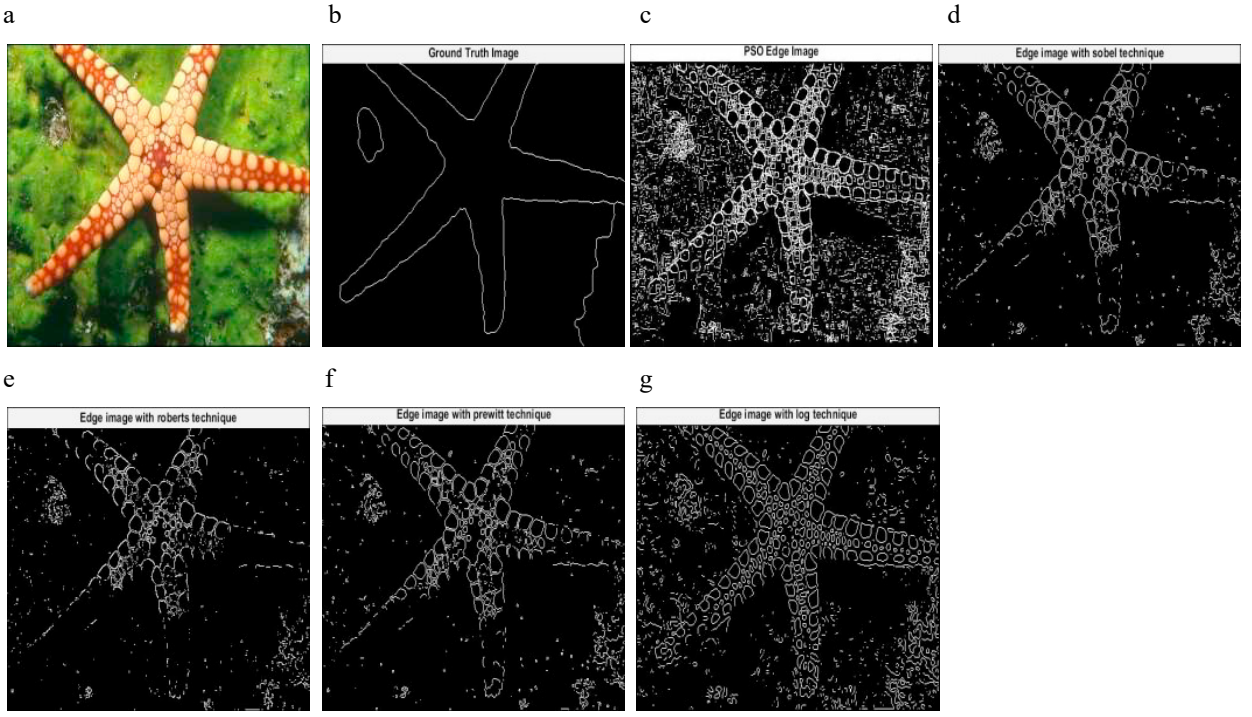


Fig. 9 (a) Test image (12003) from the BSD500 dataset; (b) Ground truth image; (c) detected image by BPSO (Binary Particle Swarm Optimisation); (d)Canny; (e) Prewitt; (f) GA (Genetic Algorithm); (g) ACO(Ant Colony Optimization)

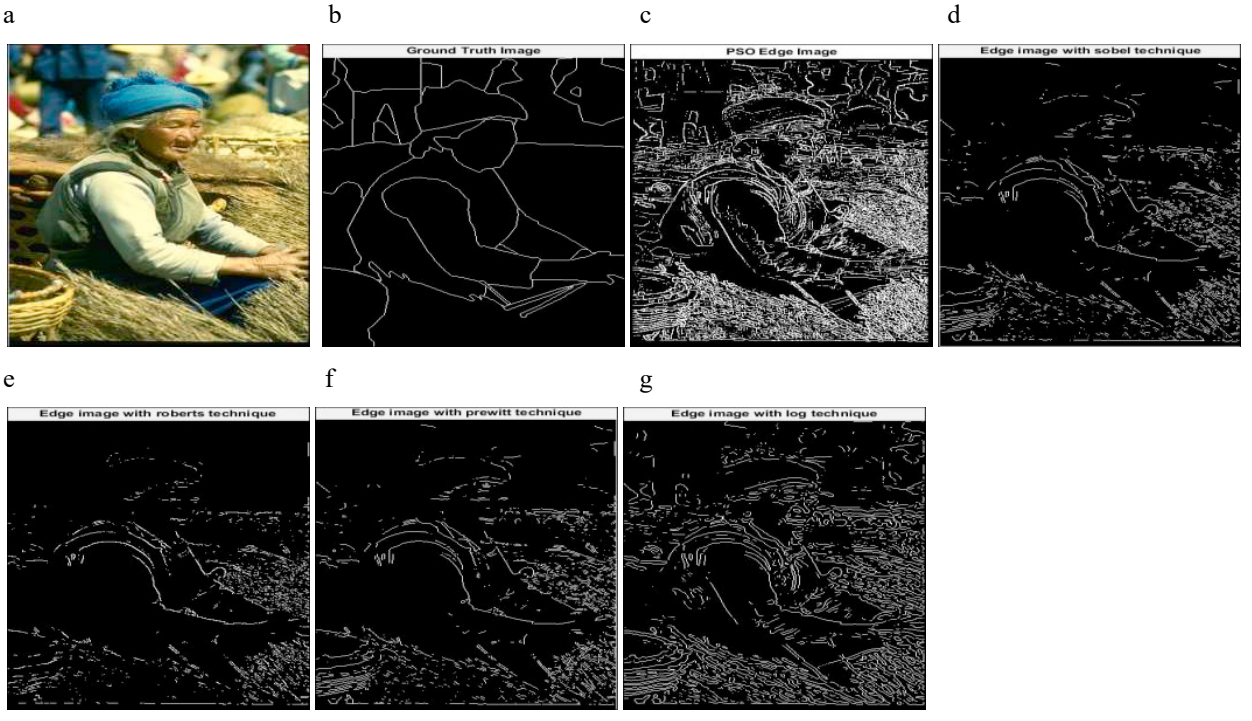


Fig. 10 (a) Test image (15004) from the BSD500 dataset; (b) Ground truth image; (c) detected image by BPSO (Binary Particle Swarm Optimisation); (d)Canny; (e) Prewitt; (f) GA (Genetic Algorithm); (g) ACO(Ant Colony Optimization)

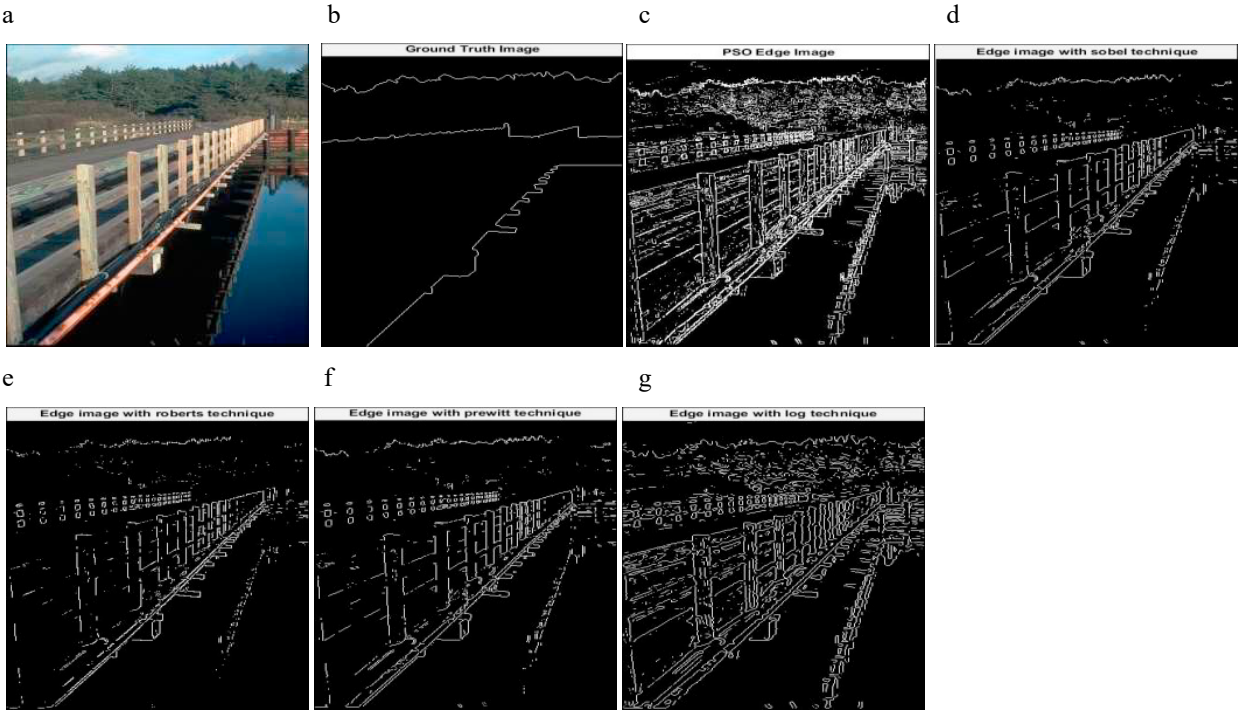


Fig. 11 (a) Test image (22013) from the BSD500 dataset; (b) Ground truth image; (c) detected image by BPSO (Binary Particle Swarm Optimisation); (d)Canny; (e) Prewitt; (f) GA (Genetic Algorithm); (g) ACO(Ant Colony Optimization)

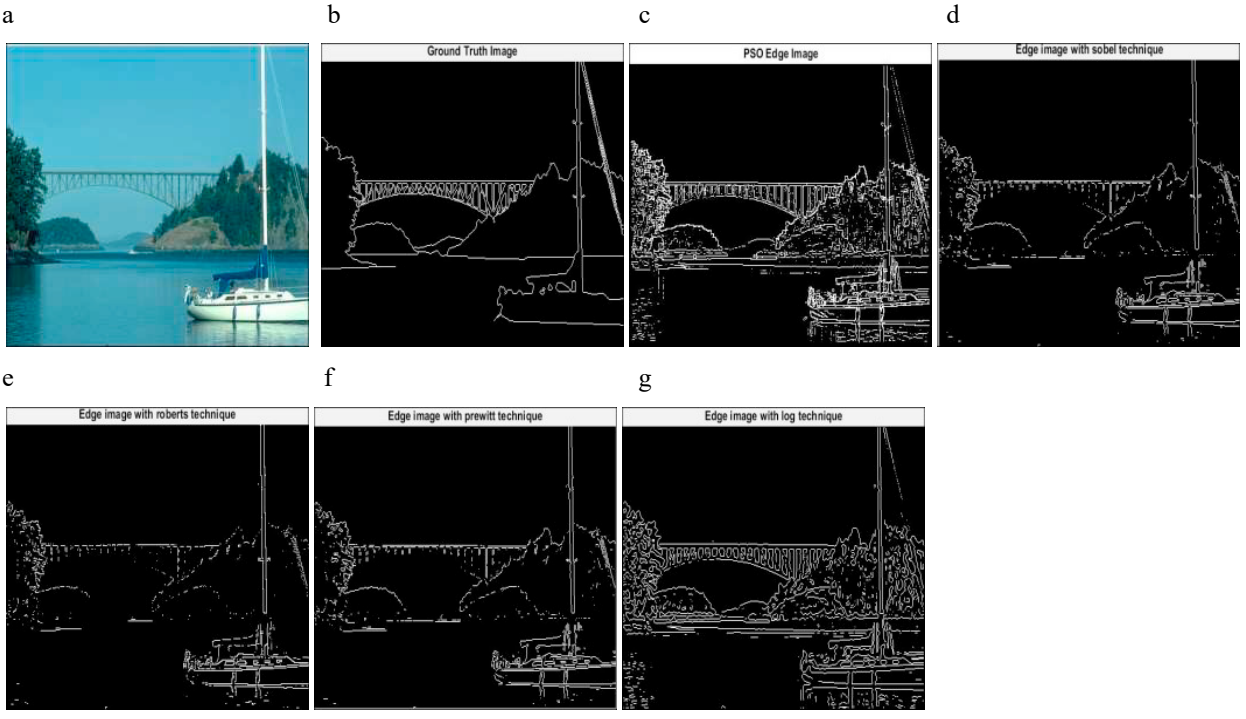


Fig. 12 (a) Test image (22090) from the BSD500 dataset; (b) Ground truth image; (c) detected image by BPSO (Binary Particle Swarm Optimisation); (d)Canny; (e) Prewitt; (f) GA (Genetic Algorithm); (g) ACO(Ant Colony Optimization)

Only 500 images from the BSD image dataset is taken to perform the experiment due to redundancies in the dataset. Three performance metric Precision, Recall, and F score is considered to calculate the performance of the proposed approach on these 500 images. The performance of the proposed approach is evaluated by comparing the edge maps obtained through the proposed approach and that of the edge pixels of the ground truth image. Performance metric Precision, Recall, and F score is also calculated for classical edge detectors like Sobel, Robert, Prewitt, and Log and compared against the proposed approach.

The five major categories of techniques are compared which are BPSO, ACO, GA, Canny & Prewitt. All the above techniques are compared for the same dataset classification [22]. The 500 test images are used from the "BSD500" dataset for the implementation of the concept. For comparison, the accuracy performance parameters like "Precision, Recall, and F-Score" are taken to get the different feature results from all perspectives.

Eight images from the BSDS500 datasets are taken. Images are numbered as 12003, 15004, 22013, 22090, 22093, 23080, 24004, 277095 respectively and the images are partitioned to obtain the ground truth images for reference edge images. The results of the four images are shown in this paper due to the limitation of the space.

The Ground truth images are taken from the BSD500 dataset. To get the idea of how the ground truth images are computed to refer the paper [21, 22] in the references.

Table 2 Test Performance (F, Recall and Precision) for BPSO, ACO, GA, Prewitt, Canny Edge Detectors on different Images

Image	Operator	Precision	Recall	F- Score
12003	**BPSO**	**0.4135**	**0.4463**	**0.4292**
	ACO	0.1218	0.2283	0.1586
	GA	0.0563	0.1625	0.0837
	Prewitt	0.0359	0.1261	0.0559
	Canny	0.0365	0.1305	0.0570
15004	**BPSO**	**0.2748**	**0.2253**	**0.2476**
	ACO	0.1154	0.1308	0.1226
	GA	0.0750	0.0927	0.0829
	Prewitt	0.0523	0.0892	0.0660
	Canny	0.0503	0.1820	0.0641
22013	**BPSO**	**0.3374**	**0.3211**	**0.3290**
	ACO	0.3103	0.2905	0.3001
	GA	0.0879	0.3272	0.1386
	Prewitt	0.0434	0.1919	0.0708
	Canny	0.0430	0.1934	0.0704
22090	**BPSO**	**0.2457**	**0.1872**	**0.2125**
	ACO	0.1097	0.2262	0.1478
	GA	0.2404	0.1851	0.2092
	Prewitt	0.1727	0.1685	0.1706
	Canny	0.1721	0.1689	0.1705

Table 3 Test Performance (F, Recall and Precision) for BPSO, ACO, GA, Prewitt, Canny Edge Detectors on different Images

Image	Operator	Precision	Recall	F- Score
22093	**BPSO**	**0.3187**	**0.3512**	**0.3342**
	ACO	0.2154	0.2271	0.2211
	GA	0.2992	0.3373	0.3171
	Prewitt	0.2010	0.2624	0.2276
	Canny	0.2027	0.2643	0.2295
23080	**BPSO**	**0.2542**	**0.2355**	**0.2445**
	ACO	0.1287	0.1685	0.1460
	GA	0.0559	0.1398	0.0799
	Prewitt	0.0404	0.1223	0.0607
	Canny	0.0413	0.1272	0.0624
24004	**BPSO**	**0.2354**	**0.2727**	**0.2567**

	ACO	0.2153	0.2601	0.2356
	GA	0.0279	0.1653	0.0478
	Prewitt	0.0276	0.2118	0.0488
	Canny	0.0274	0.2137	0.0486
277095	**BPSO**	**0.2865**	**0.2495**	**0.2667**
	ACO	0.1963	0.1814	0.1886
	GA	0.0417	0.0703	0.0523
	Prewitt	0.0376	0.1095	0.0560
	Canny	0.0375	0.1099	0.0560

Variable Settings:
The variable values for "BPSO" are: population size 200; maximum generation 500; acceleration coefficients c1 & c2 2.05; minimum and maximum velocity -6 to 6.

The test performance graphs for table 2, 3 shows that the BPSO technique shows better results as compared to all other techniques used in this paper.

a b

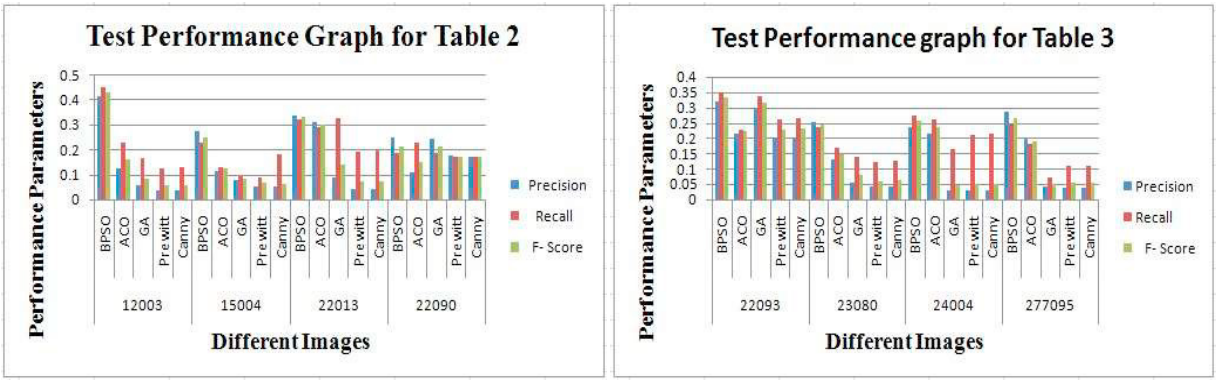


Fig. 13 (a) Test performance graphs for table 2; (b) Test Performance graph for table 3

Table 2 and 3 shows the comparison results of Precision, Recall, and F score on 8 different images and best results obtained through any of the approaches (BPSO (Proposed), LOG, Robert, Prewitt, and Sobel) is a mark in bold. The edge map of these approaches is also shown in Fig 9, 10, 11, 12. It is clearly observed from Table 2, 3 that the proposed approach overall outperforms all other edge detection approaches when compared against all performance metrics. The mean and standard deviation of all approaches when tested on 500 BSD test images are shown in Table 4. It is also clearly visible from Table 3 that BPSO also outperforms all other edge detection techniques when tested on 500 test images. After BPSO, ACO and GA hold good overall performance while Prewitt and Canny perform worst due to low precision even though Prewitt & Canny maintained a good recall rate.

Table 4 Mean & Standard Deviation test performance of all edge detectors on 500 BSD test images

Edge Detector	Precision (Mean ± Std)	Recall (Mean ± Std)	F-Score (Mean ± Std)
BPSO	**0.3021 ± 0.0531**	**0.2863 ± 0.0794**	**0.2901 ± 0.0701**
ACO	0.1712 ± 0.0711	0.2105 ± 0.0568	0.1920 ± 0.0519
GA	0.1101 ± 0.1025	0.1847 ± 0.0974	0.1246 ± 0.0961
Prewitt	0.0768 ± 0.0625	0.1609 ± 0.0610	0.0921 ± 0.0673
Canny	0.0798 ± 0.0758	0.1585 ± 0.0587	0.0977 ± 0.0652

Maximum Precision, Recall, and F score value of all approaches are shown in Table 5. The maximum value of all performance metrics for Prewitt and canny is lower than the average performance of the proposed approach. In

contrast, the maximum value of the performance metric for ACO and GA performs better than the average performance of the proposed approach.

In Table 5 BPSO returns high-performance metrics on 12003 image, ACO on 220013 images, and GA, Prewitt, Canny on 22093. Therefore, it is not necessary that all edge detection methods return high precision, recall, and F scores on the same image. If one method returns high F score value on an image then it is possible that others could perform worst on that image. As for example, on 12003 images the F score value of BPSO is 0.4292 while F scores value of ACO, GA, Prewitt, and Canny lagged by 63.05%, 80.50%, 86.98%, and 86.72%.

Table 5 Maximum value of performance metrics on all 500 BSD test images

Edge Detector	Image Name	Precision	Recall	F-Score
BPSO	**12003**	**0.4135**	**0.4463**	**0.4292**
ACO	22013	0.3103	0.2905	0.3001
GA	22093	0.2992	0.3373	0.3171
Prewitt	22093	0.2010	0.2624	0.2276
Canny	22093	0.2027	0.2643	0.2295

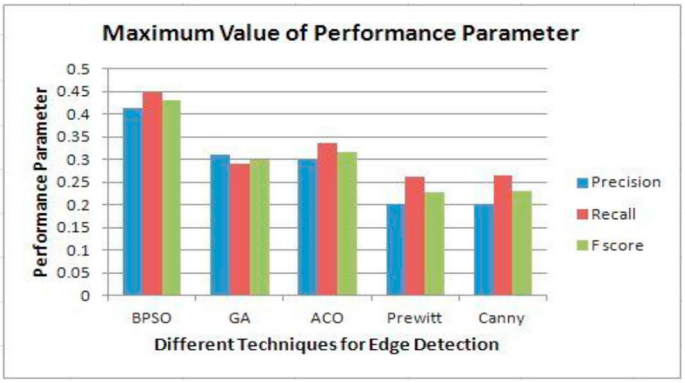


Fig. 14 Maximum value of performance parameter

7. Conclusion & Future Scope

In this paper, the Binary Particle Swarm Optimization (BPSO) based edge detection technique is proposed by minimizing the multi-objective cost function. The proposed approach is examined on 500 BSD images and obtained results are compared with classical edge detection techniques such as "Canny, Prewitt and with computational intelligence techniques such as ACO and GA. Classical edge detectors fail to maintain performance consistency, if an edge detector performs well on one image then it could perform worst on other images. Since edge detection is extensively used in computer eyesight applications, therefore, the selection of perfect edge detector for that specific task is challenging and time taking process. However, the proposed approach solves this inconsistency problem and performs well on all 500 test images. The average F score of the proposed approach is 0.2901 and has little standard deviation (0.0401). This small standard deviation shows the consistency of the proposed approach. In contrast, Canny shows better consistency among classical edge detectors having an average F score of 0.1901 and a standard deviation of 0.0591. Experimental results indicate that the Computational Intelligence-based techniques actively increase the edges of the image as compared to conventional techniques such as Robert, Prewitt, and Canny but the processing time taken by computational techniques is much more than the conventional techniques thus reducing the processing time is future study.

References

[1] L. S. Davis (1975) "A survey of edge detection techniques.", *Computer Graphics and Image Processing,* vol. 4, pp. 248–270.

[2] S. M. Bhandarkar, Y. Zhang and W. D. Potter (1994) "An edge detection technique using genetic algorithm-based optimization.", *Pattern Recognition*, vol. 27, pp. 1159–1180.

[3] V. Torte and T. A. Poggio (1986) "On edge detection.", *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8, pp. 147-163.

[4] D. Marr and E. Hildreth (1980) "Theory of edge detection." *Proceedings of the Royal Society of London - Biological Sciences*, vol. 207, pp. 187–217.

[5] S. Chaudhuri, S. Chatterjee, N. Katz, M. Nelson and M. Goldbaum (1989) "Detection of blood vessels in retinal images using two-dimensional matched filters.", *IEEE transactions on medical imaging.* Vol. 8, pp. 263–269.

[6] T. Peli and D. Malah (1992) "A study of edge detection algorithms.", *Computer Graphics and Image Processing*, vol. 20, pp. 1–21.

[7] N. S. Dagar, P. K. Dahiya (2016) "Soft Computing Techniques for Edge Detection Problem: A state-of-the-art review.", *International Journal of Computer Applications (0975 – 8887), Volume* 136 – No.12.

[8] N. S. Dagar, P. K. Dahiya (2018) "A Comparative Analysis of Edge Detection Algorithm and Performance Metric Using Precision, Recall and F- Score.", *International Journal of Modern Electronics and Communication Engineering (IJMECE)* ISSN: 2321-2152 Volume No.-6, Issue No.-6

[9] N. S. Dagar & P. K. Dahiya (2019) "A Comparative Investigation into Edge Detection Techniques Based on Computational Intelligence." *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, Vol.11, No.7, pp. 58-68

[10] N. S. Dagar & P. K. Dahiya (2019) "Edge Detection of Different Images Using Soft Computing Techniques.", *International Journal of Recent Technology and Engineering (IJRTE)* ISSN: 2277-3878, Volume-8, Issue-2.

[11] N. S. Dagar & P. K. Dahiya (2019) "Edge Detection Using Distinct Particle Swarm Optimization (DPSO).", *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* ISSN: 2278-3075, Volume-8, Issue-9.

[12] N. R. Pal and S. K. Pal (1993) "A review on image segmentation techniques.", *Pattern Recognition*, vol. 26, pp. 1277–1294.

[13] J. Canny (1986) "A computational approach to edge detection.", *IEEE Trans. Pattern Anal. Mach. lntell. PAMI*-8, pp. 679-698.

[14] Li Jing, Huang Peikang, Wang Xiaohu & Pan Xudong (2009) "Image Edge Detection Based On Beamlet Transform.", *Journal of Systems Engineering and Electronics*, vol. 20, no. 1, pp. 1-5.

[15] M. Sharifi, M. Fathy and M. T. Mahmoudi (2002) "A classified and comparative study of edge detection algorithms.", *Proceedings International Conference on Information Technology: Coding and Computing*, pp. 5–8.

[16] H. L. Tan, S. B. Gelfand and E. J. Delp (1989) "A comparative cost function approach to edge detection.", *IEEE Trans. Systems Man Cybern,* Vol. 19, pp. 1337-1349.

[17] J. Kennedy, R. Eberhart (1995) "Particle swarm optimization.", *IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948.

[18] S. L. Gupta, A. S. Baghel and A. Iqbal (2018) "Threshold Controlled Binary Particle Swarm Optimization for High Dimensional Feature Selection.", *International Journal of Intelligent Systems and Applications(IJISA)*, Vol.10, No.8, pp.75-84.

[19] S. L. Gupta, A. S. Baghel and A. Iqbal (2018) "Big Data Classification Using Scale-Free Binary Particle Swarm Optimization.", *Harmony Search and Nature Inspired Optimization Algorithms, ICHSA, Springer*, pp.1177-1187.

[20] M. Braik, A. Sheta and A. Ayesh (2017) "Image Enhancement Using Particle Swarm Optimization.", *Proceedings of the World Congress on Engineering*, pp. 1-6.

[21] D. Martin, C. Fowlkes, and J. Malik (2004) "Learning to detect natural image boundaries using local brightness, color, and texture cues.", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549.

[22] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik (2011) "Contour Detection and Hierarchical Image Segmentation.", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 33, issue-5.

[23] W. Fu, M. Johnston and M. Zhang (2014) "Low-Level Feature Extraction for Edge Detection Using Genetic Programming.", *IEEE Transactions On Cybernetics*, Vol. 44, NO. 8.