A project report on

EDGE DETECTION IN DIGITAL IMAGES USING EVOLUTIONARY ALGORITHM

Submitted in partial fulfillment of the Requirement for the award of degree of

Master of Technology

In

Information Systems

Submitted By:

Gurpreet Kaur

(2K13/ISY/06)

Under the Guidance of:

Mr. Anil Singh Parihar

(Assistant Professor, Department of Computer Science and Engineering)

2013-2015

Department of Computer Science and Engineering

Delhi Technological University

Bawana Road, Delhi – 110042

CERTIFICATE

This is to certify that Ms. Gurpreet Kaur (2K13/ISY/06) has carried out the major project titled "**Edge Detection In Digital Images Using Evolutionary Algorithm**" 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 2013-2015. The matter contained in this report has not been submitted elsewhere for the award of any other degree.

(Project Guide)

Mr. Anil Singh Parihar

Assistant Professor

Department of Computer Science and Engineering

Delhi Technological University

Bawana Road, Delhi-110042.

ACKNOWLEDGEMENT

I express my gratitude to my major project guide **Mr. Anil Singh Parihar**, Assistant Professor, Department of Computer Science and Engineering, 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.

Gurpreet Kaur

2K13/ISY/06

M.Tech (Information Systems)

E-mail: g.kaur.komal@gmail.com

ABSTRACT

An edge is an abrupt change in intensity that indicates the boundary between two adjacent regions. Its detection is the fundamental operation in image processing. Evolutionary algorithms can be used for efficient results. Evolutionary algorithms are a set of algorithms that are inspired from nature. Various algorithms have been developed in the past few years. Artificial Bee Colony algorithm is one such algorithm that copies the behavior of bees in the nature. It is used for detection of edges in the gray scale digital images. It uses the concept of 3 types of bees in the bee colony – Employee Bees that searches for the food source in the search space and inform about it to the colony, Onlooker Bees that selects one location among the previously informed locations based on the fitness values and search for food source in its neighborhood and Scout Bees move to random locations. The proposed algorithm results in edge-map that contains no thick edges. Also the corners are not missed. It is faster than edge detectors that use Particle Swarm Optimization algorithm, Ant Colony Optimization algorithm or Gravitational Search algorithm. The algorithm is then tested on different sample images and results have been compared to the results of the most commonly used edge detectors. The technique can be enhanced further for detection of edges in colored images.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER – 1

INTRODUCTION TO EDGE DETECTION

An edge is a set of contiguous pixel positions where abrupt change in intensity occurs. The intensity change indicates the boundary that differentiates two adjacent regions is referred to as edge. For a pixel to be an edge pixel or not is its local property. An image function that can be applied in the neighborhood of the pixels can be used to determine if the pixel is an edge pixel. Main causes of intensity changes include Geometric events and Non-geometric events such as reflection of light, illumination, shadows etc.

Edge detection is extensively used in image processing. It is a fundamental operation in computer vision since it is useful for image segmentation. The main concern of the process lies in the detection of variations in the gray level image. The better results of the edge detector result in better understanding of the complete image. The resultant edge map is mainly used in high-level visual processing. The main purpose of edge detection lies in 3D re-construction, scene segmentation and motion analysis. The accuracy of edge detection is the basis of high-level image processing that is concerned with the operations like object recognition in the image, robot vision, segmentation and image coding. Thus, the edge detector has to be efficient and reliable.

The quality of results generated by edge detectors is dependent on a number of conditions. These factors include –

- Similar intensity objects in the image.

- Lighting conditions.
- Presence of some pattern that would increase the density of edges.
- Threshold selected.
- Noise.

For a different set of data, the values have to be manually changes. Although there are methods that can automatically set these values, yet the results are not as good as they are expected to be.

The characteristics of edge and noise are similar to each other. They both have high frequencies. Due to this factor, the image that has been corrupted by noise is very difficult to process and the edge map generated is not good enough to be used anywhere.

Problems with gradient based edge detectors –

- Corners are often missed.
- Here we have to choose threshold values and width of the mask. Changing the size of the image complicates the setting of these values.
- Different operator is required for different feature.

There are some properties or standards that are to be incorporated in edge detectors by default. These include –

- Edges that are generated have to be defined properly and completely connected
- Presence of any kind of pattern should not be detected. For an edge detector, the patterns generate noisy edge map.
- The resultant edge map and the edges perceived by a human should resemble as much as possible.

➢

CHAPTER – 2

LITERATURE REVIEW

2.1. A FEW GENERAL EDGE DETECTORS

2.1.1. SOBEL OPERATOR[1]

Sobel operator is widely used in abundance for computer vision applications. Sobel operator is very important in edge detection algorithms since it is fast and produces fine edge images. Sobel operator produces an image with emphasis on edges and transitions due to its discrete differential operations and the fact that it approximates the gradients of the values of image intensity function. Sobel operator uses a small and separable, integer valued filter for convolving both horizontal and vertical directions, making it a CPU efficient and therefore fast in terms of execution time. The filter consists of two kernels, each of similar dimensions, $3 * 3$, which are convolved with the image provided for edge detection, resulting in an approximation of the derivatives for changes in both the direction.

For horizontal and vertical changes, kernels are described as –

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The image I can be convolved with the above mentioned kernels to determine G_x and G_y respectively. At each point, magnitude of gradient can be calculated using –

$$G = \sqrt{G_x^2 + G_y^2} \quad (1)$$

The direction of gradients could be calculated at each pixel using –

$$\theta = \tan^{-1}(G_y/G_x) \quad (2)$$

2.1.2. ROBERTS CROSS[2]

Edge detection also done by another common edge detection algorithm, The Roberts cross operator. Roberts cross is one of the oldest edge detectors and like Sobel it also uses differentiation. It uses gradient of an image for approximation through discrete differentiation. The approximation is done by calculating the sum of squares of all the differences of diagonally adjoin pixels. The operation utilizes small and integer valued only kernels but this makes the operator highly sensitive to noise.

To perform edge detection, the image is convolved with the following kernels for determining G_x and G_y :

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

At each point, magnitude of gradient can be calculated using Eq. (1). The gradient's direction can also be calculated at each pixel using Eq. (2).

2.1.3. PREWITT EDGE DETECTOR[3]

Prewitt operator is essentially a discrete differential operator and just like previously discussed two operators it also works by calculating the approximation of the image gradient by convoluting the image with a separable and small kernel that is integer valued in both the directions making it very CPU efficient and computation inexpensive though this form of gradient approximation is very crude when approximating high frequency reasons in convoluted image. Two $3 * 3$ are used by the operator for convolving the image and hence computing the derivative approximations, they are given as G_x for horizontal changes, and G_y for vertical.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

And, $G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

The resultant approximation is further combined for obtaining the gradient magnitude using Eq. (1). And the directions of gradient approximations are calculated using Eq. (2)

2.1.4. CANNY EDGE DETECTOR[4]

Canny edge detector is relatively more complex as compared to the already discussed detector functions since it uses a multistage algorithm which further enables it to detect a wide range of edges present in the inspected image. The Process of edge detection using Canny's algorithm is carried over five steps –

1) *Gaussian filter.*

   First step is to smooth the image in order to remove noise. This step includes applying a masked Gaussian filter to the image. The size of the kernel to be used with the filter is (2k + 1) * (2k + 1) and is calculated using:

$$H_{(i,j)} = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{(i-k-1)^2+(j-k-1)^2}{2\sigma^2}\right)} \quad (3)$$

   The image I is then convolved with H.

2) *Find the intensity gradients of the image.*

   Intensity gradient G and its direction θ of the image are calculated using Roberts, Prewitt or Sobel operators.

3) *Apply non-maximum suppression.*

It is done for removing possible false responses during edge detection, non-maxima suppression is applied. For each pixel in gradient image, following steps are applied:

- The edge strength of current pixels is compared to edge strength of pixels in both the gradient directions, i.e., positive and negative.
- And if the edge strength value of current pixel is larger than the value will be preserved. Else it will be suppressed.

4) *Apply double threshold.*

The two calculated threshold values are used for distinguishing different types of edge pixels. If the gradient value of edge pixel is greater than high threshold value, the pixel is marked as a strong edged pixel. While an edge pixel with value lying between the two thresholds is marked as a weak edged pixel. Whereas any pixel with value lying below the minimum specified threshold is suppressed.

5) *Track edge by hysteresis.*

Now all the edges that are weak as well are not connected are certainly false detections or a stronger part of background or out of focus region that are of no certain interest are suppressed to finalize the edge map produced till now.

The main defects of the traditional algorithm include:

- Smoothening is done using a Gaussian filter over the whole image, which will smooth out the noise but will also smooth the edges as a side effect. This will cause some edges to be falsely reported as weak edges. This will lead to the presence of higher number of isolated in the produced image maps.
- Further calculating gradient amplitude is susceptible to high sensitivity to noise and therefore is capable of detecting fake edges and loss of real edges.

2.1.5. LAPLASIAN OF GAUSSIAN[5]

LOG[5] is also one of the most common edge detection techniques. The main steps include –

1) *Application of Gaussian mask on the image.*

   The Gaussian mask of size $(2k+1)*(2k+1)$ is calculated using Eq. (3). The kernel is then convolved with the image to produce a smoothened image.

2) *Application of Laplacian mask.*

   The Laplacian of image I is given as:

$$L = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{4}$$

   The mask that is used is given as:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \text{ Or } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

   The operation requires much less computation. The kernel can be compiled before convolution so that only one convolution has to be done while applying.

2.2. A FEW OTHER TECHNIQUES INTRODUCED IN LITERATURE

2.2.1 EDGE DETECTION AND THRESHOLDING: A STATISTICAL APPROACH[6]

The specified smoother used is a bivariate version of the very well-known Priestly–Chao kernel smoother. This smoothing function is used for fitting curves when the independent variable is equally spaced in time-series sequences. For the fitted surface, calculation of the gradient vector over every pixel is done. The variance-covariance matrix is formulating for estimation of the variation in the gradient vector at every pixel position. This is statistically derived from standard formulae. Thus obtained variance and covariance matrix forms the base for standardizing the required gradient values for each pixel. Hence producing a statistic, that is further utilized for extracting regions from the analyzed image since gradient magnitudes are high comparatively. Edge pixels are then obtained from this region using various algorithms like non-maxima suppression and Thresholding using hysteresis.

2.2.2 COLOR EDGE DETECTION USING MINIMAL SPANNING TREE[7]

The method is restricted for color RGB images and for the case of a 3 × 3 rectangular sliding window W. Given a set of N =9 vectors corresponding to the pixels inside W, the Euclidean MST (represented by T) is constructed. Considering the edge types we would like to detect, three possible color distributions can be usually found inside W. If no edge is present and the central pixel is located at a uniform color region of the image, the distribution is uni-modal denoting a “plain” pixel type. If there is an

"edge" or "corner" point, a bimodal distribution is expected. If there is a "junction", pixels are expected to form three clusters.

2.2.3 A NEW SCHEME FOR ROBUST GRADIENT VECTOR ESTIMATION IN COLOR IMAGES[8]

Calculating gradient magnitude is the primary focus of gradient estimators and gradient direction is seldom calculated. This new scheme is proposed for doing calculation of both the gradient magnitude and direction accurately and efficiently.

Gradient is calculated by this approach in both the x and y directions. A pre and post filtering approach is used by this filter in both the directions, results in some immunity against noise. But a blurring effect is thus produced, which is tackled to some extent by applying the filters in perpendicular directions for reducing the blurring of edges.

Each window deploys following basic components.

I. High – pass,

II. Low – pass, and

III. Aggregation.

- The high-pass acts as a gradient estimation function.
- Low-pass operator is applied for doing pre-filtering and post-filtering processing
- Aggregating the pre-filtered and post-filtered gradients is done by aggregating operator.

The above mentioned three components can be used following four combinations: MVD-Median-Mean, MVD-Median-Max, RCMG-Median-Mean, and RCMG-Median-Max.

2.3. EVOLUTIONARY ALGORITHMS

Evolutionary computation[9] introduces the concept of evolutionary algorithms. These are a subset of population based and Meta heuristics aware optimization algorithms. Evolutionary algorithms are generally bio inspired. Few of the common concepts used by almost all evolutionary algorithms are reproduction, mutation, recombination and selection.

2.3.1. *PARTICLE SWARM OPTIMIZATION ALGORITHM*[10]

Particle swarm optimization[11, 12] (PSO) is a population-based stochastic approach for solving continuous and discrete optimization problems. In PSO, there are simple software agents, called particles. They move in the search space of an optimization problem. The candidate solution to the problem at hand is represented by position of a particle. By changing its velocity according to the rules inspired by behavior of bird flocking, each particle searches for better positions in the search space. The PSO algorithm starts within the initialization region by generating random positions for the particles. Velocities are usually initialized to zero or to some small random values. This prevents the particles from leaving the search space during the first iterations. During the iterations, the velocities and positions of the particles are iteratively updated until a stopping criterion is met.

The update rules are:

$$v_i^{t+1} = w\, v_i^t + \varphi_1 U_1^t (b_i^t - x_i^t) + \varphi_2 U_2^t (l_i^t - x_i^t) \quad (5)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (6)$$

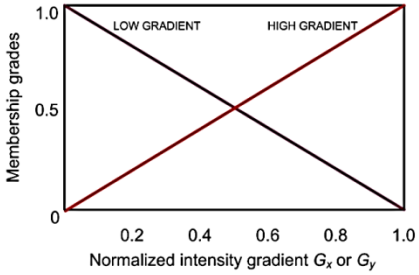
where w is referred to as inertia weight, φ_1 and φ_2 are parameters for acceleration coefficients, U_1^t and U_2^t are two $n * n$ diagonal only matrices which have the entries in main diagonal and contain random numbers only, uniformly distributed in the interval [0,1). Every iteration requires regeneration of these matrices. Vector l_i^t is the neighborhood best value, which is the best found position till now in the neighborhood of particle p_i.

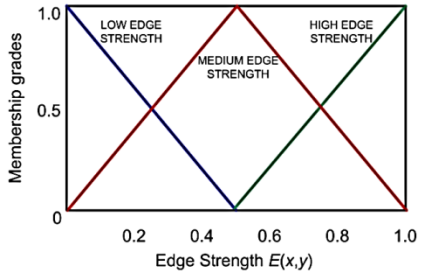
The three terms which are defined in the velocity-update rules are used to characterize the local behaviors that particles have followed. Term defined as inertia or momentum is used as a memory for the previous flight directions. That prevents the particles from changing their directions drastically. The other term is referred to as cognitive component which is used for modeling the tendency of particles to return to previously found best positions. The last term is the social component which helps in quantifying the performance of a particle relative to its neighbors.

EDGE DETECTION VIA FUZZY RULE-BASED EDGE STRENGTH ESTIMATION AND OPTIMAL THRESHOLD SELECTION USING PSO[13]

The following steps will have to be iterated as required:

1) Calculate the x-gradient and y-gradient of the image.
2) Using the given fuzzy rules, set the edge strength for each pixel
    a. Rule 1: If Gx is low and Gy is low then Edge-strength is low.
    b. Rule 2: If Gx is low and Gy is high then Edge-strength is medium.
    c. Rule 3: If Gx is high and Gy is low then Edge-strength is medium.
    d. Rule 4: If Gx is high and Gy is high then Edge-strength is high.





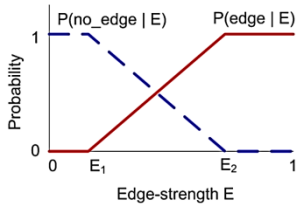
3) Set values for E_1, E_2 and a threshold value $E_{th} = (E_1 + E_2)/2$

4) For each pixel, calculate the probability of the pixel being an edge and not being an edge.

   If $edge\_strength > E_2, Prob(edge) = 1$

   If $edge\_strength < E_1, Prob(edge) = 0$

   Otherwise, calculate using the given graph:



   $Prob(no\_edge) = 1 - Prob(edge)$.

5) Distribute the Swarm particles in the search space.

6) Calculate P_{error} for each particle based on the formula:

$$P_{error} = \sum_{k=E_1}^{E_{th}} \frac{(k-E_1)}{(E_2-E_1)} \frac{n_k}{N} + \sum_{k=E_{th}}^{E_2} \frac{(E_2-k)}{(E_2-E_1)} \frac{n_k}{N} \quad (7)$$

   where n_k is no of pixels having edge-strength k and N is total no of pixels in the image.

7) For each swarm, set the initial $pBest$ values.

8) Among all the $pBest$ values, choose one $gBest$ value i.e. minimum of the $pBest$ values.

9) Modify velocity and position of each particle using Eq. (5) and Eq. (6).

10) Calculate P_{error} for each particle at its new position.

11) If the current P_{error} is smaller than previous one, the particle is shifted to the new location; else it is moved back to the previous one.

12) Update $pBest$ and $gBest$ values

13) Repeat for maximum number of generations.

2.3.2. ANT COLONY OPTIMIZATION ALGORITHM[14]

Ant colony optimization[15, 16] (ACO) is an evolutionary algorithm that can be used for finding approximations to difficult optimization problems. In ACO, a number of ants are used, which are depicted by delegated software agents. These software agents search for good and almost optimal solutions to the optimization problem.

The problem is first deduced into a problem that requires the calculation of best path on a weighted graph. The ants are then made to move on the edges of the graph leaving behind trails of pheromone, following a stochastic process since a model is used that requires a probabilistic distribution of parameters associated with the graph. These parameters are modified at runtime by ants (smallest software entities utilized by the algorithm).

Ants search for food, i.e. solution values by traversing across the edges of the constructed graph. The paths are chosen based on the value of pheromone levels present on the path probabilistically, during traversal an amount of pheromones is deposited on the edges or vertices of the graph. This deposited amount of pheromones is represented as $\Delta\tau$ which is further dependent on the quality of the solution found on the followed trail. This deposited pheromone trail is used by subsequent ants, traversing the same trails for deciding the path with maximum probability of a better solution.

The generalized Ant colony optimization algorithm is initialized by setting required parameters and initializing initial solutions and pheromone trails. And then the following steps are iterated for predefined number of times or allocated CPU frequency.

- Mutual Construction of solutions by all software agents (ants)
- Solution improvement.
- Updating of pheromone levels.

Ant colony optimization Algorithm in Pseudo-code:

1) Initialize Trail
2) Do While (Stopping Criteria Not Satisfied)? Cycle Loop
   i. Do Until (Each Ant Completes a Tour)? Tour Loop
      1. Local Trail Update
   ii. End Do
   iii. Analyze Tours
   iv. Global Trail Update
3) End Do

To solve a problem using ACO:

- The problem has to be represented in the form of sets of components and transitions, or by a set of weighted graphs, on which ants can build solutions.
- The meaning of the pheromone trails is defined.
- The heuristic preference for the ant is defined while constructing a solution.
- If possible, an efficient local search algorithm is implemented for the problem to be solved.
- A specific ACO algorithm is chosen and applied to problem being solved.
- The parameters of the ACO algorithm are tuned.

AN ANT-INSPIRED ALGORITHM FOR DETECTION OF IMAGE EDGE FEATURES[17]

1) Convert image to grey-scale, save as IM
2) Set ρ, λ and T, where ρ is the pheromone evaporation rate, λ denotes the laying of fresh pheromone by ants upon crossing the path from i to j and T denotes the initial pheromone trail.
3) Determine number and initial position of each ant.
4) Select number of iterations ($max\_iterations$)
5) Calculate Global Stimulus S for each pixel of IM:

$$\left||\nabla IM|\right| = \sqrt{\frac{\partial IM^2}{\partial x} + \frac{\partial IM^2}{\partial x}} \tag{8}$$

6) For iteration $i < max\_iterations$ do
   a. For all ants do
      i. For all not yet visited neighbours do
         a) Calculate probability of moving the ant to neighbour using

$$p(i,j) = \frac{\tau(i,j)^{\alpha}\,\eta(i,j)^{\beta}}{\sum_k \tau(i,j)^{\alpha}\,\eta(i,j)^{\beta}} \tag{9}$$

         where τ is pheromone trail and (i,j) is global stimulus of the pixel and α and β are influence coefficients.

      ii. Move the ant to neighbour having maximum probability.

   b. Update the pheromone trail using

$$\tau(i,j) = (1 - \rho)\,\tau(i,j) + \lambda \tag{10}$$

   where ρ is evaporation rate of pheromone and λ is lying of fresh pheromones by ant.

   c. i = i + 1

2.3.3. GRAVITATIONAL SEARCH ALGORITHM[18]

Gravitational Search algorithm is based on principals of gravitational forces.

Gravitational Search algorithm by the sole virtue of the nature of the system is formulated a system of masses or agents that have mass defined by the objective function values. The base entrees are defined as agents which are the fundamentals objects of the whole GSA. A small artificial world is formed by these objectives that obey Newtonian laws of motion and gravitation. By the fundamental laws, every agent attracts each other. These gravitational pulls create a global movement. Fundamentally heavier masses cause more movement in the system and depicts better solution, therefore attracting all agents towards the best solution whereas being heavier they move slower than the lighter masses. This property guarantees the exploitation steps.

In GSA, every object is characterized by four basic properties – Position which is related to the solution in the problem space, inertial mass, active gravitational mass and passive gravitational mass which are all calculated using the basic fitness function .while iterating over the algorithm all masses each other, while heaviest mass attracts lighter masses with much more force as compared to the force exerted by lighter mass on the heavier objects. The heaviest mass therefore will be the one depicting the optimum space in the solution space.

Masses obey the following laws:

- Law of gravity: every particle is attracted by another particle and the amount of forces exerted by the particles on each other, i.e. gravitational pull, is directly proportional to the product of their masses and inversely proportional to the distance between them, R.

- Law of motion: The velocity of a particle at any given time is determined by the sum of the already possessed velocity and the variation in the velocity at that time. Variation in velocity is caused by the acceleration experienced by the gravitational force on the particle.

The GSA algorithm is composed of following steps:

1) Search space identification.
2) Randomize initialization.
3) Evaluate fitness of agents.
4) Update $G(t)$, $Best(t)$, $Worst(t)$, and $M_i(t)$ for $i = 1,2,..N$.
5) Calculate the total force in different directions.
6) Calculate acceleration and velocity.
7) Update agents' position.
8) Repeat steps 3 to 7 until the stop criterion is reached.
9) End

NEWTONIAN GRAVITATIONAL EDGE DETECTION USING GRAVITATIONAL SEARCH ALGORITHM[19]

1) *Identify Search space & Randomize Initialization:*

   Initialize the positions of N agents (masses) by randomly distributing them on an image I with a size –$M1 \times M2$. Initialize the value of the gravitational constant $G(t_0)$.

2) *Calculate the total force in different directions:*

   Force exerted by an agent i at location (x, y) on agent j at location (x_1, y_1) is calculated using –

$$f_{ij}(t) = G(t)\frac{M_i(t)M_j(t)}{R_{ij}(t)+\varepsilon}(x_j - x_i) \quad (11)$$

where $M_i(t)$ and $M_j(t)$ are the grey value of the pixels located at (x, y) and (x_1, y_1). $R_{ij}(t)$ is the Euclidian distance between two agents i and j which is calculated using –

$$\|R_{ij}\| = \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \quad (12)$$

The total force on an agent is equal to the sum of the force exerted by all the agents individually on a single agent. The total force is calculated using –

$$f_i(t) = \sum_{j=1, j\neq i}^{N} rand_i f_{ij}(t) \quad (13)$$

3) *Calculate acceleration and velocity:*

The acceleration of each individual agent is calculated using –

$$a_i(t) = G(t)\sum_{j=1}^{N} rand_i \frac{M_j(t)}{R_{ij}(t)+\varepsilon}(x_j - x_i) \quad (14)$$

where $rand_i$ is a uniform random number within interval $[0,1]$

Using this acceleration, velocity of the individual agent is calculated using –

$$v_i(t+1) = rand_i v_i(t) + a_i(t) \quad (15)$$

4) *Update agent's position:*

Each individual agent's position is updated using –

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (16)$$

5) *Update $M_i(t)$, $G(t)$*

As each individual agent's position is updated .The grey value of the pixel on which it lands on becomes its new mass. The heavy masses – which correspond to good solutions – move more slowly than lighter ones. The $G(t)$ is updated using –

$$G(t) = G(t_0)(^{t_0}/_{t})^{\beta} \quad \beta < 1 \tag{17}$$

6) *Fitness evaluation of agents using the Universal law of gravity*

For each agent we consider a 3*3 neighborhood and the total force exerted on the agent pixel by neighboring pixels is calculated by the vector sum of the forces exerted individually on the agent pixel by the neighboring pixels. The magnitude of the total force is calculated using –

$$\|F\| = \sqrt{(F_x)^2 + \left(F_y\right)^2} \tag{18}$$

$$\theta = \tan^{-1}(F_y/F_x) \tag{19}$$

An appropriate threshold is applied to get the edge map.

7) *Update* K_{best}

Decrease the value of K_{best} (no. of agents) by removing agents that did not lead to edge pixels.

8) *Repeat steps 2 to 7 until* $K_{best} = 1$.

End

CHAPTER – 3

ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony[20, 21] (ABC) is another artificial biological mimicking algorithm based on the meta-heuristics of swarm optimization introduced by Karaboga in 2005 for optimization of numerical valued problems. The bio unit that is mimicked is the foraging behavior of honey bees. The model followed by ABC consists of four essential components: employee, onlooker and scout foraging bees, and food sources. There are three types of bees in a bee colony, which seek for rich food resources which is the fourth essential element, i.e. Food. They try to locate the food source as near as possible to their home. The behavior of a bee colony can be divided into two categories – exploration which involved locating food resource of maximum favorable content, i.e. Food. And the poor food sources are abandoned by bees, (foragers) causing in negative feedback. This is necessary for self-organization of the bee structure and collective intelligence.

A bee colony always hunts for food sources rich in value. Hence for application of the algorithm the optimization problem under consideration are first transformed into a problem for finding the best value of parameters which is used for optimization of the associative function. The bees search a state where a population of solution vectors is constructed and improved with every iteration by using a set of strategies namely:

Locating and moving in the direction of comparatively better solution by searching in the neighborhood and abandoning less valued solution.[22, 23]

The artificial bees are divided into 3 groups of bees:

- Employee bees – related to specific food sources.
- Onlooker bees – observe the dance of employee bees in the hive to choose a food source.
- Scout bees – search for food sources randomly.

The food source location is represented by a potential solution to the problem and the quality of nectar on a food source corresponds to the fitness, i.e. quality of the associated solutions. Number of employee bees is kept equal to the number of expected food sources because every employee bee is designated to only one food source.

The general scheme of the ABC algorithm is as follows:

1) Initialization Phase.
2) REPEAT UNTIL(Cycle=Maximum Cycle Number or a Maximum CPU time)
    a. Employed Bees Phase.
    b. Onlooker Bees Phase.
    c. Scout Bees Phase.
    d. Memorize the best solution achieved so far.

3.1. INITIALIZATION PHASE[24]

All the food resource vectors are initialized as, x_m, (m=1...SN, SN: population size) by scout bees. Parameters are set which would control the movement of all the bees. The vector x_m for each bee holds n number of variables. Every food resource vector x_m will be used as a solution vector for the problem at hand.

The following definition is used for initialization

$$x_{m_i} = l_i + rand(0,1) * (u_i - l_i) \quad (20)$$

where l_i and u_i are the lower and upper bound of the parameter x_{m_i}, respectively.

3.2. EMPLOYED BEES PHASE[24]

Search for new food source is carried by employee bees v_m. The selected food source will be considered if it contains more nectar within the neighborhood of the source. x_m. The explored neighborhood is checked for the fitness of it content and its neighborhood is also explored.

$$v_{m_i} = x_{m_i} + \varphi_{m_i}(x_{m_i} - x_{k_i}) \quad (21)$$

where x_k is a randomly selected food source, i is a randomly chosen index and φ_{m_i} is a random number within the range $[-a, a]$. After producing the new food source v_m , its fitness is calculated.

$$fit_m(x_m) = \begin{cases} 1/(1 + f_m(x_m)), & if\ f_m(x_m) \geq 0 \\ 1 + abs(f_m(x_m)), & if\ f_m(x_m) < 0 \end{cases} \quad (22)$$

where $f_m(x_m)$ is the objective function value of solution x_m.

3.3. ONLOOKER BEES PHASE[24]

Onlooker bees keep waiting in the hive. The food source information which was previously analyzed by employee bees is provided to the onlooker bees for deciding foraging to be done on the source. The information provided to the onlooker bees is then used by onlooker bees to probabilistically choose the food source. This probability based selection is done on the basis of the fitness values of the food

sources. The most commonly used selection technique is Roulette wheel selection method.

The probability value pm with which x_m is chosen by an onlooker bee can be calculated

$$prob_m = \frac{f_m(x_m)}{\sum_{i=1}^{N} f_i(x_i)} \quad (23)$$

After an onlooker bee chooses its favorable food source x_m, the neighborhood is then explored by the bees. Hence, due to the probabilistically choosing the food source, the wealthier food sources get the more onlookers which results in the appearance of positive feedback behavior.

3.4. SCOUT BEES PHASE[24]

The employee bee locations where the food source has been exploited completely and the solutions cannot be improved after a previously mentioned number of trials are abandoned. The same number of scout bees is then brought into the scene. These bees choose the food sources randomly using the Eq. (20). The abandoned resources i.e. the resources that have been completely exploited provide negative feedback that balances out the positive feedback.

CHAPTER – 4

PROPOSED ALGORITHM

4.1. APPROACH USED

The proposed approach uses a well-defined gradient of the image to serve as the food source. This gradient G is calculated from the x-gradient G_x and the y-gradient G_y using Eq. (1). The G_x and G_y are calculated by convolving the masks M_x and M_y over the image I –

$$M_x = [-1 \quad 1] \text{ and } M_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

The total number of employee bees is set directly proportional to $\sqrt{M * N}$ so as to minimize the false negative errors. The employee bees are then distributed throughout the search space. The total number of onlooker bees is twice the total number of employee bees. Basically it should be directly proportional to the number of employee bees so that they can be probabilistically distributed. The purpose of doing it is to minimize the number of abandoned sources and maximize the chances of finding edges in their neighborhood. The total number of scout bees is initially set to 0.

The meta-heuristic H is initialized to a zero matrix of size $M * N$. It is used to keep track of the best food source found till now. Whenever a bee having maximum fitness value is found, H at its location will be incremented. At the end of every iteration, H is updated.

The total number of iterations is set. The number of iterations would determine the quality of the edge-map.

The objective function is defined for each bee as –

$$fit_n = H_{(i,j)} + {}^{1}/_{G_{(i,j)}} \tag{24}$$

where (i, j) is the location of n^{th} bee.

The fitness function is defined as –

$$fitness_n = \begin{cases} 1 + abs(fit_n), & fit_n < 0 \\ {}^{1}/_{(1 + fit_n)}, & fit_n \geq 0 \end{cases} \tag{25}$$

The fitness function value for each bee is to be maximized. For all the employee bees, these both values are calculated.

Employee bees then share their food source information – fitness function values with all the onlooker bees. The onlooker bees then use the information provided by employee bees to probabilistically choose their food sources. A fitness based selection technique - the roulette wheel selection method is used. Once the onlooker bees select the food source, they then choose a neighbor in their $3 * 3$ window which has the maximum fitness function value. After choosing the best neighbor, the onlooker bee is moved to its position. The food source i.e. G at the previous position is decreased.

$$G_{(i,j)_{new}} = G_{(i,j)_{old}}(1 - {}^{1}/_{n}) \tag{26}$$

where n is the current number of bees at (i, j).

Then the employee bees where no onlooker bees have been located are marked as abandoned and their food source is depleted. Such abandoned sources are counted and the same number of scout bees are then located to a random location.

After the previously defined number of iterations, the resultant heuristic matrix H is normalized. An adaptive thresholding is then applied on the normalized matrix to determine the resultant edge-map.

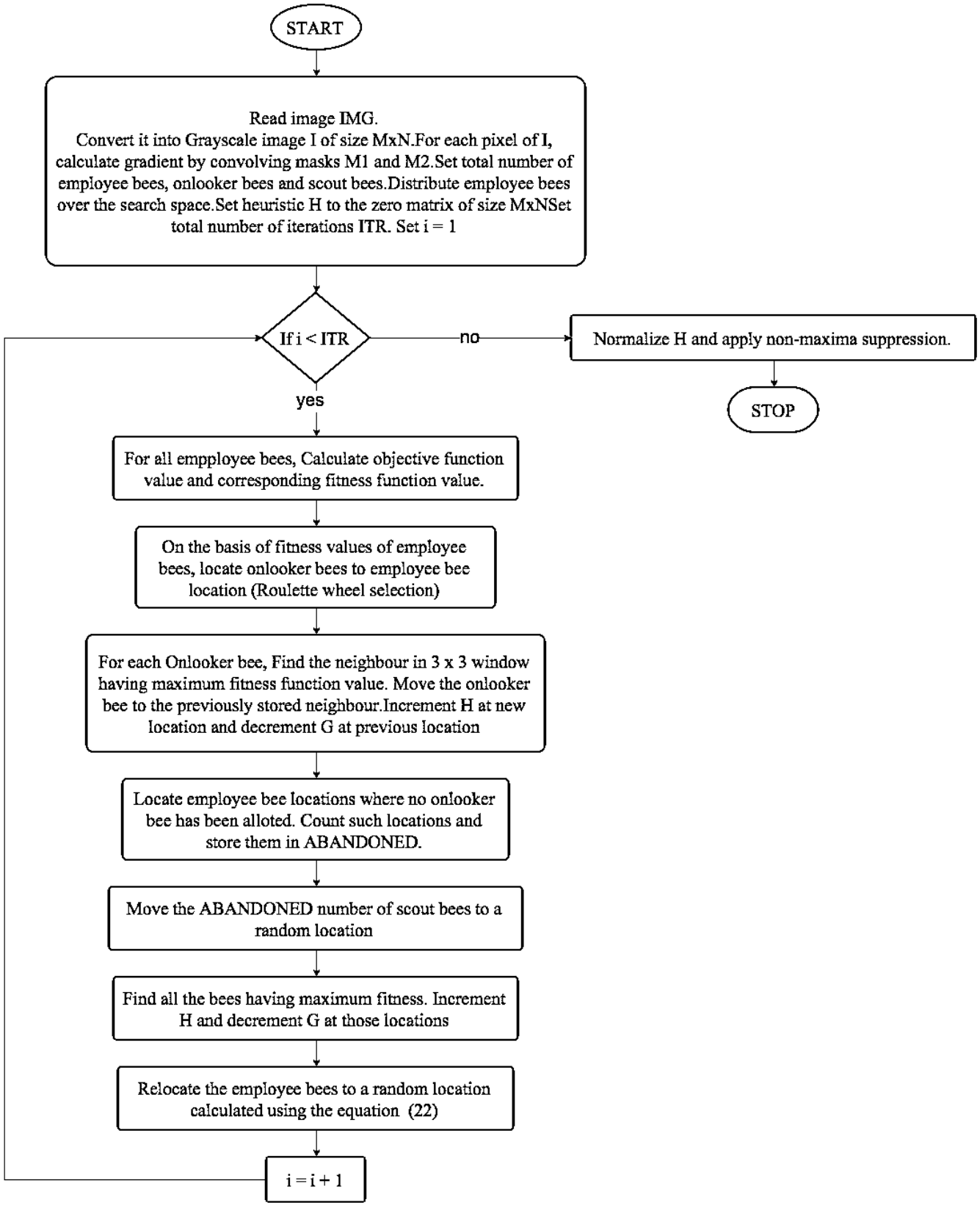


Figure 1. Flowchart of Proposed Approach

4.2. PSEUDOCODE

STEP – 1: INITIALIZATION

1) Read image Img and convert it to gray scale image I of size M * N.
2) Calculate the gradient values at each pixel using masks M_x and M_y and Eq. (1).
3) The total number of employee bees will be directly proportional to $\sqrt{M * N}$
4) The employee bees are then distributed throughout the search space uniformly or randomly. In case of uniform distribution, every 3 * 3 or 5 * 5 or 7 * 7 window will have one employee bee. In case of random distribution, note that no two employee bees can be located at the same location.
5) The total number of onlooker bees will be twice the total number of employee bees.
6) The total number of scout bees will be initialized to 0.
7) The meta-heuristic H is zero matrix of size M * N and it keeps the track of best food source. The values in this matrix will automatically update in every iteration whenever the best food source will be located by the bees.

STEP – 2: ITERATIONS

Repeat this step for the given number of iterations.

1) For all employee bees, do the following:
   a. Calculate –

$$fit\_emp_n = H_{(i,j)} + {}^{1}/_{G_{(i,j)}} \quad (27)$$

where fit_emp_n is the objective function value of n^{th} employee bee, (i, j) is the current location of the employee bee under consideration.

b. Now calculate the fitness value using –

$$fitness\_emp_n = \begin{cases} 1 + abs(fit\_emp_n), & fit\_emp_n < 0 \\ 1/(1 + fit\_emp_n), & fit\_emp_n \geq 0 \end{cases} \quad (28)$$

2) On the basis of fitness values for employee bees, move the onlooker bees to their locations. The number of onlooker bees at the position of an employee bee will be directly proportional to the fitness value of the employee bee.

3) For all onlooker bees, do the following:

a. Calculate the objective function value.

$$fit\_on_n = H_{(i,j)} + 1/G_{(i,j)} \quad (29)$$

where (i, j) is the current location of the onlooker bee under consideration

b. Calculate the fitness value.

$$fitness\_on_n = \begin{cases} 1 + abs(fit\_on_n), & fit\_on_n < 0 \\ 1/(1 + fit\_on_n), & fit\_on_n \geq 0 \end{cases} \quad (30)$$

c. For each neighbor in the 3 * 3 window, do the following:

i. Calculate the objective function value.

ii. Calculate the fitness value.

d. Move the onlooker bee to the neighbor having maximum fitness value. If the chosen best neighboring location already has the maximum number of onlooker bees that are allowed at each position, move the onlooker bee to the neighbor having next best neighbor.

e. Decrement G at previous position using Eq. (26).

4) Locate the employee bees where no onlooker bees have been allotted. Mark these locations as abandoned resources. Count the number of abandoned resources. Set the number of scout bees equal to the number of abandoned resources.

5) Place all scout bees in the search space using Eq. (20).

6) For all scout bees, do the following:

   a. Calculate the objective function value.

$$fit\_sc_n = H_{(i,j)} + {}^{1}/_{G_{(i,j)}} \tag{31}$$

   where (i, j) is the current location of the scout bee under consideration.

   b. Calculate the fitness value.

$$fitness\_sc_n = \begin{cases} 1 + abs(fit\_sc_n), & fit\_sc_n < 0 \\ {}^{1}/_{(1 + fit\_sc_n)}, & fit\_sc_n \geq 0 \end{cases} \tag{32}$$

7) Locate the positions of employee bees having maximum fitness. Increment the heuristic H at those positions by 1.

8) Locate the positions of onlooker bees having maximum fitness. Increment the heuristic H at those positions by 1.

9) Locate the positions of scout bees having maximum fitness. Increment the heuristic H at those positions by 1.

10) For all employee bees, do the following:

   c. Move the employee bee to any random position using Eq. (22).

STEP – 3: EDGE-MAP

1) Normalize the finally obtained heuristic matrix H.

2) Apply non-maxima suppression on normalized H using adaptive Thresholding to obtain the edge-map.

CHAPTER – 5

RESULTS AND COMPARISON

The Intel® Core™ i7-2630QM CPU @ 2.00 GHz and MATLAB is used to implement the proposed approach. More than 100 images of different sizes and formats have been used as test images to analyze the performance of the proposed approach. Some of these images are displayed here.

The working of the proposed approach is evaluated using the objective measures such as Shannon's entropy[25, 26], Cohen's Kappa[27] and Pratt's Figure of Merit[28]. The proposed approach is evaluated using different test images shown in Figure 2.

The proposed approach has been used on 8 different images of different sizes to extract the edges. All the images used are RGB images that are firstly converted into gray-scale images. The results are then compared to the most frequently used edge-detectors – Canny[4], Laplacian of Gaussian[5], Prewitt[3], Roberts[2] and Sobel[1]. The test images and their details are given in Figure 2. The result of application of the proposed edge detection technique is given in Figure 3.

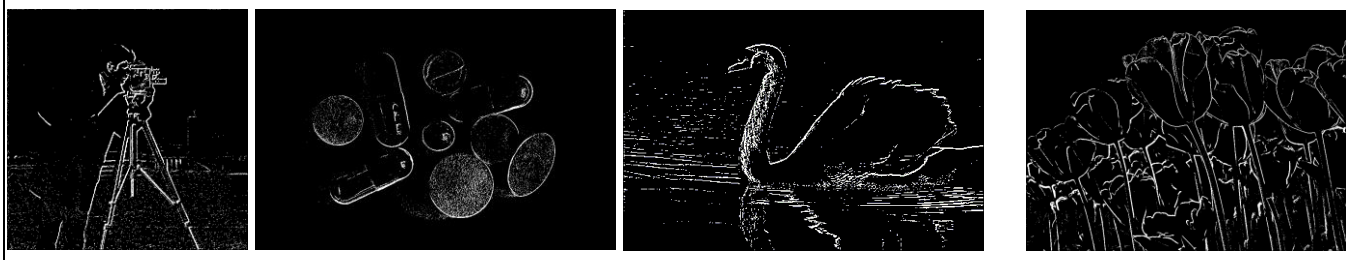
Figure 4 shows the result of application of the 5 mentioned commonly used edge-detectors and the application of the proposed approach to detect edges. The visual results show that the proposed approach provides distinct edges as opposed to results of Canny and LoG edge detectors and it provides much finer details as opposed to Prewitt, Roberts and Sobel edge detectors.

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

(e) (f) (g) (h)

Figure 2. Test images (a) 500 x 500 Lena image (b) 481 x 321 Butterfly image (c) 481 x 321 Deer image (d) 481 x 321 Car image (e) 512 x 512 Cameraman image (f) 850 x 565 Pills image (g) 481 x 321 Swan image (h) 1024 x 768 Tulips image

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



(e) (f) (g) (h)

Figure 3. The resultant edge maps for the test images using the proposed approach. The maximum number of iterations for each is set as 100.

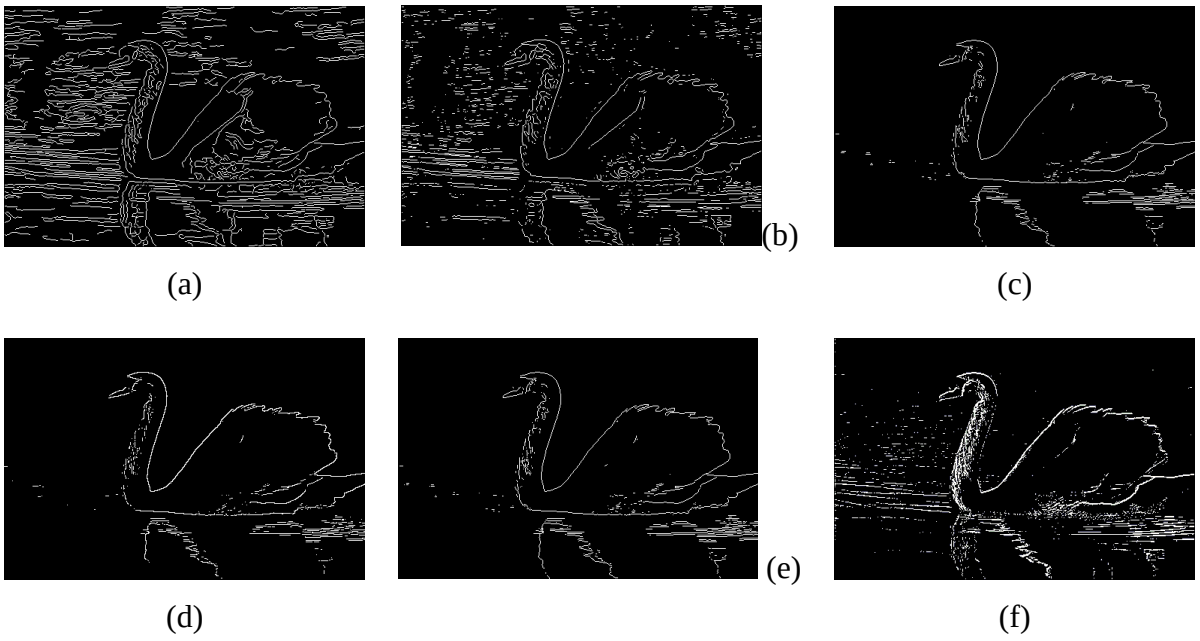


Figure 4. The results of different edge detectors on Swan image (a) Canny (b) Laplacian of Gaussian (c) Prewitt (d) Roberts (e) Sobel (f) Proposed approach.

5.1. SHANNON'S ENTROPY

In scientific theory, entropy (more specifically, Claude Shannon entropy) is that the mean value (average) of the data contained in every message received. Here, message stands for a happening, sample or character drawn from a distribution or knowledge stream. The index of the likelihood distribution is beneficial as a live of entropy as a result of its additive for freelance sources. Entropy is zero once one outcome is definite. Claude Shannon entropy quantifies of these concerns precisely once a likelihood distribution of the supply is understood. Generally, entropy refers to disorder or uncertainty. Claude Shannon entropy provides associate absolute limit on the most effective attainable average length of lossless coding or compression of any communication. Shannon's Entropy is given by –

$$H(X) = -\sum_{i=0}^{N-1} p_i \log_2 p_i \quad (33)$$

Shannon's Entropy values for the results of different edge detectors applied on all test images are shown in Table 1.

Table 1. Shannon's entropy values corresponding to different edge detectors for all test images

Test images	Canny	LoG	Prewitt	Roberts	Sobel	Proposed approach
Lena	0.43054	0.32298	0.19293	0.17136	0.20436	0.63537
Butterfly	0.33488	0.30554	0.18126	0.19094	0.18129	0.48311
Cameraman	0.39523	0.31579	0.16964	0.16266	0.16711	0.24889
Car	0.47845	0.32096	0.19953	0.18276	0.20104	0.70357
Deer	0.46951	0.39537	0.20820	0.18548	0.20918	0.56840
Pills	0.26170	0.22072	0.08726	0.10346	0.09001	0.16956
Swan	0.46910	0.32505	0.15391	0.14876	0.15402	0.39718
Tulips	0.32477	0.26561	0.16736	0.16456	0.16800	0.33334

5.2. COHEN'S KAPPA

Cohen's Kappa[27] is an indicator that measures the conformity between two raters for categorical (qualitative) items. The two raters examine the same set of data. The items indicate the degree to which these two raters agree while they are categorizing the data. Two raters each classify N items into C mutually exclusive categories. Cohen's kappa computes the conformity between two raters. The equation for κ is –

$$\kappa = \frac{\Pr(a)-\Pr(e)}{1-\Pr(e)} \tag{34}$$

Where $Pr(a)$ is observed percentage of conformity and $Pr(e)$ is expected percentage of conformity.

After adjusting for the proportion of conformities that take place unintended κ is the proportion of conformities that is really detected between raters. The value of κ ranges between –1 and +1. +1 value of kappa implies the ratings given by the two rates are in perfect agreement with each other. On the other hand –1 implies perfect disagreement. 0 value of κ implies that the ratings given by the two raters are not at all related in any possible way. 0.7 value of κ is considered to be satisfactory.

The comparison of results using Cohen's Kappa is shown in Table 2.

Table 2. Comparison of results of proposed approach with the basic edge detectors using Cohen's Kappa

Test images	Canny	LoG	Prewitt	Roberts	Sobel
Lena	0.83998	0.87825	0.90738	0.90863	0.90124
Butterfly	0.88766	0.89157	0.91657	0.91324	0.91681
Cameraman	0.89957	0.92242	0.95625	0.95271	0.95693
Car	0.78589	0.83351	0.86194	0.86215	0.86248
Deer	0.82161	0.84509	0.89085	0.89024	0.89119
Pills	0.94337	0.95249	0.97475	0.97057	0.97487
Swan	0.84594	0.89227	0.93003	0.92753	0.93008
Tulips	0.91542	0.92909	0.94415	0.93995	0.94427

5.3. PRATT'S FIGURE OF MERIT

Pratt introduced a figure of merit to examine and stabilize the related errors in edge detection process. It is calculated using –

$$PFoM = \frac{1}{\max(I_I, I_D)} \sum_{i=1}^{I_D} \frac{1}{1+d_i \alpha^2} \quad (35)$$

where I_I is number of edge-map points in an ideal edge-map, I_D is number of edge-map points in the detected edge-map, α is constant that would penalize displaced edges and d_i is the distance by which actual edge point is separated from ideal edge point.

The comparison of results using Pratt's FoM is shown in Table 3.

Table 3. Comparison of results of proposed approach with the basic edge detectors using Pratt's Figure of Merit

Test images	Canny	LoG	Prewitt	Roberts	Sobel
Lena	0.76306	0.62183	0.47549	0.48166	0.52564
Butterfly	0.58155	0.54975	0.42856	0.45688	0.42557
Cameraman	0.60581	0.65151	0.52108	0.56268	0.51330
Car	0.71211	0.57153	0.48260	0.49438	0.47784
Deer	0.70870	0.67259	0.48634	0.51648	0.48191
Pills	0.61693	0.65335	0.49901	0.57958	0.49653
Swan	0.63752	0.75035	0.49690	0.54485	0.49584
Tulips	0.66318	0.56264	0.45018	0.48324	0.44920

CHAPTER - 6

CONCLUSION

In the presented work, edges are detected from digital images. The proposed algorithm uses the concept of Artificial Bee Colony. It forges the behavior of the bee colony. The bees are divided into 3 sections – employee bees, onlooker bees and scout bees. They travel throughout the search space to find the food sources. The best food sources are then located and are stored. These best locations determine if the edge is present at that location based on the gradient matrix. This is done using a heuristic that keeps on increasing at a location whenever that location is found to be the best food source. The final edge map is determined by normalizing the heuristic and applying non-maxima suppression using adaptive thresholding. The results of the application of the proposed approach are then compared to the most commonly used edge detectors. Visually, the results of the proposed approach seem to be better than the other edge detectors because the false positives and false negatives are much less. The same can be concluded from the analysis of the results using 3 different techniques – Shannon's Entropy, Cohen's Kappa and Pratt's Figure of Merit.

CHAPTER - 7

REFERENCES

1. Sobel, I., Feldman, G., , A 3x3 Isotropic Gradient Operator for Image Processing. Stanford Artificial Intelligence Project, 1968.
2. Davis, L.S., A survey of edge detection techniques. Computer Graphics and Image Processing, 1975. 4(3): p. 248-270.
3. Prewitt, J.M.S., Object Enhancement and Extraction. Picture processing and Psychopictorics, Academic Press, 1970.
4. Canny, J., A Computational Approach To Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 1986: p. 679–698.
5. Sotak Jr, G.E. and K.L. Boyer, The laplacian-of-gaussian kernel: A formal analysis and design procedure for fast, accurate convolution and full-frame output. Computer Vision, Graphics, and Image Processing, 1989. 48(2): p. 147-189.
6. Rakesh, R.R., P. Chaudhuri, and C. Murthy, Thresholding in edge detection: a statistical approach. Image Processing, IEEE Transactions on, 2004. 13(7): p. 927-936.
7. Theoharatos, C., G. Economou, and S. Fotopoulos, Color edge detection using the minimal spanning tree. Pattern Recognition, 2005. 38(4): p. 603-606.
8. Nezhadarya, E. and R.K. Ward, A new scheme for robust gradient vector estimation in color images. Image Processing, IEEE Transactions on, 2011. 20(8): p. 2211-2220.

9. Fogel, D.B., Evolutionary computation: toward a new philosophy of machine intelligence. Vol. 1. 2006: John Wiley & Sons.
10. Marco Dorigo, M.A.M.d.O., Andries Engelbrecht, Particle Swarm Optimization. http://scholarpedia.org/article/Particle_swarm_optimization.
11. Eberhart, R.C. and J. Kennedy. A new optimizer using particle swarm theory. in Proceedings of the sixth international symposium on micro machine and human science. 1995. New York, NY.
12. Kennedy, J., Particle swarm optimization, in Encyclopedia of Machine Learning. 2010, Springer. p. 760-766.
13. Khunteta, A. and D. Ghosh. Edge detection via fuzzy rule-based edge strength estimation and optimal threshold selection using PSO. in Industrial and Information Systems (ICIIS), 2013 8th IEEE International Conference on. 2013. IEEE.
14. Dorigo, M., Ant Colony Optimization. http://www.scholarpedia.org/article/Ant_colony_optimization.
15. Dorigo, M. and L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem. Evolutionary Computation, IEEE Transactions on, 1997. 1(1): p. 53-66.
16. Dorigo, M., M. Birattari, and T. Stützle, Ant colony optimization. Computational Intelligence Magazine, IEEE, 2006. 1(4): p. 28-39.
17. Etemad, S.A. and T. White, An ant-inspired algorithm for detection of image edge features. Applied Soft Computing, 2011. 11(8): p. 4883-4893.
18. Rashedi, E., H. Nezamabadi-pour, and S. Saryazdi, GSA: A Gravitational Search Algorithm. Information Sciences, 2009. 179(13): p. 2232-2248.

19. Verma, O.P. and R. Sharma. Newtonian Gravitational Edge Detection Using Gravitational Search Algorithm. in Communication Systems and Network Technologies (CSNT), 2012 International Conference on. 2012. IEEE.

20. Karaboga, D. and B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 2007. 39(3): p. 459-471.

21. Karaboga, D. and B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, in Foundations of Fuzzy Logic and Soft Computing. 2007, Springer. p. 789-798.

22. Karaboga, D. and B. Akay, A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation, 2009. 214(1): p. 108-132.

23. Karaboga, D. and B. Basturk, On the performance of artificial bee colony (ABC) algorithm. Applied soft computing, 2008. 8(1): p. 687-697.

24. Karaboga, D., Artificial Bee Colony Algorithm. http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm.

25. Daróczy, Z., Generalized information functions. Information and control, 1970. 16(1): p. 36-51.

26. Sharma, B.D. and I.J. Taneja, Entropy of type (α, β) and other generalized measures in information theory. Metrika, 1975. 22(1): p. 205-215.

27. Berry, K.J. and P.W. Mielke, A generalization of Cohen's kappa agreement measure to interval measurement and multiple raters. Educational and Psychological Measurement, 1988. 48(4): p. 921-933.

28. Abdou, I.E. and W. Pratt, Quantitative design and evaluation of enhancement/thresholding edge detectors. Proceedings of the IEEE, 1979. 67(5): p. 753-763.