IMPLEMENTATION OF MULTILEVEL THRESHOLDING BASED ANT COLONY OPTIMIZATION ALGORITHM FOR EDGE DETECTION OF IMAGES

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Spoorthy Kanajal Chandrakanth

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Software Engineering

May 2017

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Implementation of Multi level thresholding based Ant Colony Optimization Algorithm for Edge Detection of Images

By

Spoorthy Kanajal Chandrakanth

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Maria de los Angeles Alfonseca-Cubero

Dr. Saeed Salem

Approved:

May 12, 2017
Date

Dr. Kenneth Magel
Department Chair

ABSTRACT

Edges in an image characterize object boundaries in an image, which is helpful in image processing and feature extraction in a particular scene. One of the methods used to detect edges in an image is image thresholding, which replaces a pixel in an image with black pixel if an image intensity is less than some constant T. Edge detection is used to classify, interpret and analyze the digital images in various fields such as robots, the sensitive applications in military, etc. A hierarchical multilevel thresholding method for edge detection using the Ant Colony algorithm is used in this paper. Multilevel thresholding technique is applied in this paper based on previous work done by Ashour, A. S., El-Sayed (2014). Further, the results are produced for edge detection of images using ACO algorithm with multilevel thresholding. Both Gray scale images and color images are used to evaluate the efficiency of the algorithm.

ACKNOWLEDGEMENTS

I would like to thank Dr. Simone Ludwig for her continuous guidance and help in every aspect of this study and analysis. I would also like to thank my committee members for time and invaluable advice which has helped me bring this paper to complete closure.

DEDICATION

I dedicate my paper to my parents Chandrakanth Kanajal, Surekha, Spandana & My life partner Vijay and my Friends who have helped me all along this journey with their guidance and support

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Image segmentation is the process of dividing a digital image to multiple segments where each segment of image holds pixels of some common characteristics. The objective of image segmentation is generally to represent an image in a more meaningful way that is easy to analyze [1]. This particular area of image segmentation has been a classical problem from 1970s and many algorithms have been proposed since then.

In the image segmentation process, the main content that lies in an image is decomposed to simplify its representation. There have been multiple techniques proposed and implemented in the area of image segmentation. Some of those methods are thresholding, clustering, edge detection, histogram-based methods [2] etc.

This paper focuses on one of the methods of segmentation namely the "image Edge detection combined with multi-level thresholding." Image Edge Detection is, on a very basic level component in image handling/investigation. Edges portray limits thus have an extensive variety of helpful applications, for example, division and ID of items in scenes, machine vision, cosmology and microscopy imaging to give some examples [2]. Edge recognition is a method for checking sharp intensity changes, and is vital in further breaking down image content. It establishes the framework for image combination, shape extraction, image division, image coordinating, and image following. Numerous conventional edge location methodologies result in broken pieces, which prompt inadequate resultant pictures.

In this paper we used one of the popular methods of edge detection, which is thresholding. Simple Thresholding methods replace in an image with a black pixel if the image intensity is less than some fixed constant T, or a white pixel if it is greater than the constant T. Edge detection using thresholding has its significant importance in many research areas [1, 2]. In this paper, we study the combination of Ant colony optimization with a multi threshold algorithm proposed by A.S. Ashour, El-Sayed [3] for edge detection to find the efficiency of algorithm when compared to other edge detection techniques in use.

The problem identification of this work is defined as:

"The implementation of Multi level thresholding based Ant Colony Optimization algorithm for edge detection of images".

The Ant Colony Optimization (ACO) algorithm is a process or group of steps being inspired by the natural Ant movements. ACO is motivated by the distinct pheromone generation by Ants in order to communicate with each other. This pheromone trail is released by Ants on the ground or the surface they are moving on from glands found all over their bodies. These pheromone trails are detected by the ants' antennas placed in front. The amount of the pheromone deposit helps in finding the correct orientation of the path being communicated.

Since its proposal as the first Ant Colony optimization algorithm based system known as the ANTS System[1], it has become a dominant strategy for solving optimization related problems. Image edge detection is becoming a pronounced problem with the increased usability of images in every field of life. The Ant Colony algorithm simulates the optimization of edges in the images and then extracts the possible relations and links between them.

Efficient techniques and methods have been studied to provide efficient way of image edge detection. Efficiency here is defined as the accuracy of edges detected in the images. In this paper, we want to look at the results of combining ACO algorithm with multilevel thresholding to analyze the accuracy of the results.

The following research questions are being targeted in this research study:

[1] http://www.ckrause.org/2013/04/dynamic-adaptation-in-ant-colonies-and.html

RQ1: Is the Ant Colony algorithm applied with multi-level thresholding more efficient for image edge detection in terms of accuracy?

RQ2: Does using multiple levels of threshold increases the accuracy of image edge detection process?

This document has been divided into four major chapters starting from a thorough study of the background knowledge in Chapter 2, a literature review followed by the proposed methodology in Chapter 3, and the implementations in Chapter 4. Chapter 5 includes detailed results, and the discussions with conclusions are given in the final chapter (Chapter 6).

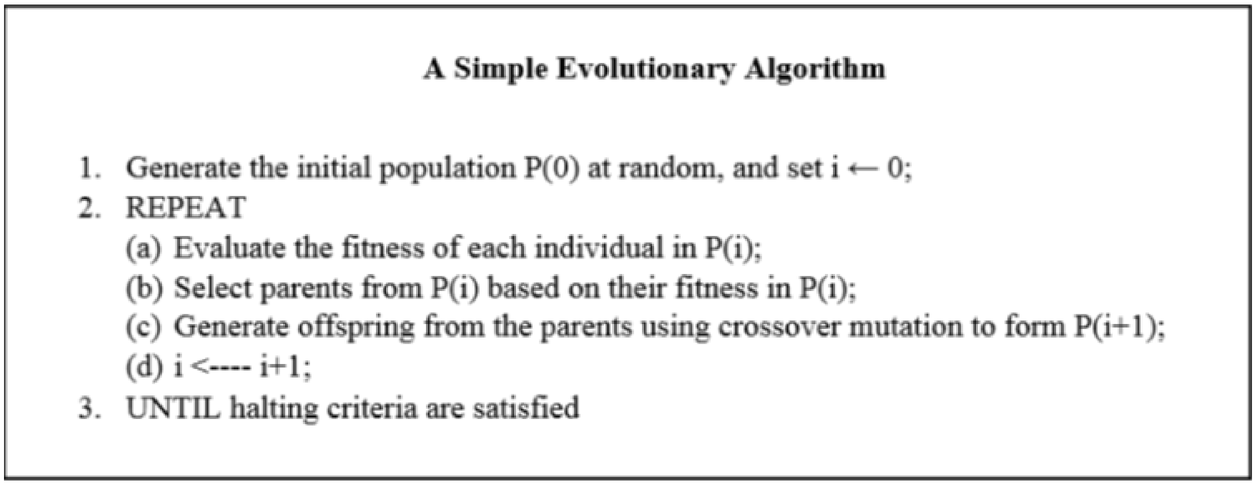
2. BACKGROUND

2.1. Evolutionary Computation

In the broadest terms, evolution can be portrayed as a two-stage iterative procedure, comprising of random variation taking place after selection. The connection between this depiction of development and the improving calculations that are the sign of transformative calculation. Just as regular advancement begins from a population of animals, the algorithmic methodology starts by selecting an introductory arrangement of contending answers for a specific issue. The set may be chosen by utilizing as to create arrangements randomly or any accessible learning about the issue.

These "parent" solutions then produce "offspring" by a pre-chosen method for arbitrary variety. The resultant arrangements are assessed for their adequacy and experience determination. Initializing a population of candidate solutions to an issue is the first step of an Evolutionary Algorithm [4]. The initial population is randomly varied for creating new solutions. Solutions are gauged on the basis of how well they address the undertaking. Finally, less fit solutions are being removed. The procedure is iterated utilizing the chosen set of arrangements until a particular stopping criterion is met.

It is the investigation of computational frameworks, which utilize thoughts and get motivations from common development. One of the standards obtained is survival of the fittest. Evolutionary Computation (EC) procedures can be utilized as a part of advancement [5], learning and outline. Evolutionary computation procedures do not require rich space learning. Nonetheless, space learning can be joined into EC procedures. Figure 1 shows pseudocode for an evolutionary algorithm



A Simple Evolutionary Algorithm

```
1. Generate the initial population P(0) at random, and set i ← 0;
2. REPEAT
   (a) Evaluate the fitness of each individual in P(i);
   (b) Select parents from P(i) based on their fitness in P(i);
   (c) Generate offspring from the parents using crossover mutation to form P(i+1);
   (d) i <---- i+1;
3. UNTIL halting criteria are satisfied
```

Figure 1: Simple evolutionary algorithm [2]

Evolutionary calculations can be viewed as population based procedure and calculations [6]. Their calculation procedure can be utilized as a part of advancement, learning and outline. Evolutionary computation procedures are adaptable and powerful.

Though this area has seen its first people introducing various methods across the globe and calling it by different names like ‘Evolutionary Programming’ by Lawrence J Fogel of US in 1960 [7], ‘Genetic Algorithm’ by John Henry Holland in 1970s [8], ‘Evolution Strategies’ by Ingo Rechenberg and Hans-Paul Schwefel of Germany [8]; it is not until the early nineties that these were recognized as the different forms of same technique called evolutionary computing. However, these three areas developed separately for around 15 years. Since the early nineties, nature inspired algorithms have been a significant part of evolutionary computation [9].

2.2. Swarm Intelligence

Swarm intelligence facilitates utilizing decentralized control and self-association. Specifically, the control concentrates on the aggregate practices that outcome from the nearby cooperation’s of the people with one another and with their surroundings. Cases of frameworks considered by swarm knowledge are provinces of ants and termites, schools of fish, groups of flying creatures, crowds of area creatures. Some human relics likewise fall into the space of swarm insight, outstandingly some multi-robot frameworks, furthermore, certain PC programs that are composed to handle advancement and information investigation issues.

Swarm intelligence has been further expanded and applied to many diverse fields including robots where it is known as swarm robotics, [5]. Some of the swarm intelligence techniques and systems are given below

2.2.1. Particle swarm optimization. Is an optimization technique where the solution is given in terms of a point or surface. The solution is represented in an n-dimensional plane. Hypothesis testing is done by plotting a seed and then looking for the desired results. Also, the particles are allowed to communicate using a communication system. The system evaluates the functionality and movement of the particles in specific time intervals against some testing criterion. Particles with better adoptability to the testing criteria are attracted to each other. This technique helps in maximizing the dominant behavior of particles

2.2.2. Artificial bee colony algorithm. Is an algorithm, which is based on the behavior of honey bees. It exploits the movements and ways in which honey bees fulfill their tasks. This algorithm works according to three main phases as employed bee, onlooker bee and scout bee [10]. In the first two phases, the bees exploit their surroundings for the solution. They explore the neighborhood. While in the third phase, selection of useful sources of food/solutions is done and the useless solutions are abandoned.

2.2.3. Differential evolution. This algorithm deals with genetics. It consists of search vectors to carry out the searching operation. It exploits the use of search agents, which fulfill the tasks of searching the patterns and genetics [11].

2.3. Ant Colony Optimization

Ant colony optimization is an optimization technique based on the behavior of the ant colony. It observes how the ants move and how they coordinate. It consists of an algorithm, which is developed based on the behavior of ant colony and a probabilistic approach is followed. Ants leave pheromones while moving for the tracing of the path. Similarly, simulated ants are used in the algorithm, which leave traces to identify the path. This algorithm is applied in many fields like image processing, best path discovery, etc. [10]

Marco Dorigo is the person who introduced Ant Colony Optimization in 1992 [10]. In the wider field of swarm intelligence, this is one of the most successful techniques. Inspired by the ants behavior in finding food and laying path from the colony to the food source, computational problems are also reduced to figuring out good paths through graphs by a probabilistic technique. Discrete optimization problems like the travelling salesman is one of the most successful application of Ant Colony Optimization.

In the real world, ants that move randomly (initially) leave pheromone trails on their return path to their colony. When other ants find that path, they would follow the trail instead of moving randomly. When they do so, these ants too leave pheromone trails along the path reinforcing the trail.

Over time, the pheromone trail evaporates. This means that if the ants do not move along that path, it becomes weaker and less attractive for the other ants to follow. The more the time it takes for an ant to travel along the path, the lesser would be its pheromone trail strength along the path. So, by comparison, a path that is shorter between the food source and the colony would have greater pheromone density. Hence an ant that finds a good path leads all others towards that path.

In terms of optimization, the pheromone evaporation avoids the local optima convergence, which is definitely a significant advantage. Had that not been the case, the first chosen path by the ants would be more likely to be followed by the other ants, restricting the exploration of solution search space. The very idea of Ant Colony Optimization algorithm is to simulate this behavior of ants, which starts with random initialization of ants that walk around the search space graph to find a solution to the optimization problem. Figure 2 shows the pseudocode of ACO algorithm.

```
1. Initialization
2. An initial pheromone value is assigned to every edge of the search space graph and
   randomly an ant(s) is located in that search space
3. Looping through population
   a. Probabilistic transition: Move ant over the solution space accordingly to a given
      probabilistic transition rule
   b. Fitness evaluation: Evaluate the goodness of fit for the solution provided by the ant
   c. Pheromone update: Reinforce pheromone level of the edges for good solutions and
      reduce (evaporation) for those with not-so-good solutions
4. Repeat step 2 till the criteria of convergence is met.
```

Figure 2: Pseudocode of ACO Algorithm

2.4. Image Segmentation

Image Segmentation is the division of an image into meaningful structures. Image segmentation, is often an essential step in image analysis, object representation, visualization, and many other image processing tasks.

Perfect image segmentation, i.e., each pixel is assigned to the correct object segment– is a goal that cannot usually be achieved. Indeed, because of the way a digital image is acquired, this may be impossible, since a pixel may straddle the “real” boundary of objects such that it partially belongs to two (or even more) objects. Most methods presented here –indeed most current segmentation methods– only attempt to assign a pixel to a single segment, which is an approach that is more than adequate for most applications. Methods that assign a segment probability distribution to each pixel are called probabilistic. This class of methods is theoretically more accurate, and applications where a probabilistic approach is the only approach accurate enough for specific object measurements can easily be named. However, probabilistic techniques add considerable complexity to segmentation- both in the sense of concept and implementation – as such are still little used.

Perfect image segmentation is also often not reached because of the occurrence of over-segmentation or under-segmentation. In the first case, pixels belonging to the same object are classified as belonging to different segments. A single object may be represented by two or more segments. Generally, a "good" complete segmentation must satisfy the following criteria [12]:

1. All pixels have to be assigned to regions.
2. Each pixel has to belong to a single region only.
3. Each region is a connected set of pixels.
4. Each region has to be uniform with respect to a given predicate.
5. Any merged pair of adjacent regions has to be non-uniform.

A great variety of segmentation methods has been proposed in the past decades, and some categorization is necessary to present the methods properly here. A disjoint categorization does not seem to be possible though, because even two very different segmentation approaches may share properties that defy singular categorization.

Broadly, segmentation techniques can be classified into these categories [12], however in practicality hybrids of these methods can also exists in specific algorithms. Histogram thresholding and slicing techniques are used to segment the image. They may be applied directly to an image, but can also be combined with pre and post-processing techniques.

2.4.1. Edge based segmentation. With this technique, detected edges in an image are assumed to represent object boundaries, and used to identify these objects.

2.4.2. Region based segmentation. Where an edge based technique may attempt to find the object boundaries and then locate the object itself by filling them in, a region based technique takes the opposite approach, e.g., by starting in the middle of an object and then “growing” outward until it meets the object boundaries.

2.4.3. Clustering techniques. Although clustering is sometimes used as a synonym for segmentation techniques, we use it here to denote techniques that are primarily used in exploratory data analysis of high-dimensional measurement patterns. In this context, clustering methods attempt to group together patterns that are similar in some sense. This goal is very similar to what we are attempting to do when we segment an image, and indeed some clustering techniques can readily be applied to image segmentation.

2.5. Thresholding Technique

Thresholding is probably the most frequently used technique to segment an image. The thresholding operation is a gray value remapping operation g defined by equation (2.1),

$$g(v) = \begin{cases} 0 & \text{if} \quad v < t \\ 1 & \text{if} \quad v \geqslant t, \end{cases} \tag{2.1}$$

Where ‘v’ represents a gray value, and t is the threshold value. Thresholding maps a grey-valued Image to a binary image. After the thresholding operation, the image has been segmented into two segments, identified by the pixel values 0 and 1, respectively. If we have an image which contains bright objects on a dark background, thresholding can be used to segment the image.

Generally, the non-contextual thresholding may involve two or more thresholds as well as produce more than two types of regions such that ranges of input image signals related to each region type are separated with thresholds. The question of thresholding is how to automatically determine the threshold value. There are many kinds of threshold detection methods like P-tile thresholding, Optimal thresholding, Mixture modelling, adaptive thresholding etc., Since in many types of images the gray values of objects are very different from the background value, thresholding is often a well-suited method to segment an image into objects and background. If the objects are not overlapping, then we can create a separate segment from each object by running a labeling algorithm on the thresholded binary image, thus assigning a unique pixel value to each object.

When several desired segments in an image can be distinguished by their gray values, threshold technique can be extended to use multiple thresholds to segment an image into more than two segments: all pixels with a value smaller than the first threshold are assigned to segment 0, all pixels with values between the first and second threshold are assigned to segment 1, all pixels with values between the second and third threshold are assigned to segment 2, etc. If n thresholds (t1, t2, ..., tn) are used, then

$$g(v) = \begin{cases} 0 & \text{if} \quad v < t_1 \\ 1 & \text{if} \quad t_1 \leqslant v < t_2 \\ 2 & \text{if} \quad t_2 \leqslant v < t_3 \\ \vdots & \vdots \quad \vdots \\ n & \text{if} \quad t_n \leqslant v. \end{cases} \tag{2.2}$$

2.5.1. Threshold selection. Many methods exist to find a suitable threshold for segmentation. The simplest method is the interactive selection of a threshold by the user, possibly with the aid of the image histogram - a method that is usually accompanied by a graphical tool which lets the user immediately assess the result of a certain choice of threshold. Automatic methods often make us of the image histogram to find a suitable threshold.

2.5.2. Multilevel thresholding. Multi-level of thresholding produces multiple threshold values, which in turn is supposed to detect more number of edges. In this paper, we implemented multilevel thresholding proposed in [3]. The algorithm is:

1) Detect threshold value using Shannon entropy method.

2) The histogram H of image with pixel values (0,1,2,…,255) is split by t1 into two parts, H1 pixel values (0,1,2,…, t1) and H2 with (t1+1,…,255). See Figure 14

3) Apply Threshold procedure with H1 to find the threshold values (t2). Then apply it with H2 to find the threshold values (t3).

4) Create binary matrix A, using the three threshold values t1, t2 and t3 according to the condition, For all 1< j< m, and 1< i< n do: IF ((f(i,j)>= t2) and (f(i,j)< t1)) or f(i,j)>= t3) Then A(i,j)=1 else A(i,j) = 0.

5) Applying EdgeDetection procedure with A matrix to obtain the edge detection image g.

2.6. Edge Detection

Edge detection is the mathematical modeling of images. The models are used to identify the sharp variations in the images with respect to the brightness of the image. It investigates the abnormal or sudden changes in brightness, which is exploited to form a curved line, generally known as edges. Edge detection is an important processing paradigm for image processing, machine learning, and computer vision, and has an important role in feature detection and extraction [12].

The identification and differentiation of edges within an image helps to recognize changes. These abrupt changes in brightness of the image are related to depth, surface orientation, material properties and scene illumination variations.

Generally, applying edge detectors on an image helps to identify sharp changes as edges. These edges are not usually connected together to form a continuous curve. The edge detection technique reduces a data set to a small size for efficient processing and feature manipulations. A successful edge detection cycle results in efficient information extraction from the data and easier information handling. The simple processing techniques bring excellent results.

In edge detection, three-dimensional real-world scenes are represented as two-dimensional data sets. Edge extraction algorithms are applied to these two-dimensional data sets. These algorithms are either viewpoint dependent or independent. Viewpoint independent algorithms are dependent on the characteristics of the image itself, while viewpoint dependent algorithms are affected by changes in the viewpoint of the image.

3. ACO ALGORITHM

Numerous techniques have been executed throughout the years for edge detection among which is utilizing the Ant Colony Optimization calculation to distinguish the edge in a picture. This undertaking actualizes the ACO calculation for Image edge discovery utilizing Mat lab. A definitive objective of this task is to demonstrate the effectiveness of ACO to detect edges and its optimization. It requires a picture as an input and produces the resultant picture. The calculation alludes to the iterative specific system to discover the limit level of a picture. This iterative method for picking a limit was produced by Rider and Calvary. It changes over the intensity image to a binary image [13]. The ACO calculation parameters can be changed in accordance with show signs of improvement resultant picture. Then again, once in a while, the calculation's execution lessens relying upon gradually better performing PCs.

There has additionally been some exploration before to apply the Genetic Algorithm alongside the Ant Colony Algorithm to discover the image edge data. The normal rate of the population answers that are near the ideal results is low, and henceforth, the answer might not merge to the absolute ideal answer. This is the situation that the ants' calculation has a low populating dispersal and the answer's development meet rapidly, however, the optimum answer is approached at high speed. The case of these specialists is that their methodology builds the normal velocity to locate the ideal answer by applying the ants' pheromone data to the hybrid capacity [14].

This methodology is excluded in this paper and will be taken as a task for further research. Since the methodology followed in this paper considers the sharp intensity changes made in the picture for distinguishing the edge, the loss of vital edges in the picture is decreased 28 when contrasted with ordinary strategies. The components considered in the edge identification procedure are Edge Orientation and Edge Structure. Edge Orientation refers to the direction in which the edges are searched for by the algorithm by the variation in intensity levels. There are numerous strategies to identify these progressions, for example, changes in intensity levels between the face and the hair in a picture.

In this area, the execution points of interest and the setup methodology are specified in subtle portions alongside the variations done to the research in [4]. The fundamental objectives of this paper are (1) to study the results produced by applying multiple thresholding technique on ACO algorithm for edge detection. (2) To study if the accuracy increases when multiple thresholding technique and ACO are combined. Accuracy is defined in terms of number of edges determined.

3.1. Methodology

In this section, the methodology of the implementation is discussed. There are two implementations of the algorithm performed. First of all an implementation is performed, which implements the logic defined as in [4]. Later on, the implementation was enhanced and vectorized [15], and multiple threshold value is applied to study the results of the algorithm. The initial implementation is referred from the research paper [4], and later improved. The primary undertaking of the improvement procedure was to choose a language for programming. Matlab was chosen over other programming languages in the light of its capability and simplicity as well as the already implemented ACO image segmentation code. It is a straightforward methodology for numeric processing and its capacity to picture the outcomes in a useful and proficient way. Matlab has been demonstrated as an effective methodology for image processing. The edge choice in light of the ACO calculation is demonstrated as an efficient way to extract edges from an image.

Image edge detection can be considered as a problem of identifying the pixels in an image, which relate to edges. A 2-D image can be represented as a two dimensional matrix in which each element is considered as the pixel (Figure 3).

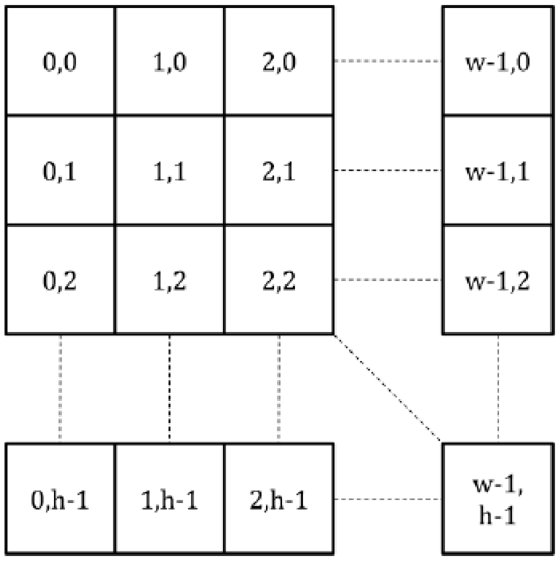


Figure 3: Representation of 2-Dimensional Image

The elements of the graphs in above figure are considered to be the pixels of an image. Only the adjacent pixels are connected in a graph. In above figure, the image is represented as a construction graph using an 8-connectivity pixel configuration.

In an 8-connectivity configuration, each pixel is connected to every other pixel that touches one of its corners or edges. Ants move on the graph from one pixel to another pixel using these connections. An ant cannot move from its current pixel to a disconnected pixel. This means they can only visit the adjacent pixel. Figure 4 and Figure 5 represents 8 connectivity pixel configuration and representation respectively.

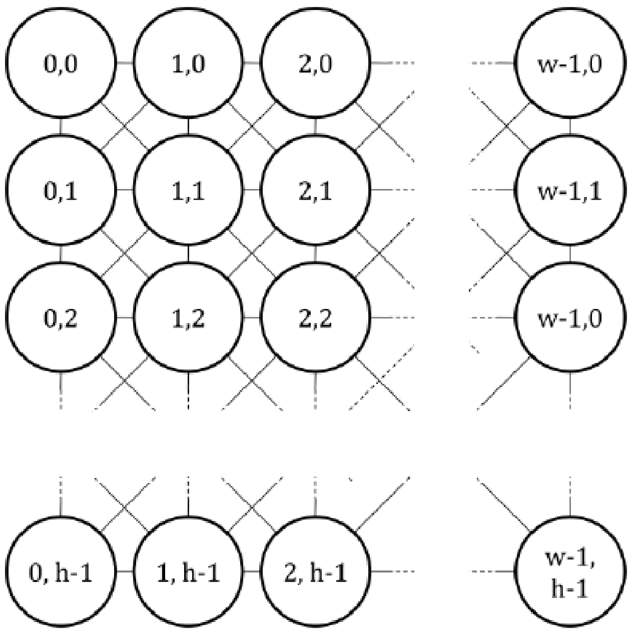
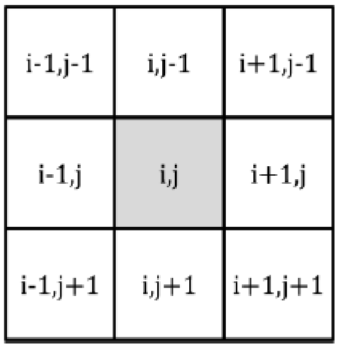


Figure 4: 8 Connectivity pixel Configuration



i-1,j-1	i,j-1	i+1,j-1
i-1,j	i,j	i+1,j
i-1,j+1	i,j+1	i+1,j+1

Figure 5: 8 Connectivity Representation

Artificial ants traverse over the image and move from one pixel to another. The movement of the ants depends on variation in the intensity values of each the pixel. The aim of the ants' movement is to construct a final pheromone matrix that represents the edge information. Each element in the pheromone matrix represents a pixel in the image. There are 3 important processes that are listed in the algorithm, first is the initialization and the second is the construction and update process, which is iterative where the aim is to build a final pheromone matrix. And the third and final step is the decision process where it is decided if the pixel or an element of the matrix is an edge or not.

3.2. Algorithm Process

3.2.1. Initialization. K number of ants are introduced on Image I of size M1 X M2 where M1 is the length, M2 is the picture's width. All the K ants are proliferated on the 2-D picture randomly such that at most one ant is on every pixel. Each pixel in the picture is a hub and the introductory estimation of the pheromone matrix τ (0) is set to a constant value τ_{init}, which is a small but non-zero value. The parameters α and β are introduced where α refers to the weighing factor for the pheromone information on each pixel, and β refers to the weighing factor for the heuristic information.

3.2.2. Construction. At every building step, one of the K ants takes L steps on the image I. The insect a_k moves from hub (l, m) to its neighboring hub (i, j) as per probabilistic transition matrix characterized as,

$$p_{(l,m),(i,j)}^{(n)} = \frac{\left(\tau_{i,j}^{(n-1)}\right)^{\alpha} (\eta_{i,j})^{\beta}}{\sum_{(i,j)\in\Omega_{(l,m)}} \left(\tau_{i,j}^{(n-1)}\right)^{\alpha} (\eta_{i,j})^{\beta}}, \quad (3.1)$$

Where,

$T_{i,j}(n-1)$ is the pheromone information value of the edge between node i and node j

$\eta_{i,j}$ is the heuristic information of moving from node i to node j

Ω_i is the neighbor node of ant a_k

α is the constant representing influence of pheromone information

β is the constant representing influence of heuristic information

Ω_l, m is the neighbor node of (l, m), i.e. it is all the pixels that can found in the 8-neighborhood of the pixel (l, m).

$\eta_{(i, j)}$ refers to the heuristic information of the node, which is determined as

$\eta(i,j)$=1$ZVC(Ii,j)$

$$\eta_{(i,j)} = \frac{1}{Z} V_C(I_{i,j}) \tag{3.2}$$

Where, Z is the normalization factor, which is determined as ΣΣVc (Ii, =1:M2i=1:M1)

$$\sum_{i=1:M1} \sum_{j=1:M2} Vc\ (I_{i,j}) \tag{3.3}$$

Where, Vc($I_{i,\,j}$) represents the variation of image's intensity value of the pixel (i, j) of image I. The variation of the image's intensity values depends on c, called clique which a group of pixels which are similar in some form. Vc(Ii, j) is the function that is formed by these group of pixels

For the pixel Ii, j the function Vc(Ii, j) is determined

$$\begin{aligned} V_c(I_{i,j}) = f\,(& |I_{i-2,j-1} - I_{i+2,j+1}| + |I_{i-2,j+1} - I_{i+2,j-1}| \\ & +|I_{i-1,j-2} - I_{i+1,j+2}| + |I_{i-1,j-1} - I_{i+1,j+1}| \\ & +|I_{i-1,j} - I_{i+1,j}| + |I_{i-1,j+1} - I_{i-1,j-1}| \\ & +|I_{i-1,j+2} - I_{i-1,j-2}| + |I_{i,j-1} - I_{i,j+1}|)\,. \ (\end{aligned} \tag{3.4}$$

This function determines the small angle turns dominate sharp turns in an image. Therefore, it causes a tendency for an ant to move in the forward direction. In order to determine f (.) in Equation (3.4) for different shapes, the following four functions are considered in the paper.

$$f(x) = \lambda x, \quad \text{for} \quad x \geq 0, \tag{3.5}$$

$$f(x) = \lambda x^2, \quad \text{for} \quad x \geq 0, \tag{3.6}$$

$$f(x) = \begin{cases} \sin\left(\frac{\pi x}{2\lambda}\right) & 0 \leq x \leq \lambda; \\ 0 & \text{else.} \end{cases} \tag{3.7}$$

$$f(x) = \begin{cases} \frac{\pi x \sin(\frac{\pi x}{\lambda})}{\lambda} & 0 \leq x \leq \lambda; \\ 0 & \text{else.} \end{cases} \tag{3.8}$$

Where the parameter λ takes different values to produce different shapes. Figure 8 represents the various outcomes of shapes for different values of λ chosen at random.

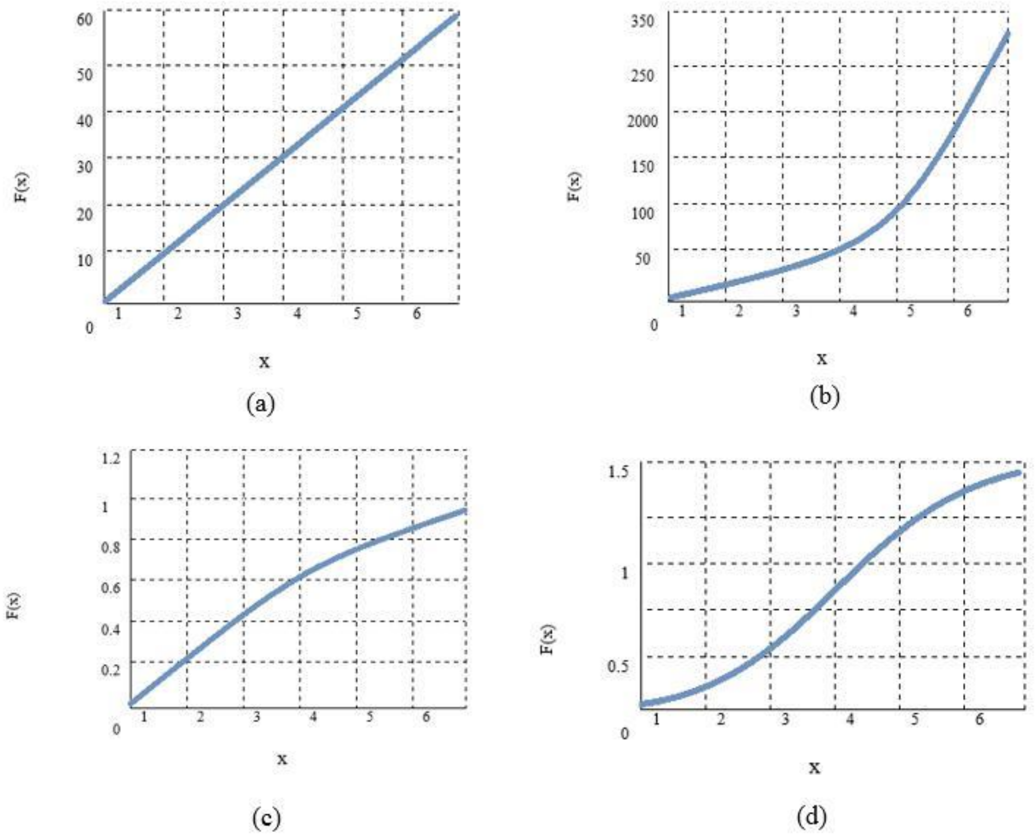


Figure 6: Graphs for various functions with λ = 10: a) Function defined in 4.5 b) Function defined in 4.6 c) Function defined in 4.7 d) Function defined in 4.8

3.2.3 Update process. In the paper, two update operations are performed for updating the pheromone matrix. The pheromone matrix is updated after each movement of an ant. After the kth ant in the nth construction step moves, the updated pheromone matrix is

$$Ti,j(n-1)=\{\ (1-\rho).Ti,j(n-1)+\rho.\Delta\ \tau i,j(k)\ ,\ if\ (i,j)\ is\ visited\ by\ the\ Kth\ ant;\ Ti,j(n-1)\ ,\ otherwise. \tag{3.9}$$

$$\Delta\ \tau i,\ (k) = (i,j) \tag{3.10}$$

Where ρ is the evaporation rate

The best trip is determined by the user based on the best path of an ant found in the current building step or the best path after considering all the paths from the start of the algorithm the ants have travelled, or a combination of both. In this paper, we consider the best path after each building step. The second update operation of the pheromone matrix is as below after all the ants have moved, i.e. is after each building step.

$$Tn = (1-\psi).Tn-1+\psi.Tn \tag{3.11}$$

Where,

Ψ is the pheromone decay coefficient

At this level the pheromone matrix is updated by considering the decay coefficient and the pheromone matrix constructed until now.

3.2.4 Decision process. A binary decision is made on the pheromone matrix to determine if each of its elements (pixels) are considered an edge or not. This decision is based on the application of a multi-level threshold on the final pheromone matrix Tn. T is considered to be the Threshold value which needs to be computed as defined in [16]. The algorithm used in this paper is as

1. The initial threshold t1 is set by computing using the *Threshold procedure based on Shannon entropy.*

2. The elements of the pheromone matrix are divided into two segments based on the threshold value t1 dividing Histogram of pixels H into H1 with *(0,1,2,…, t1)* and H_2 *(t1+1,…,255)* .

3) Apply Threshold procedure with H1 to find the threshold values (t2). Then apply it with H2 to find the threshold values (t3).

4) Create binary matrix A, using the three threshold values t1, t2 and t3 according to the condition, For all 1< j< m, and 1< i< n do: IF ((f(i,j)>= t2) and (f(i,j)< t1)) or f(i,j)>= t3) Then A(i,j)=1 else A(i,j) = 0.

5) Applying Edge Detection procedure with A matrix to obtain the edge detection image g.

Figure 7 represents the flow chart of ACO algorithm

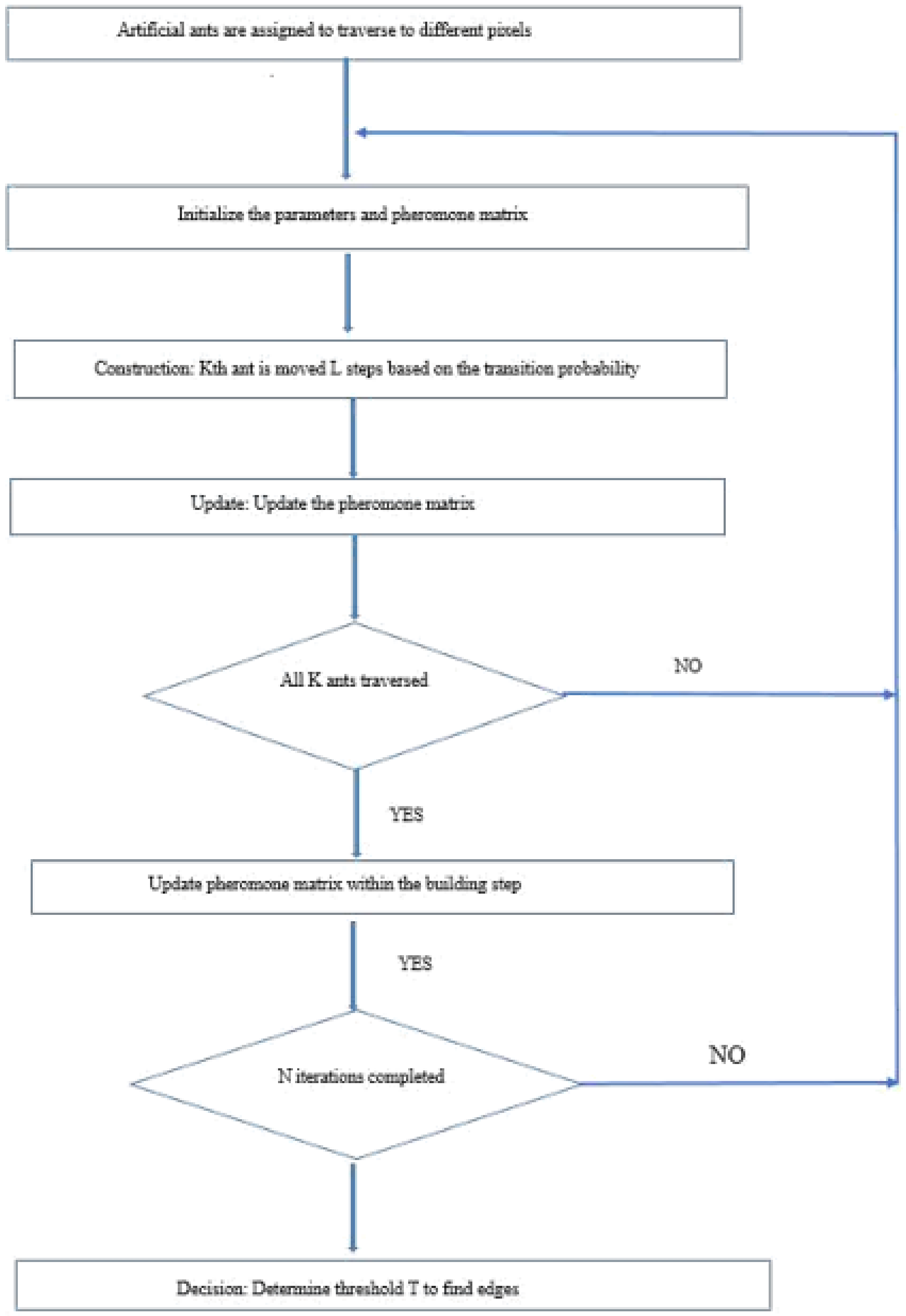
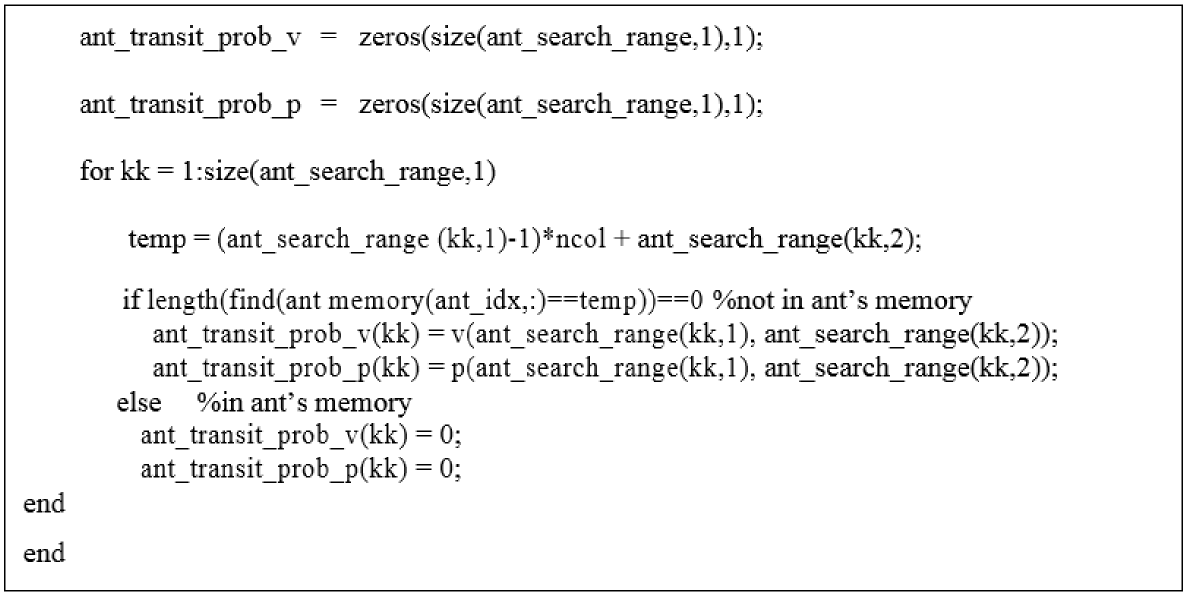

Artificial ants are assigned to traverse to different pixels
Initialize the parameters and pheromone matrix
Construction: Kth ant is moved L steps based on the transition probability
Update: Update the pheromone matrix
All K ants traversed
NO
YES
Update pheromone matrix within the building step
YES
N iterations completed
NO
Decision: Determine threshold T to find edges

Figure 7: Flow chart of ACO [16]

4. IMPLEMENTATION

The algorithm proposed in this paper has been implemented in MATLAB. The initial implementation is done as proposed by [4]. This implementation is enhanced using multilevel thresholding algorithm that was proposed by Mohamed et al. [1] to study the efficiency of algorithm. The code snippets in this chapter are a part of the implementation, which is explained in detail.

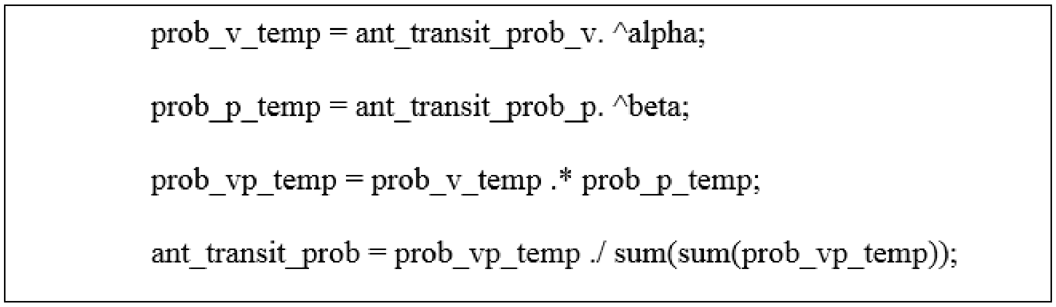
The script prompts the user to select the option of the processing greyscale image or the color image. The user is asked to select the input file, selecting a file in jpg, bmp or png format. The image is resized to 128x128 resolution and in the case of the color image, it is converted to a greyscale image and resized to 128x128. According to the algorithm, the intensity image is converted to a binary image. Therefore, the pixel values are converted to the range between 0 and 1. The number of ants is initialized based on the size of the image that is given as the input. Figure 8 represents the same



```
        ant_transit_prob_v   =   zeros(size(ant_search_range,1),1);

        ant_transit_prob_p   =   zeros(size(ant_search_range,1),1);

        for kk = 1:size(ant_search_range,1)

             temp = (ant_search_range (kk,1)-1)*ncol + ant_search_range(kk,2);

            if length(find(ant memory(ant_idx,:)==temp))==0 %not in ant's memory
                ant_transit_prob_v(kk) = v(ant_search_range(kk,1), ant_search_range(kk,2));
                ant_transit_prob_p(kk) = p(ant_search_range(kk,1), ant_search_range(kk,2));
            else     %in ant's memory
               ant_transit_prob_v(kk) = 0;
               ant_transit_prob_p(kk) = 0;
end

end
```

Figure 8: Implementation of probability transition matrix

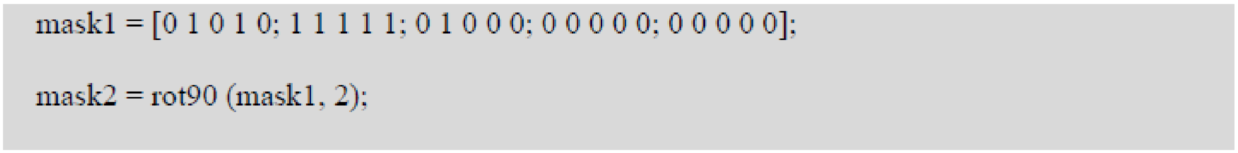
The optimization is performed with the help of vectorization of the code, which results in faster processing of the algorithm. The multiplication of the v and p vectors, which are defined in the transition probability, are parallelized and vectorized and hence the normalization operation of the ant transition probability is as shown in Figure 9.



```
prob_v_temp = ant_transit_prob_v. ^alpha;

prob_p_temp = ant_transit_prob_p. ^beta;

prob_vp_temp = prob_v_temp .* prob_p_temp;

ant_transit_prob = prob_vp_temp ./ sum(sum(prob_vp_temp));
```

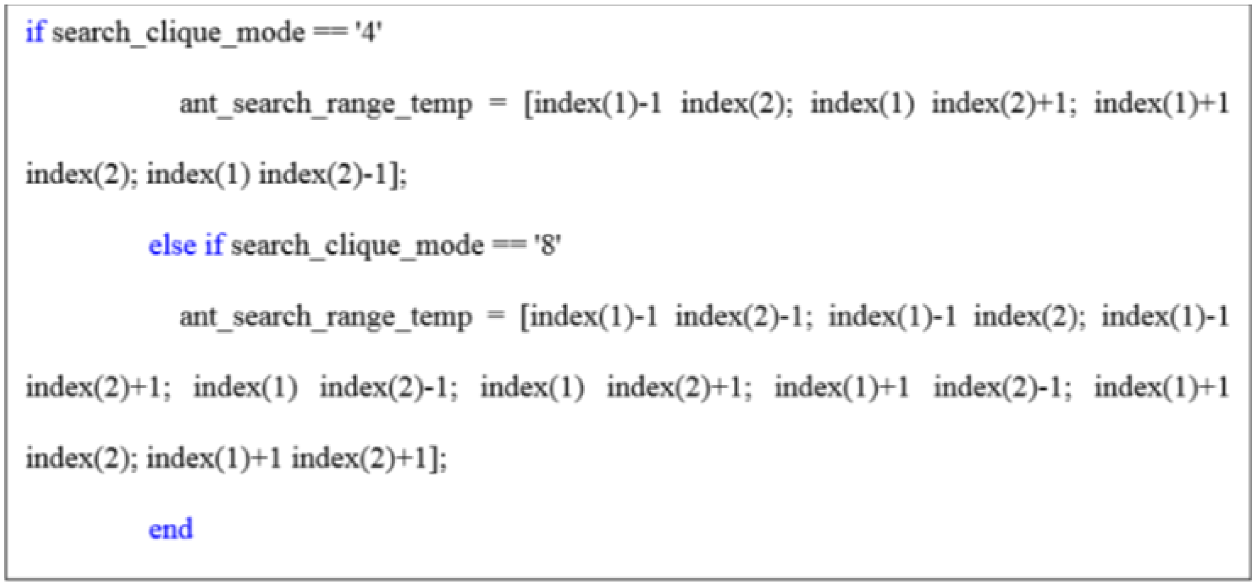
Figure 9: Normalization process of transition probability

The implementation has two masks. Masks are matrices that contain either 0’s or 1's, which help in deciding the suitable computation to perform to determine the values in a matrix. As can be easily analyzed that the selected elements can be divided into two groups, i.e., the upper side and lower side. Hence, these elements are defined by two masks; mask1 defines the upper side pixels/values and the mask2 defines the lower side:



```
mask1 = [0 1 0 1 0; 1 1 1 1 1; 0 1 0 0 0; 0 0 0 0 0; 0 0 0 0 0];

mask2 = rot90 (mask1, 2);
```

Now these two masks are used to select the corresponding pixel values. When we are looking for a 4- or 8-connected neighborhood, here instead of selecting a single pixel at a time, we accessed the whole window of the image with pixels. Figure 10 shows the implementation of the 4 or 8 connectivity of pixels.

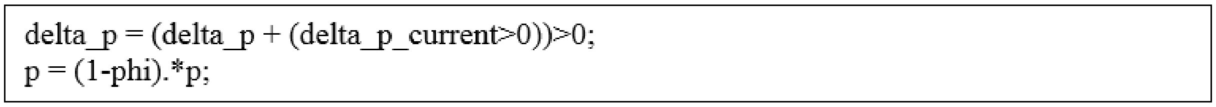


```
if search_clique_mode == '4'

            ant_search_range_temp = [index(1)-1 index(2); index(1) index(2)+1; index(1)+1
index(2); index(1) index(2)-1];

        else if search_clique_mode == '8'

            ant_search_range_temp = [index(1)-1 index(2)-1; index(1)-1 index(2); index(1)-1
index(2)+1; index(1) index(2)-1; index(1) index(2)+1; index(1)+1 index(2)-1; index(1)+1
index(2); index(1)+1 index(2)+1];

        end
```

Figure 10: Implementation for neighbor search for 4-connecity pixel and 8-connectivity pixel

The 4 kernel functions defined in Equations (4.5)-(4.8) are parallelized, which helps take the advantage of parallel processing and multicore architecture of the CPU or processing units.

The final update of the pheromone matrix is done within each construction step after all ants have traversed as appeared in Figure 11.



```
delta_p = (delta_p + (delta_p_current>0))>0;
p = (1-phi).*p;
```

Figure 11: Implementation step for updating final pheromone matrix

4.1. Multi threshold Implementation

The last step is to identify the edges from the final pheromone matrix. This is decided based on calculating the final threshold value T given in [1]. The implementation of the proposed approach to calculate multiple threshold values is applied in this part of the overall implementation.

In this paper, we combined the algorithm proposed by Amira and Ashmore with the GSO Matlab implementation by implementing the multilevel thresholding in order to study the resultant edge image. Initially, the pheromone matrix is passed to the class “func_seperate_class” to begin with the multi threshold process as:

T= func_seperate_class(p);

Since the pheromone matrix is a floating matrix that is normalized, we initially considered a matrix of 256 by 3 matrix of zeroes. Later the matrix is initialized.

The multi threshold algorithm uses histogram values whereas the pheromone matrix is normalized to floating values. Thus, we mapped the pheromone matrix values to their corresponding histogram values in order to calculate the histogram counts of the image as shown in Figure 12.

```
P= zeros(256,3);
for ii=1:256
    p(ii,1) = ii-1;
end;
[p(:,2), N] = hist(I, 256);
p(p(:,2)==0, :) = [];
N(p(:,2)==0) = [] ;
```

Figure 12: Mapping Pheromone Matrix to Histogram values

Furthermore, continuing the process as per proposed algorithm we used the Shannon entropy method to find the threshold value t1 as shown in Figure 13. The Shannon entropy provides a way to estimate the average number of bits needed to encode a string of symbols, based on the frequency of the symbols (4.1).

The Shannon entropy formula mentioned in Equation (4.1) is as follows:

$$H(X) = -\sum (P_i \log_2 P_i) \ldots \text{ where } i = 0 \text{ to } N \qquad (4.1)$$

Where P_i is the probability

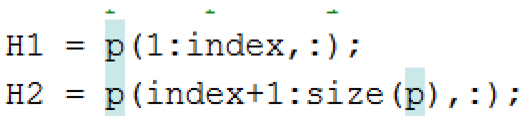
```
PA = sum(p(1:t,3));
PB = 1-PA;
p1 = p(1:t,3)./PA; % p1 is i
        probability in PA
p2 = p(t+1:M1,3)./PB; % p2 is i prob.
        in PB
Sa = -sum(p1.*log2(p1));
Sb = -sum(p2.*log2(p2));
```

Figure 13: Calculation of threshold values

Since the entropy equation (5.1) calculates the probability for a single dimension, adapting this equation we applied it to two dimensions thus calculating the probability of Xi and Yi thus accessing the whole matrix.

Using t1 threshold value, the pheromone matrix is divided into two segments H1 = 1… t1 and

H2 = t1+1… 256:



```
H1 = p(1:index,:);
H2 = p(index+1:size(p),:);
```

Figure 14: Histogram of image

Once the image is split into H1 and H2, the threshold t2 for H1 part is calculated. Similarly t2 is calculated for the other part of the histogram. Once the threshold values are created, using these threshold values a t1,t2,t3 binary matrix is created based on the condition that *1< j< m, and 1< i< n do: IF ((f(i,j)>= t2) | (f(i,j)>t3)) and f(i,j)<= t1) Then A(i,j)=1 else A(i,j) = 0.*

The code snippet below demonstrates the same, and writes the binary image v created by the binary matrix.

```
          Imwrite((uint8(abs(p>=T(3) | P>1=T(2) &
p<=T(1).*255)), gray(256), ['FROM_KERNEL_'
        num2str(nMethod)  '.bmp'], 'bmp');
```

Figure 15: Condition used to create binary matrix

5. RESULTS AND DISCUSSIONS

The proposed algorithm is implemented and executed in MATLAB. The resultant output, is a binary image in which the white pigmented areas determine the edges in order to test the method proposed in this paper, we considered both grayscale image with different sizes and also color images of different sizes.

Experiments were conducted to implement the thresholding algorithm proposed in [3] using MATLAB and study the results produced by the algorithm and compare the threshold values for various images against Mat lab standard “multithresh” function.

In this paper, the task has been to improve the performance in terms of number of edges by using test images. Table 1 gives the details of the parameters initialized and the other parameters used in the equations during the Construction, Update and Building Process phases.

Table 1: ACO Parameters description and their initialization [16]

Parameters	**Values initialized**	**Description**
K	$\lfloor\sqrt{M1XM2}\rfloor$	Total number of ants is equivalent to image size
τ_{init}	0.001	Initial value of each element of pheromone matrix
α	1	Weighing factor of pheromone value
β	0.1	Weighing factor of heuristic information
Ω	8-Connectivity	The permissible ants movement range
λ	1	Adjusting factor of the functions 3.5 to 3.8
ρ	0.1	Evaporation rate
L	40	Total number of ant movement steps within construction step
Ψ	0.05	Pheromone decay coefficient
ϵ	0.1	User defined tolerance value in the decision process
N	4	Total number of construction steps

Table 2: Values used in this paper to run the experiments

Parameters	Values Initialized	Description
α	1	Weighing factor of the pheromone value in equation
β	0.1	Weighing factor of the heuristic information
Ψ	0.001	Pheromone decay coefficient in
T	1,2,3	Threshold values considered to compare the results when used single threshold, two and three thresholds

An intensity image is considered to be the input image; the script gives a choice to enter either a color or greyscale image. While conducting experiments we used both gray scale and color images. The initial run is done by the parameters set by the previous author of the paper as shown in Table 1, later based on the results, further experiments are carried out with varying α, β, Ψ parameters, which are the weighing factor of the pheromone value, weighing factor of heuristic information, and pheromone decay coefficient, respectively. Also, we tested with different number of thresholds to study their effect on the images. We later compared these threshold values to built-in function provided by Mat lab named “multithresh”. This method is based on Otsu’s method of thresholding. Otsu’s method named after Nobuyuki Otsu, is used to automatically perform clustering-based image thresholding, or, the reduction of a gray level image to a binary image [13]. The algorithm assumes that the image contains two classes of pixels following a bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently (because the sum of pairwise squared distances is constant), so that their inter-class variance is maximal [13].

Otsu's method typically is single dimensional. An extension of original method to multi-level thresholding is referred to as the multi otsu method on which matlab's built-in function multi-threshold is based.

The function is denominated as:

thresh = multithresh (<Image I>, <num of desired thresholds) (6.1)

Where image *I* is the input image for which threshold values is calculated; *num of desired* thresholds is the number of threshold values that needs to be calculated, and the threshold values returned by multi threshold are stored in variable thresh.

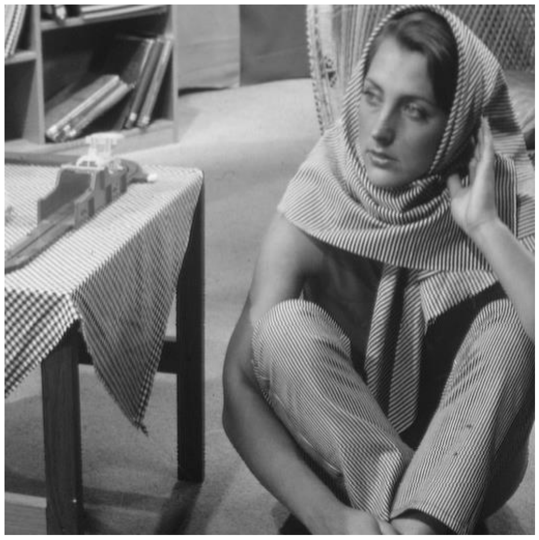


Figure 16: Lady Gray scale image

Observations are made on the output images generated by the images using 3 threshold values created by the Shannon function. Experiments involve comparing the threshold values produced by the stochastic algorithm against the multi threshold function.

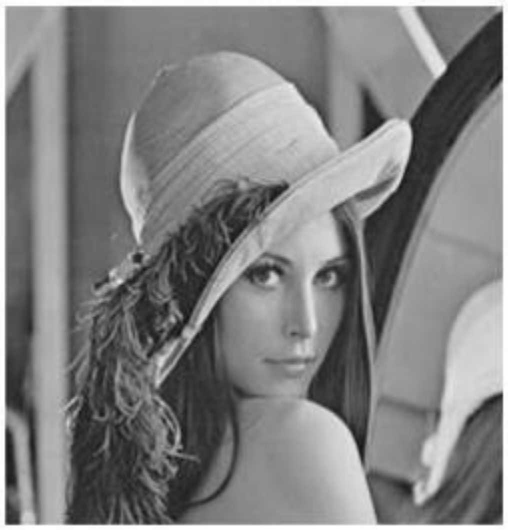


Figure 17: Lena Gray scale image

Experiments conducted included comparing the multi threshold values produced by the approach against the multi-threshold function in the Matlab. For each image, there are three threshold values produced T1, T2, T3 for each kernel function thus producing a total of 12 threshold values.

5.1. Effect of pheromone decay coefficient Ψ

Table 3 shows the threshold values for the Figures 16 and 17 when experiments are run for original set parameter as shown in Table 1.

Table 3: Threshold Values for various images

					Threshold values by multithresh		
Name	**Kernel Function**	**T1**	**T2**	**T3**	**T1**	**T2**	**T3**
Lena-GrayScaleImage	Func: 4.5	20	54	102	82	127	170
	Func 4.6	6	26	70			
	Func 4.7	11	35	81			
	Func 4.8	14	45	100			
Lady Gray Scale Image	Func: 4.5	16	53	105	77	128	174
	Func 4.6	8	31	58			
	Func 4.7	22	63	120			
	Func 4.8	7	25	66			

In Table 3, we can notice that the threshold values produced using Equation (3.5) and (3.7) are close to the threshold values produced by the Mat lab function. We then changed the standard values for the pheromone coefficient to 0.001 to see if the threshold values changed. We observed that changing the values changed the T3 values significantly keeping T2 and T1 values intact. Table 4 shows the difference in threshold values.

Table 4: Threshold Values when phi = 0.001 for various images

					Threshold values by multithresh		
Name	**Kernel Function**	**T1**	**T2**	**T3**	**T1**	**T2**	**T3**
Lena-GrayScaleImage	Func: 4.5	31	78	120	80	127	170
	Func 4.6	15	40	81			
	Func 4.7	30	80	128			
	Func 4.8	14	45	100			
Lady Gray Scale Image	Func: 4.5	16	60	106	77	127	174
	Func 4.6	14	42	66			
	Func 4.7	14	42	67			
	Func 4.8	15	60	103			

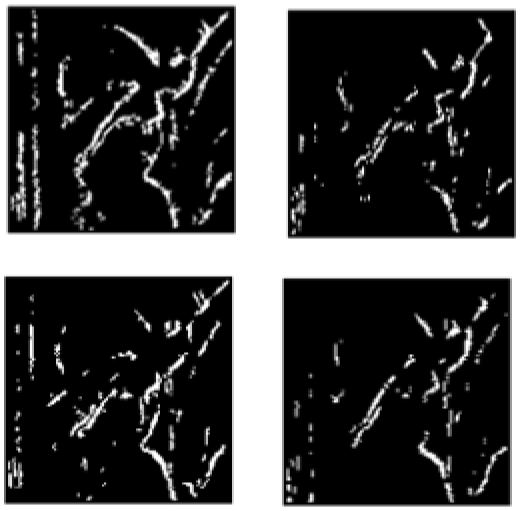


Figure 18: Various extracted edge information of grey image Lena a) image output using equation (3.5); b) image output using equation (3.6); c) image output using equation (3.7); d) image output using equation (3.8);

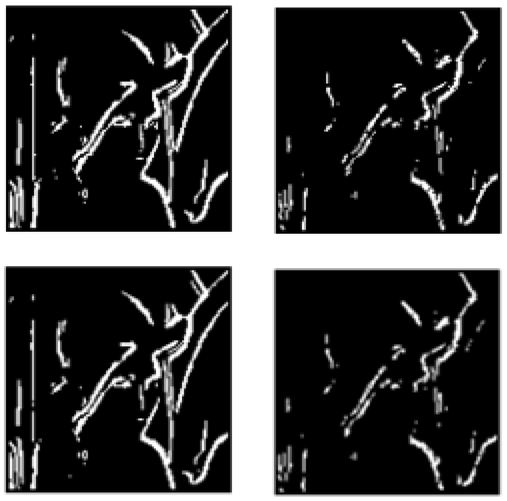


Figure 19: Various extracted edge information of Lena image for parameters in table2 – Multilevel thresholding a) image output using equation (3.5); b) image output using equation (3.6); c) image output using equation (3.7); d) image output using equation (3.8);

Changing the pheromone decay coefficient also increases the number of edges detected of the image. Figure 18 and Figure 19 shows the resultant binary images produced by the experiment runs for parameters shown in Table 1 and Table 2 for Lena gray scale image.

Figure 20: Lady Color Image

Figure 21: Ship Greyscale Image

From Figure 19 we can observe that decreasing the pheromone decay coefficient increased the efficiency. More defined edges can be found in Figure 19 and also the image displays finer details of the "hat" when compared to Figure 18. Thus, it definitely increases the efficiency.

Figures 20 and figure 21 are few other images that were used to run the experiments, and from the Figures 23, and Figure 25 we can see that decreasing the pheromone coefficient increased the efficiency significantly, when compared to Figures 22 and 24 respectively.

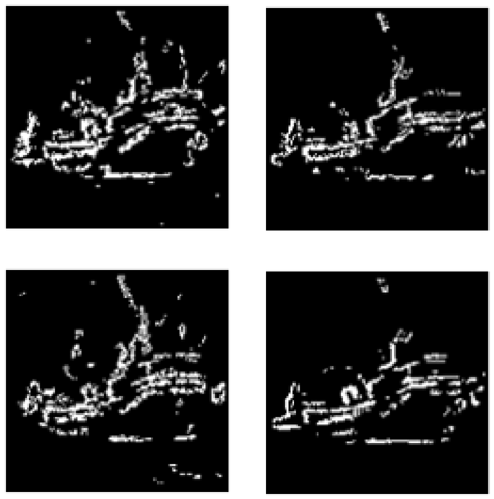


Figure 22: Various extracted edge information of Ship grayscale image - single threshold a) image output using equation (3.5); b) image output using equation (3.6); c) image output using equation (3.7); d) image output using equation (3.8);

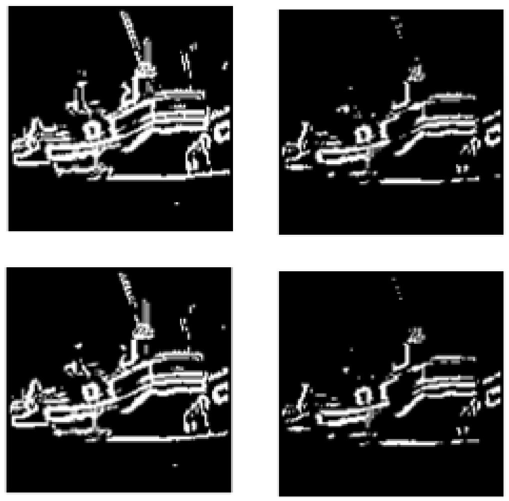


Figure 23: Various extracted edge information of Ship Greyscale using parameters in table2-Multiple threshold a) image output using equation (3.5); b) image output using equation (3.6); c) image output using equation (3.7); d) image output using equation (3.8);

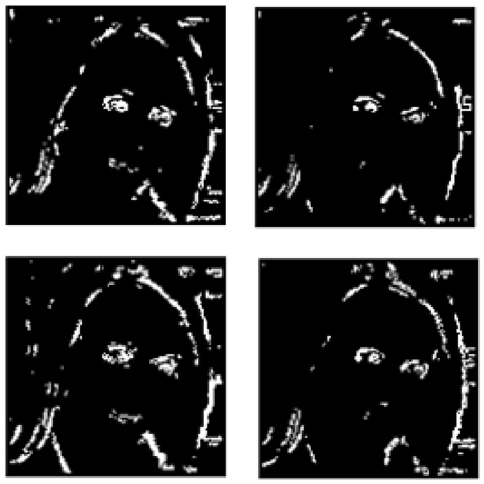


Figure 24:Various extracted edge information of lady color image - Single threshold a) image output using equation (3.5); b) image output using equation (3.6); c) image output using equation (3.7); d) image output using equation (3.8);

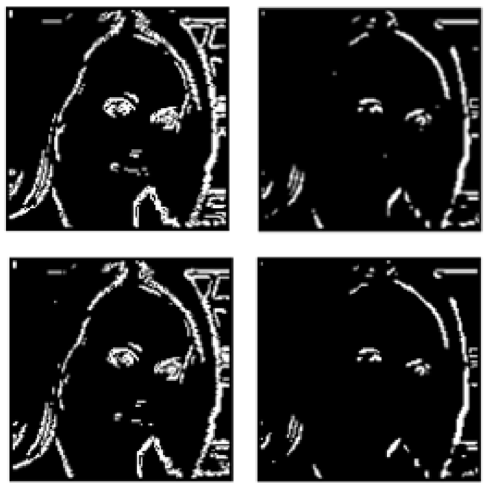


Figure 25:Various extracted edge information of lady color using parameters in table2- Multiple threshold a) image output using equation (3.5); b) image output using equation (3.6); c) image output using equation (3.7); d) image output using equation (3.8);

5.2. Effect of threshold levels values

In order to understand the effect of the levels of threshold, we run the experiments initially with 3 thresholds as per the proposed algorithm by Ashore [3]. Results for the same is discussed in Section 6.1. To further answer our research question RQ2 in Chapter 2, we carried the experiments applying a single, two levels and four levels of threshold to ship gray scale image (Figure 21).

Figure 26 displays the resultant binary image for the ship greyscale image when a single level of threshold is considered. As expected the resultant image shows a great loss of information in terms of edges. Also we compared the threshold value produced by the proposed method with threshold value produced by multi thresh function *102*. Threshold value for Equation (3.5), (3.6), (3.7), (3.8) were 76, 60, 51, and 79. As we can see, the threshold value when Equation (3.5) and (3.8) is closest to the multi thresh function.

Further, experiments were carried out with two threshold values T1 and T2. Threshold values for Equation (3.5), (3.6), (3.7), (3.8) are shown in Table 5. The threshold values found by the multithresh function were 87 and 144. Figure 26, 27 and 28 show the resultant images for single, two and four levels of thresholds.

Table 5: Threshold values when there are two levels of threshold

Equation	T1	T2
4.5	29	73
4.6	16	61
4.7	21	62
4.8	35	73

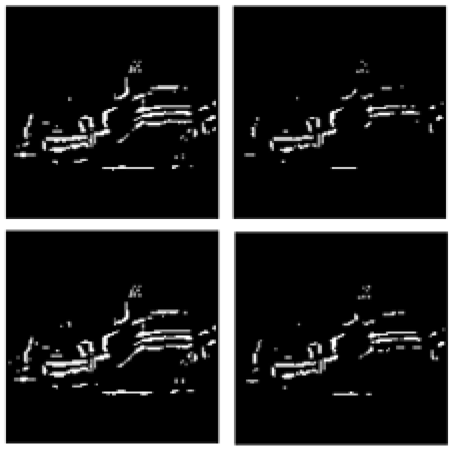


Figure 26: Binary Image Created for single level of threshold

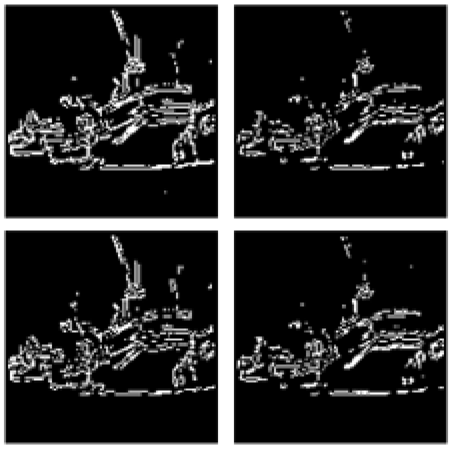


Figure 27: Binary Image Created for two level of threshold

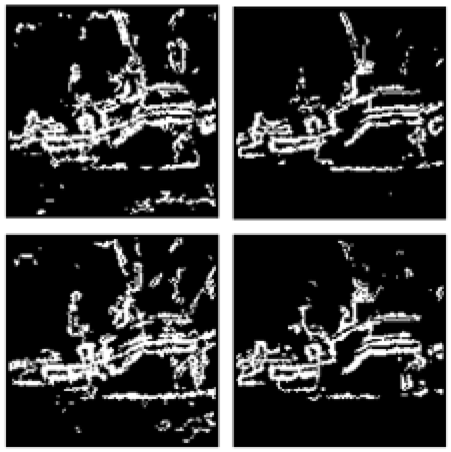


Figure 28: Binary Image Created for Four levels of threshold

According to the observations made, an improvement in the result can be achieved by analyzing and setting separate values for each RGB pixels. This would result in achieving more accurate values for RGB pixels. Also, when we used the edge detection procedure mentioned in [3], there might be chances of achieving higher performance when compared to using four kernel functions.

6. CONCLUSION AND FUTURE WORK

This research work involved the implementation of the Multi-level thresholding algorithm to be integrated with Ant colony optimization algorithm. The ACO algorithm was used to detect edges in both gray scale and color image where the decision is made using multiple threshold values as proposed in [3]. From the experiments conducted, it was observed that the results were best when kernel functions (3.5) and (3.7) were used.

The methodology applied to the image edge detection problem demonstrated its applicability to this application domain. The ACO calculation is utilized to take care of the image edge discovery issue. The intensity of the image is intact while converting it to a binary image.

We conducted two types of experiments. Initially in order to answer our first research question, experiments were conducted to implement the algorithm proposed by [3] by using MATLAB and study the results produced by the algorithm and compare the threshold values for various images against Matlab's multithresh function. From the resultant Figures 22, and 24 we can say that by decreasing the pheromone coefficient, there was an increase in the information found in terms of edges.

In order to answer the second research question, experiments were ran to understand the effect of different levels of thresholds. Different levels of thresholds single, two, and three, were used to understand and study the resultant images. From the Figures 26, 27, and 28, we can say that the efficiency increased as the levels of thresholding increased.

Also, there is a scope of improvement that can be done to increase the performance. As discussed earlier, setting different threshold values for RGB values for color images directly might increase the performance, which is a study for future.

7. REFERENCES

1. El-Sayed, M. A. (2013). Study of Edge Detection Based On 2D Entropy. I*JCSI International Journal of Computer Science Issues, Vol. 10, Issue 3, No 1, May 2013*
2. Vala, M. H. J., & Baxi, A. (2013). A review on Otsu image segmentation algorithm. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2*(2), pp-387.
3. Ashour, A. S., El-Sayed, M. A., Waheed, S. E., & Abdel-Khalek, S. (2014). New Method Based on Multi-Threshold of Edges Detection in Digital Images. *Editorial Preface*, *5*(2).
4. K. Ian (2002), "Shannon entropy", http://www.bearcave.com/misl/misl_tech/wavelets/compression/shannon.html
5. Kaur, N., & Kaur, R. (2011). A review on various methods of image thresholding. *International Journal on Computer Science and Engineering*, *3*(10), 3441.
6. Hornby, G. S., & Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial life*, *8*(3), 223-246.
7. Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial intelligence through simulated evolution. *Wiley, 1996.*
8. Eigen, M. (1973). *Ingo Rechenberg Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologishen Evolution* (pp. 33-37). mit einem Nachwort von Manfred Eigen, Friedrich Frommann Verlag, Struttgart-Bad Cannstatt.
9. Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing* (Vol. 53). Heidelberg: springer.
10. Kasam, S. K. (2016). Glowworm Swarm Optimization Algorithm for Multi-Threshold Image Segmentation. *Circulation*, *701*, 8888.

11. Nick Efford (2000), “Digital Image Processing: A Practical Introduction Using JavaTM”, *Pearson Education.*

12. Bremermann, H. J. (1962). Optimization through evolution and recombination. *Self-organizing systems*, *93*, 106.

13. Sun, C., Zhou, H., & Chen, L. (2012, May). Improved differential evolution algorithms. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on* (Vol. 3, pp. 142-145). IEEE.

14. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization (PSO). In *Proc. IEEE International Conference on Neural Networks, Perth, Australia* (pp. 1942-1948).

15. El-Sayed, M. A., Bahgat, S. F., & Abdel-Khalek, S. (2013). Novel approach of edges detection for digital images based on hybrid types of entropy. *Applied Mathematics & Information Sciences*, *7*(5), 1809.

16. Moparthi, R. (2016). Speed Optimized Implementation of Ant Colony Optimization Algorithm for Image Edge Detection. *Circulation*, *701*, 8888.