

**IMAGE EDGE DETECTION USING ANT COLONY OPTIMIZATION  
ALGORITHM**

---

A Thesis

Presented to the

Faculty of

San Diego State University

---

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Sunjna Kashyap

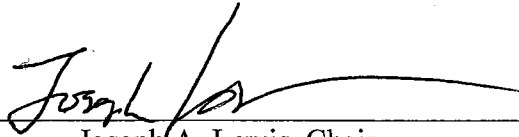
Summer 2012

**SAN DIEGO STATE UNIVERSITY**

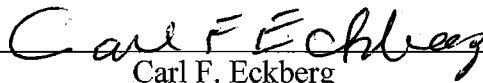
The Undersigned Faculty Committee Approves the

Thesis of Sunjna Kashyap:

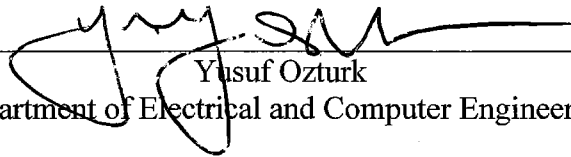
Image Edge Detection Using Any Colony Optimization Algorithm



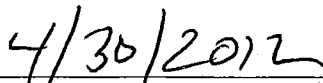
Joseph A. Lewis, Chair  
Department of Computer Science



Carl F. Eckberg  
Department of Computer Science



Yusuf Ozturk  
Department of Electrical and Computer Engineering



Approval Date

Copyright © 2012  
by  
Sunjna Kashyap  
All Rights Reserved

## **DEDICATION**

My journey towards the advanced degree in Computer Science has been through many vicissitudes. This thesis is dedicated to my father, who taught me that education not only brings intelligence but also changes your perspective about every little thing around you; my mother, who taught me that every obstacle in life is a stepping stone towards a better future; my mother-in-law, who has always prayed for my success; my brother, who has stood by me at all times. And, a special dedication to my husband for his patience and constant encouragement towards achieving my goals.

Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution.

– Albert Einstein

## **ABSTRACT OF THE THESIS**

Image Edge Detection Using Ant Colony Optimization Algorithm

by

Sunjna Kashyap

Master of Science in Computer Science

San Diego State University, 2012

Ant colony optimization (ACO) is an optimization algorithm inspired by the natural behavior of ant species, which deposit pheromone on the ground to guide their foraging. In this paper, the algorithm proposed by Jing Tian for edge detection using ant colony optimization is proposed. The experimental results obtained through the implementation are matched with the author's results to verify the claim to be true. The proposed ACO-based edge detection algorithm represents the edge pixel position of the image using a pheromone matrix in which the movements of the ants driven by the image's intensity values are recorded..

## TABLE OF CONTENTS

|   | PAGE |
|---|------|
| ABSTRACT.....                           | vi   |
| LIST OF TABLES.....                     | ix   |
| LIST OF FIGURES .....                   | x    |
| ACKNOWLEDGEMENTS.....                   | xii  |
| CHAPTER                                 |      |
| 1 INTRODUCTION .....                    | 1    |
| 2 BACKGROUND .....                      | 4    |
| 2.1 Ant System (AS).....                | 6    |
| 2.2 A Simple ACO Algorithm .....        | 6    |
| 2.3 Ant Based vs. Ant Colony .....      | 8    |
| 2.4 Artificial Intelligence .....       | 8    |
| 2.4.1 Expert Systems.....               | 9    |
| 2.4.2 Classifier Systems.....           | 10   |
| 2.4.3 Genetic Algorithms .....          | 10   |
| 2.4.4 Copycat and Starcat .....         | 11   |
| 2.4.5 Neural Systems .....              | 13   |
| 2.4.6 Swarm Particle Algorithms.....    | 13   |
| 2.4.7 Artificial Immune Systems .....   | 15   |
| 2.5 Image Processing .....              | 15   |
| 2.5.1 Pixel .....                       | 16   |
| 2.5.2 Grayscale Images .....            | 16   |
| 2.5.3 Binary and Intensity Images ..... | 17   |
| 2.6 Edge Detection.....                 | 17   |
| 2.7 Pixel Connectivity.....             | 19   |
| 3 ACO EDGE DETECTION .....              | 21   |
| 3.1 Algorithm.....                      | 22   |
| 3.1.1 Adaptive Thresholding.....        | 23   |

|   |    |
|---|----|
| 3.1.2 Understanding How the Ants are Finding Edges..... | 25 |
| 3.2 Design Details .....                                | 25 |
| 3.2.1 Initialization Process.....                       | 26 |
| 3.2.2 Building Process .....                            | 26 |
| 3.2.3 Update Process.....                               | 27 |
| 3.2.4 Decision Process .....                            | 29 |
| 3.3 Implementation .....                                | 32 |
| 4 EXPERIMENTAL RESULTS.....                             | 36 |
| 4.1 Best Results.....                                   | 41 |
| 4.2 Comparison of Results.....                          | 44 |
| 5 CONCLUSION.....                                       | 46 |
| 5.1 Summary .....                                       | 46 |
| 5.2 Outcome.....  | 47 |
| 5.3 Future Work .....                                   | 47 |
| BIBLIOGRAPHY.....                                       | 48 |



## LIST OF TABLES

|   | PAGE |
|---|------|
| Table 4.1. Parameters Used in the Experiment..... | 37   |

## LIST OF FIGURES

|   | PAGE |
|---|------|
| Figure 2.1. Movements of ants from nest (N) to food source (F).....   | 5    |
| Figure 2.2. Ants building path from source to destination. ....   | 7    |
| Figure 2.3. Artificial neuron. ....   | 14   |
| Figure 2.4. Four connectivity neighboring. ....   | 20   |
| Figure 2.5. Eight connectivity neighboring. ....  | 20   |
| Figure 3.1. The algorithm. ....   | 23   |
| Figure 3.2. Neighbors of pixel (i, j). ....   | 27   |
| Figure 3.3. The graphs for $\lambda=10$ for (a) Eq. 12, (b) Eq. 13, (c) Eq. 14, (d) Eq. 15.....   | 28   |
| Figure 3.4. Adaptive thresholding process. ....   | 30   |
| Figure 3.5. Flowchart of implementation details. ....   | 31   |
| Figure 3.6. Implementation of probability transition matrix. ....   | 32   |
| Figure 3.7. Implementation of first update step. ....   | 32   |
| Figure 3.8. Implementation of final update step. ....   | 33   |
| Figure 3.9. Implementation to compute threshold T.....  | 34   |
| Figure 3.10. Implementation of 4-connectivity. ....   | 34   |
| Figure 3.11. All four neighbors of the pixel (row, column). ....  | 35   |
| Figure 3.12. Implementation of 8-connectivity. ....   | 35   |
| Figure 3.13. The eight neighbors of the pixel (row, column). ....   | 35   |
| Figure 4.1. (a) The input image, intensity image, (b) the output image, the binary<br>image.....  | 37   |
| Figure 4.2. The tests run of images in different dimensions.....  | 38   |
| Figure 4.3. (a) Test run image, varying $\alpha$ value, (b) $\alpha=10$ , (c) $\alpha=20$ , varying $\beta$<br>value., (d) $\beta=0.1$ , (e) $\beta=0.3$ with Eq. (3.4). .... | 39   |
| Figure 4.4. (a) Input image of cameraman, (b) input image of Lena. ....   | 40   |
| Figure 4.5. Binary images of cameraman and Lena by using Eq. (3.8).....   | 40   |
| Figure 4.6. Binary images of cameraman and Lena by using Eq. (3.9).....   | 40   |
| Figure 4.7. Binary images of cameraman and Lena by using Eq. (3.10).....  | 40   |
| Figure 4.8. Binary images of cameraman and Lena by using Eq. (3.11).....  | 41   |

|  |    |
|--|----|
| Figure 4.9. (a) 4-connectivity applied, (b) 8-connectivity applied. ....           | 42 |
| Figure 4.10. Images of Lena and peppers. ....                                      | 42 |
| Figure 4.11. The result images of Nezamabadi-Pour's method.....                    | 43 |
| Figure 4.12. The resultant images of Jian Zhang's method.....                      | 43 |
| Figure 4.13. The binary images of proposed method .....                            | 44 |
| Figure 4.14. Dimensions 128 x 128, 256 x 256, 512 x 512 vs. computation time. .... | 45 |

## **ACKNOWLEDGEMENTS**

I would like to thank Dr. Joseph Lewis for inspiring me towards the concepts of Artificial Intelligence and guiding me throughout my Master degree program at every step. I would also like to thank Dr. Carl Eckberg and Dr. Yusuf Ozturk for offering to serve on my defense committee.

## CHAPTER 1

### INTRODUCTION

Ant Colony Optimization (ACO) is a population-based metaheuristic that is useful in solving complex computational problems and finding proximate solutions to difficult optimization problems. Ants in nature are impressive in finding an optimal path to their destination usually the food source. They tend to wander randomly from their hive in search of food, and when they have to travel, back to their hive they deposit a chemical substance called the pheromone throughout their way. The other ants follow the paths with pheromone content rather than wandering around multiple paths in order to reach their point of interest (usually food). In addition, these ants increase the pheromone content on the trail. Since pheromone content evaporates quickly, a shorter route proves to have a stronger content of pheromone as more number of ants traverse through it. Longer paths have a fewer number of ants passing by and hence lighter pheromone content in them. In relatively lesser time, a majority of ants will be traversing the shorter path, which has the highest pheromone content. ACO adapts this phenomenon of nature by using artificial ants which are nothing but a software agent to find good solutions to a given optimization problem.

Alberto Coloni, Vittorio Maniezzo and Marco Dorigo were the first to come up with an algorithm to simulate the ants' behavior, naming it an *Ant Colony System* (ACS) algorithm, commonly known as Ant Colony Optimizations and abbreviated as ACO. Ant Colony Optimization has made a mark in various fields with significant contributions to optimization problems. ACO inspired by a colony of ants that search for a common source of food. A randomized search heuristic technique is widely adopted for solving problems from combinatorial optimization. Construction of the solutions for a given problem is randomly walking on a construction graph. The heuristic information of the problem influences the random walk [1].

ACO algorithms have the ability to integrate information about the problem into the construction of a new solution which other randomized search heuristics like Simulated Annealing do not perform. Traveling Salesperson Problem was one of first combinatorial

optimization problem to which ACO technique [1] is applied. The ants are able to find a shortest path to a food source under certain circumstances by indirect communication. The pheromone values influence the communication. The behavior of ants stimulated into an algorithmic framework to obtain solutions for a given problem. Solutions constructed by random walks of artificial ants on a construction graph, which has weights or the pheromone values on the edges. Larger pheromone values leads to higher probability of traversal by another set of ants in the next iteration. In a series of experiments on a colony of ants with a choice between two unequal length paths leading to a source of food, biologists have observed that ants tended to use the shortest route.

Image Edge Detection is a fundamentally important feature in image processing/analysis. Edges characterize boundaries and so have a wide range of useful applications such as segmentation and identification of objects in scenes, machine vision, astronomy and microscopy imaging to name a few. Edge detection is a technique for marking sharp intensity changes, and is important in further analyzing image content. It lays the foundation for image fusion, shape extraction, image segmentation, image matching, and image tracking. Many traditional edge detection approaches result in broken pieces, which lead to incomplete resultant images. The research done in this paper as well as by many fellow researchers present that ant colony optimization based mechanism tends to compensate those lost edges. The proposed procedure indicates that the edge detection approach is efficient on compensating broken edges and more efficient than the traditional ACO approach in computation reduction.

This paper focuses on extracting edge information from an image by applying the ACO algorithm such that a number of ants which are referred as the software agents are propagated on the image. By adding the intensity values of the image, which is the edge information at each pixel, the pheromone matrix is constructed. The main motivation to this paper is the sparse research work done with solving the edge detection problem with ACO. The difference between other researches and ours is: Our approach being Ant Colony approach compared to [2] which is an Ant System and also in this paper the edge detection is directly associated with ACO (image is pre-processed) rather than using conventional edge detection algorithms to collect the edge information and enhance using ACO as followed in [3].

It is unknown at the beginning if the approach followed can yield a good result as ACO falls under the category of algorithms, which are probabilistic in nature, and hence, the best solution among the given solutions is always the choice. In addition, it is unusual to have the same results recur always. The goal of this thesis is to prove that the edge detection of an image using ACO achieves the results as claimed in [4] and compare the proposed approach's results with [2].

In Chapter 2, we deal with the background study. The goal of the project, algorithm and implementation details in detail is in Chapter 3. The results produced from the algorithm and the comparison between some of the earlier approaches with the proposed approach in detail is in the Chapter 4. The last Chapter explains in detail whether the success and failures claimed in [4] are true and discusses the results achieved by the proposed implementation method.

## CHAPTER 2

### BACKGROUND

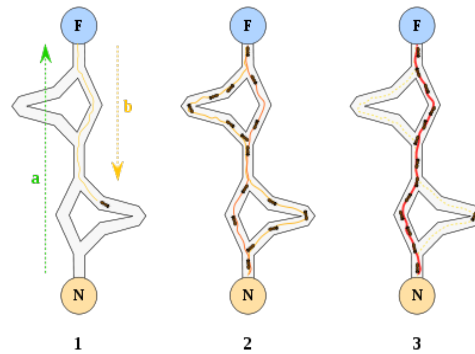
Ant colonies show a structured social organization in spite of the simplicity of their individuals. Because of their “socializing” ant, colonies are capable of solving complex tasks that are impossible to accomplish with a single ant. The ant colony algorithm is a model, inspired by close observation of the real world ants’ behavior, which is a solution to many optimization algorithms. The idea behind ant algorithms is to use a form of artificial stigmergy to coordinate societies of artificial agents. ACO inspired by the foraging behavior of ant colonies, and targets discrete optimization problems. Many ant species are completely blind and early research has proved that the ants’ behavior is a basis of their communication among individuals and the environment. They use a chemical produced by them in order to characterize this communication. The chemical is termed pheromone. The most important feature in ant species is the pheromone trail. Ants use this chemical as a marking for paths, which lead them to their food source and back to their nest. Ants tend to deposit this chemical, the pheromone all their way to the food source and on their path back to their nest. Other ants of the colony sense the strongest smell of the pheromone trails and follow the path to the food discovered by other ants. This, collective process of trail laying and trail-following behavior whereby an ant influenced by the chemical trail left by other ants is the inspiration behind ACO.

The experiments show that ant colonies have the capability of optimizations. They use probabilistic behaviors based on local information to find the shortest path between two points in their environment. For experimentation purposes, scientists have created artificial ants to mock the behavior of the real world ants. Using a graph, the shortest path between the two nodes corresponding to the nest and the food source is designed. The approach this thesis will ultimately take is akin to the ACO technique; it is of note that a simple version of this problem of finding a shortest path between nest and food was performed in experiments in another thesis at SDSU that relied on the ant-based technique [5].



As, shown in Figure 2.1:

1. The first ant finds the food source (F) in a randomly selected route (a) and then returns to the nest (N), leaving behind a trail pheromone (b).
2. Ants wander in the four different ways making the stronger pheromone containing route as the most attractive (short) route.
3. Ants take the shortest route and the pheromone content on the other trails decays.



**Figure 2.1. Movements of ants from nest (N) to food source (F).**

A metaheuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality [6]. A particularly successful metaheuristic is inspired by the behavior of real ants. A number of algorithmic approaches based on the behavior of real ants to solve combinatorial problems as well as real world applications, is developed. The Ant Colony Optimization is a metaheuristic framework, which is the basis of the algorithmic approach mentioned above. In recent years, ACO applies successfully to numerous optimization problems, such as quadratic assignment problem, data mining, data clustering, and image processing. Marco Dorigo as his thesis introduced the first ACO system in 1992. He named it Ant System (AS). Based on his research in the field of Artificial Intelligence, the Ant System came into existence. With joint efforts of Marco Dorigo, Alberto Colorni and Vittorio Maniezzo, Ant System was initially applied to Travelling Salesman Problem. Since, then Dorigo has been working on the extended versions of the AS paradigm. Some of them are the Ant Colony System (ACS) and MAX-MIN Ant System (MMAS). Both of these extensions have been applied to symmetric and asymmetric travelling salesperson problems thus, achieved satisfactory results. The most recognized version is the ant colony optimization.

## 2.1 ANT SYSTEM (AS)

Ant System (AS) was one of the first ACO algorithms. It demonstrated the behavior of ant colonies and served as a model for all the other variants developed so far. Ant System was a viable approach to many combinatorial problems and many applications however, it failed to produce extensive results for many real-world problems. The main characteristics of this model are positive feedback, distributed computation, and the use of a constructive greedy heuristic. At this time, three other important algorithms were developed - Ant-Cycle, Ant-Density and Ant-Colony.

1. *Ant-Cycle*: Ants are initialized, then those ants are set to move on the trails and ants deposit pheromone only after they have built the entire tour. Experimentally it is found that there is a linear relation between the number of towns and the best number of ants. The complexity of the algorithm is  $O(\text{Number of cycles} \cdot n^3)$ .
2. *Ant-Density*: This algorithm differs from the *Ant-Cycle* from the way the trail is updated. Ants deposit pheromone on the edge each time it passes through it and deposit a constant quantity of  $Q$  on the trail each time [3].
3. *Ant-Quality*: This algorithm functions similarly to the *Ant-Density* model but, on each time it passes through an edge a pheromone trail of  $Q/d_{ij}$  is deposited where  $d_{ij}$  is the Euclidean distance from node  $i$  to  $j$ . Therefore, the shorter paths seem to be more viable in this approach [7].

The *Ant-Cycle* has proven to show better results when compared to *Ant-Density* and *Ant-Quality* in many experimental results. *Ant-Cycle* uses global information, such that the amount of pheromone deposited on the trail is directly proportional to the quality of the solution. In fact, ants producing shorter paths contribute a higher amount of trail than ants whose tour was poor. On the contrary, both *Ant-Quality* and *Ant-Density* use local information. There seems to be no dependencies between the pheromone quantity deposited and the quality of the solution produced. Their search is not directed by any measure of the result achieved [3]. It has been shown by many scientists that *Ant-Cycle* approach is more viable compared to the other two in terms of computation time and also degree of the result produced.

## 2.2 A SIMPLE ACO ALGORITHM

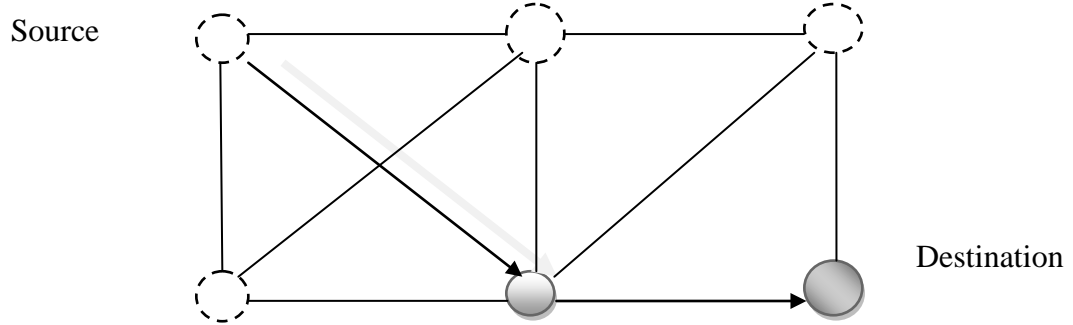
This section illustrates a simple ACO algorithm. Each artificial ant's main task is to find the shortest path between a pair of nodes on a graph. Let  $G=(N, E)$  be a connected graph with  $n=|N|$  nodes. The simple ant colony optimization algorithm can be used to find a

solution to the shortest path problem defined on graph  $G$ , where a solution is a path on the graph connecting a source node ' $s$ ' to a destination node ' $d$ ', and the path length is given by the number of hops in the path [8]. To each arc  $(i,j)$  of the graph is associated a variable  $\tau_{ij}$  which is the artificial pheromone trail. These trails are updated and handled by the ants. The intensity of pheromone trail is proportional to the estimation made by ants to build a good solution. To build the solutions each ant adds a stepwise decision called the building process. Each node stores local information, which is used in each step by the ants to get help to move to the next node. This decision step of each individual ant  $k$  located at node  $i$  use the pheromone trails  $\tau_{ij}$  to compute the probability with which it should choose the node  $j \in N_i$  as the next node to move to where  $N_i$  is the set of neighbors of node  $i$ .

$$p_{ij}^k = \begin{cases} \tau_{ij}, & \text{if } j \in N_i \\ 0, & \text{if } j \notin N_i \end{cases} \quad (2.1)$$

As seen in the Figure 2.2, ants build solutions from the source to destination nodes. While building the solution the ants deposit pheromone contents on the path they move. If they deposit  $\Delta\tau$  of pheromone then if ant moves from node  $i$  to  $j$  in  $t$  time then:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau \quad (2.2)$$



**Figure 2.2. Ants building path from source to destination.**

This increases the probability that all ants moving from node  $i$  to node  $j$ , will use the same arc in future. Just as if a real world pheromone trail will evaporate over time, the artificial ants also have this property to avoid all ants to converge into a sub-optimal path quickly [8]. In, the algorithm the evaporation is carried out in a simple way, i.e. decreasing pheromone trails by:

$$\tau \leftarrow (1 - \rho)\tau \text{ where } \rho \in [0,1] \quad (2.3)$$

at each iteration. The experimental results show that when the above graph is considered, optimal path is found from source to destination but if more than one close to optimal paths are considered then the results are not stable and the values chosen for each parameter in the algorithm becomes a crucial selection.

### 2.3 ANT BASED VS. ANT COLONY

ACO techniques are widely used for solving computational problems such as finding an optimal solution through graphs. It builds its solution based on the general idea of ant colonies. Every ant in the colony finds a solution for the problem defined and leaves a pheromone trail on the path it traversed which leads as a clue for the other ants following. Ant-Based techniques vary from the ACO techniques from the fact that all the ants collectively join to find the entire solution. Each ant in this technique finds only part of the solution. A complete solution to the problem is obtained from combining solutions discovered from all the individual ants. The major difference between Ant-Based and ACO being that ACO finds the entire solution for the problem whereas Ant-Based determines its final solution based on the cumulative results from all the ants in the colony, which makes ants in Ant-Based algorithms to solve the problem in smaller sections and advantageous for distributed networks.

### 2.4 ARTIFICIAL INTELLIGENCE

Alan Turing in 1950 discussed conditions for considering if a machine is intelligent. Can machines think or behave intelligently? In order to believe that Turing's statements claimed were right, a series of tests had to be thought of. Computers must be able to communicate to the examiner, store and retrieve information, use the stored information to solve issues and find new solutions and to adapt to new circumstances without user intervention. In 1960, the cognitive science revolution brought evidence to model activities of the brain. Many schools developed a set of rules for thoughts and logic, which came with some uncertainties such as facts being predicted wrong, or accuracy of information. AI research belongs to two main approaches, *biological* - based on the idea that AI should study humans and imitate their psychology and *phenomenal* - based on studying and formalizing common sense about the problems and achieving goals. John McCarthy believed that both the approaches are players in the game. Language processing, speech recognition, automated

reasoning and planning are some of the branches of AI. Another fundamental concept during early AI systems was that symbolic representation was all that was required for intelligent computations. The Physical Symbol System Hypothesis was a research by Newell and Simon based on this idea. The system also called formal system takes the symbols and combines them into expressions and manipulating them to produce new expressions. Newell and Simon state that “A physical symbol system has the necessary and sufficient means for intelligent action”. The implementation began with symbolic representation for a problem domain. Using basic symbol operation, the symbols were combined to form higher-level expressions. Basic letters could be symbols and based on some representation, they could form words and further form phrases or sentences. These systems applies well to logical problems but were very expensive and less optimal for real-world applications. There grew an intense requirement for knowledge base computer programs to solve a number of real-time applications. These requirement leads to the importance of Expert Systems. During 1980-88, the expert systems came into light. Expert systems are a computer system, which imitates the decision-making ability of humans. The system is used to solve complex problems of reasoning. Unlike traditional programs, the Expert System is capable of giving reasoning for a particular subject similar to the human mind. In addition, further years gave prominent contributions to Neural Systems and Genetic Algorithms (GA's).

### **2.4.1 Expert Systems**

Expert systems use human intelligence to solve problems. These expert systems represent the knowledge from human intelligence as rules within a computer. When solving problems the data or rules are utilized. Conventional computer programs perform tasks using conventional decision-making logic containing little knowledge other than the basic algorithm for solving that specific problem and the necessary boundary conditions. The program knowledge is embedded within the program and in order to change the knowledge base, the program is changed and recompiled. Fragments of human knowledge are embedded into the program into something called the knowledge base and is applied to reasoning the problem with the appropriate knowledge.

Compared to conventional programming, these systems / programs are suitable to be applied to another problem, which shares the domain of the knowledge base. Most expert

systems are developed via specialized software tools called shells. These shells come equipped with an inference mechanism (backward chaining, forward chaining, or both), and require knowledge to be entered according to a specified format (all of which might lead some to categorize OPS5 as a shell). They typically come with a number of other features, such as tools for writing hypertext, for constructing friendly user interfaces, for manipulating lists, strings, and objects, and for interfacing with external programs and databases. These shells qualify as languages, although certainly with a narrower range of application than most programming languages [8].

### **2.4.2 Classifier Systems**

Expert Systems were successful in programs, which illustrated human knowledge. But, there were many criticisms still made in machine learning. The three main criticisms were, deciding which rules in a rule-based system are responsible for its successes, particularly in long sequences, having enough monolithic rules to solve the problem and rule discovery (or its equivalent) is the most recondite problem. Among these were also another important question, How do these systems improve their performance? [9] John Holland developed a sub-symbolic system called the classifier system to address the issues. He came up with a concept of ‘micro-rules’ which were nothing but smaller symbolic rules. These rules were called classifiers. In order to perform the task and solve the potential problem these rules would interact with unknown, changing environments. A classifier system works by allowing its classifier agents to interact with an environment. The environment provides messages to the classifiers, which create actions that can affect the environment. When the actions are applied to the environment, a feedback system distributes positive or negative credit back to the rules that originated the actions. Holland’s classifier system, referred to as a Michigan classifier<sup>1</sup>, also included an evolutionary component to it in order to develop new rules based on the success of existing rules. This introduction of a genetic algorithm allowed the system to improve its performance above the level provided by the initial rules. This type of system is often referred to as a learning classifier system [10].

### **2.4.3 Genetic Algorithms**

In the domain of Artificial Intelligence, Evolutionary Algorithms are the main class of algorithms that are inspired by the process of evolution in nature such as mutation,

crossover, inheritance and selection. Among a set of solutions for a problem, the best solution is termed as the candidate solution, which acts as an individual to a problem. The fitness function is the objective that describes how close the solution is to what it claims and hence is the main factor to decide the environment of the individual. The main applications are in the areas of bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics and physics.

Genetic Algorithms is a heuristic method that is a subset of Evolutionary Algorithms. They use the same method as Mother Nature uses to reproduce. John Holland in 1975 discovered the importance of the method for solving problems. In particular, Holland suggested that GA(s) work well on continuous and discrete combinatorial problems. On the contrary showed to be computationally expensive [11]. GA's main applications are in optimization and search problems. The advantages of GA's are that it can handle varied types of constraints. To put the concept to use every solution of a problem is represented as a chromosome and the Genetic Algorithm creates a population of solutions and applies operators like mutation(altering one of more gene values in a chromosome) and crossover(taking more than one parent solutions and creating a child solution) to evolve and find the best solution [10].

Besides the computational limitations introduced by the evaluation of a fitness function, the genetic algorithm illustrates another recurring limiting phenomenon. The decision to be made for encoding all the variables in a genetic manner which tend to have an important impact on the possible solutions and how the fitness function can interpret the quality of the individual. A fixed set of parameters are to be set at the beginning of the process [12].

#### **2.4.4 Copycat and Starcat**

Copycat is a program developed by Douglas Hofstadter and Melanie Mitchell, which is capable of making analogies. Analogy making can be defined as “the perception of two or more non-identical objects or situations as being the ‘same’ at some abstract level” [13]. Copycat is a very popular and different model when compared to other cognitive science architectures such as Psi or DUAL. Copycat program is very specific to letters of English alphabets and the relationship between them. Copycats basic functionality is to find analogies

i.e. if abc as abd then xyz as what? The input to the program is three sets of letters and each set consists of three letters. The first two sets contain the transition of letters based on the rule. A rule is the guide set at the beginning of the program, which decides the transition between the first two sets of letters and the corresponding sets. Example of a rules for the analogy (abc as abd then xyz as what?) can be replace the last letter with 'd' or replace the last letter of the set with the consecutive letter. Copycat is designed to choose the rule, which can yield best solutions. Therefore, swap the last letter of the set with the consecutive letter will be the best solution in this case. Copycat's system of letters of English alphabets allows transitioning such as reversing of letters in a set, swapping, and changing cases i.e. upper to lower or vice versa. The transition rule based on the first two sets is used to find the result of the third set. The result, which forms the fourth set, is referred as the output of the program. The architecture of Copycat consists of a slipnet, which is network of nodes that hold permanent concepts and links. Links represent relations between the nodes; a workspace is the working area and coderack, which hold codelets. Codelets can modify activations in the slipnet and build structures in the working area based on the current state of the workspace and the slipnet. To provide the basic functionality of the Copycat program, Starcat a framework was created by Dr. Joseph Lewis; it allows other domains to utilize the functionality of Copycat. "Emergent representation is a key feature of [1] functioning and adaptive behavior is the desired result that it supports" [1]. Starcat can be added on top of the Copycat model. Lewis states that the Madcat and Copycat systems have "two important limitations... their inability to adapt to novel situations and their lack of true autonomy" [1]. Starcat "is an open-ended architecture for computational systems that autonomously adapt their behavior to continuously changing environments" [1]. The ability to allow for autonomous systems makes Starcat an important contribution to field of Artificial Intelligence. Starcat is an asynchronous architecture and every component in the framework runs its independent thread. The coderack is a probabilistic priority queue and the workspace can be accesses in particular ways e.g. biased distribution used for random selection. The slipnet manages activation levels and relationships among nodes whose primary job is the production of a varying population of different types of codelets [1]. Artificial Intelligence gets a new direction with these machines, which depict emergent behavior. These machines are supplementing the human minds, which are susceptible to some errors. It is a novel idea



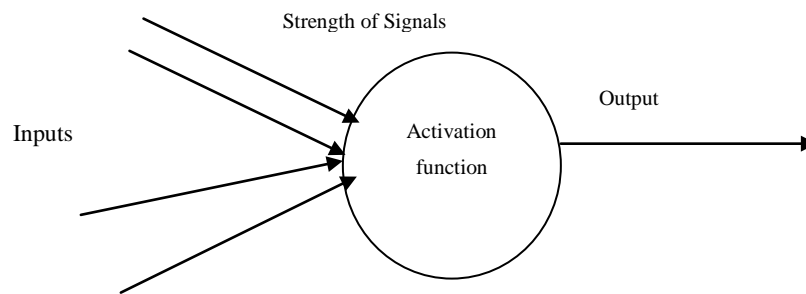
that supplements machines to new ideas without human interaction and guidance and help them to work independently. AI has taken an optimal and strong human approach. The term “AI” no longer seems to apply to off-the-shelf solved computing-science problems, which may have originally emerged out of years of AI research but is applied to simple day-to-day activities such as an online assistant.

### **2.4.5 Neural Systems**

Artificial Neural Networks are the simulation of the brain. The idea behind this theory is that certain neurons can be extracted and applied to simulations to simulate the brain. The nodes of this network are ‘artificial neurons’. Artificial neurons are a computational model inspired by the natural neurons. Natural neurons receive signals through synapses located on the membrane of the neuron. If the signals received surpass the threshold value then, the neuron is activated and emits a signal through the axon. The signal can also be sent to another synapse and in turn activate other neurons. There is a visible lack of concentration when modeling artificial neurons. The inputs i.e. synapses, which are multiplied by the strength of the signals, are computed by a function which will determine the activation of the neuron. Another function determines the output as in Figure 2.3. All artificial neurons are combined to process the information. The higher the strength the higher will be the inputs multiplied so for obtaining a strong output the strength of signal can be modified accordingly. The drawback is in a network when we have thousands of neurons, adjustment of inputs for the required output will be impossible. However, the algorithms such as learning/training are present, which help the network, handle adjustment of each neuron. McCulloch and Pitts developed the first artificial neural network in 1943. From then, many hundreds of neural networks have been designed which differ in the functions, learning algorithms or topology. Neural networks have been used since then in many engineering applications such as pattern recognition, forecasting etc. and in modeling real neural networks for analyzing animals.

### **2.4.6 Swarm Particle Algorithms**

Swarm Intelligence is a technique of Artificial Intelligence where a set of agents are liable to communicate indirectly or directly with each other and collectively solve a distributed problem. It takes inspiration from social behaviors of insects and animals. It finds



**Figure 2.3. Artificial neuron.**

approximate solutions to NP-hard problems. It is a collective behavior of natural or artificial systems. They share a particular pattern without having a central entity ruling them. The system is a set of agents, which belong to a population, and these agents interact with each other, follow a set of rules, and evolve to show intelligent behavior. Examples of swarm intelligence based systems are colonies of ants and termites, schools of fish, flocks of birds, herds of land animals. Some human artifacts also fall into the field of swarm intelligence are some multi-robot systems.

Particle Swarm Optimizations is a Swarm Intelligence technique that is population based and is inspired by social behavior such as bird flocking or fish schooling. It was first presented in 1995 by Dr. Eberhart and Dr. Kennedy. It shares traits similar to evolutionary algorithms such as Genetic Algorithms. It applies concept of social interaction to problem solving. It is initialized by set of random potential solutions called particles of a population, which search for the optimum solution based on updating which together form a generation. Each problem keeps track of the best solution attained so far and stored as the fitness value. Thus has no need of operators such as mutation and crossover as in Genetic Algorithms.

Along with the best solution value (fitness value), the feasible value attained so far by any particle in the neighbors of the particle is also kept track of. In other words, each particle is a point by itself in space and adjusts its 'flying' based on itself as well as experiences of its neighboring particles. Compared to other evolutionary algorithms PSO is computationally cheaper. The main advantage of PSO is the fewer parameters to handle and its varied applications focusing specific requirement. A variant of PSO is the Hybrid PSO, which utilizes basic mechanism of PSO and the natural selection mechanism, which is usually utilized by methods such as GAs.

### **2.4.7 Artificial Immune Systems**

A novel technique inspired by immunology is Artificial Immune Systems. The immune system consists of a number of highly mobile units called antibodies that repel invaders called antigens. As, the invaders come in infinite different forms the system cannot simply list all possible invaders. Therefore, the immune system needs to adapt the antibodies to different invaders. Some of the applications are Fault diagnosis, Computer Security, Robot behavior, Machine Learning etc. Some common techniques inspired by immune systems and their theories are Clonal Selection Algorithm, Negative Selection Algorithm, Immune Network Algorithms and Dendritic Cell Algorithms. The clonal selection is a theory that the body can generate immune cells that can specifically recognize and attack just about any pathogen you could encounter before even encountering them. Both B-cells and T-cells, which together form lymphocytes, are affected by clonal selection. When the B-cell is activated, clonal selection is said to occur. When the B-cell binds with a particular antigen, it begins to clone itself. That mean the cell begins to divide and produce cells (clones) that will also recognize the same antigen.

## **2.5 IMAGE PROCESSING**

Image processing is the study of representation and manipulation of pictorial information. An image is a matrix of square pixels i.e. picture elements arranged in columns and rows. The images are processed in order to alter the appearance, optimize the image for the application, and analyze the image in the computer for diagnosis such as to provide clues to radiologists to detect suspicious structures. Image Processing usually refers to digital image processing but analog and optical image processing also is possible. Analog image processing is any image-processing task conducted on two-dimensional analog signals by analog means. Optical Image Process focuses on the domain of optics dealing with lens, microscopes etc. Digital Image Process as the name suggests is processing of digital images. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Image Processing is divided into five groups [14]:

1. Gray level Segmentation and Thresholding methods
2. Edge Detection Techniques
3. Digital Morphology

4. Texture
5. Thinning and Skeletonization algorithms

Our approach is an edge detection technique, which uses thresholding methods to decide the edge pixel and convert intensity images to binary images. Binary images are monochromatic images of black and white and are used to represent useful information about the image such as edges, shapes, textures, number, etc. of the shape objects. Some concepts of image processing that will be used throughout this paper which are of importance are explained below. Edge is an object between boundary and background of an image. The edges of the image's objects can be identified by properties such as area, perimeter, and shape to name a few. Monochromatic images simplify the operations such as pattern recognition and classification. Digital Morphology is a section of image processing that deals with filtering of images and analysis of structural elements. To recognize the shape of an image, a set of mathematical operators are applied. Texture is repetition of similar patterns in a small region. The well know approach to identify different textures is to apply a unique gray level or color to each pixel belonging to same texture. Thinning is a method applied to skeletonization. It can change the objects appearance to fit to applied algorithm. Skeletonization algorithm is a concept of reducing the original image to a compact representation.

### **2.5.1 Pixel**

Pixel is a smallest element in any display device, which can be seen, controlled or represented in imaging. Every pixel has its own set of coordinates, which form the address of the pixel making it easy to address individual pixels. Conventionally pixels are represented by dots or squares. The intensity of each pixel is variable. In color image systems, a color is typically represented by three or four component intensities such as red, green, and blue, or cyan, magenta, yellow, and black. An original image is defined by a set of millions of pixels.

### **2.5.2 Grayscale Images**

In image processing, a grayscale image is an image in which the value of each pixel is a single sample (sampling is process of reduction of a continuous signal to a discrete signal), that is, it carries only intensity information. These are known as black and white images, which are shades of gray. The color black is referred to have the weakest intensity and white

to have the strongest. Grayscale images are also referred as monochromatic images. The monochromatic images need less information to be provided for each pixel. Grayscale images are given importance as most of image capturing hardware can only support 8-bit images. These images are very efficient and sufficient for many tasks.

### **2.5.3 Binary and Intensity Images**

In binary images, every pixel can have only two possible colors. Commonly black and white colors are used which has become a convention now. Moreover, the same convention is followed for the scope of this project. The color used for the object in the image is the foreground color while the rest of the image is the background color. A binary image is usually stored in memory as a bitmap, a packed array of bits. The input to the proposed algorithm is also bitmap images.

An intensity image is a matrix whose values represent intensities within some range. Every element of the matrix corresponds to one image pixel. The matrix can be of class double, uint8, or uint16. MATLAB uses colormap to display intensity images. Intensity images are also called indexed images. The interpretation of intensity images depends completely on the rendering device's parameters or characteristics. They are classified into intrinsic and extrinsic parameters. The extrinsic parameters transform the 'camera' reference frame to the world reference frame. The intrinsic parameters describe the optical, geometric and digital characteristics of the 'camera'.

## **2.6 EDGE DETECTION**

Edge detection is a process of identifying sharp discontinuities in an image. The discontinuities are changes in the pixel intensity, which characterize boundaries of images. There are many methods to find the edge of an image with an operator e.g. a 2-D filter. Choosing the operator depends on parameters such as noise, edge orientation and edge structure. Edges are significant properties of the image. Physical edges are often correlated with intensity changes in an image. A classical problem in computer vision is how these changes can be detected, extracted and best represented.

The four steps of edge detection are

1. Smoothing: Without destroying the edge information, remove the inference of noise.
2. Enhancement: A filter is used to sharpen the quality of the image.

3. Localization: Localization is to find the exact location of an edge. Edge thinning and linking are usually used in this step.
4. Detection: To provide a criterion to determine edge pixels, thresholding is used.

To detect an optimal edge following is the criteria

1. The detection approach should be able to detect a proper edge. It must be able to reduce the false positives and false negatives. False positives are detecting deceive edges and false negatives is detecting misleading edges which are not real.
2. The edges should be real edges and return only one edge for every edge point.

There are many methods to perform edge detection. However, most of the methods are categorized in two ways. The Gradient based edge detection and the Laplacian based edge detection. The gradient method detects the edges based on the maximum and minimum in the first derivative of the image, whereas the Laplacian method searches for zero crossings in the second derivative of the image to find edges. Some of the techniques of edge detection are Sobel Operator, Robert's cross operator, Prewitt's operator, Candy's edge detection algorithm etc. Edge detection aims to localize the boundaries of objects in an image and is a basis for many image analysis and machine vision applications. The conventional methods show some drawbacks. The computational time increases with the increase in size of the image. In addition, the operators are applied on each pixel of the image making it very expensive as a process. The basic idea behind the gray gradient method is the comparison between the pixel values, which is used in this project. The gray gradient value of a pixel is measured by the variation in the intensity values of the image. An edge is detected by a sudden jump in the gray value from one pixel to another. Gray-based edge can be defined as a curve, which the gray-level is different on both sides of the curve. Edges are points where there is a boundary between two image regions. Edges can be of any shape of length. Edges are sets of points in the image, which are nothing but a group of pixels, which have a strong gradient magnitude. Grouping the high gradient value points together to form an edge based on constraints such as shape, smoothness and gradient value are some common practices of many algorithms. The gradient value is defined as follows:

$$\Delta I(i, j) = \max\{|I(i, j - 1) - I(i, j + 1)|, |I(i - 1, j) - I(i + 1, j)|, |I(i - 1, j + 1) - I(i + 1, j - 1)|, |I(i + 1, j - 1) - I(i - 1, j + 1)|\} \quad (2.4)$$

where  $I(i, j)$  is the gray value of pixel  $(i, j)$  [4].

Finding an edge of an image is a well-addressed problem in the field of image processing. Many approaches have been implemented with many pros and cons. In this paper, the Ant Colony Algorithm is used to find the edge of the images. This approach is been applied to overcome some failures of the edge detectors which tend to lose some information towards achieving the goal. Considerable amount of work has already been performed on edge detection using Ant Colony Optimization methods [2], [4]. The goal of this paper is to prove that the ACO method is better than other conventional methods and compare the results based on other authors' research so far [2], [4], [15].

Some thresholding methods are

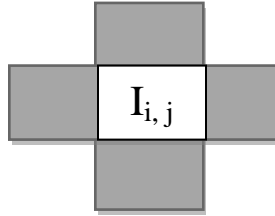
1. P-tile method: The threshold is determined by calculating the number of black pixels desired and using the histogram of gray levels in an image. The advantage of this method is that the percentage of black pixels can be changed at any point of time.
2. Edge Pixel method: The threshold value is determined by the operator based on the digital Laplacian operator.
3. Iterative method: An initial threshold value is altered by consecutive iterations of modifying the value through the image. Using the levels in an object and background of the image, the threshold value is altered such that it is improved.

The iterative method is used to find the threshold value in our paper. The initial threshold value is the mean of the gray level values of the intensity image.

## **2.7 PIXEL CONNECTIVITY**

Pixel connectivity is an important feature in image processing. In order to group pixels an image is scanned and two basic criteria are to be matched. The first of which being that all pixels share the same intensity values and the pixels have to be connected in some format with each other. Then, each pixel is color labeled or gray labeled. Extracting and labeling of various disjoint and connected components in an image is central to many automated image analysis applications [16]. The process is called 'Connected Component Labeling'. Scanning each image from top-bottom and right-left is done to find the pixel regions that are adjacent to each other and share similar intensity values ( $V$ ) i.e. a binary image  $V=\{1\}$ ; however a gray level image can have range of values  $V=\{51,52,\dots,80\}$ . Connected component labeling works on binary or gray level images and different measures of connectivity are possible. It determines how the pixels in 2 or 3 dimensional images are connected to their neighbors. The various types of connectivity for 2 dimensional images are:

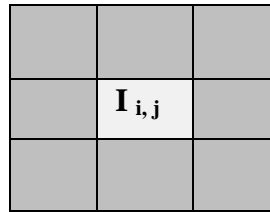
1. 4-connectivity--Each pixel has 4 neighbors such that all the 4 neighbors touch the pixel edges (see Figure 2.4).



**Figure 2.4. Four connectivity neighboring.**

A set of pixels are 4-connected if for every pair of the pixels  $I_{i,j}$  there is a sequence of pixels  $I_i \dots I_j$  such that every two pixels that are adjacent in the sequence are 4-neighbors.

2. 6-connectivity--Each pixel has 6 neighbors such that all the 6 neighbors touch the corners of the pixel.
3. 8-connectivity – 8-connected pixels are neighbors to every pixel that touches one of their edges or corners (see Figure 2.5).



**Figure 2.5. Eight connectivity neighboring.**

A set of pixels are 8-connected if for every pair of the pixels  $I_{i,j}$  there is a sequence of pixels  $I_i \dots I_j$  such that every two pixels that are adjacent in the sequence are 8-neighbors.

The 4-connected pixels can always be 8-connected. Set of 4-connected are a subset of 8-connected but not the other way. 8-connected may not always be 4-connected. The 3 dimensional images can be 6-connected, 18-connected or 26-connected. In this paper, we only refer to 2-dimensional images.



## **CHAPTER 3**

### **ACO EDGE DETECTION**

Many methods are implemented over the years for edge detection among which is using the Ant Colony Optimization algorithm to detect the edge in an image. This project implements the ACO algorithm for Image edge detection using Matlab. The ultimate goal of this project is to prove that the results of other researchers and their papers. It requires an image as an input and produces the resultant image. The algorithm refers to the iterative selective method to find the threshold level of an image. This iterative technique for choosing a threshold was developed by Ridler and Calvard. It converts the intensity image to a binary image [17]. The computation time is between 70s and 90s. This project can be further improvised by applying the parallel ACO algorithm to improve computational load. The ACO algorithms parameters can be adjusted to get a slightly better resultant image. However, occasionally, the algorithm's performance reduces depending on slowly performing computers.

There has also been some research earlier to apply the Genetic Algorithm along with the Ant Colony Algorithm to find the image edge information. The average speed of the population answers that are close to the optimal results is low and hence the answer might not converge to the absolute optimal answer. This is the case that ants' algorithm has a low populating dissipation and the answer's formation converge quickly, but it has a high approaching speed to optimum answer. The claim of these researchers is that their approach increases the average speed to find the optimal answer by applying the ants' pheromone information to the crossover function [18].

This approach is not included in the scope of this paper and will be taken as a challenge for further research. Since the approach followed in this paper considers the sharp intensity changes made in the image for detecting the edge, the loss of important edges in the image is reduced when compared to conventional methods. The factors considered in the edge detection process are Edge Orientation and Edge Structure. Edge orientation is the direction the algorithm looks for changes in intensity levels to find the edge whereas, edge

structure is the process of detecting edges which don't have a step change in intensity levels. There are many techniques to detect these changes such as changes in intensity levels between the face and the hair in the image.

In this section the algorithm, implementation details and the setup procedure are mentioned in detail along with the changes made from the referring research paper. The scope of the current capacities of the project is clearly been defined in the initial experiments. The main goals of this paper are (1) to prove that the algorithm proposed by [4] produces optimal results as it claims, (2) to improve the performance time of the algorithm.

### 3.1 ALGORITHM

The algorithm was already designed in the research paper referred throughout this project. The first task of the development process was to select a language for programming. Matlab was chosen over other programming languages because of its ease to use in small projects. It is simple approach for numeric computing and its ability to visualize the results in a convenient and efficient manner. It has been proven as a feasible approach for image processing and handling. The edge selection based on the ACO algorithm is proven an apt approach to extracting the full edge information. An ant is said to move from node  $i$  to node  $j$  with a probability of

$$p_{i,j} = \frac{(\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)}{\sum (\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)} \quad (3.1)$$

where,

$\tau_{i,j}$  is the amount of pheromone on edge  $i, j$

$\alpha$  is the constant which influences the  $\tau_{i,j}$

$\eta_{i,j}$  is the heuristic information of the edge  $i, j$

$\beta$  is the constant which influences  $\eta_{i,j}$

Ants move from a pixel to another pixel on the graph for constructing a pheromone matrix, each entry of which forms the edge information at each pixel of the image. This paper chooses to have  $K$  ants, which will move through a series of steps to find the edge information of 2-D images by keeping account of the change in intensity levels. These ants are applied in space  $\chi$  that has  $M_1 \times M_2$  nodes to find the optimal solution by building pheromone information. The algorithm has been divided into initialization, building, update,

and decision steps. To convert the intensity image to the binary image, the threshold iterative selection method is used developed by well-known scientists Ridler and Calvard. They developed a technique for determining a threshold at which to segment a picture has been demonstrated. If a picture contains an object and background occupying different average gray levels, the adaptive iterative approach discussed provides a simple automatic method for optimum threshold selection [17].

### 3.1.1 Adaptive Thresholding

Global thresholding uses a constant threshold  $T$  for all pixels in the image irrespective of their intensity levels. This method would not be appropriate to this research proposal as to detect edges in the images, variance in the intensity gradient value have to be noted. Both the adaptive and global thresholding methods are used to separate the foreground pixels from the background pixels based on the variation in the pixel intensities. Adaptive thresholding is used to find the threshold values in this research during the decision process.

The threshold is selected for each pixel based on the range of intensity values in its local neighborhood. Adaptive thresholding typically takes a gray scale or color image as input and outputs the binary image. For the scope of this paper, the intensity image is converted to the binary image, which is seen later in this chapter. The binary image contains the edge information as well as removes the background color. The threshold value is calculated for each pixel and depending on the threshold; value sets it to the background or foreground. If the pixel value falls below the threshold value then it is set to the background value otherwise to the foreground. When the threshold values starts to be constant, the process is halted. The algorithm summary is as follows (Figure 3.1).

- ❖ Initialize the positions of totally  $K$  ants, as well as the pheromone matrix  $\tau^0$ .
- ❖ For the building-step index  $n = 1 : N$ ,
  - For the ant index  $k = 1 : K$ ,
    - Consecutively move the  $k$ -th ant for  $L$  steps ,according to a probabilistic transition matrix  $p^n$  (with a size of  $M_1 M_2 \times M_1 M_2$ )
  - Update the pheromone matrix  $\tau^{(n)}$
- ❖ Make the solution decision according to the finalpheromone matrix  $\tau^{(N)}$ .

**Figure 3.1. The algorithm.**

All the  $K$  ants and the pheromone  $\tau^0$  matrix are initialized and for each ant are moved based on the probabilistic transition matrix  $p^{(n)}$ . Once, the iteration of all the ants is finished the pheromone matrix is updated to  $\tau^{(n)}$  which makes the final decision about the optimal solution found. Initially  $k$ -ants move from node  $i$  to node  $j$  based on the possibility (adjacent nodes). The pheromone information from node  $i$  to node  $j$  is  $\tau_{i,j}^{n-1}$ . According to [4] the probabilistic transition matrix is defined as

$$p_{i,j}^n = \frac{(\tau_{i,j}^{n-1})^\alpha (\eta_{i,j})^\beta}{\sum_{j \in \Omega_i} (\tau_{i,j}^{n-1})^\alpha (\eta_{i,j})^\beta} \quad (3.2)$$

$\Omega_i$ ; is the neighbor node of ant  $a_k$ ; assuming that the ant is on node  $i$ . The values  $\alpha$  and  $\beta$  are fixed values throughout the algorithm but by varying them the quality of the output is varied.  $\eta_{i,j}$ ; represents the heuristic information from node  $i$  to  $j$  which is also a fixed value set at the beginning of the algorithm. Then, the updating process is started where the pheromone matrix is updated after each  $k^{\text{th}}$  ant is moved and after all the  $k$  ants have moved within each building step [19]. Updating the matrix after the movement of the  $k^{\text{th}}$  ant is

$$\tau_{i,j}^{n-1} = \begin{cases} 1 - \rho \cdot \tau_{i,j}^{n-1} + \rho \cdot \Delta_{i,j}^k, & \text{if (i,j) belongs to best result,} \\ \tau_{i,j}^{n-1}, & \text{otherwise} \end{cases} \quad (3.3)$$

where  $\rho$  is the evaporation rate. The best result can be either the best result in the current building step or the best result since the start of the algorithm or the combination of both. It is a user-defined option, but throughout this paper, we consider it the best results at the building step. After each building step i.e. all the ants have moved, the pheromone matrix is updated as,

$$\tau^n = (1 - \psi) \cdot \tau^{n-1} + \psi \cdot \tau^n \quad (3.4)$$

where,  $\psi$ ; is the decay coefficient. In paper [4], both the updates Eq. (3.3) and Eq. (3.4) are performed

One proposed optimization, applied to a problem other than edge detection in an image, was to perform just one update operation [20], i.e. Eq. (3.4). We tried implementing this refinement in our edge detection research, but it yielded less satisfactory results. We update after each ant is moved and after all  $K$  ants within each building step have made their moves. By reducing one of the update steps, the algorithm did seem to perform faster but with information lost. The quality was subjected to further attempts at improvement under the one-update optimization by varying the initialized parameters such as  $\alpha$ ,  $\beta$ ,  $\lambda$  and the

heuristic information. The only difference attained was that the post-detection image appeared somewhat brighter, but with the same degradation in the quality of detected edges. The most useful results of this work is that there is a tradeoff between performances and number of updates, but that this tradeoff—at least in the context of image edge detection—is not worth the reduction in quality.

### **3.1.2 Understanding How the Ants are Finding Edges**

When  $K$  ants are located on the target image randomly (realizing that by “ant” here we really mean simply a sampling point in the space that will ultimately have impact on the changing entries in the pheromone matrix), each “ant” occupies one pixel. Then the pheromone matrix is initialized. Every move of an “ant” (i.e. considering some neighboring cell’s value) is based on a transfer probability, which dictates via the pheromone matrix and the heuristic information the next location to consider. The heuristic information is proportional to local changes of the image gray scale values. Ants “move towards the direction with larger gray scale change. At the end, ants (pixels of high selection probability from the paths implicit in the pheromone matrix) will concentrate on the edges. Since each move induces changes in the pheromones on the route, larger pheromone value will occur on the image edge and this is how the pheromone matrix reflects the “ants” and their “movements”.

## **3.2 DESIGN DETAILS**

The ACO based edge detection approach utilizes the ACO algorithm to detect the edges of an image by propagating ants to the pixels of the image. The image is considered to be a 2-D graph whose nodes are image pixels. The ants move from pixel to pixel to build the pheromone matrix, which represents the edge information of each pixel. The image’s intensity value is used to control the movements of the ants. The approach starts with the initialization process followed by the building process and the update process. The pheromone matrix is built by iteratively moving the ants  $N$  times, alternating between the building and the update processes. Finally, the edge is detected in the Decision process. The detail explanations of all the processes are as follows:

### 3.2.1 Initialization Process

The parameters  $\alpha$  and  $\beta$  are initialized. The heuristic information is set. The number of ants is  $K: \sqrt{M_1 \cdot M_2}$  where  $M_1$  is the length,  $M_2$  is the width of the image  $I$ . All the  $K$  ants are propagated on the 2-D image  $I$  such that at most one ant is on each pixel. Every pixel in the image is a node and the initial value of the pheromone matrix  $\tau^0$  is set to a constant value.

### 3.2.2 Building Process

At each building step, an ant, which is chosen from the  $K$  ants, moves  $L$  steps on the image  $I$ . The ant say  $a_k$  moves from node  $(l, m)$  to its neighboring node  $(i, j)$  according to the probabilistic transition matrix defined as,

$$p_{(l,m),(i,j)}^n = \frac{(\tau_{i,j}^{n-1})^\alpha (\eta_{i,j})^\beta}{\sum_{(i,j) \in \Omega_{(l,m)}} (\tau_{i,j}^{n-1})^\alpha (\eta_{i,j})^\beta} \quad (3.5)$$

Where,  $\tau_{i,j}^{n-1}$  is the pheromone value of node  $(i, j)$  and  $\Omega_{(l,m)}$  is the neighboring nodes of  $(l, m)$ . In order words,  $\Omega_{(l,m)}$  is all the pixels that can be in the 8-neighborhood of the pixel  $(l, m)$ . The heuristic information of the node  $(i, j)$  is  $\eta_{i,j}$ . In order, to determine the heuristic information [20], the local configuration at each pixel  $(i, j)$  is defined as,

$$\eta_{(i,j)} = \frac{1}{N} H_c(I_{(i,j)}) \quad (3.6)$$

Where,  $N$  is a normalization factor used to isolate error and defined as

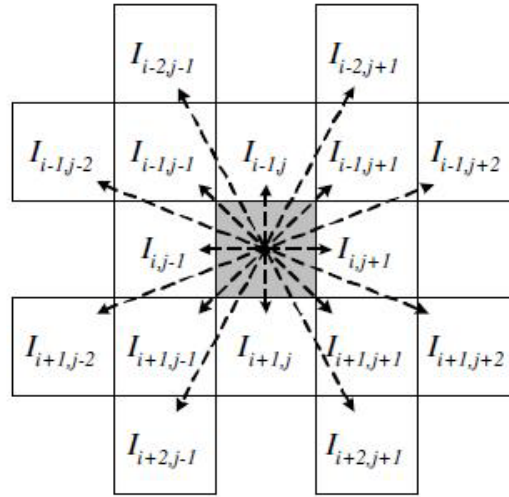
$$\sum_{i=1:M_1} \sum_{j=1:M_2} H_c(I_{i,j}).$$

Where,  $I_{i,j}$  represents the intensity value of the pixel  $(i, j)$  of image  $I$ . The variation of the image's intensity values depends on  $c$  which a group of pixels which are similar in some form. This group of pixels forms the function  $H_c(I_{(i,j)})$ .

As shown in Figure 3.2, the function  $H_c(I_{(i,j)})$  depends on its neighboring group of pixels  $c$  which is defined as,

$$H_c(I_{i,j}) = f(|I_{i-2,j-1} - I_{i+2,j+1}| + |I_{i-2,j+1} - I_{i+2,j-1}| + |I_{i-1,j-2} - I_{i+1,j+2}| + |I_{i-1,j-1} - I_{i+1,j+1}| + |I_{i-1,j} - I_{i+1,j}| + |I_{i,j-1} - I_{i,j+1}| + |I_{i,j} - I_{i+1,j}| + |I_{i,j} - I_{i-1,j}|) \quad (3.7)$$

The function ensures that sharp turns in the image are less likely than small angle turns. Thus, each ant in the colony has the tendency to move in the forward direction. In order to change the respective shapes the function Eq. (3.7) is modified mathematically by the following,



**Figure 3.2. Neighbors of pixel (i, j).**

$$\lambda x, \text{ for } x \geq 0 \quad (3.8)$$

$$\lambda x^2 \text{ for } x \geq 0 \quad (3.9)$$

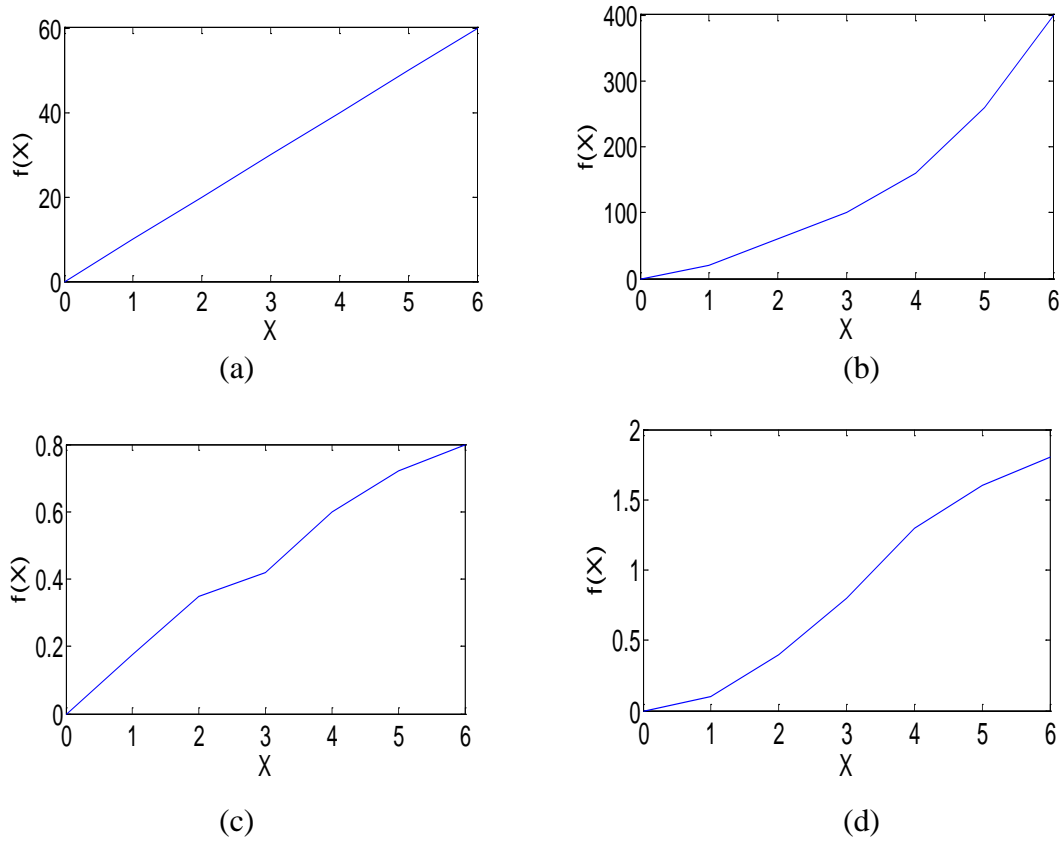
$$\begin{cases} \sin(\pi x / 2\lambda), & 0 \leq x \leq \lambda \\ 0 & \text{otherwise} \end{cases} \quad (3.10)$$

$$\begin{cases} (\pi x) \sin(\pi x / \lambda) & 0 \leq x \leq \lambda \\ 0 & \text{otherwise} \end{cases} \quad (3.11)$$

These equations are applied alternatively to the function Eq. (3.7) for different magnitudes of the changes in the direction at each step and the  $\lambda$  takes discrete values for respective shapes. Varying the values of  $\lambda$ ,  $\alpha$  and  $\beta$  play as an important influence in producing optimal results (refer to Figure 3.3). Hence, the values of  $\alpha$  and  $\beta$  are to be carefully chosen keeping them towards the lower range. In order, to find the range of  $\Omega_{(l,m)}$  i.e. the movements of the ants on the image, the 4-connectivity or the 8-connectivity neighboring feature is used (The detail explanation is given in Chapter 2, Section 2.6.1).

### 3.2.3 Update Process

The proposed paper performs both the update steps, one after each ant has moved and the other after all ants on each building step have moved, on opposed to only one update step in many other papers on the similar topic. An attempt to alter the algorithm to one update process yielded a binary image with missing information.



**Figure 3.3. The graphs for  $\lambda=10$  for (a) Eq. 12, (b) Eq. 13, (c) Eq. 14, (d) Eq. 15.**

The update process, which updates the pheromone matrix after each ant is moved, is

$$\tau_{i,j}^{n-1} = \begin{cases} (1 - \rho) \cdot \tau_{i,j}^{n-1} + \rho \cdot \Delta_{i,j}^{(k)} & \text{if (i,j) is visited by the kth ant} \\ \tau_{i,j}^{n-1} & \text{otherwise} \end{cases} \quad (3.12)$$

where,  $\rho$  is defined in Eq. (3.2) and  $\Delta_{i,j}^k$  is calculated by the heuristic information and

$$\Delta_{i,j}^k = \eta_{i,j}.$$

The heuristic information is added into the ant's memory and used for further steps.

The second update is made at the end of each building step i.e. all the ants  $K$  within the step have moved. Since all the ants have moved at the end of the building step, the equation is

$$\tau_n = (1 - \psi) \cdot \tau^{(n-1)} + \psi \cdot \tau^0 \quad (3.13)$$

where,  $\psi$  is the decay coefficient.

The pheromone matrix is updated at this stage with the consideration of the decay coefficient and the pheromone matrix built until now.



### 3.2.4 Decision Process

Being a very important process as it incorporates the results from the previous steps to determine if at each pixel it is an edge or not. In order to find out about the edge information, a threshold value  $T$  is used on the pheromone matrix  $\tau^N$ . The iterative method proposed in [17] is used to compute  $T$ . To convert the intensity image to binary image, a normalized intensity value in the range of  $[0, 1]$  is considered. Using the starting threshold value, the histogram is segmented into two parts. The mean of the gray values associated with the foreground pixels and the sample mean of the gray values associated with the background of the pixels are computed. This new threshold value is considered as the average of the two samples. This process is recurred based on the new threshold value until there is no change in the value noted. The initial threshold value  $T^{(0)}$  is considered as the mean value for the pheromone matrix. Each index value of the pheromone matrix is segregated as below the initial threshold value or above the threshold value. Based on these two categories the average of the mean values is computed which is the new threshold value. As, mentioned earlier this process is repeated till the threshold value becomes constant. The process shown in Figure 3.4 is summarized as follows:

1. Initialize  $T^{(0)}$ ,

$$T^0 = \frac{\sqrt{\sum_{i=1:M_1} \sum_{j=1:M_2} \tau_{i,j}^N}}{M_1 M_2} \quad (3.14)$$

where,  $M_1$  and  $M_2$  are the size of the image.

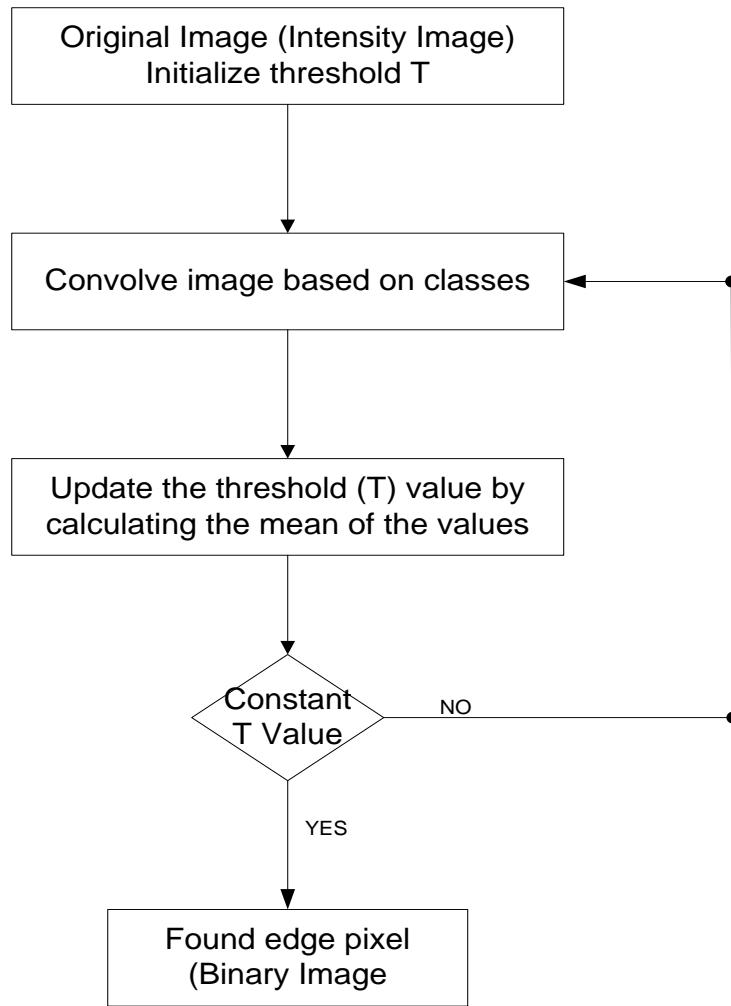
The initial threshold value in the iterative method is always the mean of the size of the image as mentioned in Chapter 2. The mean of the pheromone matrix will be the sum of all the entries of the matrix by the size of the matrix.

2. Diving the values of the pheromone matrix into two classes based on the threshold value  $T^{(l)}$ . All the values of  $\tau$  which are lesser than  $T^{(l)}$  are class  $C_1$  and the values greater than  $T^{(l)}$  are class  $C_2$ .

$$C_1^l = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} \tau_{i,j}^N}{\sum_{i=1:M_1} \sum_{j=1:M_2} 1} \quad (3.15)$$

$$C_2^l = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} \tau_{i,j}^N}{\sum_{i=1:M_1} \sum_{j=1:M_2} 1} \quad (3.16)$$

where  $C_1$  and  $C_2$  are the two classes defined to divide the pheromone matrix values based the range of threshold value.



**Figure 3.4. Adaptive thresholding process.**

3. Iterate the loop to  $l=l+1$ ; and calculate the mean between the new values of two classes and update the threshold,

$$T^{(l)} = \frac{c_1^l + c_2^l}{2} \quad (3.17)$$

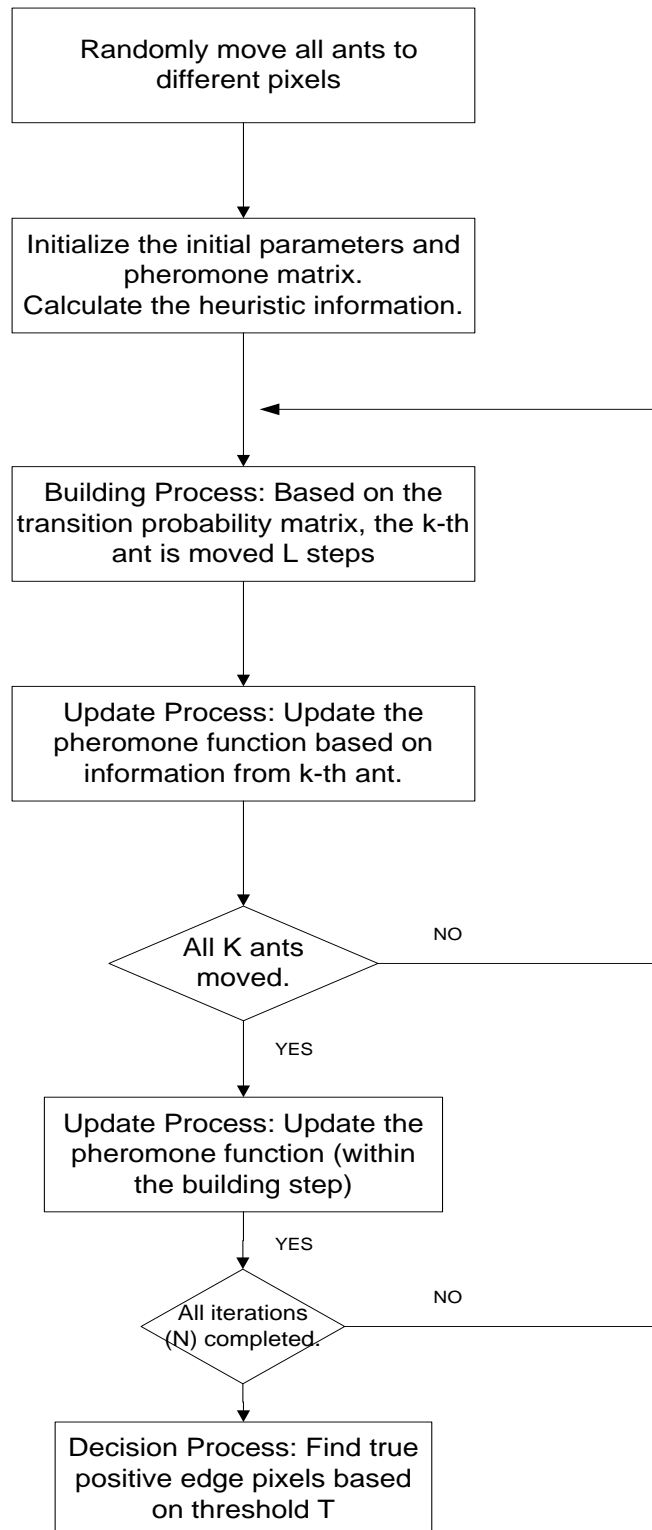
where,  $T^{(l)}$  is the new calculated threshold value.

4. Check if the new threshold  $T^{(l)}$  value is equal to  $T^{(n-1)}$ . If false, then go to step 2, otherwise the process is terminated and the threshold value is recorded. To determine if an edge  $E_{i,j}$  is found at the pixel  $(i, j)$ ,

$$E_{i,j} = \begin{cases} 1 & \text{if } \tau_{i,j}^N \geq T^l \\ 0 & \text{otherwise} \end{cases} \quad (3.18)$$

where,  $E_{i,j}$  is an edge at pixel  $(i, j)$ .

The detailed view of the implementation process is shown in Figure 3.5.



**Figure 3.5. Flowchart of implementation details.**

### 3.3 IMPLEMENTATION

The implementation is done in Matlab as it was found to be the most cost effective approach as well as simple to implement. The development process was hastened because of Matlab programming. The images used for testing were the bitmap images of various sizes (128 x 128, 256 x 256 and 512 x 512).

The algorithm is designed to convert an intensity image into binary image. The initialization phase initializes the number of ants based on the size of the image given as input. The implementation of the probability transition matrix in the building phase was done as shown in Figure 3.6.

```
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 3.6. Implementation of probability transition matrix.**

The update phase includes two steps. The first update is performed after each ant has moved as shown below in the Figure 3.7. The pheromone matrix is updated. In simple combinatorial problems like TSP, the update operation is limited to once during the entire process i.e. after each building step. In the proposed approach, reducing the update operation to one, lead to important edge information loss.

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

**Figure 3.7. Implementation of first update step.**

The final update is done within each building step after all ants have moved as shown in Figure 3.8.

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

**Figure 3.8. Implementation of final update step.**

The last step being the decision phase is to decide at each pixel whether it is an edge or not. This binary decision is based on [17]. The process of finding the threshold value which is applied to the pheromone matrix for the edge determination is a part of the adaptive approach developed by [17]. The decision process has been defined earlier and the detailed implementation is shown in the Figure 3.9. The histogram is initially segmented into two parts using a starting threshold value such as  $0=2B-1$ , half the maximum dynamic range. With the built-in function of Matlab ‘hist’, the image and the length of the color map is passed. Since, we pass an intensity image; the color map value is 256. For a binary image the color map value can be only 2. The approach developed in [17] finds the threshold value by dividing the values of image I into two classes. Lower value classes and upper values classes and finds the mean value. At end of the loop, the color map value reaches 2 thus, producing the corresponding binary image.

The neighborhood of the current position reached by an ant is determined based on the 4-connectivity and the 8-connectivity graphs. The 4-connectivity method: The implementation details and the overview is as shown in Figure 3.10.

In the Figure 3.11, the current pixel (row, column) which is the ant’s position is shown. To determine the movement of the ant’s next position, the 4-connectivity graph is used. The four neighbors of the current pixel are found such that at least one edge of the current pixel touches the neighboring pixels.

The 8-connectivity method: The implementation details and the overview is as shown in Figure 3.12.

Another approach used to determine the neighbors of the current pixel to determine the ant’s movement is 8-connectivity graph shown in Figure 3.13. The eight neighbors of the current pixel are found such that at least one edge or corner of the current pixel touches the neighboring pixels. Thus, we have eight neighbors.

```

I = I(:);
% Compute mean intensity of image from histogram, set T=mean(I)
[counts, N]=hist(I,256);

% Initialization for iteration loop
i=1;
% Calculating the cumulative sum of all values of T
cumulativeSum =cumsum(counts);%cumulative sum eg a[1:3] ans = [1 3 6]
% Initialization of the Threshold array
T(i)=(sum(N.*counts))/cumulativeSum(end);
%Step 2
%Compute the mean of the class valuesbelow the threshold values T
cumulativeSumOfLowerClass =cumsum(counts(N<=T(i)))
M2=sum(N(N<=T(i)).*counts(N<=T(i)))/cumulativeSumOfLowerClass(end);
% Compute the mean of the class values above the threshold values T
cumulativeSumOfUpperClass=cumsum(counts(N>T(i)));
M1=sum(N(N>T(i)).*counts(N>T(i)))/cumulativeSumOfUpperClass(end);
% Find total mean T value for each iteration
i=i+1;
T(i)=(M1+M2)/2;
% Comparing the values of T from iteration I and (i-1). If T(i) is lesser value then
repeat Step 2
Threshold=T(i);
while abs(T(i)-T(i-1))>=1
    cumulativeSumOfLowerClass=cumsum(counts(N<=T(i)));
    M2=sum(N(N<=T(i)).*counts(N<=T(i)))/cumulativeSumOfLowerClass(end);
    cumulativeSumOfUpperClass=cumsum(counts(N>T(i)));
    M1=sum(N(N>T(i)).*counts(N>T(i)))/cumulativeSumOfUpperClass(end);
    i=i+1;
    T(i)=(M1+M2)/2;
    Threshold=T(i);%Final value
end

```

Figure 3.9. Implementation to compute threshold T.

```

if search_clique_mode == '4'

    row = ant_current_row_idx;

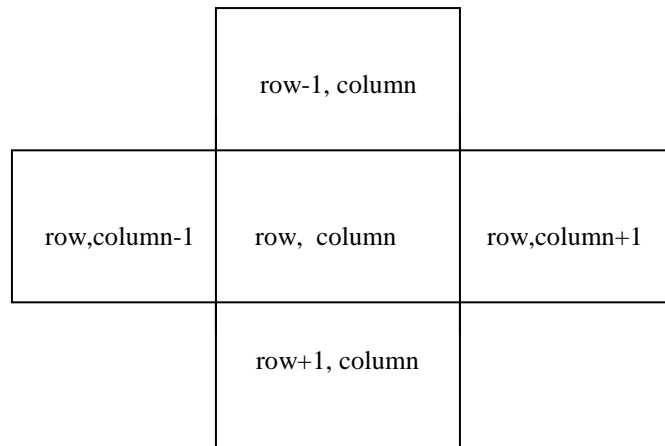
    column = ant_current_col_idx;

    ant_search_range_temp = [row-1 column; row column+1; row+1 column; row
    column-1];

end

```

Figure 3.10. Implementation of 4-connectivity.



**Figure 3.11.** All four neighbors of the pixel (row, column).

```

if search_clique_mode == '8'
row = ant_current_row_idx;
column = ant_current_col_idx;

ant_search_range_temp = [row-1 column-1; row-1 column; row-1 column+1; row
column-1; row column+1; row+1 column-1; row+1 column; row+1 column+1];

end

```

**Figure 3.12.** Implementation of 8-connectivity.

|                     |              |                     |
|---------------------|--------------|---------------------|
| row-1,column-1<br>1 | row-1,column | row-1,column+1      |
| row,column-1        | row,column   | row,column+1        |
| row+1,column-1<br>1 | row+1,column | row+1,column+1<br>1 |

**Figure 3.13.** The eight neighbors of the pixel (row, column).

## CHAPTER 4

### EXPERIMENTAL RESULTS

ACO has been applied in many areas which fueled the interest in this proposal. This section shows the experimental results and also gives an overview of all the expected, unexpected, results of the algorithm. The proposed technique is compared with the results from various other algorithms using Ant Colony Optimization approach to select the edge of the images as well as with the results from traditional edge detectors such as Canny, Prewitt and Sobel. Most of the edge detectors are executed by using MATLAB toolbox. The proposed approach of this research project has also been executed in MATLAB. The resultant output, the binary image is on a white background with the edge pixels in black. The experiments were conducted to evaluate the performance of this approach against the other approaches using five images: Camera, Lena, Peppers, House and Madonna in different sizes (128 x 128, 512 x 512). The various parameters set are shown in Table 4.1. The results will also be compared with Nezamabadi-Pour *et al.*'s edge detection method [2]. Before any experiments can be performed, the system requires some input to act on. The input would be a intensity image. The image is stored in the same folder where the script is stored. The output of the implementation which is also an image is written back to the same folder. At the first level of experiments, an input image of 128 x 128 was fed into the algorithm and latter upgraded to a higher dimensions such as 256 x 256 and 512 x 512. Figure 4.1 shows the initial test run of the algorithm with the cameraman image of 128 x 128 dimensions.

The input is an intensity image and the output is the monochromatic image. The tower in the original picture is blended well with the background of the image and also, it is not within the focus point. The initial run was done with arbitrarily set parameters and hence resulted in some lost information. After taking a closer look at setting each parameter based on the initialization of [2], [3], [17], [18], a set of parameters for the initialization process were fixed.

As shown in Table 4.1, the list of all the parameters initialized either at the initialization process or applied as a supporting parameter to other equation during the



**Table 4.1. Parameters Used in the Experiment**

| Parameters    | Values<br>Initialized                   | Description  |
|---------------|---|--|
| $K$           | $\lfloor \sqrt{M_1 \times M_2} \rfloor$ | The total number of ants is the size of the image.                                 |
| $\tau_{init}$ | 0.0001                                  | The initial value set to every index of the pheromone matrix.                      |
| $\alpha$      | 1                                       | The weighing factor of the pheromone value in Eq. (3.1)                            |
| $\beta$       | 0.1                                     | The weighing factor of the heuristic information in Eq. (3.1)                      |
| $\Omega$      | 8-connectivity                          | The permissible ant movement range in Eq. (3.1); shown in Figure 3.13              |
| $\lambda$     | 1                                       | The factor used to adjust the various shapes in Eq. (3.8)-(3.11)                   |
| $\rho$        | 0.1                                     | The evaporation rate in Eq. (3.12)   |
| $L$           | 40                                      | The total number of ant's movement steps within each building step                 |
| $\square$     | 0.1                                     | The user defined threshold value used for determining the mean threshold value $T$ |
| $\Psi$        | 0.05                                    | The pheromone decay coefficient in Eq. (3.13)                                      |
| $N$           | 4                                       | Total number of building steps   |



(a)

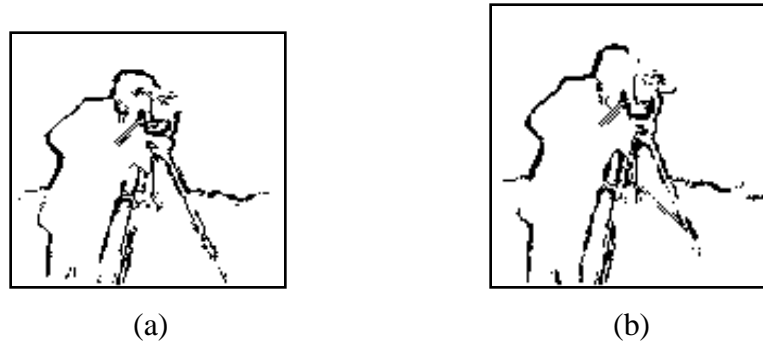


(b)

**Figure 4.1. (a) The input image, intensity image, (b) the output image, the binary image.**

building, update and decision processes. These parameters were chosen on trial and error basis and with the help of the results and calculations from other researchers who have worked on this area [2], [3], [4], [7], [9], [10], [11], [12], [15], [17], [18]. A significant number of experiments were carried out to study the influence of these parameters. The determination of these parameters plays an important role in the quality of the binary image. The variations of these parameters and the experimental results will be shown in the next subsection. Among all the research work done in this area to use ACO algorithm for image

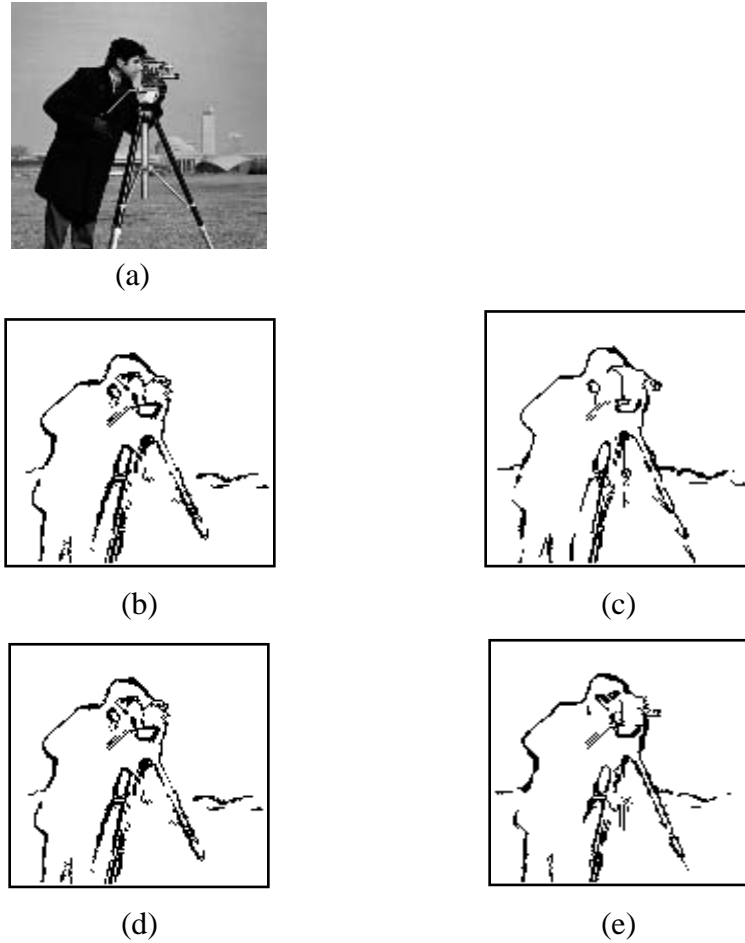
edge detection, the best results are produced by H. Nezamabadi-Pour's method. The results can be seen in Figure 4.2. The aim of this project was to implement Jing Tian's algorithm by using MATLAB and produce results similar to what is presented in his paper along with modifying the parameters to produce results as close as possible to Hossein Nezamabadi-pour's algorithm. The number of steps and the number of ants are chosen such that they are proportional to the root of the pixel numbers i.e.  $\sqrt{M_1 M_2}$  as mentioned [4]. The post-processing methods- thinning and global thresholding to remove irrelevant edges have been omitted. Hence, the comparison of the proposed output and the [2] output are done against the binary images before the processing. The algorithm is first applied to the image 'cameraman' in different sizes and the output is as shown in Figure 4.2.



**Figure 4.2. The tests run of images in different dimensions. (a) 128 x 128, (b) 512 x 512.**

In Figure 4.3, the variation in  $\alpha$  and  $\beta$  parameters are shown.  $\alpha$  and  $\beta$  are the weighing factors for the pheromone information on each pixel and the weighing factor for the heuristic information(visibility) respectively. Therefore, by increasing the constant  $\alpha$ , the ants choose edges which already have some pheromone on it, i.e. edges previously chosen by other ants. And, decreasing the  $\beta$  value increases the visibility of the images. The values chosen in our project worked well for all the images so far tested. The evaporation rate is always set to 0.1.

Another important factor is the parameter  $L$  which is the ant movement's building steps. This was a critical decision to make as when a small value was chosen it lead to stop the algorithm abruptly and the larger value lost edge information in the binary image. To change the respective shapes of the image, and show the importance of determination of the heuristic information in Eq. (3.6), Eq. (3.8)-(3.11) are developed which are applied to Eq. (3.7).



**Figure 4.3. (a) Test run image, varying  $\alpha$  value, (b)  $\alpha=10$ , (c)  $\alpha=20$ , varying  $\beta$  value., (d)  $\beta=0.1$ , (e)  $\beta=0.3$  with Eq. (3.4).**

Modifying these equations along with varying the parameters  $\alpha$  and  $\beta$  produce different results. In order to achieve good results for all the equations, the value of constants  $\alpha$  and  $\beta$  were varied and based on trial and error, a conclusion was made which can be seen in Figure 4.3. The threshold value  $T$  was calculated as the mean of the final intensity values over all nodes by applying the adaptive iterative selection method.

As shown in Figures 4.4 to 4.8 and visually comparing the results of the intensity image converted to the binary images by applying the four equations defined in Section 3.2. The results of Eq. (3.8) and Eq. (3.10) produce close to satisfactory results. The binary image produced as a result of Eq. (3.9) and Eq. (3.11) can fall easily into the category of bad results. These results have resulted in losing information. The information of the edges is either lost or was unable to be differentiated as an edge. This is an example of the false positives or



(a)



(b)

**Figure 4.4. (a) Input image of cameraman, (b) input image of Lena.**



(a)

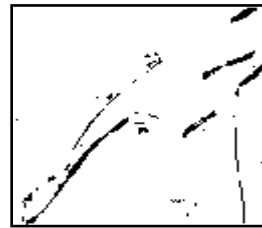


(b)

**Figure 4.5. Binary images of cameraman and Lena by using Eq. (3.8).**



(a)

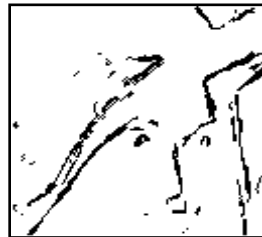


(b)

**Figure 4.6. Binary images of cameraman and Lena by using Eq. (3.9).**

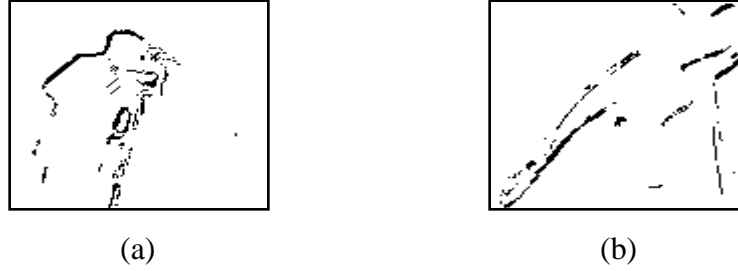


(a)



(b)

**Figure 4.7. Binary images of cameraman and Lena by using Eq. (3.10).**



**Figure 4.8. Binary images of cameraman and Lena by using Eq. (3.11).**

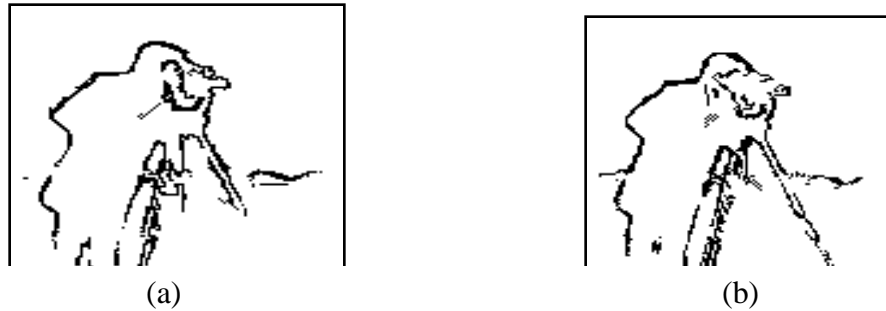
false negatives as explained in Chapter 2. In addition, something to notice is that the binary images of cameraman are far better than Lena. In the original image of Lena one can notice that her hat is not clearly differentiated from the boundary of the image as well as there aren't any changes in the color of her hat thus leaving the algorithm to consider the region of her hat till the boundary as the background of the image. The importance of the image quality is of at most interest as our algorithm attempts to differentiate between the foreground and background with the variations in the color map, which helps with finding the edges.

Some other changes in the results were noted when the method of neighborhood connectivity between the current pixel point and its neighbors were changed. The only two methods used for testing in this paper are the 4-connectivity neighborhood and 8-connectivity neighborhood. For each pixel, its neighbors using the 4-connected method would be all those pixels around the current pixel, which shared an edge with the current pixel. Similarly, in case of the 8-connectivity method, the current pixel and its neighbors have to share either their edges or corners. Since, the proposed approach only applies to 2-D images; we would not consider the other neighborhood connectivity methods.

Figure 4.9 shows the different neighborhood connectivity applied to the same image; *Cameraman*. The background of both the methods are good whereas, when comparing the edge of the image, there is loss of some parts of data in the 4-connectivity. Also, there are some false positive edges seen in the 4-connectivity applied image.

## 4.1 BEST RESULTS

Nezamabadi-Pour's method is one of the best edge detectors using ACO. The method was also the first to use ACO in edge detection. The approach is very similar to the proposed method but, Nezamabadi-Pour's approach updated the pheromone matrix only once i.e. after



**Figure 4.9. (a) 4-connectivity applied, (b) 8-connectivity applied.**

all  $K$  ants have moved within each building step. Whereas in the proposed approach, the update is performed after each ant has completed its move. And, update again when all the ants have moved within each building step. However, the other difference between our approach and the Hossein Nezamabadi-pour's approach is that the later uses the global thresholding method at the end of the algorithm to discard irrelevant edges from the decision process. A morphological thinning method is applied to thin the output image [2].

Figure 4.10 shows the images used to test the results of both Hossein Nezamabadi-Pour's approach and Jing Tian's approach. A significant change in the quality of the binary image is seen by applying the threshold and thinning algorithms. The results and performance of this method is one of the best results and are referred by all the other researchers as a model. To make a fair comparison all the results shown are before the thinning algorithm is applied. The results are shown in Figure 4.11 [2]. Learnt from [2] is that the computation time of this algorithm is significantly high in comparison to other approaches.



(a)



(b)

**Figure 4.10. Images of Lena and peppers.**



(a)



(b)

**Figure 4.11.** The result images of Nezamabadi-Pour's method. (a) Lena, (b) peppers. Source: H. Nezamabadi-Pour, S. Saryazdi, and E. Rashedi. Edge detection using ant algorithms. *Soft Computing*, 10:623-628, 2006.

Another approach which produces good results is the Jian Zhang's approach uses a combination of gradient and relative difference of statistical means for edge detection. The gray gradient value of a pixel in an image is tracked by the maximum variation of gray value in the intensity of the image. The gradient feature is very simple to achieve but, is subjective to noise and the texture quality. The statistical means is used to estimate the categories of the pixel which have a strong ability to reduce the noise but, might not provide the entire edge information. Combining both these features Jian Zhang's approach produces the edge information. This approach achieves good results (as shown in Figure 4.12) but, produces an overhead of calculating the statistical means for each pixel  $(i, j)$ .



(a)

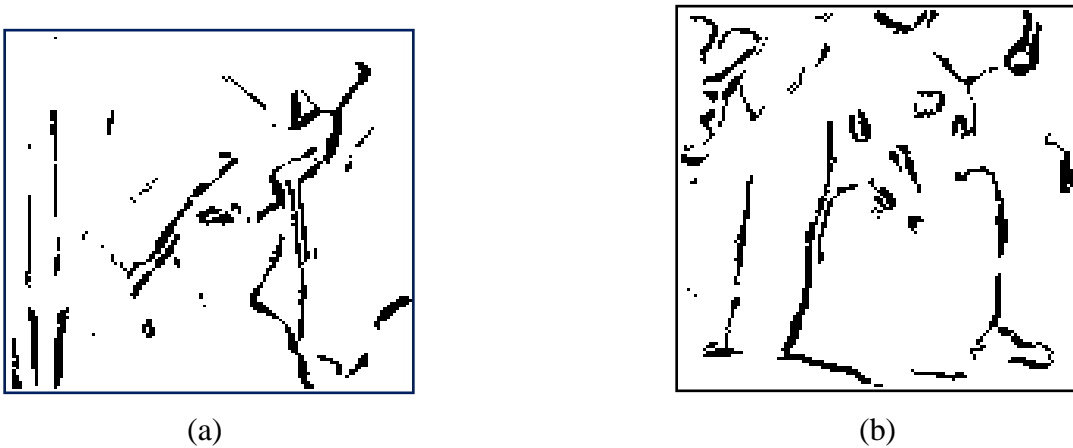


(b)

**Figure 4.12.** The resultant images of Jian Zhang's method. (a) Lena, (b) peppers.

## 4.2 COMPARISON OF RESULTS

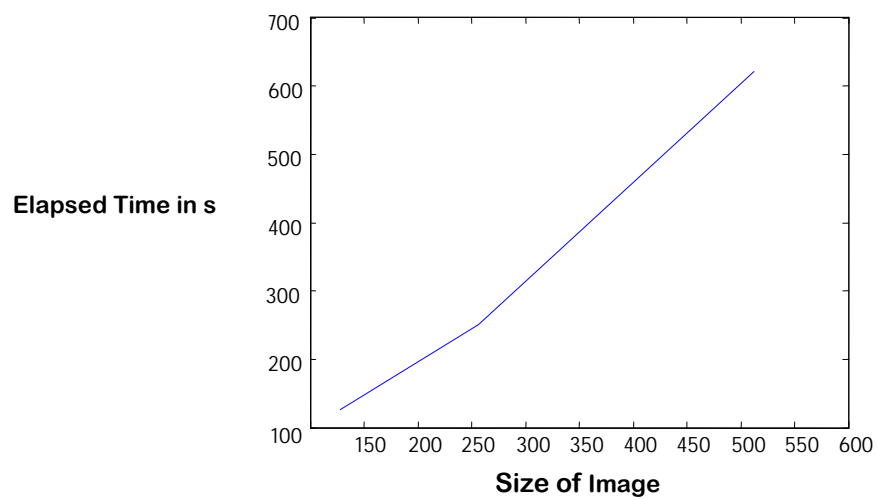
In Section 4.2, some of the best results of the ACO-edge detection method are shown. Both the approaches produce substantiate results. In comparison to the approaches [2] and [3] our proposed approach's results are shown. Our approach does not dilute the ACO algorithm by adding the gradient and relative difference of statistical means for edge detection. Also, due to time constraints; our algorithm neither considers the effects of noise and texture, nor the thinning algorithms as applied in [3]. Figure 4.13 shows the result of the images Lena and Peppers. There is strong belief that by adding the thinning methods and considering the texture, there would be a substantiate improvement in the performance of the algorithm.



**Figure 4.13. The binary images of proposed method. (a) Lena, (b) peppers.**

Based on the dimensions of the input image, there is a significant increase in the computation time of the algorithm. The dimension is directly proportional to the computation time.; while this is not factored into the time units on the diagram, it still provides a conceptual model. An increase in the computation time was witnessed when the dimensions of the images very 512 x 512 and higher. For images for very high dimensions, the algorithm took much longer and some extra load on the processor was noted. The proposed ACO based edge detection approach is implemented using Matlab programming language and run on a PC with a Intel Core™ DUO 2.13 GHz CPU and a 1 GB RAM and also on Macintosh computers. The computational time for the image cameraman of 128 x 128 dimension was 126.138771 seconds, 250.242513 seconds, 620.637727 seconds respectively as applied on Eq. (3.8) and as shown in Figure 4.14.





**Figure 4.14. Dimensions 128 x 128, 256 x 256, 512 x 512 vs. computation time.**

## **CHAPTER 5**

### **CONCLUSION**

In the Chapter 1, the introduction of this area, we have expressed to prove a set of expectations fulfilled by other fellow researchers in this area of research. In Chapter 2 we have gone through the relevant literature survey in some level of detail. In Chapter 3 on system design overview, we have set the list of expectations from Chapter 1 as implementation goals for the algorithm implemented. In Chapter 4, we have discussed the experimental results achieved with our implementation and compared it with all the other results from various other papers in this area.

#### **5.1 SUMMARY**

This paper introduces and claims that the results are true for solving optimization problems as well as a renowned problem in the field of image processing being image edge detection. The general idea behind the Ant System paradigm is that a population of agents each guided by a process is directed by greedy force whereas when considering an agent alone, the underlying process would tend to make the agent converge to a suboptimal tour with exponential speed [21]. With the inspiration gained from the Ant System, the Ant Colony Optimization was developed. The paper strongly proves that all the attempts of proving that ACO is an efficient approach for image edge detection is successful. The proposed approach reduces the complexity involved in some other approaches with two update operations. It yields similar performance to that of the existing edge detection algorithms using ACO [4] as it has been verified in Chapter 4. As already mentioned in Chapter 2, tremendous amount of research in this area has been fulfilled. Some major areas being: studies of social animal behavior; research in “natural heuristic algorithms” and stochastic optimization. The research on behavior of social animals is to be considered as a source of inspiration and as a useful metaphor to explain our ideas.

## 5.2 OUTCOME

The approach explained throughout this paper is very promising due to its generality, and its effectiveness. The results are close to accuracy to many difficult problems.

The main contributions of this paper are:

- The ACO algorithm is employed to solve the image edge detection problem which produces a desirable output when compared to many other algorithms and approaches.
- The information of the intensity image is remained intact while converting it to a binary image.
- While certain proposed and imagined refinements at local optimization within the algorithm may reduce time-to-solution, our additional experimentation suggests that this is likely to reduce the quality of the output.

We show how the algorithm can be used in many different areas not only restricting itself to image processing. The paper proves that ACO has not only limited to solving combinatorial problems but also stood high in performance towards other problems such as TSP, QAP and lead to many other important researches in computer science.

The architecture, design and the development of the algorithm have been developed in such a way that the further feature-additions and other development may be accomplished with the minimum effort.

## 5.3 FUTURE WORK

From our discussions in the preceding chapters as well as with our implementation and experimental results, we can confirm that the approach yields results up to our satisfaction. However, we have identified some other changes to the underlying Ant Colony Optimization algorithm to increase the usability and efficiency. The parallel ACO algorithm can be applied to reduce the computational load even further. Using a highly powered machine could improve the performance even with the existing algorithm. The improvement to the underlying algorithm to reduce the computation time would be to reduce the update operations to one as in [3]. It would also be of interest to explore whether the attempted optimizations in this research could be effective in other problem domains.

## BIBLIOGRAPHY

- [1] J. Lewis and J. Lawson. Starcat: An architecture for autonomous adaptive behavior. *Proceedings of the 2004 Hawaii International Conference on Computer Science*, Honolulu, HI, 2004. IEEE.
- [2] H. Nezamabadi-Pour, S. Saryazdi, and E. Rashedi. Edge detection using ant algorithms. *Soft Computing*, 10:623-628, 2006.
- [3] A. Colomi, M. Dorigo, V. Maniezzo. Distributed optimization by ant colonies. *Proceedings of the First European Conference on Artificial Life*, Paris, France, 1991. Elsevier Publishing.
- [4] J. Tian, W. Yu, and S. Xie. An ant colony optimization algorithm for image edge detection. *IEEE Congress on Evolutionary Computation*, 2008:751-756, 2008.
- [5] Azi Sharif. Antcat. Master's thesis, San Diego State University, San Diego, CA, 2004.
- [6] F. Glover and M. Laguna. *Tabu search*. Kluwer academic publishers, Boston, MA, 1998.
- [7] M. Dorigo. Optimization, learning and natural algorithms. Doctoral dissertation, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.
- [8] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo and F. Glover, editors, *New ideas in optimization*, pages 11-32. McGraw-Hill, London, UK, 1999.
- [9] H. Holland. What is a Learning Classifier System, 2000. [code.ulb.ac.be/dbfiles/HolBooCol-et al2000lcs.pdf](http://code.ulb.ac.be/dbfiles/HolBooCol-et al2000lcs.pdf), accessed Mar. 2011.
- [10] M. V. Butz. Learning classifier systems. *Proceedings of the GECCO Conference Companion on Genetic and Evolutionary Computation*, London, United Kingdom, 2007. Springer.
- [11] J. H. Holland, L. B. Booker, and D. E. Goldberg. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235-282, 1989.
- [12] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, New York, 1989.
- [13] M. Mitchell. *Analogy-making as a complex adaptive system*. MIT Press, Cambridge, MA, 1993.
- [14] J. C. Russ. *The image processing handbook*. Boca Raton, FL: CRC Press, 1992.
- [15] A. V. Bateria and C. Oppus. Image edge detection using ant colony optimization. *WSEAS Transactions on Signal Processing*, 6(2):2-5, 2010.
- [16] R. Fisher, S. Perkins, A. Walker and E. Wolfart. Connected Component Labeling, 2003. <http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm#1>, accessed Dec. 2011.

- [17] T. W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE Trans. System, Man and Cybernetics*, 8:630-632, 1978.
- [18] J. Rahebi, Z. Elmi, A. Farzamnina, and K. Shayan. Digital image edge detection using an ant colony optimization based on genetic algorithm. *IEEE Conf. on Cybernetics and Intelligent Sys.*, 2010:145-149, 2010.
- [19] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE Computational Intelligence Mag.*, 1:28-39, 2006.
- [20] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. on Systems, Man and Cybernetics*, 26:29-41, 1996.
- [21] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, 1:53-66, 1997.