Edge Detection using Evolutionary Algorithms

Major Project submitted in partial fulfillment of the requirements for the award of degree of

Master of Technology

in

Information Systems

Submitted By:

Somya Jain

(13/IS/09)

Under the Guidance of:

Prof. O. P. Verma

(HOD, IT Department)

Department of Information Technology
Delhi Technological University
Bawana Road, Delhi – 110042
(2009-2011)

CERTIFICATE

This is to certify that **Mr. Somya Jain (13/IS/09)** has carried out the major project titled "**Edge Detection using Evolutionary Algorithms**" as a partial requirement for the award of Master of Technology degree in Information Systems by Delhi Technological University.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2009-2011**. The matter contained in this report has not been submitted elsewhere for the award of any other degree.

(Project Guide)

Prof. O. P. Verma

Head of Department

Department of Information Technology

Delhi Technological University

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I express my gratitude to my major project guide **Prof. O. P. Verma**, HOD, IT Dept., Delhi Technological University, for the valuable support and guidance he provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for his constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Somya Jain
Roll No. 13/IS/09
M.Tech (Information Systems)
E-mail: somya1124@gmail.com

Abstract

The problem of edge detection in digital images is addressed using two methods. The first method proposed for edge detection in color images is based on fuzzy logic, and evolutionary learning techniques such as bacterial foraging, particle swarm optimization, genetic algorithm and gravitational search algorithms. In this method a circular mask is applied separately on R, G, and B components of RGB image to compute USAN area about every pixel in the image leading to the USAN area image. Histogram based Gaussian membership function is used to detect strong edges and the modified bell shaped membership function is used to detect weak edges from this image after fuzzification. Fuzzy entropy and edge sharpness factor involved in the objective function and the parameters that control the shape and range of modified bell shaped membership function are optimized using different evolutionary techniques. The edge maps of the three color components are obtained by subjecting the fuzzified USAN area images of color components to adaptive threshold. The final edge map is obtained by combining the edge maps of the above three color components. The second method proposed for grayscale images is based on fuzzy derivative and evolutionary algorithms. The derivatives in all eight directions for edge pixel are computed using if-then rules. Median of the derivative set is taken to obtain derivative matrix which is fuzzified using bell-shaped membership function. Another membership function HIGH acting as hedge is used to locate the edges in the image in the fuzzy domain. Fuzzy entropy involved in the objective function and parameters that control the shape and range of bell shaped and HIGH membership functions are optimized using different evolutionary algorithms. Adaptive threshold is applied to fuzzified image to obtain the final edge map. Visual comparison is performed between the experimental results of the two methods to understand their efficacy.

Table of Contents

Certificate……………………………………..………………………….i

Acknowledgement……………………………….……………………ii

Abstract…………………………..…………………………......iii

Chapter 1: Fuzzy Image Processing…………………………...….1

Chapter 2: Edge Detection…………………………………………...4

2.1 Introduction……………………………………………………………..4

2.2 Applications………………………………………………………..…8

Chapter 3: Evolutionary Algorithms…………………………...…9

3.1 Bacterial Foraging……………………………………………...….9

3.2 Genetic Algorithm………………………………………………...…11

3.3 Particle Swarm Optimization……………………………………...…12

3.4 Gravitational Search Algorithm……………………………….…………..14

Chapter 4: Proposed Approach…………………………….…………18

4.1 SUSAN principle based Method……………………………………...…18

4.1.1Computation of USAN area……………………………………………………18

4.1.2 Fuzzification……………………………………………………………..20

4.1.3 Fuzzy Measures and Optimization……………………………………………….21

4.1.3.1 Fuzzy Measures……………………………………………………...…21

4.1.4 Definition of Fuzzy Entropy………………………………………………..23

4.1.5 Sharpness Factor………………………………………………………………23

4.1.6 Optimization of an Objective Function……………………………………………..24

4.1.7 Initialization of parameters of the Evolutionary Algorithms …….….…………………..24

4.1.7.1 BF Parameters……………………………………………………………………24

4.1.7.2 GA Parameters……………………………………………………………………25

4.1.7.3 PSO Parameters…………………………………………………………………..25

4.1.7.4 GSA Parameters…………………………………………………………………..25

4.1.8 Adaptive Thresholding…………………………………………………...………26

4.1.9 An Optimal Fuzzy SUSAN principle based Algorithm for Edge Detection……………….27

4.2 Fuzzy Derivative based Method…………………………………………..….31

4.2.1 Computation of Derivative value for a pixel……………………………………..…31

4.2.2 Fuzzification ………………………………………..……………......33

4.2.3 Parameters and Objective Function Optimization…………………………………..34

4.2.4 Initialization of parameters of the Evolutionary Algorithms ……….……………………..35

4.2.4 .1 BF Parameters…………………………………………………………………...35

4.2.4.2 GA Parameters……………………………………………………………………36

4.2.4.3 PSO Parameters…………………………………………………………………..36

4.2.4.4 GSA Parameters………………………………………………………………..…36

4.2.5 An Optimal Fuzzy Derivative based Algorithm for Edge Detection……………….……37

Chapter 5: Experimental Results…………………………………...39

5.1 Results for SUSAN principle based Method………………………………..…39

5.2 Results for Fuzzy Derivative based Method……………………………………57

5.3 Comparison between SUSAN principle and Fuzzy Derivative based Methods...65

Chapter 6: Conclusion……………………………………………………70

References…………………………………………………………72

Chapter 1
Fuzzy Image Processing

Fuzzy image processing is the collection of different fuzzy approaches to image processing that understand, represent and process the images, their segments and features as fuzzy sets. The representation and processing depend on the selected fuzzy technique and on the problem to be solved.

Fuzzy image processing has three main stages: image fuzzification, modification of membership values, and, if necessary, image defuzzification.

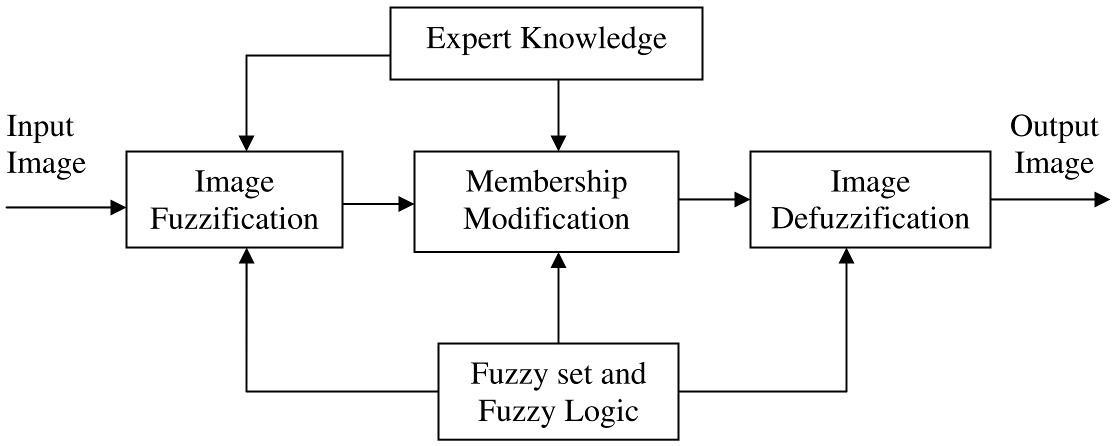


Fig.1.1**:** The General Structure of Fuzzy Image Processing.

The fuzzification and defuzzification steps are due to the fact that we do not possess fuzzy hardware. Therefore, the coding of image data (fuzzification) and decoding of the results (defuzzification) are steps that make possible to process images with fuzzy techniques. The main power of fuzzy image processing is in the middle step (modification of membership values. After the image data are transformed from gray-level plane to the membership plane (fuzzification), appropriate fuzzy techniques modify the membership values. This can be a fuzzy clustering, a fuzzy rule-based approach, and a fuzzy integration approach and so on.

There are many reasons to use fuzzy logic in image processing:

• Fuzzy techniques are powerful tools for knowledge representation and processing.

• Fuzzy techniques can manage the vagueness and ambiguity efficiently.

• In many image processing applications, we have to use expert knowledge to overcome the difficulties (e.g. object recognition, scene analysis). Fuzzy set theory and fuzzy logic offer us powerful tools to represent and process human knowledge in form of fuzzy if-then rules. On the other side, many difficulties in image processing arise because the data/ tasks/results are uncertain. This uncertainty, however, is not always due to the randomness but to the ambiguity and vagueness. Beside randomness which can be managed by probability theory we can distinguish between three other kinds of imperfection in the image processing:

i. Grayness ambiguity

ii. Geometrical fuzziness

iii. Vague (complex/ill-defined) knowledge

• General observations about fuzzy logic are:

➢Fuzzy logic is conceptually easy to understand. The mathematical concepts behind fuzzy reasoning are very simple.

➢Fuzzy logic is flexible. With any given system, it's easy to manage it or layer more functionality on top of it without starting again from scratch.

➢Fuzzy logic is tolerant of imprecise data. Everything is imprecise if we look closely enough, but more than that, most things are imprecise even on careful inspection. Fuzzy reasoning builds this understanding into the process rather than tacking it onto the end.

➢Fuzzy logic can model nonlinear functions of arbitrary complexity. We can create a fuzzy system to match any set of input-output data.

➢Fuzzy logic is based on natural language. The basis for fuzzy logic is the basis for human communication. This observation underpins many of the other statements about fuzzy logic.

➢Fuzzy logic can be blended with conventional control techniques. Fuzzy systems don't necessarily replace conventional control methods. In many cases fuzzy systems augment them and simplify their implementation.

Chapter 2
Edge Detection

2.1 Introduction

Edge detection is a fundamental problem in image analysis. Edge detection is an important early vision process for reducing the amount of data in the raw images to facilitate the high-level image processing tasks, since edges can delineate shape of the objects and contain the most information. An edge in an image is the boundary between two regions that are characterized by the sharp variations in the image intensity values and can be determined directly by processing of image in the spatial domain, or by transformation to a different domain. It is the boundary between two dissimilar regions in an image. Edges may even contain other edges, when look closer. An edge is easy to detect since differences in pixel intensity values between two regions are relatively easy to calculate. To determine the edges pixel, it is necessary to detect the position between two pixels. Image obtained after edge detection is a skeleton of the input image. Many of the important features can be extracted from the edges of an image (e.g., corners, lines, curves). Precise edge detection is required for the image analysis, evaluation and recognition techniques.

In the past, a lot of research has been done in the area of image segmentation in various applications using edge detection. The separation of a scene image into object and background, by tracing the edge between them is an important step in image interpretation. In other words, edge detection provides an effective way of segmenting the image into meaningful regions.

The underlying idea of most edge detection techniques is by the computation of a local first or second derivative operator, followed by some regularization technique to reduce the effects of noise. Edge detection methods can be classified into directional and non-directional or gradient-based operators. Directional operators use two masks and two convolutions, while non-directional use single mask and convolution but they are sensitive to noise due to gradient nature of the operators.

Many edge detection techniques are reported in the literature. Some of the earliest operators to detect edges in an image were proposed by Sobel, Prewitt and Roberts [1]. These operators use local gradient methods to detect edges under a specified direction. One of the biggest problems in these edge detectors is that they do not perform properly under noisy conditions. Canny [2] tried to overcome this problem by convolving the image with the first order derivatives of Gaussian filter for smoothing in the local gradient direction followed by edge detection using thresholding. Marr and Hildreth [3] have developed an algorithm that finds edges at the zero-crossings of the Laplacian of an image. Yuqin *et al.* [4] determine the probable position of image edge using Canny operator and then the edge detection is done by a threshold obtained from the maximal variance between-class method. It may be noted that methods involving Gaussian filtering suffer from problems such as edge displacement, vanishing edges and false edges [5]. Salinas *et al.* [6] suggest regularization as a way to fuse the outputs of the three separate edge maps found using Canny edge detector at the same time applying “well posedness” constraint to the inherently ill-posed problem of edge detection. Perona *et al.*[7] develop a technique to obtain a high quality edge detector by using anisotropic diffusion process. The method of Jiang *et al.* [8], involving a simple scan line approach, is not able to detect all the edges due to limitation of the methodology to trace only the horizontal and vertical neighbors of a point.

In the recent years, fuzzy techniques have found favor for the edge detection [9-10]. For instance, a local approach based on the fuzzy logic is used in [11] for morphological edge extraction method. Ho *et al.* [12] need both global and local image information for the categorization and classification of edges. Jinping *etal.* [13], present a fuzzy edge detection algorithm that overcomes the drawbacks of Pal algorithm[14] by using a new enhancement operator based on Gaussian function. Some improved edge detection algorithms utilize fuzzy enhancement [15-16] leading to fuzzy edge detection. Melin *et al.* [17] method improves the morphological gradient based approach to edge detection using type-1 fuzzy inference system and an interval type-2 fuzzy inference system. In another fuzzy edge detection algorithm [18], the complex transformation of the fuzzy space is undertaken using the enhancement algorithm based on gray scale fuzzy sets. This algorithm relies on increasing the variance between clusters to determine a threshold for the enhancement function. A fast multilevel fuzzy enhancement edge detection [19] overcomes the deficiency of enhancing some edges at the expense of the weakening other edges. However, when the contrast between the target and the background is weak as in a noisy image, this algorithm renders the edge detection most defective. Xiangtao *et al.* [20] have analyzed these problems and obtained an optimal threshold that grows the region of the fuzzy enhancement pixels and adopt a fuzzy entropy operator to detect edges. The method of Hamid [21] finds the rough edge map of gray scale image in a short time**.** The detection of edges using the weighted morphological operators and fuzzy morphological structuring elements is given in [22]. In another interesting work, Wafa *et* *al.* [23] compute the gradient and standard deviation for each pixel to obtain two edge sets that serve as inputs to the fuzzy system to decide whether a pixel belongs to the edge pixel or not. Yishu *et al.* [24] use both multiscale wavelet transform and fuzzy c-means clustering algorithm to obtain the edge map of the image adaptively.

Application of the evolutionary algorithms [25, 26, 27] to edge detection is getting attention in the recent years. In [28], Particle Swarm Optimization (PSO) is used to detect edges and corners as low level features from the noisy images for the recognition of simple objects. Khalid *et al.* [29] combine the fuzzy heuristic edge detection technique with the particle swarm optimization algorithm. By using Genetic Algorithm (GA) based approach [30] for edge detection, the chromosomes in the population are represented as binary arrays in which 1 represents an edge pixel and 0 represents a non-edge pixel. Ant Colony Optimization (ACO) is another swarm intelligence technique given by Dorigo *et al.* [31]. For instance in [32] edges are detected in digital images by placing artificial ants on the image and considering intensity differences between image pixels as heuristic information for the ant colony system. In [33] ACO has been used for edge enhancement. Each pixel of the image is assumed to be connected with its 8 neighborhood pixels. The ants are placed on the endpoints that are extracted from the already edge-detected image. ACO attempts to fix breaks of edges and extend the searching range of the ants to find the promising edges.

A recent algorithm in [34] detects an edge map based on the combination of bacterial foraging and probabilistic derivative technique derived from the ant colony systems. In another work by Yoshimura *et al.* [35] candidate edge regions are selected and the genetic algorithm is applied to decide the optimum edge regions in the texture image with randomness. Rashedi *et al.*[36] proposed a new population based heuristic search algorithm named Gravitational Search Algorithm (GSA). This method is yet to discover its potential on fields like edge detection. However, prior to GSA, work on edge detection was done based on the theory of universal gravity [37] in which every pixel is assumed to be an object, having some relationship with other pixels within its neighborhood through gravitational forces. For each pixel, the magnitude and the direction of the vector of the sum of all the gravitational forces the pixel exerts on its neighborhood, conveys the vitally important information about an edge structure.

2.2 Applications

Some of the applications of edge detectors are:

- ➢ Robotics, computer vision and image processing applications:
- Contour feature extraction.
- Object tracking and recognition.
- ➢ In the area of biometrics:
- Face detection
- Fingerprint recognition
- Iris identification etc.
- ➢ Machine vision gauging applications:
- Inspection for missing parts
- Measurement of dimensions of critical parts using gauging.
- ➢ Medical Imaging applications:
- X-Ray
- CT Scan

Chapter 3
Evolutionary Algorithms

3.1 Bacterial Foraging

Foraging can be modeled as an optimization process where bacteria seek to minimize the effort spent per unit time in foraging. In this scheme, an objective function is posed as the effort or a cost incurred by the bacteria in search of food. A set of bacteria tries to reach an optimum cost by following four stages such as chemotaxis, swarming, reproduction, and elimination and dispersal. There will be as many solutions as the number of bacteria. However, to arrive at an optimum path in the minimum time (i.e., convergence of solution path), we must have a sufficient number of bacteria. The bacteria have to follow four steps in the course of foraging [25].

In the chemotaxis stage, the bacteria either resort to a tumble followed by a tumble or make a tumble followed by a run or swim. On the other hand, in swarming, each *E. coli* bacterium signals another via attractants to swarm together. Furthermore, the least healthy bacteria die during the reproduction, whereas each of the healthiest bacteria splits into two, which are placed at the same location. While in the elimination and dispersal stage, any bacterium from the total set can be either eliminated or dispersed to a random location during the optimization. This stage helps the bacteria avoid the local optimum. Out of many bacteria engaged in foraging (several solution paths), some come out successful in achieving an optimum cost (an optimum solution).

Let θ be the position of a bacterium and $J(\theta)$ be a function of θ representing the value of the objective function; then, the conditions $J(\theta) < 0$, $J(\theta) = 0$, and $J(\theta) > 0$ indicate whether the bacterium at location θ is in nutrient-rich, neutral, and noxious environments, respectively. Basically a chemotaxis is a foraging behavior that implements a type of optimization.

Here, bacteria climb up the nutrient concentration (find the lower values of $J(\theta)$), avoid the noxious substances, and search for ways out of neutral media (avoid being at positions θ where $J(\theta) \geq 0$).

The four basic iterative steps of BF optimization are described below:

1) Chemotaxis: This step consists of a tumble followed by a tumble or a tumble followed by a run. Let *j* be the index of the chemotactic step, *k* be the index of the reproduction step, and *l* be the index of elimination–dispersal event. Suppose that the position of each member in the population of bacteria N_b at the *j*th chemotactic step, *k*th reproduction step, and *l*th elimination–dispersal event be given by $P(j,k,l) = \{\theta^i(j,k,l) \quad ; i = 1,2,....N_b\}$. Instead of $J(\theta)$, we consider $J(i, \theta^i(j,k,l))$ or simply $J(i,j,k,l)$ as the cost at the location of the *i*th bacterium $\theta^i(j,k,l) \; \varepsilon \; R_p$ (a space of real values of *p*) and N_c to be the length of the lifetime of the bacteria as measured by the number of chemotactic steps. To represent a tumble, a unit length in the random direction, e.g., $\varphi(j)$, is generated.

In particular, we take:

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + D(i)\varphi(j)$$

so that $D(i) > 0$, $i = 1, 2, \ldots, N_b$, is the size of the step made in the random direction specified by the tumble.

If at $\theta^{i}(j + 1, k+1, l)$,the cost $J(i, j + 1, k, l)$ is better (lower) than that at $\theta^{i}(j, k,l)$, then another step of size $D(i)$ will be made in the same direction. This movement, called swim, can be made in either direction in contrast to a tumble. It is continued until the cost is reduced but limiting the maximum number of steps $N_{s.}$

2) Swarming: As part of this task, the bacteria following the optimum path of food try to attract other bacteria so that together they more rapidly reach the desired location. The effect of swarming is to introduce an additional cost in $J(i, j , k, l)$.

3) Reproduction: For reproduction, the population is sorted in the ascending order of accumulated food so that out of the total number of bacteria N_b, the least healthy bacteria N_{br} die and the healthiest bacteria N_{br} reproduce (split into two) with no mutations. Note that $N_{br} = N_b / 2$, and let N_{re} be the number of reproduction steps.

4) Elimination and dispersal: This step helps reduce the behavior of stagnation (i.e., being trapped in a premature solution point or local optima). Each bacterium in the population undergoes this event with probability p_{ed}, and let N_{ed} be the number of these events.

3.2 Genetic Algorithm

Genetic Algorithms [26] are a family of computational models inspired by evolution. It is a global optimization method that manipulates a string of numbers in a manner similar to how chromosomes are changed in biological evolution. It solves a minimization problem by converting it into a maximization problem and finds global maxima. An initial population made up of strings of numbers is chosen at random or is specified by the user. Each string of numbers is called a "chromosome" or an "individual," and each number slot is called a "gene." A set of chromosomes forms a population. Each chromosome represents a given number of traits which are the actual parameters that are being varied to optimize the "fitness function". The fitness function is a performance index that we seek to maximize.

The operation of the GA proceeds in steps. Beginning with the initial population, "selection" is used to choose which chromosomes should survive to form a "mating pool." Chromosomes are chosen based on how fit they are (as computed by the fitness function) relative to the other members of the population. More fit individuals end up with more copies of themselves in the mating pool so that they will more significantly effect the formation of the next generation. Next, several operations are taken on the mating pool.

"Crossover" (which represents mating, the exchange of genetic material) occurs between parents. To perform crossover, a random spot is picked in the chromosome, and the genes after this spot are switched with the corresponding genes of the other parent. Following this, "mutation" occurs. This is where some genes are randomly changed to other values. After the crossover and mutation operations occur, the resulting strings form the next generation and the process is repeated. A termination criterion is used to specify when the GA should end (e.g., the maximum number of generations or until the fitness stops increasing).

3.3 Particle Swarm Optimization

The technique [27] generates a set of relevant solutions called population and then finds an optimal solution through searching and updating the past history of the particles of the population. It is influenced by cognitive and social behavior of the swarms. Each particle has some velocity according to which it moves in the multi-dimensional space. Each particle also has memory to keep information of its previously visited space.

The PSO algorithm is guided by two factors:

a) Movement of the particle in local neighborhood.

b) Movement of the particle in global neighborhood.

Local Best Solutions: - are the best solutions due to particle itself searching for the best solution in the restricted swarm.

Global Best Solution: - is the best solution due to all the particles participating in the solution space.

Local and global best positions are updated only if better solution is found for each iteration.

Notations

i) Position of the ith particle:-

$X_i = (X_{i1}, X_{i2}, \ldots, X_{iN})$ is the ith particle of the swarm. Here, the first subscript denotes the particle number and the second subscript denotes the dimension.

ii) Velocity of the ith particle:-

$V_i = (V_{i1}, V_{i2}, \ldots, V_{iN})$

iii) Local best position X_{best} of the swarm:-

$X_{best} = (P_{i1}, P_{i2}, \ldots, P_{iN})$

iv) Velocity Update:-

$$V_{id}^{(k+1)} = (\omega * V_{id}^{k} + \gamma_1 * \alpha(X_{bestid} - X_{id}^{k}) + \gamma_2 * \beta(g_{best} - X_{id}^{k}))$$

where: i= (1,2,…,m) is the number of swarms.

d= (1,2,…,N) is the dimension of the objective function to be optimized

g_{best} is the global best solution of the swarm.

k is the iteration number.

ω is the inertia weight to control the previous velocity vector of the swarm on the new one. It is a tradeoff between global and local exploration and helps in reducing the number of iterations for searching an optimal solution.

γ_1 and γ_2 are the random numbers between 0 and 1.

α is called the cognitive parameter.

β is called the social parameter.

Generally, $\alpha+\beta<=4$ and by default, $\alpha=\beta=2$.

v) Position Update:-

$$X_{id}^{(k+1)} = X_{id}^{k} + (V_{id}^{(k+1)} / q)$$

where: q is the correction factor (optional) to speed up the convergence process.

3.4 Gravitational Search Algorithm

It is an evolutionary algorithm based on the physical law of gravity and the law of motion [36]. The law of gravity states that every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them. In this method, agents are objects whose performance is dependent on their masses. All the objects attract each other by the gravity force. Therefore, all objects are pulled towards the objects with heavier masses, just as it happens in the nature.

Suppose there is a system with *N* agents, the position of every agent is a point in the search space which represents a solution to the problem. The position of the *i*th agent is defined as follows:

$X_i = (x_i^1, \ldots\ldots, x_i^d, \ldots\ldots, x_i^n)$ i=1,2,….,N

where *n* is the dimension of the problem, and x_i^d is the position of the *i*th agent in the *d*th dimension.

At starting point of the solution the agents are situated randomly. At the specific time'*t*' a gravitational force from mass '*j*' acts on mass '*i*', and is defined as follows:

$$F_{ij}^d(t) = G(t)\frac{M_i(t) \times M_j(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t))$$

where, m_i is the mass of the object *i*,

m_j is the mass of the object *j*,

G(t) is the gravitational constant at time *t*,

Rij(t) is the Euclidian distance between the two objects *i* and *j*, and

ε is a small constant.

The gravitational constant, *G*, which is initialized randomly, decreases by time to control the search accuracy.

In other words, *G* is a function of the initial value (G_0) and time (*t*):

$G(t)=G(G_O, t)$

The total force acting on agent *i* in the dimension *d* is calculated as follows:

$$F_i^d(t) = \sum_{j \in kbest, j \neq i} rand_j F_{ij}^d(t)$$

where $rand_j$ is a random number in the interval [0,1] and is used to give a randomized characteristics to the search.

According to the law of motion, the acceleration of the agent *i* at time *t*, in the *d*th dimension is directly proportional to the force acting on that agent, and inversely proportional to the mass of the agent:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}$$

Furthermore, the next velocity of an agent is a function of its current velocity added to its current acceleration. Therefore, the next position and the next velocity of an agent can be calculated as follows:

$$v_i^d(t+1) = rand_i . v_i^d(t) + a_i^d(t)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$$

The masses of the agents are calculated using fitness evaluation. The heavier the mass of an agent, the more efficient is that agent, regarding the solution it represents. It is notable that as the law of gravity and the law of motion imply, a heavy mass has a higher attraction power and moves more slowly.

The masses are updated as follows:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}$$

where $fit_i(t)$ represents the fitness value of the agent *i* at time *t*, and the *best(t)* and *worst(t)* in the population respectively indicate the strongest and the weakest agent according to their fitness route. For a minimization problem, *best(t)* and *worst(t)* can be defined as follows:

$$best(t) = \min_{j \in \{1,....,N\}} fit_j(t)$$

$$worst(t) = \max_{j \in \{1,....,N\}} fit_j(t)$$

The updated masses must be normalized using the following equation:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}$$

At the beginning of the system establishment, every agent is located at a certain point of the search space which represents a solution to the problem at every unit of time. The agents are evaluated and their next positions are calculated. The search can be stopped after a certain amount of time.

GSA can be considered as a population-based heuristic algorithm, in which the two common aspects are exploration and exploitation. The exploration is the ability to navigate through the whole search space and the exploitation is the ability to find the optima around a good solution. GSA, like many other population-based search algorithms provide satisfactory results. However, the results can be considered much more efficient in terms of speed. The exploration step can be guaranteed by choosing proper values for the random parameters in the equations mentioned earlier, and exploitation step is taken care of when the objects with heavier masses start to move more and more slowly.

Chapter 4
Proposed Approach

The proposed approach comprises of two methods:

4.1 SUSAN principle based Method**:** which is a derivative free method developed to work for color images.

4.2 Fuzzy Derivative based Method**:** which is developed to work for grayscale images.

4.1 SUSAN principle based Method

4.1.1Computation of USAN area

S (Smallest) USAN or SUSAN principle [38] has been invoked for the calculation of USAN area using a circular mask of 37 pixels with a radius of 3.4 pixels (at a few orientations 3,5,7,7,7,5,3 of pixels) shown in Figure 4.1.

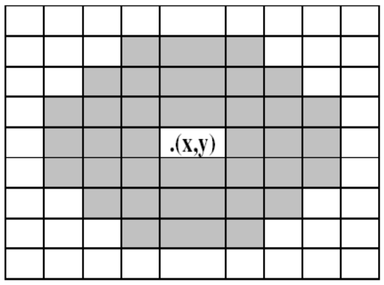


Fig. 4.1: Circular Mask

This mask encompassing 36 adjacent pixels to the central pixel, called the nucleus, is placed on every image pixel. The intensities of 37 pixels enclosed in the circular mask are compared with that of the nucleus by using a comparison equation and the outputs of 37 comparisons are algebraically summed up to yield a single value known as the “USAN”. The concept of each image pixel having associated with it a local (USAN) area of similar brightness is the basis for the SUSAN principle.

When an adjacent pixel to the center pixel in the mask is compared with the nucleus intensity then need arises for the choice of an important parameter, '*t*' (brightness of a threshold) which should be selected in order to yield a proper output. A value of 20 is found to be appropriate for *t* though this may vary according to the image type (i.e. natural images or man-made images).

The USAN area conveys the most important information about the structural element such as an edge or a corner of an image around a pixel of interest. This attains a maximum value when the nucleus lies in a flat region of the image surface; it falls to half of this maximum at a very near straight edge. It is this property of the USAN area which determines the presence of edges. As SUSAN edge detector has no image derivatives its performance in the presence of noise must be good. The integrating effect of the SUSAN principle together with its non-linear response bestows it the ability for the noise rejection. Note that the integration of individual areas reduces the effect of noise.

A comparison between the nucleus and its neighborhood pixel within the mask is governed by the equation [38]:

$$C(r, r_0) = \exp\left(-\left(\frac{I(r) - I(r_0)}{t} \right)^6 \right) \tag{1}$$

where r is the position of any pixel within the mask, r_o is the position of nucleus, I(r) is the Intensity of any pixel within the mask, $I(r_o)$ is the intensity of nucleus, and *t* is the brightness of different threshold.

Eq. (1) is applied on each pixel within the mask and sum of the outputs arising from the comparison is:

$$k(r_0) = \sum_r C(r, r_0) \tag{2}$$

This total, *k*, is just the number of pixels in the mask, i.e. USAN area.

4.1.2 Fuzzification

Fuzzy property is accrued from the membership function value. Here, we use a modified Gaussian membership function that contains only one fuzzifier f_h. As discussed above USAN area is portrayed as a histogram with USAN area is along the x-axis and frequency of USAN area as the y-axis. The frequency of occurrence of USAN area in an USAN area image is given by:

$$p(k) = \frac{h(k)}{\sum h} \quad ; \quad k=1,2...37 \tag{3}$$

where $p(k)$ is the frequency of occurrence of area k and $h(k)$ is the total number of pixels having area k.

The $p(k)$ satisfies the conditions

$$\sum_{0}^{L-1} p(k) = 1 \tag{4}$$

The histogram-based Gaussian membership function converts the spatial domain image in the form of pixel intensities into the fuzzy domain in terms of membership values as:

$$\mu_1(k) = \exp\left(-\left(\frac{(x_{\max} - k)^2}{2 * f_h^{\ 2}}\right)\right) \tag{5}$$

The fuzzifier parameter: f_h used in (5) is determined as:

$$f_h^{\ 2} = \frac{\sum_{k=0}^{L-1} (x_{\max} - k)^4 p(k)}{2\sum_{k=0}^{L-1} (x_{\max} - k)^2 p(k)} \tag{6}$$

where k is the USAN area, $x_{\max}$ is the maximum USAN area. Here L is 37 because the number of pixels in a mask is 37.

A modified bell-shaped membership function acting as hedge for enhancing the membership function (MF) values of the original gray levels of the image is given by:

$$\mu_2(k) = \max\left(\frac{4}{1+\left|\frac{\mu_1(k)-a}{c}\right|^{2b}} - 1, 0 \right) \tag{7}$$

where a , b and c are the three parameters that can be varied to control the shape and range of the membership function thus giving more flexibility to the user. It is just the upper half part of the original bell- shaped MF having a limited base width (support). This membership function helps to detect the weak edges in the image.

4.1.3 Fuzzy Measures and Optimization

4.1.3.1 Fuzzy Measures

A set of fuzzy edge sharpness measures is defined for the computation of the objective function and to further sharpen the edges. To make the proposed edge detection method converge faster, appropriate sharpness measures are needed. Here fuzzy sharpness measures have to be framed to approximate the actual edge quality. The quality of the edges can only be ascertained by a quantitative analysis of these edges and the entropy and fuzzy edge sharpness factors can serve as the quantitative measure of image edge quality.

The fuzzy edge sharpness factors of an image can be thought of the deviation of the membership values from the crossover point. Two fuzzy edge sharpness factors are separately defined for both the weak and strong edges.

The fuzzy edge sharpness factor for the weak edges of the image is given by:

$$F_W = \frac{1}{L}\sum_{k=0}^{L-1}[\mu_2(k) - c]^2\, p(k) \tag{8}$$

The average fuzzy edge sharpness factor for the weak edges of the image is given by:

$$F_{avgW} = \frac{1}{L}\sum_{k=0}^{L-1}[\mu_2(k) - c]p(k) \tag{9}$$

The fuzzy edge sharpness factor for the strong edges of the image is given by:

$$F_S = \frac{1}{L}\sum_{k=0}^{L-1}[\mu_1(k) - c]^2\, p(k) \tag{10}$$

The average fuzzy edge sharpness factor for the strong edges of the image is given by:

$$F_{avgS} = \frac{1}{L}\sum_{k=0}^{L-1}[\mu_1(k) - c]p(k) \tag{11}$$

In the above definitions, the average fuzzy edge sharpness factor gives the overall sharpness of the edges in an image whereas the fuzzy edge sharpness factor gives the deviation of the membership values from the crossover point *c* (say a reference point). The ratio of the absolute average fuzzy edge sharpness factor to the fuzzy edge sharpness factor is defined as the fuzzy edge quality factor.

The fuzzy edge quality factor for the weak edges of the image is given by:

$$Q_W = \frac{F_{avgW}}{F_W} \tag{12}$$

The fuzzy edge quality factor for the strong edges of the image is given by:

$$Q_S = \frac{F_{avgS}}{F_S} \tag{13}$$

The above definitions are designed to capture the extent of the uncertainty present in the edges.

4.1.4 Definition of Fuzzy Entropy

According to Shannon's theorem, Entropy is a measure of average information per pixel present in an image. In the fuzzy domain it is the measure of the uncertainty or randomness associated with image information and is defined by:

$$E = \frac{-1}{L\ln 2}\left[\sum_{k=0}^{L-1}\left(\mu_2(k)\ln(\mu_2(k)) + ((1-\mu_2(k))\ln(1-\mu_2(k)))\right)\right] \tag{14}$$

Since it provides useful information about the extent to which the information can be retrieved from the image, optimization of this should pave the way for the determination of the parameters: *a, b*, *c* and f_h. It may be noted that lesser amount of information is needed to distinguish between the weak edge pixel and a noisy pixel than that needed for detecting a strong edge pixel, necessitating the optimization of the fuzzy entropy.

4.1.5 Sharpness Factor

For the purpose of evaluating the fuzzy edge sharpness, the normalized fuzzy edge sharpness factor, called the sharpness factor, is on the anvil. It is intended to give an idea on the amount of sharpness concerning the edges. The sharpness factor is defined as:

$$S_f = \frac{Q_S}{Q_W} \tag{15}$$

The definition of the sharpness factor allows us to specify a range for contemplating on the normalized fuzzy edge sharpness factor such that increasing its value beyond a certain range causes the loss of the pleasing nature of the image. By experimentation, the sharpness factor is found to lie in between 1.0 and 1.5 for a good quality edge map.

4.1.6 Optimization of an Objective Function

Both entropy function and the edge sharpness factor S_f tell the edge quality; hence these must be optimized together. Their optimization is posed as the constrained optimization as in [43].

Optimize the entropy function E subject to the constraint $S_{df} = S_f$.

For this, an objective function is set up as under:

$$J = E + \lambda \exp\left(\left|S_{df} - S_f\right|\right) \tag{16}$$

where λ is chosen as 0.5 experimentally. The optimization of E and J using the evolutionary methods is subject to the constraints: $-2 < a < 0$, $3 < b < 6$ and $1 < c < 3$.

4.1.7 Initialization of parameters of the Evolutionary Algorithms

To learn the optimal values of the parameters (a, b, c, f_h) four different evolutionary algorithms namely, BF, GA, PSO and GSA have been applied for the optimization purpose. The parameters of the evolutionary algorithms: BF, GA, PSO, and GSA turn the objective function J a time-varying function during the optimization.

4.1.7.1 BF Parameters

The BF algorithm [25] is modified to reduce its complexity by ignoring the cell-to-cell attractant function in the swarming stage. The initial values of the parameters of modified BF algorithm are set as follows:

1) The number of bacteria S =10

2) The swimming length N_s =12

3) The number of iterations in a chemotactic loop N_c =15

4) The number of reproduction steps N_{re} =4

5) The number of elimination and dispersal events N_{ed} = 6

6) The probability of elimination/dispersal p_{ed} =0.26

4.1.7.2 GA Parameters

The initial values of the parameters of GA are set as follows:

1) Number of individuals in the population *pop_size*=20

2) Number of traits in each individual *num_traits* =2

3) Maximum number of generations *max_generation*=1000

4) Probability of crossover *cross_prob* =0.8

5) Probability of mutation *mutat_prob* =0.05

4.1.7.3 PSO Parameters

The initial values of the parameters of PSO are set as follows:

1) The number of swarms *swarm_size* = 27

2) Maximum number of iterations *itr*=80

3) Inertia weight ω = 1.0

4) Cognitive parameter α=2

5) Social parameter β=2

6) Parameters γ_1 and γ_2 are generated randomly in the interval [0-1]

7) Correction factor *q*=1.3

4.1.7.4 GSA Parameters

The initial values of the parameters of GSA are set as follows:

1) The number of agents *N*=50

2) Maximum number of iterations *max_it*=30

Comparison of different evolutionary algorithms in terms of their parameters indicates that crossover and mutation operators used in GA are not involved in BF, PSO and GSA.

The biological thing is called as bacterium in BF, chromosome in GA, swarm in PSO and agent in GSA. Termination criterion for the various evolutionary algorithms is: maximum number of iterations =($S*N_c* N_{re} * N_{ed}$) for BF, maximum number of generations for GA, maximum number of iterations for PSO and GSA. From experimentation, it is noticed that a considerable saving in computation time is possible by reducing the number of iterations but not at the cost of sacrificing the edge quality. The computation time also varies according to the image size.

Proper selection of the initial parameters of the evolutionary algorithm is crucial in deciding the accurate optimum values of the parameters in lesser time.

4.1.8 Adaptive Thresholding

In adaptive thresholding, a threshold is varied over different regions based on spatial variations in illumination in an image. Binary thresholding is facilitated through adaptive thresholding by analyzing each pixel with respect to its local neighborhood adaptively. One statistical method [40] to find the adaptive or local threshold is to examine the intensity values of the local neighborhood of each pixel by computing the *mean* or the *median* value; or the *median* / the *mean* of the minimum and maximum values or *(max + min)/2* value of the *local* intensity distribution. During the thresholding process, individual pixels in an image are marked as “object” pixels if their value is greater than the threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. The size of the local neighborhood window has to be large enough to cover sufficient foreground and background pixels; otherwise a poor threshold is selected.

4.1.9 An Optimal Fuzzy SUSAN principle based Algorithm for Edge Detection

The algorithm consists of the following steps:

Step 1: Input the given RGB image and operate on its three components: Red, Green and Blue components.

Step 2: Repeat Steps 3 to 10 on each component.

Step 3: Place a circular mask at each pixel of an image called the nucleus.

Step 4: Determine the number of pixels within the circular mask having similar brightness as that of the nucleus using Eq. (1). The count of these pixels defines the USAN area.

Step 5: Initialize the parameters (*a, b, c*) and calculate the fuzzifier f_h using Eq. (6).

Step6: Repeat Steps 7 to 11 to learn optimized values of the parameter set (a , b , c, f_h).

Step 7: Compute the Histogram based Gaussian membership function value $\mu_1(k)$ using Eq. (5).

Step 8: Compute modified bell- shaped membership function value $\mu_2(k)$ using Eq. (7).

Step 9: Calculate F_W , F_{avgW} , F_S , F_{avgS} , Q_W , Q_S from the initial values for the parameters a , b, c , f_h using Eq.s (8) to (13).

Step 10: Calculate the Entropy E and sharpness factor S_f using Eq.s (14) and (15) respectively and set the desired sharpness factor $S_{df} \leftarrow$ 1.5 to iteratively learn the parameters.

Step 11: Optimize the objective function *J* using Eq. (16) and applying BF, GA, PSO, GSA evolutionary algorithms separately.

Step 12: Obtain the fuzzified R, G and B component images using $\mu_2(k)$ computed using optimized values of the parameter set (a , b , c, f_h).

Step 13: Apply adaptive thresholding to each of the three USAN area fuzzy images to obtain the edge map for R, G and B components.

Step 14: Combine the edge maps of R, G and B components to obtain the final edge map.

Step 15: Reduce the isolated pixels in the final edge map by computing the sum of the immediate 4- neighborhood of pixel of interest *I(x, y)* and marking *I(x, y)* as an edge pixel only if the sum is less than 1.

Step 16: Derive the color edge map by taking the address of the edge pixels in the edge map and then use this address on the original RGB image.

The block diagram of the proposed SUSAN principle based method and a scheme for the optimal fuzzy system are shown in Figure 4.2 and Figure 4.3 respectively.

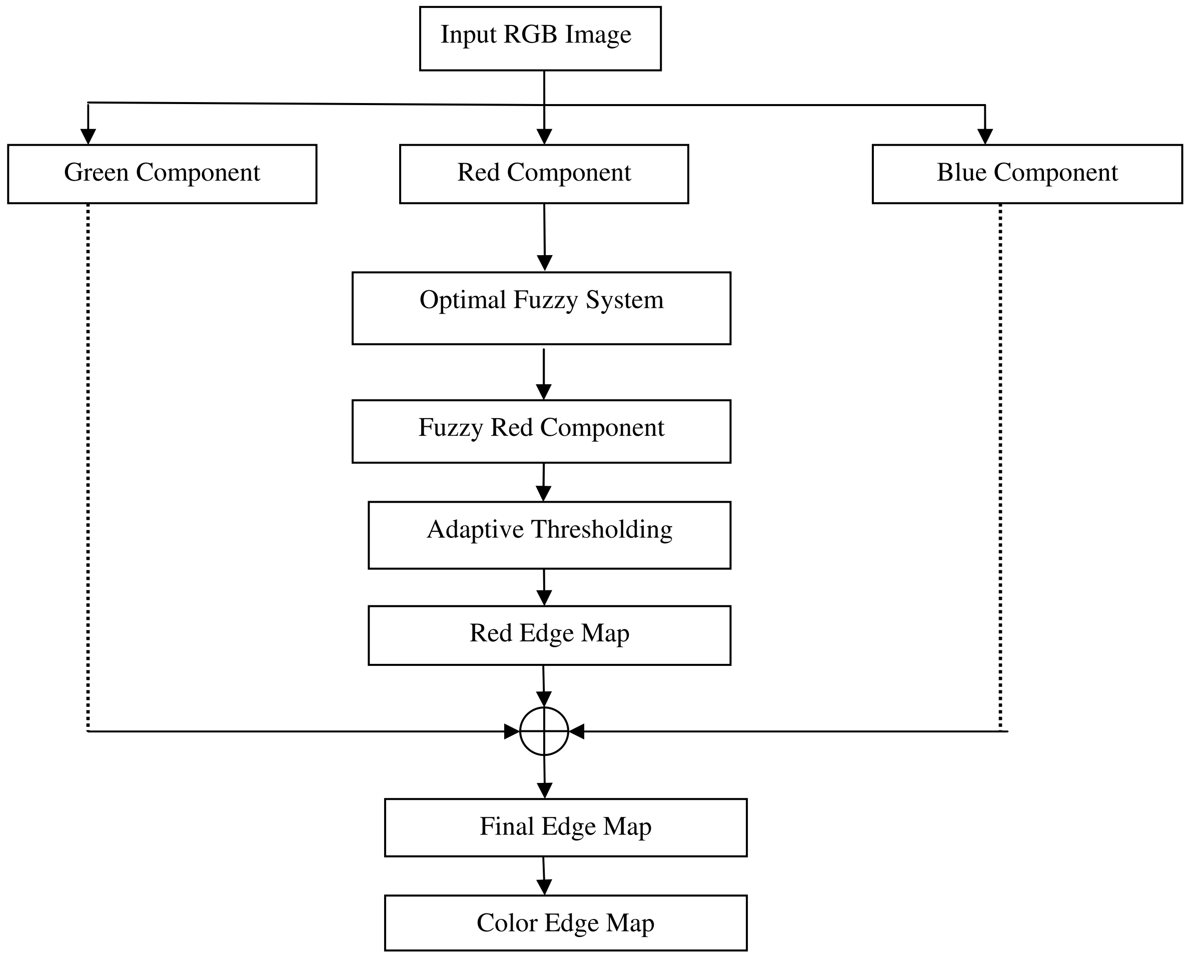


Fig.4.2: Block Diagram of proposed SUSAN principle based method

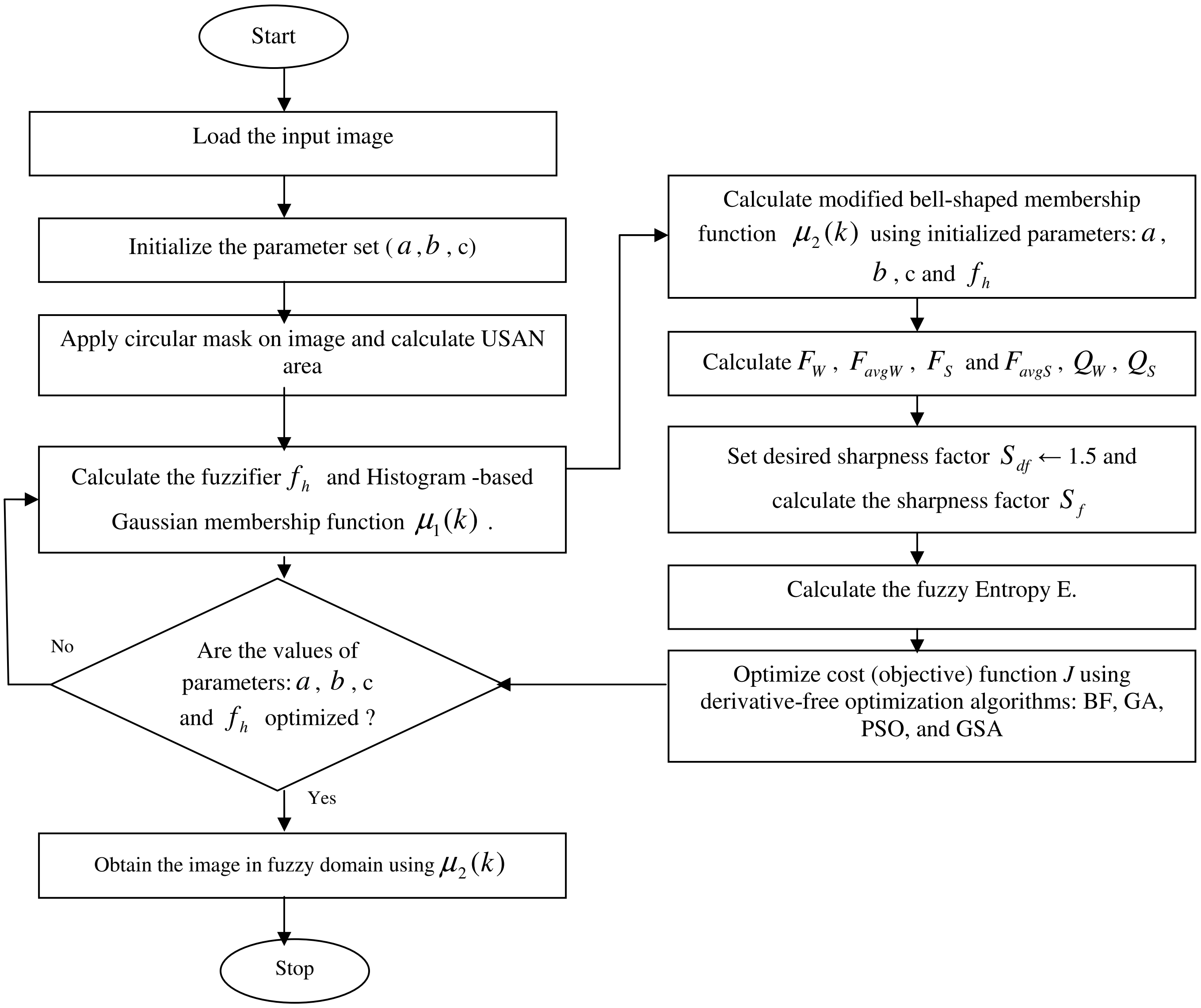

Start
Load the input image
Initialize the parameter set (a, b, c)
Apply circular mask on image and calculate USAN area
Calculate the fuzzifier f_h and Histogram -based Gaussian membership function $\mu_1(k)$.
Are the values of parameters: a, b, c and f_h optimized ?
No
Yes
Obtain the image in fuzzy domain using $\mu_2(k)$
Stop
Calculate modified bell-shaped membership function $\mu_2(k)$ using initialized parameters: a, b, c and f_h
Calculate F_W, F_{avgW}, F_S and F_{avgS}, Q_W, Q_S
Set desired sharpness factor $S_{df} \leftarrow 1.5$ and calculate the sharpness factor S_f
Calculate the fuzzy Entropy E.
Optimize cost (objective) function J using derivative-free optimization algorithms: BF, GA, PSO, and GSA

Fig.4.3: Scheme for Optimal Fuzzy System

4.2 Fuzzy Derivative based Method

4.2.1 Computation of Derivative value for a pixel

Consider the neighborhood of a pixel (x, y) as shown in Figure 4.4. A derivative at the central pixel position (x, y) in the direction D (D = {NW, W, SW, S, SE, E, NE, N}) is defined as the difference between the pixel of interest and its neighbor in the corresponding direction.

NW	N	NE
W	(x,y)	E
SW	S	SE

Fig.4.4: Pixel (x,y) with its neighborhood pixel

For example the derivative at the central pixel position (x,y) in the direction N is defined as the difference between intensity of the pixel at (x,y) and its neighbor in the direction N as:

$$\nabla_N(x, y) = I(x-1, y) - I(x, y) \qquad (1)$$

Next, the principle of derivative is based on the following observation. Consider an edge passing through the neighborhood of a pixel (x, y) in the NE-SW direction as shown in Figure 4.5. The derivative value in the NW and SE directions will be large, but also the derivative values of neighboring pixels perpendicular to edge direction can expected to be large. So if there is an edge passing through the pixel (x,y) in NE-SW direction, then derivative values in the directions NW and SE for the pixels at positions (x,y),(x+1,y-1) and (x-1,y+1) should be high. A direct estimate of intensity variation due to the pixel is calculated which is used to calculate the derivative as maximum of two highest valued absolute derivatives out of three found for a particular direction. If two out of three derivative values are small, then it is safe to assume that no edge is present in the considered direction. This observation is taken into account in formulating the *if-then* rules to calculate the net derivative values.

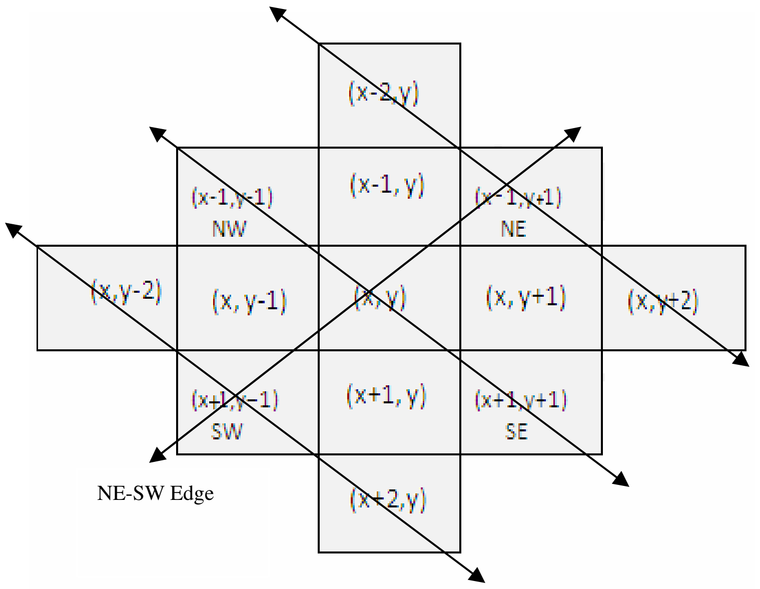


Fig.4.5: The pixels to be considered for the edge in NE-SW direction

For edge in the NE-SW direction, the three absolute derivatives in the NW direction can be defined as:

$$\nabla_{NW}(x, y) = |I(x-1, y-1) - I(x, y)| \qquad (2)$$

$$\nabla_{NW}(x-1, y+1) = |I(x-2, y) - I(x-1, y+1)| \qquad (3)$$

$$\nabla_{NW}(x+1, y-1) = |I(x, y-2) - I(x+1, y-1)| \qquad (4)$$

The net derivative value $\nabla^{net}{}_{NW}(x, y)$ for the pixel at position (x,y) in the NW direction can be calculated by applying the *if-then* rules[41] that are modified as :

Rule1: If $((\nabla_{NW}(x, y) \le \nabla_{NW}(x-1, y+1))$ and $(\nabla_{NW}(x, y) \le \nabla_{NW}(x+1, y-1)))$, then

$$\nabla^{net}{}_{NW}(x, y) = \max(\nabla_{NW}(x-1, y+1), \nabla_{NW}(x+1, y-1))$$

Rule2: If $((\nabla_{NW}(x-1,y+1) \le \nabla_{NW}(x,y))$ and $(\nabla_{NW}(x-1,y+1) \le \nabla_{NW}(x+1,y-1)))$,

then $\nabla^{net}{}_{NW}(x,y) = \max(\nabla_{NW}(x,y), \nabla_{NW}(x+1,y-1))$

Rule3: If $((\nabla_{NW}(x+1,y-1) \le \nabla_{NW}(x,y))$ and $(\nabla_{NW}(x+1,y-1) \le \nabla_{NW}(x-1,y+1)))$,

then $\nabla^{net}{}_{NW}(x,y) = \max(\nabla_{NW}(x,y), \nabla_{NW}(x-1,y+1))$

Similarly compute other $\nabla^{net}{}_{D}(x,y)$ in the direction D = {W, SW, S, SE, E, NE, N}. In this way, total of 24 *if-then* rules are formulated to compute $\nabla^{net}{}_{D}(x,y)$.

Let $\nabla^{'}$ denote the derivative set whose elements are $\nabla^{net}{}_{D}$ for D= {NW, W, SW, S, SE, E, NE, N} sorted in ascending order.

i.e. $\nabla^{'} = \{ \nabla^{net}{}_{NW}, \nabla^{net}{}_{W}, \nabla^{net}{}_{SW}, \nabla^{net}{}_{S}, \nabla^{net}{}_{SE}, \nabla^{net}{}_{E}, \nabla^{net}{}_{NE}, \nabla^{net}{}_{N} \}$

For the input image of size (NxN), the derivative matrix $\nabla(i)$ is calculated as:

$\nabla(i)$ =median ($\nabla^{'}$) ; i=1,2,….,N (5)

4.2.2 Fuzzification

In fuzzy image processing, there are a number of ways for doing edge detection. The simplest technique is to define a membership function indicating the degree of variation in pixel intensities in each neighborhood [42]. This can be achieved by fuzzifying the relevant pixel intensity values into a membership function values that indicates the degree of variation constituting an edge in a particular location. In the proposed approach the derivative matrix describing the median of the derivative values are fuzzified into MF values to indicate the degree of variation constituting an edge in a particular direction.

The derivative matrix $\nabla(i)$ is fuzzified using bell-shaped membership function $\mu_{bell}(\nabla(i))$ defined as:

$$\mu_{bell}(\nabla(i)) = \frac{1}{1+\left|\frac{\nabla(i)-\alpha}{\beta}\right|^{2\gamma}} \quad ; \quad i=1,2,\ldots,N \tag{6}$$

where α, β and γ are the three parameters that can be varied to control the shape and range of the membership function.

A membership function HIGH acting as hedge for enhancing the bell-shaped MF values is given by:

$$\mu_{HIGH}(i) = \begin{cases} 0 & , if \quad 0 < \mu_{bell}(\nabla(i)) \le a \\ \frac{(\mu_{bell}(\nabla(i)) - a)}{(b-a)} & , if \quad a < \mu_{bell}(\nabla(i)) < b \\ 1 & , if \quad \mu_{bell}(\nabla(i)) \ge b \end{cases} \quad ; i=1, 2,\ldots, N \tag{7}$$

where *a* and *b* are the parameters that can be varied to control the shape and range of the membership function thus giving more flexibility to the user. This membership function helps to detect the edges in the image.

4.2.3 Parameters and Objective Function Optimization

As the final edge output depends heavily on the values of the parameters: (α, β, γ, *a*, *b*), the optimization of these parameters is necessary.

The initial values of the parameters of bell-shaped membership function: (α, β, γ) and that of HIGH membership function: (*a*, *b*) are optimized using BF, GA, PSO, GSA evolutionary algorithms.

For optimization purpose, the objective function *J* is defined as:

$$J = E_d - E \tag{8}$$

where E_d is the desired fuzzy entropy and *E* is the actual fuzzy entropy given by the following equation:

$$E = \frac{-1}{N\ln 2}\left[\sum_{i=0}^{N-1}\left(\mu_{HIGH}(i)\ln(\mu_{HIGH}(i)) + ((1-\mu_{HIGH}(i))\ln(1-\mu_{HIGH}(i)))\right)\right] \quad (9)$$

The optimization of *J* and *E* is subject to the constraint: $0 < a < 1$ and $0 < b < 1$ because the input domain for HIGH membership function is the fuzzy domain i.e. restricted to the range (0, 1), therefore, the values of the parameters: *a* and *b* cannot exceed this range.

Since fuzzy entropy signifies the randomness present in the image information, it has to be optimized for extracting edge information from the image. Taking very less value of around 0.1 or very high value of around 0.9 for the desired fuzzy entropy does not have much effect on the edge information. Experimentally, the desired fuzzy entropy E_d has been found to lie in the range of 0.4 to 0.5. The objective function *J* is defined in such a manner such that the deviation of the obtained fuzzy entropy value from its desired value is optimized.

4.2.4 Initialization of parameters of the Evolutionary Algorithms

To learn the optimal values of the parameters (α, β, γ, *a, b*) four different evolutionary algorithms namely, BF, GA, PSO and GSA have been applied for the optimization purpose. The parameters of the evolutionary algorithms: BF, GA, PSO, and GSA turn the objective function *J* a time-varying function during the optimization.

4.2.4 .1 BF Parameters

The BF algorithm [25] is modified to reduce its complexity by ignoring the cell-to-cell attractant function in the swarming stage.

The initial values of the parameters of modified BF algorithm are set as follows:

1) The number of bacteria S =8

2) The swimming length N_s =2

3) The number of iterations in a chemotactic loop N_c =4

4) The number of reproduction steps N_{re} =4

5) The number of elimination and dispersal events N_{ed} = 3

6) The probability of elimination/dispersal p_{ed} =0.26

4.2.4.2 GA Parameters

The initial values of the parameters of GA are set as follows:

1) Number of individuals in the population *pop_size*=25

2) Number of traits in each individual *num_traits* =2

3) Maximum number of generations *max_generation*=1000

4) Probability of crossover *cross_prob* =0.6

5) Probability of mutation *mutat_prob* =0.02

4.2.4.3 PSO Parameters

The initial values of the parameters of PSO are set as follows:

1) The number of swarms *swarm_size* = 20

2) Maximum number of iterations *itr*=50

3) Inertia weight ω = 1.0

4) Cognitive parameter α=2

5) Social parameter β=2

6) Parameters γ_1 and γ_2 are generated randomly in the interval [0-1]

7) Correction factor q=1.3

4.2.4.4 GSA Parameters

The initial values of the parameters of GSA are set as follows:

1) The number of agents N=40

2) Maximum number of iterations *max_it*=20

4.2.5 An Optimal Fuzzy Derivative based Algorithm for Edge Detection

The algorithm consists of the following steps:

Step 1: Input the grayscale image of size (NxN).

Step 2: Calculate absolute derivative values for edge pixels in all 8 different directions.

Step 3: Compute net derivative value $\nabla^{net}{}_{D}(x,y)$ for pixel of interest (x,y) in all 8 different directions by applying 24 if-then rules.

Step 4: Sort 8 net derivative values in ascending order and store them in $\nabla^{'}$.

Step 5: Compute derivative matrix $\nabla(i)$ using Eq. (5).

Step 6: Initialize the parameter set (α, β, γ, *a, b*) and repeat Steps 7 to 10 to learn their optimal values.

Step 7: Fuzzify $\nabla(i)$ using bell-shaped membership function $\mu_{bell}(\nabla(i))$ in Eq. (6).

Step 8: Compute HIGH membership function $\mu_{HIGH}(i)$ using Eq. (7).

Step 9: Compute fuzzy entropy E using Eq. (9).

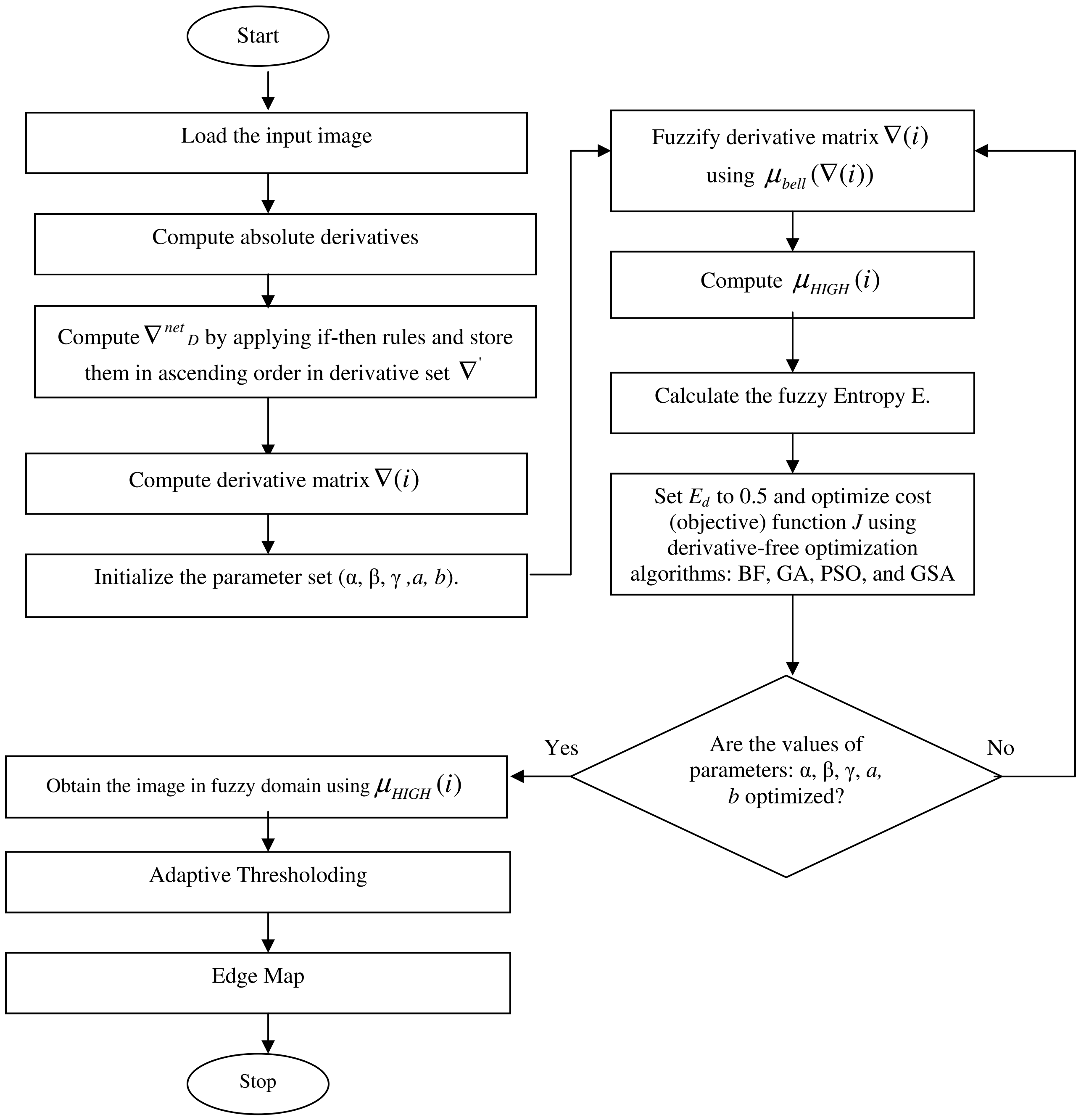
Step 10: Set E_d to 0.5 and optimize the objective function *J* using Eq. (8) and applying BF, GA, PSO, GSA evolutionary algorithms separately .

Step 11: Obtain the image in the fuzzy domain using $\mu_{HIGH}(i)$ computed using optimized values of the parameter set (α, β, γ *a, b*).

Step 12: Apply adaptive thresholding to the fuzzified image to obtain the edge map.

Step 13: Reduce the isolated pixels in the edge map by computing the sum of the immediate 4- neighborhood of pixel of interest *I(x, y)* and marking *I(x, y)* as an edge pixel only if the sum is less than 1.

The flowchart of the proposed fuzzy derivative based method is shown in Figure 4.6.



Start
Load the input image
Compute absolute derivatives
Compute $\nabla^{net}{}_{D}$ by applying if-then rules and store them in ascending order in derivative set ∇'
Compute derivative matrix $\nabla(i)$
Initialize the parameter set (α, β, γ ,*a, b*).
Fuzzify derivative matrix $\nabla(i)$ using $\mu_{bell}(\nabla(i))$
Compute $\mu_{HIGH}(i)$
Calculate the fuzzy Entropy E.
Set E_d to 0.5 and optimize cost (objective) function *J* using derivative-free optimization algorithms: BF, GA, PSO, and GSA
Are the values of parameters: α, β, γ, *a*, *b* optimized?
Yes
No
Obtain the image in fuzzy domain using $\mu_{HIGH}(i)$
Adaptive Thresholoding
Edge Map
Stop

Fig.4.6: Flowchart of proposed fuzzy derivative based method

Chapter 5
Experimental Results

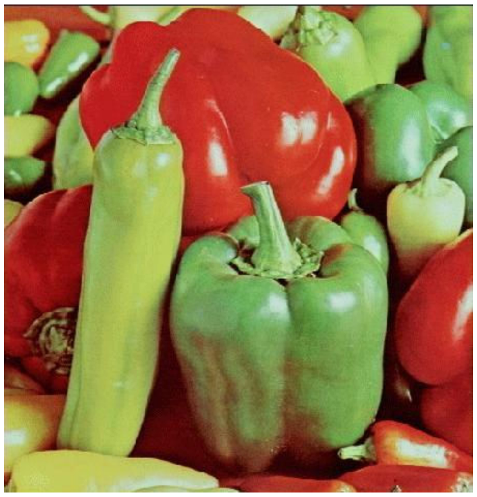
5.1 Results for SUSAN principle based Method

The proposed SUSAN principle based method for detecting the edge map in color images has been implemented in Matlab. No preprocessing is required prior to the application of this algorithm. The performance of the algorithm depends upon the values of the parameters: *t*, *mean*, *a, b*, *c* and f_h. The USAN area varies from 1 to 37 depending on the threshold'*t*'. In case of simple image a low value of '*t*' in the range of $10<t<20$ is preferable but for noisy images a higher value of '*t* 'in the range of $20<t<30$ is necessitated. Thus the choice of a threshold'*t*' is image dependent.

The local neighborhood size and the *mean* value of the adaptive thresholding also affect the performance of the algorithm. Experimentally, a (10x10) neighborhood size is found to give good results. The default *mean* value is kept at 0.1. Increasing its value leads to the removal of additional noise but at the cost of losing some edges. Decreasing the *mean* value replenishes the desired information. The original images of Lena, House and Pepper used as test images are shown in Figure 5.1. The results obtained for the test images using different variants of this method are shown in Figures 5.2 - 5.13.

(a)

(b)



(c)

Fig.5.1: Original (a) Lena Image (b) House Image (c) Pepper Image

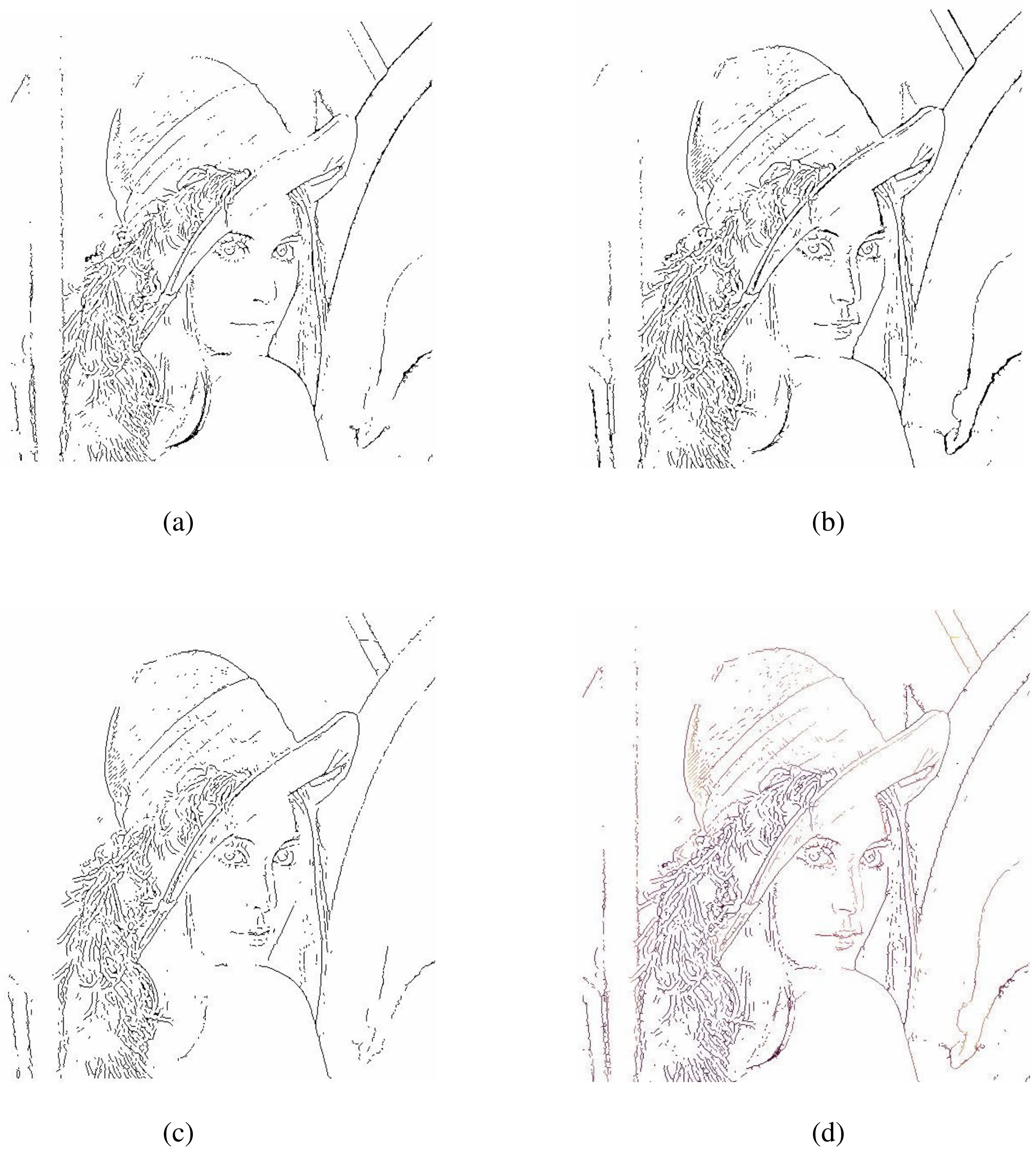
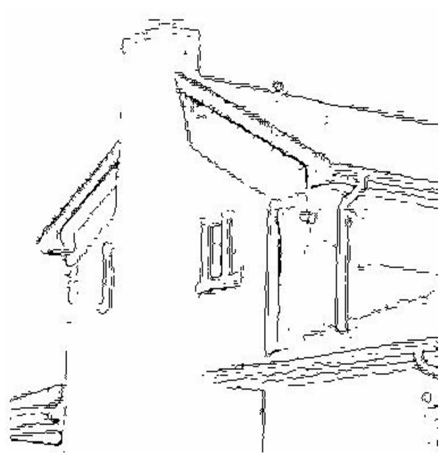
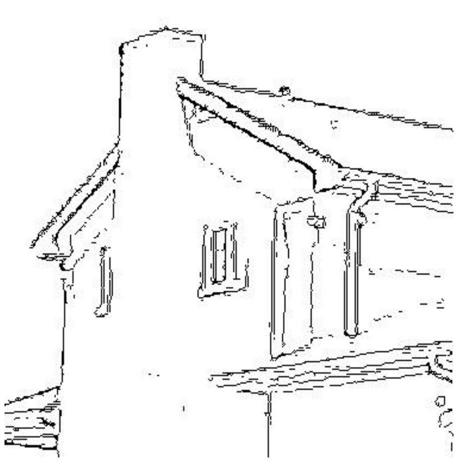


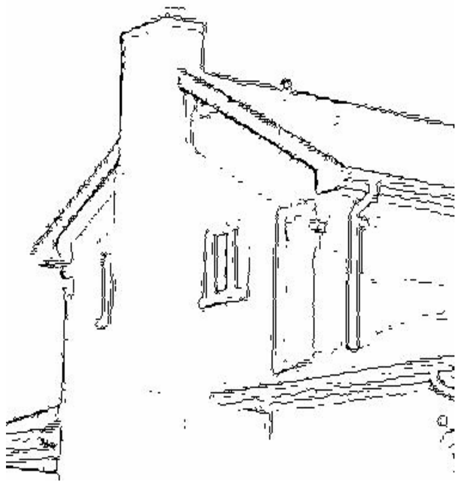
Fig.5.2: BF Lena edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



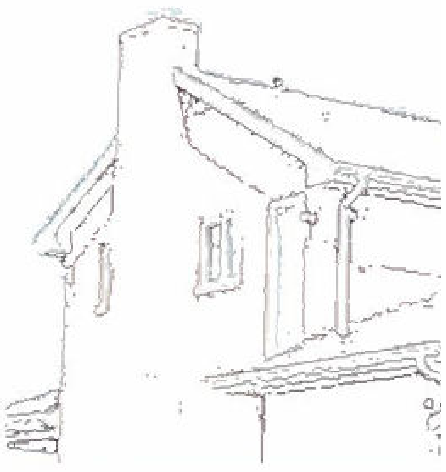
(a)



(b)

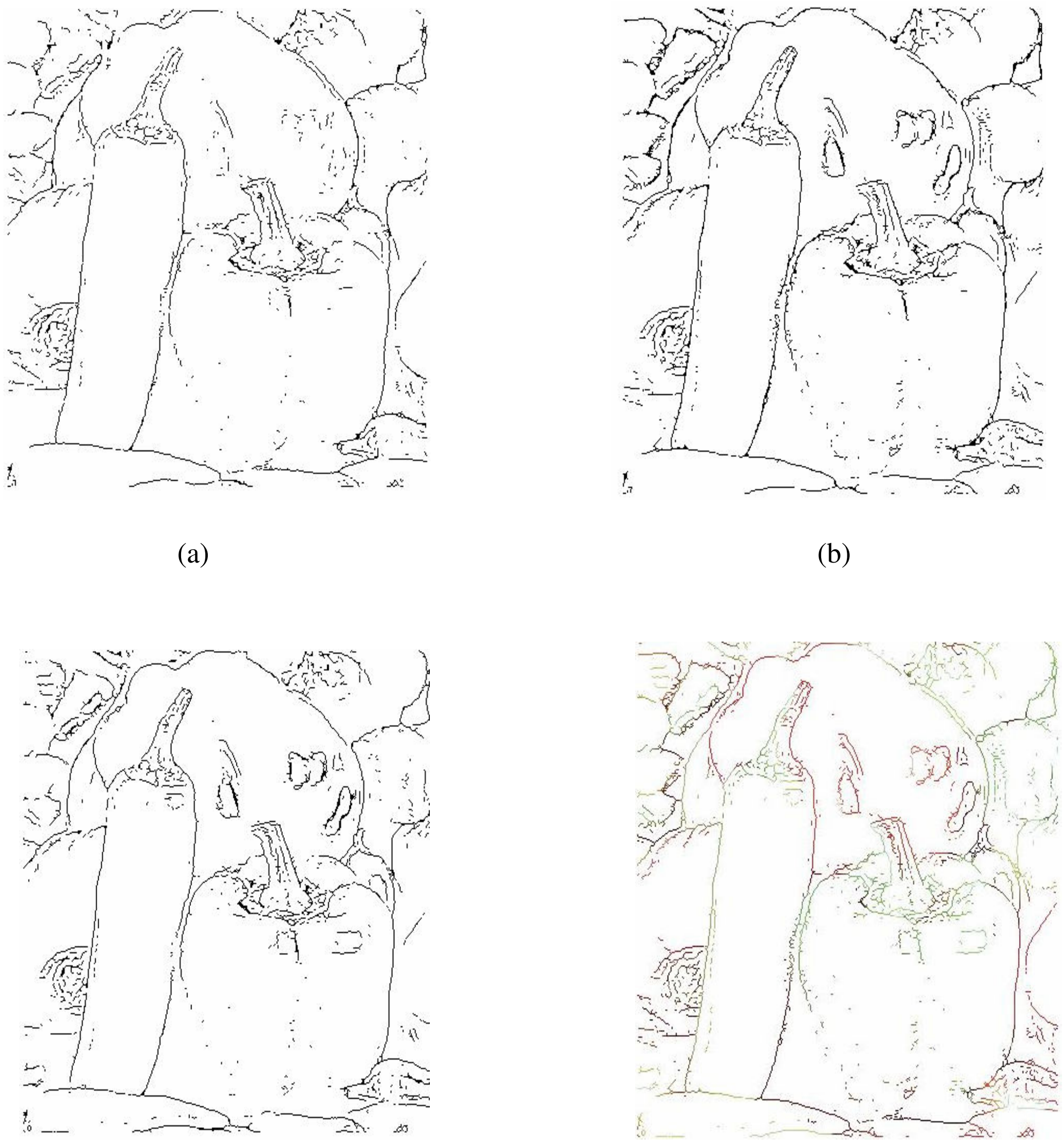


(c)



(d)

Fig.5.3: BF House edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.

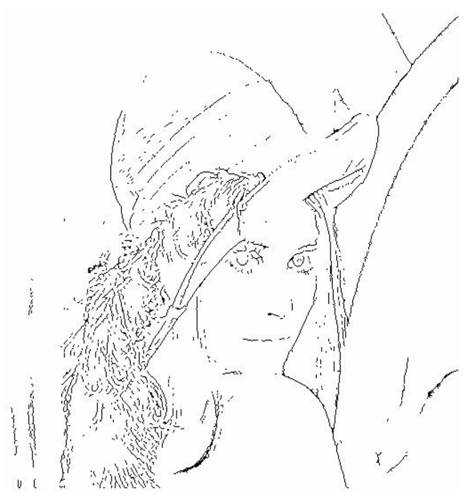

(a)

(b)

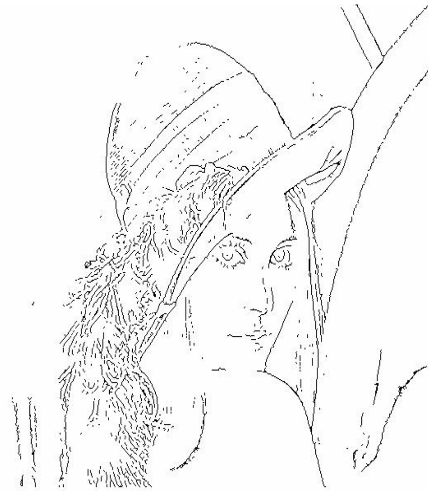
(c)

(d)

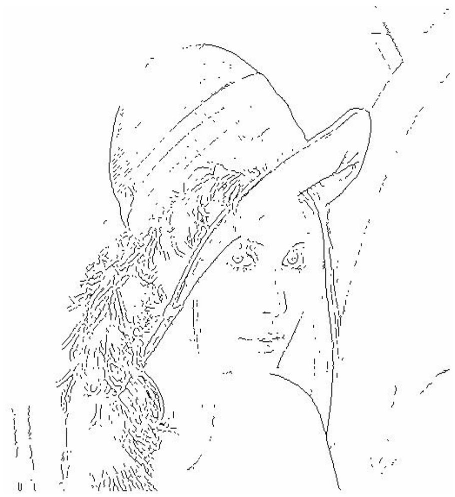
Fig.5.4: BF Pepper edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



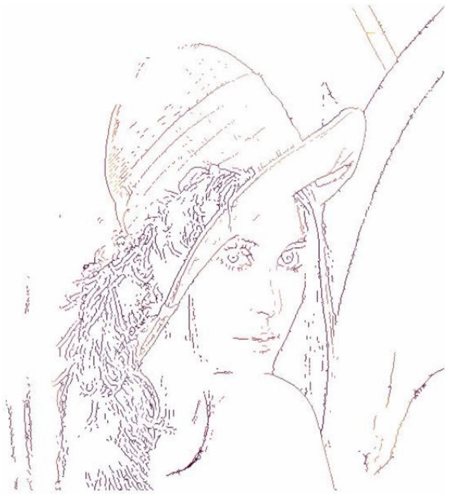
(a)



(b)

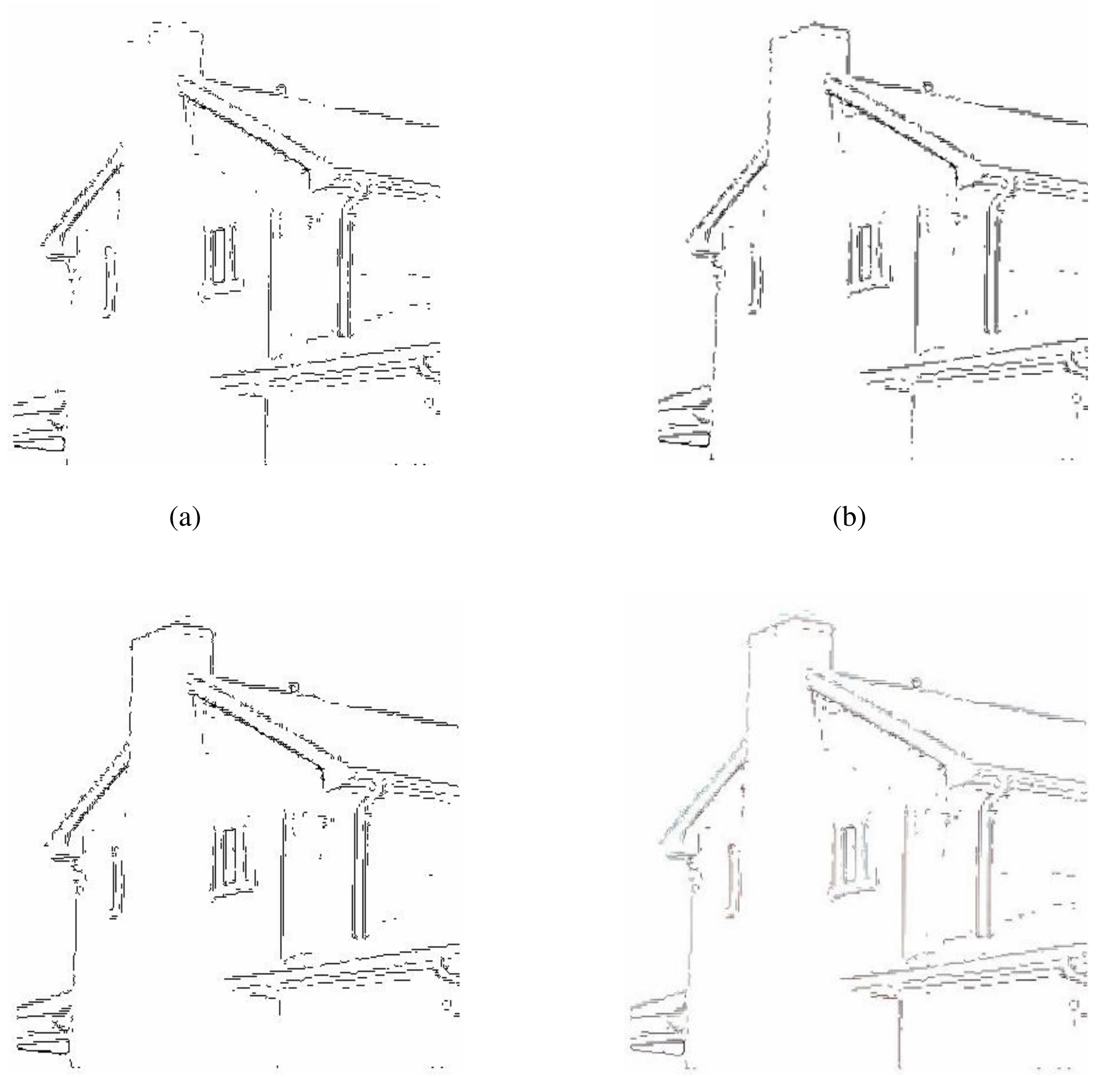


(c)



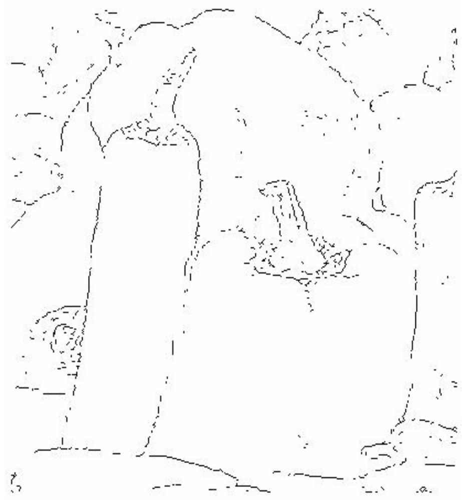
(d)

Fig.5.5: GA Lena edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.

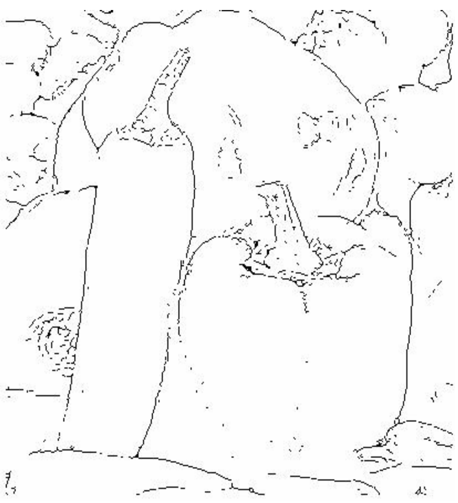

(a) (b)

(c) (d)

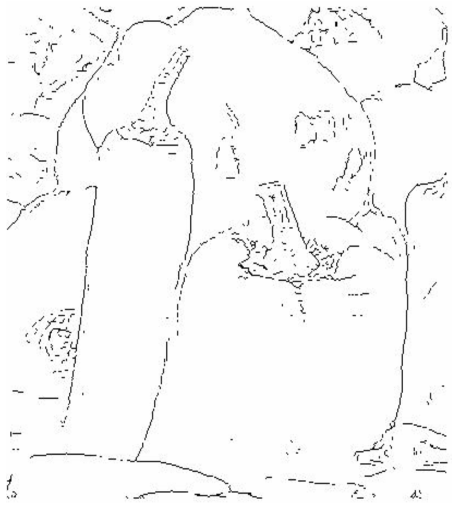
Fig.5.6: GA House edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



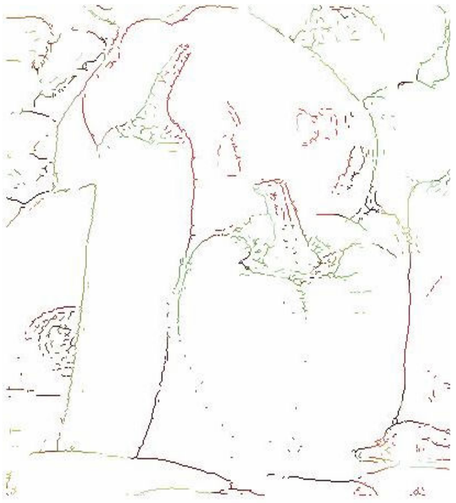
(a)



(b)

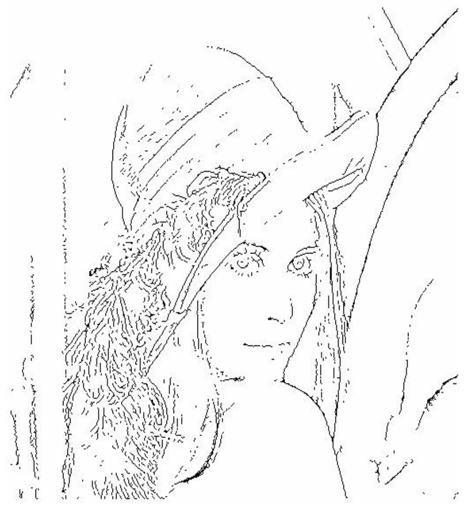


(c)

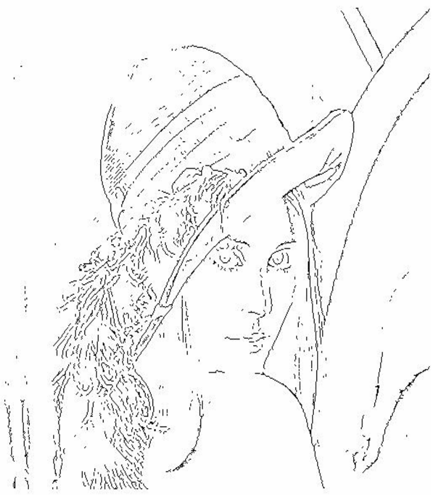


(d)

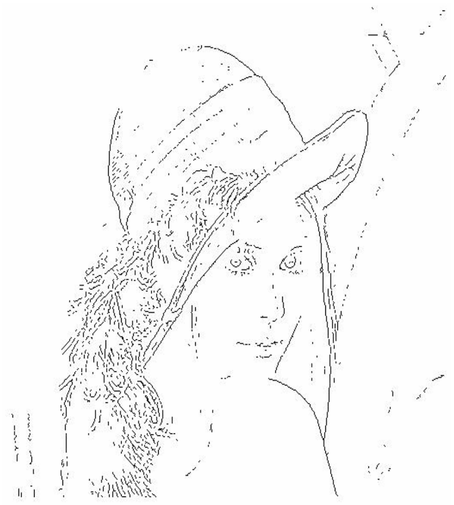
Fig.5.7: GA Pepper edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



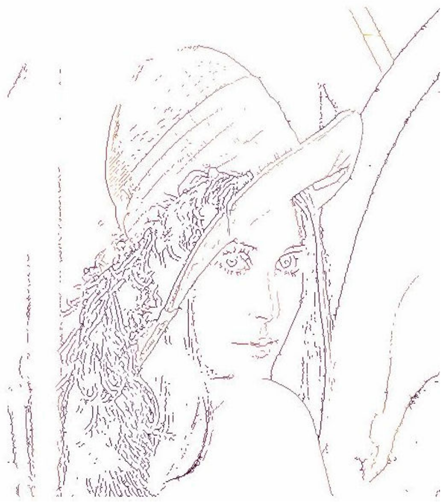
(a)



(b)

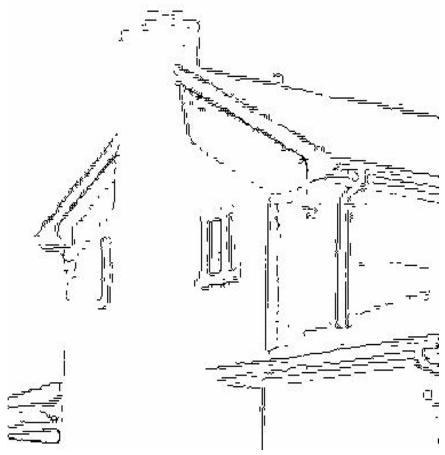


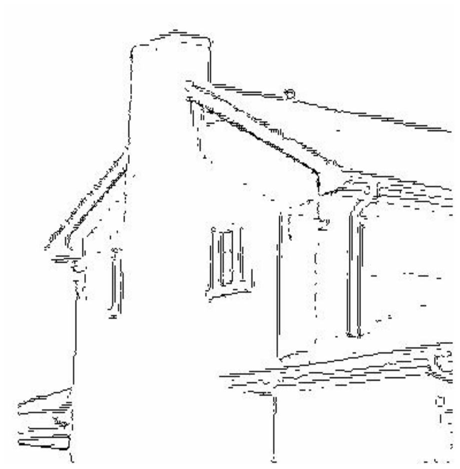
(c)

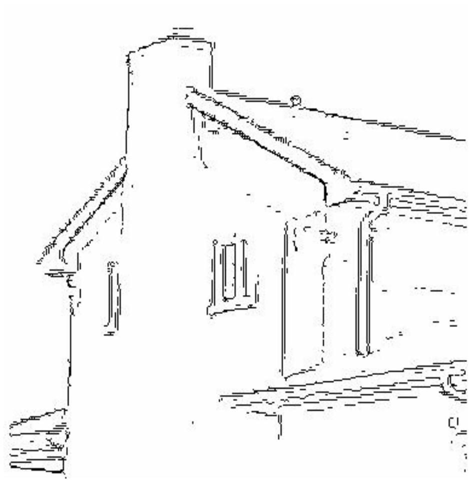


(d)

Fig.5.8: PSO Lena edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.


(a)


(b)


(c)

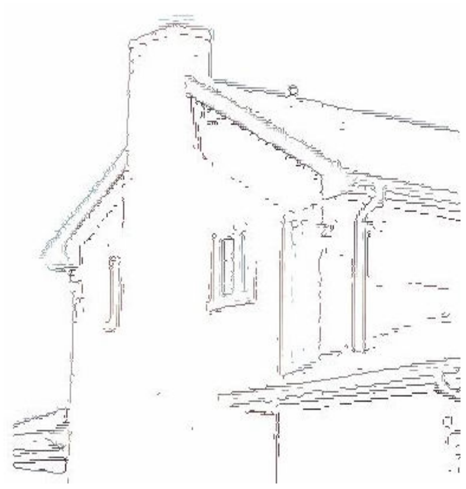
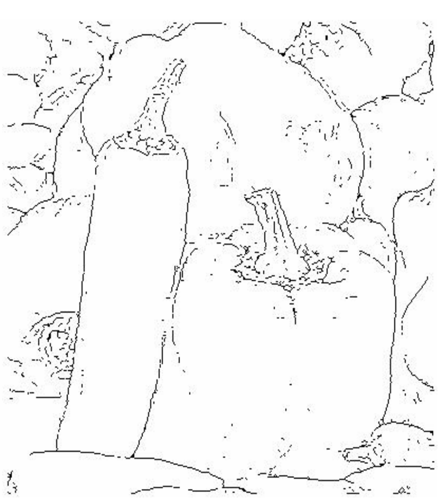
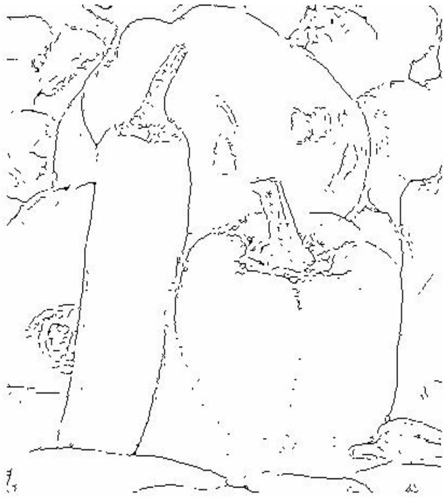

(d)

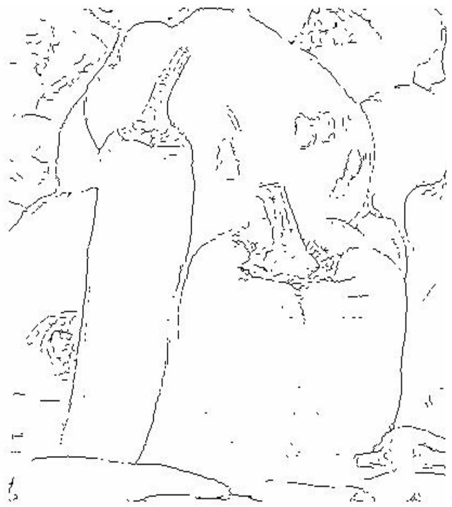
Fig.5.9: PSO House edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



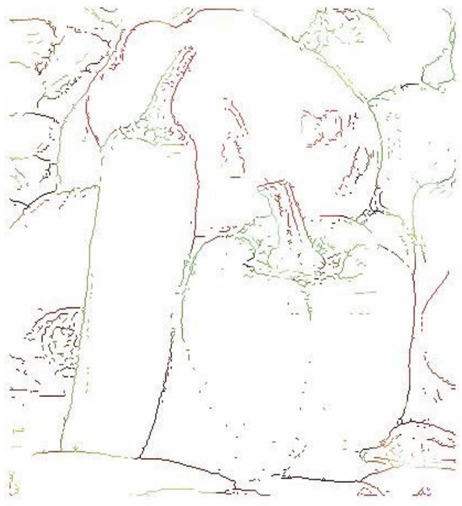
(a)



(b)

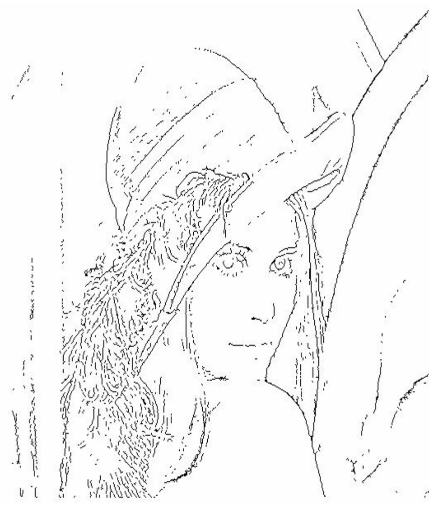


(c)

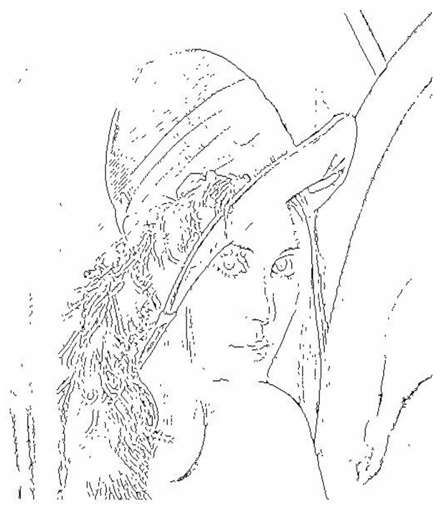


(d)

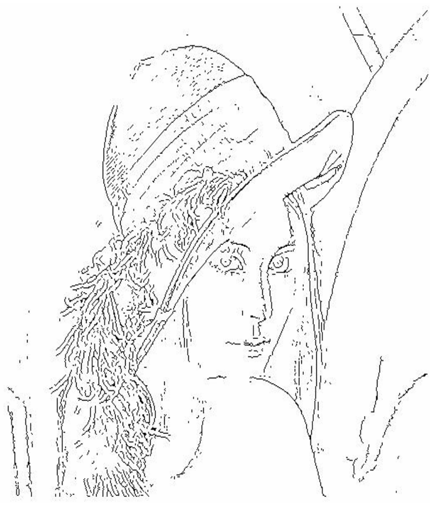
Fig.5.10: PSO Pepper edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



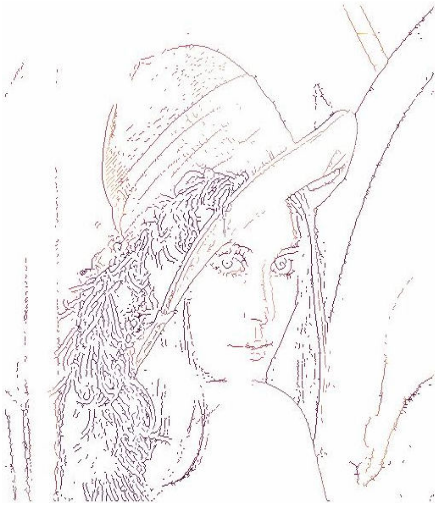
(a)



(b)

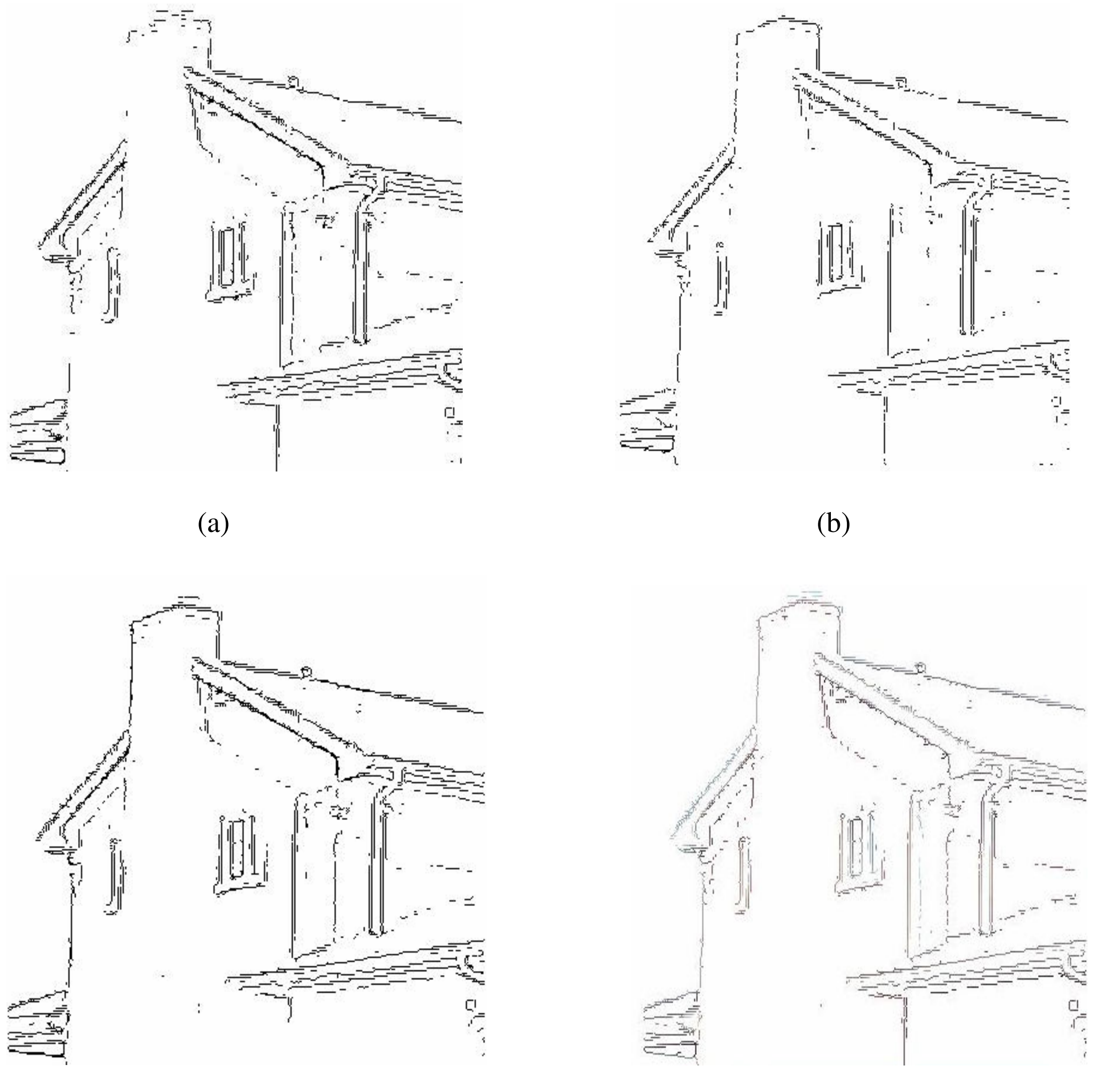


(c)



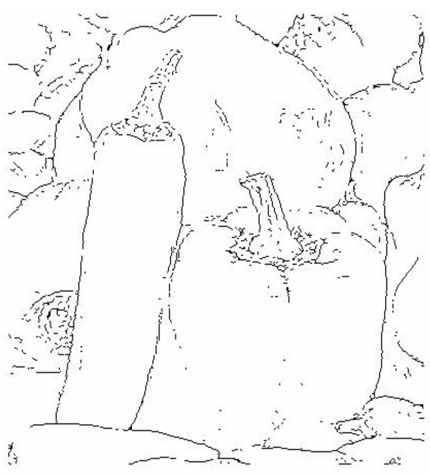
(d)

Fig.5.11: GSA Lena edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.

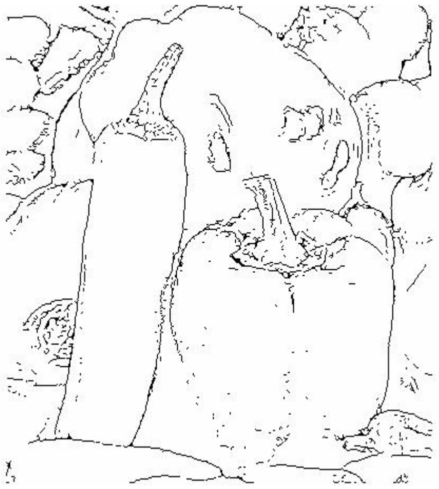


(a) (b) (c) (d)

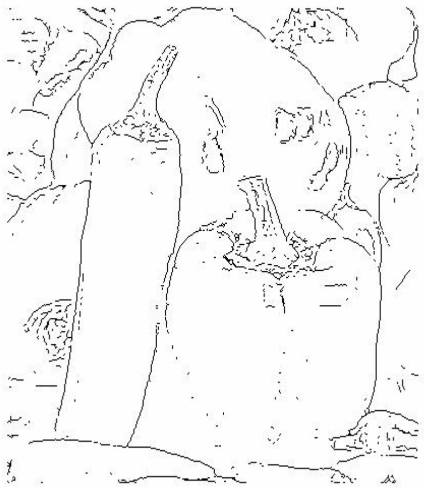
Fig.5.12: GSA House edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.



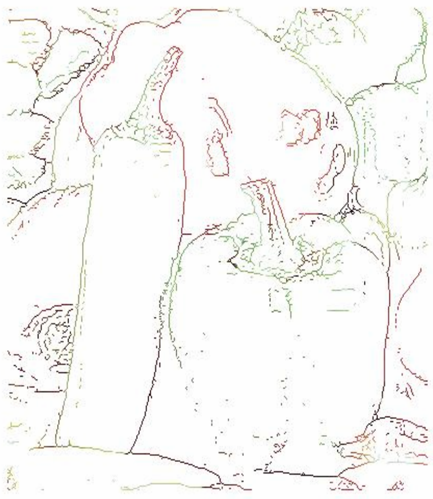
(a)



(b)



(c)



(d)

Fig.5.13: GSA Pepper edge maps: (a) Red component (b) Green component (c) Blue component (d) Final color edge map.

The brightness of threshold'*t*' and the *mean* values selected for the test images are listed in Table 1.

Table1: Selected values for threshold'*t*' and *mean*

Test Image	**Threshold '*t*'**	***mean***
Lena	22	0.10
House	22	0.12
Pepper	20	0.14

It is observed from the results that the proposed SUSAN principle based method is well suitable for edge detection in color images. The detected edge map for sample images contains both weak and strong edges with negligible number of false edges. The method does not distort the original shape of the objects in the image while detecting the edges as well as reduces false edges by distinguishing between the edge and non-edge pixels. For instance, in the case of Lena image, the expression on the face is clear and details of the image is also quite noticeable.

The connectivity between the edge pixels in the edge map is best maintained with BF and comes out worst with GA. In terms of goodness of the edge connectivity, results obtained using different evolutionary algorithms can be ranked in the order: BF >GSA >PSO>GA. Tables 2 to 4 show the initial and optimal values of the parameters: *a, b*, *c* and f_h used in the modified bell- shaped membership function to fuzzify the three components of the color image.

Table 2: Initial and Optimized Values of Parameters (**Red** Component)

Lena Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	11.523	-0.653	4.950	1.533	20.355
GA	-0.5	4	1.5	11.523	-1.459	5.045	2.520	11.394
PSO	-0.5	4	1.5	11.523	-0.591	3.856	1.350	15.087
GSA	-0.5	4	1.5	11.523	-0.539	3.940	1.360	16.656
House Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	10.097	-0.637	4.870	1.563	20.204
GA	-0.5	4	1.5	10.097	-1.462	5.230	2.357	10.377
PSO	-0.5	4	1.5	10.097	-0.562	4.350	1.302	15.180
GSA	-0.5	4	1.5	10.097	-0.489	3.970	1.357	15.453
Pepper Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	12.438	-0.596	4.576	1.682	19.626
GA	-0.5	4	1.5	12.438	-1.489	5.113	2.481	12.653
PSO	-0.5	4	1.5	12.438	-0.558	4.122	1.374	16.105
GSA	-0.5	4	1.5	12.438	-0.545	3.984	1.480	15.934

Table 3: Initial and Optimized Values of Parameters (**Green** Component)

Lena Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	11.576	-0.551	4.702	1.532	20.264
GA	-0.5	4	1.5	11.576	-1.497	4.908	2.514	11.281
PSO	-0.5	4	1.5	11.576	-0.564	4.390	1.402	16.778
GSA	-0.5	4	1.5	11.576	-0.549	3.692	1.524	13.374
House Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	9.887	-0.537	4.703	1.543	20.203
GA	-0.5	4	1.5	9.887	-1.487	5.395	2.630	10.521
PSO	-0.5	4	1.5	9.887	-0.545	3.625	1.371	14.721
GSA	-0.5	4	1.5	9.887	-0.494	4.295	1.545	15.687
Pepper Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	11.226	-0.595	4.700	1.650	19.625
GA	-0.5	4	1.5	11.226	-1.453	4.985	2.360	11.741
PSO	-0.5	4	1.5	11.226	-0.550	4.071	1.367	17.327
GSA	-0.5	4	1.5	11.226	-0.491	4.151	1.371	11.862

Table 4: Initial and Optimized Values of Parameters (**Blue** Component)

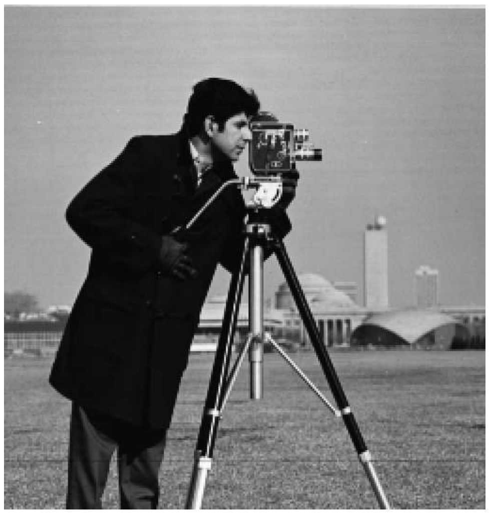
Lena Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	11.855	-0.541	4.702	1.738	20.653
GA	-0.5	4	1.5	11.855	-1.548	5.399	2.401	11.596
PSO	-0.5	4	1.5	11.855	-0.506	3.697	1.362	16.103
GSA	-0.5	4	1.5	11.855	-0.519	4.271	1.467	17.298
House Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	9.968	-0.547	4.703	1.653	20.153
GA	-0.5	4	1.5	9.968	-1.493	5.464	2.494	11.580
PSO	-0.5	4	1.5	9.968	-0.502	4.181	1.349	15.932
GSA	-0.5	4	1.5	9.968	-0.514	3.884	1.421	12.978
Pepper Image								
Evolutionary Technique	**Initialized Parameters**				**Optimized Parameters**			
	a	**b**	**c**	**f_h**	**a**	**b**	**c**	**f_h**
BF	-0.5	4	1.5	11.143	-0.525	4.701	1.750	19.614
GA	-0.5	4	1.5	11.143	-1.491	5.265	2.497	10.997
PSO	-0.5	4	1.5	11.143	-0.513	4.294	1.363	15.654
GSA	-0.5	4	1.5	11.143	-0.522	4.048	1.648	15.710

5.2 Results for Fuzzy Derivative based Method

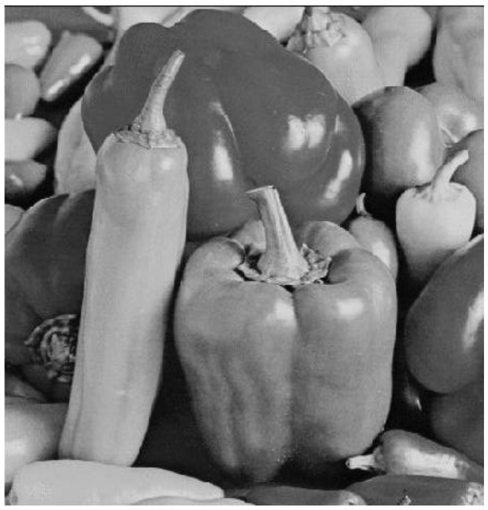
The proposed fuzzy derivative based method has been implemented in matlab for detecting the edge map in grayscale images. No preprocessing is required prior to the application of this algorithm. The performance of the algorithm depends upon the values of the parameters: *mean*, α, β, γ, *a* and *b*.

The local neighborhood size and the *mean* value of the adaptive thresholding also affect the performance of the algorithm. Experimentally, a (10x10) neighborhood size is found to give good results. Increasing the *mean* value leads to the removal of additional noise but at the cost of losing some edges. Decreasing its value replenishes the desired information. The original images of Lena, Cameraman, Pepper and Pillset used as test images are shown in Figure 5.14. The results obtained for the test images using different variants of this method are shown in Figures 5.15 - 5.18.

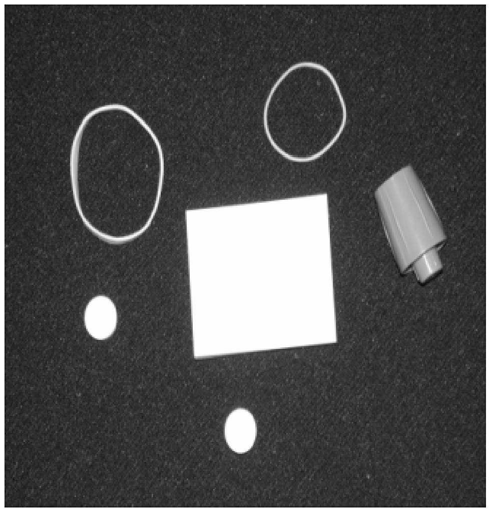
(a)



(b)

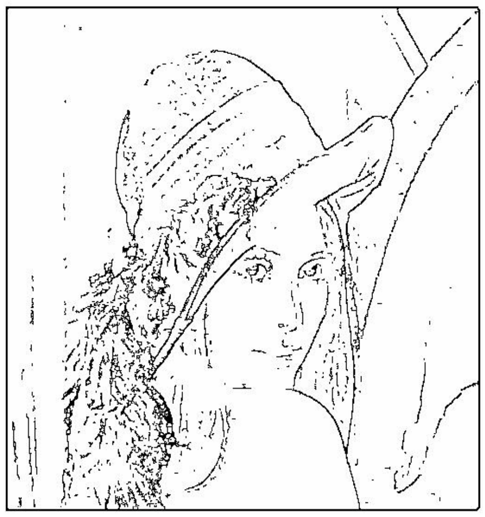


(c)

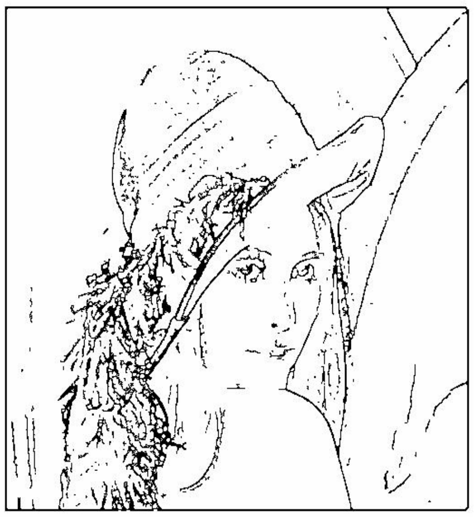


(d)

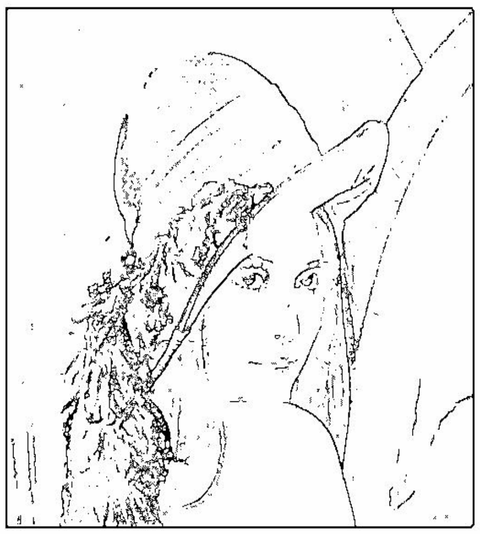
Fig.5.14: Original (a) Lena Image (b) Cameraman Image (c) Pepper Image (d) Pillset Image



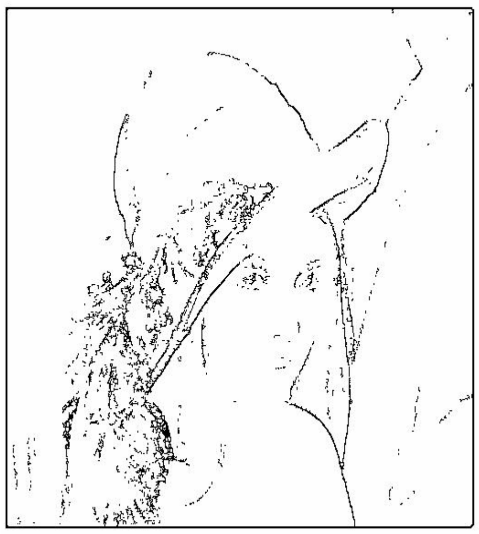
(a)



(b)

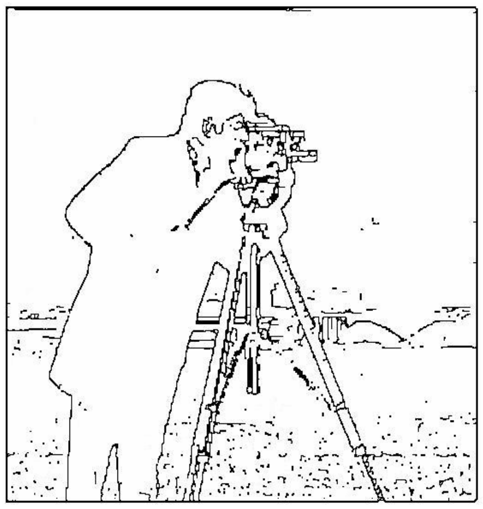


(c)

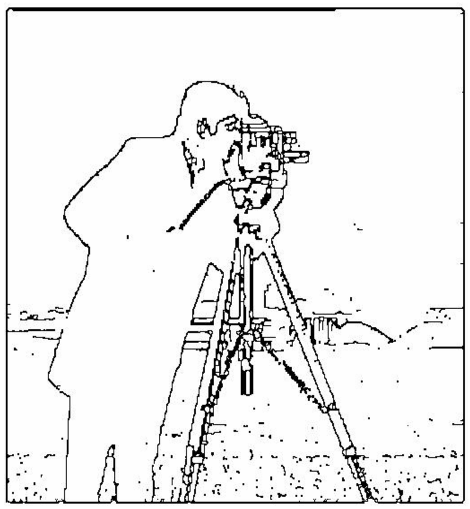


(d)

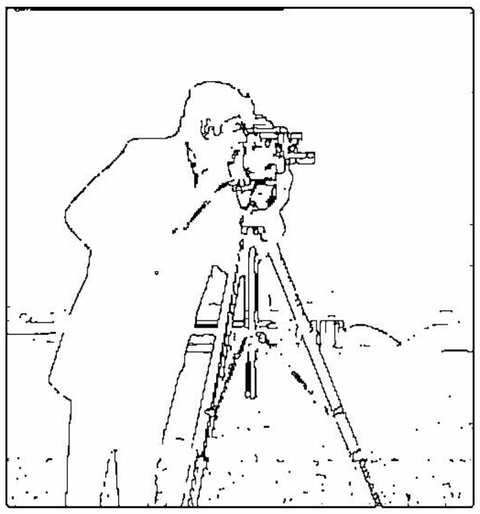
Fig.5.15: Lena Results: (a) BF edge map (b) GSA edge map (c) PSO edge map (d) GA edge map



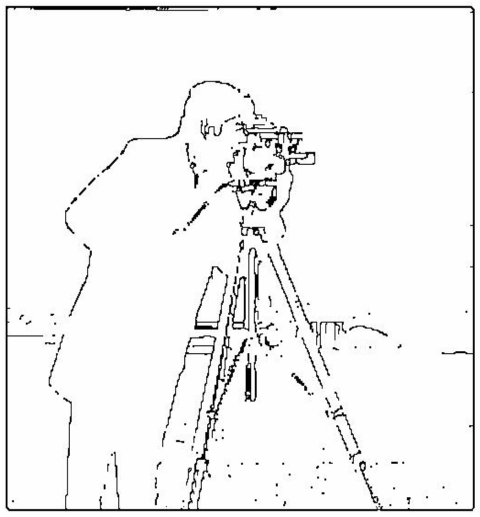
(a)



(b)

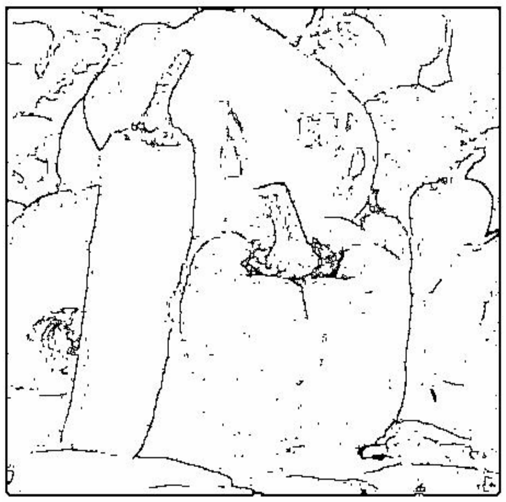


(c)

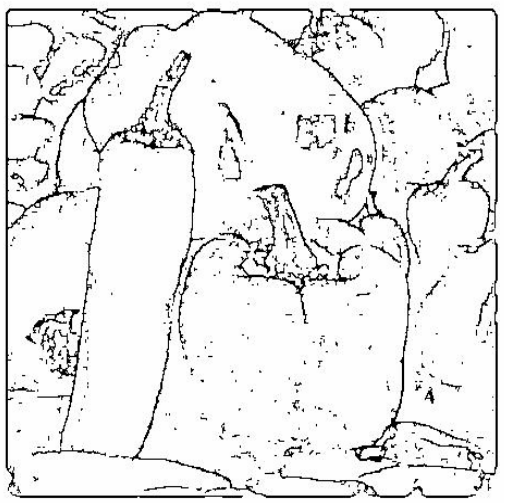


(d)

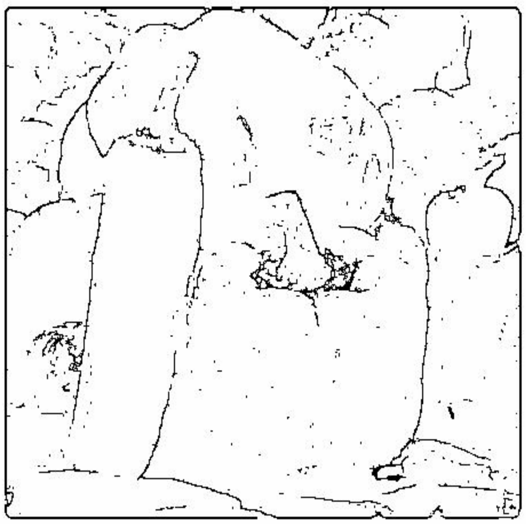
Fig.5.16: Cameraman Results: (a) BF edge map (b) GSA edge map (c) PSO edge map (d) GA edge map



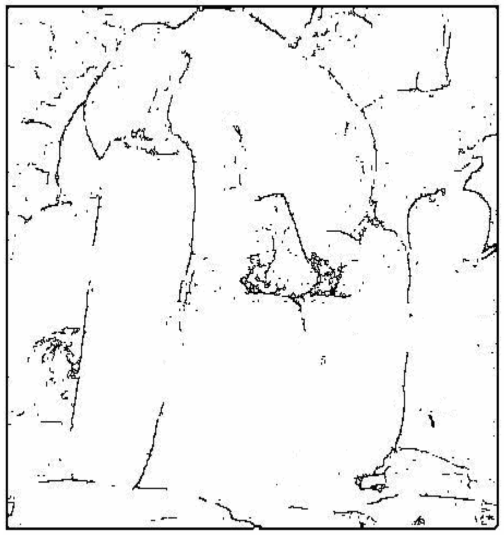
(a)



(b)

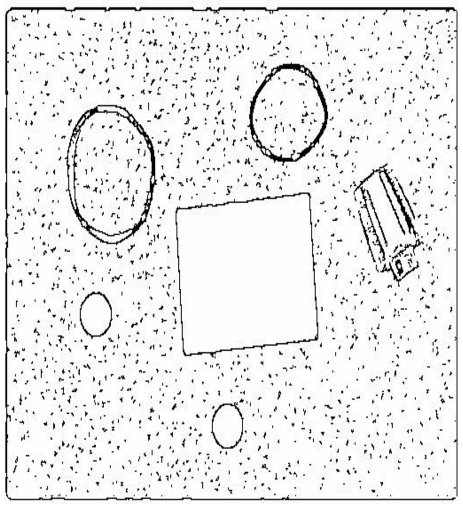


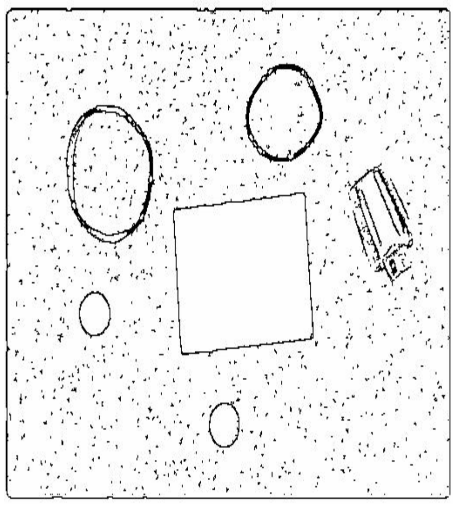
(c)

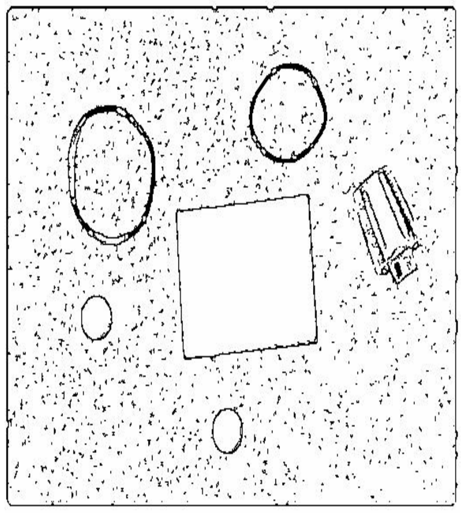


(d)

Fig.5.17: Pepper Results: (a) BF edge map (b) GSA edge map (c) PSO edge map (d) GA edge map


(a)


(b)


(c)

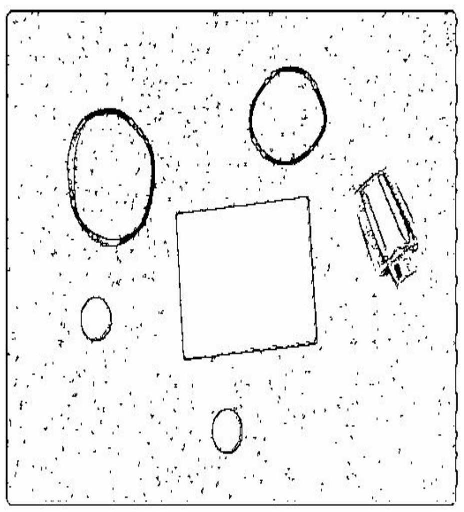

(d)

Fig.5.18: Pillset Results: (a) BF edge map (b) GSA edge map (c) PSO edge map (d) GA edge map

The *mean* value selected during adaptive thresholding for the test images are listed in Table 5.

Table 5: Selected values for *mean*

Test Image	***mean***
Lena	0.05
Cameraman	0.07
Pepper	0.15
Pillset	0.2

It is observed from the results that the proposed fuzzy derivative based method is able to detect edge pixels in grayscale images. The detected edge map for sample images contains strong edges with moderate number of false edges. The method is able to reduce false edges in the image by distinguishing between the edge and non-edge pixels.

Tables 6 to 10 show the initial and optimal values of the objective function *J* and the parameters: α, β, γ, *a, b* used in the HIGH membership function to obtain the image in the fuzzy domain.

Table 6: Initial Values of Parameters and Objective Function *J*

Test Image	**α**	**β**	**γ**	**a**	**b**	***J***
Lena	1	100	4	0.5	0.9	0.4074
Cameraman	1	100	4	0.5	0.9	0.1570
Pepper	1	100	4	0.5	0.9	0.4164
Pillset	1	100	4	0.5	0.9	0.1944

Table 7: Optimized Values of Parameters and Objective Function *J* using BF

Test Image	**α**	**β**	**γ**	**a**	**b**	***J***
Lena	0.9064	101.2279	4.4760	0.4603	0.9609	0.2503
Cameraman	1.0906	101.3132	4.4411	0.4802	0.9426	0.0707
Pepper	1.0371	100.0007	4.4963	0.4747	0.9327	0.4025
Pillset	1.0985	100.0456	4.4500	0.4926	0.9996	0.0952

Table 8: Optimized Values of Parameters and Objective Function *J* using GA

Test Image	**α**	**β**	**γ**	**a**	**b**	***J***
Lena	1.0964	101.7698	3.5060	0.4024	0.8012	0.4267
Cameraman	0.8024	102.4991	3.4114	0.4136	0.7911	0.1925
Pepper	0.9012	102.1965	3.5095	0.3748	0.7524	0.4283
Pillset	0.8917	103.8763	3.5060	0.4097	0.8032	0.2101

Table 9: Optimized Values of Parameters and Objective Function *J* using PSO

Test Image	**α**	**β**	**γ**	**a**	**b**	***J***
Lena	0.9773	99.9673	4.3074	0.5137	0.8902	0.4011
Cameraman	1.0493	99.8506	3.6075	0.4788	0.8349	0.1497
Pepper	1.0416	99.9453	4.0751	0.5517	0.8637	0.4106
Pillset	0.9587	99.9911	4.3389	0.5246	0.8492	0.1833

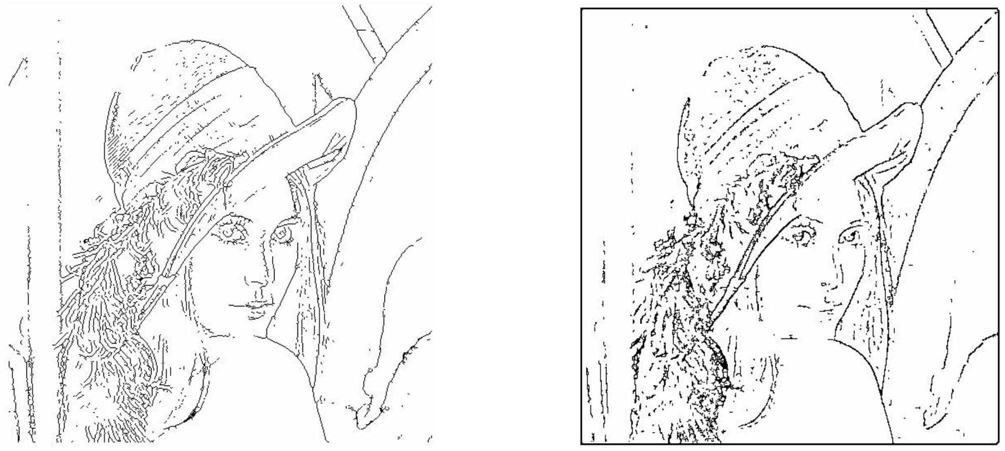
Table 10: Optimized Values of Parameters and Objective Function *J* using GSA

Test Image	**α**	**β**	**γ**	**a**	**b**	***J***
Lena	1.1419	99.9235	4.7950	0.7274	0.9416	0.4004
Cameraman	1.1582	100.5012	4.9507	0.8082	0.9452	0.1321
Pepper	1.1726	99.9253	4.8555	0.7888	0.9319	0.3893
Pillset	0.5270	100.3716	5.8876	0.5964	0.9035	0.0773

In case of Lena image, GSA gives as good result as that obtained using BF and for images like Pepper and Pillset it is even better than BF, PSO and GA results. For the Cameraman image, BF algorithm best optimizes/ minimizes the objective function (as seen from Table 7) and therefore, gives the best result but it contains more number of false edge pixels as compared to GSA, PSO and GA edge maps. The connectivity between the edge pixels in case of PSO result is better than that in GA result but inferior to BF and GSA results. In case of Lena, Cameraman and Pepper images that contain more information, edge connectivity in the edge map obtained using GA is minimum because it optimizes the objective function *J* by maximizing its initial value (as seen from Table 8) and hence, minimizes fuzzy entropy below its expected value. However, GA results also contain minimum number of false edges as compared to GSA, BF and PSO edge maps. In general, the connectivity between the edge pixels in the edge map is best maintained with GSA and comes out worst with GA. In general, results obtained using different evolutionary algorithms can be ranked in the order: GSA >=BF >PSO>GA in terms of goodness of the edge connectivity.

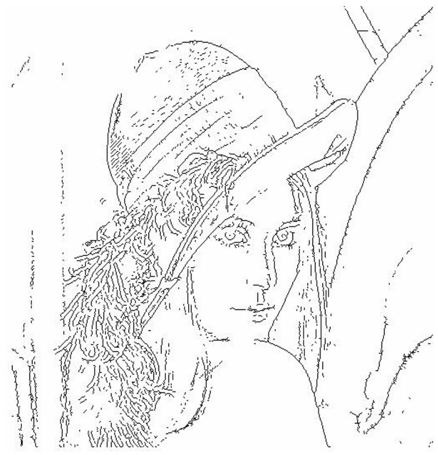
5.3 Comparison between SUSAN principle and Fuzzy Derivative based Methods

To subjectively analyze the quality of the edge map obtained and to understand the effectiveness of the two proposed methods, visual comparison is between the results obtained using different variants of the two methods. For comparison purpose, experimental results for Lena and Pepper images have been considered.



(a) (b)

Fig.5.19: BF Lena Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method


(a)

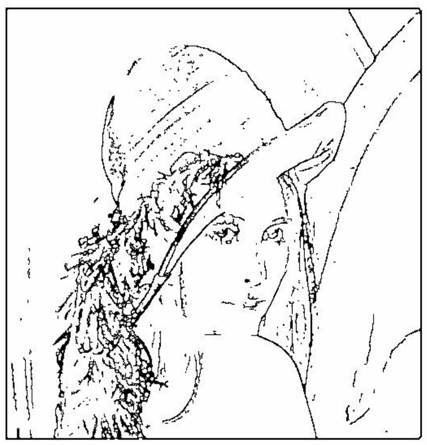
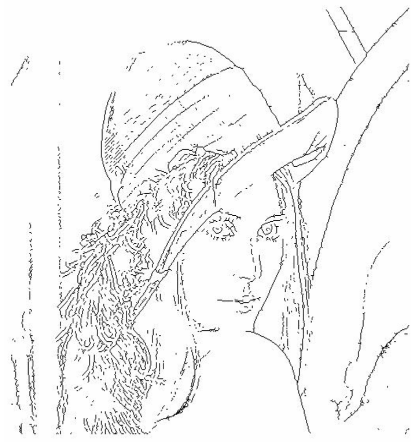

(b)

Fig.5.20: GSA Lena Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method


(a)

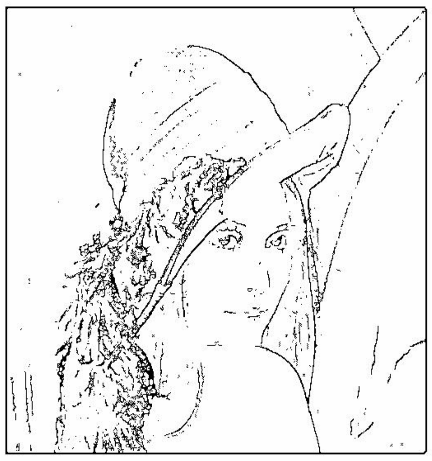
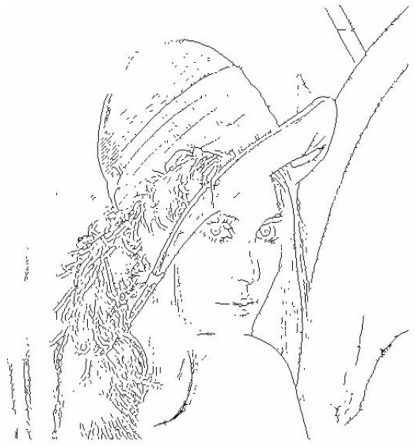
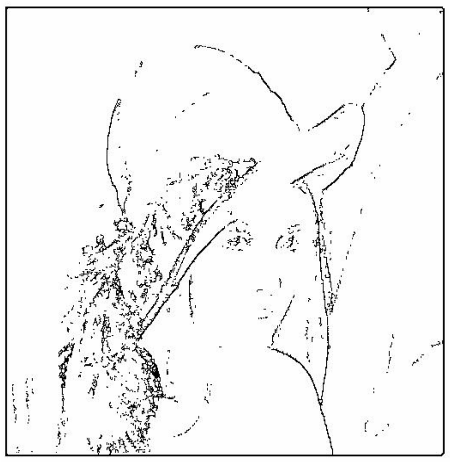

(b)

Fig.5.21: PSO Lena Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method

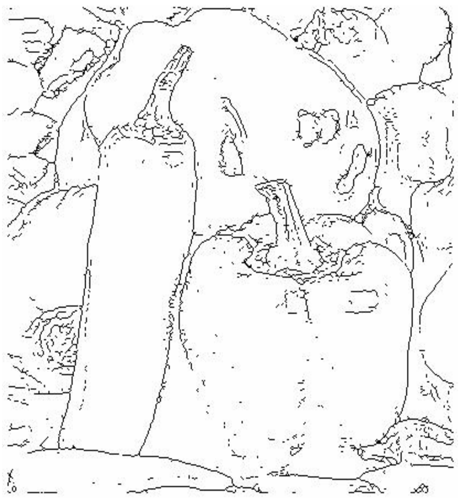


(a)

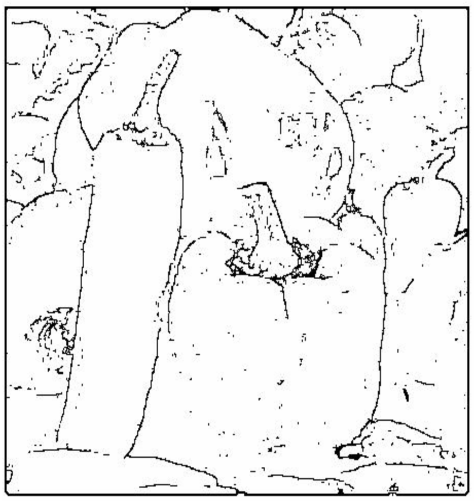


(b)

Fig.5.22: GA Lena Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method

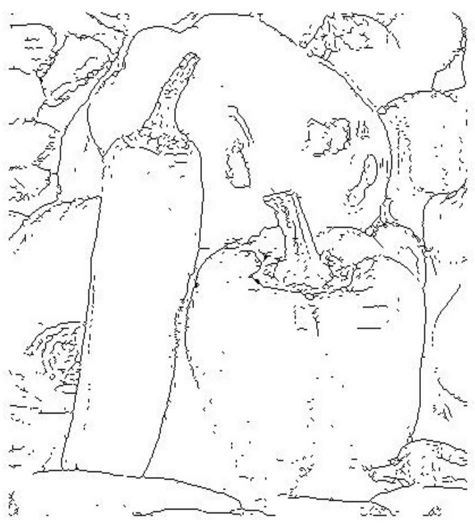


(a)



(b)

Fig.5.23: BF Pepper Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method


(a)

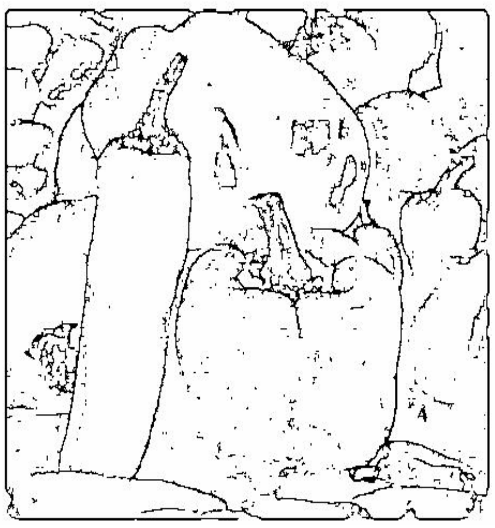
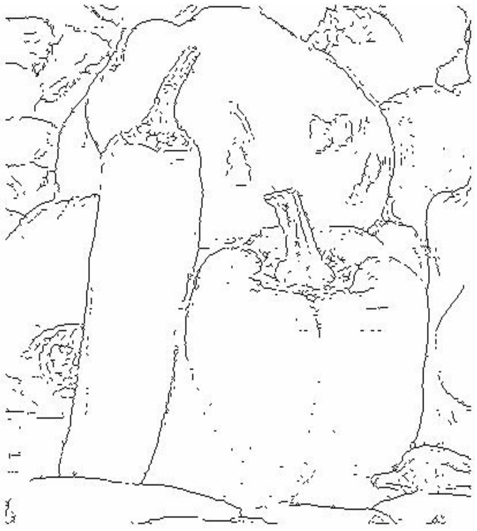

(b)

Fig.5.24: GSA Pepper Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method


(a)

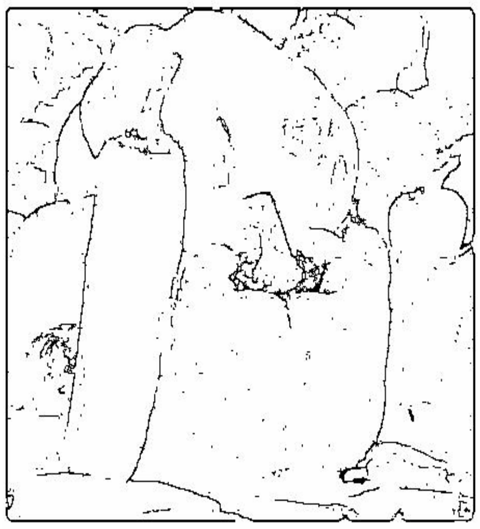
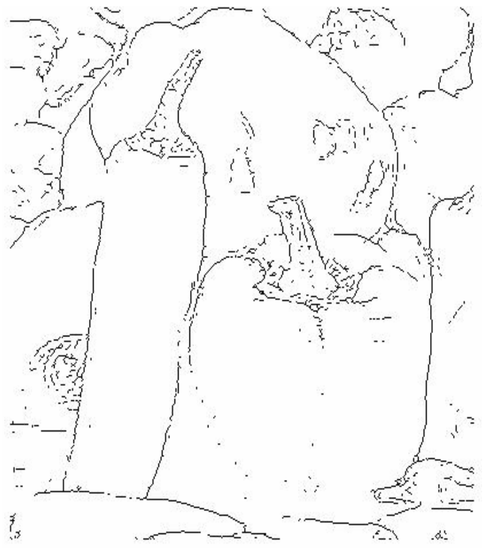

(b)

Fig.5.25: PSO Pepper Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method


(a)

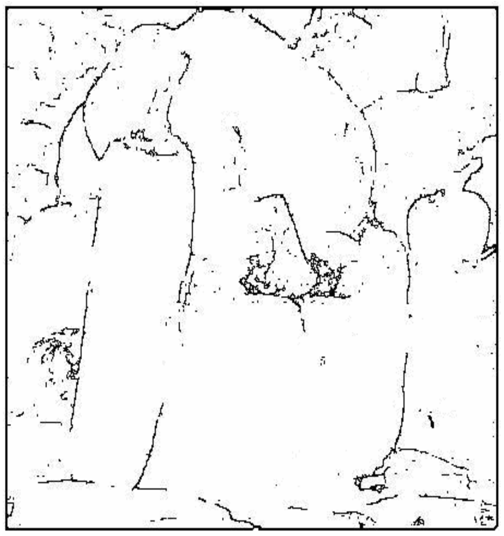

(b)

Fig.5.26: GA Pepper Results: (a) Proposed SUSAN Method (b) Proposed Fuzzy Derivative Method

It is observed that the sharpness of the edges as well as clarity of the details in the edge map obtained using proposed SUSAN principle based method (method 1) is much better than that present in the edge map obtained using proposed fuzzy derivative based method (method 2).

Method 1 is able to detect strong as well weak edges in the image whereas method 2 detects only strong edges. Also the connectivity between the edge pixels is very good in the first method results as compared to that obtained using second method. For the Lena image, in particular, the edges of the face are clearly detected using first method making it suitable for contour detection in biometric applications which is not possible with the second method.

Chapter 6
Conclusion

The current work has been focused on detecting edge map in color as well as grayscale images by proposing two methods. The first method which is based on SUSAN principle uses fuzzy logic and evolutionary techniques to effectively detect edge map in the color images. It uses Histogram based Gaussian membership function and the modified bell shaped function to locate both strong (large segments) and weak (small segments) edges respectively. The parameters involved in the modified bell shaped function are optimized using different evolutionary techniques. Adaptive thresholding helps localize the edges in order to obtain the final edge map of an image. In this work the address of the edge pixels is collected during the edge detection to derive the color edge map of the original color image.

The second method which is based on fuzzy derivative uses evolutionary algorithms to detect edge map in the grayscale images. The derivative matrix is fuzzified using bell shaped membership function and HIGH membership function helps locate strong edges in the image. The parameters involved in the bell shaped and HIGH membership functions are optimized using different evolutionary algorithms. Adaptive thresholding helps localize the edges in order to obtain the final edge map of an image.

On the basis of the ability to preserve the object shape and to retain the important edges, the usefulness of the first method to biometric applications like face detection and palmprint identification can be easily recognized. First method gives more clear and sharp edges as compared to that obtained using second method.

Selection of some of the parameters i.e. *a*, *b*, *c*, threshold‘*t*’ for the first method and α, β, γ, *a, b* for the second method along with the *mean* value during adaptive thresholding is crucial for the success of both the methods.

The future scope of the work is to explore other evolutionary algorithms available in the literature for parameter optimization as well as to extend both the proposed methods for the noisy images by means of incorporating filtering stage in the two methods.

References

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing. Reading,MA: Addison-Wesley, 1992.

[2] Canny J. “A computational approach to edge detection”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, Vol. 8, Issue 1, pp. 679-698.

[3] Marr, D., and Hildreth, E.C., “Theory of edge detection”, Proceedings of the Royal Society of London, Royal Society, Great Britain, Feb. 29, 1980, pp. 187-217.

[4] Yuqin Yao; Hui Ju,” A Sub - pixel edge detection method based on canny operator”, Proceedings of 6th International Conference on Fuzzy Systems and Knowledge Discovery, Tianjin, 16-18 Aug. 2009, pp. 97 – 100.

[5] Basu M, ”Gaussian Based Edge Detection Methods: A Survey”, IEEE Transactions on systems, man and cybernetics, 2002,Vol. 32, Issue 3, pp. 252 – 260.

[6] R.A. Salinas, C. Richardson, M.A. Abidi, and R.C. Gonzalez, “Data Fusion: Color Edge Detection and Surface Reconstruction through Regularization,” IEEE Transactions on Industrial Electronics, 1996, Vol. 43, Issue 3, pp. 355-363.

[7] Perona P., Malik J.,” Scale-Space and Edge Detection Using Anisotropic Diffusion”, IEEE Transactions on pattern analysis and machine intelligence, 1990, Vol. 12, Issue 7, pp. 629-639.

[8] Jiang X., Bunke H,” Edge Detection in Range Images Based on Scan Line Approximation”, Journal on Computer Vision and Image Understanding, February 1999, Vol. 73, Issue 2, pp.183–199.

[9] Bezdek, J.C.,Chandrasekhar, R., and Attikiouzel,” A geometric approach to edge detection”, IEEE Transactions on Fuzzy Systems, 1998, Vol. 6, Issue l, pp. 52- 75

[10] Russo, F,” FIRE operators for image processing”, Fuzzy Sets and Systems, 1999, Vol. 103, Issue 2, pp. 265-275.

[11] Bloch I., “Fuzzy sets in image processing”, Proceedings of ACM Symposium on Applied Computing, New York, USA, March 6-8, 1994, pp. 175 - 179.

[12] Ho, K.H.L., and Ohnishi, N., “FEDGE: Fuzzy edge detection by fuzzy categorization and classification of edges”, Fuzzy Logic in Artificial Intelligence Towards Intelligent Systems, Lecture Notes in Computer Science, 1997, SpringerLink, Vol. 1188, pp. 182 – 196.

[13] Zhang Jinping; Lian Yongxiang; Dong Linfu; Zhao Xueguang, Liu Jie;” A New Method of Fuzzy Edge Detection Based On Gauss Function”, Proceedings of 2nd International Conference on Computer and Automation Engineering, Singapore, Feb 26-28, 2010, pp. 559 – 562.

[14] Pal S K, King R A. On edge detection of X-ray images using fuzzy sets. IEEE Trans. Pattern Analysis and Machine Intelligence, 1983, Vol. 5, Issue 1, pp. 69-77.

[15] S. E. El-Khamy, I. Ghaleb, N. A. El-Yamany, “Fuzzy edge detection with minimum fuzzy entropy criterion,” Electrotechnical Conference MELECON, 9-9 May, 2002, Mediterranean, pp. 498-503.

[16] Yong Yang, “An Adaptive Fuzzy-based Edge Detection Algorithm” Proceedings of International Symposium on Intelligent Signal Processing and Communication System, Xiamen, Nov. 28- Dec 01, 2007, pp.276-279.

[17] Melin P.; Mendoza O.; Castillo C;**”** An improved method for edge detection based on interval type-2 fuzzy logic”, Journal on Expert Systems with Applications, Elsevier Publications, 2010, Vol.37, Issue 12, pp. 8527-8535.

[18] Daode Zhang; Bisheng Zhan; Guangyou Yang; Xinyu Hu,” An Improved Edge Detection Algorithm Based On Fuzzy Image Enhancement”, Proceedings of 4th IEEE Conference on Industrial Electronics and Applications, Xi'an, 25-27 May, 2009, pp. 2412 – 2415.

[19] Jinbo Wu, Zhouping Yin, “The Fast Multilevel Fuzzy Edge Detection of Blurry Images,” IEEE signal processing letters, 2007, Vol. 14, Issue 5, pp. 344- 347.

[20] Xiangtao Chen; Yujuan Chen," An Improved Edge Detection in Noisy Image Using Fuzzy Enhancement", Proceedings of International Conference on Biomedical Engineering and Computer Science, Wuhan, April 23-24, 2010, pp.1-4.

[21] Tizhoosh, H.R.,"Fast Fuzzy Edge Detection", Proceedings of Annual Meeting of the North American Fuzzy Information Processing Society, Waterloo University, Canada, June 27-29, 2002, pp. 239 – 242.

[22] Lihui Jiang, "Image edge detection based on fuzzy weighted morphological filter", Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, July 12-15, 2009, pp. 690 – 693.

[23] barkhoda, W., Tab, F.A., Shahryari, O.K.," Fuzzy edge detection based on pixel's gradient and standard deviation values", Proceedings of International Multiconference on Computer Science and Information Technology, Mragowo, Oct 12-14, 2009, pp. 7 – 10.

[24] Yishu Zhai; Xiaoming Liu, "Multiscale Edge Detection Based on Fuzzy C-Means Clustering", Proceedings of 1st International Symposium on Systems and Control in Aerospace and Astronautics, Harbin, June 19-21, 2006, pp. 1201-1204.

[25] K. M. Passino, "Biomimmicry of bacterial foraging for distributed optimization and control," IEEE Control Systems Magazine, 2002, Vol. 22, Issue 3, pp. 52–67.

[26] D.E. Goldberg, Genetic algorithms in search, optimiazation and machine learning, Addison-Wesley, Reading, MA, 1989.

[27] Kennedy, J. and Eberhart, R. C. "Particle swarm optimization" Proceedings of IEEE International Conference on Neural Networks Vol. IV, IEEE service center, Piscataway, NJ, Nov. 27- Dec. 1, 1995, pp. 1942-1948.

[28] Mahdi Setayesh, Mark Johnston, Mengjie Zhang, "Edge and Corner Extraction Using Particle Swarm optimisation", Proceedings of Australasian Conference on Artificial Intelligence, Adelaide, Australia, Dec. 7-10, 2010, pp. 323-333.

[29] Khalid, N.E.A.; Manaf, M.; Aziz, M.E.," Fusion of Fuzzy Heuristic and Particle Swarm Optimization as an edge detector", Proceedings of International Conference on Information Retrieval & Knowledge Management, Shah Alam, Selangor, March 17-18, 2010, pp. 250 – 254.

[30] S.M. Bhandarkar, Y. Zhang, and W.D. Potter, "An Edge Detection Technique Using Genetic Algorithm-Based Optimization", Pattern Recognition, Elsevier Publications, 1994, Vol. 27, Issue 9, pp. 1159-1180.

[31] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents", Part B: Cybernetics, IEEE Transactions on Systems, Man, Cybernetics, 1996, Vol. 26, Issue 1, pp. 29–41.

[32] Verma, O.P.; Hanmandlu, M.; Sultania, A.K.; Dhruv, D.;"A Novel Fuzzy Ant System For Edge Detection", Proceedings of 9th International Conference on Computer and Information Science, Yamagata, Japan, March 6-7, 2010, pp.228 - 233.

[33] L. De-Sian, and C. Chien-Chang, "Edge detection improvement by ant colony optimization", Pattern Recognition Letters, Elsevier Publications, 2008, Vol.29, Issue 4, pp. 416– 425.

[34] Verma, O.P.; Hanmandlu, M.; Kumar, P.; Chhabra, S.; Jindal, A.;" A Novel Bacterial Foraging Technique for Edge Detection", Pattern Recognition Letters, Elsevier Publications, 2011, Vol.32, Issue 8, pp.1187-1196.

[35] Yoshimura, M.; Oe, S.; "Edge detection of texture image using genetic algorithms "Proceedings of 36th SICE Annual Conference, Tokushima, July 29-31, 1997, pp.1261 - 1266.

[36] E.Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm", Journal on Information Science, Elsevier Publications, 2009, Vol. 179, Issue13, pp. 2232-2248.

[37] G. Sun, Q.h. Liu, Q. Liu, C. Ji, X. Li, "of A novel approach for edge detection based on the theory universal gravity", Pattern Recognition Journal, Elsevier Publications ,2007, Vol.40, Issue 10, pp. 2766 – 2775.

[38] Smith, S.M., and Brady, J.M., "SUSAN: A new approach to low level image processing", International Journal of Computer Vision, 1997, Vol. 23, Issue 1, pp.45-78.

[39] Hanmandlu, M.; Verma, O.P.; Kumar, N.K.; Kulkarni, M.," A Novel Optimal Fuzzy System for Color Image Enhancement Using Bacterial Foraging", IEEE Transactions on Instrumentation and Measurement, 2009, Vol. 58, Issue 8, pp. 2867 – 2879.

[40] Madasu, H. Verma, O.P. Gangwar, P. Vasikarla, S.," Fuzzy Edge and Corner Detector for Color Images", Proceedings of 6th International Conference on Information Technology: New Generations, Las Vegas, NV, 2009, pp. 1301 – 1306.

[41] Verma O.P., Hanmandlu M., Kumar P., Shrivastava S., 'A novel approach for edge detection using ant colony optimization and fuzzy derivative technique' ,Advance Computing Conference,2009.IEEE international, 6-7 March 2009**,** pp.1206 - 1212.

[42] Madasu, V.K. ; Vasikarla, S. ; "Fuzzy Edge Detection in Biometric Systems", 36th IEEE Applied Imagery Pattern Recognition Workshop, 10-12 Oct. 2007, Washington, DC , pp.139-144.

[43] Hanmandlu, M.; Verma, O.P.; Kumar, N.K.; Kulkarni, M.," A Novel Optimal Fuzzy System for Color Image Enhancement Using Bacterial Foraging", IEEE Transactions on Instrumentation and Measurement, 2009, Vol. 58, Issue 8, pp. 2867 – 2879.