SPEED OPTIMIZED IMPLEMENTATION OF ANT COLONY OPTIMIZATION ALGORITHM FOR IMAGE EDGE DETECTION

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

By

Rashmi Moparthi

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

Major Department:
Computer Science

November 2015

Fargo, North Dakota

North Dakota State University
Graduate School

Title

SPEED OPTIMIZED IMPLEMENTATION OF ANT COLONY OPTIMIZATION ALGORITHM FOR IMAGE EDGE DETECTION

By

RASHMI MOPARTHI

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

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Advisor

Dr. Jun Kong

Dr. Susan Cooper

Approved by Department Chair:

11/06/2015 — Date

Dr. Brian Slator — Signature

ABSTRACT

Ant Colony algorithm (ACO) is an approach used to provide a solution to an optimization problem. ACO follows the mechanism adapted by Ants to search for optimal paths by performing combined activity of all ants in the colony. Ants adopt a probabilistic approach to solve problems of path discovery and alike. The behavior of ants has been mapped to a scientific algorithm to solve optimization problems. Different modified optimization variants have been run on the basic algorithm that resulted in in efficient and effective systems for solving different optimization problems including in the area of image processing. Study in this paper is applying ACO algorithm to solve the problem of image edge detection by modifying the algorithm to improve its efficiency and speed. The algorithm has been implemented in MATLAB and its speed has been enhanced by about 40-50 percent using the vectorization of different processes of the algorithm.

ACKNOWLEDGEMENTS

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

DEDICATION

I dedicate my paper to my parents Ramesh Moparthi, Satyavani Jampani and my Friends who have helped me all along this journey with their guidance and support.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

1. INTRODUCTION

Ant Colony Optimization (ACO) is a popular metaheuristic that is helpful in tackling complex computational issues and discovering proximate answers for difficult optimization problems. Ants in nature are noteworthy in discovering an ideal way to their destination. The ants have a tendency to meander arbitrarily from their hive looking for sustenance, and when they need to go, back to their hive they store a synthetic substance called the pheromone all through their way. Alternate ants take after the ways with pheromone content instead of meandering around various ways keeping in mind the end goal to achieve their purpose of interest (generally nourishment). Also, these ants expand the pheromone content on the trail. Since pheromone substance vanishes rapidly, a shorter course demonstrates to have a more grounded substance of pheromone as more number of ants navigate through it. Longer ways have a less number of ants going by, and henceforth, lighter pheromone content on them. In generally lesser time, a larger part of ants will be navigating the shorter way, which has the most elevated pheromone content. ACO adjusts this marvel of nature by utilizing fake ants, which are only a product specialists to discover great answers for a given optimization problem.

Alberto Colorni, Vittorio Maniezzo and Marco Dorigo were the first to think of a calculation to reproduce the ants' conduct, naming it an Ant Colony System (ACS) calculation, ordinarily known as Ant Colony Optimization and contracted as ACO. Subterranean insect Colony Optimization has made an imprint in different fields with noteworthy commitments to improvement issues. ACO enlivened by a settlement of ants that scan for a typical wellspring of sustenance. A randomized inquiry heuristic strategy is broadly embraced for taking care of issues from combinatorial advancement. Development of the answers for a given issue is arbitrarily

strolling on a development diagram. The heuristic data of the issue impacts the irregular walk [1].

ACO calculations can incorporate data about the issue into the development of another arrangement which other randomized inquiry heuristics like Simulated Annealing do not perform. Voyaging Salesperson Problem was one of the first combinatorial enhancement issue to which the ACO strategy [1] is connected. The ants have the capacity to discover a most limited way to a sustenance source in specific situations by circuitous correspondence. The pheromone qualities impact the correspondence. The conduct of ants fortified into an algorithmic structure to get answers for a given issue. Arrangements built by arbitrary strolls of manufactured ants on a development diagram, which has weights or the pheromone values on the edges. Bigger pheromone qualities prompt higher likelihood of traversal by another arrangement of ants in the following cycle. In a progression of analyses on a settlement of ants with a decision between two unequal length ways prompting a wellspring of nourishment, scholars have watched that ants tended to utilize the most limited course.

Image Edge Detection is a, on a very basic level vital component in image handling/investigation. Edges portray limits thus have an extensive variety of helpful applications, for example, division and ID of items in scenes, machine vision, cosmology and microscopy imaging to give some examples. Edge recognition is a method for checking sharp intensity changes, and is vital in further breaking down image content. It establishes the framework for image combination, shape extraction, image division, image coordinating, and image following. Numerous conventional edge location methodologies result in broken pieces, which prompt inadequate resultant pictures. The examination done in this paper and additionally by numerous kindred scientists introduce that ant settlement improvement based component has

a tendency to repay those lost edges. The proposed edge detection results in less broken edges making ACO an efficient approach for edge detection.

This paper concentrates on applying to separate edge data from a picture via the ACO calculation such that various ants, which are alluded as the product operators, are engendered on the picture. By including the force estimations of the picture, which is the edge data at every pixel, the pheromone grid is built. The primary inspiration to this paper is the inadequate exploration work finished with tackling the edge location issue with ACO. The contrast between different inquires about and our own is: Our methodology being Ant Colony methodology contrasted with [2], which is an Ant System, furthermore in this paper, the edge discovery is specifically connected with ACO (picture is pre-handled) instead of utilizing routine edge recognition calculations to gather the edge data and improve utilizing ACO as in [3].

2. PROBLEM STATEMENT

The problem identification of this work is defined as:

"*The implementation of efficient Ant Colony Optimization (ACO) algorithm for edge detection in large sized images*".

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

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

Efficient techniques and methods have been studied to provide performance and speed efficient image edge detection. Speed becomes a major concern in such systems when the size of the images increases. This increases the complexity of the system, in addition to the increase in the number of pheromones. Speed optimization for large sized image analysis in terms of image edge detection is a latest research area. Big images demand lots of processing and memory processing to produce good results. The following research questions are being targeted in this research study: Is the Ant Colony algorithm being more efficient for image edge detection in

terms of speed? Does the Ant Colony algorithm provide the best approach to perform optimization for images? Is the problem of image edge detection resolved using the Ant Colony Optimization algorithm (ACO)?

During the use and preliminary experiments of the Ant Colony optimization algorithm for solving the problem of image edge detection, the following problems have been faced: an increase in the size of the image increases the number of pheromones, and thus, it makes it difficult and time consuming to track and handle these pheromones, leading to increased memory requirements, increased data handling concerns and enhanced complexity of computation.

This document has been divided into four major chapters starting form a thorough study of the background knowledge in Chapter 3, a literature review followed by the proposed methodology in Chapter 4, and the implementations in Chapter 5. Chapter 6 includes detailed results, and the discussions with conclusions are given in the final chapter (Chapter 7).

3. BACKGROUND

3.1. Evolutionary Computation

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

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

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

A Simple Evolutionary Algorithm

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

Figure 1. Evolutionary Algorithm

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

3.2. Swarm Intelligence

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

Swarm intelligence works on the inherent capability of these agents, which are not controlled or connected by a centralized control unit. Agents execute the basic rules and fulfill their tasks. Also, the agents accomplish their work based on collective efforts irrespective of what the individual agent performs. The intelligence in the activities of these agents appears due to a random and localized interaction between them. Some major examples of swarm intelligence are ant colonies, animal herding, etc. [4].

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

- **Particle swarm optimization** – is an optimization technique where the solution is given in terms of a point or surface. The solution is represented in an n-dimensional plane. Hypothesis testing is done by plotting a seed and then looking for the desired results. Also, the particles are allowed to communicate using a communication system. The system evaluates the functionality and movement of the particles in specific time intervals against some testing criterion. Particles with better adoptability to the testing criteria are attracted to each other. This technique helps in maximizing the dominant behavior of particles [6].
- **Artificial bee colony algorithm** – is an algorithm, which is based on the behavior of honey bees. It exploits the movements and ways in which honey bees fulfill their tasks. This algorithm works according to three main phases as employed bee, onlooker bee and scout bee [7]. In the first two phases, the bees exploit their surroundings for the solution. They explore the neighborhood. While in the third phase, selection of useful sources of food/solutions is done and the useless solutions are abandoned.
- **Differential evolution** – this algorithm deals with genetics. It consists of search vectors to carry out the searching operation. It exploits the use of search agents, which fulfill the tasks of searching the patterns and genetics.
- **Ant colony optimization** – is an optimization technique based on the behavior of the ant colony. It observes how the ants move and how they coordinate. It consists of an algorithm, which is developed based on the behavior of ant colony and a probabilistic approach is followed. Ants leave pheromones while moving for the tracing of the path. Similarly,

simulated ants are used in the algorithm, which leave traces to identify the path. This algorithm is applied in many fields like image processing, best path discovery, etc. [8]

Swarm intelligence has been explored to find solutions in many different areas and domains [6]. Some of the major areas of application are telecommunication where the networks are spanned by the swarm algorithms for best solutions [9]. Also, it has been used in airline systems, where it is used to determine the best gate for airline arrival. It is also applied to determine behaviors in crown simulation and human swarming.

Considering its widespread use, it is being explored for solutions in domains where more and more solutions are present but the best fit solution is to be found. In addition to design, more algorithms in are optimized for best results. The study in this paper also proposes a modification and optimization of the ant colony algorithm, which is a swarm intelligence based optimization technique. The efficiency of the system is enhanced so that the speed is increased.

Ant colonies demonstrate an organized social association regardless of the effortlessness of their people. On nature of ants to collectively find food sources are inspired and its technique is used for tackling complex problems that are difficult to perform with a solitary subterranean insect. The ant pheromone calculation is a model, motivated by close perception of this present reality ants' conduct, which is an answer for some improvement calculations. The thought behind the ACO algorithm is to utilize a type of simulated stigmergy to organize social orders of counterfeit specialists. ACO enlivened by the searching conduct of ant colonies, and targets discrete streamlining issues. Numerous insect species are totally visually impaired and early research has demonstrated that the ants' conduct is a premise of their correspondence among people and the earth. The ants utilize a compound created by them keeping in mind the end goal to portray this correspondence. The concoction is termed pheromone. The most vital component

in insect species is the pheromone trail. Ants utilize this pheromone as a check for ways, which lead them to their sustenance source and back to their home. Ants tend to store this substance, the pheromone all their way to the nourishment source and on their way back to their home.

Different ants of the settlement sense the most grounded smell of the pheromone trails and take after the way to the nourishment found by different ants. This, aggregate procedure of trail laying and trail taking after conduct whereby an ant impacted by the compound trail left by different ants is the motivation behind ACO.

The analyses demonstrate that ant colonies have the capacity of optimization. The ants utilize probabilistic practices in light of neighborhood data to discover the most limited way between two focuses in their surroundings. For experimentation purposes, researchers have made manufactured ants to mock the behavior of the real world ants. Utilizing a diagram, the briefest way between the two nodes relating to the home and the sustenance source is outlined. The methodology this theory will at last take is much the same as the ACO strategy; it is of note that a basic variant of this issue of discovering a most brief way in the middle of home and sustenance was performed in tests in another proposal at SDSU that depended on the insect based procedure [10].

As, appeared in Figure 2:

1. The first ant finds the food source (F) in an arbitrarily chose highway (an) and afterward comes back to the home (N), deserting a trail pheromone (b).
2. Ants meander in the four unique ways making the more grounded pheromone containing route as the most alluring (short) course.

3. Ants take the briefest route and the pheromone content on alternate trails rots.

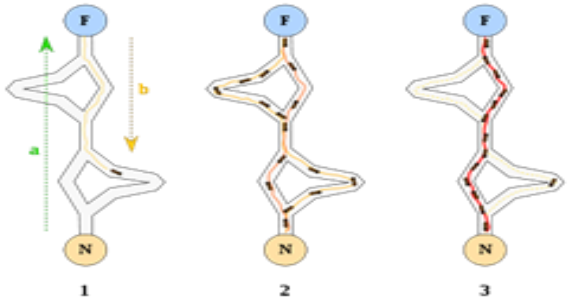


Figure 2. Ant movements from nest to source of Food (F) [1]

`

3.3. Ant Systems

The Ant System (AS) was one of the first to introduce the ACO calculations. It showed the behavior of ant provinces and served as a model for the various variations growing in this way. As of now, three other essential calculations were created - Ant-Cycle, Ant-Density and Ant-Colony.

1. Ant Cycle: Ants are instated, then those ants are set to proceed onward the trails and ants store pheromone strictly when they have manufactured the whole visit. Tentatively it is found that there is a straight connection between the quantity of towns and the best number of ants. The calculation's complexity is O (n^3), where n is the number of cycles.

2. Ant Density: This calculation contrasts from the Ant-Cycle from the way the trail is redesigned. Ants store pheromone on the edge every time it goes through it, and store a consistent amount of Q on the trail every time [3].

3. Ant Quality: This calculation capacity corresponding to the Ant-Density demonstrate at the same time, every time its goes through an edge a pheromone trail of Q/d_{ij} is kept where d_{ij}

[1] http://frakim.blogspot.com/2011/05/ant-colony.html

is the Euclidean separation from node i to j. Accordingly, the shorter ways appear to be preferred in this methodology [11].

The Ant-Cycle has demonstrated to show better results when contrasted with Ant-Density and Ant-Quality in numerous trial results. Ant Cycle utilizes overall data, such that the measure of pheromone kept on the trail is specifically corresponding to the arrangement's nature. Ants creating shorter ways contribute to a higher measure of trail than ants whose visit was shorter. Unexpectedly, both Ant-Quantity and Ant-Density use neighborhood data. There are no conditions between the pheromone amount saved and the arrangement's nature delivered. The ants hunt is not coordinated by any measure of the outcome accomplished [3]. It has been shown by numerous researchers that the Ant-Cycle methodology is more practical contrasted with the other two regarding the calculation time as well as the level of the outcome delivered.

3.4. Introduction to Ant Colony Optimization (ACO) Algorithm

The Ant colony optimization algorithm is based on probabilistic solutions for different computationally expensive problems. It provides an efficient way of path identification using graphs. ACO belongs to the basic ant colony algorithms family. It performs metaheuristic optimizations. This algorithm was put forward by Marco Dorigo since its proposal and acceptance in 1992. It was designed to solve the problem of optimal path discovery in graphs. The new concept in this algorithm was its derivation based on the behavior of ants in finding the best possible path in their colony [1] [2].

The concept of the ant colony algorithm, which was initially targeted for the resolution of optimum path discovery in graphs, later found application in many diverse and new domains related to all fields of life.

Ants are known to wander asymmetrically all around the space, but when they are able to find food, they come back to their colony by laying a trace to food in the form of pheromone trails. This trail turns into traceable paths for any other ants. If any ants find such traces, it follows it to the food. Thus, now the ants get organized and leave wandering around randomly.

Figure 3. Ant leaving traces for path discovery as pheromones[2]

The trail formed by placement of pheromones becomes light and untraceable as the pheromones evaporate. This reduces its strength. Thus, the longer the path is for the ant to move back and forth, the longer should the pheromones evaporate. If the path to be covered by the ant is short, the ant will move more frequently and marking it with a greater frequency. This results in the density of pheromones much greater on shorter paths as compared to longer ones [12]. Pheromones avoid sticking to a localized path already discovered as optimal. In absence of any evaporation, the initial path taken by the first ant would be considered as optimal. Thus, an optimal shortest path found by an ant is followed by other ants and based on good feedback, the same path is adopted by all ants to the food source.

The concepts presented by evaluation of the path discovery behavior of the ants are exploited to be simulated in modern technological problems and their solutions where graphs are involved. Simulated ants are used to mimic the same effect on the graphs as pheromones to find the shortest or optimal path [3]. Figure 4 shows that ants choose smaller drops of honey in comparison to large quantity of sugar, which is not nutritious.

[2] http://antark.net/ant-life/ant-communication/pheromones/

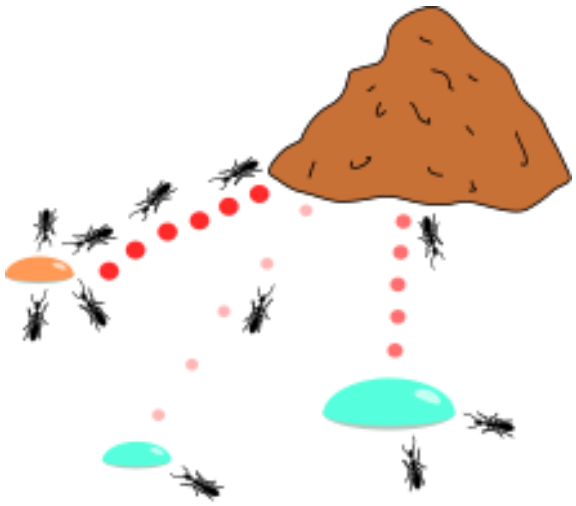


Figure 4. Knapsack problem[3]

Applications of ant colony algorithms have led to many computational demanding fields. The ant colony algorithm has the advantage to adapt at runtime to any changes in the graph. The first ant colony based system is known as ant system. This algorithm was designed to solve the problem of shortest path discovery for the salesman by travelling between cities. The major steps of the algorithm are:

1. Every ant visits every city at least once.
2. The chance of a city at a distant being selected is less.
3. The probability of selection of an edge with strong pheromone between two cities is high.
4. On completion of the journey, the ant leaves more pheromone on the edges it went through during its journey.
5. Pheromone evaporates after each iteration.

3.5. Image Processing

Feature extraction is generally applied to image processing through algorithms, which differentiate or extract specific shapes and patterns in digital images. Some of the image

[3] https://commons.wikimedia.org/wiki/File:Knapsack_ants.svg

processing techniques, which are part of most of feature extraction algorithms, are edge detection, corner detection, blob detection, ridge detection and **Scale-invariant feature transform (**SIFT). Important parameters such as curvature, image motion and shapes must be kept under consideration. The shape-based feature extraction in image processing involves thresholding, blob detection and extraction, template matching and Hough transforms [13]. These are flexible methods to determine different shapes and active contours.

3.5.1. Feature Identification

Feature identification in images deals with the methods and techniques involved to get the information, which is hidden inside the images. This is required to develop a set of image features and the distinct types for in depth analysis. This analysis involves parsing the data of the image pixel by pixel so that none of the characteristics are neglected. All the features recognized and highlighted are collected and then data analysis and observation techniques are applied using unique points or curves. Features are diverse and cannot be defined by their characteristics, as every image has its own features. Additionally, it is difficult to point out what factors make up a specific feature. Each system is defined to have its own features and properties. It is completely application dependent [10].

For feature extraction, different techniques are available in domains of computer vision and machine learning. For efficient image analysis, feature identification and extraction plays a significant role as they define the efficiency of the resulting analysis. In order to explore the images, multiple images are taken and then comparison of the images is made to see for any differences. The detection of features makes the image analysis algorithms more efficient as further analysis is just dependent on these features rather than the whole image, making the processing much more efficient [14].

This process of extracting features form the images, to apply efficient image processing algorithms to extract desired properties of the images, is complex and time consuming. It demands deep understanding of image analysis concepts and techniques [15] [16]. As time is one of the most limiting factors, feature extraction and image analysis needs to be optimized using efficient algorithms.

Feature identification and usage has been done in literature exploiting many new methods and techniques to make the image processing domain efficient irrespective of the size and the complexity of the image data present for analysis.

3.5.2. Types of Features in Images

Features differentiate images in different types. Some of these features are [10]:

- Edges
- Corners
- Blobs
- Ridges

Edges – Edges are Boundaries between image regions and are characterized as edges. They may be of any shape. Points in images with sound gradient values are known to have an edge. The edge is arbitrary in shape and its features; however, algorithms have been defined to explore the shape and other features of edges such as smoothness and dimensionality. Edges are defined as one dimensional features of images.

Corners - Corners are point like features referred to in the images as interest points. They are two dimensional structures. Corners were used in the earliest techniques of image analysis; any abrupt changes in the images were identified as corners or interest points. Edge detection is highly connected to corner identification methods; however, the presence of algorithms for

corner identification has reduced the need for edge detection procedures. Other than the actual corners of the image, other interest points are detection of small parts of the image such as the existence of a white spot on dark or black background.

Blob - Blobs is a feature, which describes the image and its structure based on the type of region, which is under analysis. Blobs are known to identify major regions in the images. Corner detectors may neglect or miss some regions, which are smooth; but these regions are easily detected by blob detectors. Some ambiguity might exist in detecting corners and blobs by corner detector and blob detectors, respectively. By shrinking the image and applying corner detectors, some regions, which were originally smooth in the real image become sharp, which can lead to incorrect results in the detection process. This can be reduced or removed by appropriately selecting the scale of the image.

Ridges - Ridges are evident and obvious in long and stretched images. They are one dimensional structures represented as curves showing the symmetrical axis. Each ridge point is also associated with a local ridge width. Extraction of ridge characteristics is cumbersome in grey images, while detection of the corners, edges and blobs is much easier and more efficient. Ridge detectors are commonly found in the construction industry for identifying roads and in medical devices and systems to recognize the vessels in the images.

3.5.3. Feature Extraction

Feature extraction is an important technique in image processing and analysis. It has laid important foundations in pattern recognition and machine learning. Feature extraction uses an initial crude data set, which it refines to become more useful. The initial data set is manipulated to give more information and to facilitate processing and learning. It also helps in better human

perceptions and conclusions, as it simplifies the data to make an undefined data set more understandable and useful.

The main steps involved in feature extraction involve reducing the data set to an optimal size to make it is easier and more efficient for complex processing. Feature extraction also helps to deal with redundant data and create parameters known as feature vectors, which quantify what the features are. These features contain the information from which the entire original set can be extracted again. This also makes it possible to get the features by processing this reduced data set instead of using a huge data set, thus saving time and complexity [17].

The property of feature extraction to reduce the size of the data set makes it a very attractive technique, as it reduces the amount of resources required to handle a bigger set of data. As the data is reduced, the relationships between the data values are shown by optimized representations. Fewer variables make the process more simple, efficient and cost effective, as it is very complex to deal with a huge number of variables at the same time. General feature extraction algorithms extract common features from the images. This can be made more efficient and refined by making an application-specific feature extraction algorithm.

3.6. Edge Detection

Edge detection is the mathematical modeling of images. The models are used to identify the sharp variations in the images with respect to the brightness of the image. It investigates the abnormal or sudden changes in brightness, which is exploited to form a curved line, generally known as edges. Edge detection is an important processing paradigm for image processing, machine learning, and computer vision, and has an important role in feature detection and extraction [18].

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

Generally, applying edge detectors on an image helps to identify sharp changes as edges. These edges are not usually connected together to form a continuous curve. The edge detection technique reduces a data set to a small size for efficient processing and feature manipulations. A successful edge detection cycle results in efficient information extraction from the data and easier information handling. The simple processing techniques bring excellent results [11].

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

3.6.1. Edge Model

Edges are considered sharp, changed edges. However, this does not hold true in real scenarios, as they show dependency on some real world factors such as point spread function, blurring caused by shadows and shading. Point spread function is the response of the imaging infrastructure to the light from a source.

To measure and analyze variations in the image data, the blurring Gaussian smoothed filter is exploited. This results in a mathematical model, which is represented as,

$$f(x) = \frac{I_r - I_l}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l. \tag{3.1}$$

Abrupt changes and variations in the data set are clearly evident.

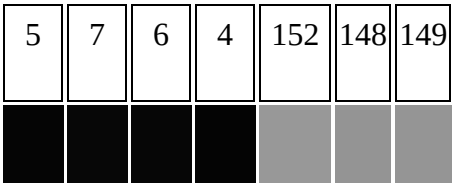


Figure 5. Dataset of variations in image intensities

Identification of edges is easily performed by looking at the data set as shown in Figure 5. The variations in the data set might be small or large. However, it is cumbersome to identify whether an edge exists or not.

For every data set, there has to be a defined threshold to assess whether an edge exists. These thresholds help to identify the edges but do not confirm if a certain value remain an edge identification value. This makes edge detection a complicated procedure.

Edge detection can be done using many different techniques. Edge detection algorithms are divided into two basic underlying branches: search-based and zero crossing-based. Search-based methods initially determine the first order derivative and then find the maxima points. The zero crossing-based edge detection algorithms first find a second order derivative expression and then extract the appropriate values. Prior to the edge detection algorithm, smoothing filters are applied to reduce or remove noise [19].

Many different edge detection algorithms are currently available, which differ in terms of the level of smoothing and the way the strength of the edges are computed and identified. Gradient calculations are also an important step in edge detection.

3.6.2. Canny Edge Detection

The Canny edge detection algorithm [7], developed by John Canny (1986), may incorrectly represent a single edge as an edge or a corner. To avoid this problem, a mathematical model is used to select the appropriate and most efficient smoothing filter. This filter is based on a first order derivative of the Gaussian curves. Canny (1986) also showed that having pre-

smoothing filters ensures that the edge points are local maxima in those regions, meaning they have a maximum value for the gradients.

Figure 6. Canny edge detection

Despite being a very old technique, canny edge detection is still applied in image processing systems. Canny edge detection is most effective. There are many other ways and techniques to determine the edges. One such technique is the use of central differences shown by the mathematical relations below [20].

$$L_x(x,y) = -1/2 \cdot L(x-1,y) + 0 \cdot L(x,y) + 1/2 \cdot L(x+1,y) \quad (3.2)$$

$$(L_y(x,y) = -1/2 \cdot L(x,y-1) + 0 \cdot L(x,y) + 1/2 \cdot L(x,y+1) \quad (3.3)$$

This involves the mask,

$$L_x = \begin{bmatrix} -1/2 & 0 & 1/2 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1/2 \\ 0 \\ -1/2 \end{bmatrix} * L.$$

Filter dependencies are explained as,

$$L_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * L.$$

Thus, the gradient magnitude and phase are given as,

$$|\nabla L| = \sqrt{L_x^2 + L_y^2} \quad (3.4)$$

$$\theta = \operatorname{atan2}(L_y, L_x). \tag{3.5}$$

Filter design can be made optimal to avoid detecting edges in regions of the image, which show low signal-to-noise ratios. This results in images with very low resolution.

3.7. Threshold Application

Once the edges and the values of the thresholds are identified, these thresholds need to be applied. The threshold values determine the quantity of edges being detected. While lower threshold values result in more edges than higher threshold values, they are easily and greatly affected by noise, which increases the chances of the image being distorted. Applying thresholds to gradient images results in thick edges, meaning thinning operations have to be applied to them later. However, if the edge detection is done without maximum suppression, the edge curves are thin and difficult to be recognized.

The problems associated with thresholds can be resolved with hysteresis. Here, instead of signal threshold values, multiple values are selected as thresholds, and the procedure to find the edge starts from the highest threshold value. The start of the edge is identified and traced until the pixel value falls below the threshold value. This method has a baseline assumption that edges are curves, and thus, they have to be traced within the image data. Although this technique shows good results, it is difficult to decide on the best threshold values for proper and efficient edge detection.

Edge thinning, which is performed to remove abrupt changes and variations to the edges, is a post noise filtering step carried out after the edge detection procedure has been executed. This results in sharp and thin edges, and results in greater efficiency for feature extraction and recognition.

Edge thinning can be done using a variety of algorithms. One such technique is explained as follows. First, the connectivity is selected, most preferably 8-connectivity. In 8-connectivity, all the pixels placed immediately next to the pixel being considered are chosen. Points are removed from all directions. This cycle is repeated many times to reach the appropriate and desired point. The higher the required accuracy, the greater the number of cycles executed for this algorithm.

3.7.1. Differential Edge Detection

In addition to first order techniques, several second order techniques are also used. One such technique is differential edge detection in which a local coordinate system is proposed at every point. Pre-smoothing is already applied using the Gaussian smoothing filters. The gradient for scale space representation is given as,

$$\partial_v(L_v) = 0$$

Giving a negative second order derivative,

$$\partial_{vv}(L_v) \leq 0.$$

The equation is given as,

$$L_v^2 L_{vv} = L_x^2\, L_{xx} + 2\, L_x\, L_y\, L_{xy} + L_y^2\, L_{yy} = 0, \tag{3.6}$$

Based on the following condition,

$$L_v^3 L_{vvv} = L_x^3\, L_{xxx} + 3\, L_x^2\, L_y\, L_{xxy} + 3\, L_x\, L_y^2\, L_{xyy} + L_y^3\, L_{yyy} \leq 0 \tag{3.7}$$

Where L_x, L_y... L_{yyy} denote partial derivatives.

This results in the detection of edges, which are already continuous curves, instead of tracing the start of the edges till the end. Second order derivations are given as,

$$L_{xx}(x, y) = L(x - 1, y) - 2L(x, y) + L(x + 1, y). \tag{3.8}$$

$$L_{xy}(x,y) = (L(x-1,y-1) - L(x-1,y+1) - L(x+1,y-1) + L(x+1,y+1))/4, \tag{3.9}$$

$$L_{yy}(x,y) = L(x,y-1) - 2L(x,y) + L(x,y+1). \tag{3.10}$$

The filter mask used is,

$$L_{xx} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} * L \quad \text{and} \quad L_{xy} = \begin{bmatrix} -1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4 \end{bmatrix} * L \quad \text{and} \quad L_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} * L.$$

The same procedure can be used to find higher order derivatives.

4. ACO EDGE DETECTION

Numerous techniques are executed throughout the years for edge detection among which is utilizing the Ant Colony Optimization calculation to distinguish the edge in a picture. This undertaking actualizes the ACO calculation for Image edge discovery utilizing Matlab. A definitive objective of this task is to demonstrate the effectiveness of ACO to detect edges and its optimization. It requires a picture as an input and produces the resultant picture. The calculation alludes to the iterative specific system to discover the limit level of a picture. This iterative method for picking a limit was produced by Ridler and Calvard. It changes over the intensity image to a binary image [21]. The ACO calculation parameters can be changed in accordance with show signs of improvement resultant picture. Then again, once in a while, the calculation's execution lessens relying upon gradually better performing PCs.

There has additionally been some exploration before to apply the Genetic Algorithm alongside the Ant Colony Algorithm to discover the image edge data. The normal rate of the population answers that are near the ideal results is low, and henceforth, the answer might not merge to the absolute ideal answer. This is the situation that the ants' calculation has a low populating dispersal and the answer's development meet rapidly, however, the optimum answer is approached at high speed. The case of these specialists is that their methodology builds the normal velocity to locate the ideal answer by applying the ants' pheromone data to the hybrid capacity [22].

This methodology is excluded in this paper and will be taken as a task for further research. Since the methodology followed in this paper considers the sharp intensity changes made in the picture for distinguishing the edge, the loss of vital edges in the picture is decreased when contrasted with ordinary strategies. The components considered in the edge identification

procedure are Edge Orientation and Edge Structure. Edge Orientation refers to the direction in which the edges are searched for by the algorithm by the variation in intensity levels. There are numerous strategies to identify these progressions, for example, changes in intensity levels between the face and the hair in a picture.

In this area, the execution points of interest and the setup methodology are specified in subtle portions alongside the variations done to the research paper [23]. The fundamental objectives of this paper are (1) to demonstrate that the calculation proposed by [23] produces good results, (2) to enhance the execution time of the calculation.

4.1. Methodology

In this section, the methodology of the implementation is discussed. There are two implementations of the algorithm performed. First of all an implementation is performed, which implements the logic defined as in [23]. Later on, the implementation was enhanced and vectorized, such that the operation is sped up without affecting the results. The initial implementation is referred from the research paper [23], and later improved. The primary undertaking of the improvement procedure was to choose a language for programming. Matlab was chosen over other programming languages in the light of its simplicity and processing capability. It is a straightforward methodology for numeric processing and its capacity to picture the outcomes in a useful and proficient way. Matlab has been demonstrated as an effective methodology for image processing. The edge choice in light of the ACO calculation is demonstrated as an efficient way to extract edges from an image.

Image edge detection can be considered as a problem of identifying the pixels in an image, which relate to edges. A 2-D image can be represented as a two dimensional matrix in which each element is considered as the pixel (Figure 7).

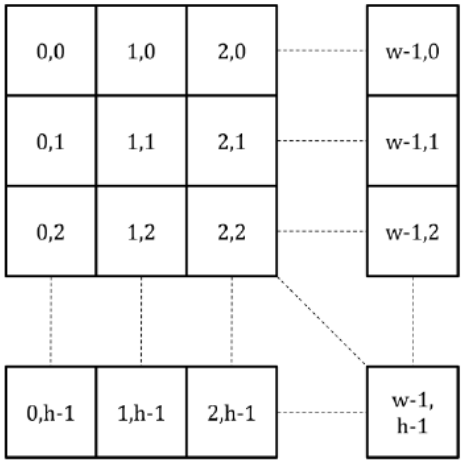


Figure 7. 2-D image represented as matrix

The elements of the graphs in Figure 8 are considered to be the pixels of an image. Only the adjacent pixels are connected in a graph. In Figure 8, the image is represented as a construction graph using an 8-connectivity pixel configuration (Figure 9).

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

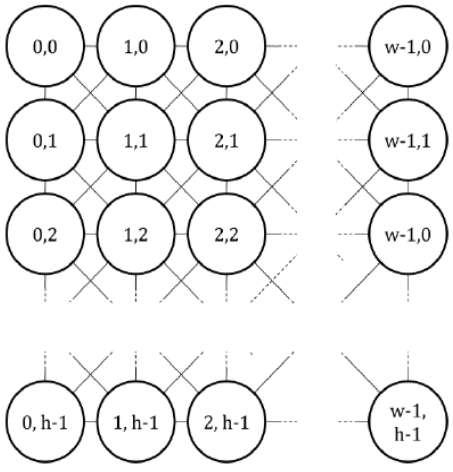


Figure 8. Graph representation of an image

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

Figure 9. 8-connectivity pixel configuration

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

Figure 10 lists the ACO pseudo code.

Function ACO

- Initialize the positions of totally K ants, as well as the pheromone matrix τ^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 $M_1M_2 \; X \; M_1M_2$)
  - Update the pheromone matrix $\tau^{(n)}$
- Make the solution decision according to the final pheromone matrix $\tau^{(N)}$

Figure 10. ACO pseudo code for edge detection [23]

4.2. Algorithm Process

4.2.1. Initialization

K number of ants are introduced on Image I of size M_1 X M_2 where M_1 is the length, M_2 is the picture's width. All the K ants are proliferated on the 2-D picture randomly such that at most one ant is on every pixel. Each pixel in the picture is a hub and the introductory estimation of the pheromone matrix τ (0) is set to a constant value τ_{init}, which is a small but non-zero value. The parameters α and β are introduced where α refers to the weighing factor for the pheromone information on each pixel, and β refers to the weighing factor for the heuristic information.

4.2.2. Construction

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

$$p^{(n)}_{(l,m),(i,j)} = \frac{\left(\tau^{(n-1)}_{i,j}\right)^{\alpha} (\eta_{i,j})^{\beta}}{\sum_{(i,j)\in\Omega_{(l,m)}} \left(\tau^{(n-1)}_{i,j}\right)^{\alpha} (\eta_{i,j})^{\beta}}, \qquad (4.1)$$

Where,

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

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

Ω_i is the neighbor node of ant a_k

α is the constant representing influence of pheromone information

β is the constant representing influence of heuristic information

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

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

$$\eta_{(i,j)} = \frac{1}{Z} V_C(I_{i,j}) \tag{4.2}$$

Where, Z is the normalization factor, which is determined as

$$\sum_{i=1:M1} \sum_{j=1:M2} Vc\ (I_{i,j}) \tag{4.3}$$

Where, Vc ($I_{i,j}$) represents the variation of image's intensity value of the pixel (i, j) of image I. The variation of the image's intensity values depends on c, called clique (as shown in Figure 11), which a group of pixels which are similar in some form. Vc $\left(I_{i,j}\right)$ is the function that is formed by these group of pixels.

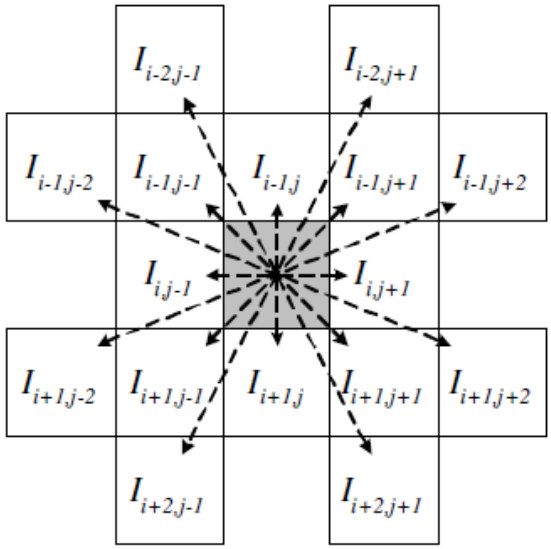


Figure 11. A local configuration at the pixel position $I_{i,j}$ for computing the Variation Vc $\left(I_{i,j}\right)$.

For the pixel $I_{i,j}$ the function Vc $(I_{i,j})$ is determined as:

$$V_c(I_{i,j}) = |I_{i-1,j-1} - I_{i+1,j+1}| + |I_{i-1,j} - I_{i+1,j}| + |I_{i-1,j+1} - I_{i+1,j-1}| + |I_{i,j-1} - I_{i,j+1}| \quad (4.4)$$

This function determines the small angle turns dominate sharp turns in an image. Therefore, it causes a tendency for an ant to move in the forward direction. In order to determine f (.) in Equation (6) for different shapes, the following four functions are considered in the paper.

$$f(x) = \lambda x, \quad \text{for} \quad x \geq 0, \quad (4.5)$$

$$f(x) = \lambda x^2, \quad \text{for} \quad x \geq 0, \quad (4.6)$$

$$f(x) = \begin{cases} \sin\left(\frac{\pi x}{2\lambda}\right) & 0 \leq x \leq \lambda; \\ 0 & \text{else.} \end{cases} \quad (4.7)$$

$$f(x) = \begin{cases} \frac{\pi x \sin(\frac{\pi x}{\lambda})}{\lambda} & 0 \leq x \leq \lambda; \\ 0 & \text{else.} \end{cases} \quad (4.8)$$

Where the parameter λ takes different values to produce different shapes. Figure 12 represents the various outcomes of shapes for different values of λ chosen at random.

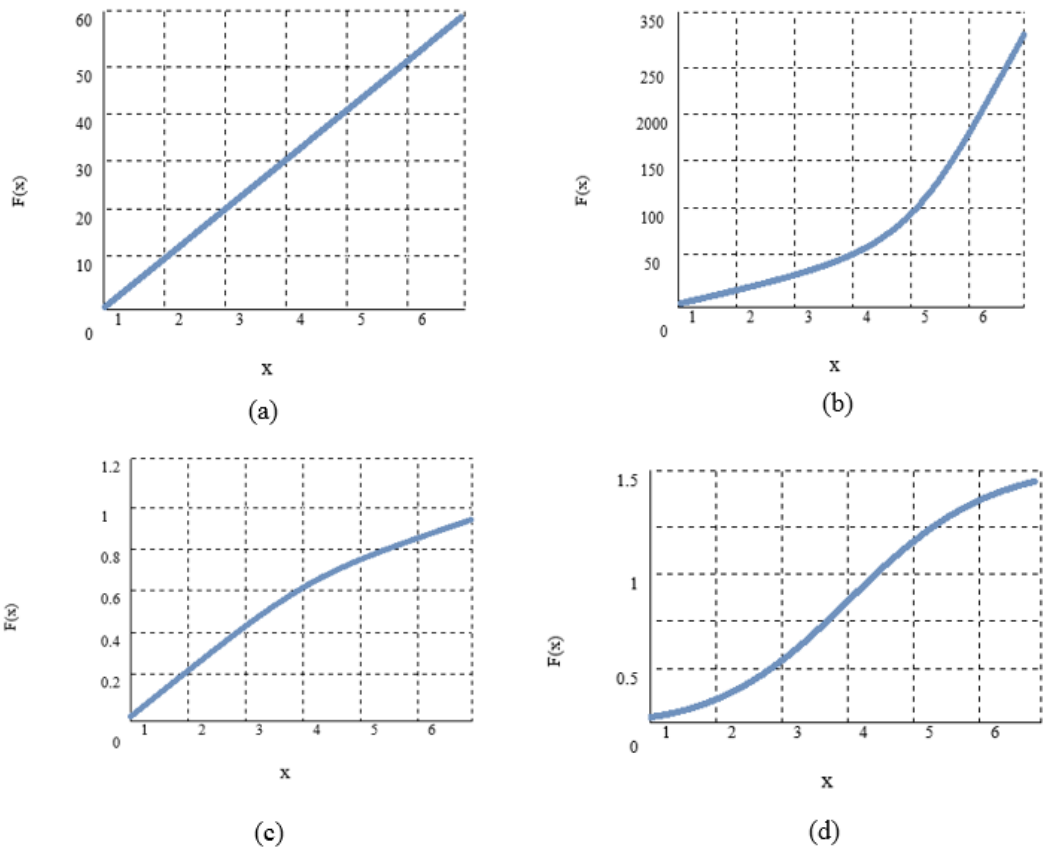


Figure 12. Graphs for various functions with λ=10. (a) Function defined in (4.5). (b) Function defined in (4.6). (c) Function defined in (4.7). (d) Function defined in (4.8)

4.2.3. Update Process

In the paper, two update operations are performed for updating the pheromone matrix. The pheromone matrix is updated after each movement of an ant. After the k^{th} ant in the n^{th} construction step moves, the updated pheromone matrix is

$$T_{i,j}^{(n-1)} = \begin{cases} (1-\rho).T_{i,j}^{(n-1)} + \rho.\Delta\,\tau_{i,j}^{(k)} \ , & if\ (i,j)\ is \\ & visited\ by\ the\ Kth\ ant; \\ T_{i,j}^{(n-1)} \ , & otherwise. \end{cases} \tag{4.9}$$

Where ρ is the evaporation rate

$$\Delta\,\tau_{i,j}^{(k)} = \eta_{(i,j)}$$

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

$$T^n = (1 - \psi).T^{n-1} + \psi.T^n \tag{4.10}$$

Where,

Ψ is the pheromone decay coefficient

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

4.2.4. Decision Process

A binary decision is made on the pheromone matrix to determine if each of its elements (pixels) are considered an edge or not. This decision is based on the application of a threshold on the final pheromone matrix T_n. T is considered to be the Threshold value which needs to be computed as defined in [14].

The initial threshold $T^{(0)}$) is set by computing the mean value of the pheromone matrix. The elements of the pheromone matrix are divided into two segments based on the criteria that its values is lesser than $T^{(0)}$ or greater than $T^{(0)}$. The mean value of each segment is calculated and then the average of these two means is calculated and it is assigned as the new Threshold value. This process is continued until the threshold values stop changing. Below process explains elaborately the steps of the decision process.

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

$$T^{(0)} = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} \tau_{i,j}^{(N)}}{M_1 M_2} \tag{4.11}$$

And set the iteration index l as zero.

2. The pheromone matrix is divided into two segments using T (l), where the first segment consists of entries of the matrix elements, which have lesser value than T (l) and the second segment consists of the remaining entries of the pheromone matrix. The mean value of each segments are calculated as below

$$m_L^{(l)} = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} g_{T^{(l)}}^L(\tau_{i,j}^{(N)})}{\sum_{i=1:M_1} \sum_{j=1:M_2} h_{T^{(l)}}^L(\tau_{i,j}^{(N)})}, \tag{4.12}$$

$$m_U^{(l)} = \frac{\sum_{i=1:M_1} \sum_{j=1:M_2} g_{T^{(l)}}^U(\tau_{i,j}^{(N)})}{\sum_{i=1:M_1} \sum_{j=1:M_2} h_{T^{(l)}}^U(\tau_{i,j}^{(N)})}, \tag{4.13}$$

Where,

$$g_{T^{(l)}}^L(x) = \begin{cases} x, & \text{if } x \le T^{(l)}; \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

$$h_{T^{(l)}}^L(x) = \begin{cases} 1, & \text{if } x \le T^{(l)}; \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

$$g_{T^{(l)}}^U(x) = \begin{cases} x, & \text{if } x \ge T^{(l)}; \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

$$h_{T^{(l)}}^U(x) = \begin{cases} 1, & \text{if } x \ge T^{(l)}; \\ 0 & \text{otherwise.} \end{cases}$$

1. Iterative index is set as l =l+1 and the threshold is computed as the average value of the mean values of each segment calculated above which is as follows:

$$T^{(l)} = \frac{m_L^{(l)} + m_U^{(l)}}{2}. \tag{4.18}$$

2. Check if $\left|T^{(l)} - T^{(n-1)}\right| > \in$ where ϵ is a user defined tolerance value. If it satisfies the condition go back to Step 2. This iteration runs till the condition fails. On failure of the condition, the iteration terminates and a binary decision is made on the pixel (i,j) if it is an edge or not based on the following equation,

$$E_{i,j} = \begin{cases} 1, & \text{if } \tau_{i,j}^{(N)} \geq T^{(l)}; \\ 0 & \text{otherwise.} \end{cases} \quad (4.19)$$

Where,

$E_{i,j} = 1$ is the pixel (i, j) which is considered as an edge.

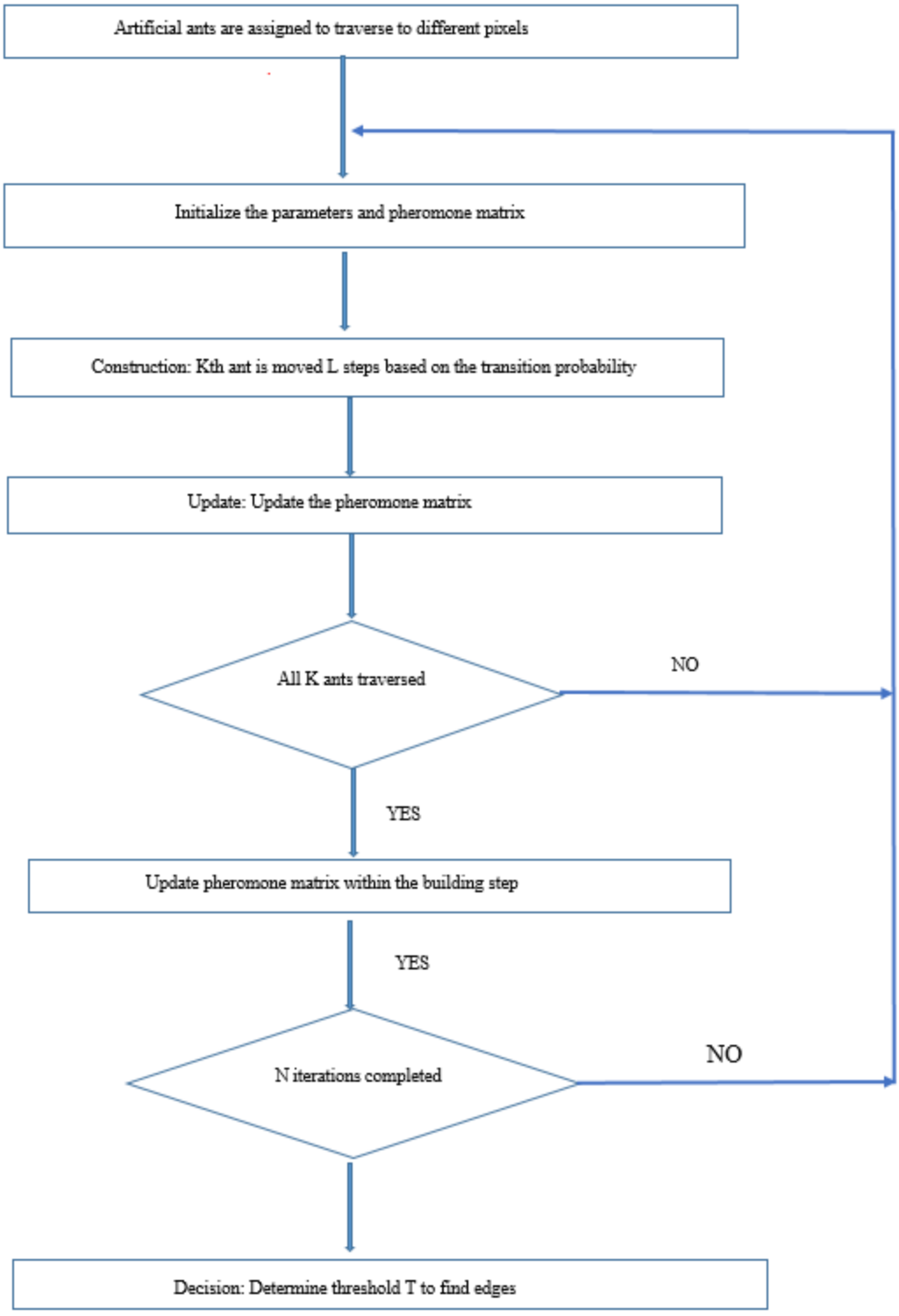


Figure 13. Flowchart of ACO for image edge detection process

5. IMPLEMENTATION

The ACO for edge detection algorithm has been implemented in MATLAB. The initial implementation is done as proposed by [23]. This implementation is enhanced to achieve a speed efficient implementation. The code snippets in this chapter are a part of Implementation 2, which explain in detail the improvements made.

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

```
    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 14. Implementation of probability transition matrix

The optimization is performed with the help of vectorization of the code and which results in faster processing of the algorithm. The modifications in the instructions increase the speed of the operation while keeping the result integrity intact.

The multiplication of the v and p vectors, which are defined in the transition probability, are parallelized and vectorized and hence the normalization operation of the ant transition probability is as shown in Figure 15.

```
prob_v_temp = ant_transit_prob_v. ^alpha;

prob_p_temp = ant_transit_prob_p. ^beta;

prob_vp_temp = prob_v_temp .* prob_p_temp;

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

Figure 15. Normalization process of transition probability

Furthermore, in the implementation of the algorithm proposed in the paper, the rows and columns of the image window are selected at the run time, which takes a lot of time. But this operation is modified with the definition of the masks. Masks are matrices that contain either 0's or 1's, which help in deciding the suitable computation to perform to determine the values in a matrix. As can be easily analyzed that the selected elements can be divided into two groups, i.e., the upper side and lower side. Hence, these elements are defined by two masks; mask1 defines the upper side pixels/values and the mask2 defines the lower side.

```
mask1 = [0 1 0 1 0; 1 1 1 1 1; 0 1 0 0 0; 0 0 0 0 0; 0 0 0 0 0];
mask2 = rot90 (mask1, 2);
```

Now these two masks are used to select the corresponding pixel values. In the implementation proposed in the paper, the pixels are accessed one by one to calculate the variation $Vc(I_{i,j})$, which makes the process slow, but here a complete image window is accessed at once, which makes the operation faster as compared to the proposed implementation. When

we are looking for a 4- or 8-connected neighborhood, here instead of selecting a single pixel at a time, we accessed the whole window. This operation helped to increase the execution time.

```
if search_clique_mode == '4'
            ant_search_range_temp = [index(1)-1 index(2); index(1) index(2)+1; index(1)+1
index(2); index(1) index(2)-1];
        else if search_clique_mode == '8'
            ant_search_range_temp = [index(1)-1 index(2)-1; index(1)-1 index(2); index(1)-1
index(2)+1; index(1) index(2)-1; index(1) index(2)+1; index(1)+1 index(2)-1; index(1)+1
index(2); index(1)+1 index(2)+1];
        end
```

Figure 16. Implementation for neighbor search for 4-connecity pixel and 8-connectivity pixel

The 4 kernel functions defined in Equations (4.5)-(4.8) have been parallelized, this enables us to take the advantage of the parallel processing and multicore architecture of the CPU or processing units.

```
parfor nMethod=1:4
```

The final update of the pheromone matrix is done within each construction step after all ants have traversed as shown in Figure 17.

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

Figure 17. Implementation step for updating final pheromone matrix

The last step is to identify the edges from the final pheromone matrix, this is decided based on calculating the final threshold value T [14].

This decision process has already been discussed in the decision process of previous chapter. The detailed implementation is shown in the Figure 18. The histogram is divided into two segments, the initial threshold value being the mean value of the pheromone matrix. Matlab's built in function "hist" is used to pass the length of the color map and the image.

Since, the image passed is an intensity image, the color map value is set to 256. For a binary image the color map value will always be 2. The threshold value is computed by segmenting the image values of I into two sections, and computing the mean value of these two sections. At end of the loop, the color map value reaches 2, thus, producing the corresponding binary image.

```
% STEP 1: Compute mean intensity of image from histogram, set T=mean (I)
[counts, N]=hist(I,256);
i=1;
mu=cumsum(counts);
T(i)=(sum(N.*counts))/mu(end);
% STEP 2: compute Mean above T (MAT) and Mean below T (MBT) using T
from step 1
mu2=cumsum(counts(N<=T(i)));
MBT=sum(N(N<=T(i)).*counts(N<=T(i)))/mu2(end);
mu3=cumsum(counts(N>T(i)));
MAT=sum(N(N>T(i)).*counts(N>T(i)))/mu3(end);
i=i+1;
T(i)=(MAT+MBT)/2;
% STEP 3 to n: repeat step 2 if T(i)~=T(i-1)
Threshold=T(i);
while abs(T(i)-T(i-1))>=1
 mu2=cumsum(counts(N<=T(i)));
 MBT=sum(N(N<=T(i)).*counts(N<=T(i)))/mu2(end)
 mu3=cumsum(counts(N>T(i)));
 MAT=sum(N(N>T(i)).*counts(N>T(i)))/mu3(end);
 i=i+1;
 T(i)=(MAT+MBT)/2;
 Threshold=T(i);
end
```

Figure 18. Implementation of computing threshold T [14]

6. RESULTS AND DISCUSSIONS

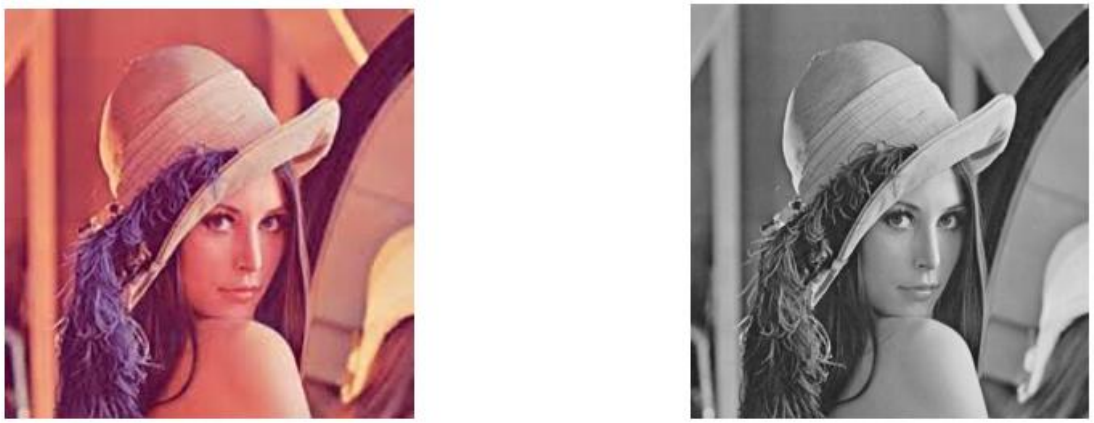
The proposed algorithm is implemented and executed in MATLAB. The resultant output, is a binary image in which the black pigmented areas determine the edges. Experiments were conducted to implement the algorithm proposed by [23] by using MATLAB and produce results similar to what is presented in his paper along with modifying the computations to enhance the speed of the computation process using test images. Initially the implementation was run with arbitrarily set parameters, which resulted in some lost information. After doing much research on the initialization of parameters in [5] [6] [7], the initialization values for the parameters is set as given in Figure 19.

Figure 19 gives the details of the parameters initialized and the other parameters used in the equations during the Construction, Update and Building Process.

Parameters	Values Initialized	Description
K	$\lfloor\sqrt{M1XM2}\rfloor$	Total number of ants is equivalent to the image size
τ_{init}	0.001	Initial value of each element of pheromone matrix
α	1	Weighing factor of the pheromone value in Equation
β	0.1	Weighing factor of the heuristic information in (4)
Ω	8-connectivity	The permissible ants movement range in (4)
λ	1	Adjusting factor of the functions in (7)-(10)
ρ	0.1	Evaporation rate in (11)
L	40	Total number of ant movement steps within each construction step
ψ	0.05	Pheromone decay coefficient in (12)
ϵ	0.1	User defined tolerance value used in the decision process
N	4	Total number of construction steps

Figure 19. Description of Parameters used for implementation

An intensity image is considered to be the input image; the script gives a choice to enter either a color or greyscale image. In this experiment, both 20(a) and 20(b) are considered. The expected output is a monochromatic image. The initial run is done by the arbitrarily set parameters and later based on the best results, further experiments are carried out with varying α and β parameters, which are the weighing factor of the pheromone value and weighing factor of heuristic information, respectively.



(a) (b)

Figure 20. Input images used for test runs. (a) Lena color image. (b) Lena grey image [4]

ACO for Edge detection algorithm incorporates the functions defined in Equations (4.5)-(4.8) for generating the binary images. Observations are made on the output images generated by these four functions for further test runs where the pheromone weighing factors will be varied for achieving the best possible results. The computation time of the algorithm is directly proportional to the size of the image. There is a significant increase in the computation for images larger than 256x256. In our approach, the computation time was reduced in one way by resizing the image to 128x128. The other was by vectorising few of the operations in the algorithm.

[4] Reprinted from Ant Colony Optimization, by Marco Dorigo, Thomas Stützle, 2004, London, England: MIT Press. Copyright 2004 by Massachusetts Institute of Technology.

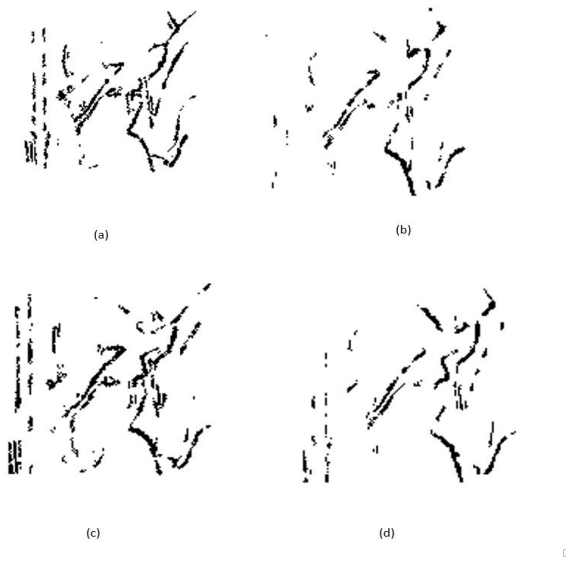


Figure 21. Various extracted edge information of Grey image Lena. (a) image output using equation (4.5). (b) Image output using equation (4.6). (c) Image output using equation (4.7). (d) Image output using equation (4.8)

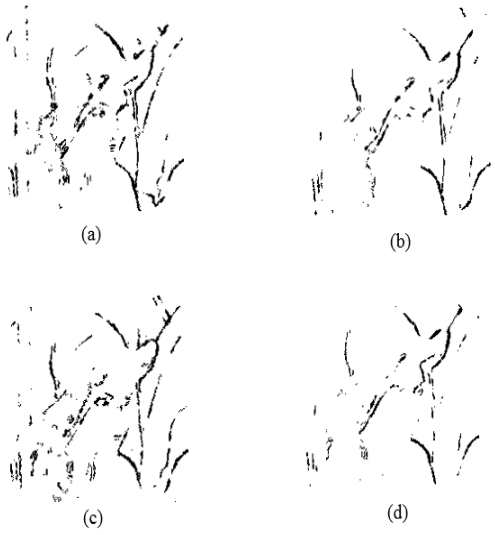


Figure 22. Various extracted edge information of Color image Lena. (a) Image output using equation (4.5). (b) Image output using equation (4.6). (c) Image output using equation (4.7). (d) Image output using equation (4.8)

The algorithm implementation was enhanced with the help of operation transformation using the vectorization of the operation, which enables to increase the speed of the operation. In the meantime, we had to ensure the correct operation of the algorithm, without the significant

loss in the quality of the output. Figure 22 shows the outputs generated by the enhanced implementation, corresponding to the four kernels defined. It can be noticed that there is no degradation in the results as compared to the previous implementation.

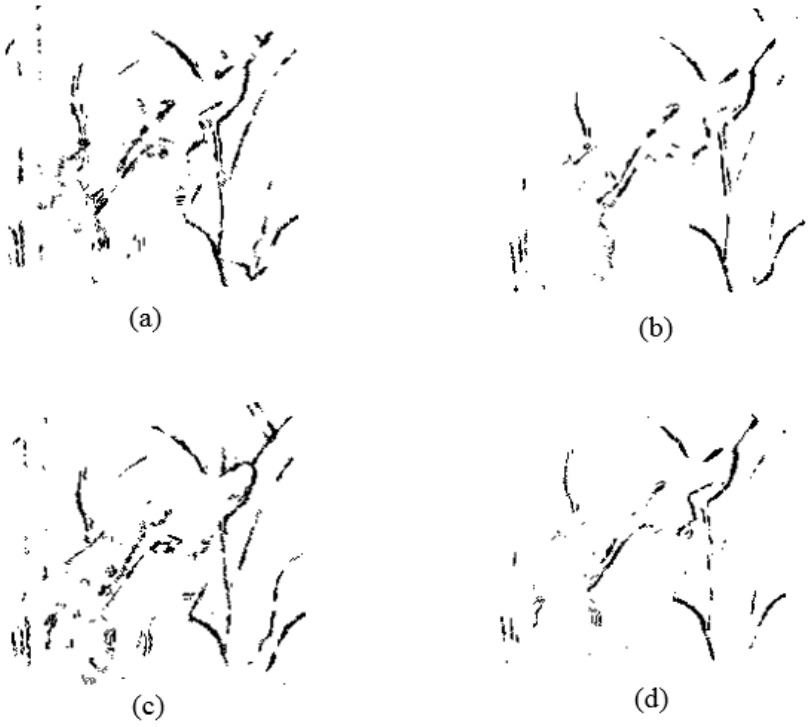


Figure 23. Various extracted edge information of Color image Lena using the speed optimized algorithm. (a) Image output using equation (4.5). (b) Image output using equation (4.6). (c) Image output using equation (4.7). (d) Image output using equation (4.8)

From Figures 21 and 22 we can observe that the color images give better results than the black and white images, this is because the edge information is directly proportional to the image intensity and quality of the image. Also, the best results are generated using the kernel function (4.5) and (4.7) in both cases. Images extracted using Equations (4.6) and (4.8) are considered as bad results since output show a significant loss of edge information. Also, it can happen that the edge information was not identified as an edge.

In order to achieve better results the parameters α and β can be modified to generate different results. α and β are the weighing factors for the pheromone content and heuristic

information, respectively. The values of α and β are considered after running several tests based on trial and error for Equation (4.5)-(4.8). The image outputs along with the values considered can be seen in Figure 24. We can see that the edges are clearer and the extracted edge information is much better than the previous results. This is because we increased the values of α and β, which influence the pheromone and heuristic information during the transition probability, by increasing these values we are increasing the chances of an ant to visit every edge, which prevents loss of edge information.

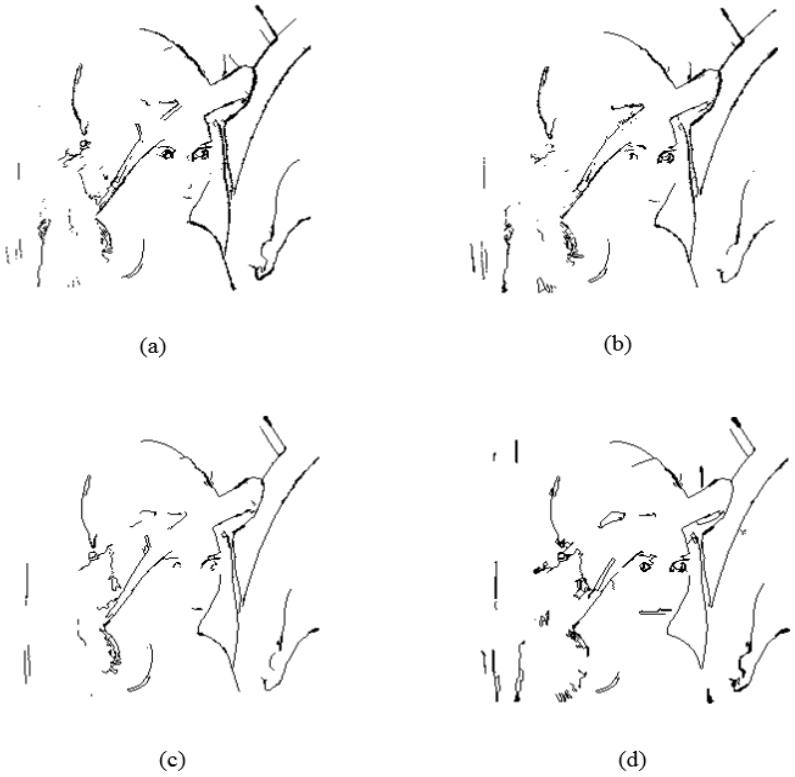


Figure 24. Various extracted edge information varying α and β values; α =20, β=0.4 for (a) image extracted using (4.5). (b) Image extracted using (4.6). (c) Image extracted using (4.7). (d) Image extracted using (4.8)

calculation in Matlab. As a result, the overall speed of the algorithm had increased nearly 2 to 2.5 times as of the original implementation. Figure 25 shows the speed comparison for the different

image resolution. It can be seen that when we increase the image resolution the timing requirements can increase manifold, and the difference in time taken by Jing Tian's implementation [23], and the proposed implementation increases widely.

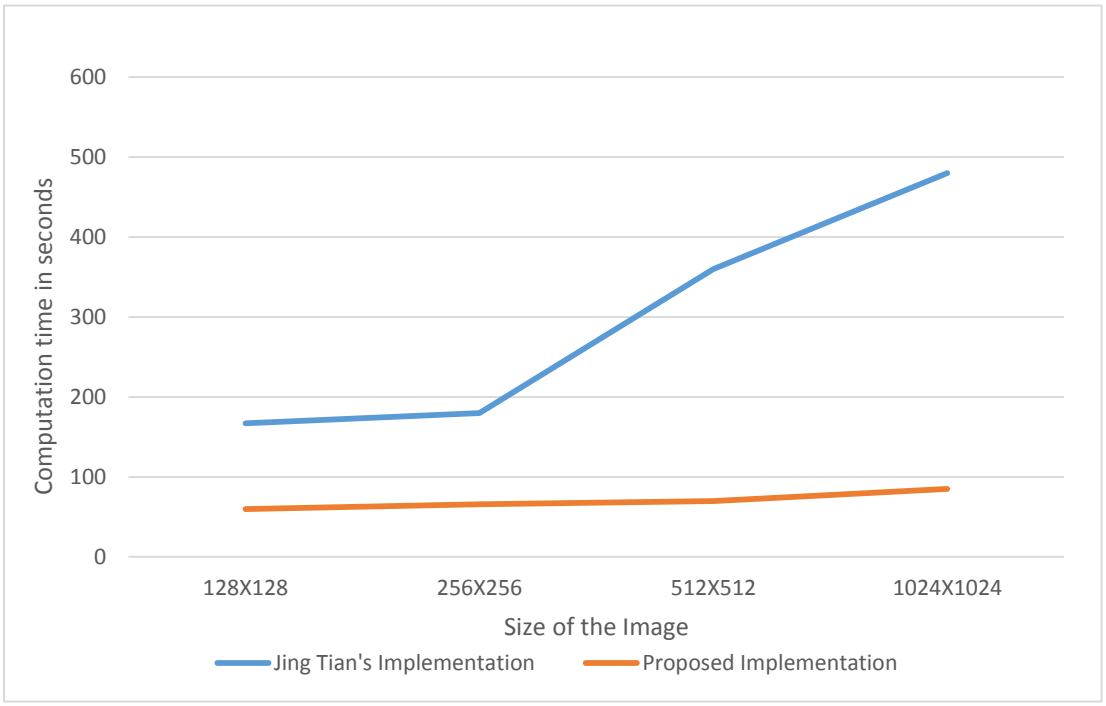


Figure 25. Time consumption comparison between the Jing Tian's implementation and the proposed implementation

7. CONCLUSION AND FUTURE WORK

This research work involved the implementation of the Ant Colony Optimization algorithm. The ACO algorithm was used for the edge detection for color and greyscale images, and produces improved results as compared to the already existing techniques. Two implementations were evaluated for the same ACO algorithm. This research work had also explored the potential of the Matlab software for the processing of vectors and matrices. The vectorization of the operation enabled us to speed up the processing of the algorithm without reducing the efficiency of the algorithm and degradation of the results. Furthermore, four different kernel functions were applied to the processing of the algorithm. The generated output images show the differences of edges obtained for each kernel function. From the experiments it was observed that the kernel function (4.5) and (4.7) generate the best possible results.

Color Shade variations are also important for detecting edges as it becomes difficult to detect an edge if there is lack of contrast in the color at the edges of the image. Also it is very evident that the results obtained are better where there is a significant increase in the pheromone and heuristic values, which promise a better solution for edge detection.

The methodology applied to the image edge detection problem demonstrated its applicability to this application domain. The ACO calculation is utilized to take care of the image edge discovery issue. The intensity of the image is intact while converting it to a binary image. Optimization is performed to increase the speed of computation while keeping the integrity of the ACO algorithm for edge detection intact.

This research work dealt with the implementation of the ACO algorithm and performed the enhancement in the implementation in terms of speed, and was able to increase the speed by 2.5 to 3 times. Still there is a lot of room for enhancement and research in this area. Future

research in this area could target the implementation of the algorithm in the real-time, which will enable this algorithm to be usefully applied in several high-end applications like real time feature extractions, 2D to 3D conversion, etc.

There is another research area, which targets the application to be run on GPU, and thus the overall speed of the process could be further enhanced. Researchers could explore the GPU implementations of the ACO algorithm to produce real time features. Also, researchers could aim for porting this algorithm on the hardware platforms like Field Programmable Gate Arrays (FPGA).

8. BIBLIOGRAPHY

[1] D. Martens, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with Ant Colony Optimization," IEEE Trans. on Evolutionary Computation, Oct. 2007, vol. 11, pp. 651–665.

[2] R. S. Parpinelli, H. S. Lopes, and A. A. Freitas, "Data Mining with An Ant Colony Optimization Algorithm," IEEE Trans. on Evolutionary Computation, Aug. 2002, vol. 6, pp. 321–332.

[3] S. Ouadfel and M. Batouche, "Ant Colony System with Local Search for Markov Random Field Image Segmentation," in Proc. IEEE Int. Conf. on Image Processing, Barcelona, Spain, Sep. 2003, pp. 133–136.

[4] A. P. Engelbrecht, "Fundamentals of Computational Swarm Intelligence", John Wiley & Sons, 2006.

[5] G. Beni and J. Wang, "Swarm Intelligence in Cellular Robotic Systems", Robots and Biological Systems: Towards a New Bionics, Springer Berlin Heidelberg, NATO ASI Series, 1993, Vol. 102, pp 703-712.

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz, "From Natural to Artificial Swarm Intelligence", 1999, Oxford University Press.

[7] D. Karaboga and B. Akay, "A Survey: Algorithms Simulating Bee Swarm Intelligence", Artificial Intelligence Review, 2009, Vol. 31 Issue Pp.61 – 85.

[8] M. Dorigo, "Ant Colony Optimization and Swarm Intelligence", Proceedings of 6th International Conference, 2008, ANTS 2008, Brussels, Belgium.

[9] F. Ducatelle, G. A. Di Caro, And L. M. Gambardella, "Principles And Applications Of Swarm Intelligence For Adaptive Routing In Telecommunications Networks", Swarm Intelligence, Springer US, 2010, 4(3), pp 173-198.

[10] R. C. Gonzalez and R. E. Woods, "Digital Image Processing". Harlow: Prentice Hall, 2007.

[11] D.S. Lu and C.C. Chen, "Edge Detection Improvement by Ant Colony Optimization," Pattern Recognition Letters, Mar 2008, vol. 29, pp. 416–425.

[12] A. R. Malisia and H. R. Tizhoosh, "Image Thresholding Using Ant Colony Optimization," in Proc. Canadian Conf. on Computer and Robot Vision, Quebec, Canada, Jun. 2006, pp. 26–26.

[13] J. Canny, "A Computational Approach to Edge Detection", IEEE Trans. on Pattern Analysis and Machine Intelligence 8(6), 1986.

[14] N. Otsu, “A Threshold Selection Method from Gray Level Histograms,” IEEE Trans. Syst., Man, Cybern, Jan 1979, vol. 9, pp. 62–66.

[15] J. Tang, R. Rangayyan, Y. Yang and J. Yao, "Special Issue on Digital Picture Processing", Proc. IEEE, 1972, vol. 60, pp.763 -922.

[16] H. C. Andrews , A. G. Tescher and R. P. Kruger "Image Processing By Digital Computer", IEEE Spectrum, 1972, vol. 9, pp.20 -32.

[17] J. Bernd, “Practical Handbook on Image Processing for Scientific Applications”, CRC Press, New York, 1997.

[18] H. Nezamabadi-Pour, S. Saryazdi, and E. Rashedi, “Edge Detection Using Ant Algorithms,” Soft Computing, May 2006, vol. 10, pp. 623–628.

[19] M. Randall and A. Lewis, “A Parallel Implementation Of Ant Colony Optimization,” Journal of Parallel and Distributed Computing, Sep 2002, vol. 62, pp. 1421–1432.

[20] P. Zhou, W. Ye, Y. Xia and Q. Wang, “An Improved Canny Algorithm for Edge Detection, Journal of Computational Information Systems”, 7:5 (2011) 1516-1523.

[21] S. Calvard and T. W. Ridler, “Picture Thresholding Using an Iterative Selection Method. IEEE Trans. System, Man and Cybernetics”, 1978, 8:630-632.

[22] J. Rahebi, Z. Elmi, A. Farzamnia, 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.

[23] J. Tian, W. Yu, and S. Xie, “An Ant Colony Optimization Algorithm for Image Edge Detection. IEEE Congress on Evolutionary Computation”, 2008, 751-756.