Chapter 1

INTRODUCTION

Boundary Detection in real images poses a fundamental problem in various computer vision processes. The process of boundary detection is distinguished from that of Edge detection in the sense that Boundary represents the contour in a given image's plane that reflects the change and variation in image pixel's ownership from one object to another one. Whereas, edge refers to the set of pixels representing abrupt changes in an image's low level features like color and brightness. Image boundary detection contributes significantly for higher order problems like object recognition, scene understanding and component analysis.

Major issue concerning image boundary detection in images is that not even human beings can agree upon a single unambiguous boundary for a given sample input image [1]. Different people find different components of the image to be a part of the boundary pixels thus resulting in lack of an unambiguous unique ground truth. This is turn makes the process, of boundary detection and image segmentation algorithm's comparison, complicated. However, for objective comparison of boundary detection algorithms, some frameworks have been set up. For example, the Berkeley Dataset (BSDS). It provides a large database of natural images along with different human perspective of each image's ground truth and are provided with Precision –Recall based measurement for comparison [2]. Figure 1 shows four example images along with their respective ground truth images as suggested by different people.

The solutions for solving the problem of image boundary detection and then segmentation come from various imminent fields like graph theory [3, 4], clustering [5], image feature based methods [6, 7], optimization of an evolution function [8] as well as region growing [9]

Contour or boundary detection is generally performed as a pre-step for performing image segmentation because boundaries correspond to the borders between multiple objects. It defines the process of detecting and locating sharp discontinuities in image regions. This, in turn, closely relates to the process of edge detection. If only edge detection is considered, then it is not enough for detecting boundaries as it registers any kind of abrupt changes in brightness as well as misses smooth transitions of brightness of the image on the other hand. This causes false positives like textured regions, or one sharply highlighted spot and true negatives like region contours of low contrast adjustment.

One of the ways to overcome these shortcomings is to combine edge detection techniques with soft computing techniques such as Fuzzy Logic, or Genetic Algorithms etc [10, 11]. Other procedures involve oriented local image cues, like color, texture and brightness descriptors with edge detection to get boundary estimation. Such combination procedures include energy and probability weighting [12], likelihood methods [13] , or multi-scale cue combination [14].

In this paper, we propose two algorithms that utilizes the concept of swarm Intelligence techniques namely the Ant Colony Optimzation (ACO) and the Firefly algorithm respectively in order to detect boundary in images. Although a variety of Swarm Intelligence inspired algorithms already exist , their swarms usually work to find one global optimal solution in the given search space since these algorithms come from the field of optimization.

The two proposed algorithms find non- predefined number of contours with arbitrary sizes, shape and positions in an image which together or independently give the solution to the problem.

Recently, there has been an increase in the number of boundary detection approaches being developed on the basis of Genetic Algorithms, Evolutionary Computing and also Swarm Intelligence to be applied. These have, mostly, been used to optimize and improve the performance of crucial steps of boundary detection and hence segmentation problem. Genetic and Evolutionary algorithms have been used to enhance the process of clustering in image segmentation in [15, 16]. In [17], authors have combined the low level features of an image with the high level features as produced by a visual attention model to the region growing algorithm, in which the optimal cut offs for region growing task were detected via the Particle Swarm Optimization (PSO) algorithm. In [18], gradient images have been high pass filtered in the frequency domain to filter out the texture regions leaving behind the boundary images.

PSO has been applied to locate the optimal fuzzy entropy cut offs to segment images into background and foreground in

[19]. A similar approach has been introduced wherein, PSO has been used to tune the cut offs in 2D-histograms, thus maximizing entropy in order to obtain segmentation in infrared images in [20]. In [21], Swarm Intelligence has been used to create contour models using different agents that interact with each other, and the boundary is obtained by using various fitness functions and agent contour models.

Swarm Intelligence has also been used to improve Image segmentation based on clustering approaches that can get stuck in local optima, on the basis on initialization.

In [22], a modified PSO algorithm is used to find a proper edge detection filter size for noisy images. Although combining edge filters of various sizes can yield better edge detection results, it is usually expensive when seen from the computational point of view, when applied to the entire image. Hence, the heuristic abilities of ACO and firefly algorithms were respectively utilized to effectively and adaptively decide, where to use the filters on the image of various sizes. The evaluation and comprehension of edge intensity have analogy to a certain extent.

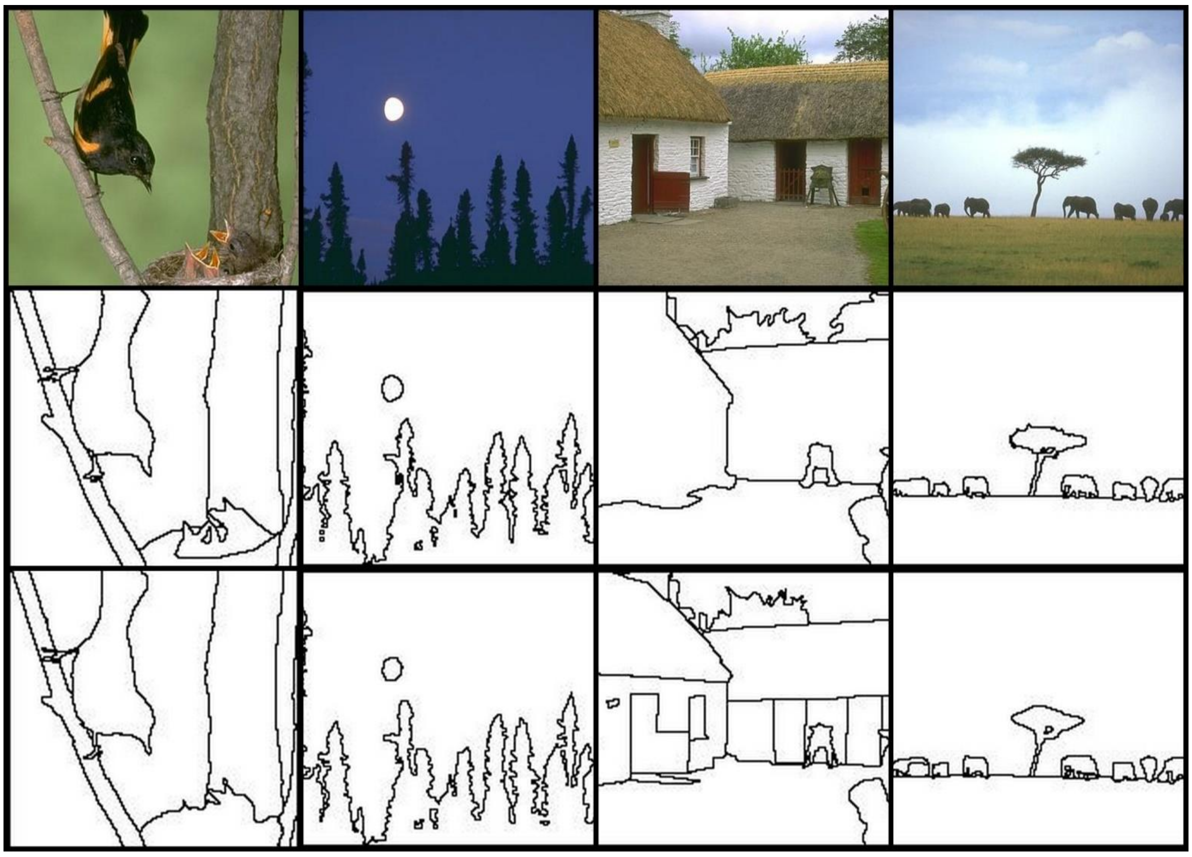


Fig. 1. Four sample images from Berkeley Segmentation Dataset (BSDS 500) [3], each with corresponding three different human marked ground truth images

Chapter 2

LITERATURE REVIEW

2.1. Different Image Boundary Detection Techniques

Numerous image Boundary detection methods have been developed in the past. Some of them are described below:

2.1.1. Boundary Detection using morphological operation: Erosion

Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations use a structuring element on the input image under consideration, producing the corresponding output image of the same size. In a morphological operation, output image's every pixel's value is based on a comparison of the corresponding pixel in the input image with its neighbors. When performing the morphological operation of Erosion, the output pixel's value is equal to the minimum value of all the pixels present in the corresponding input pixel's neighbourhood. In a binary image, if any of the pixels' value is equal to 0, then its corresponding output pixel is set to 0.

In order to detect boundaries in an image, the image is first converted into its corresponding black and white image. This black and white image is now eroded and the result is then subtracted from the original black and white image. The resultant image is of the same size as the input image and contains the boundary pixels highlighted.

. The steps for boundary detection can be summed up as follows:

1. Convert the image into corresponding binary image

2. Perform Erosion on this binary image, A using a structuring element B

   $(A\theta B)$

3. Subtract the result of step 2 from the binary image obtained in step 1.

   $$\beta(A) = A - (A\theta B) \tag{1}$$

   The resultant is the boundary image of the input sample image.

2.1.2. Boundary Detection in images by frequency domain filtering of gradient image [18]

This technique focuses on improving the edge detection mechanism in order to extract boundaries from the image. Frequency domain is achieved by obtaining fourier transform on the image. The basic property used here is that convolution in space domain is equivalent to direct multiplication in the frequency domain. This is really a question that is more for your class instructor. The general idea is that the image (f(x,y) of size M x N) will be represented in the frequency domain (F(u,v)). The equation for the two-dimensional discrete Fourier transform (DFT) is:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)} \tag{2}$$

The concept behind the Fourier transform is that any waveform that can be constructed using a sum of sine and cosine waves of different frequencies. The exponential in the above formula can be expanded into sines and cosines with the variables u and v determining these frequencies.

The inverse of the above discrete Fourier transform is given by the following equation:

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)} \qquad (3)$$

Thus, if we have F(u,v), we can obtain the corresponding image (f(x,y)) using the inverse.

The process of boundary detection using the morphological operation of erosion is described below:

1. Read the input image and find the gradient value at all pixel positions in the image
2. Now , perform the FFT on the gradient image produced after step 1 as follows

   $$F(u,v) = \tau[M_\sigma(x,y)] = A(u,v)\exp(j\emptyset(u,v)) \qquad (4)$$
3. The frequency transform now contains an amplitude spectrum component as well as a phase spectrum component.τ
4. The amplitude component is now multiplied with a high pass filter – the Gaussian high pass filter.

   $$A_f(u,v) = A(u,v)\,.H(u,v) \qquad (5)$$

   Where A(u,v) is the amplitude spectrum of the frequency domain gradient image and H(u,v) is the high pass filter. H(u,v) is defined as follows :

   $$H(u,v) = \exp\left[\frac{-D^2(u,v)}{2D_0^2}\right] \qquad (6)$$

   And

   $$D(u,v) = \left[\left(\frac{u-N}{2}\right)^2 + \left(\frac{v-N}{2}\right)^2\right]^{\frac{1}{2}} \qquad (7)$$

   Where D_0 is the cut off frequency at a distance from the origin and is evaluated experimentally by observation.

5. The spatial domain image is reconstructed using the inverse frequency transform as given below :

$$M_{\sigma}^{f}(x,y) = \left|\tau^{-1}\left[A_f(u,v)\exp(j\emptyset(u,v))\right]\right|^2 \quad (8)$$

Where τ^{-1} [] represents the Inverse Fourier transform and the square computation done in the aforementioned equation is for enhancing the contrast of contour over the edges that cover the textures.

2.2.Ant Colony Optimization

One of the most widely used examples for implementation of the SI principles are the Ant colony Optimization (ACO) [28] The communication of the ants in ACO is indirect as well as location based, i.e., the ant leaves a pheromone trail at every location it moves to which defines the position measure with respect to the

Ant Colony Optimization (ACO) was proposed by Dorigo et al. [ ]. The major application of ACO is in obtaining a graph's optimal paths. ACO is based on the definitive property of ants by which they leave pheromone trails during their search for food. The other ants follow these pheromone trails. The probability by which an ant k moves from position i to position j is determined by :

$$p_{i,j} = \frac{\left(\tau_{i,j}^{\alpha}\right)\left(\eta_{i,j}^{\beta}\right)}{\sum_k\left(\tau_{i,j}^{\alpha}\right)\left(\eta_{i,j}^{\beta}\right)}$$

Where $\tau_{i,j}$ is the amount of pheromone trail between position i and position j , whereas , $\eta_{i,j}$ is the heuristic function value between position i and position j . The heuristic function $\eta_{i,j}$ determines the desirability of the path from position I to position j. The parameters α and β are the influence coefficients of the pheromone amount and the heuristic function respectively. At every step an ant moves, the pheromone value of the path taken from position I to position j is updated as

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \lambda$$

Where ρ is the pheromone evaporation rate i.e., it determines the evaporation of pheromone and, $\tau_{i,j}$ denotes the laying of fresh pheromone by ants upon crossing the path from i to j. The value of λ is dependent on how successful an ant is at solving the problem in consideration

2.2.1 Algorithm

1. Position one ant at each pixel position in the input image
2. Save the positions of ants and number each ant uniquely as 1,2..*nAnt*
3. Initialize the pheromone values as 1 for each pixel position since each neighbor is as likely as the next initially
4. Heuristic value, η , the food for the ants is initialized as 0 for each ant position throughout the image
5. Calculate and evaluate the value of each food location as follows :

$$\eta\left(x_p, y_p\right) = \frac{1}{9}\sum_{i=-1}^{i=1}\sum_{j=-1}^{j=1} B(x_p + i,\ y_p + j) \qquad (3)$$

where *B(x,y)* being the boundary value at point (x,y)

which is calculated as given below

$$B(x,y) = \sqrt{B_x(x,y)2 + B_y(x,y)2} \qquad (4)$$

Where $B_x(x, y)$ and $B_y(x, y)$ determine the intensity difference between the neighbors of the particles horizontally as well as vertically respectively. The masks that can be used for calculating the horizontal as well as the vertical difference can be

1	1	1
0	0	0
-1	-1	-1

And

1	0	-1
1	0	-1
1	0	-1

respectively.

These masks are hovered over the entire image and the result is stored and saved.

6. For *nItr* number of iterations , repeat step 7
7. For each ant,

- set *tab* for current ant at current position as 1 i.e., saving the path of the current ant so far

  $$tab(locx, locy, ant_{id}) = 1 \qquad (5)$$

  Where (locx, locy) define the current position of ant with unique number id.

- In the 5x5 neighborhood , calculate the maximum probability to move to position *pnew* as

  $$p_{pnew} = \frac{(\tau_{pnew}^{\alpha})(\eta_{pnew}^{\beta})}{\sum_k (\tau_{pnew}^{\alpha})(\eta_{pnew}^{\beta})} \qquad (6)$$

Where the parameters hold their usual values as described in eq(). The modification made is that instead of calculating the probability of taking a pnew in the neighborhood path from current ant position to a position p, we calculate the probability of the new position to be acquired (*pnew*).

- If the ant is relocated to a new position, the pheromone values are re-evaluated as

$$\tau_{pnew} = (1 - \rho)\tau_{pnew} + \lambda \qquad (7)$$

Where parameters hold the usual meaning as mentioned in eq() with the modification that τ_{pnew} is the heuristic function value or the food value at pixel position now acquired by the ant under consideration i.e., *pnew*.

8. Pheromone matrix now obtained contains the heuristic value or the boundary value possessed by each ant
9. Threshold is applied as follows :

$$Threshold(m) = \begin{cases} Displayed, \ f(m) \geq \text{thres} \\ Removed, \ f(m) < thres \end{cases} (8)$$

Where thres is the predefined value for threshold.

2.3. Performance Metrics

2.3.1. Precision , Recall and f- measure

As described in Section 1 , the major problem with comparison of boundary detection algorithms is the lack of uniqueness of ground truth. One of the most widely used dataset for image boundary detection and image segmentation algorithms is the Berkeley segmentation dataset(BSDS) [1], which provides 500 natural images of size 481x321, each along with different human generated segmentations, that pose as the ground truth images.

Generally, the metrics used for segmentation evaluation are for example, the Jaccard Index, the probabilistic rand Index[31], or the Object level consistency error OCE[32]. Such metrics are generally dependent on the amount of overlap between the resultant boundaries of the algorithm under consideration and the ground truth images. They differ in measure attributes such as distance measures, the punishment for over or under segmentation, or the manner in which overlap is weighed w.r.t sizes. But these require that the closed boundary regions are fully separated from one another so that a pixel can be allocated to one region only. Hence, these metrics cannot be applied to the boundary detection algorithm evaluation because such algorithms produce non closed contours rather than segmented regions.

Nevertheless, boundary algorithm evaluation metrics exist, like the precision-recall curves [1].

The precision-recall curve—is a parametric curve that captures the trade-off between accuracy and noise as the detector threshold varies. Precision is the fraction of detections that are true positives rather than false positives, while recall is the fraction of true positives that are detected rather than missed. In terms of probability, precision is the probability that the detector's signal is valid, and recall is the probability that the ground truth data was detected. A similar approach was taken by Bowyer et al. [30] for boundary detector evaluation with receiver operating characteristic (ROC) curves. The axes for an ROC curve are fallout and recall. Recall, or hit rate, is the same as mentioned. Fallout, or false alarm rate, is the probability that a true negative pixel was marked as false positive pixel

To approximate the Precision-Recall curves' optimal trade-off, F-measure is calculated for every value of threshold. The F-measure is the harmonic mean of Precision and recall. It is computed as follows

$$F\ measure = \frac{2.Precision.Recall}{(Precision+Recall)} \tag{10}$$

Chapter 3

PROPOSED METHODOLOGY

1. *Behavior of the fireflies*

Fireflies belong to the family of insects that live in tropical environments. Fireflies have wings and they produce cold light via Bioluminescence. Their larvae are called glowworms. The flashes produced by fireflies have three main purposes:

- To attract mating partners
- To attract a potential prey
- Protective warming mechanism

Each firefly has a unique flashing pattern. They have limited light intensity. The flashing light of fireflies can be manipulated in a way that is linked with the objective function that is to be optimized.

2. *Firefly Algorithm*

The flashing capabilities of fireflies is used to obtain optimization algorithms. The basic assumptions made for firefly algorithms are as follows:

- Fireflies are unisex so that a firefly is attracted to others irrespective of their sex
- Attractiveness depends directly on the brightness of the flash i.e., less bright firefly will move towards brighter firefly. If there is no brighter firefly, the current firefly will move randomly
- Brightness of a firefly depends upon the optimization function.

The two issues in the firefly algorithm are : the variation of intensity of light and the derivation of formula for attractiveness

For problems involving maximization of objective function, the brightness ***I*** of a particular location ***x*** can be formulated as

$$I(x) \propto f(x) \qquad (9)$$

Where ***f(x)*** is the optimization function at location ***x.***

The attractiveness is not absolute but relative i.e., it is seen from the point of view of other fireflies or as they see it. Therefore, attractiveness varies with the distance r_{ij} between firefly *i* and firefly *j*. Also, the intensity of flash decreases as the distance between the source and the viewer. The light intensity *I(r)* varies in accordance with the inverse square law as follows –

$$I(r) = \frac{I_s}{r^2} \tag{10}$$

Where I_s is the intensity of the source of the flash. For a medium that has a fixed value of ligt absorption, γ , the intensity I varies with distance value r as shown below .

$$I(r) = I_0\, e^{-\gamma r} \tag{11}$$

Where I_0 is the original light intensity.

Also, as firefly's attractiveness is dependent on the light intensity as seen by nearby fireflies , the attractiveness β of a firefly is defined as

$$\beta = \beta_0 e^{-\gamma r^2} \tag{12}$$

Where β_0 is the attractiveness value at r= 0

The distance between any two fireflies *i* and *j* at x_i and x_j, respectively is equal to the value of Cartesian distance between them .

$$r_{i,j} = \left\| \boldsymbol{x_i} - \boldsymbol{x_j} \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,j} - x_{j,k})^2} \qquad (13)$$

The movement of a firefly *i* attracted to a brighter firefly *j* is formulated as follows –

$$x_i = x_i + \beta_0 e^{-\gamma r_{i,j}^2} \left(x_j - x_i\right) + \alpha \,(rand - 0.5) \qquad (14)$$

Where the second term includes attraction and third is the randomization with α as the randomization parameter.

The major steps involved in the traditional firefly algorithm are provided in Fig (3).

3.1. Firefly algorithm

The first step in the proposed algorithm is to suppress the texture and enhance the low contrast regions. In order to achieve that, the image is (1) Contrast Adjusted to enhance the low contrast region. This, in turn, enhances the textures too. Hence the image is (2) Softened using a blurring filter, for instance the Gaussian Blur Filter. Since blurring filter results in blurring of the image, the textures are suppressed subtly. This does not result in the loss of boundary contrast as they were made stronger using the contrast adjustment that was performed prior to softening. Thus, the texture is suppressed while maintaining the contrast amongst the boundary pixels

3. *Proposed Algorithm*

Adapting to the principles of Firefly algorithm as explained in the last section, we distribute N fireflies throughout the image. The value of N depends upon the content and size of the image. The aim here is to cover the entire image with fireflies in order to gather the information of entire image.

For computational ease, a unique number is assigned to each firefly, from *1, 2…..N*.

A. *Assumptions*

- All fireflies are unisex
- The brightness is defined by the optimization function
- A firefly moves only if it finds a brighter firefly else it does not move

The proposed firefly algorithm begins with reading the digital image whose boundaries are to be obtained. It is then converted to its equivalent grayscale image.

B. *Initialization*

At every pixel *p*, the optimization function *f(p)* is calculated as given below :

$$f(p) = \frac{1}{9}\sum_{i=-1}^{i=1}\sum_{j=-1}^{j=1} B(x_p + i,\ y_p + j) \qquad (15)$$

Where *B(x,y)* being the boundary value at point (x,y) which is calculated as given below

$$B(x,y) = \sqrt{B_x(x,y)2 + B_y(x,y)2} \qquad (16)$$

Where $B_x(x, y)$ and $B_y(x, y)$ determine the intensity difference between the neighbors of the particles horizontally as well as vertically respectively. The masks that can be used for calculating the horizontal as well as the vertical difference can be

1	1	1
0	0	0
-1	-1	-1

And

1	0	-1
1	0	-1
1	0	-1

respectively.

These masks are hovered over the entire image and the result is stored and saved.

C. Fireflies

The number of fireflies, *nfly* , is initialized so is the number of generations involved for calculations and movements, *ngen*. The *nfly* number of fireflies are now distributed over the entire image randomly. The value of *ngen* shouldn’t be too large or too small. Larger value of *ngen* results in high computational time whereas a lower value results in vague optimization results.

D. Attraction and Movement

For each firefly, number of surrounding fireflies is calculated. Sequentially, the distance between the firefly under consideration, say firefly i and the neighboring fireflies is calculated using the equation (15) and (16).

The value of attraction factor β is calculated as follows

$$\beta = \beta_0 \exp(-\gamma r^2) \qquad (17)$$

Where value of β_0 is the initial value of attraction factor at distance r = 0.

The randomness factor α is also calculated as follows

$$\alpha = \alpha_0 \, \delta^{itr} \qquad (18)$$

Where α_0 is the initial value of the randomness factor, δ is the value of randomness reduction factor. This ensures that during the random movement of the firefly, it doesn't go out of meaningful locations to a completely random location.

Now, for every neighboring firefly, say j, the movement criteria of firefly *i* is as follows

If $f(j) > f(i)$

Then, firefly *I* moves towards firefly *j* via the following rule

$$x_{new} = x_i(1 - \beta) + x_j(\beta) + \alpha(rand - 0.5) \qquad (19)$$

And

$$y_{new} = y_i(1 - \beta) + y_j(\beta) + \alpha(rand - 0.5) \qquad (20)$$

where (x_i, y_i) is the initial position of firefly *i* before movement and (x_{new}, y_{new}) is the position of firefly *i* after movement towards the firefly *j* located at (x_j, y_j) in the input image and

rand is a random value generated to provide appropriate randomness to the movement of the firefly under consideration.

This process is repeated for *nfly* fireflies for *ngen* number of generations.

E. Boundary particles selection

Now that the movement phase is over, the entire image is now covered with fireflies that possess their respective optimization function's value or in other terms their own respective brightness intensity values. This value determines the strength of boundary at the given pixel position. Low value of ***f*** means boundary is weak here or not present at all. Higher Values of ***f*** means strong shift in the brightness of the given image at the pixel position under consideration.

Hence, in order to separate the boundary pixels from the remaining pixels, we apply a threshold value to the ***f*** values of all the fireflies. All the fireflies that possess a ***f*** value greater than or equal to *thres,* where *thres* stands for the threshold Value for the image particles' optimization function's value, are displayed or selected as the boundary pixels. The remaining ones are omitted from the result.

This can be mathematically shown as follows –

$$Thresholding(m) = \begin{cases} Displayed, f(m) \geq \text{thres} \\ Removed,\ f(m) < thres \end{cases} \qquad (21)$$

After the thresholding has been performed on all the particles, the image now contains only the boundary pixels. Threshold value thres is determined experimentally by observation. The value of thres depends upon the Boundary Value range of the image's particles. The value of the variable thres shouldn’t be too less or too large but optimal. Fig 5 shows the effects of different values of thres on the output image displaying boundaries.

3.2.Algorithm for Boundary detection

```
Step 1: Read the input image img.
Step 2: Apply contrast adjustment to the image.
Step 3: Blur the image using a softening filter.
Step 4: Initialize the number of fireflies, nfly.
Step 5: Spread the nfly fireflies throughout the image uniformly or randomly.
Step 6: Update the value of brightness function, randomization factor and attraction factor via
    equations ( ) , () and ( )
Step 7: For each firefly, repeat till exhaustion is achieved:
    Step 7.1: Calculate the number of fireflies around it.
    Step 7.2: The firefly moves towards the brightest firefly in vicinity in accordance with
        equation ( )
    Step 7.3:  Exhaustion is achieved when either of the following occurs :
        • Each firefly is brightest in its vicinity
        • Pre-defined number of maximum iterations have passed
    Step 7.4:  The value of attraction factor is updated as equation ( )
Step 8: Apply thresholding to the fitness function or the brightness values of the fireflies
    using eq (20)
```

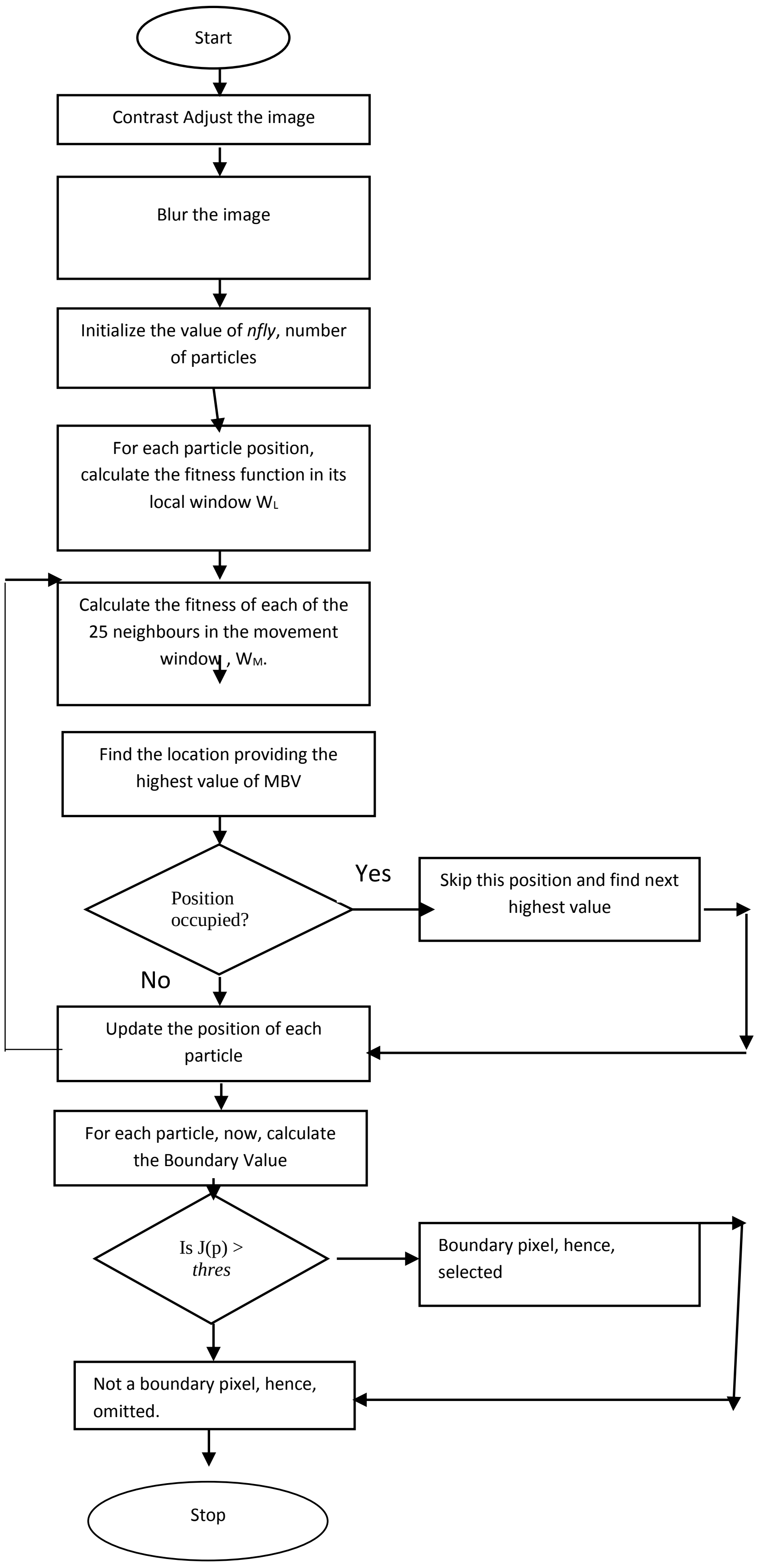
The flowchart of the algorithm is given in Fig. 3.2.

3.3. Parameter Selection

The parameters involved are the number of particles, N, distance between every two pixels , *dba*,

a) The number of particles *N* = (size of input image)

b) Distance between any two particles *dba*=1



Start
Contrast Adjust the image
Blur the image
Initialize the value of *nfly*, number of particles
For each particle position, calculate the fitness function in its local window W_L
Calculate the fitness of each of the 25 neighbours in the movement window , W_M.
Find the location providing the highest value of MBV
Position occupied?
Yes
Skip this position and find next highest value
No
Update the position of each particle
For each particle, now, calculate the Boundary Value
Is J(p) > *thres*
Boundary pixel, hence, selected
Not a boundary pixel, hence, omitted.
Stop

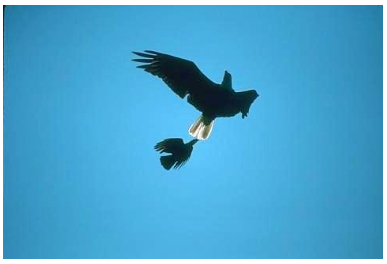
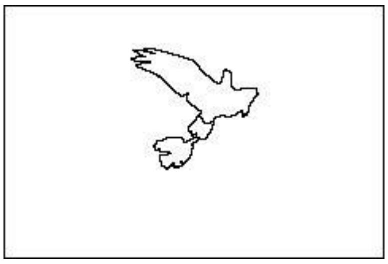
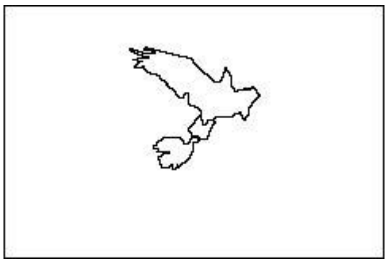
Fig. 3. Flowchart of the Proposed Algorithm

Chapter 4

EXPERIMENTAL RESULTS

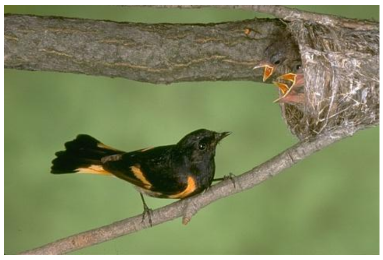
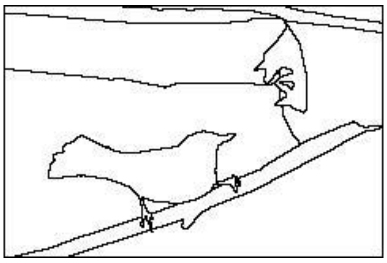
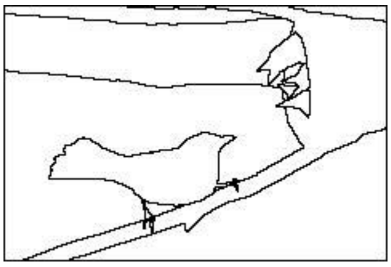
The proposed algorithm has been implemented on Intel Core i5 CPU at 2.50 GHz and MATLAB is used to implement it .The BSDS image dataset is partitioned into training images and test images. We ran the algorithm on the test of 100 overall images. The training set was used to empirically find the value of thres.

The ambiguity in the ground truth of images is shown in Fig 4.1 – 4.5.The reason for arise of ambiguity lies in the fact that different people categorize different pixels into boundaries which, in turn, produces different boundary images for the same original sample input image. Thus, more the number of people suggesting, more is the number of different boundary images for the same sample input image.

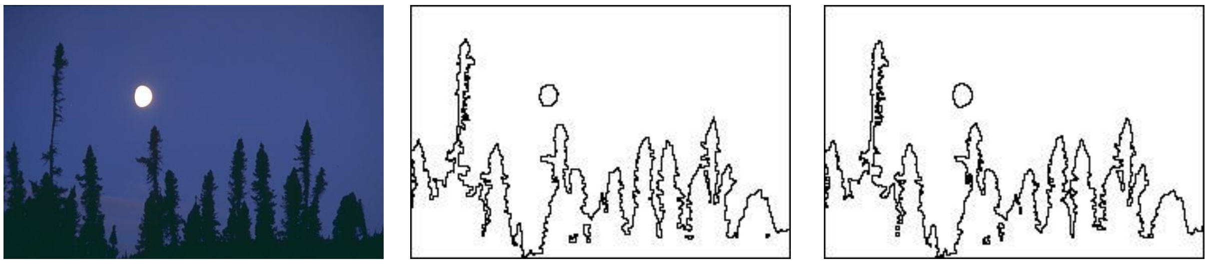
1(a) 1(b) 1(c)

Fig.4.1 Original Image “Eagle” and its corresponding two ground truth images

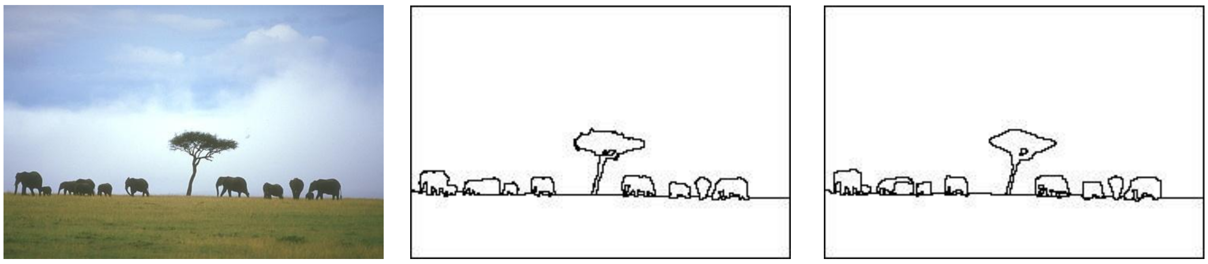
2(a) 2(b) 2(c)

Fig.4.2 Original Image “Bird” and its corresponding two ground truth images



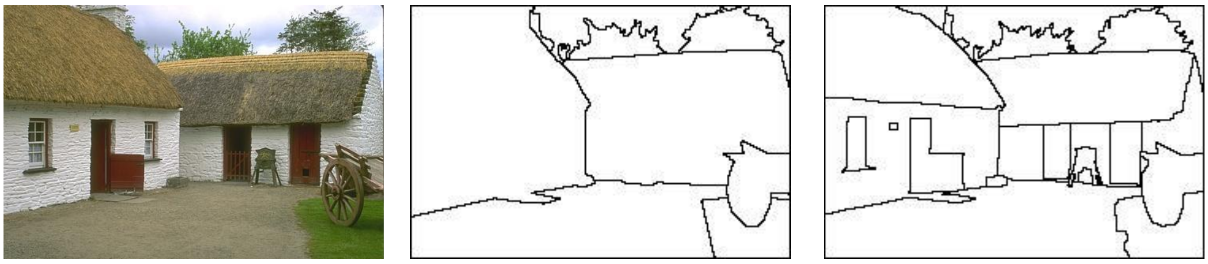
3(a) 3(b) 3(c)

Fig 4.3 Original Image “Night” and its corresponding two ground truth images



4(a) 4(b) 4(c)

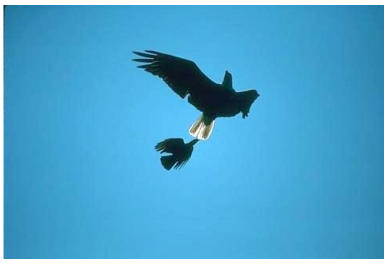
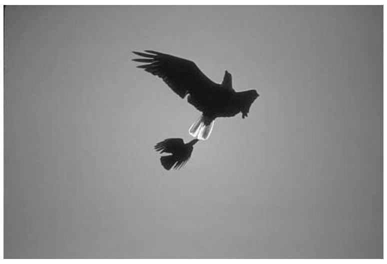
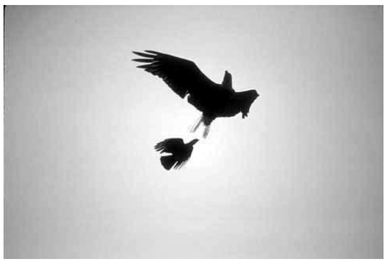
Fig. 4.4 Original Image “Elephants” and its corresponding two ground truth images



5(a) 5(b) 5(c)

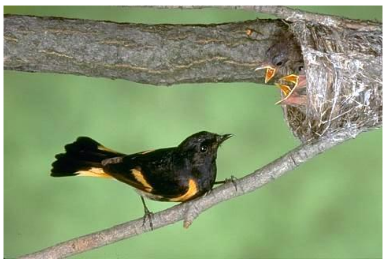
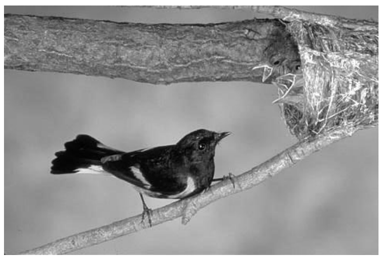
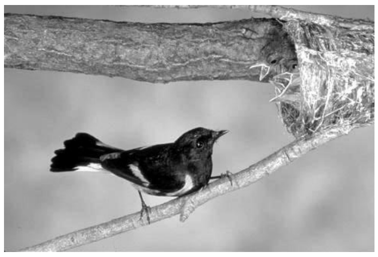
Fig. 4.5 Original Image “Hut” and its corresponding two ground truth images

Figure 4.6 - 4.10 display the pre-process of the algorithm – the task of suppressing the texture. This is achieved by performing the contrast adjustment using the MATLAB command *imadjust* . Now the image is softened using a blurring filter. The order of the sub-images is as follows : First the original image is displayed , its corresponding grayscale image and then the corresponding texture suppressed gray scale image.

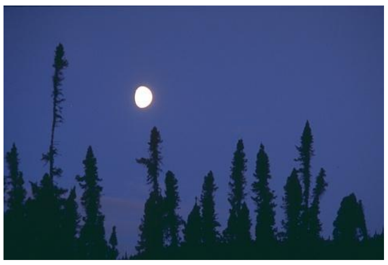
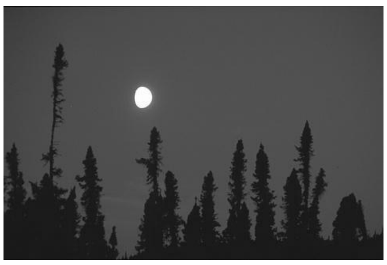
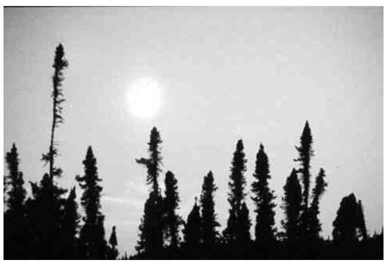
6(a) 6(b) 6(c)

Fig. 4.6 (a) Original Image “Eagle” (b) corresponding gray scale image (c) after texture suppression

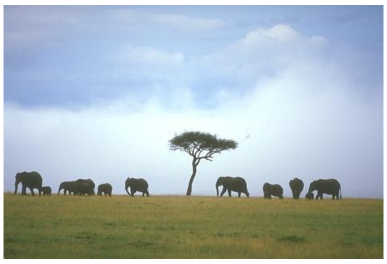
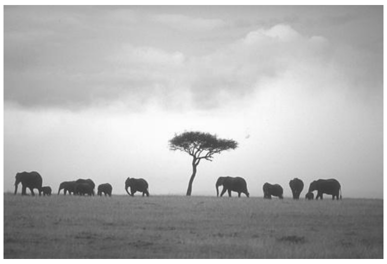
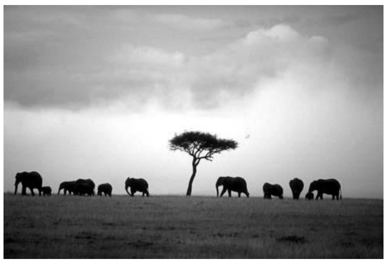
7(a) 7(b) 7(c)

Fig. 4.7 (a) Original Image “Bird” (b) corresponding gray scale image (c) after texture suppression

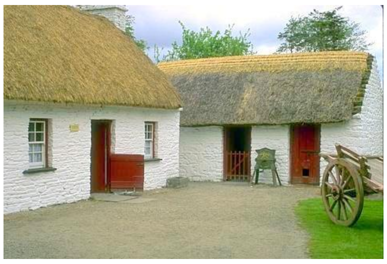
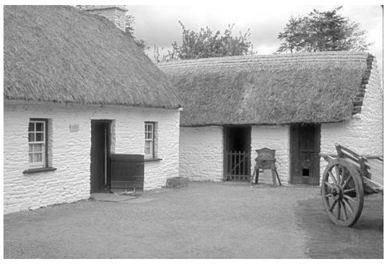
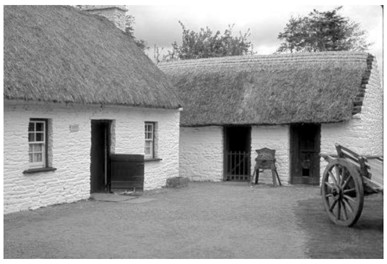
8(a) 8(b) 8(c)

Fig. 4.8 (a) Original Image “Night” (b) corresponding gray scale image (c) after texture suppression

9(a) 9(b) 9(c)

Fig. 4.9 (a) Original Image "Elephants" (b) corresponding gray scale image (c) after texture suppression

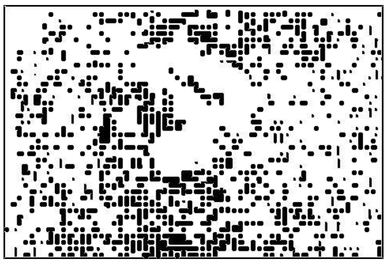
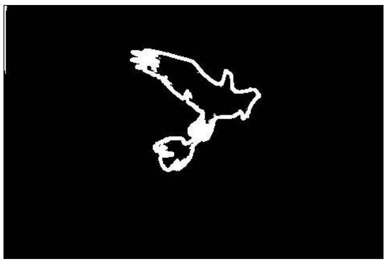




10(a) 10(b) 10(c)

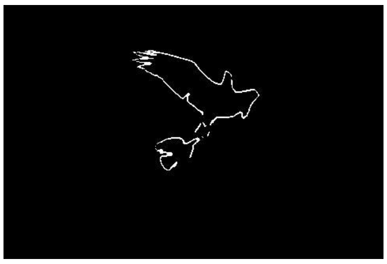
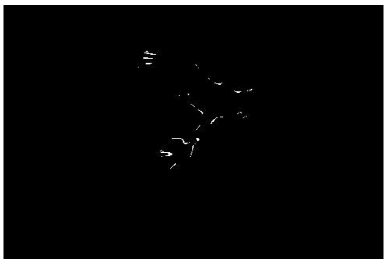
Fig. 4.10 (a) Original Image "Hut" (b) corresponding gray scale image (c) after texture supression

After that the fireflies are spreaded out . The value of *nfly* was taken to be 200. The respective values of α, β , γ and δ are set to be 0.2, 1 , 1 and 0.97 respectively. After the movement phase been completed, the value of *thres* is set.It should be optimal – not too high not too low. If the value of threshold is too high, only a few particles will be selected as the boundary pixels. Hence, no or incomplete boundaries are detected. If the value is higher than the optimal value but lower than the highest one, incomplete boundaries are displayed. The boundaries detected are not continuous but are missing at placed. If the value is too low, almost all the particles are selected as the boundary pixel hence almost no boundary is noticeable. If the value is low, then too much details are displayed and identified as boundary pixels or firefly positions. Texture is also displayed along with the boundaries. Hence, the value should be taken experimentally.

For our implementation, for images taken from BSDS(300) [1] with resolution 481x 321 , the value of *thres* is almost equal to 150.

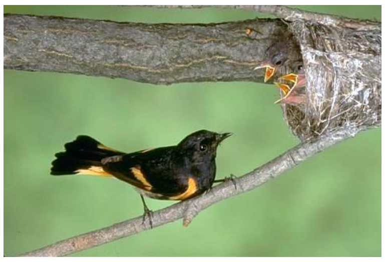
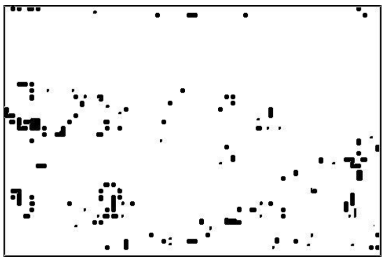
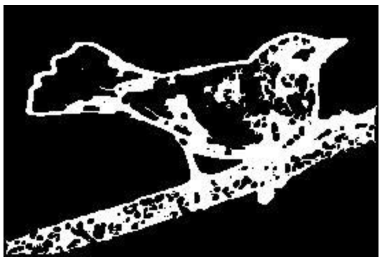



11(a) 11(b) 11(c)

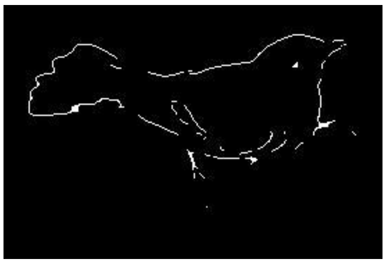
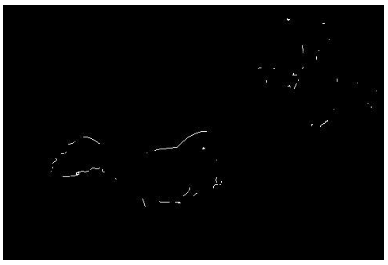
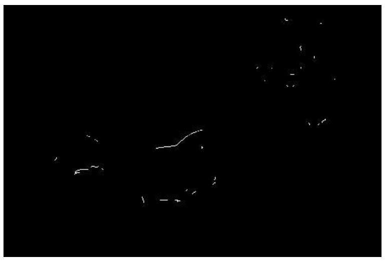



11(d) 11(e) 11(f)

Fig. 4.11 (a) Original Image “eagle” (b) output with thres=0 (c) output with thres=40 (d) output with thres = 150 (e) output with thres = 240 (f) output with thres = 300

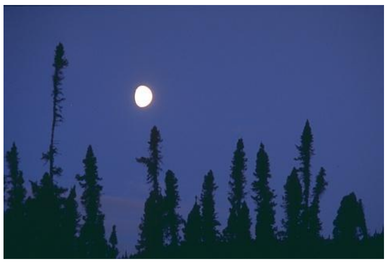




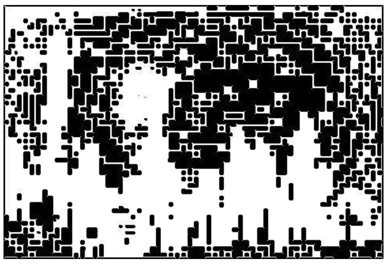
12(a) 12(b) 12(c)

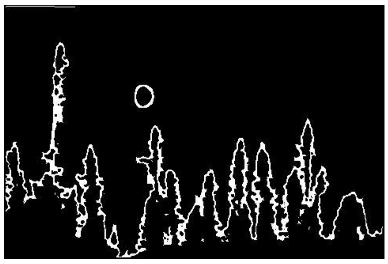




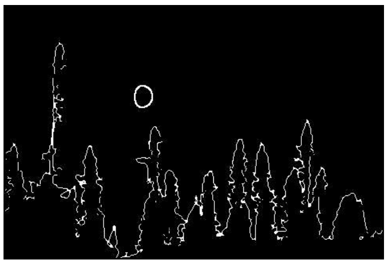
12(d) 12(e) 12(f)

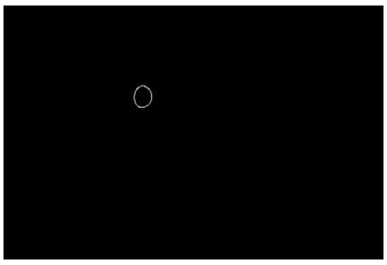
Fig. 4.12 (a) Original Image “Bird” (b) output with thres=0 (c) output with thres=40 (d) output with thres = 150 (e) output with thres = 240 (f) output with thres = 300


13(a)


13(b)


13(c)


13(d)


13(e)

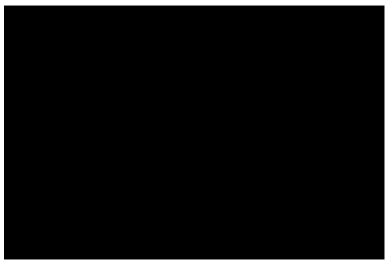
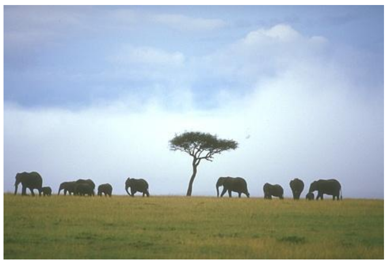
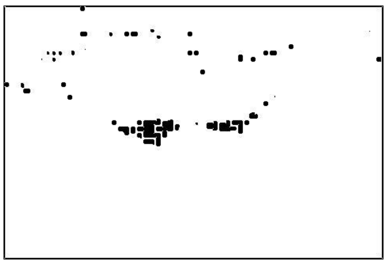
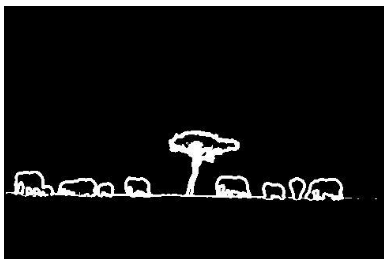
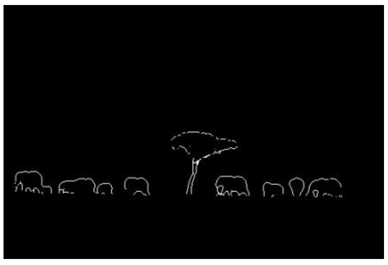

13(f)

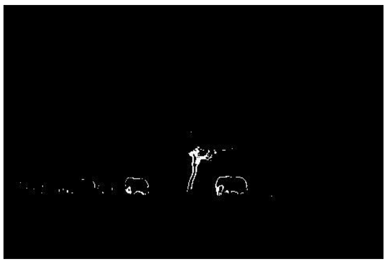
Fig. 4.13 (a) Original Image “Night” (b) output with thres=0 (c) output with thres=40 (d) output with thres = 150 (e) output with thres = 240 (f) output with thres = 300


14(a)


14(b)


14(c)


14(d)


14(e)

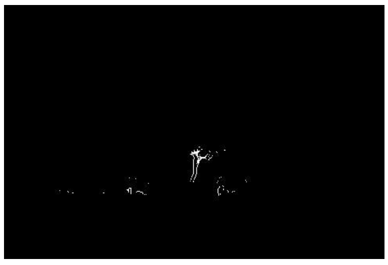
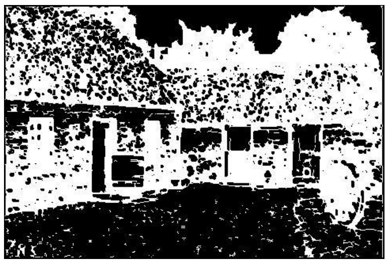
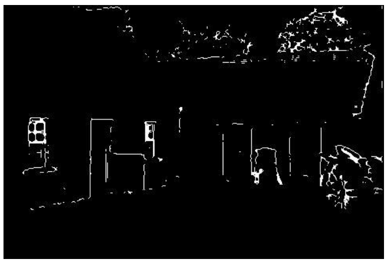

14(f)

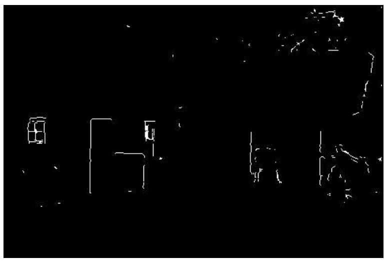
Fig. 4.14 (a) Original Image “Elephants” (b) output with thres=0 (c) output with thres=40 (d) output with thres = 150 (e) output with thres = 240 (f) output with thres = 300

15(a)

15(b)


15(c)


15(d)


15(e)

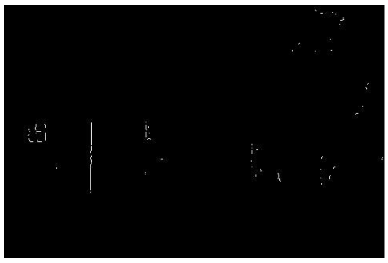
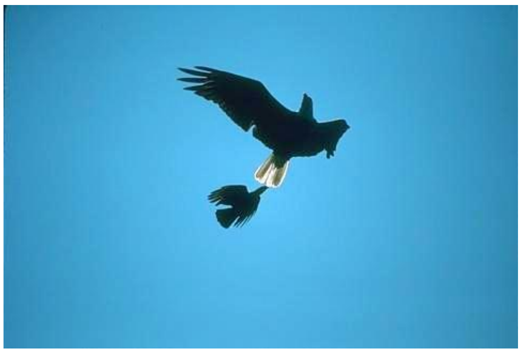
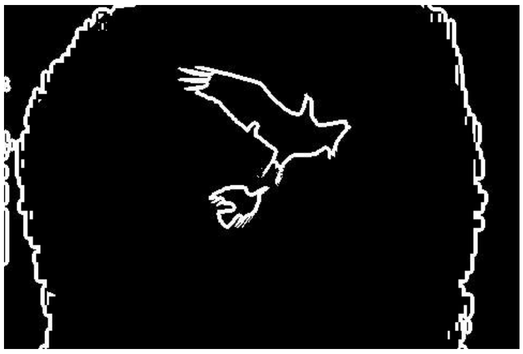

15(f)

Fig. 4.15 (a) Original Image "Hut" (b) output with thres=0 (c) output with thres=40 (d) output with thres = 150 (e) output with thres = 240 (f) output with thres = 300

One of the prominent methods for the extracting boundaries from a given image is to perform morphological operation of Erosion and then subtracting it from the corresponding black and white image (Explained in Section). Figures 4.16 – 4.20 . As can be observed from the figures below, erosion succeeds in drawing out the high contrast boundaries but also , drags along the texture with high contrast or the slightest change in contrast or brightness. Low contrast regions are not covered by this process . This is evident by the figure 4.18 in which trees of low contrast are not covered in the boundary pixels of the original image.Also, as can be observed in the "Bird" image, the trunk of the tree is full of texture and this method counts those texture pixels in the boundary pixels as well.

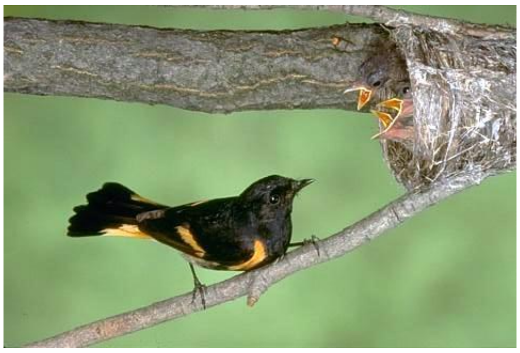


16(a)

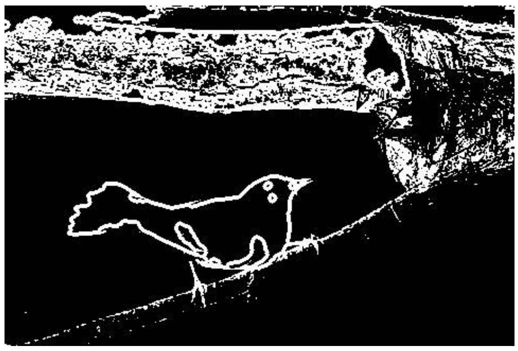


16(b)

Fig. 4.16 (a) Original Image “Eagle” (b) Boundary image as produced by morphological operation of erosion

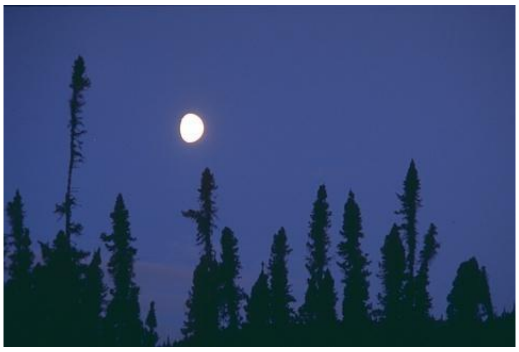


17(a)

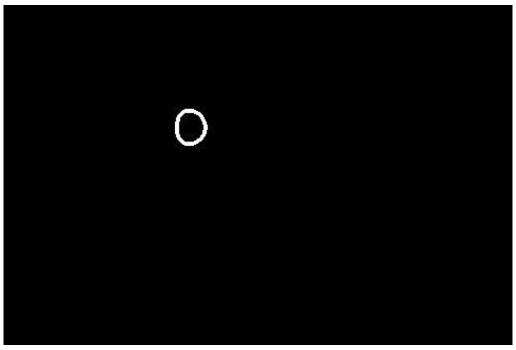


17(b)

Fig. 4.17 (a) Original Image “Bird” (b) Boundary image as produced by morphological operation of erosion

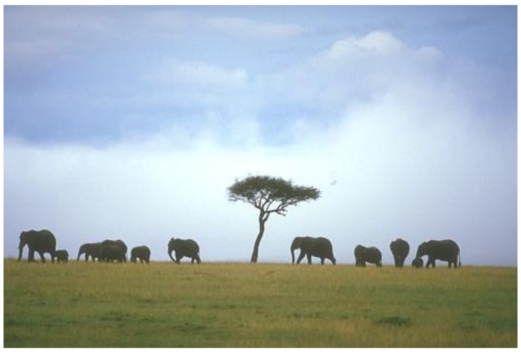


18(a)

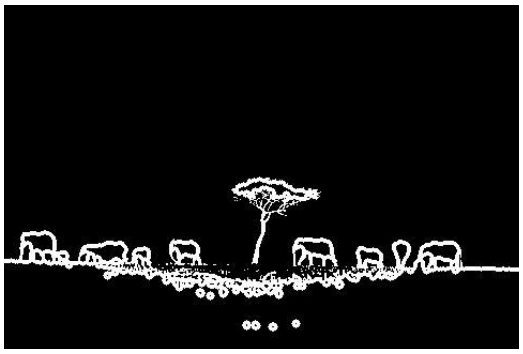


18(b)

Fig. 4.18 (a) Original Image “Night” (b) Boundary image as produced by morphological operation of erosion

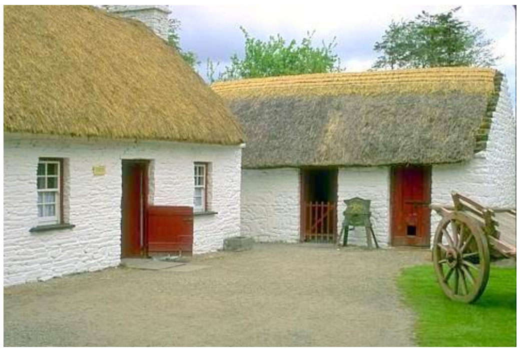


19(a)

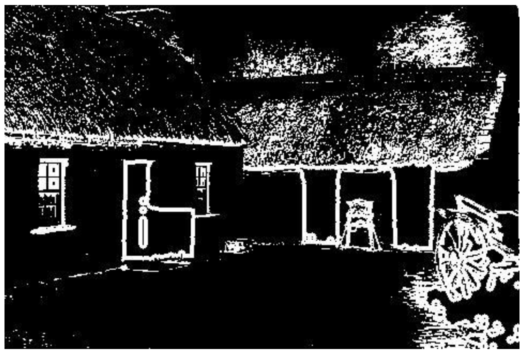


19(b)

Fig. 4.19 (a) Original Image “Elephants” (b) Boundary image as produced by morphological operation of erosion



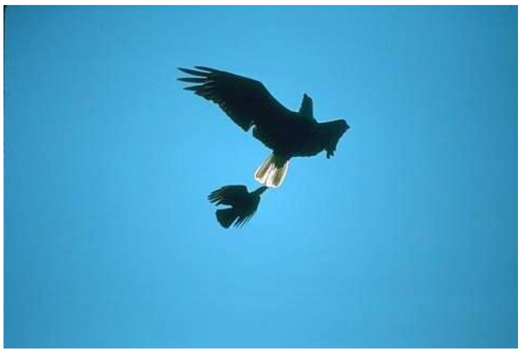
20(a)



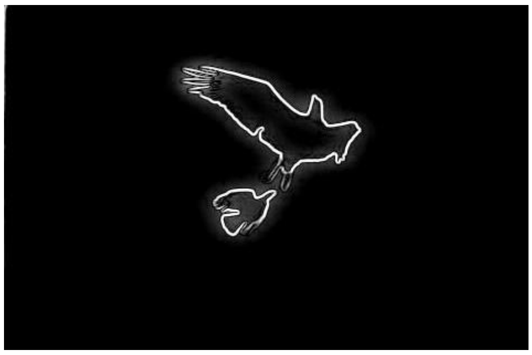
20(b)

Fig. 4.20 (a) Original Image “Hut” (b) Boundary image as produced by morphological operation of erosion

Next, the process of extracting the boundaries is performed by the technique mentioned in [18]. Figures 4.21-4.25 include the results obtained by the process of filtering the input image in frequency domain with a high pass filter and then converting it back into spatial domain . This filter removes crowded particle regions. As can be observed in fig. 21-25 , the boundaries produced are not crisp but blurred and have a soft touching to it at every pixel. Also as can be seen in the “night” image, the moon is not evaluated to be a boundary pixel at all.

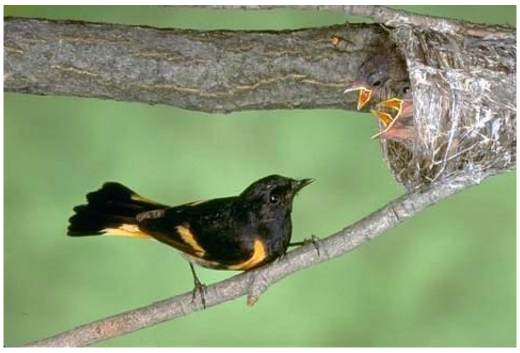


21(a)

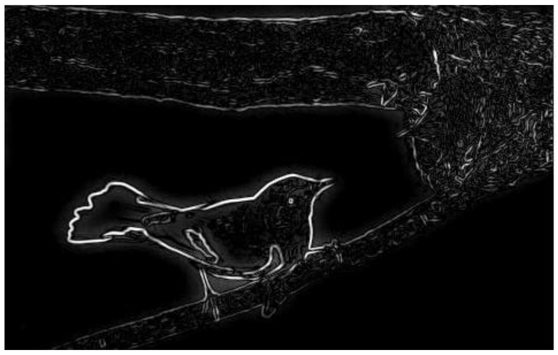


21(b)

Fig. 4.21 (a) Original Image "Eagle" (b) Boundary image as produced by frequency domain HP filtering of gradient image[18]

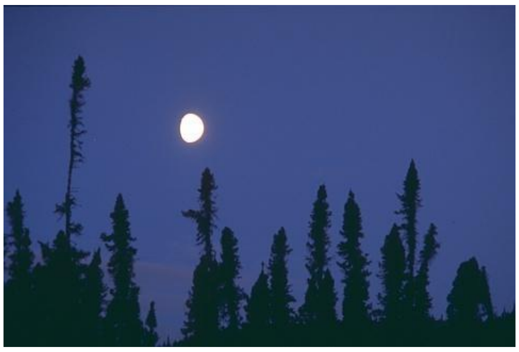


22(a)

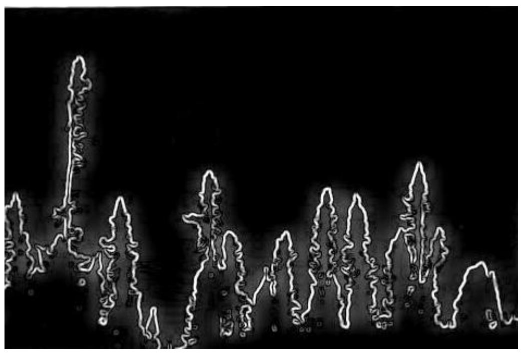


22(b)

Fig. 4.22 (a) Original Image "Bird" (b) Boundary image as produced by frequency domain HP filtering of gradient image[18]

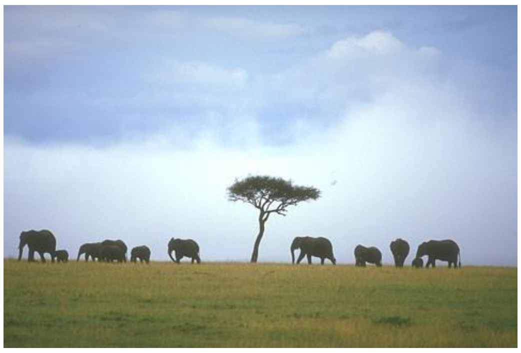


23(a)

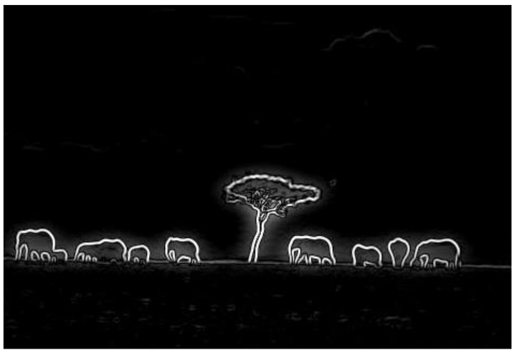


23(b)

Fig. 4.23 (a) Original Image "Night" (b) Boundary image as produced by frequency domain HP filtering of gradient image[18]

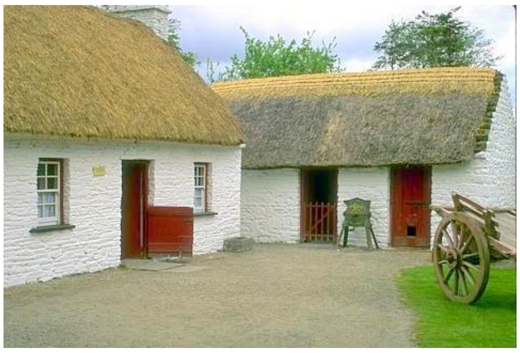


24(a)

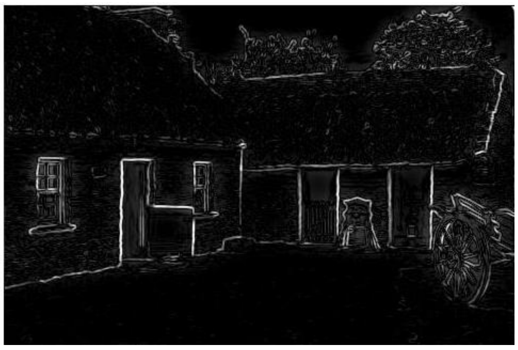


24(b)

Fig. 4.24 (a) Original Image "Elephants" (b) Boundary image as produced by frequency domain HP filtering of gradient image[18]



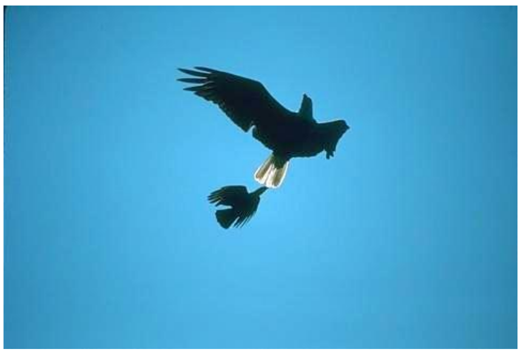
25(a)



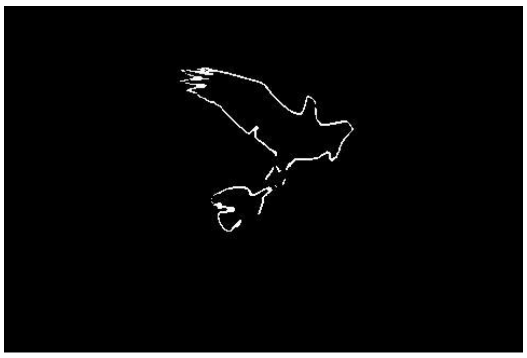
25(b)

Fig. 4.25 (a) Original Image "Hut" (b) Boundary image as produced by frequency domain HP filtering of gradient image[18]

Figures 4.26-4.30 show the boundaries as produced by the ACO method. As can be observed, the boundaries produces are crisp and with the exception of high contrast textures, all the boundaries are clearly extracted and the textures have also been suppressed to a certain degree . The results produced are better than those produced in figures 4.16-20 and those of fig 4.21 – 4.25. Visually, our algorithm has performed better than the process of extracting boundary via the morphological operation of Erosion and the method mentioned in [18].

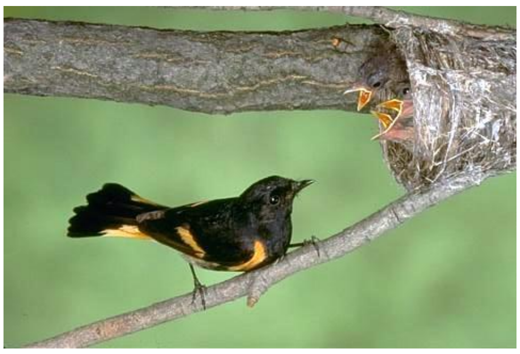


26(a)

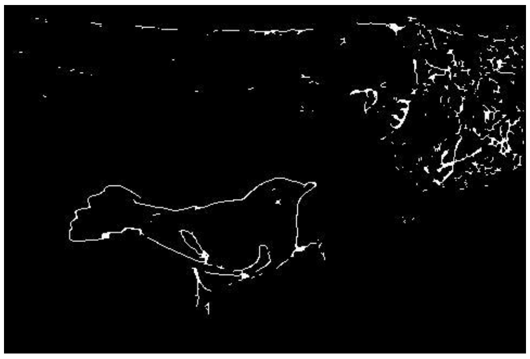


26(b)

Fig. 4.26 (a) Original Image “Eagle” (b) Boundary image as produced by ACO for boundary detection

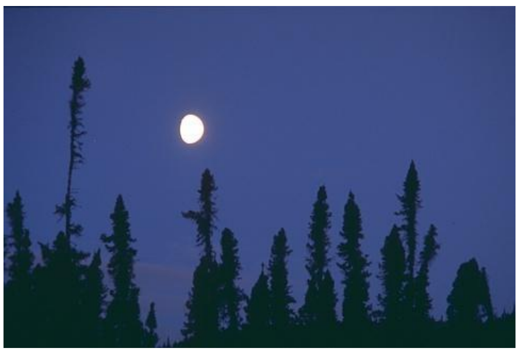


27(a)

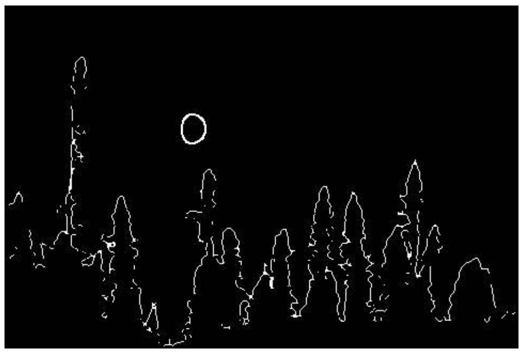


27(b)

Fig. 4.27 (a) Original Image “Bird” (b) Boundary image as produced by ACO for boundary detection

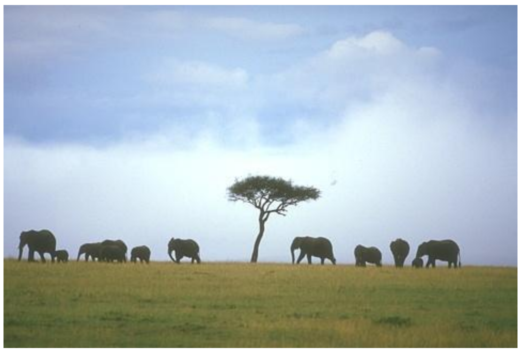


28(a)

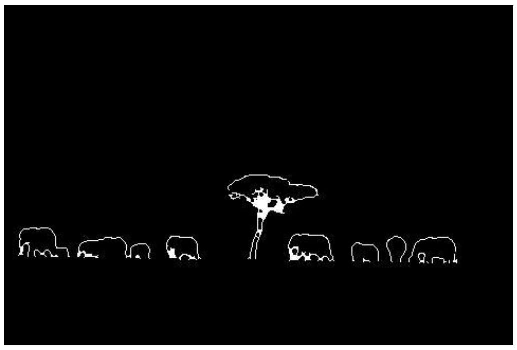


28(b)

Fig. 4.28 (a) Original Image “Night” (b) Boundary image as produced by ACO for boundary detection

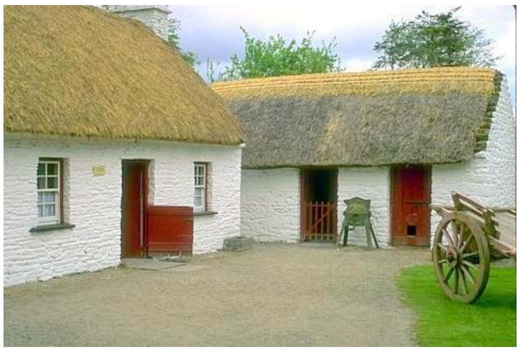


29(a)

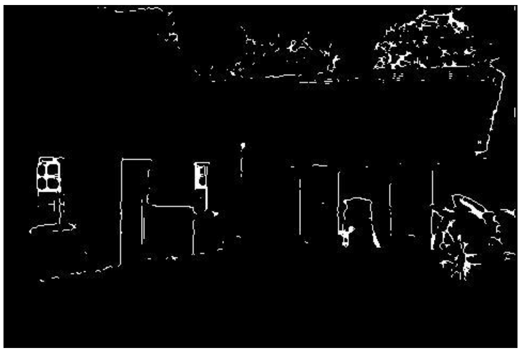


29(b)

Fig. 4.29 (a) Original Image “Elephants” (b) Boundary image as produced by ACO for boundary detection



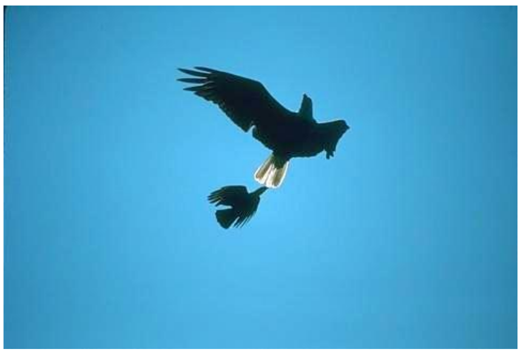
30(a)

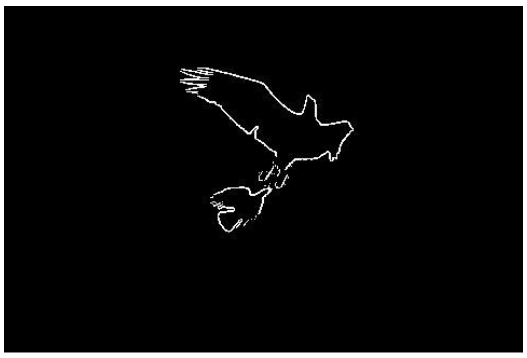


30(b)

Fig. 4.30 (a) Original Image “Hut” (b) Boundary image as produced by ACO for boundary detection

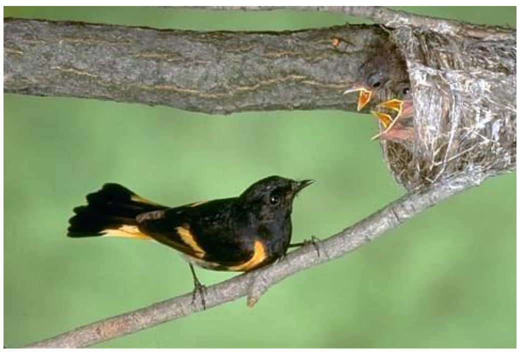
Figures 4.31-4.35 show the boundaries as produced by the Firefly method. As can be observed, the boundaries produces are sharp and with the exception of high contrast textures, all the boundaries are clearly extracted and the textures have also been suppressed to a significant amount . The results produced are better than those produced in figures 4.16-20 and those of fig 4.21 – 4.25. Visually, our algorithm has performed better than the process of extracting boundary via the morphological operation of Erosion and the method mentioned in [18].

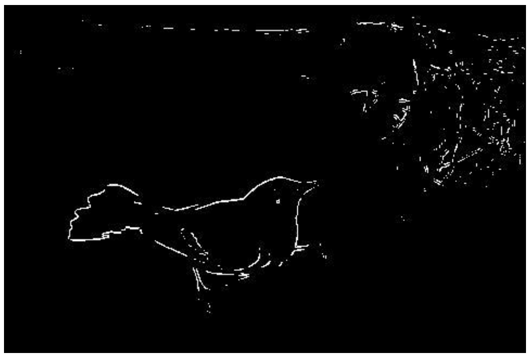




31(a) 31(b)

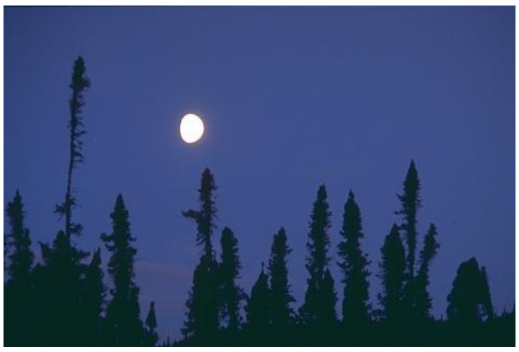
Fig. 4.31 (a) Original Image “Eagle” (b) Boundary image as produced by firefly algorithm of boundary detection.

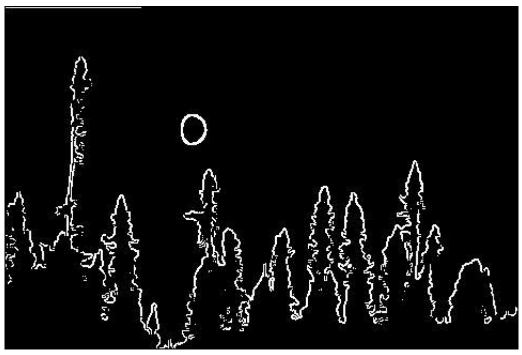




32(a) 32(b)

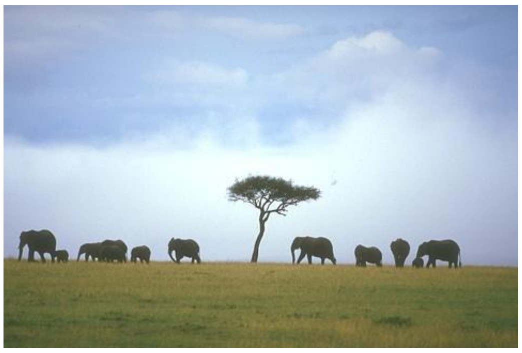
Fig. 4.32 (a) Original Image “Bird” (b) Boundary image as produced by firefly algorithm of boundary detection.





33(a) 33(b)

Fig. 4.33 (a) Original Image “Night” (b) Boundary image as produced by firefly algorithm of boundary detection.


34(a)

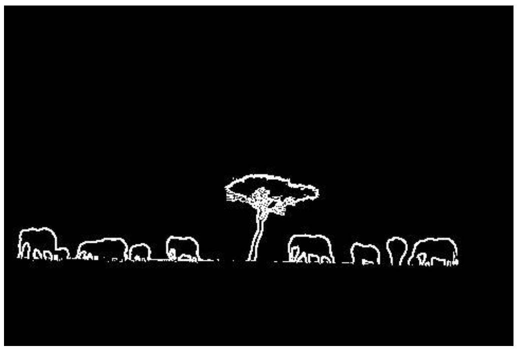
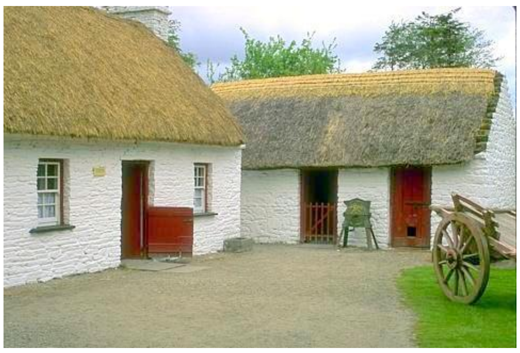

34(b)

Fig. 4.34 (a) Original Image “Elephants” (b) Boundary image as produced by firefly algorithm of boundary detection.


35(a)

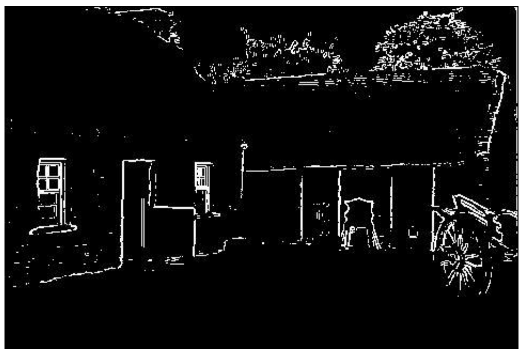

35(b)

Fig. 4.35 (a) Original Image “Hut” (b) Boundary image as produced by firefly algorithm of boundary detection.

Table 1 shows the values of three metrics as evaluated on these three algorithms to extract out the boundary pixels from an input image. Column 1 states the name of the image , Column 2 shows the F-measure and the precion recall values as produced by the erosion method of boundary extraction from the 5 images which have already been used above. Column 3 displays the various values of metrics Precision, Recall and the F-measure obtained by applying the frequency domain high pass filter on the gradient image as done in [18]. Column 4 depicts the outputs as produced by the ACO method of boundary detection. Last column includes the resultant values of precision, recall and the f-measure for the results produced by the proposed algorithm. As can be observed from the table, the proposed algorithm has worked better than the aforementioned methods of boundary extraction with acceptable exceptions.

TABLE 1. Comparison of F measure values obtained by performing boundary detection by different methods vs the proposed methods

Image	***Boundary detection by erosion***	***High Pass filtering of gradient Image in frequency domain***	***ACO method of boundary detection***	***Firefly Algorithm***
Eagle	F(0.0956,0.4329) = 0.1566	F(0.5614,0.4921) = 0.5245	F(0.5612,0.5597) = 0.5604	F(0.5733,0.5271)= 0.5492
Bird	F(0.0428,0.4069) = 0.0775	F(0.5122,0.5317) = 0.5217	F(0.4872,0.5734) = 0.5267	F(0.4921,0.5639) = 0.5256
Night	F(0.1777,0.0100) = 0.0189	F(0.5016,0.4986) = 0.4889	F(0.5161, 0.5782) = 0.5454	F(0.4525,0.4713) = 0.4617
Elephants	F(0.0601, 0.3555) = 0.1028	F(0.5712,0.5214) = 0.5451	F(0.5439 , 0.4912) = 0.5162	F(0.5691,0.5322) = 0.5500
Hut	F(0.1287, 0.3075) = 0.1815	F(0.5008,0.5222) = 0.5113	F(0.4821, 0.4964) = 0.4891	F(0.4928,0.5160) = 0.5041

Chapter 5

CONCLUSION

In this work, we present a novel technique for boundary detection inspired by the concept of Swarm Intelligence. Instead of optimizing some operation, our algorithm focuses on the goal of boundary detection by analyzing the image content right from the scratch.

In the nutshell, the major accomplishments of this algorithm include a novel movement scheme for the particles in order to collect the local information. This technique does not depend on any training sets or explicit object models. No pre-defined user knowledge is required.

Effort needs to be spent on the ambiguous regions of the image like the texture region as well as the low contrast boundary pixels.

Chapter 6

REFERENCES

[1] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application toevaluatingsegmentationalgorithmsandmeasuringecologicalstatistics, in: InternationalConferenceon Computer Vision, 2001, pp. 416–423.

[2] D. Martin, C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEETransactionson Pattern Analysis and Machine Intelligence26(5) (2004)530–549.

[3] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation, International Journal of Computer Vision 59 (2) (2004) 167–181.

[4] T. Cour, F. Benezit, J. Shi, Spectral segmentation with multiscale graph decomposition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 1124 – 1131.

[5] S. Thilagamani, N. Shanthi, A survey on image segmentation through clustering, International Journal of Research and Reviews in Information Sciences 1 (1) (2011) 16–19.

[6] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603 –619.

[7] J. Chen, T. Pappas, A. Mojsilovic, B. Rogowitz, Adaptive perceptual color-texture image segmentation, IEEE Transactions on Image Processing 14 (10) (2005) 1524 – 1536.

[8] J. Chang, J. Fisher, Efficient mcmc sampling with implicit shape representations, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 2081 –2088.

[9] W.Cui, Z.Guan, Z.Zhang ,An improved region growing algorithm for image segmentation ,in: International Conference on Computer Science and Software Engineering, 2008, pp. 93 – 96.

[10] N. Senthilkumaran, R. Rajesh, Edge detection techniques for image segmentation – a survey of soft computing approaches, International Journal of Recent Trends in Engineering 1 (2) (2009) 250–254.

[11] Y. Ramadevi, T. Sridevi, B. Poornima, B. Kalyani, Segmentation and object recognition using edge detection techniques, International Journal of Computer Science & Information Technology 2 (6) (2010) 153– 161.

[12] W. Ma, B. S. Manjunath, Edgeflow: a technique for boundary detection and image segmentation, IEEE Transactions on Image Processing 9 (8) (2000) 1375–1388.

[13] H. Wang, J. Oliensis, Generalizing edge detection to contour detection for image segmentation, Computer Vision and Image Understanding 114 (7) (2010) 731 – 744.

[14] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (5) (2011) 898–916.

[15] K.Hammouche, M.Diaf,P.Siarry, A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation, Comput. Vis. Image Underst. 109 (2) (2008) 163–175.

[16] S. Das, S. Sil, Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm, Inf. Sci. 180 (8) (2010) 1237–1256.

[17] C.-Y. Lee, J.-J. Leou, H.-H. Hsiao, Saliency-directed color image segmentation using modified particle swarm optimization, Signal Processing 92 (1) (2012) 1 – 18.

[18] Zhi-guo Qu, et.al. , Frequency domain filtering of gradient image for contour detection, Optik 124(2013) 1398-1401

[19] L. Li, D. Li, Fuzzy entropy image segmentation based on particle swarm optimization, Progress in Natural Science 18 (9) (2008) 1167 – 1171.

[20] D. Feng, S. Wenkang, C. Liangzhou, D. Yong, Z. Zhenfu, Infrared image segmentation with 2-d maximum entropy method based on particle swarm optimization (pso), Pattern Recognition Letters 26 (5) (2005) 597–603.

[21] Ulrich K, Simon Hawe, Klaus Diepold, A swarm intelligence inspired algorithm for contour detection in images, Applied soft computing 13 (2013) 3118-3129

[22] M. Setayesh, M. Zhang, M. Johnston, Edge detection using constrained discrete particle swarm optimisation in noisy images, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, 2011, pp. 246 –253.

[23] G.Beni, J.Wang, Swarmintelligenceincellularroboticsystems,in: NATOAdvancedWorkshoponRobots and Biological Systems, 1989, pp. 26–30.

[24] W. Yong, C. Jun, Using ant swarm intelligence for data clustering analysis, in: 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 429 –432.

[25] A. V. Baterina, C. Oppus, Image edge detection using ant colony optimization, WSEAS Transactions on Signal Processing 6 (2) (2010) 58–67.

[26] A.Amali Asha, S.Victor, A.Lourdusamy , Feature extraction in medical image using ant colony optimization : A study, International Journal on Computer Science and Engineering 3 (2) (2011) 714–721.

[27] E.Lakehal , A swarm intelligence based approach for image feature extraction ,in: International Conference on Multimedia Computing and Systems, 2009, pp. 31 –35.

[28] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: Proceedings of the First European Conference on Artificial Life, 1992, pp. 123–142.

[29] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[30] F. M. A. Mohsen, A new optimization-based image segmentation method by particle swarm optimization, International Journal of Advanced Computer Science and Applications Special Isssue (Image Processing and Analysis) (2011) 10–18.

[31] R. Unnikrishnan, C. Pantofaru, M. Hebert, Toward objective evaluation of image segmentation algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6) (2007) 929–944.

[32] M. Polak, H. Zhang, M. Pi, An evaluation metric for image segmentation of multiple objects, Image and Vision Computing 27 (8) (2009) 1223–1227.