

Summer 7-29-2016

# Color Image Segmentation Using the Bee Algorithm in the Markovian Framework

Vehbi Dragaj  
*Kennesaw State University*

Follow this and additional works at: [http://digitalcommons.kennesaw.edu/cs\\_etd](http://digitalcommons.kennesaw.edu/cs_etd)



Part of the [Other Computer Sciences Commons](#)

---

## Recommended Citation

Dragaj, Vehbi, "Color Image Segmentation Using the Bee Algorithm in the Markovian Framework" (2016). *Master of Science in Computer Science Theses*. Paper 5.

This Thesis is brought to you for free and open access by the Department of Computer Science at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in Master of Science in Computer Science Theses by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact [digitalcommons@kennesaw.edu](mailto:digitalcommons@kennesaw.edu).

# COLOR IMAGE SEGMENTATION USING THE BEE ALGORITHM IN THE MARKOVIAN FRAMEWORK

A Thesis Presented to  
The Faculty of the Computer Science Department

by

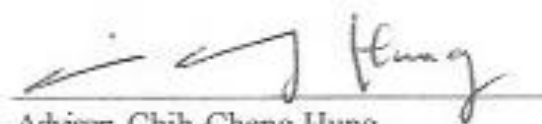
Vehbi Dragaj

In Partial Fulfillment  
of Requirements for the Degree  
Masters of Science in Computer Science

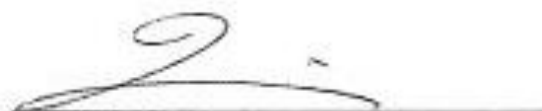
Kennesaw State University  
July 2016

COLOR IMAGE SEGMENTATION USING THE BEE ALGORITHM IN  
THE MARKOVIAN FRAMEWORK

Approved:



Advisor: Chih-Cheng Hung



Department Chairperson: Frank Tsui



Dean: Mike Dishman

In presenting this thesis as a partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from or publication of, this thesis which involves potential financial gain will not be allowed without written permission.

A handwritten signature in dark ink, appearing to read 'Vehbi Dragaj', is written over a horizontal line. The signature is fluid and cursive, with the first letter 'V' being particularly large and stylized.

Vehbi Dragaj

## STATEMENT TO BORROWERS

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Vehbi Dragaj  
vehbi.dragaj@gmail.com

The director of this thesis is:

Dr. Chih-Cheng Hung  
chung1@kennesaw.edu

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Name of user	Address	Date	Type of use (examination only or copying)
--------------	---------	------	---

# COLOR IMAGE SEGMENTATION USING THE BEE ALGORITHM IN THE MARKOVIAN FRAMEWORK

An Abstract of  
A Thesis Presented to  
The Faculty of the Computer Science Department

by

Vehbi Dragaj  
Bachelor of Science in Computer Science, The Georgia Institute of Technology, 2006

In Partial Fulfillment  
of Requirements for the Degree  
Masters of Science in Computer Science

Kennesaw State University  
July 2016

Abstract

This thesis presents color image segmentation as a vital step of image analysis in computer vision. A survey of the Markov Random Field (MRF) with four different implementation methods for its parameter estimation is provided. In addition, a survey of swarm intelligence and a number of swarm based algorithms are presented. The MRF model is used for color image segmentation in the framework. This thesis introduces a new image segmentation implementation that uses the bee algorithm as an optimization tool in the Markovian framework. The experiments show that the new proposed method performs faster than the existing implementation methods with about the same segmentation accuracy.

# COLOR IMAGE SEGMENTATION USING THE BEE ALGORITHM IN THE MARKOVIAN FRAMEWORK

A Thesis Presented to  
The Faculty of the Computer Science Department

by

Vehbi Dragaj

In Partial Fulfillment  
of Requirements for the Degree  
Masters of Science in Computer Science

Advisor: Dr. Chih-Cheng Hung  
Kennesaw State University  
July 2016



Dedication

To my parents.

### Acknowledgements

I would like to express my gratitude to my advisor Dr. Chin-Cheng Hung for encouraging me to do my thesis and for giving me the idea of what to do my thesis on. I would also like to thank him for his advice and support throughout my thesis process.

In addition, I would like to thank Dr. Jeffrey Chastine and Dr. Dan Lo for being in my committee and for their questions and comments during my thesis defense.

## TABLE OF CONTENTS

I.	Introduction.....	1
II.	Literature Survey.....	5
	2.1 Markov Random Fields.....	5
	2.1.1 What Is A Markov Random Field?.....	5
	2.1.2 Neighbors And Neighborhood.....	8
	2.1.3 Markov Properties.....	10
	2.1.4 Clique.....	10
	2.1.5 Clique Potential.....	12
	2.1.6 Energy Function.....	12
	2.1.7 Equivalence Between Markov Random Fields And Gibbs Random Filed.....	12
	2.2 Markov Models.....	13
	2.2.1 Markov Chains.....	13
	2.2.2 Ising Model.....	15
	2.3 Swarm Intelligence.....	16
	2.3.1 Particle Swarm Optimization (PSO) Algorithm.....	19
	2.3.2 Ant colony Optimization.....	21
	2.3.3 Honey Bee Behavior.....	23
	2.3.4 Artificial Bee Colony.....	25
	2.3.5 Improved Bee Colony Algorithm For Multi-Objective Optimization (IBMO).....	26
	2.3.6 Bee Swarm Optimization Algorithm.....	28
	2.3.7 BeeHive Algorithm.....	29

2.3.8	The Bee Algorithm.....	31
2.4	Other Algorithm.....	34
2.4.1	Hill-Climbing Search.....	35
2.4.2	Simulated Annealing.....	36
2.4.3	Metropolis.....	37
2.4.4	ICM.....	39
2.4.5	Gibbs Sampler.....	39
2.4.6	MMD.....	40
2.5	Applications Of MRF And Bee Behavior Algorithms	
	In Image Segmentation.....	41
2.5.1	Color Image Segmentation And Parameter Estimation	
	In A Markovian Framework.....	41
2.5.1.1	The Model.....	41
2.5.1.2	Parameter Estimation.....	42
2.5.1.3	Obtaining initial parameters.....	44
2.5.1.4	Color Image Segmentation Based On Multiobjective	
	Artificial Bee Colony Optimization.....	45
III.	Proposed Solution.....	48
3.1	The Model.....	48
3.2	Parameter Estimation and Segmentation.....	49
3.3	Parameter Initiation.....	49
3.4	Bee Algorithm Parameter.....	50
IV.	Experiments And Results.....	51
4.1	Experiments.....	51
4.2	Time Comparison.....	62

4.3	Accuracy Comparison.....	63
V.	Conclusion.....	68
VI.	Appendix A. Probability And Statistical Concepts.....	69
VII.	BIBLIOGRAPHY.....	78

## LIST OF FIGURES

Figure	Caption	Page
1	Computer vision applications.....	2
2	Six by six empty field .....	6
3	First throw sixth order MRFs.....	9
4	MRF neighborhood where cells are not directly connected.....	9
5	MRF Properties.....	10
6a	First order MRF and its cliques.....	11
6b	Second order MRF and its cliques.....	11
6c	Third order MRF and its cliques.....	11
7	Unconnected MRF neighborhoods with their corresponding cliques.....	14
8	First order of Marko Chain for weather prediction.....	14
9	Weather prediction 2X2 matrix.....	15
10	Ising model graph.....	15
11	Natural Swarm Systems.....	17
12	Artificial Swarm System.....	18
13	Ant path stimulation.....	21
14	Waggle dance.....	24
15	Bees Algorithm's flowchart diagram.....	34
16	Flower image segmented.....	51-52
17	Mountain image segmented.....	52-53
18	Satellite image segmented.....	53-54
19	Pepper image segmented.....	55-57
20	Synthetic noise image one segmented.....	57-58

21	Synthetic noise image two segmented.....	58-59
22	Texture image one segmented.....	59-60
23	Texture image two segmented.....	61-62
24	Time comparison for each algorithm.....	62
25	Accuracy and KHAT comparison.....	67

### LIST OF ALGORITHM PSEUDO-CODES

Algorithm	Caption	Page
1	Pseudo-code for particle swarm optimization.....	20
2	Ant Colony Optimization Pseudo-code.....	22
3	Artificial Bee Colony pseudo-code.....	26
4	Pseudo-code for IBMO.....	27
5	Bee Swarm Optimization pseudo-code.....	29
6	BeeHive Pseudo-code.....	31
7	The bee algorithm pseudo-code.....	32
8	Hill-climbing algorithm.....	36
9	Simulated Annealing pseudo-code.....	37
10	Metropolis pseudo-code.....	38
11	Pseudo-code for one iteration of Gibbs Sampler.....	40
12	Adaptive segmentation.....	43

### LIST OF TABLES

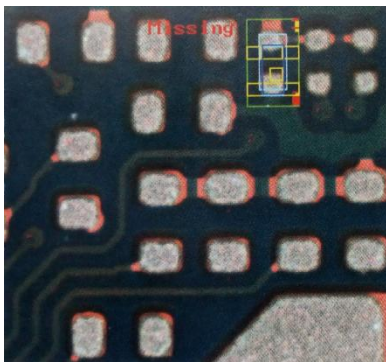
Table	Caption	Page
1	Real and artificial ant differences.....	22
2	The bee algorithm parameters.....	32

3	Bee algorithm parameter settings.....	50
4	Time comparison table.....	63
5	Error Matrix for Satellite image using Bee Algorithm.....	64
6	Error Matrix for Satellite image using ICM Algorithm.....	65
7	Error Matrix for Satellite image using MMD Algorithm.....	65
8	Error Matrix for Satellite image using Metropolis Algorithm.....	66
9	Error Matrix for Satellite image using Gibbs Sampler.....	66
10	Accuracy and KHAT comparison for all algorithms.....	67

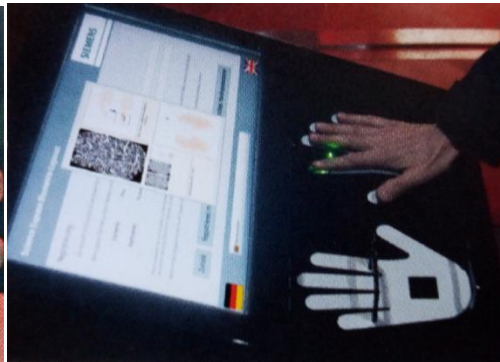


## I INTRODUCTION

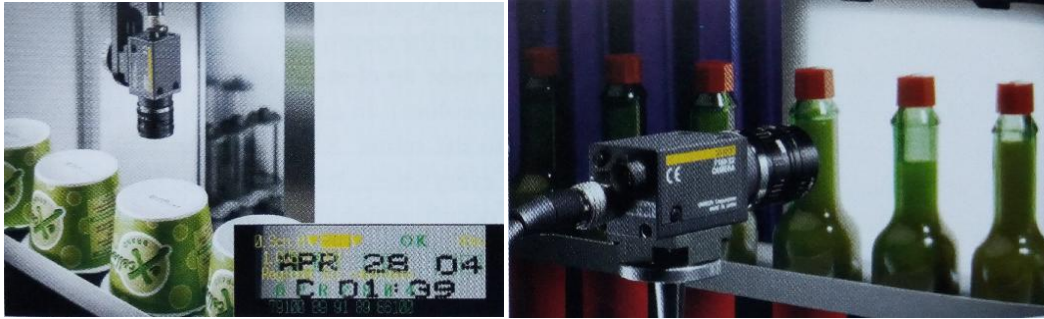
Did you know that it is estimated that 25% to 50% or more of human brain activity is spent in processing visual perceptions? In reality, we make sense of our visual perceptions intuitively and therefore, without thinking about it, we understand it to be a simple task. However, making sense of visual feed is a very complex function. Likewise, mistakenly computer vision was understood to be a simple task because of its influence from human vision. What exactly is computer vision? "Computer vision is the automatic analysis of images and videos by computers in order to gain some understanding of the world" (Dawson-howe, 2014). There are uses of computer vision applications across many disciplines such as inspection of circuit boards, inspection of labels, inspection of how full bottles are, reading license plates, biometric security, landmine detection (Dawson-howe, 2014), and healthcare (Grath, 2003) as shown in Figure 1, to mention a few.



(a)

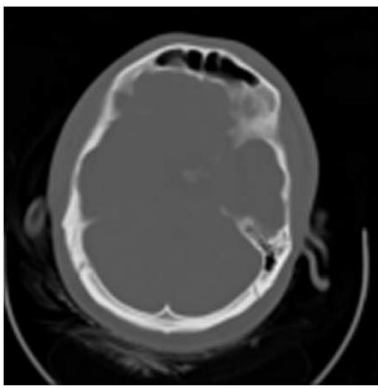


(b)



(c)

(d)



(e)

**Figure 1:** Computer vision applications, a) circuit board inspection, b) biometric security, c) label inspection, d) inspection of how full glasses are, and e) brain image using CT.

Many of the computer vision applications are based on one of the early image analysis tasks known as image segmentation. Image segmentation is a process by which image pixels are grouped into homogenous groups, causing the image to be segmented or in other words to be split into homogenous regions known as segments. Homogeneity can be measured against some characteristics. In our case we are interested in color image segmentation, which means that color values are used to measure the homogeneity of the pixels (Kato, 2001). There are other characteristics such as texture and intensity that can

also be used to measure homogeneity of image pixels. However, regardless of which characteristic or combination of characteristics are used in homogeneity measure or what segmentation techniques are used, segmenting an image with perfect accuracy is a very difficult task to achieve (Sag, 2015).

There are many different ways of performing image segmentation. One such way is to use Markov Random Field (MRF) as image segmentation model. MRF will be explained in details in chapter two, however, it is important to notice that there are many different implementations of MRF models that have shown great success in image segmentation. There are four MRF implementations using Iterated Conditional Model, Modified Metropolis Dynamic, Metropolis and Gibbs Sampler presented in the demo of (Kato, 2001). It is also important to notice that there are many swarm based algorithms, including algorithms based on bee swarm intelligence, used to implement image segmentation which have shown success as well (Sag, 2015). Therefore, it seems natural to combine these two methods together, especially knowing that MRFs are a great way of representing images and bee swarm based algorithms are a great optimization tool. Considering that until the writing of this thesis there does not seem to be a MRF model implementation based on the bee swarm, in this thesis we introduce an image segmentation algorithm using the bee algorithm optimization in a Markovian framework. A number of different images are used to test the solution and presented in chapter four.

The rest of this thesis is organized as follows. In chapter two, we give a brief literature survey. In chapter three, we explain our proposed solution. In chapter four, we go over the experiments and findings. In chapter five, we provide a conclusion. Finally,

in Appendix we provide an overview of some of the probability and statistics concepts related to this thesis.

## II LITERATURE SURVEY

In this chapter, we organize the literature survey in a few sections. In the first section, we review the Markov Random Fields (MRF) and related concepts. In section two, we provide two examples of MRF models. In section three, we outline the swarm intelligence in general along with algorithms and then specifically bee swarm behavior and a few algorithm that are based on bee swarm behavior. In section four, we review a few other search algorithms. Finally, in section five, we summarize two papers where MRF and bee swarm are used in image segmentation.

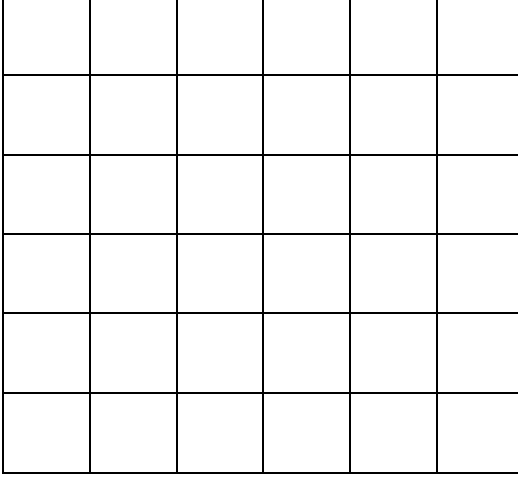
### 2.1 MARKOV RANDOM FIELDS

In the next few subsections we will try to explain the Markov Random Fields (MRF) and its related concepts.

#### 2.1.1 WHAT IS A MARKOV RANDOM FIELD?

Let us answer this question by considering the following example. Consider that we have an empty six by six grid, like the one shown in Figure 2, that represents an image. Now suppose that we want to assign a grayscale value to each one of the pixels/cells on the image by throwing a coin 255 times for each pixel and then assigning the number of heads to the given pixel in the image. Here, we randomly assigned grayscale values to each one of the cells in the field/image and therefore created a random field. In other words, a random field is a field that is created by performing a

random experiment for each of the cells on the field and assigning the outcome of the experiments to the given cells.



**Figure 2:** Six by Six Empty Field.

For any given cell the probability  $p(k)$  of getting  $k$  heads from  $n$  throws of the coin is calculated by using the probability distribution formula in equation 2.1 and 2.2:

$$p(k) = \binom{n}{k} \theta^k (1 - \theta)^{n-k} \quad 2.1$$

and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad 2.2$$

Here  $\theta$  represents the probability of getting a head per given throw.

Now consider that we ran the above experiment for each of the cells and we assigned a random generated value to each one of the cells of the grid/image. Once that is done, imagine that we modified the coin to become biased in such a way that when

calculating the grayscale value for each cell, the grayscale values of the neighboring cells are also taken into consideration. Please notice that only the neighboring cells are used to impact the value of the given cell and not all the cells. Which cells are considered to be neighboring cells will be explained later. Let us now assume that we want to modify the coin in this way as formulated in equation (2.3).

$$\frac{p(head|given\_values\_of\_nbrs)}{p(tail|given\_values\_of\_nbrs)} = e^s \quad 2.3$$

Here  $s$  represents a function of neighboring cell values. To calculate the bias of the coin, notice that

$$p(head | given\_values\_of\_nbrs) = \theta \text{ and,}$$

$$p(tail|given\_values\_of\_nbrs) = 1 - \theta$$

Therefore we get equation (2.4).

$$\frac{\theta}{1 - \theta} = e^s \Rightarrow \theta = \frac{e^s}{1 + e^s} \quad 2.4$$

If we use a coin, as the one explained above, to create the grayscale values per each cell on the grid, we create a Markov Random Field. In other words, a Markov random field is a random field that has Markovian property. The Markovian property is a property of a random field that defines the value of a given cell to be directly dependent only on the values of the neighboring cells and not the others.

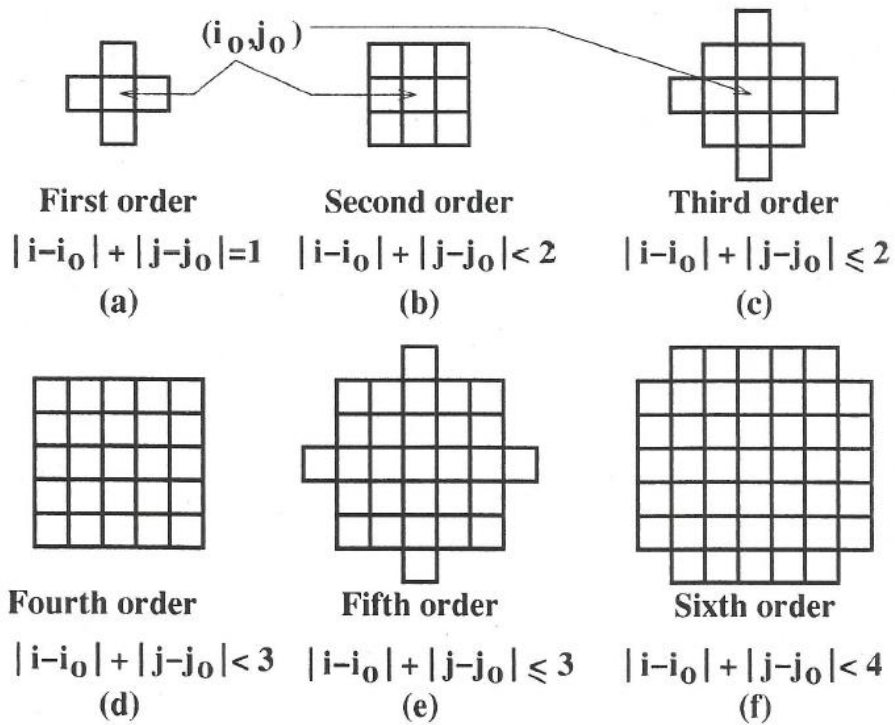
For the given example, we obtain a conditional probability function for a given grayscale value  $k$  for any cell given the neighboring cell values as in equation (2.5) (Petrou, 2006).

$$p(k|given\_values\_of\_nbrs) = \binom{n}{k} \frac{e^{ks}}{(1 + e^s)^n} \quad 2.5$$

### 2.1.2 NEIGHBORS AND NEIGHBORHOOD

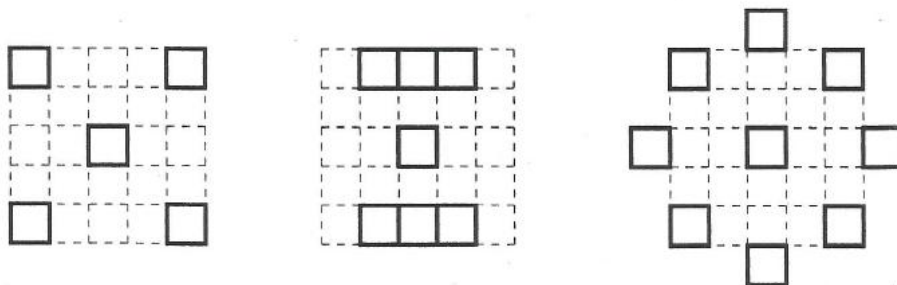
What cells or pixels are considered to be neighbors for a given cell depend on how neighborhood is defined. Usually neighborhoods are defined as spatial proximities, meaning the value of a given cell is directly dependent on spatially proximate cells, like the examples in Figure 3:





**Figure 3:** First order, second order, third order, fourth order, fifth order, and sixth order MRFs.

However, this is not always the case. In some cases neighborhood is defined to specifically include cells that are not directly connected but are in some spatial order like the examples below (Petrrou, 2006).



**Figure 4:** MRF neighborhood where cells are not directly connected.

### 2.1.3 MARKOV PROPERTIES

Earlier we briefly mentioned MRF properties, but let us take another look. When we mentioned the function  $s$  in equation (2.3), we said that it is a function of neighboring cell values. So, if we consider a first order Markov random field neighborhood and if we define the function  $s$  as in equation (2.6),

$$s = a(gl + gr) + b(gt + gb) \quad 2.6$$

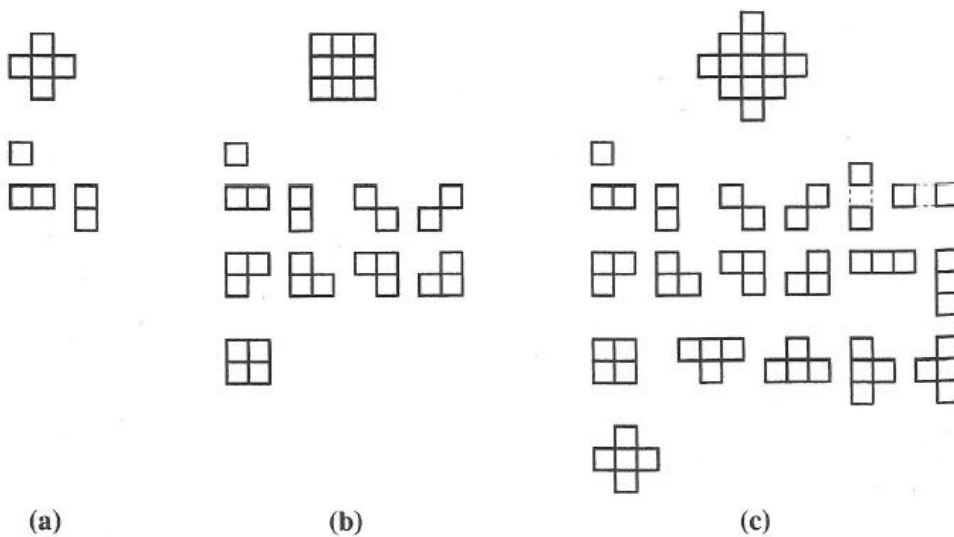
where  $gl$ ,  $gr$ ,  $gt$  and  $gb$  are the grayscale values for left cell, right cell, top cell, bottom cell respectively, as shown in Figure 5. In this case, parameters  $a$  and  $b$  are Markov properties (Petrrou, 2006).

		gt			
	gl	?	gr		
		gb			

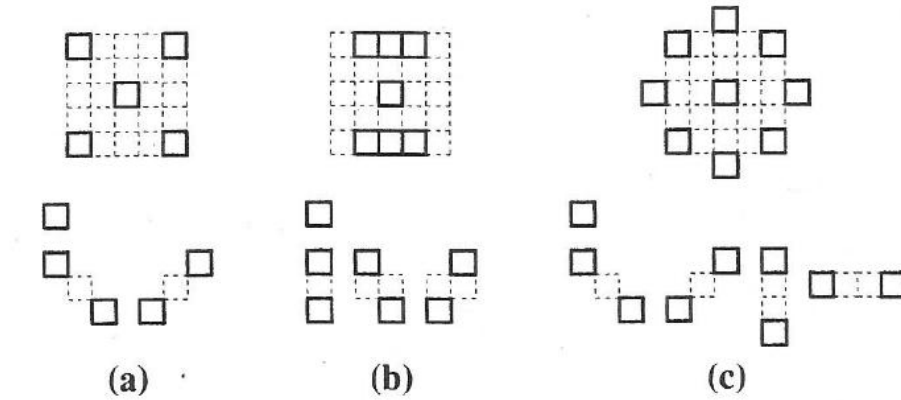
**Figure 5:** MRF properties.

### 2.1.4 CLIQUE

A clique is defined as a set of pixels that are neighbors according to the neighborhood definition. The following figures show different neighborhood structures with their corresponding cliques. For each neighborhood, a cell is a neighbor of itself. For every pair of cells they are neighbors of each other as well. On the second order MRFs we have triple and even quadruple cells that are neighbors. In the higher order MRFs, the structure of neighboring cells becomes even more complex. A clique that contains  $n$  cells is called  $n$ th order clique (Petrov, 2006).



**Figure 6:** a) First order MRF and its cliques, b) second order MRF and its cliques and c) third order MRF and its cliques.



**Figure 7:** Unconnected MRF neighborhoods with their corresponding cliques.

#### 2.1.5 CLIQUE POTENTIAL

Clique potential is defined as a function of cell values of all cells within a clique. The clique potential defines the relationship of clique cells. The form of the function and the numerical variables differ based on the clique order. Earlier when we talked about Markov properties and we defined function  $s$  to be  $s = a(gl + gr) + b(gt + gb)$ , in that case function  $s$  itself is a clique potential (Petrrou, 2006).

#### 2.1.6 ENERGY FUNCTION

When we deal with the Markov random fields in image segmentation the energy function is the sum of all clique potentials within the same lattice. The value of clique potentials depends on the clique configurations (Grath, 2003; Li, 2006). The role of energy function is classified as one of the following two kinds; first, as a global quality

measure of the solution in a quantitative way, and second, as a search guide for a minimal solution (Li, 2006).

## 2.1.7 EQUIVALENCE BETWEEN MARKOV RANDOM FIELDS AND GIBBS RANDOM FIELD

Markov random fields are characterized by Markovian properties or Markovianity and Gibbs Random Fields are characterized by Gibbs distribution. Hammersley-Clifford theorem states that the two are equivalent. There are many proofs for this theorem, one in (Li, 2006), however, the proof will not be provided in this thesis.

## 2.2 MARKOV MODELS

To better understand Markov random fields let us take a look at a couple of simple yet useful MRF models.

### 2.2.1 MARKOV CHAINS

The Markov Chain model is often considered to be the simplest Markov model. In Markov Chain model, any given sequence of random variables, say  $X=(X_1, X_2, \dots)$ , has a joint probability distribution as in equation (2.7).

$$P(X_i | X_{i-1}, X_{i-2}, \dots, X_1)$$

2.7

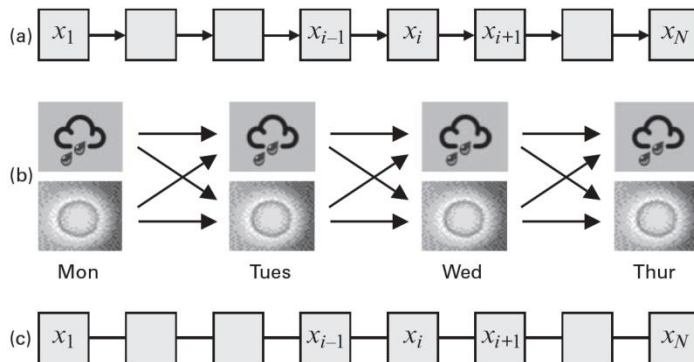
In other words, in the first order Markov Chain, the probability of any given variable is directly impacted only by its first direct neighbor. A simple yet powerful example that helps us better understand Markov Chain is weather prediction model. In the weather prediction model, the weather at any given day can be either sunny or raining, and we note that as below.

$$X_i \in \mathcal{L} = \{sunny, rainy\} \quad 2.8$$

This means that in the simplest form of Markov Chain, the weather at any given day is explicitly impacted only by the weather on the day before. However, it is implicitly impacted by many days before by the "knock-on effect." The first order Markov Chain assumes that:

$$P(X_i | X_{i-1}, X_{i-2}, \dots, X_1) = P(X_i | X_{i-1}) \quad 2.9$$

This means that the impact of weather conditions of many days before a given day  $X_i$  is equaled to the impact of the weather condition on day  $X_{i-1}$ . This is graphically illustrated below.



**Figure 8:** First order of Marko Chain for weather prediction; a) Directed graph representation of conditional dependencies, b) State transition diagram of weather forecast and c) Undirected graph representation.

The conditional probability for the weather prediction model is a 2X2 matrix. An example is given in Figure 9. In this example, if yesterday was raining, the probability of it being raining today is 0.4 and the probability of it being sunny today is 0.6. However, if yesterday was sunny, the probability of it being raining today is 0.8 and the probability of it being sunny today is 0.2

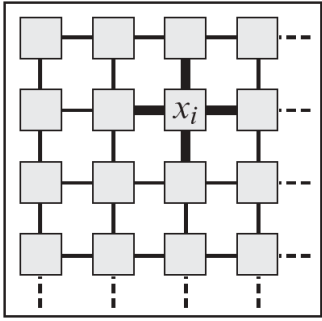
		Yesterday ( $X_{i-1}$ )	
		Rain	Sun
Today ( $X_i$ )	Rain	0.4	0.8
	Sun	0.6	0.2

**Figure 9:** Weather prediction 2X2 matrix

As we stated earlier in the first order Markov Chain model of weather prediction, the weather at any given day is implicitly impacted by the weather condition many days before in the row. So, it is important to notice that the impact that conditions of many days in advance have on the condition on a given day can be calculated by multiplying the weather condition matrices in the same order. Another important fact to notice is that the higher order Markov Chains models can also be used to solve other problems. One good example is the text prediction. Only one letter is not enough to predict what is the next letter in a word, however, two letters often can be used to predict the third one. In this case we have a second order Markov Chain models (Blake, 2011).

### 2.2.2 ISING MODEL

Ising model is another MRF model. Let us consider Ising model with one parameter  $\omega = \{\gamma\}$ . Its state space is based on Boolean variable  $x_i \in \mathcal{L} = \{0, 1\}$  and the energy function  $E(\mathbf{x})$ . Considering that the energy function takes in a Boolean variable and produces a real energy value, it is called "Pseudo-Boolean Function (PBF)." The maximal cliques in a rectangular graph (i.e. field of pixels), as shown in Figure 10, are the vertical and horizontal connected edges.



**Figure 10:** Ising model graph.

In this model the cliques are made of two pixels/nodes and therefore their clique potentials are based on pairs of cliques and are known as "pairwise potentials" and are represented as  $\Psi_{ij}(x_i, x_j) = \gamma |x_i - x_j|$ . Here  $\gamma$  increases the energy  $E$ , and decreases the joined probability  $P(\mathbf{x})$  by  $e^\gamma$  when adjacent  $x_i$  and  $x_j$  are different. This increases the probability of configuring a more homogenous  $\mathbf{X}$ . In other configurations,  $\mathbf{X}$  may be more agreeing or alike adjacent pixel values (Blake, 2011).



## 2.3 SWARM INTELLIGENCE

Swarm Intelligence is defined somewhat differently by researchers but essentially it is a collective of many self-organized agents within a decentralized organism to find an optimal solution to a problem. A swarm intelligent system is made of many simple self-organized agents that follow simple rules to interact with one another and the environment at local level and in somewhat random order. Agents interact locally without a centralized mechanism to control their behavior and without a sense of awareness of the global impact. This leads to a global dynamic structure or behavior that is known as swarm intelligence (Yuce, 2013 ; Swarm i., 2016). Swarm Intelligence is usually inspired by nature, especially, biological systems; however, it now can be found in many artificial systems as well. Examples of natural swarm intelligent systems include ant colonies, bird flocking, fish schooling and of course honey bees. See Figures 11 and 12 below for visual representation of natural and artificial swarm examples (Swarm b., 2016).



(a)



(b)



(c)



(d)

**Figure 11:** Natural Swarm Systems, a) ant colony, b) bird flocking, c) fish schooling, and d) honey bees swarming.



**Figure 12:** Artificial Swarm System. One thousand robot swarm, named Kilbot, developed by Harvard University.

The self-organization of swarm intelligence is made possible as a result of four characteristics. First rule is positive feedback, which is a set of simple rules helping generate the structure of swarm intelligent organism. An example of positive feedback is recruitment of honey bees to flower patches. Second element is negative feedback. It reduces the effect of positive feedback and helps maintain a balanced mechanism. An example of negative feedback is a limited number of foragers. Third element is randomness. It helps create new (and unpredicted) solutions to given problems. The final element is interaction between local agents. There should be a limited number of agents who can interact with each other. The combination of the above mentioned four rules or elements makes possible the creation of a decentralized system or structure. In this system there is no central control and the hierarchy is only to divide the duties and not to oversee individual agents. In this way a dynamic and efficient system is created enabling it to overcome given challenges (Yuce, 2013).

### 2.3.1 PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM

Particle swarm optimization algorithm is a search algorithm to find optimal solution to given problems in  $n$  dimensional search space. It is based on the collective behavior of animals (i.e. swarm intelligence) such as flocking of the birds. PSO is described through "position" of the swarm in the  $n$ -dimensional space and the "velocity" or the rate of change of the swarm's position. Each member of the population, or swarm position in the  $n$ -dimensional space, is known as a particle, and from that the name Particle Swarm Optimization comes from. Each particle keeps track of its best position.

In addition, it communicates its best position with the neighboring particles. Therefore, each particle is not only aware of its best positions so far but that of its neighbors as well. Each particle changes its position and velocity by measuring against a fitness value and by taking under consideration its historical information and that of its neighboring particles (Rini, 2011; Ahmed, 2012). The following equations represent velocity and position (Rini, 2011).

$$\begin{aligned} v_{id(t+1)} &= v_{id(t)} + c_1 \mathbf{R}_1(p_{id(t)} - x_{id(t)}) + c_2 \mathbf{R}_2(p_{gd(t)} - x_{id(t)}) \\ x_{id(t+1)} &= x_{id(t)} + v_{id(t+1)} \end{aligned} \quad 2.10$$

Where:

$v_{id}$  represents velocity of the  $i^{\text{th}}$  particle in the  $d^{\text{th}}$  dimension at  $t$  iteration.

$x_{id}$  represents position of the  $i^{\text{th}}$  particle and  $d^{\text{th}}$  dimension and  $\mathbf{x}_i$  is the  $i^{\text{th}}$  particle itself.

$p_{id}$  represents best historic position of the  $i^{\text{th}}$  particle in the  $d^{\text{th}}$  dimension

$p_{gd}$  represents the position swarm's global best particle  $\mathbf{x}_g$

$\mathbf{R}_1$  and  $\mathbf{R}_2$  represent n-dimensional vectors with random number, introducing randomness to the search

$c_1$  and  $c_2$  are positive constants, known as cognitive and social parameters, respectively, controlling the importance of individual particle in comparison to that of the global swarm

Below is the pseudo-code for particle swarm optimization algorithm (Ahmed, 2012).

**Algorithm 1:** Pseudo-code for particle swarm optimization

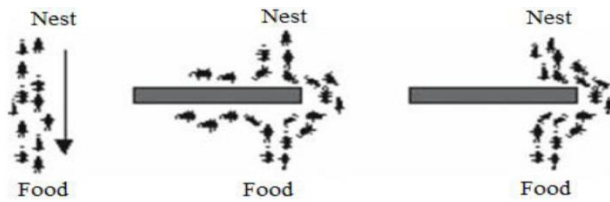
1. *Initialize the swarm by randomly assigning each particle to an arbitrarily initial velocity and a position in each dimension of the solution space*
2. *Evaluate the desired fitness function to be optimized for each particle's position*
3. *For each individual particle, update its historically best position so far,  $P_i$ , if its current position is better than its historically best one*
4. *Identify/update the swarm's globally best particle that has the swarm's best fitness value, and set/reset its index as  $g$  and its position at  $P_g$ .*
5. *Update the velocities of all the particles using velocity equation*
6. *Move each particle to its new position using position equation*
7. *Repeat steps 2 - 6 until convergence or a stopping criterion is met*

### 2.3.2 ANT COLONY OPTIMIZATION

Ant colony optimization is based on the swarm behavior of ant colonies. Ant colonies find the shortest path from their nest to the food source by depositing or laying a substance called pheromone (Selvi, 2010; Ahmed, 2012). Pheromone comes from Greek word pherein which means to transport and hormone to stimulate. In other words, pheromone means to stimulate transportation, or in ant colony optimization case to stimulate the path of transportation (Ahmed, 2012). If an obstacle is placed on the path of the ants, they will go randomly on the left and the right leaving pheromone on their way. As more ants go to the food source and back more pheromone will be placed on the shortest path because the ants will be able to come back faster. As more pheromone is



placed on the shortest path, eventually all ants will take the shortest path to the food and back because they tend to follow the path with more pheromone. A graphical representation is sketched in Figure 13 (Selvi, 2010; Ahmed, 2012).



**Figure 13:** Ant path stimulation.

In the case of ant colony optimization, the problem is represented on weighted graph and the artificial ants move from one node to the other setting weights on graph edges for path optimization. There are some difference between real ants and artificial ants. The table below lists some of these differences.

**Table 1:** Real and artificial ant differences.

Criteria	Real Ants	Artificial Ants
<i>Pheromone Depositing Behaviour</i>	Pheromone is deposited both ways while ants are moving (i.e. on their forward and return ways).	Pheromone is often deposited only on the return way after a candidate solution is constructed and evaluated.
<i>Pheromone Updating Amount</i>	The pheromone trail on a path is updated, in some ant species, with a pheromone amount that depends on the quantity and quality of the food [31].	Once an ant has constructed a path, the pheromone trail of that path is updated on its return way with an amount that is inversely proportional to the path length stored in its memory.
<i>Memory Capabilities</i>	Real ants have no memory capabilities.	Artificial ants store the paths they walked onto in their memory to be used in retracing the return path. They also use its length in determining the quantity of pheromone to deposit on their return way.
<i>Return Path Mechanism</i>	Real ants use the pheromone deposited on their forward path to retrace their return way when they head back to their nest	Since no pheromone is deposited on the forward path, artificial ants use the stored paths from their memory to retrace their return way.
<i>Pheromone Evaporation Behaviour</i>	Pheromone evaporates too slowly making it less significant for the convergence.	Pheromone evaporates exponentially making it more significant for the convergence.
<i>Ecological Constraints</i>	Exist, such as predation or competition with other colonies and the colony's level of protection.	Ecological constraints do not exist in the artificial/virtual world.

To pull all of these together let us take a look at the pseudo-code for ant colony optimization algorithm (Ahmed, 2012).

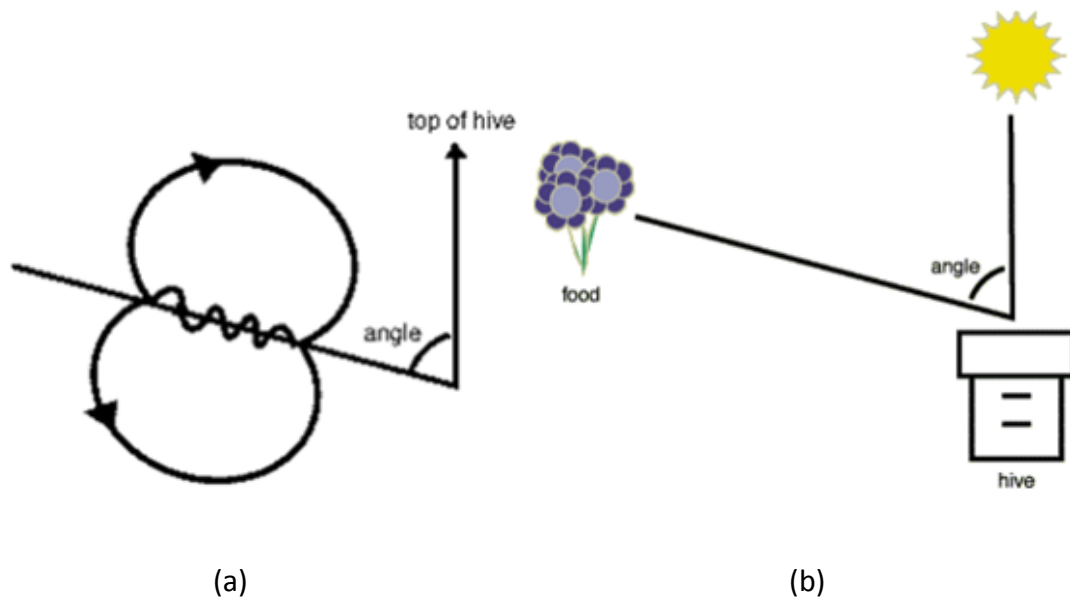
**Algorithm 2:** Ant Colony Optimization Pseudo-code

1. *Represent the solution space by construction graph*
2. *Set ACO parameters and initialize pheromone*
3. *Generate ant solution from each ant's walk on the construction graph mediated by pheromone trail*
4. *Update pheromone intensities*
5. *Go to step 3 and repeat until convergence or termination conditions are met*

### 2.3.3 HONEY BEE BEHAVIOR

Before we review a number of algorithms inspired by honey bee swarms, let us first look at the natural behavior of honey bees. Honey bees go over 10 km away in multi-directions in search for flower patches that have quality pollen or nectar. A number of honey bees known as scouts go in random directions harvesting and evaluating flower patches. Once they return to the hive, they deposit the nectar and then go to a place inside the hive known as the dance floor. There the scouts inform other bees about flower patches they found through what is known as the waggle dance. The waggle dance takes its name from the wagging run which produces a loud buzzing sound while bees move their body from one side to the other. Through this dance scouts inform other bees about the quality of the flower patch they found, the direction to the patch and how far it is. The path of wagging dance is shaped as number eight. The scout starts wagging on a

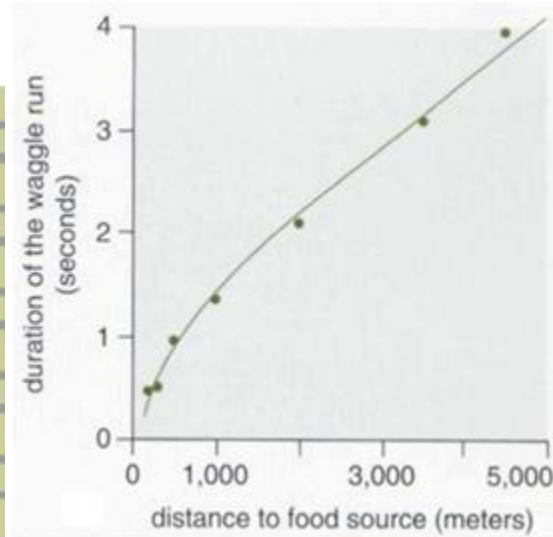
straight line which creates an angle to the top of the hive. This represents the angle between the direction of the sun and the flower patch, giving other bees the direction to the flower patch. Then, the bee alternates turning right and left. The speed and duration of the dance indicates the distance of the flower patch. However, the speed of the waggle and the buzzing sound indicate the quality of the patch. The better the quality the more bees will be recruited. Once the scout finishes the dance, it goes to the flower patch again to harvest and evaluate. If the flower patch is still good, the process of recruiting new bees and going again to the same patch continues. The recruited bees upon return will do the waggle dance if the flower patch is still evaluated as in good condition (Pham, 2005; Yuce, 2013). To visualize the waggle dance, see Figure 14 below.







(c)



(d)

**Figure 14** : Waggle dance. a) waggle dance angle; b) orientation to the food source, from the hive utilizing sun to create the angle; c) waggle dance on the dance floor (Yuce, 2013) and d) graph of distance to food source per duration of the waggle(Seeley, 2006).

#### 2.3.4 ARTIFICIAL BEE COLONY (ABC)

Artificial bee colony (ABC) is one of the optimization algorithms influenced by the honey bees. In the ABC algorithm, the food source represents a solution in the solution space and the nectar amount represents the fitness of the given solution. In addition, there are three groups of bees in the ABC algorithm. First, the employed bees are the bees that collect the nectar. Second, the onlookers are bees that wait at the dance floor for employed bees to return. Finally, the scouts are bees that search for solutions, food sources, randomly. The number of employed bees or onlookers is equals to the number of solutions in the solution space. There are three parameters to be estimated in

the ABC algorithm. First, the population size, SN, or the number of food sources.

Second, the maximum cycle number, MCM, or the maximum number of generations

before the algorithm has to terminate. Finally, the number of times a food source (i.e.

limit) is visited without any improvements before a given food source is abandoned.

Before we give a brief explanation of each step, let us take a look at the pseudo-code for the ABC algorithm.

**Algorithm 3:** Artificial Bee Colony pseudo-code

1. *Initialize the ABC and problem parameters*
2. *Initialize the Food Source Memory (FSM)*
3. *repeat until termination criterion are met*
  - a. *Send the employed bees to food source*
  - b. *Send the onlookers to select a food source*
  - c. *Send the scouts to search possible new food*
  - d. *Memorize best food source*

In the first step, solutions are selected at random as well as the population size, maximum cycle number, and the limit. In the second step, the food source memory is initialized with information for each food source. Then, the employed bees are assigned to given food sources. Also, the onlookers are sent to food sources based on the information from employed bees. Each onlooker selects a food source with highest probable fitness calculated as below.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

where  $fit_i$  is the fitness for each food source and SN is similarly defined.

After that, as a food source is abandoned, the employed bees become scouts and randomly look for new food sources. Finally, at the end of each generation, the best food source so far is memorized. The process of sending employed bees, onlookers, and scouts and memorizing the best solution repeats until the maximum number of cycles is met (Bolaji, 2013; Karaboga, 2009).

### 2.3.5 IMPROVED BEE COLONY ALGORITHM FOR MULTI-OBJECTIVE OPTIMIZATION (IBMO)

The IBMO algorithm is an improvement to ABC for use with multi-objective optimization that considers "non-dominated sorting strategy and principal concepts of Pareto-Optimal," and it also includes an improvement step to local search. The algorithm begins by creating a initial solution through a process known as diversification technique where the parameter ranges are divided equally in subranges and eachsub range is given a frequency of zero. Then, the parameters are calculated for each selected subrange by increasing the frequency by one and the probability of selecting a subrange is inversely proportional to the frequency. After the initial solution, there are three major steps to each iteration, the employed, onlooker and scout bee steps. During the employed bee phase for every solution, a new solution is created between the existing solution and a randomly selected neighbor from the food set. Then, an improvement step which uses a local search begins. If the new solution dominates, the new solution is added to the food set in place of the existing solution. Finally, in employed bee phase, the non-dominated solutions are

updated in external archive (EXA). During the onlooker bee phase, the IBMO determines the neighbors from the EXA and not from food source because the best solutions are stored in EXA. During the scout phase, for every solution that was not selected by employed or onlooker phases the trail count is increased by one. If it exceeds a preset value or there are no improvements, these solutions are regenerated and their trail values are set to zero. After the scout phase, the foods with EXA are combined and sorted to create the food set for the next iteration. Below is the pseudo-code for IBMO (Sag, 2015).

**Algorithm 4:** Pseudo-code for IBMO

if $\delta$ equals to zero; Set $\delta = \text{rand}(-0.5, 0.5)$ otherwise continue
Set $\text{newSol} = v_i$
Set $\text{temp} = v_i$
Set $\text{limit\_count} = 0$
REPEAT
Set $\text{limit\_count} = \text{limit\_count} + 1$
Set $\text{temp}_j = \text{temp}_j + \delta * \text{limit\_count}$
if $\text{temp}$ dominates $\text{newSol}$ ; break
Set $\text{newSol} = \text{temp}$
UNTIL $\text{limit\_count} < \text{limit}$
Store $\text{newSol}$ into $\text{Foods}$

### 2.3.6 BEE SWARM OPTIMIZATION ALGORITHM

Bee swarm optimization (BSO) is an optimization algorithm based on the food foraging behavior of the honey bees. In this algorithm there are three kinds of bees: experienced foragers, onlookers, and scouts, in D-dimensional space  $S \in R^D$ . The number of each type of bees in swarm  $\beta$  are set manually; however, the number of experienced foragers and onlookers is equivalent and the number of scouts is smaller. The swarm is represented symbolically as  $\beta = \xi \cup \kappa \cup \vartheta$ . The solution for every bee in D-dimensional

space  $S$  is represented by the vector  $\vec{x}(\beta, i) = (x(\beta, i1), x(\beta, i2), \dots, x(\beta, iD))$ . The fitness of the solution (i.e. the quality of the food source) is represent by  $fit(\vec{x}(\beta, i))$ .

The algorithm starts by setting the parameters and an initial solution. Once an initial solution is created for every iteration, the bees are sorted per their fitness. The food sources for the group of bees with lowest fitness is abandoned and those bees are turned into scout bees. The scouts are sent in random direction to look for new food sources. The other bees are divided in the equal number where the group with the highest fitness are set as experienced foragers and the other half as onlookers. The process continues until terminating condition is met (Akbari, 2009). The pseudo-code is listed below.

**Algorithm 5:** Bee Swarm Optimization pseudo-code

**Initialization:**

Determine  $n(\beta)$ , percentage of bees,  $D$ , and  $\tau$   
**For**  $i = 1$  **to**  $n(\beta)$   
 $\vec{x}_{init}(\beta, i) = Init(i, S)$   
**Next**  $i$

**Do**

**Compute** fitness of the bees  
**Sort** bees based on their fitness  
**Partition** the swarm into the experienced forager,  $\zeta$ ,  
onlooker,  $\kappa$ , and scout,  $\vartheta$ , bees

**For**  $i = 1$  **to**  $n(\zeta)$   
**if**  $fit(\vec{x}(\zeta, i)) > fit(\vec{b}(\zeta, i))$  **then**  $\vec{b}(\zeta, i) = \vec{x}(\zeta, i)$   
**if**  $fit(\vec{b}(\zeta, i)) > fit(\vec{e}(\zeta, \bullet))$  **then**  $\vec{e}(\zeta, \bullet) = \vec{b}(\zeta, i)$

**Next**  $i$

**For**  $i = 1$  **to**  $n(\zeta)$   
**For**  $d = 1$  **to**  $D$   
 $x_{new}(\zeta, id) = x_{old}(\zeta, id) + \omega_b r_b (b(\zeta, id) - x_{old}(\zeta, id))$   
 $+ \omega_e r_e (e(\zeta, \bullet d) - x_{old}(\zeta, id))$   
**Next**  $d$   
**Next**  $i$

```

For  $i = 1$  to  $n(\kappa)$ 
  Select elite bee
  For  $d = 1$  to  $D$ 
     $x_{new}(\kappa, id) = x_{new}(\kappa, id) + \omega_e r_e (e(\xi, id) - x_{old}(\kappa, id))$ 
  Next  $d$ 
Next  $i$ 

For  $i = 1$  to  $n(\vartheta)$ 
  For  $d = 1$  to  $D$ 
     $x_{new}(\vartheta, id) = x_{old}(\vartheta, id) + R_w(\tau, x_{old}(\vartheta, id))$ 
  Next  $d$ 
Next  $i$ 
Adjust radius  $\tau$  and step size  $s$ 
Until termination condition is met

```

### 2.3.7 BEEHIVE ALGORITHM

BeeHive algorithm is another algorithm based on the behavior of the honey bees.

In BeeHive the dance floor is represented by a routing table where bees arriving at a given node from different neighbors can exchange information. Bees are grouped into two groups; short distance forgers, which explore nodes close from the hive, and long distance forgers which explore nodes farther away. In BeeHive the search space or network is divided into so called foraging regions. Each foraging region is represented by a node with the lowest IP address (i.e. closest node). In addition, each node has a foraging zone which includes all nodes that a short distance bee can go to from a given node. Each non-representative node sends bee agents to nodes in the bee zone updating the path information. Likewise, the representative nodes send bee agents to long distance nodes. In this way each node maintains an up-to-date routing information. The next node on the path is selected in probabilistic way with accordance to the quality of the neighbors. This means that not every bee agent is sent through the best possible path, but this helps with maximizing performance. Below is the pseudo-code (Wedde, 2004).

**Algorithm 6:** BeeHive Pseudo-code

```

t := current_time, tend := time to end simulation
// Short_Limit := 7, Long_Limit := 40, Bee_Generation_Interval := 1
// i = current node, d = destination node, s = source node
// n = successor node of i, p = predecessor node of i
// z = Representative node of the foraging region containing s
// w = Representative node of the foraging region containing d
// q is queuing delay estimate of a bee agent from node p to s
// p is propagation delay estimate of a bee agent from node p to s
 $\Delta t$  := Bee_Generation_Interval,  $\Delta h$  := hello packet generation interval
bip := estimated_link_band_width_to_neighbor_p
pip := estimated_propagation_delay_to_neighbor_p
hi := hop limit for bees of i, lip := size_normal_queue_i_to_p (bits)
foreach Node // concurrent activity over the network
  while (t ≤ tend)
    if (t mod  $\Delta t$  = 0)
      if (i is representative node of the foraging region)
        set hi := Long_Limit, bi is long distance bee agent
      else
        set hi := Short_Limit, bi is short distance bee agent
      endif
      launch a bee bi to all neighbors of i
    endif

  foreach bee bs received at i from p
    if (bs was launched by i or its hop limit reached)
      kill bee bs
    elseif (bs is inside foraging zone of node s)
       $q := q + \frac{l_{ip}}{b_{ip}}$  and  $p := p + p_{ip}$ 

      update IFZ routing table entries  $q_{ps} = q$  and  $p_{ps} = p$ 
      update  $q$  ( $q := \sum_{k \in N(i)} (q_{ks} \times g_{ks})$ ) and  $p$  ( $p := \sum_{k \in N(i)} (p_{ks} \times g_{ks})$ )
    else
       $q := q + \frac{l_{ip}}{b_{ip}}$  and  $p := p + p_{ip}$ 
      update IFR routing table entries  $q_{pz} = q$  and  $p_{pz} = p$ 
      update  $q$  ( $q := \sum_{k \in N(i)} (q_{kz} \times g_{kz})$ ) and  $p$  ( $p := \sum_{k \in N(i)} (p_{kz} \times g_{kz})$ )
    endif
    if (bs already visited node i)
      kill bee bs
    else
      use priority queues to forward bs to all neighbors of i except p
    endif
  endfor

```

```

foreach data packet  $d_{sd}$  received at  $i$  from  $p$ 
  if ( node  $d$  is within foraging zone of node  $i$  )
    consult IFZ routing table of node  $i$  to find delays to node  $d$ 
    calculate goodness of all neighbors for reaching  $d$  using equation 2
  else
    consult FRM routing table of node  $i$  to find node  $w$ 
    consult IFR routing table of node  $i$  to find delays to node  $w$ 
    calculate goodness of all neighbors for reaching  $w$  using equation 2
  endif
  probabilistically select a neighbor  $n$  ( $n \neq p$ ) as per goodness
  enqueue data packet  $d_{sd}$  in normal queue for neighbor  $n$ 
endfor
if (  $t \bmod \Delta h = 0$  )
  send a hello packet to all neighbors
  if (time out before a response from neighbor) (4th time)
    neighbor is down
    update the routing table and launch bees to inform other nodes
  endif
endif
endwhile
endfor

```

### 2.3.8 THE BEE ALGORITHM

The bee algorithm is an optimization algorithm inspired by the food foraging behavior of honey bees. The original bee algorithm which is what we use in this thesis takes in a number of parameters. Before we give a pseudo-code for the algorithm, let us take a look at each parameter on Table 2 below.

**Table 2:** The bee algorithm parameters.

Symbol	Meaning of parameters
$n$	Number of scouts
$m$	Number of sites selected out of $n$ visited
$e$	Number of best sites
$nep$	Number of bees recruited for best $e$ sites
$m-e$	Number of bees recruited for other sites
$nsp$	Selected sites
$ngh$	Neighborhood size

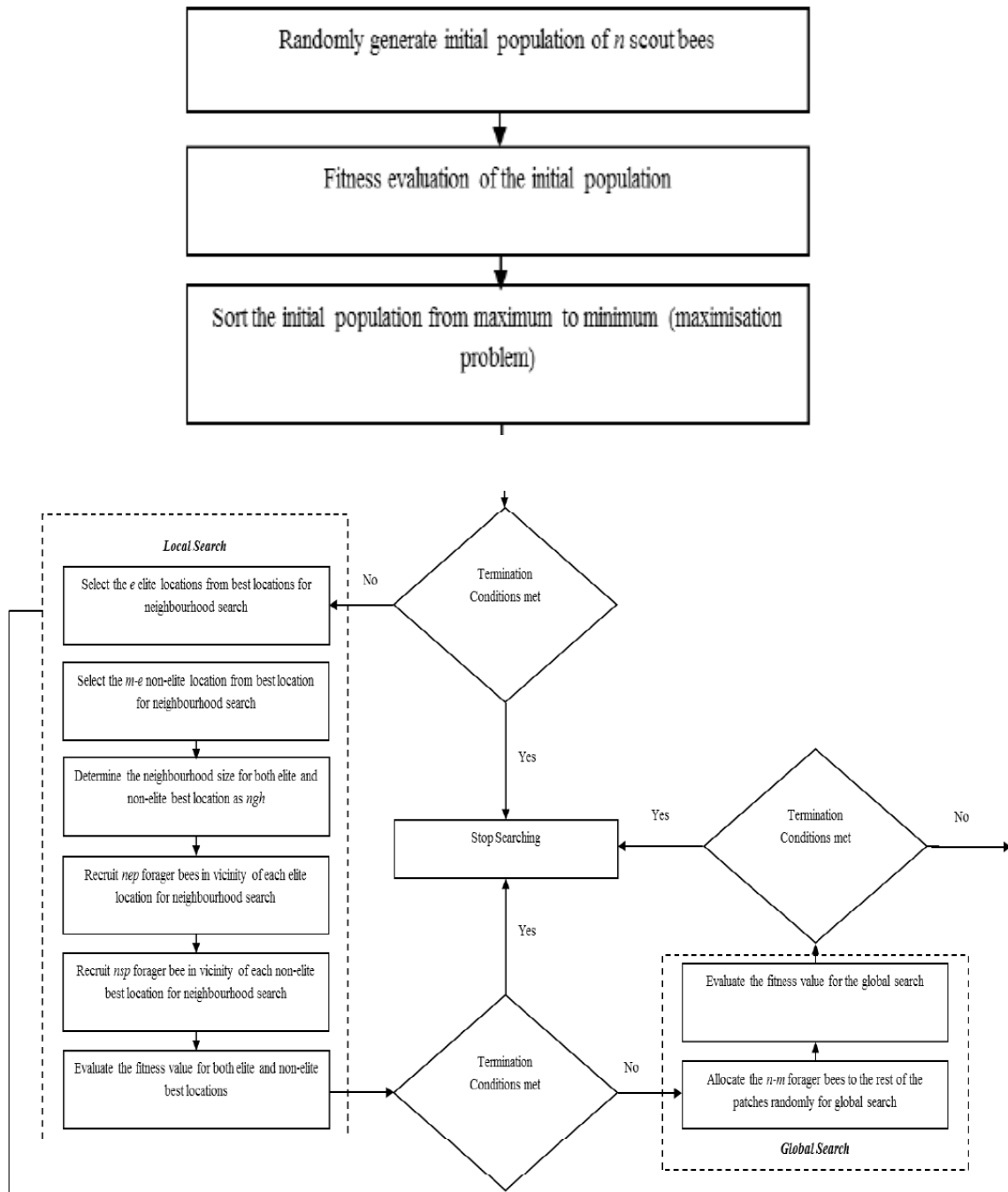


Now that we know what parameters the bee algorithm takes, let us take a look at the pseudo-code:

**Algorithm 7:** The bee algorithm pseudo-code

1. *Initialize the population with random solutions*
2. *Evaluate the fitness of the population*
3. *While (stopping criterion not met) //Forming new population*
4.       *Select sites for neighborhood search*
5.       *Recruit bees for selected sites (more bees for best e sites) and evaluate fitness*
6.       *Select the fittest bee from each patch*
7.       *Assign remaining bees to search randomly*
8. *End while*

As it can be seen above, the algorithm starts by placing  $n$  scouts randomly in the search space and then evaluating the fitness. Then in step 4, bees with highest fitness are selected and the sites they visited are chosen for neighborhood search. In steps 5 and 6, the algorithm performs neighborhood search. The bees can be chosen by evaluating the fitness and the fitness value is used as basis of probabilistic decision. More bees are used to improve search for the best sites. Important to notice that in step 6 only the best bee is selected to form the next generation of the population. In step 7, the unselected bees are placed randomly on search space looking for new flower patches. The process is repeated until the finishing criteria is met (Pham, 2005). To better understand the algorithm, the flow chart is given below (Yuce, 2013).



**Figure 15 :** A flowchart diagram of bee algorithms.

## 2.4 OTHER ALGORITHMS

So far, we have reviewed a number of search algorithms based on swarm intelligence. Now let us examine a few more search algorithms that are not based on swarm intelligence but some of them will be used in our thesis.

#### 2.4.1 MARKOV CHAIN MONTE CARLO (MCMC)

Before we assess a few more algorithms, it is important to notice that Markov Chain Monte Carlo (MCMC) methods are a set of algorithms based on Markov Chain model. They are used for sampling data from probability distributions (Markov, 2016). There are two such methods, Metropolis and Gibbs sampler, used in our literature survey. They will be explained later in this section.

#### 2.4.2 HILL-CLIMBING SEARCH

Hill-climbing is a search algorithm that tends to get the best possible solution through iteration towards the best solution. In each step, it chooses the next best solution by selecting the neighbor with the highest value. At each iteration, it replaces the current value with next best value. It iterates until it reaches the "peak" or the point where all its neighbors have a lower value than the current value. At this case, the algorithm will stop. Hill-climbing is classified as a greedy algorithm because it picks the next best value without understanding what happens next. Hill-climbing is usually fast because it is easy to find the next best solution. However, sometimes it gets stuck at a point that is not the optimal solution for the following reasons; local maxima issue which is a local maximum

or a peak that is higher or better value than all its neighbors but not the best global solution. Second reason is ridges, which are a series of local maximums. The third is plateaux issue which is a flat area or a point where the value of the best neighbor is equal to the current value. Below is the pseudo-code for hill climbing algorithm (Russell, 2009).

**Algorithm 8:** Pseudo-code for Hill-climbing algorithm

*Function Hill-climbing (problem) returns a state that is local maximum*

*current* <- *Make-Node(problem.Initial-State)*

*loop do*

*neighbor* <- *a highest-valued successor of current*

*if neighbor.Value* <= *current.Value* *then return current.State*

*current* <- *neighbor*

### 2.4.3 SIMULATED ANNEALING

Simulated annealing is a search algorithm that tries to solve Hill-climbing's problem of getting stuck at a local maximum and still performs efficiently. Let us consider the pseudo-code.

**Algorithm 9:** Simulated Annealing pseudo-code

*function Simulated-Annealing (problem, schedule) returns a solution state*

*inputs: problem(a problem), schedule (a mapping from time to "temperature")*

*current <- Make-Node(problem.Initial-State)*

*for  $t = 1$  to  $\infty$  do*

*$T <- \text{schedule}(t)$*

*if  $T = 0$  then return current*

*next <- a randomly selected successor of current*

*$\Delta E <- \text{next.Value} - \text{current.Value}$*

*if  $\Delta E > 0$  then current <- value*

*else current <- next only with probability  $e^{\frac{\Delta E}{T}}$*

As it can be seen from the algorithm above, the move is always accepted if the next value is better than current value. However, unlike the hill-climbing, in simulated annealing even if the next value is worse than current value, it is accepted based on a probability value. The amount of allowed worse values to be accepted decreases as the temperature decreases, making it harder to accept worse values towards the end of the process (Russell, 2009).

#### 2.4.4 METROPOLIS

Metropolis algorithm is used as a sampling algorithm to select random samples out of a random distribution  $P(x)$  when it is difficult to get direct samples. Metropolis is

considered as a Markov Chain Monte Carlo method. Metropolis works in an iterative way where at each iteration only the current sample value is used to generate the next value, making it a Markov chain method. The selection of the next value is based on a given probability, therefore if the new sample has a better probability of being part of the distribution it is selected and used to generate the next iteration sample, if not it is discarded and the old value is kept. The Metropolis pseudo-code is listed below.

**Algorithm 10:** Metropolis pseudo-code

*Initialization: Choose an arbitrary point  $x_0$  to be the first sample, and choose an arbitrary probability density  $g(x|y)$  which suggests a candidate for the next sample value  $x$ , given the previous sample value  $y$ . For the Metropolis algorithm,  $Q$  must be symmetric; in other words, it must satisfy  $g(x|y) = g(y|x)$ . A usual choice is to let  $g(x|y)$  be a Gaussian distribution centered at  $y$ , so that points closer to  $y$  are more likely to be visited next-making the sequence of samples into a random walk. The function  $g$  is referred to as the proposal density or jumping distribution.*

1. For each iteration  $t$ :

2.1 Generate a candidate  $x'$  for the next sample by picking from the distribution

$$g(x'|x_t)$$

2.2 Calculate the acceptance ration  $\alpha = f(x')/f(x_t)$ , which will be used to decide

whether to accept or reject the candidate. Because  $f$  is proportional to the density of  $P$ , we have that  $\alpha = f(x')/f(x_t) = P(x')/P(x_t)$ .

2.3 if  $\alpha \geq 1$ , then the candidate is more likely than  $x_t$ ; automatically accept the candidate by setting  $x_{t+1} = x'$ . Otherwise, accept the candidate with probability  $\alpha$ ; if the candidate is rejected, set  $x_{t+1} = x_t$ , instead.

From the pseudo-code above we can see that Metropolis randomly visits through the search space. In each iteration, it accepts the next selected sample if the probability is better. In other words, it tends to stay in the higher density regions (Metropolis-Hastings, 2016).

#### 2.4.5 ITERATED CONDITIONAL MODELS (ICM)

The ICM is an algorithm that uses the greedy strategy to maximize the local conditional probability in an iterative way. The ICM takes the label  $d$  and all other labels  $f_{S-\{i\}}^{(k)}$  and updates each label  $f_i^{(k)}$  into  $f_i^{(k+1)}$  by maximizing the conditional (posteriori) probability  $P(f_i | d, f_{S-\{i\}})$ . To use the conditional probability  $P(f_i | d, f_{S-\{i\}})$ , the following two assumptions are made; it assumes that the given function  $f$ , each component  $d_1, \dots, d_m$  is conditionally independent and every  $d_i$  has the same conditional density function  $p(d_i | f_i)$  which means

$$p(d | f) = \prod_i p(d_i | f_i) \tag{2.12}$$

where  $\prod$  represents the multiplication. The other assumption is that  $f$  is a Markovianity, which means it depends on local neighborhood labels. The two assumptions above and Bayes theorem give us

$P(f_i \mid d, f_{S-\{i\}}) \propto p(d_i \mid f_i) P(f_i \mid f_{N_i})$ , which is easier to estimate (Li, 2006).

#### 2.4.6 GIBBS SAMPLER

Gibbs sampler, like Metropolis, is a Markov Chain Monte Carlo method used to iteratively generate samples from probability distributions to be used to approximate joint distributions. Gibbs sampler is used when the conditional distribution of each variable is known but joint distribution is hard to calculate. Gibbs sampler starts with a given solution which can be generated at random or by using a specific method and then it iterates a given number of times (Gibbs, 2016). Below is the pseudo-code for a given iteration in Gibbs sampling (Garth, 2003).

**Algorithm 11:** Pseudo-code for one iteration of Gibbs Sampler

*repeat*

*Select a site  $i$  from set  $S$*

*Sample the conditional probability density of the label at site  $i$  given the labels in the neighborhood of site  $i$*

*Replace the old label with the label just sampled*

*until all sites in  $S$  have been sampled*

#### 2.4.7 MODIFIED METROPOLIS DYNAMIC (MMD)



The MMD is yet another algorithm that can be used for the local energy optimization. A new label is selected at random using the uniform distribution, however, the acceptance of the new state is done deterministically. The difference between the MMD and the original Metropolis algorithm is that in the original method the threshold is set randomly during each iteration, however, in MMD it is set as a constant at the beginning of the algorithm (Sziranyi, 2000).

## 2.5 APPLICATIONS OF MRF AND BEE BEHAVIOR ALGORITHMS FOR IMAGE SEGMENTATION

During the research in this study we did not find any papers using bee behavior algorithms to implement MRF models. However, there are a number of papers that use MRF models for image segmentation and a few papers that use different bee behavior algorithms. Below we give a brief overview on how a MRF model is used for the color image segmentation.

### 2.5.1 COLOR IMAGE SEGMENTATION AND PARAMETER ESTIMATION IN A MARKOVIAN FRAMEWORK

In (Kato, 2001), a Bayesian classification algorithm is provided for color image segmentation, where multivariate Gaussian distribution is used to represent pixel classes and the first order Markov Random Field is used as a priori model. The process is

presented in three sections below; the model, parameter estimation, and obtaining the initial parameters. Let us review each section step by step.

### 2.5.1.1 THE MODEL

The first decision made in this study is to use  $L^*u^*v^*$  color space as the color representation schema for a color image. Each pixel  $s \in \mathcal{S}$  in the image  $\mathcal{F} = \{f_s | s \in \mathcal{S}\}$  consists of three components  $L^*u^*v^*$ , represented by vector  $f_s$ . They look for labeling  $\hat{\omega}$  that maximizes the posteriori probability  $P(\omega | \mathcal{F})$ , which is a MAP or a maximum a posteriori estimation represented by  $\max_{\omega \in \Omega} \prod_{s \in \mathcal{S}} P(f_s | \omega_s) P(\omega)$ . Here  $\Omega$  represents the set of all possible labels. To be able to segment the image into homogenous regions, the pixel class  $\lambda$  has to be one of the color patches of the input image. Therefore,  $P(f_s | \omega_s)$  follows Gaussian or normal distribution and pixel classes,  $\lambda \in \Lambda = \{1, 2, \dots, L\}$ , are represented by the mean vectors  $\mu_\lambda$  and covariance matrices  $\Sigma_\lambda$ . In addition  $P(\omega)$  is a first order MRF. Per Hammersley-Clifford theorem it follows Gibbs distribution. So,

$$P(\omega) = \frac{\exp(-U(\omega))}{Z(\beta)} = \frac{\prod_{C \in \mathcal{C}} \exp(-V_C(\omega_C))}{Z(\beta)} \quad 2.13$$

$U(\omega)$  is the energy function,  $Z(\beta) = \sum_{\omega \in \Omega} \exp(-U(\omega))$  is the normalizing constant,  $V_C$  is the clique potential of clique  $C \in \mathcal{C}$  having label  $\omega_C$ , and  $\mathcal{C}$  is the set of doubletons.

Important to notice that singletons impact the probability of the label directly without taking into the consideration of the neighboring pixels. However, doubletons account for the neighboring pixels. The above model favors similar classes in neighboring pixels and therefore the energy function of this MRF image segmentation model is:

$$\begin{aligned}
U(\omega, \mathcal{F}) = & \sum_{s \in \mathcal{S}} \left( \ln \left( \sqrt{(2\pi)^3 |\Sigma_{\omega_s}|} \right) \right. \\
& + \frac{1}{2} (\mathbf{f}_s - \mu_{\omega_s}) \Sigma_{\omega_s}^{-1} (\mathbf{f}_s - \mu_{\omega_s})^T \\
& \left. + \beta \sum_{\{s,r\} \in \mathcal{C}} \delta(\omega_s, \omega_r), \right)
\end{aligned} \tag{2.14}$$

where,  $\delta(\omega_s, \omega_r)$  is 1 when  $\omega_s$  and  $\omega_r$  are different and 0 when they are the same.  $\beta > 0$  controls the homogeneity of the regions. Higher  $\beta$  is, more homogenous regions we have (Kato, 2001).

### 2.5.1.2 PARAMETER ESTIMATION

The parameter values, namely mean vector  $\mu_\lambda$ , covariance matrix  $\Sigma_\lambda$ , and  $\beta$ , are denoted by  $\Theta$ , and the labeled data set are not known for the above model. Therefore, an adaptive estimation method is proposed for parameter estimation and image segmentation simultaneously. The estimation is represented as a MAP below:

$$(\hat{\omega}, \hat{\Theta}) = \arg \max_{\omega, \Theta} P(\omega, \mathcal{F} \mid \Theta) \tag{2.15}$$

This can be estimated using  $\hat{\omega} = \arg \max_{\omega} P(\omega, \mathcal{F} \mid \hat{\Theta})$  and  $\hat{\Theta} = \arg \max_{\Theta} P(\hat{\omega}, \mathcal{F} \mid \Theta)$  where the first one is a MAP estimation for the labels and the second is a ML estimation for the parameters based on label estimations. To solve the above equations, the following iterative algorithm is provided.

**Algorithm 12:** Adaptive segmentation

1. Set  $k = 0$  and initialize  $\hat{\Theta}^0$

2. Maximize  $P(\omega, \mathcal{F} \mid \hat{\Theta}^k)$  using an optimization algorithm, the resulting labeling is denoted by  $\hat{\omega}^{k+1}$
3. Update the current estimate of parameters  $\hat{\Theta}^{k+1}$  to the ML estimate based on the current labeling  $\hat{\omega}^{k+1}$
4. Go to step 2 with  $k = k + 1$  until  $\hat{\Theta}^k$  stabilizes

To further explain step 3, the probability at the right-hand side of the ML estimation equation (2.15) can be written as  $P(\hat{\omega}, \mathcal{F} \mid \Theta) = P(\mathcal{F} \mid \hat{\omega}, \Theta) P(\hat{\omega} \mid \Theta)$ . The first term is a Gaussian density and second is a first order MRF. Now consider the log-likelihood function as below

$$\begin{aligned}
 \mathcal{L}(\Theta) &= -\ln(P(\hat{\omega}, \mathcal{F} \mid \Theta)) \\
 &= \sum_{\lambda \in \Lambda} \sum_{s \in \mathcal{S}_\lambda} \left( \ln \left( \sqrt{(2\pi)^3 |\Sigma_\lambda|} \right) \right. \\
 &\quad \left. + \frac{1}{2} (\mathbf{f}_s - \mu_\lambda) \Sigma_\lambda^{-1} (\mathbf{f}_s - \mu_\lambda)^T \right) \\
 &\quad - \beta \sum_{\{s,r\} \in \mathcal{C}} \delta(\hat{\omega}_s, \hat{\omega}_r) - \ln(Z(\beta))
 \end{aligned} \tag{2.16}$$

here  $\mathcal{S}_\lambda$  is the pixel set where  $\hat{\omega} = \lambda$ . The ML function is minimized when the derivative  $\partial \mathcal{L}(\Theta) / \partial \Theta$  is zero at  $\hat{\Theta}$ . The solution is the mean and covariance with respect to  $\mu_\lambda(i)$  and  $\Sigma_\lambda(i, j)$  respectively:

$$\begin{aligned}
 \mu_\lambda(i) &= \frac{1}{|\mathcal{S}_\lambda|} \sum_{s \in \mathcal{S}_\lambda} \mathbf{f}_s(i), \quad \Sigma_\lambda(i, j) \\
 &= \frac{1}{|\mathcal{S}_\lambda|} \sum_{s \in \mathcal{S}_\lambda} (\mathbf{f}_s(i) - \mu_\lambda(i)) \cdot (\mathbf{f}_s(j) \\
 &\quad - \mu_\lambda(j))
 \end{aligned} \tag{2.17}$$

However, with respect to  $\beta$  the solution is:

$$N^{\text{ih}}(\hat{\omega}) = \frac{\sum_{\omega \in \Omega} N^{\text{ih}}(\omega) \exp(-\beta N^{\text{ih}}(\omega))}{\sum_{\omega \in \Omega} \exp(-\beta N^{\text{ih}}(\omega))} \quad 2.18$$

where  $N^{\text{ih}}(\hat{\omega}) = \sum_{\{s,r\} \in \mathcal{C}} \delta(\hat{\omega}_s, \hat{\omega}_r)$  represents the number of inhomogeneous cliques in  $\hat{\omega}$ .

This is all estimated through the iterative algorithms (Kato, 2001).

### 2.5.1.3 OBTAINING INITIAL PARAMETERS

Initial values for the parameters are required in order to run the estimation process explained above. It has been empirically proven that an initialization of the mean value plays an important role on how good the final image segmentation will be, but the other parameters are not as important. Getting a good initial value for the mean is proven to be a difficult problem for images that either have sparse histogram or do not have clear peaks. In this study, a new approach based on pre-segmentation is proposed to solve this problem. A split-and-merge algorithm is used to obtain an initial segmentation where color differences are used as a homogenous measure. The Euclidean distance of two color vectors in the  $L^*u^*v^*$  color space is used to obtain the color difference. The average of original colors is used to represent a region. Two neighboring regions with color difference smaller than threshold  $\tau$  are merged together where  $\tau$  is calculated from the original image. It is set to 25% of the maximum color difference. Once the image is pre-segmented, then its histogram has clear peaks and it can be used for image analysis. Considering that each pixel is replaced by the color average of its neighborhood, the histogram's peaks are around the mean values. Since pre-segmentation can produce slightly different color average, it can produce more than one peak per pixel class and

therefore false mean values. To get rid of the extra peaks, neighboring histogram peaks are merged together through image quantization. It is shown that 20% color reduction produces good results to remove extra histogram peaks. Finally, mean vectors are constructed from top L peaks. This approach of image pre-segmentation is proven to produce better mean initial values than image pre-segmentation based on K-means algorithm (Kato, 2001).

## 2.5.2 COLOR IMAGE SEGMENTATION BASED ON MULTIOBJECTIVE ARTIFICIAL BEE COLONY OPTIMIZATION

In the literature an image segmentation method based on the improved bee colony algorithm for multi-objective optimization (IBMO) is presented. The method proposed consists of three steps, extraction of features from color image, use of IBMO to find optimal center points, known as seeds, and optimal similarity values, and applying the process of region growing. Here the segmentation is performed through seeded region growing (SRG) process, which groups homogenous pixels into groups or regions. The SRG process begins at seeds and examines adjacent pixels incorporating homogenous pixels in the region. Each pixel is assigned a label for the region it belongs to. Then, the seeds are replaced with the new center pixel in the region which includes the newly added pixels. The process is repeated until all pixels are assigned to a region.

The SRG process is explained in four steps; first, a vector with distances between seeds and pixels in the image is found. Second, the probability of pixel belonging to seeds is calculated. Third, a binary matrix is created with zero values when the threshold

is greater than probability and one otherwise. Finally, a labeling method is applied to the connected component in the binary matrix for the related seed. The process is repeated for all seeds and unlabeled pixels. Important to note that the IBMO provides three improvements to the SRG process. First, it finds optimal positions for seeds. Second, it determines the threshold values for seeds to be used as homogenous criteria. Third, it provides the ability to evaluate segmentation quality using multiple criteria for the multi-objective optimization.

A post processing step is applied to the pre-segmented image. In this stage, first, regions that are smaller than a pre-set threshold for region size are assigned to the nearest region. Then, the SRG process is repeated until all unlabeled images are assigned to a region (Sag, 2015).

### III THE PROPOSED SOLUTION

In this chapter we present the work developed in this thesis. It is based on the work presented in (Kato, 2001), which we introduce and utilize the Bee algorithm as an optimization mechanism to perform image segmentation in an iterative process. Below we explain the segmentation model, the parameter estimation and segmentation, parameter initialization, and bee algorithm parameter settings.

#### 3.1 THE MODEL

As it is pointed out above, this thesis is based on the work in (Kato, 2001). Similarly to (Kato, 2001), we are interested in grouping similar color pixels into segments. Each pixel in a given image consists of three color components using  $L^*u^*v^*$  color schema which is represented as a vector. We need to estimate labeling that maximize posteriori probability. Pixels in the segments are replaced by one of the given labels from the homogenous sections in the original image. Here, the probability of pixel colors given a labeling follows the Gaussian distributions. Also, mean vectors and covariance matrices are used to represent pixel classes for each patch. In addition, the probability of a labeling is a first order MRF and per Hammersley-Clifford theorem (Kato, 2001), it follows Gibbs distribution. As we have seen in the survey of (Kato, 2001), Gibbs distribution depends on normalizing constant and energy function which in turn depends on clique potential of cliques for the given pixels. Considering this is a first order MRF representation, cliques are either singletons or doubletons. This means that in this model the posteriori probability for pixel labels is impacted by the neighboring pixel



labels. This is different from the model used in (Kato, 2001) in which we apply the Bee Algorithm instead of the algorithms used in (Kato, 2001) such as Metropolis and ICM.

### 3.2 PARAMETER ESTIMATION AND SEGMENTATION

In order to use the above model in image segmentation, a number of parameters, namely mean and covariance for each class, and the normalizing constants have to be estimated. To estimate these values, we use the following adaptive iteration process.

1. Initialize parameters.
2. Use Bee algorithm for the optimization of the probability of labeling given the parameters.
3. Update the parameter estimations according to the labeling above.
4. Repeat steps 2 and 3 until a change is small.

This adaptive process is the same as in (Kato, 2001) with exception that we use Bee algorithm for optimizing the labels and parameters instead of other algorithms such as ICM and Metropolis. It is also important to state that this iterative process at each iteration produces a new segmented image and calculates new parameter estimations.

### 3.3 PARAMETER INITIATION

In the parameter estimation above, we can see that the first step in the iterative process is to initialize the parameters. In (Kato, 2001), they talk about a process of initializing parameters; however, in their implementation the user selects a region of the

image per class out of which the mean and other parameters are calculated. Our implementation is built on top of their implementation and the section of parameter initialization is not modified. The fact that the user selects the regions to represent each class making this process a supervised method; otherwise, everything else in the process is unsupervised.

### 3.4 PARAMETERS IN THE BEE ALGORITHM

As explained above we use Bee algorithm optimization to implement the above explained model for image segmentation in an iterative process. In literature survey we explained that Bee algorithm has several parameters that are usually set by the user. However, less parameters the user has to provide better it is. Therefore, the parameters in our process are programmatically set as listed in the table below. Experiments showed that these are good values for each one of the given parameters.

**Table 3:** Bee algorithm parameter settings.

Symbol	Meaning of parameters	Values of parameters
n	Number of scouts	2% of number of pixels
m	Number of sites selected out of n visited	All whose fitness is positive
e	Number of best sites	20% of scouts
nep	Number of bees recruited for best e sites	25
m-e	Number of bees recruited for other sites	1
nsp	Selected sites	Same as e
ngh	Neighborhood size	25

## IV EXPERIMENTS AND RESULTS

In this chapter we show the experimental results tested on several images. We also evaluate the segmentation accuracy and time complexity of the algorithms tested.

### 4.1 EXPERIMENTS

We use a number of different images to run our experiments. Each image is tested using the four existing implementations ICM, HHD, Metropolis, and Gibbs Sampler in (Kato, 2001) as well as our implementation using Bee Algorithm. Considering that each one of the implementations is based on the pre-segmentation, we show the original image, pre-segmented image, and final-segmented image in the order in Figures 16, 17, 18, 19, 20, 21, 22, and 23.



(a)



(b)



(c)



(d)



(e)

**Figure 16:** Flower image segmented. (a) Using Bee algorithm, number of iterations 3, CPU time 38 ms; (b) using ICM algorithm, number of iterations 4, CPU time 104 ms; (c) using MMD algorithm, number of iterations 199, CPU time 649 ms; (d) using Metropolis algorithm, number of iterations 147, CPU time 629ms; (e) using Gibbs Sampler, number of iterations 143, CPU time 2174 ms.



(a)



(b)



(c)



(d)



(e)

**Figure 17:** Mountain image segmented. (a) Using Bee Algorithm, number of iterations 2, CPU time 25 ms; (b) using ICM algorithm, number of iterations 3, CPU time 54 ms; (c)

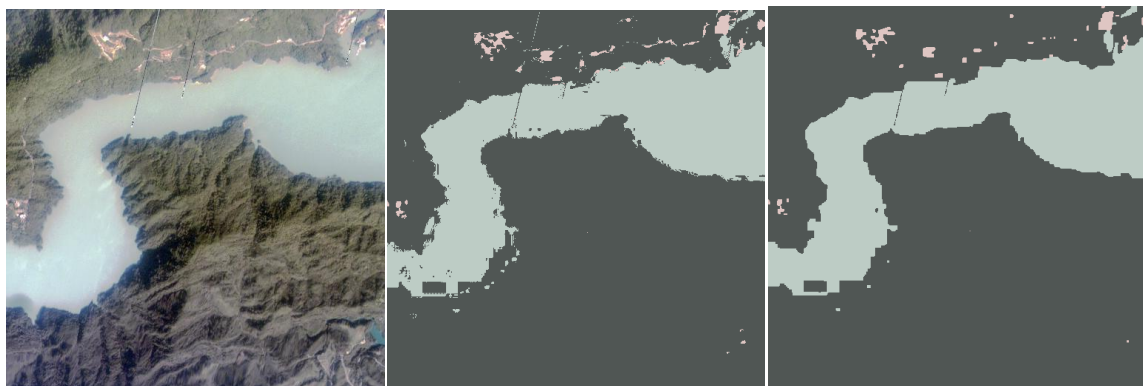
using MMD algorithm, number of iterations 148, CPU time 450 ms; (d) using Metropolis algorithm, number of iterations 97, CPU time 358 ms; (e) using Gibbs Sampler, number of iterations 35, CPU time 363 ms.



(a)



(b)



(c)

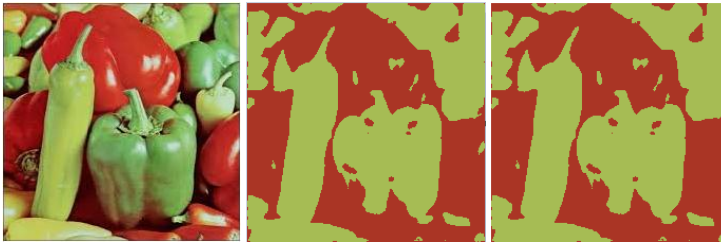


(d)

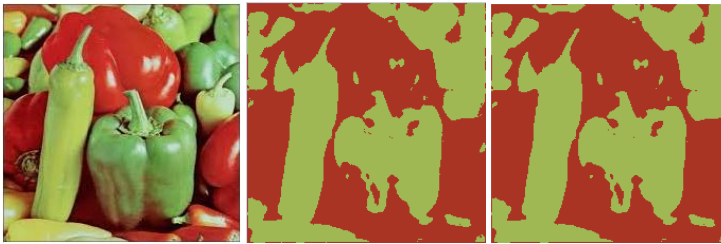


(e)

**Figure 18:** Satellite image segmented. (a) Using Bee algorithm, number of iterations 5, CPU time 367 ms; (b) using ICM algorithm, number of iterations 5, CPU time 620 ms; (c) using MMD algorithm, number of iterations 281, CPU time 10948 ms; (d) using Metropolis algorithm, number of iterations 260, CPU time 12835 ms; (e) using Gibbs Sampler, number of iterations 196, CPU time 16957 ms.



(a)



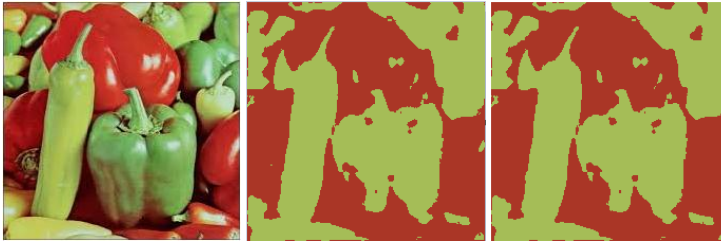
(b)



(c)



(d)

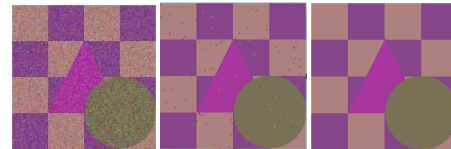


(d)

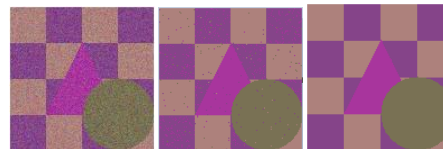
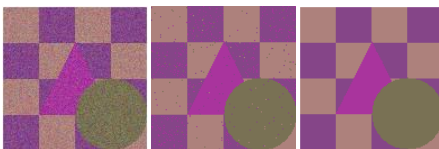
**Figure 19:** Peppers image segmented. (a) Using Bee algorithm, number of iterations 2, CPU time 37 ms; (b) using ICM algorithm, number of iterations 3, CPU time 72 ms; (c) using MMD algorithm, number of iterations 269, CPU time 1661 ms; (d) using Metropolis algorithm, number of iterations 172, CPU time 1841 ms; (e) using Gibbs Sampler, number of iterations 109, CPU time 1373 ms.



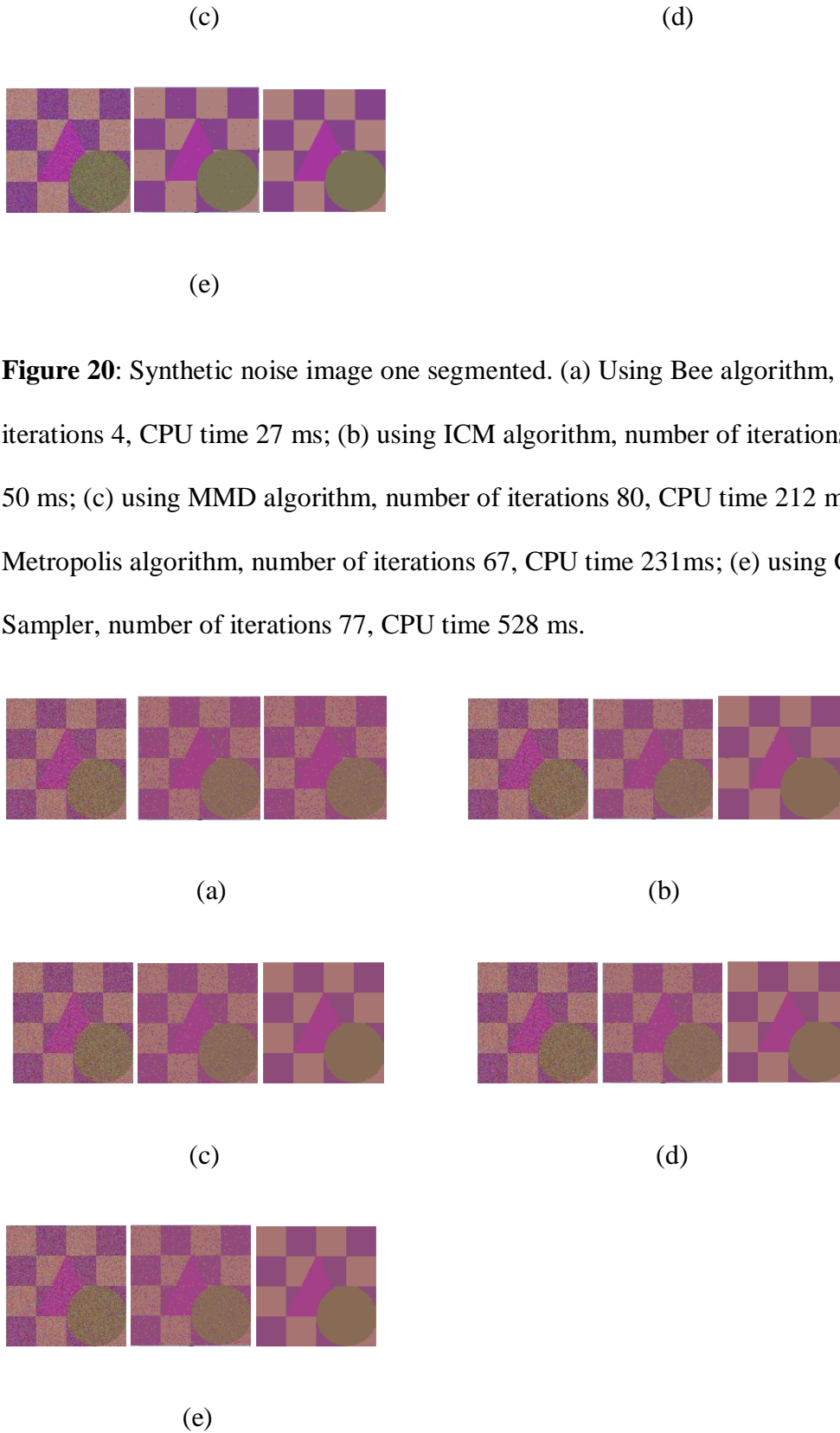
(a)



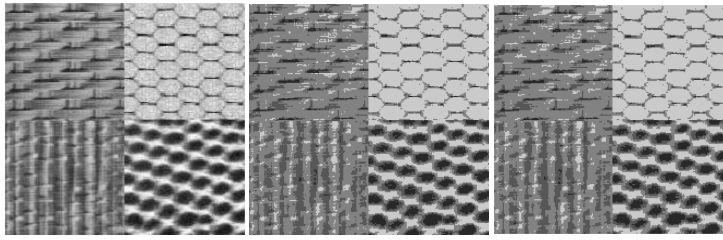
(b)



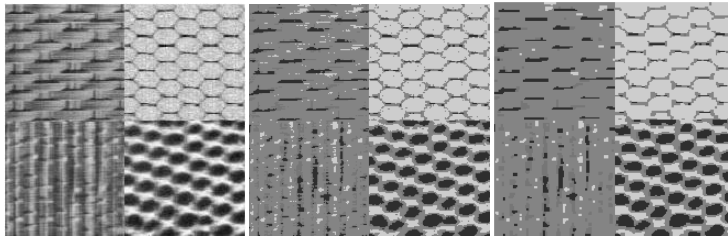




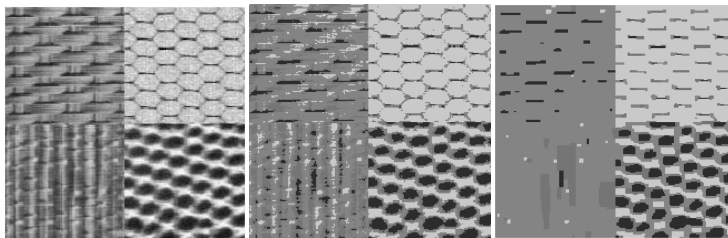
**Figure 21:** Synthetic noise image two segmented. (a) Using Bee algorithm, number of iterations 5, CPU time 30 ms; (b) using ICM algorithm, number of iterations 5, CPU time 74 ms; (c) using MMD, number of iterations 150, CPU time 430 ms; (d) using Metropolis algorithm, number of iterations 154, CPU time 503 ms; (d) using Gibbs Sampler, number of iterations 124, CPU time 836 ms.



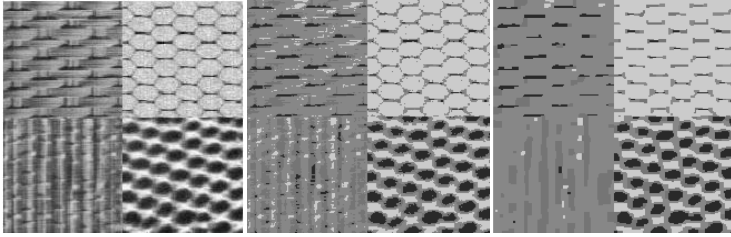
(a)



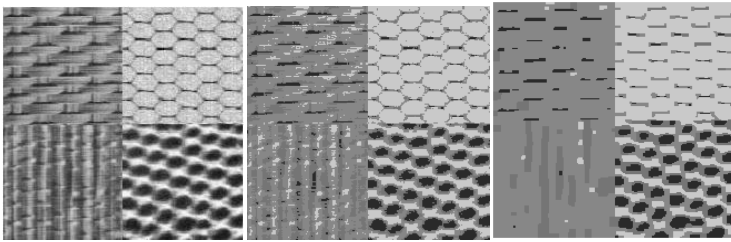
(b)



(c)

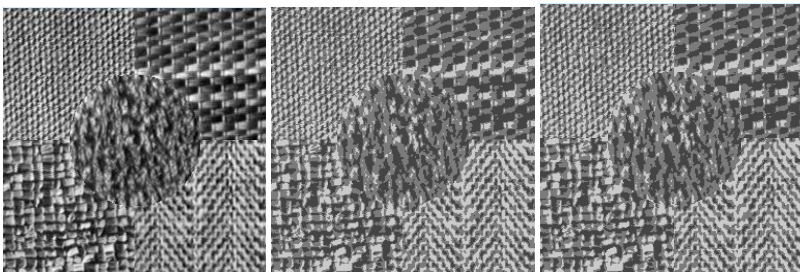


(d)

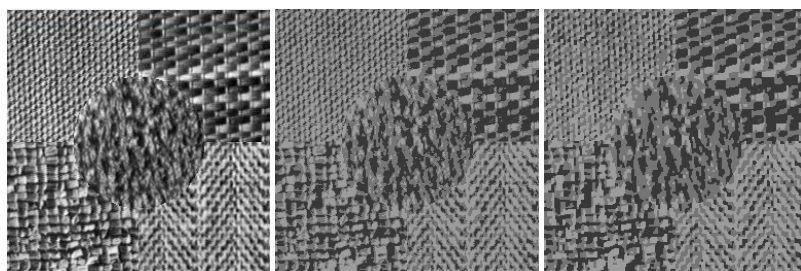


(e)

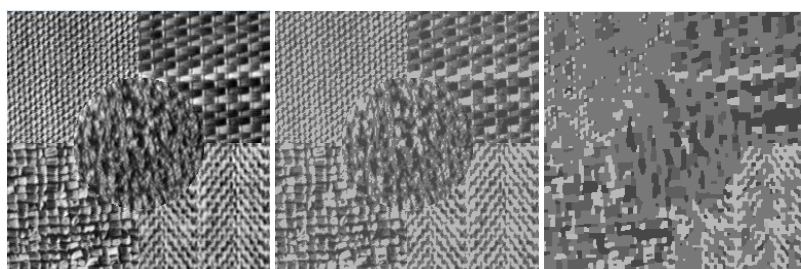
**Figure 22:** Texture image one segmented. (a) Using Bee algorithm, number of iterations 6, CPU time 90 ms; (b) using ICM algorithm, number of iterations 8, CPU time 227 ms; (c) using MMD algorithm, number of iterations 268, CPU time 1881 ms; (d) using Metropolis algorithm, number of iterations 243, CPU time 2246 ms; (e) using Gibbs Sampler, number of iterations 186, CPU time 3550 ms.



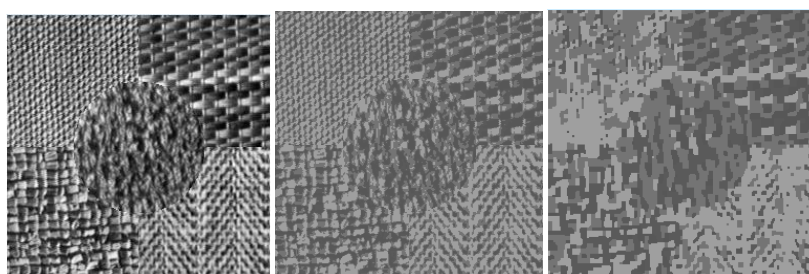
(a)



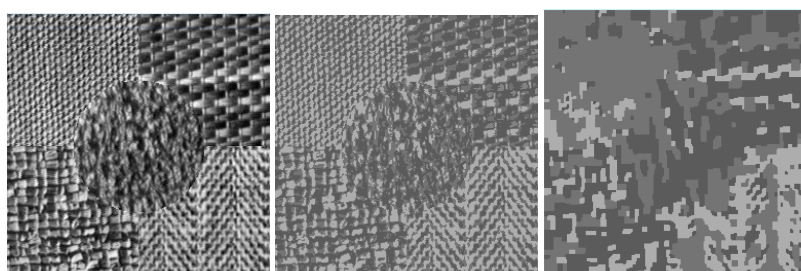
(b)



(c)



(d)



(e)

**Figure 23:** Texture image two segmented. (a) Using Bee algorithm, number of iterations 6, CPU time 125 ms; (b) using ICM algorithm, number of iterations 10, CPU time 507 ms; (c) using MMD algorithm, number of iterations 274, CPU time 2861 ms; (d) using Metropolis algorithm, number of iterations 278, CPU time 3733 ms; (e) using Gibbs Sampler, number of iterations 161, CPU time 5159 ms.

## 4.2 TIME COMPARISON

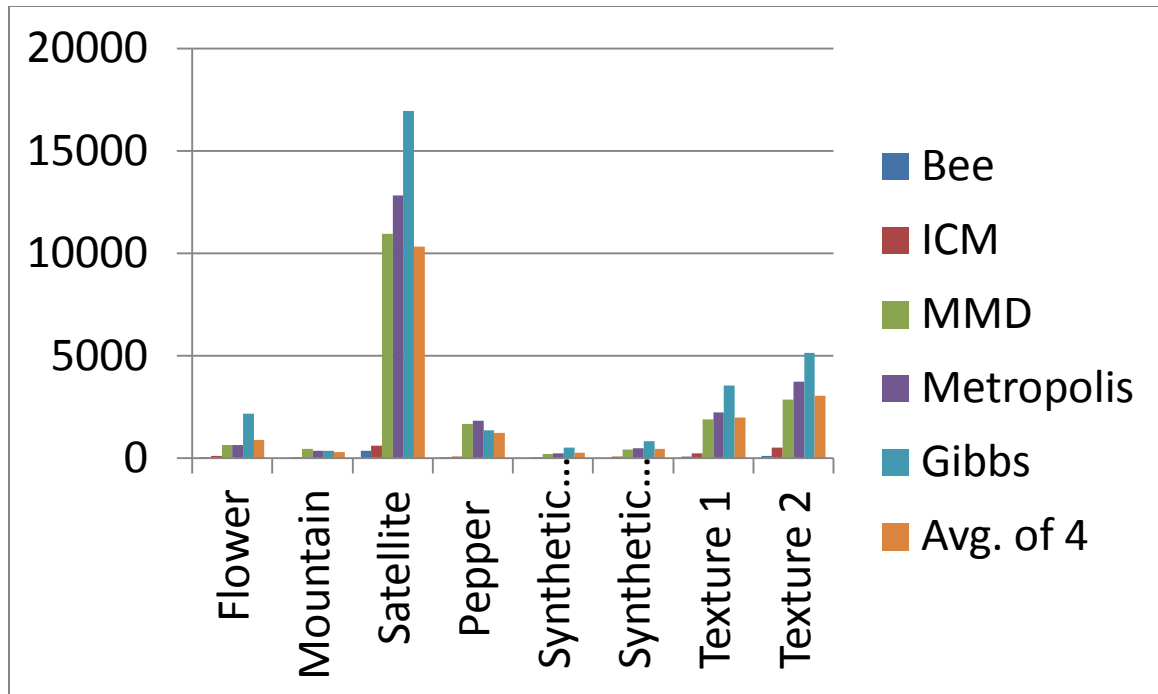
In the table below, we present the time in milliseconds that each algorithm took to process each image. Then, we take the average of running time for ICM, MMD, Metropolis and Gibbs. The ICM algorithm is the closest to Bee algorithm in terms of running time.

**Table 4:** Time comparison table.

	Bee	ICM	MMD	Metropolis	Gibbs	Avg. of 4	Bee/ICM ratio	Bee/Avg. ratio
Flower	38ms	104ms	649ms	629ms	2174ms	889ms	36.5%	4.3%
Mountain	25ms	54ms	450ms	358ms	363ms	306ms	46.3%	8.2%
Satellite	367ms	620ms	10948ms	12835ms	16957ms	10340ms	59.2%	3.5%
Pepper	37ms	72ms	1661ms	1841ms	1373ms	1237ms	51.4%	3.0%
Synthetic image 1	27ms	50ms	212ms	231ms	528ms	255ms	54%	10.6%
Synthetic image 2	30ms	74ms	430ms	503ms	836ms	461ms	40.5%	6.5%
Texture 1	90ms	227ms	1881ms	2246ms	3550ms	1976ms	40%	4.6%
Texture	125ms	507ms	2861ms	3733ms	5159ms	3065ms	24.7%	4.1%

2	s	s						
---	---	---	--	--	--	--	--	--

Time comparison can also be represented graphically as in Figure 24.



**Figure 24:** Time comparison for each algorithm.

From the table above we can see that Bee algorithm is much faster than all other algorithms. Comparing to the ICM algorithm which is the closest in time and accuracy to Bee algorithm, Bee algorithm only took between 25% and 60% of the time of ICM algorithm. However, if we compare with the average of all other algorithms ICM, MMD, Metropolis, and Gibbs, Bee algorithm only took about 11% of their average time.

#### 4.3 ACCURACY COMPARISON

By visually inspection on the results generated by algorithms compared, we can see that Bee algorithm, performs about the same as all other algorithms, except for the images with synthetic noise. When it comes to noisy images, we can see clearly that Bee algorithm does not perform as well as the others. In some other cases like the texture images, it might even look better. However, in this section we want to compare the segmented images against the ground truth data. We only have the ground truth data for the Satellite image, so we are going to compare the segmented images by each algorithm to the ground truth. Tables 5, 6, 7, 8, and 9 are the list of Error Matrix, and KHAT statistics for each algorithm. The formula for KHAT statistics is listed below.

$$k = \frac{N \sum_{i=1}^r X_{ii} - \sum_{i=1}^r (X_{i+} * X_{+i})}{N^2 - \sum_{i=1}^r (X_{i+} * X_{+i})} \quad 4.1$$

where N is the number of pixels; r is the number of labels,  $X_{ii}$  is the number of pixels correctly classified;  $X_{i+}$  and  $X_{+i}$  are the numbers of misclassified pixels.

**Table 5:** Error Matrix for Satellite image using Bee Algorithm.

Classification data	Mountain	River	Village	Row Total
Mountain	186474	8916	1863	197253
River	839	61708	128	62675
Village	804	21	1391	2216
Column total	188117	70645	3382	262144

Producer's Accuracy	User's Accuracy
Mountain = $186474/188117 = 99\%$	Mountain = $186474/197253 = 95\%$
River = $61708/70645 = 87\%$	River = $61708/62675 = 98\%$
Village = $1391/3382 = 41\%$	Village = $1391/2216 = 63\%$

Overall accuracy =  $(186474 + 61708 + 1391)/262144 = 95\%$

$$k = \frac{262144 * (186474 + 61708 + 1391) - ((188117 * 197253) + (70645 * 62675) + (3382 * 2216))}{262144^2 - ((188117 * 197253) + (70645 * 62675) + (3382 * 2216))} = 0.8787$$

So KHAT statistics for the satellite image segmented using Bee algorithm is about .88 or about 7% lower than overall accuracy.

**Table 6:** Error Matrix for Satellite image using ICM Algorithm.

Classification data	Mountain	River	Village	Row Total
Mountain	186832	8969	2208	198009
River	596	61631	36	62263
Village	689	45	1138	1872
Column total	188117	70645	3382	262144

Producer's Accuracy	User's Accuracy
Mountain = $186832/188117 = 99\%$	Mountain = $186832/198009 = 94\%$
River = $61631/70645 = 87\%$	River = $61631/62263 = 99\%$
Village = $1138/3382 = 34\%$	Village = $1138/1872 = 61\%$
Overall accuracy = $(186832 + 61631 + 1138)/262144 = 95\%$	

$$k = \frac{262144 * (186832 + 61631 + 1138) - ((188117 * 198009) + (70645 * 62263) + (3382 * 1872))}{262144^2 - ((188117 * 198009) + (70645 * 62263) + (3382 * 1872))} = 0.8785$$

So KHAT statistics for the satellite image segmented using ICM algorithm is about .88 or about 7% lower than overall accuracy.

**Table 7:** Error Matrix for Satellite image using MMD Algorithm.

Classification data	Mountain	River	Village	Row Total
Mountain	186918	9123	2237	198278
River	460	61480	13	61953
Village	739	42	1132	1913
Column total	188117	70645	3382	262144

Producer's Accuracy	User's Accuracy
Mountain = $186918/188117 = 99\%$	Mountain = $186918/198278 = 94\%$
River = $61480/70645 = 87\%$	River = $61480/61953 = 99\%$



Village = $1132/3382 = 33\%$	Village = $1132/1913 = 59\%$
Overall accuracy = $(186918 + 61480 + 1913)/262144 = 95\%$	

$$k = \frac{262144 * (186918 + 61480 + 1132) - ((188117 * 198278) + (70645 * 61953) + (3382 * 1913))}{262144^2 - ((188117 * 198278) + (70645 * 61953) + (3382 * 1913))} = 0.8776$$

So KHAT statistics for the satellite image segmented using MMD algorithm is about .88 or about 7% lower than overall accuracy.

**Table 8:** Error Matrix for Satellite image using Metropolis Algorithm.

Classification data	Mountain	River	Village	Row Total
Mountain	187115	9040	2349	198504
River	478	61598	30	62106
Village	524	7	1003	1534
Column total	188117	70645	3382	262144

Producer's Accuracy	User's Accuracy
Mountain = $187115/188117 = 99\%$	Mountain = $187115/198504 = 94\%$
River = $61598/70645 = 87\%$	River = $61598/62106 = 99\%$
Village = $1003/3382 = 30\%$	Village = $1003/1534 = 65\%$

Overall accuracy =  $(187115 + 61598 + 1003)/262144 = 95\%$

$$k = \frac{262144 * (187115 + 61598 + 1003) - ((188117 * 198504) + (70645 * 62106) + (3382 * 1534))}{262144^2 - ((188117 * 198504) + (70645 * 62106) + (3382 * 1534))} = 0.8792$$

So KHAT statistics for the satellite image segmented using Metropolis algorithm is about .88 or about 7% lower than overall accuracy.

**Table 9:** Error Matrix for Satellite image using Gibbs Sampler.

Classification data	Mountain	River	Village	Row Total
Mountain	187068	8974	2239	198281
River	475	61656	16	62147
Village	574	15	1127	1716
Column total	188117	70645	3382	262144

Producer's Accuracy	User's Accuracy
Mountain = $187068/188117 = 99\%$	Mountain = $187068/198281 = 94\%$
River = $61656/70645 = 87\%$	River = $61656/62147 = 99\%$
Village = $1127/3382 = 33\%$	Village = $1127/1716 = 66\%$

Overall accuracy =  $(187068 + 61656 + 1127)/262144 = 95\%$

$$k = \frac{262144 * (187068 + 61656 + 1127) - ((188117 * 198281) + (70645 * 62147) + (3382 * 1716))}{262144^2 - ((188117 * 198281) + (70645 * 62147) + (3382 * 1716))} = 0.8807$$

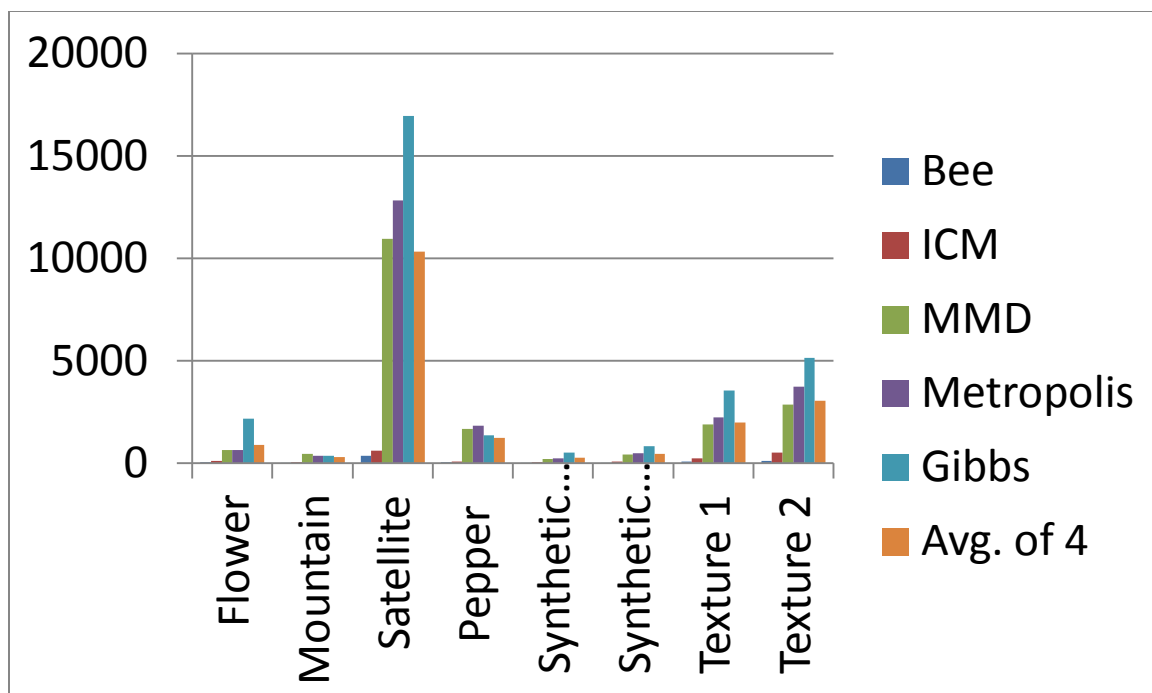
So KHAT statistics for the satellite image segmented using Gibbs Sampler algorithm is about .88 or about 7% lower than overall accuracy.

As we see in the tables above, all algorithms including our bee implementation have about the same accuracy and KHAT statistical values, with some small exception when it comes to any of the given individual classes. However, the overall accuracy and KHAT statistics value are about the same for all algorithms. A comparison among all algorithms tested is listed in Table 10.

**Table 10:** Accuracy and KHAT comparison for all algorithms.

	Bee	ICM	MMD	Metropolis	Gibbs
P.A. Mountain	99%	99%	99%	99%	99%
P.A. River	87%	87%	87%	87%	87%
P.A. Village	41%	34%	33%	30%	33%
U.A. Mountain	95%	94%	94%	94%	94%
U.A. River	98%	99%	99%	99%	99%
U.A. Village	63%	61%	59%	65%	66%
Overall Accuracy	95%	95%	95%	95%	95%
KHAT	.88	.88	.88	.88	.88

Accuracy and KHAT comparisons can also be represented graphically as in Figure 25.



**Figure 25:** Accuracy and KHAT comparisons.

## V CONCLUSION

Computer vision is widely used across many disciplines making our daily lives easier in ways that we either take for granted or do not even fully understand the importance of, and image segmentation is a vital step in making that possible. In the literature survey, we showed some methods of performing image segmentation. We explained Markov Random Field and a few ways of implementing MRF models in image segmentation. We also talked about swarm intelligence and explained a number of swarm based algorithms and provided an example of a swarm based image segmentation method.

In this thesis we proposed a new algorithm based on the bee algorithm to perform the optimization of image segmentation in a Markov Random Field model. We showed through our experiments and results that its accuracy is just as good as the other algorithms, if not better in some cases, with exception of noisy images. However, our proposed algorithm performed much faster than existing solutions.

Considering that image segmentation is a complex problem to get exactly correct, it remains as an active area of research. One discipline that is gaining popularity and seems to have a lot of potential to solve different problems, including image segmentation, is deep learning. It would make a very interesting study on the image segmentation method to apply deep learning techniques and MRF models.

## VI APPENDIX

To better understand image segmentation, it is important to understand different probability distributions and a few other probability and statistical concepts. Therefore, let us do a brief review.

### A.1 MAXIMUM LIKELIHOOD AND LOG LIKELIHOOD

We represent a probability density function that depends on some parameters by  $p(x|\Omega)$ , where  $\Omega$  represents a vector of various parameters in  $p(x)$ . For now, let us say that  $p(x)$  depends only on one parameter,  $\omega$ , also let us say that we observe one value for variable  $x$ ,  $x = x_1$ . If we decide to select  $\omega$  such that it maximizes the probability of observing  $x_1$ , it is called maximum likelihood. However, if we have  $n$  distinct observations for  $x$ , we can calculate the likelihood  $p(\text{Data}|\Omega)$  as shown in equation A.1.

$$p(\text{Data}|\Omega) \equiv \prod_{k=1}^N p(x_k|\Omega) \tag{A.1}$$

Here, we select an  $\Omega$  to maximize  $p(\text{Data}|\Omega)$ . However, in sometimes in practice  $p(x_k|\Omega)$  causes underflow, so we compute the log-likelihood as in equation A.2.

$$l(\text{Data}|\Omega) \equiv \sum_{k=1}^N \ln p(x_k|\Omega) \tag{A.2}$$

Where  $N$  is the number of observations. Finally, we maximize the log-likelihood instead (Petrou, 2006).

## A.2 PRIOR AND POSTERIOR PROBABILITY DISTRIBUTIONS

In Bayesian statistics prior probability distribution (prior), refers to the probability distribution before taking into account any evidence. However, posterior probability distribution is the conditional probability distribution after taking into account the evidence (Prior probability 2016, Posterior probability 2016).

## A.3 LINEAR COMBINATION

Linear combination is the sum of multiplications of terms by constants. For example given terms  $x$  and  $y$  and constants  $a$  and  $b$ , the linear combination would be  $ax + by$ . Here is a more precise definition from Wikipedia:

*Suppose that  $K$  is a field (for example, the real numbers) and  $V$  is a vector space over  $K$ . As usual, we call elements of  $V$  vectors and call elements of  $K$  scalars. If  $v_1, \dots, v_n$  are vectors and  $a_1, \dots, a_n$  are scalars, then the linear combination of those vectors with those scalars as coefficients is*

$$a_1 v_1 + a_2 v_2 + a_3 v_3 + \dots + a_n v_n$$

(Linear combination, 2015).

#### A.4 DISCRETE AND CONTINUES VARIABLES

Discrete variables are variables that can only take discrete or distinct values, such as 0 or 1. However, they cannot take all values between 0 and 1. On the other hand, continuous variables are variables that if they can take two distinct values say a and b, they can also take all the values between a and b. Their value set, from a to b, is uncountable, it is ongoing. Therefore, they are called continues (Continues, 2015).

#### A.5 STANDARD DEVIATION AND VARIANCE

In probability and statistics, standard deviation is used to measure the variation of a given set of data. In other words, given the set of data, standard deviation tells us how close to each other or how apart from each other they are. When standard deviation is small the data are closer to each other, closer to the mean; greater the standard deviation more variant data we have. If a discrete random variable has random values such as  $x_1, x_2, \dots, x_n$  with equal probability, then standard deviation can be calculated as shown in equation A.3.

$$\sigma = \sqrt{\frac{1}{N} [(x_1 - \mu)^2 + (x_2 - \mu)^2 + \dots + (x_N - \mu)^2]}, \text{ where } \mu = \frac{1}{N}(x_1 + \dots + x_N) \quad \text{A.3}$$

This can also be written using sigma notation as in equation A.4.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}, \text{ where } \mu = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{A.4}$$

Likewise for continues random variables standard deviation can be calculated as shown in equation A.5.

$$\sigma = \sqrt{\int_{\mathbf{x}} (x - \mu)^2 p(x) dx}, \text{ where } \mu = \int_{\mathbf{x}} x p(x) dx \quad \text{A.5}$$

Here, the limits of the integral are set from the lowest possible value of x to the highest possible value of x. It is also important to notice that in addition to measuring the variation of random variables, standard deviation can also be used to measure the confidence of given conclusions to statistical measures (Standard, 2016).

Likewise, variance is a measure of how different the given data are. It is equals to the standard deviation squared. For discrete random variables, if the random variables are not equally likely then standard deviation can be written as in equation A.6.

$$\text{Var}(X) = \sum_{i=1}^n p_i \cdot (x_i - \mu)^2 \quad \text{A.6}$$

$\mu$  is the mean or expected value and it is equals to :

$$\mu = \sum_{i=1}^n p_i \cdot x_i \quad \text{A.7}$$

Likewise for continues variables, standard deviation is calculated as follows:

$$\text{Var}(X) = \sigma^2 = \int (x - \mu)^2 f(x) dx = \int x^2 f(x) dx - \mu^2 \quad \text{A.8}$$

And



$$\mu = \int x f(x) dx \quad \text{A.9}$$

(Variance, 2016).

## A.6 BAYES' THEOREM AND NAIVE BAYES CLASSIFIER OR BAYESIAN CLASSIFIER

Bayes' theorem is used to find conditional probability of an event given certain conditions. Mathematically Bayes' theorem is expressed as shown in equation A.10.

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)} \quad \text{A.10}$$

$P(A)$  and  $P(B)$  are probabilities of two independent events A and B respectively.

$P(A|B)$  is the conditional probability of A given condition B is met.

$P(B|A)$  is the conditional probability of B given condition A is met.

And of course,  $P(B)$  cannot be zero (Bayes, 2016).

Naive Bayes classifiers, also known as Bayesian classifiers in computer science are probabilistic classifiers based on Bayes' theorem. Classifiers are methods used to assign classes to different problem instances. Each classifier can be trained using different algorithms and there is always a finite number of classes. Given  $n$  features, or independent variables, for a problem instance, represented by  $X = (x_1, x_2, \dots, x_n)$ , the Naive Bayes as a conditional probability model assign probabilities for each of the possible class outcomes as shown in equation A.11.

$$p(C_k|x_1, \dots, x_n) \quad \text{A.11}$$

However, if the number of features is large or if the number of the values each feature can have is high, assigning the probabilities becomes hard to achieve. Therefore, Bayes' theorem is used and it takes the following form:

$$p(C_k|\mathbf{x}) = \frac{p(C_k) p(\mathbf{x}|C_k)}{p(\mathbf{x})} \quad \text{A.12}$$

This can be expressed in more simple terms to understand as follows:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \quad \text{A.13}$$

Naive Bayes' classifier has wide range of uses including text classification and retrieval (Naive, 2016).

## A.7 BINOMIAL DISTRIBUTION

Binomial distribution is a discrete probability distribution. It shows the probability of outcome for n experiments. Each one of which has two possible outcomes, such as yes or no, with equal probability p. Binomial distribution is used to model cases when a small number of samples are experimented out of a much larger data set. The probability of getting true k times within n experiments for a random variable x, in binomial distribution, can be calculated using equation A.14.

$$f(k; n, p) = \Pr(X = k) = \binom{n}{k} p^k (1 - p)^{n-k} \quad \text{A.14}$$

for  $k = 0, 1, 2, \dots, n$  and

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad \text{A.15}$$

this is known as binomial coefficient, from which the name of binomial distribution comes. Wikipedia interprets the formula as: "we want exactly  $k$  successes ( $pk$ ) and  $n - k$  failures  $(1 - p)^{n-k}$ . However, the  $k$  successes can occur anywhere among the  $n$  trials, and there are  $\binom{n}{k}$  different ways of distributing  $k$  successes in a sequence of  $n$  trials" (Binomial, 2016).

## A.8 GAUSSIAN DISTRIBUTION AND MULTIVARIATE GAUSSIAN DISTRIBUTION

Gaussian distribution, also known as normal distribution, is a probability distribution for continuous random variables whose probability density function is the following:

$$f(x | \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{A.16}$$

Note that:

$\mu$  represents the mean

$\sigma$  represents the standard deviation

$\sigma^2$  represents variance

The shape of the Gaussian distribution is a bell shape, which means that most values are relatively close to the mean. Smaller the standard deviation, the values are close the mean. Because of its shape, Gaussian distribution is sometimes called "the bell curve" but that can be misleading because there are other distributions that have bell shapes. In probability theory, the Gaussian distribution is considered as a very important distribution of continuous variables because of its wide applicability. Gaussian or normal distribution is often used to represent values of random variables with unknown distributions, both in natural and social sciences (Normal, 2016).

In addition, multivariate Gaussian distribution or multivariate normal distribution is a generalization of the Gaussian distribution from one dimension to a higher number of dimensions. If every linear combination of k-components of a random vector has Gaussian distribution then the vector has a k-variant Gaussian distribution (Multivariate, 2016).

## A.9 GIBBS DISTRIBUTION AND GIBBS RANDOM FIELD

Gibbs distribution is defined as "a function that specifies, in terms of clique potentials, the joint probability density function of a particular combination X of values to exist over the whole grid." The general form of Gibbs distribution is as follows:

$$p(X) = \frac{1}{Z} e^{\sum_{all\ types\ of\ clique} \sum_{all\ cliques\ of\ this\ type} x_{i_1} x_{i_2} \dots x_{i_{c_k}} U_k(x_{i_1} x_{i_2} \dots x_{i_{c_k}})}$$
A.17

here  $c_k$  is the number of cells or pixels in a given clique of type  $k$ ,  $x_{ij}$  is the value of  $ij$ th pixel or cell in the given clique (see section about clique below), and  $U_k(x_{i1}x_{i2}...x_{ic_k})$  is the clique potential (see section about clique potential below) of type  $k$ . However,  $Z$  is the normalizing constant, also known as partition function which is used to make  $p(X)$  a probability density function. In other words the sum of all possible combination of  $X$  values is equals to 1 and the value for any given combination is between 0 and 1 inclusively (Petrrou, 2006). However, Gibbs random filed is a random field that follows Gibbs distribution (Li, 2009).

## VII BIBLIOGRAPHY

Ahmed, Hazem, & Glasgow, J. (2012). Swarm Intelligence: Concepts, Models and Applications (Research Report No. 858). Retrieved from <http://ftp.qucis.queensu.ca/TechReports/Reports/2012-585.pdf>

Akbari, R., & Mohammadi, A., & Ziarati, K. (2009). A Powerful Bee Swarm Optimization Algorithm. *IEEE*, 13.

Bayes' theorem (2016, June 29). Retrieved from [https://en.wikipedia.org/wiki/Bayes'\\_theorem](https://en.wikipedia.org/wiki/Bayes'_theorem)

Binomial distribution (2016, June 6). Retrieved from [https://en.wikipedia.org/wiki/Binomial\\_distribution](https://en.wikipedia.org/wiki/Binomial_distribution)

Blake, A. & Kohli, P. (2011). Introduction To Markov Random Fields. Retrieved from [https://mitpress.mit.edu/sites/default/files/titles/content/9780262015776\\_sch\\_0001.pdf](https://mitpress.mit.edu/sites/default/files/titles/content/9780262015776_sch_0001.pdf)

Bolaji, A. L., & Khader, A. T., & Al-Betar, M. A. (2013). Artificial Bee Colony Algorithm, Its Variants And Applications: A Survey. *Journal of Theoretical and Applied Technology*. Retrieved from <http://www.jatit.org/volumes/Vol47No2/2Vol47No2.pdf>

Continues And Discrete Variables (2015, November 12). Retrieved from [https://en.wikipedia.org/wiki/Continuous\\_and\\_discrete\\_variables](https://en.wikipedia.org/wiki/Continuous_and_discrete_variables)

Dawson-howe, K. (2014). A Practical Introduction To Computer Vision With OpenCV.

West Sussex, United Kingdom: Wiley.

Geman, S., & Geman, D. (1984). Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 6. Retrieved from

<https://cs.uwaterloo.ca/~mannr/cs886-w10/GemanandGeman84.pdf>

Gibbs sampling (2016, June 18). Retrieved from

[https://en.wikipedia.org/wiki/Gibbs\\_sampling](https://en.wikipedia.org/wiki/Gibbs_sampling)

Grath, M. M. (2003). Markov Random Field Image Modeling (Master Thesis).

Karaboga, D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Elsevire*, 214. Retrieved from

<http://natcomp.liacs.nl/SWI/papers/artificial.bee.colony.algorithm/a.comparative.study.of.abc.algorithm.pdf>

Kato, Z. & Pong, T. & Lee, J. C. (2001). Color image segmentation and parameter estimation in a markovian framework. *Pattern Recognition Letters*, 22. Retrieved from <http://www.inf.u-szeged.hu/~kato/papers/pattreclet2001.pdf>

Li, S. Z. (2009). Markov Random Field Modeling in Image Analysis (3rd ed.). London, United Kingdom: Springer.

Linear combination (2015, October 10). Retrieved from

[https://en.wikipedia.org/wiki/Linear\\_combination](https://en.wikipedia.org/wiki/Linear_combination)

Markov chain Monte Carlo (2016, May 29).

[https://en.wikipedia.org/wiki/Markov\\_chain\\_Monte\\_Carlo](https://en.wikipedia.org/wiki/Markov_chain_Monte_Carlo)

Metropolis-Hastings algorithm (2016, July 2). Retrieved from

[https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings\\_algorithm](https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm)

Multivariate Normal Distribution (2016, June 23). Retrieved from

[https://en.wikipedia.org/wiki/Multivariate\\_normal\\_distribution](https://en.wikipedia.org/wiki/Multivariate_normal_distribution)

Naive Bayes Classifier (2016, June 28). Retrieved from

[https://en.wikipedia.org/wiki/Naive\\_Bayes\\_classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)

Normal Distribution (2016, June 18). Retrieved from

[https://en.wikipedia.org/wiki/Normal\\_distribution](https://en.wikipedia.org/wiki/Normal_distribution)

Petrou, M., & Sevilla, P. G. (2006). Image Processing Dealing with Texture. West

Sussex, England: Wiley.

Pham, D.T., & Koc, Ghanbarzadeh, A., & Koc, E., & Otri, S., & Rahim, S., & Zaidi, M.

(2005). The Bees Algorithm - A Novel Tool for Complex Optimization

Problems. Manufacturing Engineering Center, Cardiff University. Retrieved

from [https://svn-d1.mpi-inf.mpg.de/AG1/MultiCoreLab/papers/Pham06%20-](https://svn-d1.mpi-inf.mpg.de/AG1/MultiCoreLab/papers/Pham06%20-%20The%20Bee%20Algorithm.pdf)

[%20The%20Bee%20Algorithm.pdf](https://svn-d1.mpi-inf.mpg.de/AG1/MultiCoreLab/papers/Pham06%20-%20The%20Bee%20Algorithm.pdf)

Posterior probability (2016, June 21). Retrieved from:

[https://en.wikipedia.org/wiki/Posterior\\_probability](https://en.wikipedia.org/wiki/Posterior_probability)



Prior probability (2016, May 14). Retrieved from

[https://en.wikipedia.org/wiki/Prior\\_probability](https://en.wikipedia.org/wiki/Prior_probability)

Rini, D. P., & Shamsuddin, S. M., & Yuhaziz, S. S. (2011). Particle Swarm

Optimization: Technique, System and Challenges. *International Journal of Computer Applications*, 14. Retrieved from

<http://www.ijcaonline.org/volume14/number1/pxc3872331.pdf>

Russell, S., & Norvig, P. (2009). Artificial Intelligence: A Modern Approach (3rd ed.).

New Jersey: Pearson.

Sag, Tahir, & Cunkas, Mehmet. (2015). Color Image Segmentation Based On

Multiobjective artificial Bee Colony Optimization. *Elsevier*, 34. Retrieved from

<http://isiarticles.com/bundles/Article/pre/pdf/46216.pdf>

Seeley, T.D., & Visscher, P.K., & Passino, K.M. (2006). Group Decision Making in

Honey Bee Swarms. *American Scientist*, 94. Retrieved from

<http://www2.ece.ohio-state.edu/~passino/PapersToPost/GrpDecMakHoneyBees-AmSci.pdf>

Selvi, V., & Umarani, R. (2010). Comparative Analysis Of Ant Colony And Particle

Swarm Optimization Techniques. *International Journal of Computer Applications*. Retrieved from

<http://www.ijcaonline.org/volume5/number4/pxc3871286.pdf>

Standard Deviation (2016, June 27). Retrieved from

[https://en.wikipedia.org/wiki/Standard\\_deviation](https://en.wikipedia.org/wiki/Standard_deviation)

Swarm behavior (2016, June 29). [https://en.wikipedia.org/wiki/Swarm\\_behaviour](https://en.wikipedia.org/wiki/Swarm_behaviour)

Swarm intelligence (2016, June 21). [https://en.wikipedia.org/wiki/Swarm\\_intelligence](https://en.wikipedia.org/wiki/Swarm_intelligence)

Sziranyi, T., & Zerubia, J., & Czuni, L., & Geldreich, D., & Kato, Z. (2000). Image Segmentation Using Markov Random Field Model in Fully Parallel Cellular Network Architectures. *Real-Time Image*, 6. Retrieved from <http://www.inf.u-szeged.hu/rgvc/papers/rti2000.pdf>

Variance (2016, June 19). Retrieved from <https://en.wikipedia.org/wiki/Variance>

Wedde, H. F., & Farooq, M., & Zhang, Y. (2004). BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired By Honey Bee Behavior. *Springer-Verlag Berlin Heidelberg*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.525.4960&rep=rep1&type=pdf>

Yuce, B., & Packianather, M.S., & Mastrocinque, E., & Pham, D.T., & Lambiase, A. (2013). Honey Bees Inspired Optimization Method: The Bees Algorithm. *Insects*, 4. Retrieved from <https://orca-mwe.cf.ac.uk/53653/1/Yuce%25202013.pdf>.